Universiteit Utrecht

Faculteit Bètawetenschappen

Dictaat discrete wiskunde

Bachelor Thesis

Joris van Gool

Wiskunde en Informatica

Supervisors:

Dr. Han Hoogeveen
Universiteit Utrecht

17 juni 2017

Inhoudsopgave

0.1 Introductie scriptie

Deze wiskunde scriptie is tevens een dictaat voor het vak discrete wiskunde gegeven aan de universiteit Utrecht. Het vak wordt niet alleen bij wiskunde maar ook bij informatica gegeven. Doordat er geen voorkennis van wiskunde aanwezig is bij informaticastudenten zal het niveau soms wat lager liggen. Het is echter wel een niveau drie vak gegeven binnen de wiskunde faculteit. Het laatste hoofdstuk dynamische stroomproblemen is het onderzoek hoofdstuk waarvoor ik zelf literatuuronderzoek gedaan heb.

Er is bewust voor gekozen om het dictaat in lijdende vorm te schrijven om te voorkomen dat het "populair" aan voelt. Alhoewel het voor een docent vaak als een pluspunt gezien wordt om dicht bij zijn studenten te staan is mijn verwachting dat een dictaat dat door een student geschreven is al snel zijn autoriteit en daarmee zijn didactische waarde verliest.

Dit dictaat is geschreven voor het vak discrete wiskunde en Han Hoogeveen. Deze heeft het volste recht dit dictaat aan te passen en in zijn vak te gebruiken.

0.2 Introductie vak

Ergens op het grensvlak van de wiskunde en informatica bevindt zich dit vak. Afhankelijk van persoonlijke voorkeur voor een van deze twee vakgebieden zal dit vak anders ervaren worden. Er is in dit dictaat aandacht besteed aan beide beelden en soms zullen er stukken in het dictaat staan die alleen interessant zijn voor mensen met een voorkennis in een van de twee vakgebieden.

Dit vak bestaat uit twee duidelijke delen. Het eerste deel beslaat hoofdstukken 1 tot 4 en het tweede deel hoofdstukken 5 tot 9. De ervaring leert dat het eerste deel over het algemeen makkelijker is voor wiskundigen en het tweede deel over het algemeen makkelijker is voor informatici. In het eerste deel houden we ons voornamelijk bezig met telproblemen van enige moeilijkheidsgraad. In het tweede deel houden we ons bezig met het analyseren of een probleem een oplossing heeft(kort) en met het optimaliseren van een oplossing voor een probleem.

Inleveropgave voor dit vak zijn lang maar ook een accurate representatie van wat de toets zal inhouden.

Hoofdstuk 1

Voorkennis en basisprincipes

1.1 Introductie

Om te beginnen moet er wat kennis komen om basis telproblemen op te lossen. Dit hoofdstuk zal de benodigde voorkennis behandelen. Te beginnen met de product- en somregel.

Definitie 1.1. De productregel: Gegeven twee sets mogelijkheden x en y waarbij iedere mogelijkheid in x met iedere mogelijkheid in y gecombineerd kan worden. Dan is de totale hoeveelheid mogelijkheden voor x en y gelijk aan de hoeveelheid mogelijkheden voor x keer de hoeveelheid mogelijkheden voor y.

Voorbeeldopgave 1.2. Een rode en een blauwe dobbelsteen worden gegooid, hoeveel mogelijkheden zijn er voor de uitkomst van de worp?(waar rood en blauw dus van elkaar te onderscheiden zijn)

Neem voor x het aantal mogelijke uitkomsten van de blauwe dobbelsteen en y idem voor de rode dobbelsteen. Uitgaande van dobbelstenen met zes zijde krijgen we dus $x = y = 6$ en dus $6 \cdot 6 = 36$ mogelijkheden.

Definitie 1.3. De somregel: Als er iets uit x of y moet worden gekozen onstaat er een optelling van het aantal mogelijkheden in x en het aantal mogelijkheden in y.

Voorbeeldopgave 1.4. Bij de plaatselijke snackbar kunnen er 12 verschillende pizza's en 6 verschillende frituurmaaltijden besteld worden. Hoeveel maaltijden kunnen er in het totaal besteld worden?

Nu zijn de pizza's x en de frituurmaaltijden y dus $x = 12$, $y = 6$ en $x + y = 12 + 6 = 18$

Met deze twee regels is er al een redelijke basis om telproblemen op te lossen.

Voorbeeldopgave 1.5. In een snoepwinkel kunnen tien soorten snoep worden gekocht voor een euro en vijf soorten voor vijftig cent. Hoeveel verschillende combinaties kunnen er gekocht worden voor twee euro waarbij de volgorde waarin het snoep gekocht wordt uitmaakt?

Bij dit probleem moeten de mogelijkheden worden afgesplitst. Als de vraag tegenvalt beantwoord dan eerst de vraag "Op hoeveel manieren kan twee euro worden opgesplitst in euro's en vijftig centen". Als dat gelukt is, kun je de opties voor elk van deze manieren uitrekenen met behulp van de productregel en het eindantwoord met behulp van de somregel.
Tot het goede antwoord komen met de som- en productregel gaat hier zo: er zijn 3 opties om 2 euro op te delen, namelijk een euro munt en twee keer vijftig cent, twee euro munten of vier keer vijftig cent. Deze opties kunnen berekend worden met de productregel door voor x de soorten snoep van vijftig cent en voor y de soorten snoep van een euro te nemen. Een euromunt en twee keer vijftig cent wordt dan yx^2, xyx of x^2y. Twee euro munten wordt y^2 en vier keer vijftig cent wordt x^4. Deze opties tellen we op met de somregel en ons eindantwoord wordt $3yx^2 + y^2 + x^4 = 3 \cdot 5 \cdot 10^2 + 10^2 + 5^4 = 2225$.
Zoals net aangetoond is kan een telprobleem opgesplitst worden in meerdere gevallen als deze gevallen geen overlap hebben en samen het hele probleem vormen.

Voorbeeldopgave 1.6. Bij het bestellen van pizza zijn er 5 mogelijkheden voor ingrediënten waarvan er twee gekozen moeten worden(er mag ook twee keer hetzelfde mogen gekozen: dubbele portie!) als er echter ananas besteld wordt dan wordt de kok boos en is het andere ingrediënt altijd ham. Verder maakt het uit welk ingrediënt eerst gekozen wordt omdat daar meer van op de pizza komt. Hoeveel opties zijn er voor je pizza?

Hier is het handig om de pizza met ananas te isoleren. Aangezien deze altijd met ham komt, is dat gewoon één optie die er uit gehaald kan worden en later met behulp van de somregel kan worden opgeteld. Dan is er dus de losse optie "ananas met ham" en alle andere opties: met de productregel $4 \cdot 4 = 16$ maakt een totaal van 17 met behulp van de somregel. Omdat de optie met ananas niet dubbel geteld wordt (de ananas wordt niet meegenomen in de tweede berekening) en wel alle mogelijkheden geteld zijn, zijn dit twee gevallen zonder overlap die samen het hele probleem vormen en dus een goede oplossing.

1.2 Permutaties en combinaties

1.2.1 Permutaties

Een permutatie π is een bewerking op een set objecten die deze van volgorde laat veranderen. Een voorbeeld van een permutatie is $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix}$ en laat zien waar elk element naar toe gaat. De onderste rij geeft per kolom aan waar ieder element uit de bovenste rij naar toe gaat.

Voorbeeldopgave 1.7. Stel π wordt toegepast op het woord "stoel"wat is dan de uitkomst?

Aangezien het element op de eerste plaats, de "s", naar de tweede plaats gaat enzovoort krijgen we dus dat het hele woord een plaats opschuift en de laatste letter naar de eerste plaats gaat en dan wordt het "lstoe".

Een interessante vraag wordt dan hoeveel permutaties er mogelijk zijn met n elementen. Aangezien het eerste element op n plaatsen gezet kan worden, daarna het tweede element nog op $(n-1)$ enzovoorts, zijn dit er $n(n-1)(n-2)(n-3)(n-4)...2\cdot1$. Een vermenigvuldinging die een geheel getal met alle positieve gehele getallen eronder vermenigvuldigt wordt de faculteit van n genoemd. De notatie hiervoor is:

Definitie 1.8. n-faculteit: $n! = n \cdot (n-1) \cdot (n-2) \cdot ... \cdot 3 \cdot 2 \cdot 1$

Voorbeeld 1.9. Met deze kennis kan er met wat vragen gewerkt gaan worden. Stel er moet bepaald worden hoeveel opties er zijn om een commissie van drie mensen te vullen met 3 functies, voorzitter secretaris en penningmeester. Er zijn twaalf mensen om uit te kiezen. Voor de eerste functie kan er uit 12 mensen worden gekozen, voor de tweede uit 11 en voor de derde uit 10. Dus $12\cdot11\cdot10 = 1320$ totale mogelijkheden.

We veralgemeniseren dit naar n mensen en m functies, dan wordt het antwoord $n \cdot (n-1) \cdot ... \cdot (n-m+1)$. Wat ook te schrijven is als $\frac{n!}{(n-m)!} = \frac{n\cdot(n-1)\cdot....\cdot(n-m)\cdot....\cdot2\cdot1}{(n-m)\cdot....\cdot2\cdot1} = n \cdot (n-1) \cdot ... \cdot (n-m+1)$.

1.2.2 Combinaties

Neem nu hetzelfde probleem, 12 mensen en 3 plaatsen, maar de commissieleden krijgen geen functie meer. Dat betekent dus dat als persoon 1, 2 en 3 gekozen worden deze altijd dezelfde commissie vormen ongeacht in welke volgorde deze gekozen worden. Bij het aantal dat er zonet gevonden is, zitten dus identieke samenstellingen. Maar hoe vaak wordt iedere oplossing geteld?
Zoals eerder uitgelegd is dit 3! want er zijn 3! manieren om 3 mensen in een volgorde te zetten. $3! = 6$ wat gecontroleerd kan worden door de volgordes uit te schrijven 123, 312, 231, 213, 132 en 321. Dit betekent dat het probleem kan worden uitgerekend via $\frac{12!}{(12-3)!} \cdot \frac{1}{3!} = \frac{12!}{(12-3)!3!}$. Hier is ook een wiskundige notatie voor namelijk $\binom{12}{3}$, spreek uit "twaalf boven drie", wat neerkomt op 3 elementen uit 12 kiezen waarbij volgorde niet uitmaakt. Dit wordt in de wiskunde het aantal combinaties genoemd.

Definitie 1.10. $\binom{a}{b} = \frac{a!}{(a-b)!b!}$

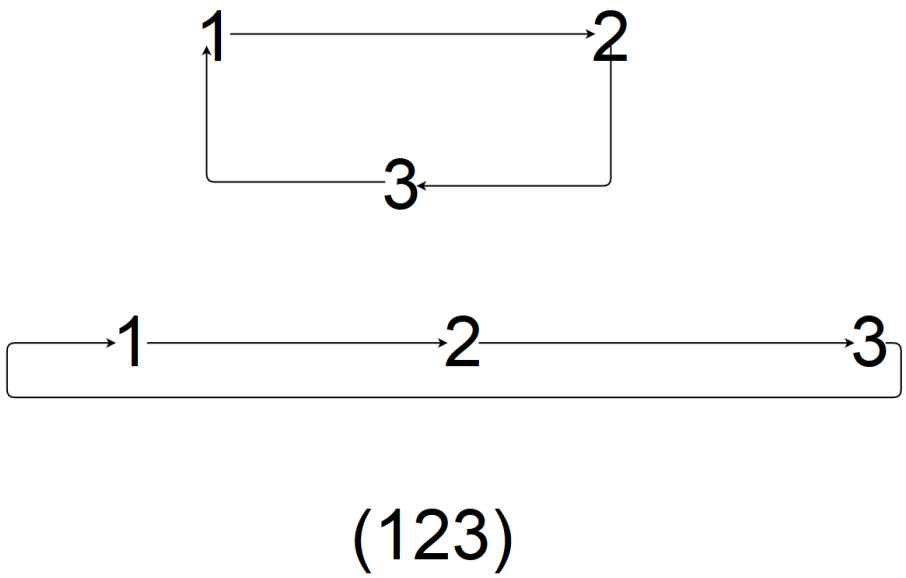
In een handig overzicht:

$n!$	Aantal permutaties van n elementen
$\frac{a!}{(a-b)!}$	Permutaties, kies b elementen uit a waarbij volgorde uit maakt
$\binom{a}{b}$	Combinaties, kies b elementen uit a waarbij volgorde niet uit maakt

Voorbeeldopgave 1.11. Een vriendengroep van 8 mensen wil een potje voetballen en heeft hiervoor twee teams van 4 nodig. Op hoeveel manieren kunnen zij deze teams maken?

Aangezien er maar een team gemaakt hoeft te worden omdat het andere team daaruit volgt, moeten er 4 van de 8 mensen gekozen worden zonder dat de volgorde uitmaakt wat $\binom{8}{4} = 70$ geeft.

Een handige alternatieve manier om permutaties op te schrijven die minder tijd kost is de cykel notatie. Bijvoorbeeld de π in het begin van dit hoofdstuk zou er uit komen te zien als (12345). Dit definieert dan een bewerking op een set van vijf elementen. Deze bewerking wordt gelezen door ieder stuk tussen haakjes te zien als een cykel. Als we bijvoorbeeld de letters "ABCDE"met deze cykel-permutatie zouden verschuiven, gebeurt dit door te beginnen bij het haakje openen. De 1 betekent dat de cykel begint bij de "A" en de 2 dat het tweede element van de cykel de "B" is. Aangezien het een cykel is die we een stap doorschuiven, betekent dit dat de "A" op de plek van "B" komt te staan. Om uit te vinden waar de "B" heen moet moet er gekeken worden naar wat er achter de 2 staat in de cykel notatie. Dit gaat dan zo door tot het einde. Het laatste element wordt naar de eerste plaats toe gebracht. Zo vormt het geheel een cykel die eentje doorschuift onder deze permutatie.



Figuur 1.1: Een visuele hulp voor hoe een permutatie naar cykelnotatie (123) gaat.

Voorbeeldopgave 1.12. Schrijf $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 1 & 4 & 5 \end{pmatrix}$ in cykelnotatie.

Nu zijn er drie cykels. De cykel die de eerste 3 elementen doorschuift en de twee cykels die de laatste twee op hun plaats houden. De cykelnotatie wordt dan (123)(4)(5).

De permutatie die toegepast werd op de letters heeft maar één cykel en die in het voorbeeld 3. Een permutatie met één cykel wordt ook wel een "tour" genoemd. Dit omdat bij het heel vaak toepassen van een tour alle getallen een keer op iedere plaats staan. De hoeveelheid mogelijke touren voor n elementen is op een zelfde wijze te berekenen als hoeveel permutaties er zijn. Het is aan te raden eerst zelf een poging te doen en daarna pas verder te lezen. (...) . Het eerste element wat verschoven

wordt, heeft $(n-1)$ plaatsen waar deze kan staan, voor een permutatie was dit n maar deze keer mag het element niet op zijn eigen plaats terugkomen want dan zou dit een eigen cykel vormen. Nu moet worden verder gegaan met het element dat op de plaats staat waar het eerste element heen ging. Dit element kan nog op $(n-2)$ plaatsen, niet zijn eigen plaats want daar staat element één al en niet op de eerste plaats want dan vormt dat een cykel van grootte twee. Zo verder redenerend wordt duidelijk dat er $(n-1)!$ mogelijke touren zijn.

Dan nog een kaartprobleem en tevens het eerste probleem waarbij kansen een rol zullen spelen. Rekenen met kansen zal vaker voorkomen in dit vak, maar zal altijd van dezelfde vorm zijn: mogelijkheden tellen en door elkaar delen, dit werkt alleen als iedere mogelijkheid dezelfde kans heeft wat tijdens dit vak altijd waar zal zijn. Bridge is een kaartspel dat iets weg heeft van klaverjassen maar een veel grotere complexiteit heeft en daarom ook als denksport aangemerkt wordt. Dit wordt gespeeld met vier spelers verdeeld over twee teams die ieder dertien willekeurige kaarten krijgen (daarmee is het deck dus ook op). Tijdens het spelen komt het volgende vraagstuk naar boven.

Voorbeeldopgave 1.13. Als een team samen 7 schoppens heeft wat is dan de kans dat de andere zes schoppens gelijk verdeeld zijn over de overige twee spelers? (3 bij iedere speler)

Omdat alleen naar de schoppens gekeken wordt moeten er dus 26 kaarten waarvan 6 schoppens over twee spelers verdeeld worden. Dit kan in het totaal op $\binom{26}{13}$ manieren. Als de schoppens gelijk verdeeld moeten zijn kan dit op $\binom{6}{3}\binom{20}{10}$ manieren. $\binom{6}{3}$ manieren om een speler 3 schoppens te geven en $\binom{20}{10}$ om hem nog 10 andere kaarten te geven die geen schoppens zijn. Omdat de kaarten willekeurig gedeeld zijn, is de kans op iedere toewijzing even groot en wordt de kans dus $\frac{\binom{6}{3}\binom{20}{10}}{\binom{26}{13}} \approx 0.36$. De kans is dus ongeveer 36%.

Voorbeeld 1.14. Een (vaak) bekende denkpuzzel onder wiskunde studenten is het gevangenen probleem. Hierbij zijn er $2n$ gevangenen die allemaal een nummer hebben. Deze gevangenen krijgen allemaal een kans om vrij te komen op de verjaardag van de cipier. Zij worden allemaal één voor één een ruimte ingestuurd met daarin $2n$ dozen. Hiervan mogen zij n dozen openmaken, in iedere doos zit een nummer en een hoedje. Het hoedje is verder niet belangrijk maar zit altijd in dit soort problemen. Als ze hun eigen nummer vinden mogen ze het hoedje houden en als iedereen zijn of haar eigen nummer vind zijn ze allemaal vrij. De vraag is nu, wat is de beste tactiek gegeven dat ze alleen voordat de eerste persoon naar binnen gaat met elkaar mogen overleggen. Het zou handig zijn als bij dit probleem de permutatie van de nummers bekend zou zijn. Dan zou iedere gevangenen gewoon naar zijn eigen doos toe kunnen lopen en zijn hoedje eruit kunnen halen. Dit is echter niet zo maar met deze permutatie kan wel een leuk trucje uitgehaald worden. Als de eerste persoon eerst in doos 1 kijkt, daar een nummer a vind en dan in doos a gaat kijken, en zo door. Vind iedereen zijn eigen nummer als er in de permutatie geen cykel van langer dan n zit. Immers met een cykel korter dan n vind persoon 1 na minder dan n stappen de doos met zijn nummer die de cykel afmaakt waar hij bij zijn eigen doos

aan begonnen is. De kans dat het maximale aantal stappen in een cykel gelijk is aan k met k groter dan n (er kan zoizo maar 1 van deze cykels bestaan omdat deze meer dan de helft van de punten beslaat) kan berekent worden met

$$\frac{\binom{2n}{k}(k-1)!(2n-k)!}{(2n)!} = \frac{1}{k}$$

Het aantal mogelijke permutaties met precies k (waar bij $k > n$) punten in de grootste subcykel kan namelijk als volgt bepaald worden: de eerst k getallen uit$\{1, ..., 2n\}$ worden eerst gekozen, waarna die k getallen zo gepermuteerd moeten worden dat die permutatie niet uiteenvalt in kleinere subcykels (dit kan op $(k-1)!$ manieren, omdat er voor gezorgd moet worden dat die k punten één subcykel moeten gaan vormen) en waarna de overige $(2n-k)$ punten willekeurig gepermuteerd kunnen worden. De succeskans wordt dan gelijk aan

$$1 - \sum_{k=n+1}^{2n} \frac{1}{k} = 1 - \sum k = 1^{2n}\frac{1}{k} + \sum_{k=1}^{n} \frac{1}{k}$$

wat ongeveer gelijk is aan 0.307 wat een stuk betere kans is dan iedere persoon n willekeurige doosjes open laten maken.

1.3 Trekken met en zonder terugleggen, multinomiaal

Tot nu toe zijn er problemen langsgekomen waarbij er objecten geselecteerd zijn met of zonder terugleggen. Bij de snoepwinkel waar er genoeg van al het snoep aanwezig was was er spraken van trekken met terugleggen. Nadat een object gekozen was kon het opnieuw gekozen worden. Als dit niet het geval is is er spraken van trekken zonder terugleggen dan kan een object slechts een maal gekozen worden zoals bij het kiezen van de commissie. Een ander voorbeeld van trekken met terugleggen zijn pincodes.

Voorbeeld 1.15. Een pincode bestaat uit 4 getallen geselecteerd uit 0...9, hiervoor zijn dus 10^4 opties. Dit is trekken met terugleggen, als het eerste getal een negen is kan het tweede getal nogsteeds een negen zijn. Een pincode die niet twee keer hetzelfde getal bevat zou trekken zonder terugleggen zijn de formule wordt dan $10 \cdot 9 \cdot 8 \cdot 7$.

Bij dit soort problemen is er spraken van een multinomiaal verdeling. Bij een binomiaal verdeling, zoals het opgooien van een munt, is iedere actie een succes of niet. Bij multinomiaal heeft iedere beslissing meerdere mogelijkheden zoals in het voorbeeld hiervoor de getallen 0...9. De multinomiaal verdeling beschrijft de kans op een mogelijke set uitkomsten van een experiment met meer dan twee uitkomst mogelijkheden.

Voorbeeld 1.16. In het vorige voorbeeld werd gebruikt gemaakt van pincodes. De kans dat een pincode bijvoorbeeld 2 negens bevat is multinomiaal verdeeld en zou uitgerekent kunnen worden door $\frac{1}{10}^2\frac{9}{10}^2$ te vermenigvuldigen met het aantal mogelijke volgordes $\binom{4}{2}$ voor een totaal van $\frac{1}{10}^2\frac{9}{10}^2\binom{4}{2}$

1.4 Herkenbaarheid

Zoals hierboven al een keer langskwam, verandert een probleem drastisch zodra een element geïdentificeerd kan worden. Toen de posities binnen de commissie weggehaald werden in voorbeeld 1.9. werd het een heel ander probleem. Dit soort problemen vallen in de categorie verdelingsproblemen die in acht sub-categorieën onder te verdelen zijn. Hierbij wordt vaak gekeken naar n objecten en m dozen waar deze objecten over verdeeld worden. Dit kan met herkenbare objecten en herkenbare dozen, onherkenbare objecten en herkenbare dozen, herkenbare objecten en onherkenbare dozen of onherkenbare objecten en onherkenbare dozen. En dan wordt er nog onderscheid gemaakt tussen of de dozen leeg mogen zijn of niet.

Als beide de dozen en objecten niet herkenbaar zijn wordt er gesproken van een partitie probleem: hoeveel verschillende sets van m cijfers of minder kunnen er gevonden worden die opgeteld n zijn. Bij $m = 3$ en $n = 7$ bijvoorbeeld zijn er de volgende 8 mogelijkheden:

$$\{1,1,5\},\ \{1,2,4\},\ \{1,3,3\},\ \{2,2,3\},\ \{0,1,6\},\ \{0,2,5\},\ \{0,3,4\},\ \{0,0,7\}$$

Merk op dat $\{0,1,6\}$ en $\{0,6,1\}$ als dezelfde mogelijkheid gezien worden. Bij de variatie waarin er geen dozen leeg mogen zijn blijven er vier opties over namelijk $\{1,1,5\}, \{1,2,4\}, \{1,3,3\}, \{2,2,3\}$

Als de dozen nu herkenbaar zijn en de ballen niet, kan dit probleem aangepakt worden als een rij chocolaatjes die in dozen verdeeld moeten worden. Eerst wordt iedere doos een bal gegeven, aangezien dozen niet leeg mogen zijn, als dit niet kan is er ook geen oplossing voor het probleem. Daarna wordt er gekeken naar op hoeveel manieren er splitsingen in de rij met chocolaatjes gemaakt kunnen worden. In een visueel voorbeeld met $n = 7$ en $m = 3$:

$$\cdot\cdot\,|\,\cdot\cdot|\,\cdot\,|\,\cdot\cdot$$

Zo heeft de eerste doos 2 chocolaatjes/ballen, de tweede doos ook, de derde doos één en de laatste doos twee. Hiermee is iedere mogelijkheid gedekt en geen mogelijkheid wordt dubbel geteld. Dit geeft $\binom{n+m-1}{m-1}$ als aantal mogelijke verdelingen met $m-1$ voor het aantal te zetten streepjes en $n+m-1$ voor het aantal plaatsen waar deze kunnen staan. Dit is ook direct het geval waarin dozen leeg mogen zijn. Samengevat wordt deze methode gebruikt, en als de dozen niet leeg mogen zijn wordt eerst in iedere doos een chocolaatje gestopt voordat de streepjes gezet worden.

m onherkenbare dozen en n herkenbare ballen hebben een moeilijkere relatie die in het volgende hoofdstuk meer uitgediept zal worden. Voor nu wordt de volgende formule aangenomen als de correcte manier om uit te rekenen op hoeveel manieren n herkenbare ballen over m onherkenbare dozen verdeeld kunnen worden zonder lege dozen.

$$S(n,m) = \frac{1}{m!}\sum_{j=0}^{m}(-1)^j\binom{k}{j}(k-j)^n \tag{1.1}$$

Als de dozen wel leeg mogen zijn, wordt de formule $\sum_{j=0}^{m} S(n,j)$. Op deze manier wordt er namelijk gekeken naar alle mogelijkheden voor ieder aantal dozen kleiner

dan of gelijk aan m.

Dan zijn er nog herkenbare ballen en herkenbare dozen. Deze situatie kan bereikt worden door bij herkenbare ballen en onherkenbare dozen labels op de dozen te plakken om ze herkenbaar te maken. Dit kan altijd op $m!$ manieren dus de totale formule wordt dan:

$$S(n,m)m! = \sum_{i=0}^{m}(-1)^i\binom{k}{i}(k-i)^n \tag{1.2}$$

Als ze leeg mogen zijn is het m^n want iedere bal kan in iedere doos wat allemaal unieke opties geeft.

In een handig overzichtje:

Ballen	Dozen	Leeg	Formule
Onherkenbaar	Onherkenbaar	Nee	Zie hoofdstuk 2
Onherkenbaar	Onherkenbaar	Ja	Zie hoofdstuk 2
Onherkenbaar	Herkenbaar	Nee	$\binom{n-1}{m-1}$
Onherkenbaar	Herkenbaar	Ja	$\binom{n+m-1}{m-1}$
Herkenbaar	Onherkenbaar	Nee	$S(n,m) = \frac{1}{m!}\sum_{j=0}^{m}(-1)^j\binom{k}{j}(k-j)^n$
Herkenbaar	Onherkenbaar	Ja	$\sum_{j=0}^{m} S(n,j)$
Herkenbaar	herkenbaar	Nee	$S(n,m)m! = \sum_{i=0}^{m}(-1)^i\binom{k}{i}(k-i)^n$
Herkenbaar	herkenbaar	Ja	m^n

1.5 Binomiaalexpansie

Tijdens dit vak zullen er vaak coëfficiënten uitgerekend moeten worden. Bij het uitzoeken wat de coëfficiënt is van $x^{11}y^9$ in $(x+y)^{20}$ is het volledig uitschrijven geen goede methode. Hiervoor is er de binomiaalexpansie. Dit is een handige formule om snel een bepaalde coëfficiënt uit te rekenen van bijvoorbeeld $(x+y)^{20}$.

$$(x+y)^n = \sum_{j=0}^{n}\binom{n}{j}x^j y^{n-j} \tag{1.3}$$

Uitleg:
Als $(x+y)^n$ uitgewerkt moet worden worden alle mogelijke combinaties met elkaar vermenigvuldigd worden (net als bij een tweemacht maar nu met veel meer combinaties). Dus eigenlijk zijn er 2^n unieke rijtjes van de vorm $xxxyxyyyxyxxxyyx$...(één

voor iedere mogelijkheid). De vraag is nu: hoeveel rijtjes worden er verkregen met exact p keer een x en dan dus $n - p$ keer een y? Aangezien ieder mogelijk rijtje voorkomt, kan dit gevonden worden door te kijken naar het aantal manieren om p elementen uit n te kiezen waarbij volgorde niet uitmaakt. Dit brengt ons naar de hierboven gegeven formule die voor iedere mogelijkheid van x en y op de genoemde manier berekent wat de factor is van die coëfficiënt. Het totaal van alle termen in een binomiaal expansie is altijd 2^n. Dit is makkelijk na te rekenen door $x = y = 1$ in te vullen. $(1 + 1)^n = 2^n$
Een korte voorbeeldberekening:

Voorbeeldopgave 1.17. De coëfficiënt van x^5y^7 in $(x + y)^{12}$ met behulp van de binomiaal expansie. Hier worden niet alle coëfficiënten uitgerekend maar alleen die van x^5y^7 door $j = 5$ in te vullen. Dan wordt er gevonden dat de coëfficiënt $\binom{12}{5} = 792$ is.

1.6 Pigeonhole principle

Het pigeonhole principle komt van duiven (pigeons) en hokjes waarin duiven zitten (pigeonholes). Als er meer duiven dan duivenhokjes zijn dan zitten er altijd ergens op zijn minst twee in een hokje. Dit klinkt waarschijnlijk heel makkelijk maar laat de aandacht nog niet verslappen, want de vraag van deze categorie op het deeltentamen wordt altijd erg slecht gemaakt. Een basisvoorbeeld.

Voorbeeldopgave 1.18. Bij bridge wordt iedere speler 13 kaarten van een normaal kaartspel toebedeeld. Kan een speler alleen maar sets van 3 van dezelfde kleur(harten/schoppen/klaver/ruiten) of minder hebben?

Met vier mogelijke kleuren en drie kaarten van iedere kleur zijn er slechts 12 kaarten in totaal. De laatste kaart zal dus altijd zorgen voor een set van vier van dezelfde kleur. Dan is het dus onmogelijk dat er alleen sets van drie of minder in een hand zitten. Dan een iets moeilijker voorbeeld:

Voorbeeldopgave 1.19. Mark en negentien vrienden gaan iedere zondag bier drinken en drinken ieder tussen de tien en twintig glazen bier. Daarna volgt er een ronde bridge (waarin vier mensen spelen in paren van 2) waarin beide paren evenveel glazen bier gedronken hebben. Kan het zijn dat er na een avond bier drinken nog iemand een glaasje bij moet drinken om twee teams van deze vorm te kunnen maken?

Denk er even over na, het is niet makkelijk maar met het pigeonhole principle wel goed te doen! Om te beginnen zijn er 11 getallen tussen de 10 en de 20 en kunnen het aantal gedronken glazen per paar tussen de 20 en de 40 in liggen. Dit zijn 21 mogelijkheden. Met 20 spelers zijn er in het totaal $\binom{20}{2} = 190$ mogelijke paren. Er kan dus gegarandeerd voor één van de mogelijkheden voor het aantal gedronken glazen bier minstens 9 paren gemaakt worden, immers $190/21 \approx 9$. 2 van deze 9 paren zijn genoeg om een wedstrijd te doen zolang dezelfde persoon niet in beide paren zit. Als dezelfde persoon echter in alle paren zit, zodat er geen twee paren gemaakt kunnen worden waarbij hij maar in een van de twee paren zit, hebben de andere 8 personen hetzelfde gedronken en kunnen zij dus makkelijk twee paren vormen. Zo zijn er dus altijd genoeg paren om een wedstrijd te doen.

1.7 Combinatorisch bewijzen

Bij een combinatorisch bewijs willen we iets bewijzen door het gelijk te stellen aan een telprobleem. Een gelijkheid kan bijvoorbeeld bewezen worden door te laten zien dat beide kanten de uitkomst zijn van hetzelfde telprobleem. Een voorbeeld:

Voorbeeldopgave 1.20. Gebruik een combinatorisch bewijs om te bewijzen dat $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$

Er zijn n ballen waarvan er r gekozen worden, dit kan op $\binom{n}{r}$ manieren. Een van deze ballen krijgt een kruisje, er zijn $\binom{n-1}{r}$ manieren om r ballen te kiezen zonder die met het kruisjes. Daarnaast zijn er als de bal met het kruisje gekozen is nog $\binom{n-1}{r-1}$ manieren om de rest van de ballen te kiezen. Aangezien er 1 manier is om de bal met het kruisje te kiezen zijn er dus in het totaal $\binom{n-1}{r} + \binom{n-1}{r-1}$ manieren om r ballen te kiezen. Nu is bewezen dat $\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$

Nog een voorbeeld ter verduidelijking:

Voorbeeldopgave 1.21. Bewijs $\binom{n+r+1}{r} = \binom{n}{n} + \binom{n+1}{n} + ... + \binom{n+r}{n}$ met een combinatorisch bewijs

Hier is $\binom{n+r+1}{r}$ het aantal manieren om r witte en $n+1$ zwarte ballen op een rij te zetten. De andere kant van de vergelijking is een uitsplitsing voor de positie van de eerste zwarte bal. Als de eerste zwarte bal op de eerste plaats komt te liggen zijn er nog $\binom{n+r}{n}$ manieren om de andere te verdelen. Bij de tweede plaats zijn dit er $\binom{n+r-1}{n}$ enzovoorts tot de eerste zwarte bal op de $r+1$de plaats komt te liggen en er nog maar $\binom{n}{n} = 1$ manier is om de andere zwarte ballen te plaatsen. Dit bewijst de gelijkheid.

Bij combinatorische bewijzen is het net als bij normale bewijzen zo dat meer oefening meer vaardigheid geeft. Ook helpt het altijd om de vraag te stellen "Hoe kan ik dit in een bekend verschijnsel omschrijven" voor beide kanten van een vergelijking.

1.8 Opgaven

Opgave 1. "Gegeven zijn twee gesorteerde rijen$a_1, ..., a_k$ en $b_1, ..., b_n$. Deze willen we samenvoegen tot één rij, waarin de elementen $a_2, ..., a_k$ en $b_1, ..., b_n$ in de juiste volgorde blijven staan (waarbij de elementen uit de andere rij er tussen door komen); element a_1 mag overal komen te staan, zolang het maar niet links van a_2 staat. Hoeveel verschillende rijen kunnen op deze manier worden gevormd? Deze formule mag geen sommatie bevatten."

Opgave 2.
a. Bepaal het aantal getallen tussen 1000 en 9999 die bestaan uit vier verschillende cijfers die alle ongelijk 0 zijn.
b. Voeg de extra eis toe dat de getallen deelbaar moeten zijn door 3. Hoeveel getallen zijn er dan?
c. Zelfde vraag als bij (b), maar nu is het toegestaan dat er een 0 voorkomt.

Opgave 3. Iedere student op de UU heeft een student nummer van zeven cijfers. Bij het vak discrete wiskunde zatten in 2017 84 mensen. Kan het zijn dat er geen groep van 9 mensen is met hetzelfde eindcijfer bij hun studentnummer?

Opgave 4. Hoeveel getallen van 4 cijfers zijn er die ofwel allemaal even ofwel allemaal oneven zijn?

Opgave 5. Een voetbal elftal bevat 11 mensen waarvan 1 keeper, hoeveel verschillende elftallen kunnen er gemaakt worden uit een selectie van 21 mensen met daarin 3 keepers? (waarbij de keepers natuurlijk alleen op de keeper positie mogen spelen)

Opgave 6. Bewijs de gelijkheid met een combinatorisch en een algebraïsch bewijs:

$$\binom{n}{m}\binom{m}{k} = \binom{n-k}{m-k}$$

Hint: gebruik de binomiaalexpansie voor $x = 1$ en $y = -1$.

Opgave 7. Op hoeveel manieren kunnen 20 unieke pokemon kaarten verdeeld worden over 15 kinderen?

Opgave 8. Het Bell number B_n is het aantal manieren om een set van n elementen in niet lege niet onderscheidbare dozen te stoppen. Merk op dat $B_n = S(n,0) + S(n,1) + ... + S(n,n)$. Toon nu aan dat

$$B_n = \binom{n-1}{0}B_0 + \binom{n-1}{1}B_1 + ... + \binom{n-1}{n-1}B_n$$

Opgave 9. Een DNA string wordt geschreven als een string van de volgende 4 letters G, C, A en T. Op hoeveel manieren kan een stukje DNA van 8 lang gemaakt worden waarbij 4 T's, 2 A's en 2 C's gebruikt worden?

Opgave 10. Er wordt 10 keer een muntje op gegooid, hoe groot is de kans dat, gegeven dat er exact 6 keer kop gegooid wordt de gegooide resultaten te omschrijven zijn als $kkkkkkmmmm$ met k voor kop en m voor munt.

Opgave 11. Op hoeveel manieren kan een kleuterklas van 25 kleuters met daarin 5 jongens op een rijtje gezet worden als geen twee jongens naast elkaar mogen staan?

Opgave 12. Vind de coëfficiënt van $a^3b^2c^2$ in $(a+b+c)^7$.

Opgave 13. Vind het aantal manieren om n keer met een munt te gooien waarbij er een even aantal keer kop gegooid wordt.

Opgave 14. Vind $\sum_{k=0}^{n} 2^k\binom{n}{k}$ en vind $\sum_{k=0}^{n} k(k-1)\binom{n}{k}$ met een combinatorisch bewijs.

Opgave 15. Hoeveel studenten moeten er op een feestje zijn om te zorgen dat twee dezelfde verjaardag hebben? Dezelfde 4 laatste cijfers van hun student nummer? Of dezelfde letter als eerste in hun achternaam?

Opgave 16. Gegeven is een mp3-speler waarop in totaal n liedjes staan, die daar op zijn gezet door verschillende mensen. Smaken verschillen, en ieder liedje is door één van hen gekwalificeerd als 'een bak herrie' (hier zijn er k stuks van) en 'een lust voor het oor' (de overige $n-k$ stuks). De mp3-speler speelt de liedjes in willekeurige volgorde af. Bereken de kans dat de mp3-speler de liedjes afspeelt in een volgorde waarin nooit twee prutliedjes (de bak herrie dus) achter elkaar worden gedraaid. Maakt het hierbij uit of de liedjes onherkenbaar zijn (afgezien van de kwalificatie prut of prachtig)?

Opgave 17. Een citaat uit een bericht naar de Wiskunde hulplijn:
... In de wijk waarin ik woon zetten de bewoners iedere eerste maandagavond in de maand hun zakken met gebruikt plastic buiten, waarna ze dinsdag worden opgehaald. Bij ieder van de 100 *huizen liggen er altijd* 1, 2, 3, 4, *of* 5 *zakken buiten; in totaal betreft het altijd tussen de* 250 *en* 260 *zakken. Ik heb er samen met een vriend (laten we hem Piet noemen) een sport van gemaakt om te kijken of we een serie opeenvolgende adressen kunnen vinden waar precies* 10 *of* 20 *zakken liggen (we hebben een vaste route die we lopen naar school toe). Tot nu toe is dat altijd gelukt. Nu heb ik met Piet gewed dat dat toeval is, en dat het geen* 10 *keer achter zou lukken. De afgelopen* 9 *keer is het echter wel gelukt. Daarom wil ik jullie vragen om een voorbeeld te maken waarbij het niet lukt. Het gaat dus om* 100 *adressen en ongeveer* 250 *zakken in totaal (zeg maar* $240-250$*, maar een stuk of twintig meer of minder is ook goed; die resterende zakken haal ik wel ergens anders vandaan of leg ik wel ergens anders neer); het mag niet mogelijk zijn om een serie van opeenvolgende adressen te vinden waar precies* 10 *of* 20 *zakken liggen. Wanneer jullie mij die oplossing mailen, dan ga ik dinsdagochtend heel vroeg de zakken buiten wel overeenkomstig met jullie voorbeeld klaarleggen. ....*
Ook al ben je het misschien niet eens met de praktijken van deze persoon, construeer het gevraagde voorbeeld, of bewijs dat dat niet bestaat

1.9 Bronverantwoording

Dit hoofdstuk is voor een groot deel gebaseerd op Applied combinatorics second edition door Fred S. Roberts en Barry Tesman uitgegeven door CRC press. De volgorde waarin onderwerpen behandeld worden komt eldover het algemeen overeen maar de uiteindelijke tekst, voorbeelden, bewijzen en voorbeeldopgaven zijn origineel met uitzondering van het stuk over herkenbaarheid. Ook is inspiratie opgedaan uit de college aantekeningen van Han Hoogeveen.

Opgaves in dit hoofdstuk zijn een selectie van de door Han Hoogeveen opgegeven opgaves(1,2,16,17), enkele unieke zeer leerzame opgaves geselecteerd uit Applied combinatorics(6,8) en enkele orginele opgaves.

Hoofdstuk 2

Genererende functies

2.1 Introductie

In dit hoofdstuk zal er gekeken worden naar een deel van de theorie achter genererende functies en hoe deze toegepast kunnen worden. Na een theoretische inleiding bevat de tweede paragraaf van dit hoofdstuk voorbeelden over hoe deze theorie gebruikt kan worden.

Generende functies kunnen worden gezien als een codering van data. Er is een (oneindige) reeks $A = (a_0, a_1, a_2, a_3, ...)$ die omgezet wordt in een polynoom$f(x) = a_1 + a_2x + a_3x^3 + ...$ Deze polynoom wordt vervolgens weer herschreven als een veel kortere functie die dezelfde uitkomsten heeft als de polynoom als er maar dicht genoeg bij 0 gekeken wordt. Zo wordt een vaak oneindige polynoom opgeslagen als een vaak korte functie. Door deze functie weer om te zetten naar de polynoom die erbij hoort en dan die polynoom weer om te zetten naar een reeks wordt de initiële informatiestroom teruggevonden.

De eerste oneindige polynoom is $f(x) = 1 + x + x^2 + x^3 + ...$ met als bijbehorende reeks $(1, 1, 1, 1, ...)$. Als $|x| < 1$ dan convergeert de limiet van deze oneindige polynoom naar $f(x) = \frac{1}{1-x}$. Dan wordt 1 de convergentiestraal van deze genererende functie genoemd, deze bestaat voor alle genererende functies aangezien alle generende functies machtsexpansies zijn. Mocht het woord limiet en de betekenis van deze laatste zin uit de lucht komen vallen, dan geeft dat niet het is voor algemeen begrip fijn maar is niet binnen de stof van dit vak.

Voor mensen met kennis van Maclaurin- en Taylorreeksontwikkeling is het interessant om te weten dat het uitbreiden van $\frac{1}{1-x}$ gedaan kan worden door de Taylorreeks uit te werken. Dit doen verhoogt het algemene begrip van dit vak (als er reeds kennis aanwezig is van hoe Taylorreeksen werken).

Met deze reeks kan een aantal andere reeksen al gemaakt worden. Dingen als $(2, 2, 2, 2, ...)$ en $(0, 0, 1, 1, 1, 1, ...)$ zijn hier direct uit te halen. Denk er even over na, het i'de getal tussen haakjes correspondeert met de factor voor x^i(beginnend bij $i = 0$). $f(x) = \frac{1}{1-x} = 1+x+x^2+x^3+...$ hoe wordt $2+2x+2x^2+...$ dan verkregen? Het antwoord wordt dan $2 + 2x + 2x^2 + ... = 2(1 + x + x^2 + x^3 + ...) = 2\frac{1}{1-x} = \frac{2}{1-x}$. Voor de tweede is er weer een vermenigvuldiging met de al bekende reeks. Vermenigvuldigen met een constante gaat duidelijk niet werken aangezien dat alle getal-

len verandert. Vermenigvuldiging met x^2 geeft ons het gewenste resultaat, immers $x^2(1 + x + x^2 + x^3 + ...) = x^2 + x^3 + x^4 + ...$ Als laatste voorbeeld is er nog $(1, 1, 1, 1, 0, 1, 1, ...)$ deze wordt gemaakt door simpelweg de 4de macht af te trekken $\frac{1}{1-x} - x^4$.

De volgende twee veel gebruikte machtsreeksen zijn $g(x) = \frac{1-x^{n+1}}{1-x} = 1 + x + x^2 + ... + x^n$ en $h(x) = e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + ... + \frac{x^i}{i!} + ...$ Deze werken op exact dezelfde manier als de vorige en er kan op eenzelfde wijze mee gerekend worden.

Zoals er hierboven simpele wiskundige operaties gebruikt zijn om een reeks aan te passen kunnen deze ook gecombineerd worden om nieuwe te maken. Optellen wordt paarsgewijs gedaan, voor een reeks $A(x) = (a_0, a_1, a_2, ...)$ en $B(x) = (b_0, b_1, b_2, ...)$ wordt $A(x) + B(x) = C(x) = (a_0 + b_0, a_1 + b_1, ...)$.

Reeksen kunnen ook vermenigvuldigd worden op een iets complexere manier. Als $(1, 1, 1)$ met zichzelf vermenigvuldigd wordt dan wordt dit $(1, 2, 3, 2, 1)$ omdat $(1 + x + x^2)(1 + x + x^2) = 1 + 2x + 3x^2 + 2x^3 + x^4$ in dezelfde notatie als bij het optellen: $A(x)B(x) = C(x) = (a_0b_0, a_0b_1 + a_1b_0, a_2b_0 + a_1b_1 + a_0b_2, a_3b_0 + a_2b_1 + a_1b_2 + a_0b_3, a_2b_2, ...)$ en dus $c_k = a_kb_0 + a_{k-1}b_1 + ... + a_0b_k$.

Genererende functies kunnen ook geïntegreerd of geprimitiveerd worden. Gelukkig zijn het allemaal simpele polynomen dus valt dit mee. Een voorbeeld waar ook direct een waardevolle functie inzit:

Voorbeeld 2.1. Er wordt gezocht naar de genererende functie voor de reeks $(1, 2, 3, 4, 5, 6, 7, 8, 9, ...)$. De reeks $1 + x + x^2 + ...$ wordt gedifferentieerd voor het gewenste resultaat. Dat maakt de genererende functie voor deze reeks dus de afgeleide van $\frac{1}{1-x}$ wat $\frac{1}{(1-x)^2}$ is.

2.2 Werken met genererende functies

Nu er naar een paar genererende functies gekeken is en er wat ervaring opgedaan is met het zelf maken van genererende functies kan er gekeken worden naar een toepassing.

Theorema 2.2. *"Stel dat er p typen voorwerpen zijn; van voorwerp i hebben we n_i identieke kopieën $(i = 1, ..., p)$. Het aantal mogelijke manieren om k voorwerpen te kiezen is gelijk aan de coëfficiënt van x^k in $G(x) = (1 + x + ... + x^{n_1})...(1 + x + ... + x^{n_p})$."[1]*

Bewijs. Deze stelling zal bewezen worden met behulp van inductie. Het is waar voor $p = 1$ dit geeft $G(x) = 1 + ... + x^{n_1}$ wat betekent dat er 1 manier is om iedere set van k te kiezen namelijk allemaal voorwerp 1. De inductiehypothese wordt nu dat het geldt voor $p = p_0$. Er moet nu bewezen worden dat het ook geldt voor $p_0 + 1$. Er wordt gedefinieerd dat $f(p_0, h)$ het aantal mogelijke manieren is om h voorwerpen uit p_0 verschillende te kiezen met n_i identieke voorwerpen van soort i. Er geldt dat $f(p_0, h) = \sum_{g=0}^{n_{p_0}+1} f(p_0, h - g)$, waarbij $f(p_0, h - g) = 0$ als $h - g < 0$, hierbij wordt

het aantal van type $p_0 + 1$ afgesplitst in de som. Nu geldt dat

$$(1+...+x^{n_1})...(1+x+...+x^{n_{p_0}})(1+x+...+x^{n_{p_0+1}}) = (\sum_{h=0}^{\infty} f(p_0,h)x^h)(1+x+...+x^{n_{p_0+1}})$$

Dit laaste kan omgeschreven worden in $\sum_{h=0}^{\infty} f(p_0+1,h)x^h$ aangezien hier een vermenigvuldiging staat van de vorige som en hoeveel er daarna nog opgevuld wordt met objecten van soort $p_0 + 1$. Dit bewijst de stelling. □

Met deze stelling kunnen enkele telproblemen makkelijker opgelost worden. Een voorbeeld:

Voorbeeldopgave 2.3. Eva wil Naomi een bos van 15 rozen geven, samengesteld uit rode en witte rozen. Hoeveel verschillende bossen rozen kan Eva aan Naomi geven?

Deze vraag is zonder genererende functies relatief simpel, maar kan ook opgelost worden mét genererende functies. Op dezelfde manier als deze vraag ook moeilijkere vragen. De genererende functie $\frac{1}{1-x}$ kan genomen worden als alle opties met rozen van een kleur. Waarbij 1 geen rozen van die kleur, x een roos van die kleur en x^i i rozen van die kleur. De coëfficiënt van de term x^i is nu het aantal manieren om x^i rozen te kiezen, met een kleur is dit 1. Om dit voor twee kleuren te berekenen wordt er vermenigvuldigd, $\frac{1}{1-x}$ voor het aantal rode rozen, met $\frac{1}{1-x}$ voor het aantal witte rozen. Dit geeft $\frac{1}{(1-x)^2} = 1 + 2x + 3x^2 + 4x^3 + ...$ Nu is er iets interessants gebeurd, bij het uitvermenigvuldigen wordt iedere x macht(inclusief 0) aan de linker kant met iedere x macht aan de andere kant vermenigvuldigd. Aangezien ze aan de ene kant voor het aantal rode en aan de andere kant voor het aantal witte rozen stonden, staat voor x^i nu hoeveel manieren er zijn om een bos van i rozen te maken. Nu staat dus bij x^{15} het aantal verschillende bossen dat Eva aan Naomi kan geven!

Tot zover het makkelijke voorbeeld. Het wordt iets moeilijker als er wat vereisten toegevoegd worden:

Voorbeeldopgave 2.4. Naomi is blij met de rozen en wil een bos van 15 witte blauwe en paarse rozen terug geven. Maar als meer dan de helft van de rozen paars zijn wordt het boeket wel erg donker en dat wil Naomi niet. Ook moeten er meer blauwe dan witte rozen in zitten want witte rozen doen Naomi denken aan haar gefaalde huwelijk.

Deze iets moeilijkere opgave is nogsteeds goed te doen. De vereiste zijn $x+y+z = 15$, $x \leq 7$ en $y \leq z + 1$. De reeks voor de paarse rozen kan gemaakt worden door $\frac{1}{1-x}$ af te kappen bij 7, zodat $1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7$ verkregen wordt. Hoe kan het probleem met de andere rozen dan opgelost worden? Door de blauwe en witte rozen per paar te trekken(!), dus met de reeks $1 + x^2 + x^4 + ...$ Deze zorgt er voor dat voor iedere witte roos die gekozen wordt er ook meteen een blauwe bijkomt. Er zijn in ieder geval evenveel witte als blauwe rozen! Nu moeten er nog extra blauwe rozen gepakt worden, zodat er hiervan meer zijn dan witte. Dit wordt gedaan door een genererende functie te starten bij x waardoor er altijd op zijn minst een roos gepakt wordt en mogelijk meer. De laatste van de

drie genererende functies die vermenigvuldigd wordt om het eindresultaat te krijgen wordt dan $x + x^2 + x^3 + x^4 + x^5 + ...$ Het totaalplaatje wordt dan

$$(1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7)(1 + x^2 + x^4 + ...)(x + x^2 + x^3 + x^4 + x^5 + ...)$$

Wat uitgewerkt kan worden tot

$$x+2x^2+4x^3+6x^4+9x^5+12x^6+16x^7+20x^8+24x^9+28x^10+32x^11+36x^12+40x^13+44x^14+48x^15+...$$

Dit leidt tot het eindantwoord 48

2.3 Veralgemenisering van het Binomium van Newton

De binomiaal expansie in het vorige hoofdstuk kan voor een deel herschreven worden om bij genererende functies ook weer goed gebruikte te kunnen worden. Dit is het makkelijkst door gewoon 1 voor y in te vullen.

Opmerking 2.5. $\binom{n}{0} = 1$ want $0! = 1$ en dus $\binom{n}{0} = \frac{n!}{0!n!} = \frac{n!}{1n!} = 1$.

Theorema 2.6. $(1 + x)^n = \sum_{r=0}^{n} \binom{n}{r} x^r$

Bewijs. Vul in het vorige hoofdstuk $y = 1$ in bij de binomiaal expansie en laat de som doorlopen tot ∞ in plaats van tot n. □

Ter verheldering nog een toepassing die zeker van pas kan komen:

Voorbeeld 2.7. Uitwerking $\frac{1}{(1-y)^u}$ met behulp van Theorema 2.4. Aangezien $\frac{1}{(1-y)^u} = (1 - y)^{-u} = (1 + x)^{-u}$ kan de formule gebruikt worden met $n = -u$ en $x = -y$. Dit geeft:

$$\sum_{r=0}^{\infty} \binom{-u}{r} (-1)^r y^r$$

Wat verder uitgewerkt kan worden met

$$\binom{-u}{r} (-1)^r = \frac{-u(-u-1) \cdot ... \cdot (-n-r+1)(-1)^r}{r!}$$

waarna $(-1)^r$ verspreid kan worden om de volgende formule te maken:

$$\frac{(n+r-1) \cdot ... \cdot n(n-1)}{r!} = \frac{(n+r-1)!}{r!(n-1)!} = \binom{n+r-1}{r}$$

Wat leidt tot een handig resultaat:

$$\frac{1}{(x-1)^n} = \sum_{r=0}^{\infty} \binom{n+r-1}{r} x^r$$

2.4 Exponentieel genererende functies

Bij normale genererende functies werden de combinaties geteld. Voor het tellen van permutaties kan er een kleine aanpassing gemaakt worden aan de normale genererende functies. Net als bij de combinaties en permutaties in het vorige hoofdstuk is er een factor $\frac{1}{k!}$ verschil. Nu moet deze factor meegenomen worden in iedere term van de exponentieel genererende functie bijvoorbeeld voor $(1, 1, 1, 1, 1, 1, ...)$ komt deze er zo uit te zien: $\frac{1}{0!} + \frac{x}{1!} + \frac{x^2}{2!} + ... + \frac{x^k}{k!} + ...$ Herkenbaar uit de introductie paragraaf is dit $f(x) = e^x$. De algemene vorm wordt dan $a_0\frac{1}{0!} + a_1\frac{x}{1!} + a_2\frac{x^2}{2!} + ... + a_k\frac{x^k}{k!} + ...$ voor de reeks $(a_0, a_1, a_2, ..., a_i, ...)$. Dat dit werkt wordt bewezen in de volgende stelling.

Theorema 2.8. *"Stel dat er p typen voorwerpen zijn en van voorwerp i zijn er* n_i $(i = 1, ..., p)$ *onherkenbare copieën. Definieer* g_k *als het aantal verschillende, herkenbare permutaties van lengte k, nu geldt*

$$G(x) = \sum_{k=0}^{\infty} g_k \frac{x^k}{k!} = \left(1 + x + \frac{x^2}{x!} + ... + \frac{x^{n_1}}{n_1!}\right) ... \left(1 + x + \frac{x^2}{2!} + ... + fracx^{n_p} n_p!\right)$$

(het product van de afzonderlijke exponentieel genererende functies)."[1]

Bewijs. Er wordt gebruik gemaakt van een bewijs met inductie. Het klopt voor $p = 1$ aangezien er één manier is om allemaal dezelfde objecten te kiezen, zelfs als volgorde uitmaakt. Er wordt aangenomen dat het klopt voor $p = p_0$, te bewijzen is dat het ook klopt voor $p = p_0 + 1$. Definieer $f(p_0, k)$ als het aantal verschillende mogelijke herkenbare permutaties met lengte k bestaande uit p_0 typen.

$$G(x) = \left(\sum_{k=0}^{\infty} f(p_0, k)\frac{x^k}{k!}\right)\left(1 + x + \frac{x^2}{2!} + ... + \frac{x^{n_{p_0+1}}}{n_{p_0+1}!}\right)$$

In dit product komt bij $\frac{x^k}{k!}$ de sommatie van producten van x^j in de linker kant en x^{k-j} in de rechter kant. Dit zijn alle termen die exact x^k bevatten. De coëfficiënt wordt dan $k!\sum_{j=0}^{n_{p_0+1}} \frac{f(p_0,k-j)}{(k-j)!}\frac{1}{j!}$ waarbij de $k!$ erbij zit omdat er gekeken wordt naar coëfficiënten van $\frac{x^k}{k!}$ en niet naar x^k. Dit kan omgeschreven worden in $\sum_{j=0}^{n_{p_0+1}} \binom{k}{j} f(p_0, k-j)$ wat gelijk is aan g_k omdat de voorwerpen van type p_0+1 op $\binom{k}{j}$ manieren ingevoegd kunnen worden in de gegeven volgorde van $(k - j)$ voorwerpen. □

Voorbeeldopgave 2.9. Eva wil een rij van 15 rode en witte rozen voor Naomi kopen. Hoeveel verschillende rijen rozen kan Eva voor Naomi kopen?

Hier zijn niet alleen verschillende combinaties maar ook verschillende permutaties mogelijk. De oplettende lezer ziet al dat het aantal 2^{15} is omdat iedere plaats in de rij twee opties heeft voor een kleur. Voor dit voorbeeld wordt dit vergeten en worden exponentieel genererende functies gebruikt. Voor beide kleuren rozen wordt weer dezelfde functie gebruikt, maar dit keer moet de rij $(1, 1, 1, 1, 1, 1, ...)$ in een exponentieel genererende functie opgenomen worden. Dit wordt zoals hierboven aangegeven e^x. De rode rij met de witte rij vermenigvuldigen geeft dan e^{2x} als de genererende functie van het totaal aantal opties. Dit uitwerken tot de 15de term, of de reeks ontwikkelen voor de vijftiende term geeft $\frac{16x^{15}}{638512875} = 32768\frac{x^{15}}{15!}$, wat $32768 = 2^{15}$ het antwoord maakt.

Omdat er in de twee originele reeksen gecompenseerd was voor de manieren waarop de rozen van een kleur onderling konden wisselen kan in de laatste reeks gekeken worden naar het totaal aantal permutaties. De compensatie aan het begin zijn de faculteiten en aan het eind is het de 15! die onder x^{15} staat en niet in het antwoord meegenomen wordt.

Nu een iets moeilijkere vraag:

Voorbeeldopgave 2.10. Een gameshowhost trekt willekeurig 7 ballen uit een vat met roze blauwe en gele ballen. Hij legt deze vervolgens op een rijtje. Hoeveel verschillende rijtjes kunnen er gemaakt worden gegeven dat hij 2 gele ballen en meer roze dan blauwe trekt?

Hier worden drie exponentieel genererende functies opgesteld. Aangezien de gameshowhost altijd 2 gele ballen trekt hebben we daarvoor de reeks $(0, 0, 1, 0, 0, 0, ...)$ wat als functie $\frac{x^2}{2!}$ oplevert. De techniek die eerder gebruikt is voor meer van de een dan van de ander kan helaas bij exponentieel genererende functies niet gebruikt worden, omdat dan de ballen die in paren gepakt worden met elkaar uitwisselbaar zouden zijn. Hier kunnen we de drie opties voor de roze en blauwe ballen uitschrijven, aangezien ze samen precies 5 plekken moeten vullen als er alleen naar x^7 gekeken gaat worden. Dit wordt dan $(\frac{x^3}{3!}\frac{x^2}{2!} + \frac{4}{4!}\frac{x}{1!} + \frac{x^5}{5!})$. Hierbij is steeds de eerste breuk de roze en de tweede breuk de blauwe ballen. Dit wordt weer vermenigvuldigd met de gele ballen om de uiteindelijke genererende functie en het antwoord te krijgen. $\frac{x^2}{2!}(\frac{x^3}{3!}\frac{x^2}{2!} + \frac{4}{4!}\frac{x}{1!} + \frac{x^5}{5!}) = \frac{x^7}{15} = 336\frac{x^7}{7!}$. En dat maakt het eindantwoord 336. Dit kan gecontroleerd worden met behulp van simplele telregels. Er zijn $\binom{7}{2}$ manieren om de gele ballen te plaatsen en $\binom{5}{0} + \binom{5}{1} + \binom{5}{2}$ manieren om de overgebleven 5 met blauw en roze te vullen. Dit geeft een totaal van $\binom{7}{2}(\binom{5}{0} + \binom{5}{1} + \binom{5}{2}) = 336$.

2.5 Opgaves

Opgave 18. Gebruik voor de volgende set functies al bekende generende functies om de generende functies die bij deze functies horen te vinden:
a. $x^2 + \frac{1}{1-x}$
b. $\frac{1}{4-x}$
c. $5e^x + e^{3x}$
d. $\frac{1}{1-2x}e^{2x}$

Opgave 19. Vind voor de volgende reeksen de bijbehorende genererende functie:
a. $(1, 1, 1, 1, 1, 2, 1, 1, 0, 0, ...)$
b. $(1, 1, 2, 3, 4, 5, 6, 7, 8, 9, ...)$
c. $a_k = \frac{3^k}{k!}$
d. $(1, 0, 1, 0, 1, 0, 1, 0, ...)$

Opgave 20. Vind de reeksen die horen bij de volgende functies:
a. $\frac{1}{1+x^2}$
b. $x \sin x$
c. $2x + e^{-x}$

Opgave 21. Gegeven zijn P dobbelstenen; op iedere dobbelsteen staan 6, niet noodzakelijk verschillende, willekeurige, niet-negatieve getallen. Voor iedere dobbelsteen $p(p = 1, ..., P)$ wordt het aantal vlakken met waarde k genoteerd als a_{kp} . We zijn geïnteresseerd in het totaal aantal ogen van de P dobbelstenen in één worp. Bepaal een genererende functie om dit aantal te vinden; bewijs de correctheid van uw antwoord (als je bijv. een stelling wilt gebruiken, ga dan na of je mag gebruiken). Geef ook aan hoe je het aantal mogelijkheden kunt vinden om tot een totaal van n ogen te komen bij twee worpen met alle dobbelstenen.

Opgave 22. Bewijs dat

$$\binom{p}{0} - \binom{p}{1} + \binom{p}{2} - \binom{p}{3} \pm ... + (-1)^p \binom{p}{p} = 0$$

met een combinatorisch en een algebraïsch bewijs

Opgave 23. Vind de bijbehorende reek a_k:
a. $\left(\frac{4}{1-x}\right)\left(\frac{3}{1-x}\right)$
b. $\frac{x^3+x^5}{1-x}$

Opgave 24. Vind een genererende functie voor de volgende reeksen:
a. $a_k = 7k$
b. $a_k = 5 + k$
c. $a_k = 3k + 2$

Opgave 25. Los de volgende opgaven op met genererende functies, uitwerken is niet nodig, vermeld de genererende functie en de coëfficiënt van welke term het antwoord bevat:
a. Een korfbal team bestaat uit 8 spelers, 4 vrouwen en vier mannen, gegeven drie vrouwen en twee mannen uit Egypte, twee vrouwen en vier mannen uit Peru, en twee mannen en vrouwen uit Nederland, hoeveel verschillende teams kunnen er gemaakt worden als er alleen naar de nationaliteit en geslacht gekeken wordt?
b. Een brief naar een tropische bestemming moet met $5,46$ gefrankeerd worden. Op hoeveel manieren kan dit met 5 zegels van 1, 10 zegels van 5, 15 zegels van 20, 19 zegels van 1 en 10 zegels van 2?
c. Bij een bepaald spel worden twintig zijdige dobbelstenen gebruikt, hoeveel manieren zijn er om met 10 twintig zijdige dobbelstenen 98 te gooien?

Opgave 26. a. Toon aan dat voor $|x| < 1$ geld

$$(1 - x)(1 + x)(1 + x^2)(1 + x^4)...(1 + x^{2^k}... = 1$$

b. Laat hiermee zien dat $1 + x + x^2 + ... = (1 + x)(1 + x^2)...(1 + x^{2^k}...$
c. Bewijs dat ieder getal geschreven kan worden als een som van unieke niet negatieve tweemachten. Dat wil zeggen $a_0 1 + a_1 2 + a_2 4 + ... + a_k 2^k + ...$ waarbij a_i gelijk kan zijn aan nul of één. Dit is de basis van het binaire telsysteem wat in computers gebruikt wordt.

Opgave 27. Jan trekt elke dag een van 6 verschillende broeken aan, hij wast ze nooit en in het weekend maakt het geen verschil of hij een bepaalde broek op zaterdag of

op zondag aan heeft(niemand ziet de broeken in het weekend alleen na het weekend dat ze gedragen zijn, Jan speelt zijn hele weekend alleen maar computerspelletjes). Op hoeveel voor een buitenstaander verschillende manieren kan hij zijn week indelen in broeken?

Opgave 28. Los het probleem van herkenbare dozen en onherkenbare ballen op met genererende functies. Stel eerst een genererende functie op die bij het probleem $D(n, k)$ hoort waarbij er n ballen in k dozen verdeeld moeten worden en stel aan de hand daarvan een directe formule op.

Opgave 29. Gebruik genererende functies om de volgende vragen te beantwoorden, geef weer geen antwoord maar alleen naar welke term er gekeken moet worden.
a. Hoeveel verschillende pincodes kunnen er gemaakt worden met de cijfers $1, 2, 3, 4, 6$ en 8?
b. Er wordt een rij van 20 kippen en hanen gemaakt met maximaal 7 hanen, hoeveel verschillende rijen kunnen er zijn?
c. Hoeveel verschillende al dan niet bestaande woorden kunnen er gemaakt worden met de letters j, m, h en e gegeven dat er maximaal 1 j en 2 $m$en in zitten?

Opgave 30. $A(x)$ en $B(x)$ zijn twee exponentieel genererende functies met a_k en b_k als elementen van hun reeksen. Druk het $k$de element van de reeks $C(x) = A(x) + B(x)$ uit. Doe dit ook voor $C(x) = A(x)B(x)$.

2.6 Bronverantwoording

Dit hoofdstuk is gebaseerd op Applied Combinatorics[2] met daar ook enkele opgaven uit met hogen didactische waarde(22, 30). De rest zijn opgaven van Han Hoogeveen[1](21) of originele opgaven.

Hoofdstuk 3

Recurrente betrekkingen

3.1 Introductie

Dit hoofdstuk gaat in op wat recurrente betrekkingen zijn en hoe deze opgelost kunnen worden. Een recurrente betrekking is een formule die het volgende getal in een reeks uitrekent aan de hand van voorgaande getallen. Een bekend voorbeeld zijn de Fibonacci getallen, dit is een reeks van getallen die begint met $0, 1$ en daarna verder gaat door de vorige twee getallen bij elkaar op te tellen. De reeks wordt dan verder $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...$ De recurrente betrekking die bij deze reeks hoort is $f(n) = f(n-1) + f(n-2)$ met als startwaarde $f(0) = 0$ en $f(1) = 1$.

Voorbeeldopgave 3.1. Najib heeft 4.000 euro te besteden om een buis te vullen met koffie en meel. Koffie kost 2.000 per twee meter buis en meel 500 euro per halve meter buis, deze hoeveelheden worden altijd in een keer in de buis gedaan. Als zijn geld op is is de buis ook vol. Op hoeveel verschillende manieren kan Najib de buis vullen?

Hiervoor kan een recurrente betrekking gebruikt worden die simuleert dat er iedere keer een van de twee opties gekozen wordt. Deze recurrente betrekking wordt gegeven door $f(x) = f(x-2000) + f(x-500)$. De eerste term van deze betrekking is welke opties er zijn als het volgende deel van de buis met koffie gevuld wordt en de tweede term als het volgende deel van de buis met meel gevuld wordt. Daarna zijn er weer unieke opties die dan weer recurrent berekend worden. Uiteindelijke geeft $f(4.000) = 5$ ons het gewenste antwoord.

Voorbeeldopgave 3.2. Tibor krijgt ieder jaar 5% rente over zijn kapitaal van de bank en van zijn ouders 20% rente op alles wat erbij gekomen is sinds vorig jaar, voordat de rente uitgekeerd werd, om hem aan te moedigen meer te sparen en niet uit te geven. Hij begint met 20.000 op de bank en stort nooit iets bij, hoeveel heeft Tibor na 15 jaar?

Hiervoor kan weer een recurrente betrekking opgesteld worden. Ieder jaar krijgt hij 1.05 keer het bedrag van vorig jaar en dan nog een keer 1.20 keer het verschil tussen vorig jaar en het jaar ervoor. Dan wordt de uiteindelijke recurrente betrekking $f(x) = 1.05f(x-1) + 1.20(f(x-1) - f(x-2))$ met als startwaarde $f(0) = 20.000$ en x als het jaar waarop het saldo berekend moet worden.

3.2 Verstoring

Bij een verstoring van een set gaat het er om dat alle elementen gepermuteerd worden zonder dat ook maar één van deze elementen op zijn eigen plaats terug komt. Dit is de basis van het lootjestrekken, hierbij wordt natuurlijk ook gepermuteerd zonder dat iemand zijn eigen lootje krijgt. De vraag is nu natuurlijk op hoeveel manieren dit kan. Dit is recurrent op te lossen en heeft ook zijn eigen notatie.

Definitie 3.3. Verstoringsgetal: !n = (n-1)(!(n-1)+!(n-2)) met beginwaarde !0 = 1 en !1 = 0

Voorbeeldopgave 3.4. Er is een sportdag waarbij 9 groepjes 9 activiteiten gedaan hebben en allemaal door moeten wisselen naar een andere activiteit. Op hoeveel manieren kan dit?

Een kwestie van de formule invullen, dit wordt aan de lezer overgelaten. Interessanter is waarom deze formule werkt. De groepjes worden genummerd van 1 tot en met 9 en de activiteiten ook. Voor het gemak wordt aangenomen dat groepje x net activiteit x afgerond heeft. Er wordt naar groepje 1 gekeken. Deze wisselen door naar activiteit p. Als we naar groepje p kijken zijn daar twee opties voor, ze gaan naar activiteit 1 of naar activiteit l met $l \neq 1$. Als ze naar activiteit 1 gaan dan is het overgebleven probleem !7 geworden, groepjes 1 en p doen immers niet meer mee. Als ze naar een andere activiteit gaan is het overige probleem 8! aangezien het volgende groepje weer dezelfde twee mogelijkheden heeft, bij activiteit 1 gaan staan of niet. Merk op dat ze niet bij activiteit p kunnen gaan staan want daar staat 1 al. Uiteindelijk zijn er $(n-1)$ mogelijke activiteiten voor groepje 1 om heen te gaan en dus wordt de formule $(n-1)(!(n-1)+!(n-2))$.

3.3 Sommatiefactor methode

Bij het oplossen van recurrente betrekkingen komt het relatief vaak voor dat een recurrente betrekking alleen afhankelijk is van de vorige term. Bij deze soort recurrente betrekkingen kan de sommatiefactor methode gebruikt worden. Deze formules hebben dus de vorm $f_n a_n = g_n a_{n-1} + c_n$ met f_n, g_n en c_n gegeven functies. Als geldt dat $g_n = f_{n-1}$ dan kan de substitutie $T_n = f_n a_n$ gebruikt worden om een simpelere vergelijking te krijgen met alleen een te sommeren term. Deze substitutie levert dan $T_n = T_{n-1} + c_n$ op wat makkelijk omgeschreven kan worden in een niet recurrente functie. Dit is echter lang niet altijd het geval. Hiervoor moet de recurrente betrekking een beetje geholpen worden door aan beide kanten te vermenigvuldigen met een sommatiefactor s_n. Dit geeft $s_n f_n a_n = s_n g_n a_{n-1} + s_n c_n$. Om nu te zorgen dat de substitutie toe gepast kan worden moet gelden dat $s_{n-1} f_{n-1} = s_n g_n$. Hieruit volgt dat $s_n = \frac{s_{n-1} fn-1}{g_n}$ en met het verder uitwerken van s_n dat $s_n = \frac{f_{n-1}}{g_n}\frac{f_{n-2}}{g_{n-1}}\frac{f_{n-3}}{g_{n-2}}...\frac{f_1}{g_2}s_1$. De constante factor s_1 is niet van belang voor de berekening aangezien er aan beide kanten met deze factor vermenigvuldigd wordt. Er moet wel op gelet worden dat als een van de formules f_n, g_n nul dreigt te worden er gestopt wordt met s_n expanderen.

Voorbeeldopgave 3.5. Los op met de sommatiefactor methode $n^2 a_n = (n-1)a_{n-1} + 2$

Hier is er $f_n = (n^2)$ en $g_n = (n-1)$ en dus $s_n = \frac{(n-1)^2}{n-1}...\frac{1^2}{1} = (n-1)!$. De versimpelde recurrente betrekking wordt dan $T_n = T_{n-1} + 2(n-1)!$ en $a_n = \frac{T_n}{(n-1)!}$.

3.4 JBF methode

Bij de Jan Boeren Fluitjes(JBF) methode worden recurrente betrekkingen van de vorm $a_n = c_1 a_{n-1} + c_2 a_{n-2} + ... + c_k a_{n-k} + f(n)$ met startwaarde $a(0) = \alpha_0, a(1) = \alpha_1, ..., a(k) = \alpha_k$opgelost. Dit wordt gedaan door een algemene oplossing voor de homogene vergelijking(AOHV) te vinden, Daarna een particuliere oplossing voor de inhomogene vergelijking(POIV) en daarna deze bij elkaar te voegen om een algemene oplossing te vormen voor de inhomogene vergelijking(AOIV). Mensen die de cursus infinitesimaalrekening A gevolgd hebben zullen dit herkennen aan de manier waarop differentiaalvergelijkingen opgelost worden tijdens die cursus. In formule vorm $AOIV = AOHV + POIV$.

Voor een inhomogene recurrente betrekking $a_n = c_1 a_{n-1} + c_2 a_{n-2} + ... + c_k a_{n-k} + f(n)$ kan een homogene recurrente betrekking gevonden worden door de term $f(n)$ weg te laten. Dit is de enige term die niet afhankelijk is van een vorig getal in de reeks a_n. Dan wordt de homogene recurrente betrekking gevonden $a_n = c_1 a_{n-1} + c_2 a_{n-2} + ... + c_k a_{n-k}$ die opgeloste kan worden door $a_n = q^n$ in te vullen voor zekere $q \in \mathbb{R}$. Dit levert de volgende formule op die omgeschreven wordt tot een polynoom van graad k:

$$\begin{aligned} q^n &= c_1 q^{n-1} + ... + c_k q^{n-k} \\ 0 &= -q^n + c_1 q^{n-1} + ... + c_k q^{n-k} \\ 0 &= -q^k + c_1 q^{k-1} + ... + c_{k-1} q + c_k \end{aligned} \tag{3.1}$$

Deze polynoom heet de karakteristieke vergelijking. De nulpunten van de karakteristieke vergelijking zijn ook stabiel onder de recurrente betrekking, dat wil zeggen $a_n = x_0^n$ met x_0 een nulpunt van de karakteristieke vergelijking voldoet aan de gelijkheid gedefinieerd door de recurrente betrekking. Bij meervoudige nulpunten is het zelfs zo dat, als μ de multipliciteit van het nulpunt, $a_n = x_0^n$, $a_n = nx_0^n$, ... , $a_n = n^{\mu-1}x_0^n$ allemaal voldoen aan de gelijkheid gedefinieerd door de recurrente betrekking. Nu wordt de algemene oplossing van de homogene vergelijking al deze waardes die voldoen aan de gelijkheid gedefinieerd door de recurrente betrekking keer hun eigen constante bij elkaar opgeteld. Dus $a_n = \lambda_{0_0} x_0^n + ... + \lambda_{0_{\mu-1}} n^{\mu-1} x_0^n + ... + \lambda_{k_0} x_k^n + ... + \lambda_{k_{\phi-1}} n^{\phi-1} x_k^n$.

Met deze algemene oplossing voor de homogene vergelijking kunnen we de rest van het probleem aanpakken. Als de op te lossen vergelijking een homogene vergelijking was worden nu de startwaarde gebruikt om de constante λ's te bepalen. Dit wordt gedaan door de startwaardes in te vullen en dan te berekenen welke waardes de λ's moeten krijgen, zodat dit klopt.

Het vinden van een particuliere oplossing voor de inhomogene vergelijking gebeurt door één waarde p_n te vinden waarvoor geldt dat $p_n = c_1 p_{n-1} + c_2 p_{n-2} + ... + c_k p_{n-k} + f(n)$. Zoals sommige reeds weten is het onbekend hoe een koe een haas vangt, dus een perfecte methode om deze p_n te vinden bestaat niet. Er is wel een methode die

meestal werkt en voor de loop van dit vak altijd. Neem $p_n = cf(n)$ met c een constante, dus bijvoorbeeld voor $f(n) = 3 \cdot 2^n$ wordt $p_n = c2^n$ genomen. Vul dan p_n in in de recurrente betrekking en kies c zo dat $p_n = c_1p_{n-1} + c_2p_{n-2} + ... + c_kp_{n-k} + f(n)$. Als dit niet werkt probeer het dan nog een keer maar vermenigvuldig p_n eerst met n. Dus dan probeer je $ncf(n)$, $n^2cf(n)$ enzovoorts.

Voor het uiteindelijke antwoord wordt de formule $AOIV = AOHV + POIV$ gebruikt en wordt de particuliere oplossing bij de algemene oplossing opgeteld. Daarna worden de λ's op zelfde wijze uitgerekend als bij de algemene oplossing voor de homogene vergelijking om een eindantwoord te vormen.

Dit is een grote lap tekst maar met een paar keer oefenen komt de vaardigheid vanzelf:

Voorbeeldopgave 3.6. Vindt een algemene oplossing voor $a_n = 4a_{n-1} + 3a_{n-2} + 2^n$ met startwaarde $a(0) = 1$, $a(1) = 1$

Eerst wordt de algemene oplossing voor de homogene vergelijking gezocht. De homogene vergelijking is $a_n = 4a_{n-1} + 3a_{n-2}$. De karakteristieke vergelijking wordt dan $0 = q^2 + 4q + 3$ met as oplossingen $x = 3$ of $x = 1$ en dus als algemene oplossing voor de homogene vergelijking $\lambda_0 3^n + \lambda_1 1^n$. Dan wordt de particuliere oplossing gevonden door 2^n te proberen. Dit geeft de vergelijking $c2^n = 4c2^{n-1} + 3c2^{n-2} + 2^n$ dan wordt deze vergelijking uitgewerkt:

$$\begin{aligned} c2^n &= 4c2^{n-1} + 3c2^{n-2} + 2^n \\ c2^n &= 2c2^n + 0,75c2^n + 2^n \\ c2^n &= 2,75c2^n + 2^n \\ -1,75c &= 1 \end{aligned} \tag{3.2}$$

Wat het antwoord $\frac{4}{7}2^n$ oplevert. Dit wordt opgeteld bij de algemene oplossing voor de homogene vergelijking voor $\lambda_0 3^n + \lambda_1 + \frac{4}{7}2^n$. Nu worden de startwaardes ingevuld om de λ's uit te rekenen. $\lambda_0 3^0 + \lambda_1 + \frac{4}{7}2^0 = 1$ en $\lambda_0 3^1 + \lambda_1 + \frac{4}{7}2^1 = 1$ dus $\lambda_0 + \lambda_1 = \frac{3}{7}$ en $3\lambda_0 + \lambda_1 = -\frac{1}{7}$ wat $\lambda_0 = -\frac{2}{7}$ en $\lambda_1 = \frac{5}{7}$ oplevert. Het eindantwoord is dan $a(n) = -\frac{2}{7}3^n + \frac{5}{7} + \frac{4}{7}2^n$

3.5 Opgave

Opgave 31. Om het spaargedrag te bevorderen, geeft een bank jaarlijks 8% rente op spaartegoeden die tussen 1 en 2 jaar oud zijn en 10% rente op spaartegoeden die minstens 2 jaar oud zijn. Stel een recurrente betrekking op voor het kapitaal na n jaar, aannemend dat het beginkapitaal 100 euro is en dat geen stortingen of opnamen plaatsvinden.

Opgave 32. De waarde van een bepaald tweedehands artikel wordt als volgt bepaald: na 1 jaar 80% van de nieuwwaarde, daarna vermindert de waarde elk jaar met de helft van de vermindering in het vorige jaar. Stel een recurrente betrekking op voor de waarde van een artikel dat n jaar oud is, uitgaande van een nieuwwaarde van 200 euro.

Opgave 33. Het konijnenprobleem van Fibonacci had de volgende karakteristieken:

- In maand 0 zijn er geen konijnen
- In maand 1 komen er tienduizend jonge konijnenparen (oorspronkelijk 1)
- Een paar is volwassen vanaf de tweede maand
- Een volwassen paar krijgt iedere maand één nieuw paar nakomelingen
- De konijnen sterven niet.

Aan dat laatste gaan we wat doen. De natuurlijke sterfte is 10% per maand. Van de overige 90% wordt 20% geschoten door jagers; dit gebeurt voordat er jonge konijnen worden geboren. De kans om te sterven is onafhankelijk van de leeftijd van de konijnen. Bepaal een recurrente betrekking voor het aantal konijnenparen in maand n. Wordt het een plaag, of sterven de konijnen uit?

Opgave 34. Een ietwat onhandige Osiris administrator verwijdert per ongeluk de studentnummers van 6 studenten. Omdat studenten systematisch genummerd worden is het niet heel moeilijk te reconstrueren welke nummers dit waren. De studentnummers worden willekeurig teruggegeven aan de studenten, hoeveel manieren zijn er om alle studenten een ander nummer te geven? En hoeveel om een student zijn eigen nummer te geven?

Opgave 35. Na een lang vermoeiend traject is dit dictaat afgeschreven. Om wat stress los te laten gaat de schrijver naar de wellness. Hier zijn twee verschillende behandelingen te krijgen. De schrijver voelt zich erg gestrest en besluit de hele 400 euro die hij meegenomen heeft aan deze behandelingen uit te geven. De ene behandeling kost 25 en de ander 50 euro. Het effect is natuurlijk verschillend afhankelijk van in welke volgorde en welke behandelingen er gebruikt worden. Op hoeveel manieren kan de schrijver ontstressen?

Opgave 36. De verwachte waarde van een relatief zelfdzaam stripboek wordt afgeschat door aan te namen dat de waarde omhoog gaat met het gemiddelde van 10% meer dan vorig jaar en het verschil vorig jaar en het jaar daarvoor. Stel een recurrente betrekking op.

Opgave 37. Tijdens een bridge toernooi moeten $2n$ paren n wedstrijden spelen. Dit gebeurt zoals met veel sporten, een paar tegen een ander paar. Op hoeveel manieren kunnen deze wedstrijden gehouden worden?

Opgave 38. Er staan n stoelen in een cirkel. Laat L_n, *Lucas number*, het aantal subsets van n stoelen zijn waar geen twee stoelen achter elkaar staan.
a. Toon aan dat $L_n + 1 = F_n + F_{n+2}$ met F_n het $n$de Fibonacci getal.
b. Bepaal L_1 en L_2
c. Stel een recurrente betrekking op voor L_n gebruikmakende van enkel andere lucas getallen.

Opgave 39. laat F_n het n'de Fibonacci getal en $F_0 = 1$, bewijs dat ieder $n$de Fibonacci getal deelbaar is door F_{n-1}. Het kan handig zijn dit eerst voor $n = \{2, 3, 5\}$ te bewijzen.

Opgave 40. Los de volgende recursieve vergelijking op met de gegeven startwaardes, de karacteristieke wortels zijn i en $-i$:

$$a_n = -2a_{n-2} - a_{n-4} \qquad a_0 = 0, a_1 = 1, a_2 = 2, a_3 = 3$$

Opgave 41. Los de volgende recurrente betrekking op:
a. $a_n = 5a_{n-1} - 6a_{n-2}$ voor $n \geq 2$, met $a_0 = 0$ en $a_1 = 1$
b. $a_n = 5a_{n-1} - 6a_{n-2} + n^2$ voor $n \geq 2$, met $a_0 = 0$ en $a_1 = 1$
c. $a_n = 5a_{n-1} - 6a_{n-2} + 2^n$ voor $n \geq 2$, met $a_0 = 0$ en $a_1 = 2$
d. $a_n = 5a_{n-1} - 6a_{n-2} + n2^n$ voor $n \geq 2$, met $a_0 = 0$ en $a_1 = 2$
e. $a_n = 5a_{n-1} - 6a_{n-2} + 2n^2 + n2^n$ voor $n \geq 2$, met $a_0 = 0$ en $a_1 = 4$

Opgave 42. Los het verstoringsprobleem op met de sommatie factor methode.

Opgave 43. Los de recurrente betrekking $a_n = (n-1)a_{n-1} - 2n!$ op met de sommatie factor methode voor $n \geq 1$.

Opgave 44. Los de recurrente betrekking $(n^2 - 2n)a_n = (n^2 - 3n + 2)a_{n-1} + (n^2 + 6n - 16)$ op met de sommatie factor methode.

Opgave 45. Bepaal de coëfficiënten u_k van de genererende functie

$$U(x) = \sum_{k=0}^{\infty} u_k x^k = \sqrt{2 + 4x}$$

De verkregen uitdrukking hoeft niet verfraaid te worden.

Opgave 46. Vind voor de volgende genererende functies de bijbehorende a_k:
a. $\frac{1}{(1-x)(1-3x)}$
b. $\frac{2x+1}{(1-3x)(1-2x)}$
c. $\frac{x}{x^2-5x+6}$

Opgave 47. Bewijs dat alle Catalan getallen $u_n = \frac{1}{1+x}\binom{2n}{n}$ voor $n = 0, 1, 2, ...$ gehele getallen zijn door twee binomiaalcoefficienten te vinden die als verschil u_n hebben. Hint: kijk naar $\binom{2n}{n}$

3.6 Bronverantwoording

De uitleg van de JBF methode gebruikt zelfde termen als op dit moment gebruikt worden in de colleges infinitesimaalrekening A om begrip bij wiskundigen te verhogen (het vak is verplicht voor wiskunde studenten). De opgaven zijn een combinatie van opgaven opgegeven door Han Hoogeveen[1](31 ,32 ,33 , 41), didactisch erg goede opgaven uit Applied combinatorics[2](38, 44, 47) en originele opgaven.

Hoofdstuk 4

Inclusie(f) en exclusie(f)

4.1 Introductie

Om direct het nut en wat motivatie achter inclusie en exclusie te zetten start dit hoofdstuk met een voorbeeld.

Voorbeeldopgave 4.1. Hoeveel getallen zijn er tussen de 100 en de 1000 die niet deelbaar zijn door 3 of 5?

Eerst wordt het aantal getallen niet deelbaar door drie bepaald. het eerste getal deelbaar door 3 is 102 en het laatste is 999, dus het verschil gedeeld door 3 plus 1 voor het eerste getal is het aantal getallen deelbaar door 3. Dit is 300 en het aantal niet deelbaar door 3 is dus 598. Dan zijn er 180 deelbaar door vijf op soortgelijke wijze wat de totaalsom van 418 geeft. Helaas zijn een deel van de getallen deelbaar door 3 ook deelbaar door 5 en die zijn nu dubbel geteld. We moeten de getallen deelbaar door 5 en 3 dus weer optellen. Dit doen we door te kijken welke getallen deelbaar zijn door 3 en 5. Een getal is deelbaar door een ander getal dan en slechts dan als het deelbaar is door alle priemdelers van het andere getal. Een getal is dus deelbaar door 3 en 5 dan en slechts dan als het deelbaar is door 15. Tussen de 105 en de 990 zijn 59 getallen ook deelbaar door 15 plus die ene die we nu overgeslagen hebben is 60. Dat tellen we dan weer op voor het totaalantwoord 478

Hierbij is het principe van inclusie exclusie gebruikt. Eerst werden de getallen deelbaar door 3 of 5 afgetrokken waarna we de getallen deelbaar door 3 en 5 weer moesten optellen. Dit omdat als het aantal getallen deelbaar door 3 afgetrokken wordt en daarna ook nog de getallen deelbaar door 5 getallen deelbaar door 3 en 5 twee keer afgetrokken zijn en dus weer opgeteld moeten worden.

Dit concept kan ook algemeen toegepast worden op een set A met N objecten die eigenschappen $a_1, ..., a_n$ hebben. Neem nu $N(a_k)$ het aantal objecten met eigenschap a_k en $N(a'_k)$ het aantal objecten zonder eigenschap a_k. We vinden nu dat $N = N(a_k) + N(a'_k)$. We definiëren $N(a_k a_j)$ als het aantal objecten dat beide eigenschappen a_k en a_j heeft, dit werkt natuurlijk identiek met 3 of meer eigenschappen en met het niet hebben van eigenschappen.

Met deze notatie kunnen we een regel definiëren om het concept inclusie en exclusie wat meer vorm te geven.

In de voorbeeldopgave was A gelijk aan de getallen tussen de 100 en de 1000 dit waren er $N = 899$. Dan was er a_1 als deelbaarheid door 3 en a_2 als deelbaarheid door 5. De formule wordt dan $N(a_1'a_2') = N - N(a_1) - N(a_2) + N(a_1a_2)$ wat ons exact de gemaakte berekening geeft.

Dit kan voor ieder aantal vereiste, bijvoorbeeld voor $N(a_1'a_2'a_3')$ is de formule:

$$N(a_1'a_2'a_3') = N - N(a_1) - N(a_2) - N(a_3) + N(a_1a_2) + N(a_1a_3) + N(a_2a_3) - N(a_1a_2a_3)$$

De eerste vier termen zijn evident, de volgende 3 zijn omdat er weer dingen dubbel afgetrokken zijn. De laatste term is omdat als een object aan alle drie de eisen voldoet hij drie keer opgeteld en drie keer afgetrokken wordt en dus uiteindelijk nog niet meegerekend wordt.

Dit kan ook geheel in een stelling gevat worden:

Theorema 4.2. *In een set A met N objecten wordt het aantal objecten dat geen van de eigenschappen $a_1, a_2, ..., a_n$ heeft gegeven door*

$$N(a_1', a_2', ..., a_n') =$$
$$N - \sum_i N(a_i) + \sum_{i \neq j} N(a_i, a_j) - \sum_{i \neq j, j \neq k, i \neq k} N(a_i, a_j, a_k) \pm ... + (-1)^n N(a_1, a_2, ..., a_n) \tag{4.1}$$

Hier wordt eerst het aantal objecten met eigenschap a_1 afgetrokken waarna alle ongeordende paren i, j weer toegevoegd worden en zo door.

Bewijs. Dit kan bewezen worden door te laten zien dat ieder object met geen van de eigenschappen exact een keer geteld wordt en de rest 0 keer. In de term N worden alle objecten een keer geteld, in de daarop volgende termen worden objecten met geen van de eigenschappen nooit meer geteld of afgetrokken. Uiteindelijk worden deze dus 1 keer geteld. Stel een object heeft p van de eigenschappen, dan wordt dit object in de eerste som $\sum_i N(a_i)$ p keer geteld. In de tweede som wordt dit object $\binom{p}{2}$ keer geteld, een keer voor ieder paar eigenschappen. Aangezien dit er p zijn zijn er dus $\binom{p}{2}$ paren. Op dezelfde manier telt de volgende som weer $\binom{p}{3}$ op. Uiteindelijk is de totale som voor een object met p eigenschappen gelijk aan

$$1 - p + \binom{p}{2} - \binom{p}{3} \pm ... + (-1)^n \binom{p}{n}$$

. De laatste paar termen kunnen hiervan afgehaald worden aangezien $p \leq n$ en $\binom{r}{r+k} = 0$ als k positief is. Dit maakt de nieuwe vergelijking

$$\binom{p}{0} - \binom{p}{1} + \binom{p}{2} - \binom{p}{3} \pm ... + (-1)^p \binom{p}{p}$$

. Deze is gelijk aan 0 zoals gezien in een opgave in hoofdstuk 1. □

De laatste theorie in dit hoofdstuk gaat over het tellen van voorwerpen met exact m eigenschappen. Dit aantal wordt e_m genoemd. Ook wordt s_t gedefinieerd als $\sum_{i_1,i_2,...,i_t} N(a_{i_1}, ..., a_{i_t})$. Met deze definities kan de volgende stelling bewezen worden.

Theorema 4.3. *Het aantal objecten met exact m eigenschappen wordt gegegeven door*

$$e_m = s_m - \binom{m+1}{1} s_{m+1} + \binom{m+2}{2} s_{m+2} \pm ... + (-1)^{r-m} \binom{m+r-m}{r-m} s_r$$

ofwel $\sum_{t=0}^{r-m} \binom{m+t}{t} s_{m+t}$

Bewijs. Merk op dat het vorige theorema een speciaal geval van dit theorema is maar dan met $m = 0$. Een zelfde soort bewijs volstaat. Alles met minder dan m eigenschappen wordt niet geteld, alles met m eigenschappen wordt 1 keer geteld in s_m en nooit in de rest. Alles met meer dan m eigenschappen, namelijk $m + j$. Bij het berekenen van s_{m+p} wordt dit object $\binom{m+j}{m+p}$ keer geteld voor $p <= j$ en verder nergens. In iedere term wordt hij dus $\binom{m+j}{m+p}\binom{m+p}{p}$ keer geteld. In een opgave in hoofdstuk 1 is reeds bewezen dat $\binom{m+j}{m+p}\binom{m+p}{p} = \binom{m+j}{m}\binom{j}{p}$. Iedere s_{m+p} kan dus geschreven worden als $\binom{m+j}{m}\binom{j}{p}$ maar aangezien $\binom{m+j}{m}$ niet afhankelijk is van p kan deze voor de sommatie gezet worden. Het totale aantal keer dat een object met $m + j$ eigenschappen nu geteld wordt is

$$\binom{m+j}{m}\left(\binom{j}{0} - \binom{j}{1} \pm ... + (-1)^j \binom{j}{j}\right)$$

zoals in theorema 7.1 is de rechter kant van deze vergelijking gelijk aan 0. □

4.2 Werken met inclusie en exclusie

Deze paragraaf bevat twee wat lastigere voorbeeldopgave.

Voorbeeldopgave 4.4. Een klas van 28 mensen bevat 10 hockeyers 12 voetballers en 13 handballers, 3 mensen doen aan voetbal en hockey, 5 aan voetbal en handbal, 2 aan handbal en hockey en 2 aan alle drie de sporten. Hoeveel mensen doen er geen sport?

Hierbij is de formule weer te gebruiken, voor $N = 28$, het antwoord is $N(a'_1a'_2a'_3)$ met a_1, a_2, a_3 hockeyers, voetballers en handballers respectievelijk. Dus $N(a'_1a'_2a'_3) = N - N(a_1) - N(a_2) - N(a_3) + N(a_1a_2) + N(a_1a_3) + N(a_2a_3) - N(a_1a_2a_3) = 28 - 10 - 12 - 13 + 3 + 5 + 2 - 2 = 1$ en dat is de wiskundige.

Voorbeeldopgave 4.5. Hoeveel getallen zijn er van 1000 tot en met 10000 die niet op een 0 eindige, niet deelbaar zijn door 3 en niet deelbaar door 7

We gebruiken weer dezelfde formule. De eigenschappen a_1, a_2, a_3 zijn, eindigt op een 0, deelbaar door 3 en deelbaar door 7 respectievelijk. Eindigen op 0 en deelbaar zijn door 10 zijn hetzelfde. $N = 10000 - 1000 = 9000$, $N(a_1) = 9000/10 = 900$, $N(a_2) = 9000/3 = 3000$, $N(a_3) = 9000/7 = 1285$. Dan de overlap uitrekenen: $N(a_1a_2) = 9000/30 = 300$, $N(a_1a_3) = 9000/70 = 128$ en $N(a_2a_3) = 9000/21 = 428$. Dan de laatste term nog $N(a_1a_2a_3) = 42$. Nu kan dit alles de formule in voor een totaal antwoord van $9000 - 900 - 3000 - 1285 + 300 + 128 + 428 - 42 = 4629$.

Voorbeeldopgave 4.6. Een groep van n mensen gaat lootjes trekken, gebruik inclusie exclusie om te bepalen op hoeveel manieren niemand zijn eigen lootje kan krijgen.

Hierbij worden $a_1, ..., a_n$ gekozen als de eigenschap dat persoon 1 zijn eigen lootje heeft. Nu kan de som simpelweg ingevuld worden

$$n - (n-1)!\binom{n}{1} + (n-2)!\binom{n}{2} \pm ... + (-1)^n 0!\binom{n}{n}$$

En deze som geeft dan het antwoord.

4.3 Opgave

Opgave 48. Hoeveel getallen tussen 1 en 1000 zijn deelbaar door geen van de getallen $2, 3, 5, 7$?

Opgave 49. Bepaal het aantal positieve, gehele getallen ≤ 1000 die aan de volgende eisen voldoen:

- Ze zijn niet deelbaar door 3
- Ze zijn wel deelbaar door 4
- Ze zijn niet deelbaar door 16

Opgave 50. Gebruik inclusie exclusie om te bepalen hoeveel manieren er zijn om n mensen hun hoed opnieuw over die n mensen te verdelen.

Opgave 51. Gebruik theorema 4.3 om te bepalen hoeveel permutaties er zijn waarbij er exact 4 van 8 elementen op hun plaats blijven staan. Bereken het ook direct.

Opgave 52. Van 80 discrete wiskunde studenten studeren er 48 informatica en 32 wiskunde, er zijn ook 5 TWINFO'ers. Hoeveel mensen komen van een andere studie?

Opgave 53. Een wijn proever proeft 5 wijnen en noemt 5 namen, hij noemt nooit dezelfde naam en weet welke vijf wijnen hij gaat proeven. Hij heeft er 3 van de 5 goed, als hij gegokt heeft hoe groot is dan de kans dat hij er op zijn minst 3 goed heeft.

4.4 Bronvermelding

De notatie is overgenomen van wat Han Hoogenveen[1] in zijn colleges gebruikt en wat er gebruikt wordt in Applied Combinatorics[2]. De Opgaven zijn een combinatie van opgaven van Han Hoogeveen (49), opgaven uit Applied Combinatorics (53) met een hoge didactische waarde en originele opgaven.

Hoofdstuk 5

Polya theorie

Mensen met kennis van groepentheorie zullen in dit hoofdstuk veel herhaling vinden. Enkele stellingen en lemma's zijn nieuw en vereisen zeker wat oefening.

5.1 Equivalentierelaties

Een equivalentierelatie S op een set A is een set van geordende paren die equivalente objecten bevatten.

Voorbeeld 5.1. Een tuin bevat een roodborstje, een kat, een merel en een duif. Iemand telt hoeveel diersoorten er in de tuin zitten en concludeert dat dit er twee zijn, katten en vogels. Bij het kijken naar soort is de equivalentierelatie op deze set

$$S = \{\{duif, merel\}, \{merel, duif\}, \{merel, roodborstje\}, \{roodborstje, merel\}, \{duif, roodborstje\}, \{roodborstje, duif\}, \{merel, merel\}, \{duif, duif\}, \{roodborstje, roodborstje\}, \{kat, kat\}\} \quad (5.1)$$

Om aan te duiden dat twee elementen equivalent zijn onder een bepaalde equivalentierelatie gebruikt men de volgende notatie aSb, a is equivalent aan b onder S. Een equivalentierelatie heeft enkele eigenschappen:

Definitie 5.2. een Equivalentierelatie S voldoet aan

- Reflexiviteit: aSa, a is equivalent aan zichzelf.
- Symmetrie: $aSb \Leftrightarrow bSa$, als a equivalent is aan b dan is b ook equivalent aan a.
- Transitiviteit: $aSb, bSc \Rightarrow aSc$, als a equivalent is aan b en b equivalent is aan c dan is a equivalent aan c.

In voorbeeld 5.1 werd een equivalentierelatie gedefinieerd door twee verschillende soorten dieren. Hierbij moet wel opgelet worden dat dit daadwerkelijk een equivalentierelatie is.

Voorbeeldopgave 5.3. Neem S de set geordende paren die alle combinaties van getallen met zichzelf bevat, alle combinaties (a, b) waarvoor geldt dat $a = b + 3k$ met k een geheel getal en alle combinaties (a, b) waarvoor geldt dat $a = b + 5k$ met k een geheel getal. Definieert S op alle gehele getallen een equivalentierelatie ?

Het antwoord hier is nee, 105 is equivalent aan 7, want 105 is deelbaar door 7, en aan 15, want 105 en 15 zijn deelbaar door 3, maar 7 is niet equivalent aan 15. Dit breekt met de transitiviteit in de definitie van equivalentierelaties. Want $7S105$, $105S15$ maar 7 is niet equivalent aan 15. Dit is dus geen equivalentierelatie.

Een equivalentierelatie definieert ook equivalentieklassen. Dit zijn groepen objecten die equivalent aan elkaar zijn. Voor een equivalentieklasse C van een equivalentierelatie S op een set A moet gelden $a, b \in C \Rightarrow aSb$ en er moet ook gelden dat $a \in C,\ b \in A,\ aSb \Rightarrow b \in C$. In het dieren voorbeeld zitten twee equivalentieklasse, $\{kat\}$ en $\{duif, merel, roodborstje\}$.

Voorbeeldopgave 5.4. Is de equivalentierelatie S op de gehele getallen gedefinieerd door $aSb \Leftrightarrow a = b + 2k$ met k in de gehele getallen een equivalentierelatie? Wat zijn de equivalentieklasse?

Dit is een equivalentierelatie want:

- aSa met $k = 0$
- als aSb met k dan bSa met $-k$
- als aSb met k_1 en bSc voor k_2 dan aSc met $k = k_1 + k_2$

De equivalentieklasse worden gegeven door $\{..., -3, -1, 1, 3, ...\}$ en $\{..., -4, -2, 0, 2, 4, ...\}$.

5.2 Elementaire groepentheorie

Een aantal stellingen in dit hoofdstuk die gebruikt zullen worden om telproblemen op te kunnen lossen met equivalentierelaties vereisen een basiskennis van groepentheorie. Deze paragraaf zal hierover uitweiden en ook groepen van permutaties introduceren als een manier om een equivalentierelatie te definiëren. Tijdens dit vak zal er eigenlijk alleen gekeken worden naar groepen van permutaties maar het is goed om te weten dat groepen ook een algemeen wiskundig begrip zijn.

Definitie 5.5. Groep, een groep is een set G en een operatie $\circ$ met de volgende eigenschappen:

- G1, geslotenheid: als $a, b \in G$ dan $a \circ b \in G$
- G2, associativiteit: $(a \circ b) \circ c = a \circ (b \circ c)$
- G3, het bestaan van een eenheid: er is een $I \in G$ zodat $I \circ a = a$ voor alle $a \in G$
- G4, het bestaan van een inverse: voor alle $a \in G$ is er een $-a \in G$ zodat $a \circ (-a) = I$

Voorbeeld 5.6. De gehele getallen en optelling vormen samen een groep want

- G1: Elke twee gehele getallen kunnen bij elkaar opgeteld worden en zijn dan weer een geheel getal.

- G2: $(a+b)+c = a+b+c = a+(b+c)$ want de volgorde van optelling maakt niet uit.
- G3: De eenheid is het getal 0 want $0 + a = a$
- G4: De inverse van een getal a is $-a$ want $a - a = 0$

Dit was het enige voorbeeld waarin er gekeken wordt naar een groep die niet uit permutaties bestaat. Groepen van permutaties moeten wel voldoen aan dezelfde vereiste, als ze niet aan de vereiste van een groep voldoen werken de stellingen later in dit hoofdstuk ook niet. Check dus altijd als een antwoord niet lijkt te kloppen of de groep van permutaties wel echt een groep is. De operatie die bij een permutatiegroep hoort is $\circ$ waarbij $\pi_1 \circ \pi_2$ de permutatie definieert waarbij eerst π_2 uitgevoerd wordt en dan op het resultaat π_1 ook nog uitgevoerd wordt. Ter illustratie het volgende voorbeeld.

Voorbeeld 5.7. Zarah heeft een ketting met 3 kralen om die ze achter haar nek door kan schuiven. Alle permutaties die zij op de volgorde van de kralen in de ketting uit kan voeren geven de volgende permutatie groep

$$G = \{\pi_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}, \pi_2 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}, \pi_3 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}\}$$

Om aan te tonen dat dit inderdaad een permutatiegroep is worden alle vereiste gecheckt:

- G1: $\pi_1 \circ \pi_2 = \pi_2$, $\pi_1 \circ \pi_3 = \pi_3$, $\pi_2 \circ \pi_3 = \pi_1$, $\pi_1 \circ \pi_1 = \pi_1$, $\pi_2 \circ \pi_2 = \pi_3$ en $\pi_3 \circ \pi_3 = \pi_2$. De groep is dus gesloten.
- G2: Doordat permutaties altijd van links naar rechts doorgelopen worden komt er altijd hetzelfde antwoord uit. bijvoorbeeld $a(bc) = abc = (ab)c$ omdat volgorde van behandelen niet uitmaakt maar volgorde van permutaties wel, zo is het meestal zo dat $ab \neq ba$.
- G3: π_1 is de eenheid zoals bij G1 al te zien is.
- G4: Zoals bij G1 te zien is π_1 zijn eigen inverse en zijn π_2 en π_3 elkaars inverse.

In dit voorbeeld wordt een echte wereld handeling omgezet in een permutatiegroep. Als de ketting nu twee kleuren kralen krijgt, blauw en rood, kan Zarah bijvoorbeeld de kleurcombinatie rrb omzetten in brr door de blauwe kraal achter haar nek door te schuiven. In de permutatiegroep komt dit overeen met π_2. Voor Zarah is de kleurcombinatie rrb dus gelijk aan de kleurcombinatie brr. Op deze manier definieert de permutatiegroep een equivalentierelatie op de verzameling van uitkomsten.

Een permutatiegroep G definieert een equivalentierelatie op een set door aSb dan en slechts dan als $\pi(a) = b$ voor een $\pi \in G$. Dat wil zeggen twee objecten zijn equivalent op het moment dat er een permutatie in de groep zit die het ene object in het andere veranderd(de permutatie de andere kant op zit er ook in door $G4$). Omdat G een groep is is dit een geldige equivalentierelatie:

- Reflexiviteit: de identiteit is een element van G en $\pi_{id}(a) = a$ dus aSa

- Symmetrie: Ieder element in G heeft een inverse, iedere aSb heeft een π zodat $\pi(a) = b$ en iedere π heeft een $-\pi$ zodat $-\pi \circ \pi(a) = -\pi(\pi(a)) = a$ wat betekent dat $-\pi(b) = a$ en dus bSa
- Transitiviteit: als aSb met π_1 en bSc met π_2 dan is aSc met $\pi_1 \circ \pi_2$

Het volgende voorbeeld ter illustratie:

Voorbeeldopgave 5.8. Hoeveel unieke kleurcombinaties kan Zarah hebben met de kleuren blauw en rood?

Bij deze vraag worden de kleurcombinaties opgedeeld in verzamelingen die hetzelfde zijn onder toepassing van een van de permutaties van de permutatiegroep. Het totale aantal verzamelingen is vervolgens het aantal verschillende kleurcombinaties. In dit geval zijn de verzamelingen

$$\{rrb, brr, rbr\}, \{rbb, brb, bbr\}, \{bbb\}, \{rrr\}$$

en zijn er dus 4 unieke kleurcombinaties mogelijk. De kleurcombinaties binnen een van deze verzamelingen zijn equivalent aan elkaar onder de equivalentierelatie S gedefinieerd als aSb dan en slechts dan als er een π in G zit waarvoor geldt dat $\pi(a) = b$. Op deze manier kunnen de equivalentierelaties van de permutatiegroep geteld worden om te berekenen hoeveel unieke kleuringen er zijn. In de volgende paragraaf zal er gekeken worden naar een snellere manier van tellen.

5.3 Burnside's lemma

Voor Burnside's lemma wordt eerst nog wat nodige theorie behandeld. Er is redelijke overlap met groepentheorie en voordat Burnside's lemma bewezen wordt staat nog een bondig overzicht van alle nodige kennis voor die mensen die denken bekend te zijn op het gebied van groepen. Het bewijs van Burnside's lemma is niet nodig voor de toepassing ervan maar verhoogt wel begrip van permutatiegroepen en hun werking op sets.

Burnside's lemma kijkt naar het aantal elementen wat invariant blijft onder iedere permutatie van een permutatiegroep. Een element blijft invariant onder een permutatie als het niet van plaats verandert door deze permutatie.

Voorbeeld 5.9. In de permutatie $\pi_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}$ blijft 1 invariant en de rest niet.

Voorbeeld 5.10. In de permutatie (in cykel notatie) $\pi_2 = (12)(3)(4)$ blijven 3 en 4 invariant en de rest niet.

Het aantal elementen wat invariant blijft onder een permutatie π wordt aangeduid met $Inv(\pi)$. In de twee voorbeelden $Inv(\pi_1) = 1$ en $Inv(\pi_2) = 2$.

Voorbeeldopgave 5.11. In een vorig voorbeeld had Zarah een kralenketting met 3 kralen. Hoeveel elementen zijn er invariant onder de permutatie die ontstaat als zij een kraal achter haar nek langs naar de andere kant van de kralenketting beweegt?

Hier zijn geen invariante elementen, de permutatie die hierbij hoort is:

$$\pi = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} = (123)$$

Deze bevat geen invariante elementen.

Dan zijn er nog twee sets die benoemd moeten worden. De eerste is een subgroep van een permutatiegroep, de stabalizer van een element a in A. Dit zijn alle π in G waarvoor geldt dat $\pi(a) = a$. De stabalizer van a bevat alle elementen waaronder a stabiel blijft.

Theorema 5.12. *Laat A een set en a een element uit die set. Laat G een permutatiegroep op A dan is de stabalizer van a, $St(a)$ een subgroep van G*

Bewijs. Voor dit bewijs checken we de eigenschappen van de groep:

- G1: G was al gesloten dus voor alle π_1, π_2 in $St(a)$ zit $\pi_1 \circ \pi_2$ in G en omdat a stabiel is onder beide π_1 en π_2 is deze ook stabiel in $\pi_1 \circ \pi_2$ wat dus ook een element is van de stabalizer van a. Hiermee is de geslotenheid van $St(a)$ bewezen.
- G2: De associativiteit van $St(a)$ volgt uit de associativiteit van G.
- G3: Het eenheidselement van G is ook het eenheidselement van $St(a)$ aangezien a invariant blijft onder de identiteit
- G4: alle π in $St(a)$ houden a stabiel dus de inverse van π moet a ook stabiel houden. Aangezien π een inverse had in G en de inverse a stabiel houdt zit deze inverse ook in $St(a)$.

□

Voorbeeld 5.13. In de permutatiegroep

$$G = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$$

is de stabalizer van 1 alleen de identiteit en de stabalizer van 2 de hele groep. Bij de tweede permutatie wordt 1 namelijk naar 3 verplaatst dus die permutatie maakt geen deel uit van de stabalizer van 1. 2 blijft onder beide permutaties stabiel en heeft dus de hele groep als zijn stabalizer.

De tweede set die benoemd moet worden is de orbit van een element a in de set A, geschreven als $C(a)$. Dit zijn alle elementen die met a in één equivalentieklasse zitten en a zelf. Dus alle b in A waarvoor er een π in G is zodat $\pi(a) = b$.

Hoe noem je 8 orbits? Een orbyte. (grap)

Voorbeeld 5.14. De orbit van 3 in voorbeeld 5.13 is $C(3) = \{1, 3\}$ omdat $\pi_2(3) = 1$.

Handig overzichtje van een set A met willekeurig element a en permutatiegroep G met willekeurig element π:

Notatie	Betekenis
$Inv(\pi)$	Het aantal elementen wat invariant blijft onder π, dat wil zeggen a waarvoor $\pi(a) = a$.
$St(a)$	De stabalizer van a, alle $\pi \in G$ waarvoor $\pi(a) = a$
$C(a)$	De orbit van a, alle b waarvoor er een π in G zit zodat $\pi(a) = b$ deze b vormen samen met a zelf de equivalentieklasse van a.

Theorema 5.15. ***Burnside's lemma****: Laat $A = \{1, 2, ..., n\}$ een set zijn met een groep van permutaties op die set $G = \{\pi_1, \pi_2, ..., \pi_m\}$ dan kan het totaal aantal equivalentieklassen in A onder G gevonden worden met de formule:*

$$\frac{1}{|G|} \sum_{\pi \in G} Inv(\pi)$$

Bewijs. Om te beginnen bewijzen we het volgende lemma:

Lemma 5.16. *Neem G een groep van permutaties op een set A en a een element van A dan $|St(a)| \cdot |C(a)| = |G|$*

Bewijs. Stel er is $C(a) = \{b_1, b_2, b_2, ..., b_r\}$, dan is er voor iedere b_i een π_i die a naar b_i stuurt. Mogelijk meer maar in ieder geval 1. Er is dan $P = \{\pi_1, \pi_2, ..., \pi_r\}$ een verzameling van een π_i per b_i. Merk op dat $|P| = |C(a)|$. Er zal nu aangetoond worden dat iedere permutatie in G gemaakt kan worden door samenvoeging van een permutatie in P en een permutatie in $St(a)$. Dan volgt door middel van de productregel dat $|G| = |P| \cdot |St(a)| = |C(a)| \cdot |St(a)|$. Gegeven een π in G merk op dat $\pi(a) = b_k$ voor zekere k. Dus $\pi(a) = \pi_k(a)$ en $\pi_k^{-1} \circ \pi$ houdt a invariant en is dus een element van $St(a)$. En dus:

$$\pi_k(\pi_k^{-1} \circ \pi) = (\pi_k \circ \pi_k^{-1}) \circ \pi = I \circ \pi = \pi$$

wat betekent dat π een product is van een element uit P en een element uit $St(a)$. Om de gelijkheid van het lemma te bewijzen dient nu nog bewezen te worden dat er geen twee combinaties van een element uit P en een element uit $St(a)$ zijn die dit effect bewerkstelligen. Stel $\pi = \pi_k\delta = \pi_l\gamma$, met δ, γ in $St(a)$. Dan $\pi_k\delta(a) = b_k$ en $\pi_l\gamma(a) = b_l$ en aangezien $\pi_k\delta = \pi_l\gamma$ moet ook $b_l = b_k$ wat alleen waar is als $l = k$ en dus $\pi_l = \pi_k$ en dat levert een tegenspraak. □

Dan zal nu Burnside's lemma bewezen worden. Het is bekend dat

$$Inv(\pi_1) + Inv(\pi_2) + ... + Inv(\pi_m) = St(1) + St(2) + St(3) + ... + St(n)$$

aangezien dit beide manieren zijn om te tellen hoeveel paren (π, a) er zijn waarvoor geldt $\pi(a) = a$. Verder volgt uit lemma 5.16 dat

$$\frac{1}{|G|}(Inv(\pi_1) + Inv(\pi_2) + ... + Inv(\pi_m)) = \frac{1}{C(1)} + \frac{1}{C(2)} + \frac{1}{C(3)} + ... + \frac{1}{C(n)}$$

Merk op dat x altijd in $C(x)$ zit en $C(x) = C(y)$ dan en slechts dan als x bevat is in $C(y)$. Dus als $C(x) = \{b_1, b_2, ..., b_k\}$ dan zijn er exact k equivalentieklasse

$C(b_1), C(b_2), ..., C(b_k)$ die gelijk zijn aan $C(x)$. Nu kunnen in de optelling deze equivalentieklasse in groepen opgedeelt worden exact zo dat iedere groep bestaat uit orbits die dezelfde equivalentieklasse omschrijven: $\frac{1}{C(b_1)} + \frac{1}{C(b_2)} + ... + \frac{1}{C(b_k)} = 1$. Dit levert het aantal equivalentieklasse in A geinduceerd door G op en daarmee is Burnside's lemma bewezen. □

Tot slot nog een voorbeeld opgave doorgaand op voorbeeldopgave 5.8.

Voorbeeldopgave 5.17. Bereken nogmaals het aantal verschillende kleurcombinaties die Zarah kan maken maar nu met behulp van Burnside's lemma.

Onder π_1 blijven alle 8 kleurcombinaties invariant. Onder π_2 en π_3 blijven de combinaties $\{rrr\}$ en $\{bbb\}$ invariant. $\frac{1}{|G|} \sum_{\pi \in G} Inv(\pi) = \frac{1}{3}(8 + 2 + 2) = 4$.

5.4 G*

Tot nu toe hebben zijn de permutaties genoteerd door aan te geven welke elementen waarnaartoe gingen. Permutaties kunnen ook genoteerd worden door niet de handeling maar het effect op te schrijven op de set waar we mee werken. Een permutatie zoals $\pi = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$ op een set van 3 elementen met 2 kleuren, rood en blauw) wordt dan $\begin{pmatrix} rrr & rrb & rbr & rbb & brr & brb & bbr & bbb \\ rrr & brr & rbr & bbr & rrb & brb & rbb & bbb \end{pmatrix}$. Hiervoor wordt de notatie π^* gebruikt. De groep van deze permutaties gebaseerd op de groep G wordt dan ook G^* genoemd en de bijbehorende equivalentierelatie S^*.

5.5 Polya's theorema

Deze paragraaf zal eerst uitleg geven over wat een cykel index is om daarna door te gaan naar het gebruiken van een cykel index in een pattern inventory. Een pattern inventory lijkt op een genererende functie in de zin dat er aan het einde uitgelezen kan worden hoeveel mogelijkheden er zijn voor een bepaalde verdeling in een polynoom.

Eerst zal een speciaal geval van Polya's theorema behandeld worden. Hiervoor moet eerst kort teruggeblikt worden naar de cykel notatie die in hoofdstuk 1 is behandeld. Hierbij werden permutaties omgeschreven naar cykels. Zo zou de permutatie $\begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$ in cykelnotatie (13)(2) zijn. Het aantal cykels in een permutatie π schrijven we als $cyc(\pi)$ in dit geval zouden dit er 2 zijn.

Theorema 5.18. *Het aantal equivalentieklasse in een groep A met grootte $|A| = m$ en permutatiegroep G is gelijk aan*

$$\frac{1}{|G|} \left(m^{cyc(\pi_1)} + m^{cyc(\pi_1)} + ... + m^{cyc(\pi_1)} \right)$$

Deze formule is zeker bij het berekenen van het aantal equivalentieklasse gegeven een G^* een stuk prettiger in het gebruik dan Burnside's lemma.

Voorbeeldopgave 5.19. Rixt heeft een ketting met 5 kralen en 3 kleuren, deze kralen kan ze doorschuiven maar ze kan de ketting ook omgekeerd omdoen, dit spiegelt de volgorde van de kralen. Hoeveel unieke kleurcombinaties kan Rixt hebben?

Het aantal unieke combinaties is gelijk aan het aantal equivalentieklasse zoals in paragraaf 2 uitgelegd werd. Om het antwoord te vinden worden dus alle cykels van de beschreven permutaties bepaald en de formule ingevuld. Eerst moeten alle permutaties bepaald worden.

$$\begin{aligned}
\pi_1 &= \begin{pmatrix}1&2&3&4&5\\1&2&3&4&5\end{pmatrix}, \pi_2 = \begin{pmatrix}1&2&3&4&5\\2&3&4&5&1\end{pmatrix}, \pi_3 = \begin{pmatrix}1&2&3&4&5\\3&4&5&1&2\end{pmatrix}\\
\pi_4 &= \begin{pmatrix}1&2&3&4&5\\4&5&1&2&3\end{pmatrix}, \pi_5 = \begin{pmatrix}1&2&3&4&5\\5&1&2&3&4\end{pmatrix}, \pi_6 = \begin{pmatrix}1&2&3&4&5\\5&4&3&2&1\end{pmatrix}\\
\pi_7 &= \begin{pmatrix}1&2&3&4&5\\4&3&2&1&5\end{pmatrix}, \pi_8 = \begin{pmatrix}1&2&3&4&5\\3&2&1&5&4\end{pmatrix}, \pi_9 = \begin{pmatrix}1&2&3&4&5\\2&1&5&4&3\end{pmatrix}\\
\pi_{10} &= \begin{pmatrix}1&2&3&4&5\\1&5&4&3&2\end{pmatrix}
\end{aligned} \tag{5.2}$$

En vervolgens ook de cykel notatie van al deze permutaties.

$\pi_1 = (1)(2)(3)(4)(5)$ $\pi_2 = (12345)$ $\pi_3 = (13524)$
$\pi_4 = (14253)$ $\pi_5 = (15432)$ $\pi_6 = (15)(24)(3)$
$\pi_7 = (14)(23)(5)$ $\pi_8 = (13)(2)(45)$ $\pi_9 = (12)(35)(4)$
$\pi_{10} = (1)(25)(34)$

Nu kan de formule ingevuld worden:

$$\frac{1}{10}(3^5 + 3^1 + 3^1 + 3^1 + 3^1 + 3^3 + 3^3 + 3^3 + 3^3 + 3^3) = 39$$

Dit leidt tot het eindantwoord 39.

Iedere permutatiegroep G heeft een cykel index. Deze kan bepaald worden door de permutaties om te zetten in cykel notatie en dan te noteren hoeveel cykels er van elke grootte zijn. Stel dat een permutatie π b_1 cykels van grootte 1, b_2 cykels van grootte twee tot b_n van grootte n heeft, dan wordt de cykel index van de hele groep $P_G(x_1, x_2, ..., x_n) = \frac{1}{|G|}\sum_{\pi \in G} x_1^{b_1} x_2^{b_2} ... x_n^{b_n}$. Hierbij is op te merken dat $P(m, m, m, , ..., m)$ exact dezelfde uitkomst geeft als de speciale variant van Polya's theorema aangezien $b_1 + b_2 + ... + b_n$ voor iedere π gelijk is aan $cyc(\pi)$.

Met deze formule kan een pattern inventory gemaakt worden. Deze pattern inventory neemt van iedere equivalentieklasse een vertegenwoordiger waarmee je de waarde van ieder element vermenigvuldigd. Dit zorgt ervoor dat er met behulp van het wiskundig uitwerken van de formule bepaald wordt hoeveel unieke varianten er zijn die gebruik maken van een bepaalde set kleuren. Dit doen we door voor $x_1, x_2, ..., x_n$ de kleuren in te vullen verheven tot de macht die bij de cykelgrootte hoort. Met de kleuren r en b wordt dit $P_G((r + b), (r^2 + b^2), ..., (r^n + b^n))$. Nu blijkt dat dit uitwerken exact het aantal unieke patronen geeft. Het geeft zelfs het aantal unieke patronen en de kleuren die in die patronen gebruikt worden. Ter illustratie een voorbeeld:

Voorbeeld 5.20. Rixt heeft de kleuren rood, blauw en groen. Hoeveel unieke kettingen kan ze maken met exact 3 blauwe kralen? Dit kan berekent worden met

behulp van de pattern inventory door de waarde $2 + b^n$ in te vullen voor x^n. De b staat voor de blauwe kralen en de 2 voor de twee andere opties, rode en groene kralen $r = g = 1$, waar verder geen rekening mee gehouden wordt. De volledige niet ingevulde formule kan gevonden worden uit de eerder bepaalde cykels en is:

$$\frac{1}{|G|}(x_1^5 + 4x_5 + 5x_2^2x_1)$$

Dit kan nu op de beschreven manier ingevuld worden.

$$\frac{1}{10}((2+b)^5 + 4(2+b^5) + 5(2+b^2)^2(2+b)$$

Dit uitwerken is veel moeite, gelukkig wordt alleen b^3 gezocht en kunnen er dus termen weggestreept worden. De term $4(2 + b^5)$ gaat nooit wat opleveren omdat deze alleen b^5 zal bevatten en bij het uitwerken van $5(2 + b^2)^2(2 + b)$ hoeven we ook alleen te kijken naar $5(2b^2)b$ want de andere uitwerkingen leveren geen b^3 op. Dit wordt dus $(2 + b)^5 + 10b^3$, gebruikmakend van het binomium van Newton kan de term van b^3 in $(2 + b)^5$ makkelijk gevonden worden. Dit wordt $\binom{5}{2}4b^3 = 40b^3$. In totaal zijn er dus 4 mogelijkheden.

5.6 Opgave

Opgave 54. Bepaal voor de volgende situaties of er spraken is van een equivalentierelatie als dit niet het geval is bepaal welke voorwaarde niet geldt:
a. Op de set van gehele getallen, aSb dan en slechts dan als $a = b + 7k$ met k een geheel getal.
b. Op de set van gehele getallen, aSb dan en slechts dan als $a = -b$.
c. Op de set van gehele getallen, aSb dan en slechts dan als $a = b + 4k$ met k een positief geheel getal.
d. Op alle mensen in de UU, aSb als het studentnummer van a met hetzelfde getal begint als het studentnummer van b.
e. Op alle mensen in de UU, aSb als het studentnummer van a deelbaar is door het studentnummer van b.
f. Op alle mensen in de UU, aSb als a een broer of zus is van b.

Opgave 55. Is de relatie over de mensen die het vak discrete wiskunde volgen en hetzelfde aantal tienen hebben een equivalentierelatie.

Opgave 56. Neem E_n het aantal mogelijke equivalentierelaties op een set $A = \{1, 2, 3, ..., n\}$ toon aan dat E_n voldoet aan de volgende recurrente betrekking:

$$E_n = \sum_{i=0}^{n-1} \binom{n-1}{i} E_i, n \geq 1$$

Opgave 57. Gebruik burnsides lemma om voor de volgende permutatiegroepen de hoeveelheid equivalentieklassen te vinden:
a. $A = \{1, 2, 3\}$ $G = \left\{ \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} \right\} \begin{Bmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{Bmatrix}$

b. $A = \{1,2,3,4,5\}$ $G = \left\{ \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix} \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 3 & 2 & 1 \end{pmatrix} \right\}$

c. $A = \{1,2,3,4,5,6\}$

$$G = \{ \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 5 & 4 & 3 & 2 & 1 \end{pmatrix}, \tag{5.3}$$

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 1 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 1 & 2 & 3 \end{pmatrix} \tag{5.4}$$

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 6 & 1 & 2 & 3 & 4 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 1 & 2 & 3 & 4 & 5 \end{pmatrix} \} \tag{5.5}$$

Opgave 58. Vind in de vorige opgave $St(2)$ en $C(2)$ voor alle deelopgaven.

Opgave 59. Gebruik burnsides lemma om te bepalen op hoeveel manieren koning Artuur en 4 andere ridders aan de ronde tafel kunnen zitten.

Opgave 60. Bekijk $A = \{1,2,3,4\}$
$G = \left\{ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 4 & 3 \end{pmatrix}, \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix} \right\}$
is $|St(1)||C(1)| = |G|$? Wat is hier gebeurt?

Opgave 61. Bereken in opgave 57a alsof de permutaties op een kralenketting met 2 mogelijke kleuren uitgevoerd worden: $G*$ $S*$

Opgave 62. Bereken voor alle deelopgaven in opgave 57 $cyc(\pi)$ voor alle permutaties. Bereken ook de cykel index.

Opgave 63. Bereken de cykel index van de permutatiegroep die een rubix cube omschrijft. Bereken het aantal mogelijke manieren om een rubix cube te kleuren met 2 kleuren. En met 5 kleuren als er op zijn minst 1 groen en 1 blauw vakje moet zijn?

Opgave 64. Gegeven is een ketting met $k = 8$ kralen; deze kralen kunnen drie kleuren krijgen: rood, blauw en wit. Op deze ketting kunnen twee soorten permutaties uitgevoerd worden: wel/niet omdraaien en met 0,2,4,6 plaatsen verschuiven.
a. Bepaal de groep van mogelijke permutaties en toon aan dat het een groep is (de eis van associativiteit mag genegeerd worden).
b. Geef aan hoe het aantal verschillende patronen (kleuringen) van deze ketting die niet door middel van de bij (a) gevonden permutaties in elkaar over te voeren zijn gevonden kan worden.
c. Gegeven de gewichten $w(rood) = r$, $w(blauw) = b$ en $w(wit) = w$, bepaal de pattern inventory (hierbij hoeft geen polynomen uitgewerkt te worden).
d. Bereken het aantal verschillende patronen (kleuringen) van deze ketting die niet door middel van de bij (a) gevonden permutaties in elkaar over te voeren zijn met precies 5 rode kralen.

5.7 Bronverantwoording

Burnside's lemma is een losse vertaling uit Applied Combinatorics(8.3.2, p.459)[2]. De volgorde van behandelen is ook (los) gebaseerd op dit hoofdstuk. De opgaven zijn een combinatie van opgaven van Han Hoogeveen (64). Enkele opgaven met hoge didactische waarden uit het boek Applied Combinatorics (60) en originele opgaven.

Hoofdstuk 6

NP-volledigheid

6.1 Berekensnelheden

Veel van de problemen tot nu toe vereisen een redelijk lange berekening die in de praktijk bij grotere problemen vaak door een computer gedaan wordt. Om te kijken of een probleem in redelijke tijd oplosbaar is is het daarom belangrijk om een concept te hebben van rekentijden. Een tijd in seconde is per computer verschillend aangezien snellere computers minder lang rekenen dan langzame computers, deze tijdsaanduiding is daarom erg onbetrouwbaar. Hierom wordt er gekeken naar de orde grootte van het aantal operaties dat de computer moet uitvoeren. Dit wordt dan vervolgens opgeschreven in zijn eigen ordegrootten notatie.

Voorbeeld 6.1. Een probleem waarbij n mensen n taken toebedeeld moeten krijgen op een optimale manier gedicteerd door een arbitraire voorwaarde wordt door een computer berekent door iedere optie te proberen. De orde grootte van de berekentijd is nu $n!$ aangezien er $n!$ mogelijkheden zijn om de mensen taken te geven. Iedere mogelijkheid moet natuurlijk door de computer doorgerekend worden om te bepalen wat de score is maar dit gebeurt altijd in een constante tijd en wordt daarom in de orde grootte niet meegenomen. Een bovengrens van de berekentijd is dus $n!$, dit wordt genoteerd als $O(n!)$, spreek uit "grote-O n-faculteit".

Op deze zelfde manier wordt een ondergrens van deze berekentijd genoteerd als $\Omega(n!)$ aangezien alle opties altijd geprobeerd worden en de computer dus altijd orde grootte $n!$ rekent. Als de onder en bovengrens aan elkaar gelijk zijn zoals in het voorbeeld wordt dit genoteerd als $\Theta(n!)$.

In wiskundige zin wordt deze notatie ook gebruikt maar dan om aan te geven dat een functie asymptotisch gedomineerd wordt door een andere functie, dat wil zeggen $f(x) = O(g(x))$ als $cg(x) \geq f(x)$ voor alle x vanaf een zekere x_0 met c een constante. Dit is ook de zin waarin de functie hier gebruikt wordt maar dan met rekentijd en zonder dat we de functie $f(x)$ echt weten, we weten alleen ongeveer hoeveel rekenstappen de computer er voor nodig zal hebben. Voor $\Omega(g(x))$ geldt dat $f(x) \leq c(g(x))$ met c een constante die niet nul is. En voor $\Theta(g(x))$ moet gelden dat $c_1(g(x)) \leq f(x) \leq c_2(g(x))$ met c_1, c_2 constanten groter dan 0.

Definitie 6.2. $f(x) = O(g(x)) \Leftrightarrow f(x) \leq cg(x), x > x_0 \in \mathbb{R}^+, c \in \mathbb{R}^+$

Definitie 6.3. $f(x) = \Omega(g(x)) \Leftrightarrow f(x) \geq cg(x), x > x_0 \in \mathbb{R}^+, c \in \mathbb{R}^+$

Definitie 6.4. $f(x) = \Theta(g(x)) \Leftrightarrow c_1 g(x) \leq f(x) \leq c_2 g(x), x > x_0 \in \mathbb{R}^+, c_1, c_2 \in \mathbb{R}^+$

Voorbeeldopgave 6.5. Jan is glazenwasser en probeert in een wijk in te delen welke glazen hij gaat wassen. De opbrengst van zijn dag hangt af de combinatie van huizen waarvoor hij glazen wast. Het kan dus zijn dat slechts een huis wassen veel meer oplevert dan allemaal. Hij schrijft een computerprogramma om alle opties langs te gaan. Er wordt uitgegaan van n huizen en het feit dat Jan alle huizen zou kunnen wassen. Druk dit probleem uit in O, Ω en Θ notatie.

Hierbij zijn er voor ieder huis 2 mogelijkheden, wel wassen of niet wassen. Er zijn dus 2^n gevallen die doorgerekend moeten worden wat ons een bovengrens voor de rekentijd geeft van $O(2^n)$, een ondergrens van $\Omega(2^n)$ en dus een rekentijd van $\Theta(2^n)$.

6.2 Opslag

Ook de hoeveelheid ruimte die er nodig is om iets op een harde schijf op te slaan kan uitgedrukt worden in grote-O notatie. Dit is handig omdat als de grootte van een probleem of oplossing uit de klauwen loopt het de computer erg veel tijd kan kosten om het alleen te lezen. Als er M dingen opgeslagen moeten worden kan dit in $O(M)$ ruimte. Hier moet wel opgelet worden dat de "dingen" een begrensde grootte hebben. Als er bijvoorbeeld M getallen opgeslagen moeten worden waarvan we geen grens op de grootte van het getal kunnen leggen is het $O(MN)$, waarbij N de hoeveelheid ruimte is die nodig is om het grootste getal op te slaan.

Een getal kan op twee manieren opgeslagen worden en afhankelijk daarvan kan de vorige bovengrens preciezer gedefinieerd worden. Unair, hierbij zijn er voor een getal n ook n bits nodig om het op te slaan. Het kan ook binair opgeslagen worden, hierbij zijn er voor een getal n slechts $log(n)$ bits nodig om het op te slaan. Meestal worden getallen binair opgeslagen dus dan wordt de afschatting van de hoeveelheid benodigde opslag om M getallen op te slaan van onbekende grootte $O(Mlog(n))$ met n de grootte van het grootste getal.

6.3 De klasse P en NP

De klasse P en NP zijn klassen van beslissingsproblemen. Een beslissingsprobleem is een probleem met een "ja" of "nee" antwoord. De klasse P bevat beslissingsproblemen die in polynomiale tijd oplosbaar zijn. Dit houdt in dat ze maximaal $O(n^c)$ met c een constante mogen zijn. De klasse NP bevat beslissingsproblemen die niet-deterministisch polynomiaal zijn. Dit houdt in dat ze in polynomiale tijd op te lossen zijn door een niet-deterministische Turing machine. Niet-deterministische Turing machines zijn computers die niet bestaan. Deze computers zouden een probleem op kunnen lossen door alle mogelijke oplossingen "tegelijk" te proberen. Tegelijk houdt hier in dat de machine iedere keer dat er een beslissing gemaakt moet worden beide opties uitvoert. En zo dus een binaire n diepe boom (een boom waarbij iedere vertakking twee kinderen heeft en waarbij de hoogste node $n-1$ kinderen onder zich

heeft) in n stappen doorloopt in plaats van in 2^n stappen. De klasse P is dus ook bevat in de klasse NP aangezien ieder probleem dat door een normale computer in polynomiale tijd opgelost kan worden ook door deze computers in polynomiale tijd opgelost kan worden.

Een beslissingsprobleem behoort tot de klasse NP als het aan twee voorwaarden voldoet. Een oplossing voor het probleem moet op te slaan zijn in polynomiale ruimte en het moet polynomiale tijd kosten om een oplossing te verifiëren.

Er zal nu gekeken worden naar de beslisvariant van het eerdere glazenwassersprobleem. In deze beslisvariant is de vraag of de glazenwasser een opbrengst van $\geq k$ kan bewerkstelligen. Hier is een "ja" of "nee" antwoord op te geven. Een "ja" oplossing voor dit probleem is een manier $\geq k$ opbrengst te verkrijgen.

De eerste vereiste, hoeveel ruimte het kost om een oplossing op te slaan kan beredeneerd worden door te kijken naar hoeveel informatie er opgeslagen moet worden. Een oplossing heeft data die deze oplossing codeert. Een oplossing voor het glazenwassersprobleem zou bevatten welke huizen er gewassen moeten worden en heeft dus $O(m)$ ruimte nodig. De maximale grootte is namelijk alle huizen wat van orde m is.

De tweede vereiste, de tijd die het kost om een oplossing verifieren. Voor het glazenwassersprobleem kijken we naar de oplossing en bepalen hoeveel opbrengst het oplevert. Dit kan in $O(m)$ tijd omdat er maximaal m huizen in de oplossing zitten. Dit probleem is dus NP aangezien beide de opslaggrootte en de rekentijd voor het verifieren van een oplossing polynomiaal zijn.

6.4 NP-volledigheid

De klasse NP bevat veel problemen. Een deel daarvan zit in een deelverzameling van NP volledige problemen. Dit zijn problemen waarvoor een reductie bestaat naar ieder ander probleem in de klasse NP. Deze reducties zullen later in deze paragraaf aan bod komen. Als dit probleem dus opgelost kan worden in polynomiale tijd kunnen alle problemen in de klasse NP opgelost worden in polymomiale tijd en is $P = NP$ bewezen.

In 1971 bewees het Cook-Levin theorem dat satisfyability een NP-volledig probleem is. Dit door aan te tonen dat alle andere problemen in de klasse NP opgelost kunnen worden door dit probleem op te lossen. Dit is een zware taak en ook niet iets waar deze tekst zich mee bezig houdt. Alle bewijzen tijdens dit vak zullen een andere vorm hebben.

Het bewijzen dat een probleem A NP-volledig is kan ook door aan te tonen dat een instantie van A in polynomiale tijd omgeschreven kan worden naar een instantie van een al NP-volledig probleem B. In dit dictaat staan enkele problemen die NP-volledig zijn als voorbeeldproblemen om vanuit te werken. Een bewijs van deze

vorm heet een reductie. Als we weten dat B een NP-volledig probleem is bewijzen we dat A dit ook is door een reductie uit te voeren van A naar B. Bij een reductie wordt een willekeurige variant van het probleem B als een speciale variant van het probleem A geschreven. Door dit te doen wordt aangetoond dat A minstens net zo moeilijk is als B. Een reductie moet wel te doen zijn binnen polynomiale tijd, anders is het algoritme voor probleem B niet goed genoeg omdat probleem A dan nog steeds meer dan polynomiale tijd kost.

Een reductie bewijs waarbij A tot B gereduceerd wordt gebeurt concreet in de volgende stappen:

- Neem een willekeurige variant van probleem B.
- Construeer een speciale instantie van het probleem A waarbij het antwoord op dat probleem "ja" is dan en slechts dan als het antwoord op de willekeurige variant van probleem B ook "ja" is.
- Controleer dat deze constructie in minder dan polynomiale tijd kan.

Dan is bewezen dat A tot de klasse van NP-volledige problemen behoort. Als er namelijk een polynomiaal algoritme voor probleem A zou bestaan dan zou met deze reductie dat algoritme ook gebruikt kunnen worden om B in polynomiale tijd op te lossen. Dit is opnieuw de reden dat de reductie in polynomiale tijd moet, anders zou een polynomiaal algoritme voor A nog geen polynomiale oplossing voor B geven omdat de reductie meer dan polynomiale tijd vereist.
Een voorbeeld:

Voorbeeld 6.6. In het traveling-salesman probleem moet een verkoper langs n steden eindigend in de beginstad. Het probleem wordt gegeven door een complete graaf, een graaf met een kant van ieder punt naar ieder punt, met n punten. Aan iedere kant van de graaf is een reistijd verbonden. De "ja" oplossing van de beslissingsvariant is een tour, een pad dat ieder punt een keer bezoekt en eindigt in het eerste punt, die minder dan of gelijk aan een gegeven k aan reistijd met zich meebrengt. Om te bewijzen dat dit probleem NP-volledig is wordt eerst bewezen dat het tot de klasse NP behoort. Een oplossing is te verifiëren door te kijken of ieder punt exact een keer bezocht wordt, de reistijden bij elkaar op te tellen en te kijken of dit kleiner dan gelijk aan een gegeven k is. Dit is $O(n)$ aangezien er n punten gecheckt worden, n waardes bij elkaar opgeteld worden en een vergelijking gedaan wordt. Dan wordt er een reductie bewijs gemaakt vanuit het hamiltonian cykel probleem:
Het HAMILTONCYKEL probleem is alsvolgt gedefinieerd: Gegeven een **willekeurige samenhangende** graaf $G = (V, E)$, bevat deze graaf een tour die ieder punt $v \in V$ precies éénmaal bezoekt?
Deze reductie wordt gedaan door in de TSP graaf alle kanten die ook in de Hamiltonian cykel graaf zitten een kosten van 0 te geven en alle andere een kosten van 1. Als er nu gekeken wordt of er een oplossing is met reistijd 0, betekent dat dat er ook een Hamiltonian cykel zit in de eerste graaf. Immers de oplossing met reistijd 0 is die tour. Deze reductie kan in n stappen omdat er maximaal zoveel kanten als in de Hamiltonian cykel zitten een waarde van 0 krijgen en de rest standaard 1 gemaakt kan worden. Dit is in polynomiale tijd en dus is TSP een NP-volledig probleem

Voorbeeldopgave 6.7. Bewijs dat het probleem partitie NP-compleet is door een reductie vanuit subset sum. En andersom.
Het probleem PARTITIE, dat als volgt is gedefinieerd: Gegeven n niet-negatieve gehele getallen $a_1, \ldots, a_n$, bestaat er een deelverzameling S van de indexverzameling $\{1, \ldots, n\}$ waarvoor geldt

$$\sum_{j \in S} a_j = (\sum_{j=1}^{n} a_j)/2?$$

Het probleem SUBSET SUM is als volgt gedefinieerd: gegeven t niet-negatieve gehele getallen $a_1, \ldots, a_n$ en gegeven een niet-negatief geheel getal B, bestaat er een deelverzameling S van de indexverzameling $\{1, 2, \ldots, n\}$ waarvoor geldt

$$\sum_{j \in S} a_j = B?$$

De reductie van SUBSETSUM naar PARTITIE gaat alsvolgt: Door een subset grootte van exact de helft en de elementen uit de verzameling hetzelfde te nemen is dit probleem in minder dan polynomiale tijd gereduceerd. Bij partitie moet de set exact door twee te delen zijn en bij subset sum moet er een subset zijn die in het totaal een som van een bepaalde waarde heeft. Als deze waarde exact de helft is is dit hetzelfde als een partitie probleem.
De andere kant op: Voor een willekeurige instantie van SUBSET SUM wordt bij het partitie probleem dezelfde elementen als bij het subset sum probleem gebruikt met een extra element a_0. Dit element wordt exact zo groot gekozen dat de helft van de totale som gelijk is aan $a_0 + B$. Als hier een oplossing voor bestaat, betekent dit dat zonder het toegevoegde element a_0 de subset B gemaakt kan worden. Deze reductie is polynomiaal aangezien er slechts een element toegevoegd wordt.

6.5 NP-Lastig

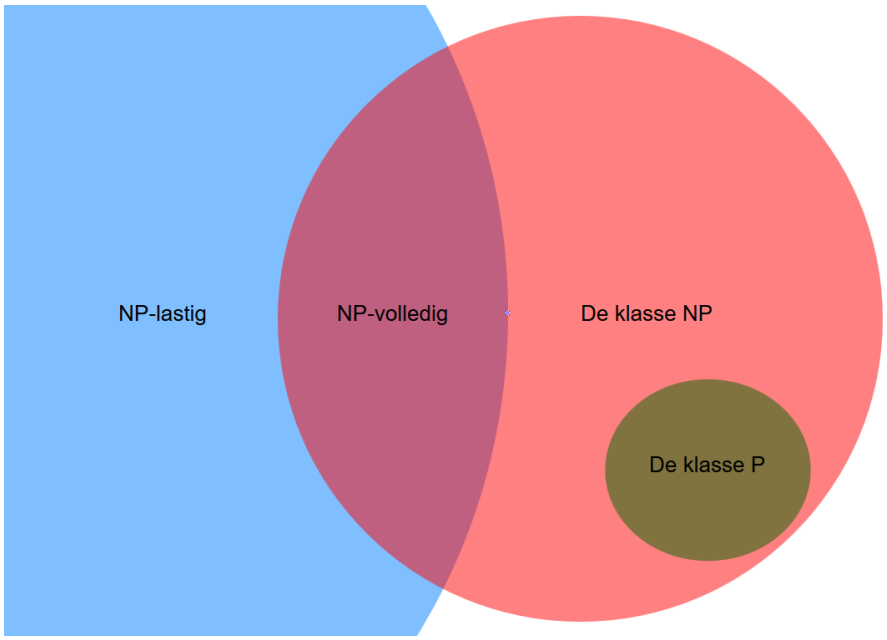
Naast NP-volledig zijn er ook nog problemen die NP-lastig zijn. Dit zijn problemen die net als NP-volledige problemen gebruikt kunnen worden om alle problemen in de klasse NP op te lossen. Alle NP volledige problemen zijn dus ook NP-lastig. Het kan ook voorkomen dat een probleem niet in de klasse NP zit maar wel NP-lastig is. Voor het bewijzen dat een probleem NP-lastig is wordt vaak van de beslissingsvariant gebruik gemaakt. Dit omdat een probleem geen beslissingsprobleem hoeft te zijn en het bewijzen dat het beslissingsprobleem NP-compleet is ook bewijst dat het orginele probleem NP lastig is.

Voorbeeld 6.8. Er is een optimalisatieprobleem waarbij opbrengst geoptimaliseerd moet worden. De beslissingsvariant is nu, bestaat er een oplossing zodat er meer dan x opbrengst is. Als dit probleem NP-volledig is dan is het orginele probleem NP-lastig want het vinden van de optimale oplossing is ook voldoende om te vinden of er meer dan x opbrengst verkregen kan worden.

Voorbeeldopgave 6.9. Bewijs dat het optimalisatieprobleem van het TSP, dus het vinden van een zo laag mogelijke reistijd, NP-lastig is.

Hierbij maken we gebruik van het feit dat de beslissingsvariant al bewezen NP-compleet is in de vorige paragraaf en met het vorige voorbeeld impliceert dit dat het probleem NP-lastig is.

Het volgende plaatje laat zien hoe de relaties tussen alle klasse zijn **gegeven dat** $P \neq NP$ als echter blijkt dat $P = NP$ dan zouden de klasse P en NP volledig overlappen.



Figuur 6.1: De relaties tussen alle klasse in een overzichtelijk plaatje

6.6 NP-volledige voorbeeldproblemen

- Het probleem PARTITIE, dat als volgt is gedefinieerd: Gegeven n niet-negatieve gehele getallen $a_1, \ldots, a_n$, bestaat er een deelverzameling S van de indexverzameling $\{1, \ldots, n\}$ waarvoor geldt

$$\sum_{j \in S} a_j = (\sum_{j=1}^{n} a_j)/2?$$

- Het probleem GEHEELTALLIGE LINEAIRE PROGRAMMERING IN BINAIRE VARIABELEN, dat als volgt is gedefinieerd: Gegeven een geheeltallige $m \times n$ matrix A en een geheeltallige $m \times 1$ vector b, bestaat er een vector $x = (x_1, \ldots, x_n)^T$ met $Ax \leq b$ en $x_j \in \{0, 1\}$ voor alle j?

- Het HAMILTONCYKEL probleem dat als volgt is gedefinieerd: Gegeven een **willekeurige samenhangende** graaf $G = (V, E)$, bevat deze graaf een tour die ieder punt $v \in V$ precies éénmaal bezoekt?

- Het VERTEX COVER probleem, dat als volgt is gedefinieerd: Gegeven een graaf $G = (V, E)$ en een natuurlijk getal k, bevat G een verzameling U van ten hoogste k punten zodat elke kant in E tenminste één punt uit U bevat?

- Het SATISFIABILITY probleem, dat als volgt is gedefinieerd: Gegeven n Boolean variabelen $x_1, \ldots, x_n$ en m 'clauses' $C_1, \ldots, C_m$, die deze variabelen bevatten, is de formule $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ satisfiable, dus bestaat er een toekenning van de waarden True en False aan $x_1, \ldots, x_n$ zodanig dat iedere clause True is?

  Een clause bevat een aantal x_j's of de ontkenning daarvan; een voorbeeld van een clause is $(x_1 \vee x_3 \vee \bar{x}_6)$; hierbij geeft $\bar{x}_6$ de ontkenning van x_6 aan.

6.7 Opgave

Opgave 65. Beschouw het volgende machinevolgordeprobleem, dat we zullen noteren als $P2$. Gegeven zijn twee machines M_1 en M_2 die beschikbaar zijn vanaf tijdstip 0 en ieder ten hoogste één taak tegelijkertijd kunnen uitvoeren. Verder zijn gegeven m taken T_j met bewerkingsduren p_j $(j = 1, \ldots, m)$. Gevraagd wordt om een toewijzing van de taken aan M_1 en M_2 te vinden zodanig dat het tijdstip waarop de laatste machine klaar is met de verwerking van de taken minimaal is.

a. Bepaal de inputgrootte van $P2$.

b. Formuleer de beslissingsvariant $BVP2$ van $P2$ en bewijs dat deze tot de klasse $\mathcal{NP}$ behoort.

c. Bewijs dat $BVP2$ $\mathcal{NP}$-volledig is door middel van een reductie vanuit PARTITIE waarvan bekend is dat het $\mathcal{NP}$-volledig is.

Het probleem PARTITIE is als volgt gedefinieerd: Gegeven n niet-negatieve gehele getallen $a_1, \ldots, a_n$, bestaat er een deelverzameling S van de indexverzameling $\{1, \ldots, n\}$ waarvoor geldt

$$\sum_{j \in S} a_j = (\sum_{j=1}^{n} a_j)/2?$$

d. Voeg een extra machine M_3 toe die beschikbaar is vanaf tijdstip 0 en ten hoogste één taak tegelijkertijd kan uitvoeren; dit geeft probleem $P3$.

Bewijs dat het probleem $P3$ $\mathcal{NP}$-lastig is door middel van een reductie vanuit $BVP2$.

e. Beschouw de volgende variant op $P2$. Aan de machines M_1 en M_2 wordt een positieve snelheid, s_1 en s_2, toegekend: de verwerking van T_j duurt nu tijd p_j/s_1 op M_1 of p_j/s_2 op M_2. Dit probleem wordt genoteerd als $Q2$.

Bewijs dat $Q2$ $\mathcal{NP}$-lastig is door middel van een reductie vanuit $BVP2$.

Opgave 66. Het HANDELSREIZIGERSPROBLEEM is als volgt gedefinieerd: Gegeven een **volledige** graaf $G = (V, E)$ en een afstand c_e voor iedere kant $e \in E$, bepaal een tour van minimale lengte die ieder punt $v \in V$ precies éénmaal bezoekt. Het 0-1 HANDELSREIZIGERSPROBLEEM is gedefinieerd als de versie van het HANDELSREIZIGERSPROBLEEM waarbij geldt dat iedere afstand c_e gelijk is aan 0 of 1.

a. Bewijs dat de beslissingsvariant van het 0-1 HANDELSREIZIGERSPROBLEEM tot de klasse $\mathcal{NP}$ behoort.

b. Bewijs dat het 0-1 HANDELSREIZIGERSPROBLEEM $\mathcal{NP}$-lastig is door middel

van een reductie vanuit het HAMILTONCYKEL probleem, waarvan bekend is dat het $\mathcal{NP}$-compleet is. Het HAMILTONCYKEL probleem is als volgt gedefinieerd: Gegeven een **willekeurige samenhangende** graaf $G = (V, E)$, bevat deze graaf een tour die ieder punt $v \in V$ precies éénmaal bezoekt?

c. Bewijs dat het bestaan van een polynomiaal ρ-approximatie algoritme voor het 0-1 HANDELSREIZIGERSPROBLEEM, waarbij ρ een willekeurige constante > 1 is, impliceert dat de beslissingsvariant van het 0-1 HANDELSREIZIGERSPROBLEEM in polynomiale tijd beslisbaar is.

Een polynomiaal ρ-approximatie algoritme is gedefinieerd als een algoritme dat voor iedere instantie van het probleem in polynomiale tijd een oplossing vindt met een waarde die ten hoogste ρ maal zo groot is als de waarde van een optimale oplossing.

6.8 Extra opgaves

Opgave 67. Het HAMILTONCYKEL probleem is als volgt gedefinieerd: Gegeven een **willekeurige samenhangende** graaf $G = (V, E)$, bevat deze graaf een tour die ieder punt $v \in V$ precies éénmaal bezoekt?

Het HAMILTONPAD probleem is als volgt gedefinieerd: Gegeven een **willekeurige samenhangende** graaf $G' = (V', E')$, bevat deze graaf een pad dat ieder punt $v \in V'$ precies éénmaal bezoekt? Je moet dus in het Hamiltonpad probleem beginnen in een punt, alle punten bezoeken, maar dan hoef je niet terug te gaan naar je beginpunt; het beginpunt mag je dus zelf kiezen.

a. Toon aan dat het HAMILTONPAD probleem $\mathcal{NP}$-volledig is door middel van een reductie vanuit het HAMILTONCYKEL probleem.

b.Toon aan dat het HAMILTONCYKEL probleem $\mathcal{NP}$-volledig is door middel van een reductie vanuit het HAMILTONPAD probleem.

Opgave 68. Het probleem SUBSET SUM is als volgt gedefinieerd: gegeven t niet-negatieve gehele getallen $a_1, \ldots, a_n$ en een niet-negatief geheel getal B, bestaat er een deelverzameling S van de indexverzameling $\{1, 2, \ldots, n\}$ waarvoor geldt

$$\sum_{j \in S} a_j = B?$$

Het is bekend dat SUBSET SUM $\mathcal{NP}$-compleet is.

Het probleem PARTITIE wordt als volgt gedefinieerd: gegeven n niet-negatieve gehele getallen $a_1, \ldots, a_t$, bestaat er een deelverzameling S van de indexverzameling $\{1, 2, \ldots, t\}$ waarvoor geldt

$$\sum_{j \in S} a_j = (\sum_{j \in \bar{S}} a_j)?$$

De deelverzameling $\bar{S}$ is hierbij gedefinieerd als het complement van S ten opzichte van de volledige indexverzameling.

a. Bewijs dat het probleem PARTITIE tot de klasse $\mathcal{NP}$ behoort.

b. Bewijs met behulp van een reductie vanuit het probleem SUBSET SUM dat het probleem PARTITIE $\mathcal{NP}$-compleet is.

c. Stel dat het aantal getallen t in de instantie van PARTITIE even is. Bewijs dat het probleem PARTITIE $\mathcal{NP}$-compleet blijft als de eis wordt toegevoegd dat de deelverzameling S precies $t/2$ elementen dient te bevatten.

(d) Het probleem EVEN-ONEVEN PARTITIE wordt als volgt gedefinieerd: gegeven een verzameling A van $2r$ niet-negatieve gehele getallen $\{a_1, \ldots, a_{2r}\}$ met $a_i \geq a_{i+1}$ $(i = 1, \ldots, 2r-1)$, bestaat er een deelverzameling S van de indexverzameling $\{1, 2, \ldots, 2r\}$ zodanig dat $\sum_{j \in S} a_j = \sum_{j \in \bar{S}} a_j$, waarbij S precies één element van $\{2i-1, 2i\}$ bevat voor iedere $i = 1, \ldots, r$?

Bewijs dat het probleem EVEN-ONEVEN PARTITIE $\mathcal{NP}$-compleet is.

Opgave 69. Het GARAGE PROBLEEM wordt als volgt gedefinieerd: Er is één monteur die n auto's moet repareren. Voor iedere auto j is gegeven hoeveel tijd die reparatie kost; noem deze tijdsduur p_j $(j = 1, \ldots, n)$. Verder is het tijdstip d_j gegeven waarop auto j $(j = 1, \ldots, n)$ klaar moet zijn (dit is een strikte, keiharde deadline). De monteur begint op tijdstip 0, werkt continu door zonder pauze, en kan hoogstens één auto tegelijkertijd repareren. Wanneer de monteur met de reparatie van een auto is begonnen, dan moet hij die afmaken voordat aan de reparatie van een andere auto begonnen kan worden. De vraag is of er een reparatievolgorde bestaat zodanig dat iedere auto op tijd klaar is.

a. Beschouw de reparatievolgorde waarin de auto's in volgorde van niet-afnemende d_j worden gerepareerd. Bewijs dat dit tot een toegelaten oplossing leidt, als die bestaat.

b. Uiteraard zijn er situaties waarin een auto pas wordt binnengebracht na tijdstip 0, maar ook die auto moet dan op tijd worden gerepareerd. Toon aan dat de variant van het GARAGE PROBLEEM waarin één auto na tijdstip 0 wordt binnengebracht $\mathcal{NP}$-compleet is. U mag hierbij gebruiken dat de problemen van vraag 2 $\mathcal{NP}$-compleet zijn.

Opgave 70. Beschouw het volgende machine probleem. Gegeven zijn m machines die m taken moeten uitvoeren. Iedere taak bestaat uit m deeltaken; in iedere taak is er precies één deeltaak die door machine i $(i = 1, \ldots, m)$ moet worden uitgevoerd. Van iedere deeltaak is gegeven door welke machine deze moet worden uitgevoerd; de bewerking van een deeltaak duurt altijd 1 tijdseenheid, ongeacht de deeltaak.

a. Stel dat de deeltaken van een taak in iedere willekeurige volgorde kunnen worden uitgevoerd. Construeer een oplossing om al het werk in m tijdseenheden uit te voeren.

b. Stel nu dat de deeltaken in een gegeven volgorde moeten worden uitgevoerd; dit legt de route van de taken door de machines vast. Ga na aan welke voorwaarden voldaan moet zijn om een oplossing te vinden waarbij al het werk in m tijdseenheden uitgevoerd kan worden.

Opgave 71. Van een gegeven verzameling van n^2 personen is bekend dat er 2 mensen een afwijkende bloedgroep hebben. Om uit te zoeken wie het zijn wordt bij iedere persoon een beetje bloed afgenomen. Vervolgens worden er bloedtesten uitgevoerd: het bloed van een verzameling personen wordt bij elkaar gevoegd, waarna wordt getest of er iemand in de groep die afwijkende bloedgroep heeft (maar het is dan niet bekend wie dat is). Geef een constructie waarbij je in maximaal $3n$ testen kunt uitzoeken welke twee personen de afwijkende bloedgroep hebben. De testen moeten gelijktijdig worden uitgevoerd.

6.9 Bronverantwoording

Dit hoofdstuk bevat enkele versimpelde definities uit "Introduction to algorithms"[3], daar wordt het concept "languages" gebruikt om een groot deel van dit hoofdstuk uit te leggen. Omdat dit geen onderdeel is van dit vak wordt dat hier niet uitgelegd of behandeld. Ook het TSP voorbeeld probleem komt hier vandaan. De opgaven zijn aangereikt door Han Hoogeveen[1].

Hoofdstuk 7

Dynamisch programmeren

7.1 Introductie

Bij dynamisch programmeren(DP) gaat het om een het reduceren van de rekentijd door een techniek die gebruik maakt van memoization en recurrente berekeningen. Een DP oplossing bestaat altijd uit twee delen, een initialisatie en een recurrente berekening. Het memoization deel houdt in dat waardes die twee keer nodig zijn niet twee keer berekend worden. Het is memoization en niet memorization omdat er waardes opgeslagen worden om snel terug te kunnen kijken en niet onthouden worden als het probleem eenmaal opgelost is.

Een klassiek voorbeeld is het staaf snij probleem. Een bedrijf snijdt staven in stukken om die vervolgens door te verkopen. Snijden is gratis maar staven van verschillende lengtes leveren verschillende prijzen op. Het doel is om zo veel mogelijk te verdienen aan een staaf van lengte n gegeven prijzen p_i met i de lengte van de te verkopen staaf. Iedere mogelijke staaflengte kan verkocht worden ze leveren echter niet allemaal hetzelfde op. Het kan dus ook zo zijn dat het het meest voordelig is om de staaf niet te snijden.
De meest voor de hand liggende manier om dit probleem te tackelen is door iedere combinatie van lengtes uit te proberen, dit kost $O(2^{n-1})$ tijd aangezien iedere staaf op 2^{n-1} manieren gesneden kan worden. Bij DP wordt dit ook gedaan maar dan op een manier die de hoeveelheid rekenwerk significant vermindert. Door memoization te gebruiken worden de optimale waardes van een staaf in iedere lengte opgeslagen. Een array, simpel gezegd een matrix met maar een rij, bevat op plaats t de maximale opbrengst voor een staaf van lengte t. Deze wordt dan van achter naar voor of van voor naar achter gevuld. In dit voorbeeld wordt hij van voor naar achter gevuld. Eerst wordt de waarde voor lengte 1 berekend. Dit is p_1. De waarde voor lengte 2 is het maximum van p_2 en $p_1 + a_1$ waarin a_1 de eerste waarde in de array is. Voor a_n is het het maximum van p_n, $a_{n-1} + p_1$, $a_{n-2} + p_2$, ..., $a_1 + p_{n-1}$. Dit geeft een looptijd van $O(n^2)$ wat een groot verschil is met $O(2^{n-1})$!.

Tot zover het idee, een volledig correcte oplossing bestaat zoals eerder genoemd uit een initialisatie en een recurrente berekening. In dit geval is de initialisatie een n grootte array met alle waardes op $-\infty$. Dit zodat het algoritme weet dat het deze waarde nog moet berekenen, dit kan niet door de waarde op 0 of een negatief getal te zetten omdat het bedrijf bij verkeerd snijden misschien wel verlies lijdt. En de

0 plaats in de array als winst 0. Dan met deze methode wordt voor het berekenen van a_n met n de lengte van de staaf de volgende recurrente formule gegeven:

$$a_n = max_{0 \leq k < n}(p_k + a_{n-k})$$

Als laatste wordt nog opgemerkt dat er bottom-up gewerkt wordt wat betekent dat de laagste waarde eerst uitgerekend wordt, daarna de waarde erboven enzovoorts. Een andere mogelijkheid is om top-down te werken waarbij direct de hoogste waarde gevraagd wordt en het algoritme dan recurrent naar onder gaat rekenen. Beide manieren van doorlopen zijn valide maar bij de tweede manier moeten alle waardes die opgevraagd worden nog steeds uitgerekend worden en moet er wel gezorgd worden dat er geen dingen dubbel berekent worden.

Om nu te weten te komen wat de optimale manier van knippen is (dus niet alleen de optimale opbrengst) moet de array nog een keer doorlopen worden. Hierbij wordt voor iedere waarde uitgerekend van welke optie deze waarde kwam. Als het een prijs is wordt in een nieuwe array gezet dat er niet meer geknipt wordt en als het een knip is wordt een van de twee stukken genoteerd waar het in geknipt wordt (het andere stuk is immers af te leiden). En hiermee kan dan bepaald worden welke lengtes geknipt moeten worden om de optimale oplossing te krijgen.

7.2 Toestandsvariabelen

Bij Dynamisch programmeren wordt een probleem in stappen opgelost. In ieder van deze stappen wordt een optimale oplossing gezocht voor iedere toestand die in die stap zit. Hierbij wordt gebruik gemaakt van toestandsvariabelen die alle informatie bevatten voor het berekenen van de volgende toestandsvariabelen, ongeacht hoe de deeloplossing er uit ziet die correspondeert met de vorige toestandsvariabelen. In het voorbeeld aan het begin van dit hoofdstuk maakt het niet uit op welke manier de rest van de staaf gesneden wordt er wordt alleen gekeken naar het overgebleven stuk wat nu nog gesneden moet worden. Hoe de rest van de staaf gesneden gaat worden zit al in een toestandsvariabelen opgeslagen.

Naast het vinden van de waarde die bij de optimale oplossing hoort moet vaak ook nog berekent kunnen worden wat de optimale oplossing is. Dit kan vaak weer met een toestandsvariabelen gedaan worden met behulp van de al uitgerekende waardes. Zoals in het staafsnijdprobleem hierboven gebeurt met het uitvinden welke lengtes geknipt moeten worden.
Als de opgave van het vorige hoofdstuk gemaakt zijn is het knapzak probleem reeds bekend. Er wordt nu gekeken naar een iets lastigere vorm van het knapzakprobleem.

Voorbeeldopgave 7.1. In een knapzak moet zoveel mogelijk waarde aan spullen worden gestopt. De knapzak heeft een maximale hoeveelheid inhoud die hij kan hebben B en een maximaal gewicht wat er in kan zitten C. Gegeven voorwerpen a_i met grootte b_i, gewichten c_i en opbrengst d_i geef een DP algoritme wat de maximale opbrengst kan bereken.

Dit probleem is later ook een opgave. Hieronder staat een grove lijn voor hoe dit soort problemen opgelost moeten worden maar het is aan te raden eerst zelf de

opgave te proberen. Bij het vorige probleem was de "grondstof"die gebruikt werd de staaflengte. Nu zijn er twee "grondstoffen"namelijk de grootte en het gewicht. Een twee dimensionale array brengt het antwoord. Op de ene as de grootte en op de andere as het gewicht. Dan kan er door alle objecten heen gelopen worden om de array te vullen. Hoe dit exact gedaan moet worden om de optimale oplossing te vinden wordt aan de lezer overgelaten in de opgave paragraaf.

7.3 Behandelvolgorde

De tot nu toe behandelde problemen hebben geen vereiste aan de behandelvolgorde. Het maakt niet uit hoe de items in de knapzak genummerd worden en bij het staafsnijdprobleem is het voor de hand liggend. Het kan echter zo zijn dat een probleem een vaste behandelvolgorde heeft waar bij gebleven moet worden. Een goed voorbeeld is het volgende machinevolgorde probleem:

Voorbeeld 7.2. Twee machines moeten n taken uitvoeren. Deze taken $1, ..., n$ hebben allemaal een gewicht w_j en een tijdsduur p_j. Iedere taak moet aan een machine toegewezen worden en een completeringstijd C_j krijgen. Het opslaan van de machine waar een taak op uitgevoerd wordt is niet nodig. De machines zijn identiek en aan de hand van de tijdsduur van een taak en de completeringstijd kan terug geredeneerd worden om te bepalen welke machine deze taak uit moet voeren. De bedoeling is de kostenfunctie $\sum_{i=1}^{n} w_i C_i$ te minimaliseren. Het is gemakkelijk aan te tonen met behulp van een omwisselingsargument dat het niet nadelig kan zijn om alle taken te behandelen in volgorde van dalende $\frac{w_j}{p_j}$ de taak met de kortste tijdsduur en hoogste gewicht komt dan eerst wat ook voor de grootste vermindering in kosten zorgt. Het is ook zeker dat een machine altijd aan het werk is totdat iedere taak uitgevoerd is of bezig is. Hierdoor is het overgebleven probleem alleen nog maar welke machine welke taak doet aangezien de completeringstijd daar uit volgt als de machines de hun toegewezen taken in dalende $\frac{w_j}{p_j}$ doen en nooit stilstaan. In de toestandsvariabelen zal echter met de completeringstijd gewerkt worden omdat hiermee ook de laagste kosten direct uitgerekend kan worden.
Het idee is nu dat voor iedere set al toegevoegde taken j de taak $j+1$ toegevoegd wordt op de goedkoopste manier. Hiervoor wordt de toestandsvariabelen $f_j(t)$ introduceert die aangeeft wat de minimale kosten zijn als j taken uitgevoerd worden zodat machine een exact op tijdstip t klaar is. De initialisatie is nu als volgt:

$$f_j(t) = \begin{cases} 0 & j = t = 0 \\ \infty & \text{anders} \end{cases}$$

Aangezien de tijd die machine 1 aan taken besteed bekend is en alle taken tot nu toe een vaste tijdsduur hebben kan ook de tijd die machine 2 al bezig is berekend worden. Hierna kan met de volgende functie de optimale manier gevonden worden om taak $j+1$ toe te voegen aan het probleem waarbij $P(j) = \sum_{i=1}^{n} p_i$ is (de tijd die alle taken tot nu toe kosten):

$$f_{j+1}(t) = \min\{f_j(t) + w_{j+1}(P(j+1) - t), f_j(t - p_{j+1}) + w_{j+1}t\}$$

Dit is een minimalisatieprobleem omdat de kosten zo laag mogelijk moeten zijn. De eerste term correspondeert met de keuze om taak $j+1$ op machine 2 te zetten, de

tweede term voor machine 1. Uiteindelijk moet $\min \sum_{i=0}^{P(n)} f_n(i)$ bepaald worden om de optimale oplossingswaarde te vinden. Door terug te zoeken kan bepaald worden of de laatste taak op machine 1 of machine 2 uitgevoerd moet worden. In het totaal worden $nP(n)$ toestandsvariabelen uitgerekend en dus is de rekentijd $O(nP(n))$.

In dit probleem moet er rekening gehouden worden met het feit dat de taken altijd in dezelfde volgorde uitgevoerd worden voor een optimale oplossing anders is het probleem niet te doen binnen aanzienlijke tijd.

7.4 Opgave

Opgave 72. Beschouw het volgende twee-dimensionale knapzak probleem. Er zijn n items; ieder item kun je maximaal één maal meenemen. Ieder item i ($i = 1, \ldots, n$) heeft een positief, geheeltallig gewicht a_i, een positieve, geheeltallige prijs p_i, en een opbrengst c_i. Het doel is om een toegelaten verzameling items te kiezen zodanig dat de totale opbrengst wordt gemaximaliseerd onder de voorwaarde dat het totale gewicht van de gekozen items niet meer is dan B en de totale kosten van de gekozen items niet meer dan P bedragen, waarbij B en P gegeven positieve gehele getallen zijn.

Geef aan hoe een optimale oplossing van het twee-dimensionale knapzak probleem gevonden kan worden met behulp van DP. Kun je dit algoritme altijd toepassen, of moet je nog speciale eisen stellen aan de opbrengsten c_i ($i = 1, \ldots, n$)?

Opgave 73. Beschouw het volgende telprobleem. Er zijn n items; item i ($i = 1, \ldots, n$) heeft een positief geheeltallig gewicht a_i. Je hebt een rugzak van omvang B.

a. Bepaal het aantal verschillende mogelijke toegestane beladingen, waarbij de rugzak precies gevuld is. Ieder voorwerp kan maximaal één maal worden meegenomen.

b. Hoe moet je de toestandsvariabele veranderen indien je het aantal verschillende mogelijke toegestane beladingen wilt weten waarbij je de rugzak niet volledig hoeft te vullen, zonder de aantallen mogelijkheden op te tellen voor $0, 1, \ldots, B$?

Opgave 74. Beschouw het probleem van het vermenigvuldigen van n matrices $A_1 \circ \ldots \circ A_n$ (het symbool $\circ$ staat hierbij voor de standaard matrixvermenigvuldiging). De uitkomst hangt niet af van de volgorde waarin we de tussenstappen berekenen, maar de hoeveelheid werk wel. Stel dat we een algoritme gebruiken waarbij het vermenigvuldigen van een $(m \times n)$ met een $(n \times p)$ algoritme mnp stappen kost. Het berekenen van $A_1 \circ A_2 \circ A_3$ kan door eerste $A_1 \circ A_2$ te berekenen of eerst $A_2 \circ A_3$. Wanneer A_1 een (3×1) matrix is, A_2 een (1×3) matrix en A_3 weer een (3×1) matrix, dan kost het beginnen met $A_1 \circ A_2$ in totaal $9 + 9 = 18$ stappen werk, terwijl het beginnen met $A_2 \circ A_3$ maar $3 + 3 = 6$ stappen werk kost. We gaan de optimale volgorde berekenen met behulp van DP. Neem aan dat A_i een $(p_{i-1} \times p_i)$ matrix is,

voor $i = 1, \ldots, n$.

a. Definieer $f(i,j)$ $i = 1, \ldots, n$; $i \leq j \leq n$, als de minimale hoeveelheid stappen werk die het kost om de matrices $A_i \circ \ldots \circ A_j$ uit te vermenigvuldigen. Bepaal een geschikte initialisatie voor alle $f(i,j)$ waarden.

b. Stel dat je voor een gegeven k met $i \leq k < j$ reeds de waarden $f(i,k)$ en $f(k+1,j)$ hebt berekend. Bereken hoeveel werk het kost om $A_i \circ \ldots \circ A_j$ te bepalen door gebruik te maken van de tussenproducten $A_i \circ \ldots \circ A_k$ en $A_{k+1} \circ \ldots \circ A_j$.

c. Formuleer een recurrente betrekking om $f(i,j)$ te bepalen.

d. Geef aan hoe je het maxtrixvermenigvuldigingsprobleem optimaal op kunt lossen.

Opgave 75. Beschouw het probleem van het balanceren van witruimtes aan het eind van de zin. We moeten een verzameling woorden met lengtes $l_1, \ldots, l_n$ typen, waarbij we steeds links beginnen en precies één spatie tussen twee opvolgende woorden op dezelfde regel zetten; iedere regel heeft precies M posities. De volgorde van de woorden is gegeven, maar we mogen deze woorden verdelen over de regels zodanig dat de hoeveelheid witruimte aan het eind van de regels enigszins gebalanceerd is. We berekenen de "gebalanceerdheid"van een oplossing door de derde macht te nemen van het aantal spaties op het eind van iedere regel behalve de laatste en vervolgens te sommeren.

Geef aan hoe een optimale oplossing van dit probleem berekend kan worden met behulp van DP.

Opgave 76. Beschouw het volgende machinevolgordeprobleem. Er zijn n taken gegeven. Je kunt kiezen of je een taak uit wilt voeren; dit levert geld op, maar het kost bewerkingscapaciteit, en de taak moet op tijd af zijn. Voor taak i ($i = 1, \ldots, n$) is gegeven dat het uitvoeren ervan c_i oplevert, mits taak i uiterlijk op tijdstip d_i klaar is; het uitvoeren van taak i vergt een ononderbroken periode van p_i tijdseenheden, waarbij p_i een positief geheel getal is. Voor het uitvoeren van de taken is één machine beschikbaar vanaf tijdstip 0; deze machine heeft een capaciteit van 1.

Het is gemakkelijk aan te tonen met behulp van een omwisselingsargument dat het nooit slecht is om de verzameling van geselecteerde taken uit te voeren in volgorde van niet-afnemende d_j, waarbij je ervoor zorgt dat er geen wachttijd tussen de taken zit. Geef aan hoe je een toegelaten selectie met bijbehorende planning kunt vinden met behulp van DP.

Opgave 77. Het HAMILTONCYKEL probleem is als volgt gedefinieerd: Gegeven een **willekeurige samenhangende** graaf $G = (V, E)$, bevat deze graaf een tour die ieder punt $v \in V$ precies éénmaal bezoekt?

Het HAMILTONPAD probleem is als volgt gedefinieerd: Gegeven een **willekeurige samenhangende** graaf $G = (V, E)$, bevat deze graaf een pad dat ieder punt

$v \in V$ precies éénmaal bezoekt? Je moet dus in het Hamiltonpad probleem beginnen in een punt, alle punten bezoeken, maar dan hoef je niet terug te gaan naar je beginpunt; het beginpunt mag je dus zelf kiezen.

a. Toon aan dat het HAMILTONPAD probleem $\mathcal{NP}$-volledig is door middel van een reductie vanuit het HAMILTONCYKEL probleem.

b. Toon aan dat het HAMILTONCYKEL probleem $\mathcal{NP}$-volledig is door middel van een reductie vanuit het HAMILTONPAD probleem.

7.5 Bronverantwoording

Opgaves en het machinevolgordeprobleem zijn aangereikt door Han Hoogeveen[1].

Hoofdstuk 8

MST & Kortste pad

8.1 Grafen

Vanaf hier zal dit dictaat zich alleen nog maar bezig houden met grafen en gerelateerde concepten. Een graaf is een verzameling punten en een verzameling kanten/pijlen. In het geval van een ongerichte graaf kanten, in het geval van een gerichte graaf pijlen. Notatie $G = (V, E)$ voor een ongerichte graaf en $G = (V, A)$ voor een gerichte graaf, hier staat de V voor vertices/punten, de E voor edges/kanten en de A voor arrows/pijlen. Een graaf waarin ieder punt vanaf ieder ander punt bereikbaar is heet samenhangend.

Bij een gerichte graaf worden voor ieder punt de begrippen in- en uitgraad gedefinieerd. Hierbij is de ingraad het aantal pijlen wat naar dit punt toewijst en de uitgraad het aantal pijlen dat van dit punt afwijst. De totale graad van een punt is het totale aantal pijlen en dus de ingraad en de uitgraad bij elkaar opgeteld.

Een pad in een graaf is gedefinieerd als een verzameling opvolgende pijlen/kanten die op elkaar aansluiten. Een pad kan ook weergegeven worden als een opvolgende verzameling punten. Het pad gedefinieerd door de verzameling punten $(v_0, v_1, ..., v_n)$ correspondeert met het pad van de verzameling pijlen/kanten $(v_0, v_1), (v_1, v_2), ..., (v_{n-1}, v_n)$. Een pad wat begint en eindigt in hetzelfde punt heet een cykel. Een graaf waar het niet mogelijk is een cykel te vormen heet een acyclische graaf.

Voorbeeldopgave 8.1. Bewijs dat indien de uitgraad van ieder punt groter dan gelijk aan 1 is de graaf een cykel bevat.

Doordat ieder punt op zijn minst een uitgaande pijl heeft kan er altijd doorgelopen worden met een nieuwe pijl. Dan doorlopen totdat een punt tweemaal bezocht is levert altijd een cykel op.

8.2 Minimaal opspannende boom

Een opspannende boom is een set kanten die alle punten van een samenhangende nietgerichte graaf verbindt met zo min mogelijk kanten. Voor een graaf $G = (V, E)$

is dat dus E' een subset van E zodat alle punten verbonden zijn met $|E'|$ zo klein mogelijk. Dit betekent ook dat E' geen cykels bevat. Uit een cykel kan altijd een kant weggehaald worden waarna nogsteeds alle punten bereikt kunnen worden. Een gewone boom is een deelverzameling van kanten die een opspannende boom vormt voor een deel van de graaf.

Theorema 8.2. *Voor een opspannende boom $B = (V, E)$ geldt $|E| = |V| - 1$*

Bewijs. Bewijs met behulp van inductie. Het klopt voor $|E|$ is 1 want dan is er maar een punt en geen kant. Nu wordt er bewezen dat het ook voor $E = n + 1$ geldt. Aangenomen dat het waar is voor n. Nu is er een opspannende boom met $n+1$ punten waarvan niet bekend is of dit waar is. Een van deze punten heeft graad 1 want als alle punten graad 2 hebben is er een cykel, dit bewijzen is een opgave. Dit punt met graad 1 kan samen met de bijbehorende kant weggehaald worden om een opspannende boom van n punten te krijgen. Voor deze boom hebben we aangenomen dat de stelling geldt en voor de nieuwe boom is maar 1 kant en 1 punt toegevoegd en dus geldt de stelling altijd. □

Als kanten een lengte krijgen en niet alleen maar een verbinding meer zijn dan zijn er nog steeds veel opspannende bomen maar wordt er vaak gezocht naar de minimaal opspannende boom. Dat wil zeggen de set kanten die een opspannende boom vormen zodat de lengte van deze kanten in het totaal minimaal is. Het algoritme van Kruskal zal behandeld worden en het algritme van Prim zal in een opgave naar voren komen. Het is sterk aan te raden die opgave te maken.

Een minimale opspannende boom vinden met het algoritme van Kruskal werkt als volgt. De kanten worden op volgorde van kort naar lang gesorteerd. De kortste kant wordt toegevoegd, als deze een cykel vormt word deze weer weg gehaald daarna wordt de volgende kortste kant toegevoegd aan de verzameling van reeds geselecteerde kanten. Totdat de boom exact $|V| - 1$ kanten bevat en dus een minimaal opspannende boom is.

Theorema 8.3. *Het algoritme van Kruskal geeft een minimaal opspannende boom*

Bewijs. Dit bewijs is gebaseerd op een tegenspraak. De opspannende boom van het algoritme van Kruskal noemen we K. Stel er is een opspannende boom met een lengte kleiner dan K, we noemen deze boom OPT. Als in beide bomen de kanten op lengte van kort naar lang gesorteerd worden is er ergens een index i waar voor de i'de kant in OPT niet gelijk is aan de i'de kant in K. Deze kanten worden e_{opt} en e_k genoemd respectievelijk. De kant e_{opt} is langer dan of even lang als de kant e_k. Als de kant e_{opt} even lang is kan deze vervangen worden door een andere even lange kant of een kortere kant, anders had Kruskal de kant ook toegevoegd. Nu werd er een cykel gevormd bij het toevoegen van die kant dus kan die kant vervangen worden door een op zijn minst even lange kant. Dan moet er nog een verschil zijn tussen OPT en K anders is OPT niet kleiner, bekijk dan het volgende verschil op dezelfde manier. Als de kant korter was geweest had Kruskal hem ook gebruikt, dat is niet gebeurt dus moeten alle kortere kanten een cykel opleveren. De kant e_k wordt nu toegevoegd aan OPT. Nu kan de kortste kant uit deze cykel verwijderd worden

om een nieuwe opspannende boom te maken. Deze kant is niet e_k aangezien alle andere kanten dan korter zouden zijn en dus al in K zaten toen e_k daar toegevoegd werd. Dit zou betekenen dat de toevoeging van e_k een cykel zou maken en dus niet in K kan zitten. Dan wordt een andere kant met lengte groter dan e_k uit de cykel verwijdert. Dit levert een nieuwe boom op met een lengte kleiner dan OPT. Dit is een tegenspraak aangezien OPT al optimaal was. □

8.3 Kortste pad algoritme

Kortste pad algoritme zijn belangrijk binnen de informatica. Deze paragraaf zal het algoritme van Dijkstra behandelen en een korte toekomstblik geven op algoritmes die ook met negatieve lengtes en cykels kunnen werken. Deze algoritmes zullen verder uitgediept worden in de opgaves. Dijkstra's algoritme bepaalt de optimale afstand van een punt s (source) naar alle andere punten in een graaf $G = (V, A)$. Deze graaf mag beide gericht of ongericht zijn, dit maakt verder niet uit zolang er maar geen negatieve kanten in zitten. De afstand naar alle andere punten word altijd berekend, er is geen mogelijkheid om een punt los te doen. Eerst is er een lege verzameling Q. Het algoritme kent alle punten verbonden aan s een waarde toe gelijk aan de lengte van de kant (s, v): $f(v)$ met v een punt in V. Het punt met de laagste waarde wordt in de verzameling Q toegevoegd. Hierna berekent het algoritme voor alle punten die verbonden zijn met het punt v_0 wat zonet toegevoegd is de afstand tot s door v_0 dus $f(v_0) + l(v_0, v)$ waarbij $l(v_0, v)$ de lengte is van de kant tussen v_0 en v. De waardes van de punten v worden veranderd in de minimale waarde van wat er al in v stond en $f(v_0) + l(v_0, v)$. Hierna herhaalt het algoritme zichzelf tot alle punten toegevoegd zijn.

Dit garandeert de kortste route naar ieder punt omdat ieder punt wat een waarde krijgt op dat moment het dichts bijzijnde toevoegbare punt is. Stel er is wel een andere kortere weg naar een punt p_0. De stap voordat p_0 toegevoegd wordt zijn er dan twee opties waardoor een snellere weg gevonden zou kunnen worden. Door eerst een ander punt toe te voegen of door vanuit een ander al toegevoegd punt naar het nieuwe punt te gaan. Als er eerst een ander punt toegevoegd wordt is dat punt dus verder weg dan p_0 en omdat er geen negatieve kanten zijn is er door dat punt geen beter route te vinden. Als het punt vanuit een ander punt wat al door het algoritme gevonden is zou moeten gaan dan heeft het punt die waarde al aangezien bij het toevoegen van dat punt die mogelijkheid al berekend is. Het is eigenlijk een soort DP waarbij $f(v)$ het kortste pad is om van s naar v te komen waarbij alle punten in Q gebruikt mogen worden.

8.4 Opgaves

Opgave 78. Beschouw een willekeurige opspannende boom T in een gegeven graaf $G = (V, E)$. Wanneer we een kant $e \in E \setminus T$ toevoegen aan T, dan ontstaat er een cykel. Laat zien dat het weglaten van een willekeurige kant uit dat cykel leidt tot een opspannende boom T'.

Opgave 79. Het algoritme van Prim (of van Prim-Dijkstra) berekent een kortste opspannende boom op de volgende manier:

1. Kies een willekeurig punt $v \in V$ als beginpunt. Begin met $V' = \{v\}$ en $E' = \emptyset$.
2. Bepaal de kant $\{x, y\}$ met minimale lengte waarvoor geldt dat $x \in V'$ en $y \notin V'$. Voeg $\{x, y\}$ toe aan E' en voeg y toe aan V'.
3. Indien $V' \neq V$, ga door met (2).

a. Bewijs de volgende stelling, of toon aan dat zij onjuist is: *De boom die wordt geconstrueerd bij het voor de $k$de keer ($k = 1, \ldots, |V| - 1$) doorlopen van Stap 2 is de boom met minimale lengte die punt v bevat.*

b. Toon aan dat het algoritme van Prim-Dijkstra een optimale opspannende boom vindt. Wanneer je hebt bij (a) hebt aangetoond dat de stelling correct is, dan is het verstandig om die te gebruiken; wanneer je bij (a) hebt aangetoond dat de stelling onwaar is, dan kun je het best op een tegenspraak aansturen, waarbij je de opspannende boom van Prim-Dijkstra vergelijkt met een optimale opspannende boom.

Opgave 80. Een simpele manier om voor ieder tweetal punten het kortste pad ertussen te bepalen (all pairs shortest) is het toepassen van een single-source kortste pad algoritme waarbij je als startpunt steeds een ander punt neemt. Wanneer alle afstanden ≥ 0 zijn, dan kun je n maal Dijkstra's algoritme toepassen, wat dus $O(n^3)$ aan tijd kost (dit is redelijk efficiënt), maar wanneer er negatieve lengtes bij zitten (maar geen negatieve cykels), dan kost het $O(n^4)$ tijd om n maal Bellman-Ford toe te passen. Johnson heeft voor deze situatie een sneller algoritme gevonden, waarbij je eerst de afstanden aanpast, zodat iedere lengte ≥ 0 is. Dit moet natuurlijk wel op een dusdanige wijze dat de optimaliteit van een pad van s naar t er niet onder lijdt.

Gegeven de graaf $G = (V, A)$, voeg een extra punt s toe aan V en voeg extra pijlen (s, v) (voor alle $v \in V$) toe aan A; deze pijlen krijgen allemaal lengte 0. Bepaal nu voor s de afstand $l(v)$ van s naar ieder punt $v \in V$ met behulp van Bellman-Ford. Pas vervolgens de afstand $d(v, w)$ van iedere pijl (v, w) aan tot $c(v, w) = d(v, w) + l(v) - l(w)$.

a. Toon aan dat $c(v, w) \geq 0$ voor alle $(v, w) \in A$.

b. Toon aan dat de kortste paden gevonden met behulp van Dijkstra's algoritme op basis van de afstanden $c(v, w)$ ook optimaal zijn voor de oorspronkelijke afstanden matrix.

Opgave 81. Een ander all pairs shortest algoritme is het algoritme van Floyd-Warshall. Dit algoritme is gebaseerd op DP en werkt als volgt. Nummer de punten in V als $1, \ldots, n$; de gekozen volgorde is niet van belang. Voor ieder tweetal punten v en w in V definieer je $f_i(v, w)$ als de lengte van het kortste pad tussen v en w, waarbij je uitsluitend de punten $1, 2, \ldots, i$ als tussenpunt mag gebruiken.

Ga na hoe je dit probleem op die manier met DP kunt oplossen. Wat is de rekentijd?

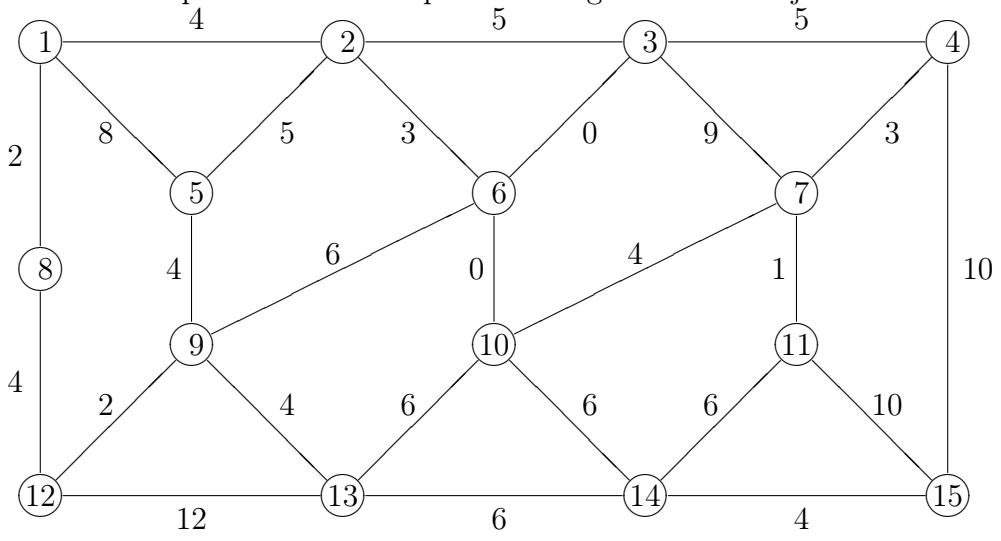
Opgave 82. Dijkstra's algoritme werkt in het algemeen niet in geval van negatieve afstanden (zoals we hebben gezien op college). In geval van een acyclische graaf blijkt dat we toch een Dijkstra-achtig algoritme kunnen toepassen, ook in geval van negatieve lengtes.

Ontwerp zo'n algoritme, en toon aan dat het correct werkt. Uiteraard mag u hierbij de correctheid van Dijkstra's algoritme als gegeven beschouwen.

Opgave 83. Beschouw de onderstaande graaf.

a. Bereken de minimale opspannende boom in deze graaf met de algoritmen van Kruskal en/of Prim-Dijkstra.

b. Bereken in de onderstaande graaf de kortste paden vanuit punt 1 en vanuit punt 6 naar alle andere punten met behulp van het algoritme van Dijkstra.



Figuur 8.1: Graaf1

Opgave 84. In geval van het *gate-assignment probleem* is het de bedoeling om vliegtuigen toe te wijzen aan een gate. Hierbij is voor ieder vliegtuig het interval bekend gedurende welke het een gate nodig heeft en wat voor type gate dat moet zijn. Uiteraard moet dit op een optimale wijze, waarbij de kwaliteit van een oplossing wordt beoordeeld op basis van de tijd tussen twee opvolgende vliegtuigen die aan dezelfde gate zijn toegewezen; wanneer die tijd t minuten is, dan levert dit een score

van $f(t)$ op, waarbij $f(t) = -\infty$ voor $t \leq 15$ en voor $t > 15$ is $f(t)$ een niet-dalende eindige functie. Omdat je natuurlijk niet de triviale oplossing van een lege gate wilt, krijg je verder nog een bonus $b(v)$ voor ieder vliegtuig v dat je aan een gate zet; deze bonus is afhankelijk van het vliegtuig.

Stel dat we een gate hebben van een gegeven type. Laat zien hoe je het maken van de beste toewijzing aan die gate kunt oplossen als een kortste pad probleem. Geef ook aan hoe je om kunt gaan met de extra beperking dat je minstens 5 vliegtuigen aan de gate wilt zetten.

Opgave 85. Het LEVERINGSPROBLEEM is als volgt gedefinieerd. Gegeven is een gerichte graaf $G = (V, A)$ met capaciteiten op de pijlen. De punten $s_1, \ldots, s_q$ corresponderen met fabrieken die producten willen leveren; s_i kan maximaal a_i eenheden product leveren. De punten $t_1, \ldots, t_r$ corresponderen met afnemers die producten willen ontvangen; t_i wil b_i eenheden product ontvangen. Gegeven dat $\sum_{i=1}^{q} a_i \geq \sum_{i=1}^{r} b_i$, is het mogelijk om de afnemers volledig te beleveren?

Geef aan hoe het bovenstaande probleem als een stroomprobleem kan worden opgelost.

Opgave 86. Het vinden van een toegestane stroom
Gegeven is een gerichte graaf $G = (V, A)$ met een bron s en een put t. Voor iedere pijl is gegeven hoeveel stroom er minimaal en hoeveel er maximaal door moet gaan. Het probleem is het vinden van een stroom (hoeft geen maximale omvang te hebben) die aan de onder- en bovengrenzen voldoet.

Geef aan hoe het bovenstaande probleem als een stroomprobleem kan worden opgelost.

Opgave 87. Afsluiting van de steden ten behoeve van het rekening rijden
Voor de invoering van het rekening rijden is het noodzakelijk om de vier grote steden 'volledig af te sluiten', dat wil zeggen, het moet onmogelijk worden om zo'n stad binnen te kunnen komen met de auto zonder gedetecteerd te worden (=zonder te betalen). Om te kunnen detecteren moet over de weg een detectiepoort worden gebouwd. Voor ieder type weg is bekend hoeveel dit gaat kosten. Verder is bepaald dat de detectiepoortjes aan de rand van de stad dienen te komen, dat wil zeggen, binnen vijf kilometer van de rand van de stad. Het probleem is dan het vinden van de goedkoopste toegestane afsluiting.

Geef aan hoe je je (inmiddels opgedane) kennis van het maximale stroomprobleem kunt gebruiken om dit probleem op te lossen.

8.5 bronverantwoording

Dit hele hoofdstuk is in de bij dit dictaat horende stijl en met wat voorbeelden in elkaar gezet, grote delen van de inhoud behandelvolgorde en notatie zijn identiek aan

de samenvatting en college aantekeningen van Han Hoogeveen over het onderwerp[1]. De uitleg van Dijkstra's algoritme komt van mij met enige inspratie van beide Han Hoogeveen en Introduction to algorithms[3].

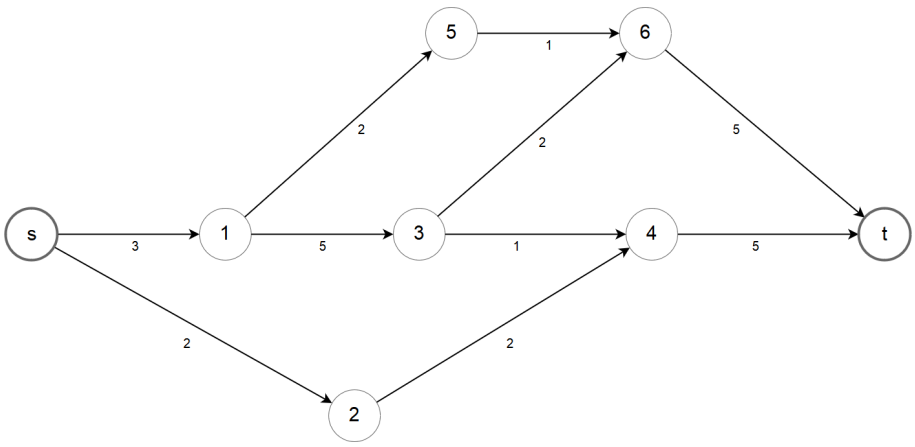
Hoofdstuk 9

Basisproblemen Combinatorische Optimalisering

Dit hoofdstuk bevat beide de laatste concepten van het vak discrete wiskunde en de inleiding van het onderzoeksdeel van het dictaat. In dit hoofdstuk zullen statische stroomgrafen behandeld worden.

9.1 Stroomgrafen

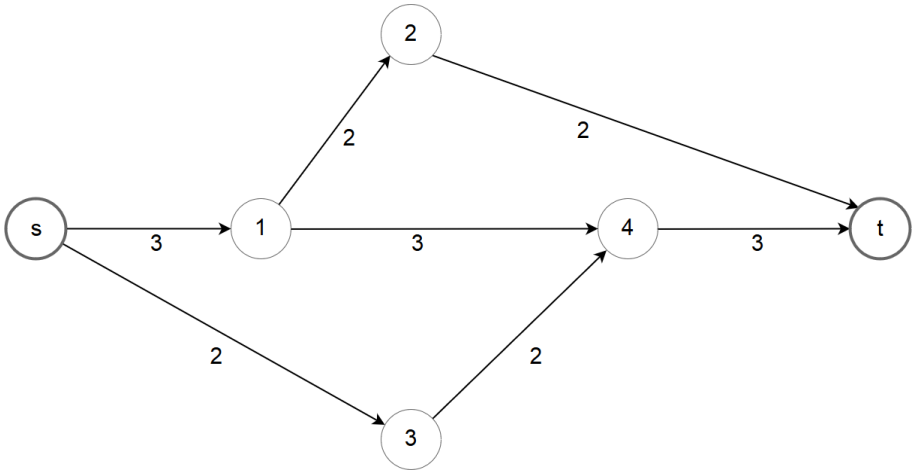
Een stroomgraaf of netwerk (G, c) bestaat uit een gerichte Graaf $G = (V, A)$ en een set c met capaciteit waarde voor de pijlen. Deze graaf bevat een punt s, source, en een punt t, sink. De pijlen geven aan dat er stroom kan zijn tussen twee punten. De capaciteit waarden geven aan hoeveel stroom er kan zijn door een gegeven pijl. Verder is gegeven dat de stroom die bij een punt binnenkomt even groot is als de stroom die er uit gaat, behalve bij de punten s en t. Bij s gaat er alleen stroom uit en bij t gaat er alleen stroom in. De hoeveelheid stroom die s verlaat wordt ook wel de flow value van (G, c) genoemd. Aangezien alleen stroomgrafen waarin s geen binnenkomende pijlen heeft behandeld zullen worden zal dit altijd gelijk zijn aan de stroom die s verlaat.



Figuur 9.1: Een voorbeeld van een stroomgraaf

9.2 Max flow, min cut

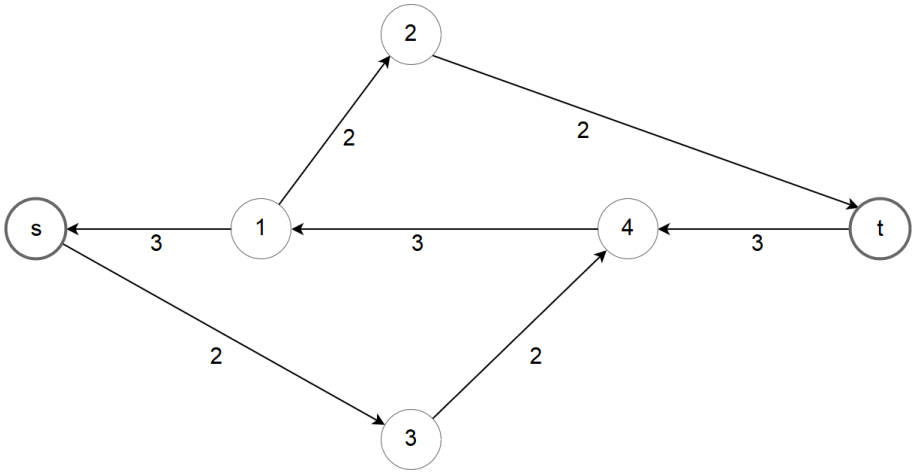
Bij stroomgrafen gaat het er vaak om de maximale flow te bepalen. Dat wil zeggen, wat is de maximale hoeveelheid die uit s naar t kan stromen. Dit wordt gedaan met kortste pad algoritmes en de residuele graaf. Eerst wordt een pad van s naar t bepaald. Dan wordt de stroom in iedere buis met de minimum capaciteit van de gebruikte pijlen opgehoogd, dat wil zeggen met zoveel als er door de kleinste pijl heen kan. Nu is dit pad niet verder meer te gebruiken en wordt de residuele graaf geconstrueerd. Deze graaf geeft aan waar nog wel stroming kan plaatsvinden. Dit is een combinatie van de niet gebruikte pijlen, de wel gebruikte pijlen die niet volledig gebruikt zijn en een verzameling terugpijlen. De niet gebruikte pijlen kunnen natuurlijk nog steeds gebruikt worden, het deel van de wel gebruikte pijlen dat nog niet gebruikt wordt kan ook nog gebruikt worden. De terugpijlen zijn pijlen die stroom opheffen, het kan namelijk zo zijn dat een eerder gemaakt pad helemaal niet voordelig blijkt te zijn in een later scenario en dan moet dit wel ongedaan gemaakt kunnen worden. Ter illustratie een voorbeeld:



Figuur 9.2: Een voorbeeld van een stroomgraaf

In de stroomgraaf van figuur 9.2 wordt eerst het middenpad gevonden. En voor 3 gevuld. Dit is echter niet de optimale oplossing. Met de twee vertakkingen boven en onder kan namelijk een stroom van 5 gevonden worden. Hiervoor wordt bij het tekenen van de residuele graaf ook het terughalen van stroom meegenomen als mogelijkheid. De residuele graaf voor deze situatie komt er dan uit te zien als in figuur 9.3
In deze graaf wordt het pad door punten $(s, 3, 4, 1, 2, t)$ gevonden met als maximale stroom 2 met een simpel labeling algoritme wat de bereikbare punten bepaald. Nu is het duidelijk dat de maximale stroom in deze graaf bereikt is. Er kan geen stroom meer uit s stromen en ook niet meer naar t.

In het voorgaande voorbeeld werd de stroom zo groot dat s niet meer kon afgeven. Dan is het duidelijk dat er een maximale stroom bereikt is. Als er echter ergens in het midden van de graaf geen stroom meer door komt is het lastiger te zien dat de maximale stroom bereikt is. Een algoritme dat de residuele graaf gebruikt om te berekenen of er meer stroom doorkomt dat geen verbetering vind is voldoende. De residuele graaf geeft immers iedere mogelijkheid voor een andere stroom die in de

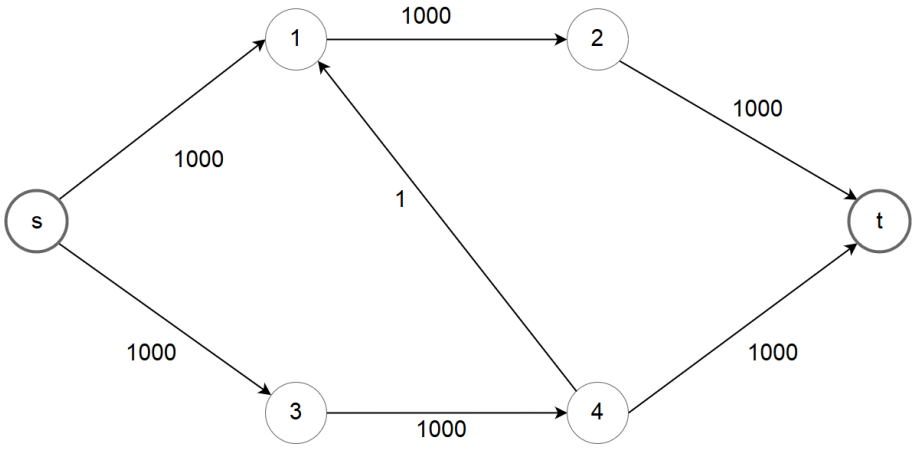


Figuur 9.3: Een voorbeeld van een residuele graaf

graaf zit. Om aan te geven waar de limiterende factor zit is het concept $(s,t) - cut$ uitgevonden.

Een $(s,t) - cut$ is het in twee sets, V_1 en V_2, verdelen van alle punten in de graaf. Waarbij V_1 het punt s bevat en V_2 het punt t. De capaciteit van de flow van een $(s,t) - cut$ wordt berekent door de capaciteiten van ieder punt uit V_1 naar V_2 bij elkaar op te tellen. Als deze gelijk is aan de maximale flow dan wordt dat een minimale cut genoemd. Aangezien de maximale flow van de set V_1 naar de set V_2 gelijk is aan de maximale flow is dit ook een limiterende factor, als er niet meer van V_1 naar V_2 kan kan er ook niet meer van s naar t omdat deze in V_1 en V_2, respectievelijk, zitten. Het bepalen van V_1 kan gedaan worden door alle punten die in de residuele graaf bereikbaar zijn vanuit s te nemen.

Eerder in deze paragraaf werd genoemd dat er gebruik gemaakt wordt van een kortste pad algoritme, dit wordt gedaan omdat er anders inefficiënte paden gekozen kunnen worden op een dusdanige manier dat het extreem lang duurt voordat een netwerk opgelost is. Neem bijvoorbeeld het onderstaande netwerk.



Figuur 9.4: Een voorbeeld van een residuele graaf

Als hierbij niet het korste pad gekozen wordt maar $(s,3,4,1,2,t)$ wordt deze maar

voor 1 gevuld. De residuele graaf kan dan het pad $(s, 1, 4, t)$ volgen wat weer maar 1 vulling oplevert. Zo duurt een probleem wat in 2 stappen met een kortste pad algoritme klaar is 2000 stappen. Een Israëlische wetenschapper heeft bewezen dat een algoritme dat gebruikt maakt van het kortste pad in de residuele graaf altijd minder dan $|V| \cdot |A|$ iteraties nodig heeft.

Theorema 9.1. *Stelling van Dinic: "Wanneer in de residuele graaf steeds een stroomvermeerderend pad wordt gekozen met eenminimaal aantal pijlen, dan bedraagt het aantal iteraties ten hoogste $|V| \cdot |A|$ "[1]*

Bewijs. Dit bewijs bestaat uit 2 claims gevolgd door een conclusie aan de hand van die claims. Eerst worden $D_1, ..., D_n$ gedefinieerd als de $1, ..., n$'de residuele graaf. Dan wordt $\phi(D_i)$ gedefinieerd als de lengte van een kortste pad in D_i en $\psi(D_i)$ als het aantal pijlen wat in tenminste één van deze kortste paden voorkomt.

Hiervoor geldt dat $\phi(D_i) \leq |V|$ want een kortste pad bevat geen cykels en $\psi(D_i) \leq |A|$ want dit kan alleen groter zijn als er van een kant beide de heen en terug pijl gebruikt worden. Dit zou betekenen dat er twee paden zijn van de vorm $(s \to v)(v, w)(w \to t)$ en $(s \to w)(w, v)(v \to t)$. Maar dan zijn er ook paden van de vorm $(s \to v)(v \to t)$ en $(s \to w)(w \to t)$. Een van deze twee paden is altijd korter dan de vorige twee genoemde paden omdat deze paden samen lengte 2 minder hebben vanwege de twee verwijderde pijlen. Dit is een tegenspraak en dus $\psi(D_i) < |A|$.

Nu hebben we de volgende twee claims:

1. $\phi(D_{i+1}) \geq \phi(D_i)$
2. als $\phi(D_{i+1}) = \phi(D_i)$ dan $\psi(D_{i+1}) < \psi(D_i)$

Voor nu worden deze twee claims als waarheid aangenomen en wordt het bewijs afgemaakt. Hierna zullen beide claims los bewezen worden. Er geldt $\phi(D_1) \leq |V|$ en $\psi(D_1) \leq |A|$ zoals eerder vermeldt. Met deze twee claims geldt nu ook dat voor de laatste residuele graaf D_n geldt dat $n < |A| \cdot |V|$. Dit omdat als n groter zou zijn een van de twee genoemde claims ergens in de reeks met residuele grafen niet meer kan gelden. Er kan maar maximaal $|V| \cdot |A|$ keer aan een van de twee voorwaarde voldaan worden en er wordt altijd aan een van de twee voldaan aangezien de een de ontkenning van de ander in de "als" clausule heeft. □

Claim 1. $\phi(D_{i+1}) \geq \phi(D_i)$

Definieer α_v als het kortste pad van s naar v in D_i. Nu geldt dat voor iedere kant (v, w) in D_{i+1} dat $\alpha_w \leq \alpha_v + 1$ als (v, w) in D_i zit aangezien α_w bereikt kan worden vanuit v. En als de pijl de andere kant op staat dat $\alpha_w \leq \alpha_v - 1$ omdat de pijl gebruikt is in het kortste pad en het kortste pad van v naar w dus via die pijl was. Dit laat zien dat $\alpha_w - \alpha_v \leq 1$. Dit kan gebruikt worden door te kijken naar het kortste pad van D_{i+1}. Dit pad is van lengte k en samen met $\alpha_w - \alpha_v \leq 1$ geldt dat $\phi(D_{i+1}) = k \geq (\alpha_t - \alpha_{v_1}) + ... + (\alpha_{v_k} - \alpha_s) = \alpha_t - \alpha_s = \alpha_t = \phi(D_i)$ dit bewijst claim 1.

Claim 2. als $\phi(D_{i+1}) = \phi(D_i)$ dan $\psi(D_{i+1}) < \psi(D_i)$

Bij de vorige iteratie is er op zijn minst 1 pijl volledig benut en zit dus nu niet meer in de verzameling pijlen die deel uit maken van een kortste pad. De waarde wordt hierdoor in ieder geval een lager. Eventuele terugpijlen kunnen geen deel uitmaken van dit kortste pad omdat het kortste pad bijvoorbeeld $(s \to v)(v, w)(w \to t)$ is en $(s \to w)(w, v)(v \to t)$ daarna een nieuw kortste pad zou moeten zijn. Dit betekent zoals eerder genoemd dat er daarvoor al een korter pad aanwezig was.

9.3 Min cost, max flow

Aan het laten stromen van de meeste dingen in het echte leven is een variabele koste verbonden. Of dat nou een kosten voor onderhoudt of voor gebruik van het netwerk is, het kost altijd wel iets. Als iedere pijl naast maximale capaciteit ook nog een koste per stroomeenheid heeft krijgt het probleem een nieuwe dimensie. Er wordt nog steeds gezocht naar de maximale flow maar dan wel voor de minimale koste. Een min cost, max flow probleem.

Voor deze problemen is een aanpassing op het vorige algoritme genoeg in plaats van het gebruiken van een kortste pad algoritme dat de hoeveelheid te gebruiken pijlen bijhoudt wordt er een kortste pad algoritme gebruikt op de kosten. Dit houdt in dat de lengte tussen twee punten gegeven wordt door de kosten van het laten stromen van een eenheid tussen die twee punten. Hiervoor moet een kortste pad algoritme gekozen worden wat om kan gaan met negatieve kanten. Terugstroom pijlen zullen immers meestal negatief zijn. De volgende stelling toont aan dat het minste kosten pad algoritme tot een optimale oplossing leidt.

Theorema 9.2. *Een graaf $\tilde{G} = (V, \tilde{E})$ die optimale koste heeft voor de huidige hoeveelheid stroom blijft optimale kosten hebben als er meer stroom wordt toegevoegd door het kortste pad berekent met kosten van de edges.*

Bewijs. Er zal bewezen worden dat voor iedere min cost stroom x een nieuwe stroom x' die een min kost pad gebruikt en y' een andere stroom met $omvang(y') = omvang(x')$ geldt dat $kost(y') \geq kost(x')$.

Gebruik zoals eerder beschreven de kosten van een eenheid laten stromen als lengtes definieer deze afstanden als $l(i, j)$. Neem $\phi(v)$ als de kosten van het kortste pad van v naar t gebruikmakend van $l(i, j)$ afstanden. Stroom x' is uit x ontstaan door x te verhogen met ϵ voor vooruitgaande pijlen op het stroom vermeerderende pad en met ϵ te verlagen voor achterwaarts gaande pijlen op het stroom vermeerderende pad. Dus $kost(x') = kost(x) + \epsilon\phi(s)$. Er zal nu met behulp van een hulpresultaat een grens afgeleid worden voor $kost(y') - kost(x)$ waarna het hulpresultaat bewezen zal worden.

Claim 3. Hulpresultaat:
Voor alle $(i, j) \in A$ geldt:

- Als $y_{ij} > x_{ij}$ dan $k_{ij} = l(i, j) \geq \phi(i) - \phi(j)$
- Als $y_{ij} < x_{ij}$ dan $k_{ij} \leq -l(j, i) \leq \phi(i) - \phi(j)$

Hieruit volgt dat $k_{ij}(y'_{ij} - x_{ij}) \geq (\phi(i) - \phi(j))(y'_{ij} - x_{ij})$ wat op zijn beurt weer $kost(y') - kost(x) = \sum_{(i,j)\in A} k_{ij}(y'_{ij} - x_{ij}) = \sum_{(i,j)\in A}(\phi(i) - \phi(j))(y'_{ij} - x_{ij})$ geeft. In deze laatste uitdrukking komt $\phi(i)$ voor als positieve term voor alle pijlen $(i,j) \in A$ en als een negatieve term voor alle pijlen $(h,i) \in A$. Door om te schrijven kan

$$\sum_{i\in V} \phi(i) \left(\sum_{(i,j)\in A} (y'_{ij} - x_{ij}) - \sum_{(h,i)\in A} (y'_{hi} - x_{hi}) \right)$$

verkregen worden. Hier kunnen opnieuw omschrijving plaatsvinden waarna het deel met $y'_{ij} - y'_{hi}$ en $x_{ij} - x_{hi}$ gelijk is aan 0 voor alle punten die niet de source of sink zijn aangezien die altijd gelijke in en uitstroom hebben.

$$\begin{aligned} &\sum_{i\in V} \phi(i) \left(\sum_{(i,j)\in A} y'_{ij} - \sum_{(h,i)\in A} y'_{hi} \right) - \sum_{i\in V} \phi(i) \left(\sum_{(i,j)\in A} x_{ij} - \sum_{(h,i)\in A} x_{hi} \right) \\ =&\phi(s) \left(\sum_{(s,i)\in A} y'_{si} - \sum_{(i,s)\in A} y'_{is} \right) - \phi(s) \left(\sum_{(s,i)\in A} x_{si} - \sum_{(i,s)\in A} x_{is} \right) \end{aligned}$$

Hier is het stuk tussen de haken gelijk aan de totale grootte van de stroom. Dit lever ons de gewenste ongelijkheid op aangezien:

$$\phi(s)(omvang(y') - omvang(x)) = \phi(s)(omvang(x') - omvang(x)) = \epsilon\phi(t) = kost(x') - kost(x)$$

En dus $kost(y') - kost(x) \geq kost(x') - kost(x)$.

Bewijs. **Bewijs van het hulpresultaat:**
Als $y_{ij} > x_{ij}$ dan zat de pijl (i,j) niet vol en zit (i,j) in de residuele graaf. Vanwege de bepaling van $l(i,j)$ geldt dan $k_{ij} \geq l(i,j)$ (geen gelijkheid, want (j,i) kan ook tot A behoren). Aangezien $\phi(i)$ gelijk is aan de afstand van i naar t en het pad $(i,j), (j \to t)$ een mogelijkheid is, volgt $\phi(i) \leq l(i,j) + \phi(j)$.

Voor het bewijzen van de tweede ongelijkheid wordt gebuikt dat uit $x_{ij} > 0$ volgt dat $(j,i) \in \tilde{A}$, zodat $l(j,i) \leq -k_{ij}$ of te wel, $k_{ij} \leq -l(j,i)$. Verder geldt als boven dat $\phi(j) \leq l(j,i) + \phi(i)$, en dus $-l(j,i) \leq \phi(i) - \phi(j)$. □

□

9.4 Opgave

Opgave 88. Een simpele manier om voor ieder tweetal punten het kortste pad ertussen te bepalen (all pairs shortest) is het toepassen van een single-source kortste pad algoritme waarbij je als startpunt steeds een ander punt neemt. Wanneer alle afstanden ≥ 0 zijn, dan kun je n maal Dijkstra's algoritme toepassen, wat dus $O(n^3)$ aan tijd kost (dit is redelijk efficiënt), maar wanneer er negatieve lengtes bij zitten (maar geen negatieve cykels), dan kost het $O(n^4)$ tijd om n maal Bellman-Ford toe te passen. Johnson heeft voor deze situatie een sneller algoritme gevonden, waarbij je eerst de afstanden aanpast, zodat iedere lengte ≥ 0 is. Dit moet natuurlijk wel op een dusdanige wijze dat de optimaliteit van een pad van s naar t er niet onder

lijdt.

Gegeven de graaf $G = (V, A)$, voeg een extra punt s toe aan V en voeg extra pijlen (s, v) (voor alle $v \in V$) toe aan A; deze pijlen krijgen allemaal lengte 0. Bepaal nu voor s de afstand $l(v)$ van s naar ieder punt $v \in V$ met behulp van Bellman-Ford. Pas vervolgens de afstand $d(v, w)$ van iedere pijl (v, w) aan tot $c(v, w) = d(v, w) + l(v) - l(w)$.

a. Toon aan dat $c(v, w) \geq 0$ voor alle $(v, w) \in A$.

b. Toon aan dat de kortste paden gevonden met behulp van Dijkstra's algoritme op basis van de afstanden $c(v, w)$ ook optimaal zijn voor de oorspronkelijke afstanden matrix.

Opgave 89. Een ander all pairs shortest algoritme is het algoritme van Floyd-Warshall. Dit algoritme is gebaseerd op DP en werkt als volgt. Nummer de punten in V als $1, \ldots, n$; de gekozen volgorde is niet van belang. Voor ieder tweetal punten v en w in V definieer je $f_i(v, w)$ als de lengte van het kortste pad tussen v en w, waarbij je uitsluitend de punten $1, 2, \ldots, i$ als tussenpunt mag gebruiken.

Ga na hoe je dit probleem op die manier met DP kunt oplossen. Wat is de rekentijd?

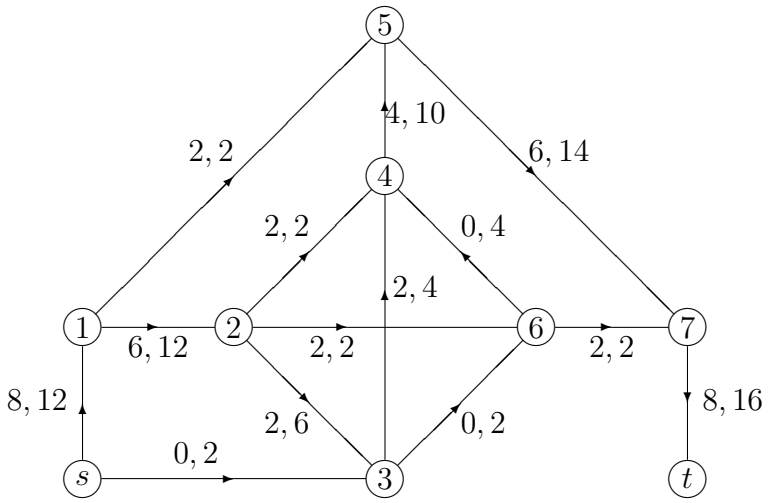
Opgave 90. Dijkstra's algoritme werkt in het algemeen niet in geval van negatieve afstanden (zoals we hebben gezien op college). In geval van een acyclische graaf blijkt dat we toch een Dijkstra-achtig algoritme kunnen toepassen, ook in geval van negatieve lengtes.

Ontwerp zo'n algoritme, en toon aan dat het correct werkt. Uiteraard mag u hierbij de correctheid van Dijkstra's algoritme als gegeven beschouwen.

Opgave 91. Beschouw het netwerk op de volgende bladzijde. De getallen bij de pijlen geven respectievelijk de waarde van de huidige stroom door die pijl en de capaciteit van die pijl aan. Bepaal de maximale stroom door dit netwerk en bewijs de maximaliteit van deze stroom met behulp van een minimale snede.

Opgave 92. Voor de invoering van het rekening rijden is het noodzakelijk om de vier grote steden 'volledig af te sluiten', dat wil zeggen, het moet onmogelijk worden om zo'n stad binnen te kunnen komen met de auto zonder gedetecteerd te worden (=zonder te betalen). Om te kunnen detecteren moet over de weg een detectiepoort worden gebouwd. Voor ieder type weg is bekend hoeveel dit gaat kosten. Verder is bepaald dat de detectiepoortjes aan de rand van de stad dienen te komen, dat wil zeggen, binnen vijf kilometer van de rand van de stad. Het probleem is dan het vinden van de goedkoopste toegestane afsluiting.

Geef aan hoe je dit probleem kunt oplossen.



Figuur 9.5: Netwerk.

Opgave 93. Beschouw de volgende variant van het max-flow probleem. Voor iedere kant is niet alleen een bovengrens op de stroom gegeven, maar ook een ondergrens (deze is niet-negatief). Geef aan hoe deze variant van het max-flow probleem opgelost kan worden.

Opgave 94. Bij het college is het reeds aangekondigd: Je kunt bewijzen dat de lengte van de twee stroomvermeerderende paden die in opeenvolgende iteraties worden gebruikt om het Min-cost Max-flow probleem op te lossen niet afneemt. Ga uw gang.

Opgave 95. Beschouw het volgende planningsprobleem. Er zijn m klassen $k_1, \ldots, k_m$ en n docenten $d_1, \ldots, d_n$; voor iedere leraar-klas combinatie is het aantal lessen l_{ij} die klas i krijgt van docent j. Het doel is om een planning te vinden zodanig dat alle lessen zo snel mogelijk kunnen worden gegeven. Hierbij mag je aannemen dat iedereen altijd beschikbaar is, en dat lessen parallel gegeven mogen worden zolang het om verschillende docent/klas combinaties gaat.

Om de lessen te plannen wordt gebruik gemaakt van een bipartite multigraaf. Iedere klas en iedere docent correspondeert met een punt; tussen klas i en docent j trek je l_{ij} kanten. Deze kanten probeer je dan te kleuren met zo min mogelijk verschillende kleuren, waarbij twee kanten alleen dezelfde kleur mogen krijgen indien ze geen punt gemeen hebben. Uiteraard correspondeert een kleur dan met een lesuur, waarop de lessen met die kleur worden gegeven; het maakt niet uit welke kleur met welk lesuur correspondeert.

a. Toon aan dat de maximale graad een ondergrens is op het aantal benodigde kleuren.

b. Construeer een toegelaten kleuring met dit aantal kleuren. Hierbij mag je gebruik maken van de stelling van König die zegt dat er in een bipartite graaf zo'n kleuring

bestaat met een aantal kleuren dat gelijk is aan de maximale graad. **Hint:** wat voor deelgraaf wordt gevormd door de kanten van gelijke kleur?

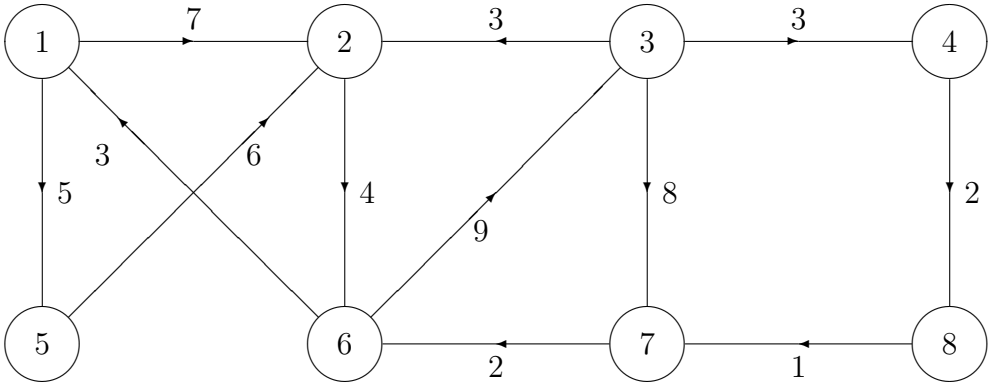
c. Stel dat er maar h klaslokalen beschikbaar zijn (zodat je dus maar h lessen tegelijkertijd kunt geven). Pas de ondergrens van (a) aan door een extra ondergrens te geven op basis van het aantal beschikbare lokalen.

d. Geef aan hoe je de graaf aan kunt passen zodat je met behulp van de stelling van König een toegelaten kleuring kunt vinden met een aantal kleuren dat gelijk is aan het maximum van de ondergrenzen van (a) en (c). **Hint:** Voer dummy punten in bij de verzameling klassen en voeg dummy kanten toe.

Opgave 96. Beschouw het Chinese postbode probleem op een *gerichte* graaf: gegeven een gerichte graaf, bepaal een tour die alle *pijlen* in de graaf minimaal één keer doorloopt van minimale lengte. (Uiteraard mag een pijl alleen in de gegeven richting worden doorlopen).

(a) Aan welke eisen moet de graaf voldoen wil zij een Eulercykel bevatten?

(b) Los het Chinese postbode probleem op voor de hieronder getekende gerichte graaf op 8 punten; de afstanden staan bij de pijlen.



Opgave 97. Het probleem DIRECT VERVOER is als volgt gedefinieerd: Er is één vervoerder, gevestigd in plaats 0, die n vervoersopdrachten V_i $(i = 1, \ldots, n)$ moet uitvoeren. Vervoersopdracht V_i $(i = 1, \ldots, n)$ omhelst het vervoeren van een vracht van een gegeven plaats a_i naar een gegeven plaats b_i. Wanneer vracht i is opgehaald, dan moet deze direct naar plaats b_i worden gebracht; het combineren van twee vrachten is niet mogelijk.

De tijd die gemoeid is met het uitvoeren van V_i is gelijk aan de gegeven hoeveelheid d_i; hierin is alles inbegrepen (in de instantie hieronder is die tijd niet gespecificeerd, aangezien het de oplossing niet beïnvloedt). De tijd die nodig is om met een lege truck van plaats j naar plaats k te rijden is gelijk aan de gegeven hoeveelheid c_{jk}. De vervoerder vertrekt op tijdstip 0 uit plaats 0. Het doel is om een toegelaten oplossing van het probleem te vinden waarbij de vervoerder zo snel mogelijk weer thuis is.

Bepaal een optimale oplossing voor de hieronder gegeven instantie. Zou de door u gebruikte oplosstrategie voor iedere instantie van het probleem DIRECT VERVOER een optimale oplossing zou hebben gevonden?

9.5 bronverantwoording

De bewijzen van theorema 9.1 en 9.2 zijn samen met de opgaven door Han Hoogeveen aangereikt. De rest van het hoofdstuk is ook los gebaseerd op al bestaand lesmateriaal van Han Hoogeveen.[1]

Hoofdstuk 10

Dynamische stroomproblemen

10.1 Introductie

Dit hoofdstuk is extra en zit eigenlijk alleen in het dictaat om mijn scriptie te verantwoorden. Het zal zeker niet op het tentamen komen maar geeft wel een veel dieper inzicht in de stof over stroomgraven en kan zeker ook voor academische verrijking gelezen worden. In dit hoofdstuk zal zoals dat een wiskunde scriptie betaamt de eerste twee jaar van de wiskunde studie als voorkennis beschouwd worden. Voorgaande hoofdstukken van dit dictaat zijn ook voorkennis, in het bijzonder de laatste 4. Ook zal dit hoofdstuk dezelfde stijl en hetzelfde doel aanhouden als de rest van het dictaat.

In dit laatste hoofdstuk zullen dynamische stroomproblemen behandeld worden. Veel mogelijkheden binnen dynamische stroomproblemen zullen benoemd maar niet verder verkend worden om niveau en hoeveelheid te handhaven. In de eerste paragraaf zullen dynamische stroomproblemen in het algemeen behandeld worden. Er zal ook een soort "brute-force" algoritme gegeven worden wat bij alle varianten van dynamische stroomproblemen werkt in het bijzonder problemen met een hoge complexiteitsgraad. Hierna zal voor minder complexe problemen een veel sneller algoritme gegeven worden.

10.2 Tijdsafhankelijkheid

Een dynamisch stroomprobleem is een stroomprobleem wat ook rekening houdt met de tijd die het kost om te stromen. Een stroomgraaf tijdsafhankelijk maken kan op twee manieren continue of discreet. Een continue tijdsafhankelijkheid houdt in dat de tijd niet in stapjes maar met een continue schaal voortbeweegt. Bij een discrete tijdafhankelijkheid wordt er gekeken naar een tijd T, opgedeeld in intervallen van t. Een discrete tijdsafhankelijke stroomgraaf wordt dan ook gegeven door $G = (V, A, T)$. Met V de verzameling punten, A de verzameling pijlen en T de verzameling tijdsintervallen.Iedere pijl (i, j) in de verzameling pijlen moet ook een reistijd λ_{ij} hebben en natuurlijk nog steeds aan capaciteit μ_{ij}.

De volgende paragraaf zal een methode beslaan om discrete dynamische stroomgra-

fen om te zetten in statische stroomgrafen. De laatste paragraaf zal een algoritme bevatten voor dynamische stroomgrafen van iedere soort.

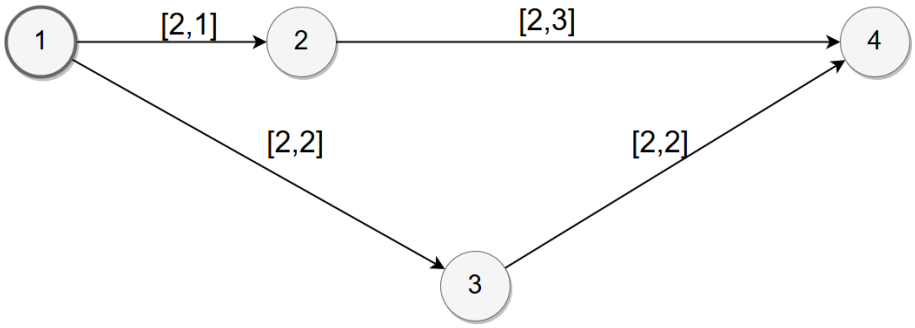
10.3 Evacuatieprobleem

Een goed voorbeeld van een complex dynamisch stroomprobleem is het evacuatieprobleem. Gegeven een blauwdruk van een gebouw en een schatting van de hoeveelheid mensen die in iedere locatie zitten, hoe lang duurt het voordat al deze mensen in veiligheid zijn als het brandalarm af gaat? Dit probleem kan met een dynamische stroomgraaf geconstrueerd worden waar veel tijdsafhankelijk gemaakt wordt. Onder andere de capaciteit en reistijd moeten tijdsafhankelijk worden maar het grootste probleem is dat een stroom mensen langzamer wordt naarmate de stroom groter wordt. Hierdoor moet er veel in de graaf gebeuren en is het daarom nodig de dynamische graaf om te zetten in een (veel grotere) statische graaf. Deze paragraaf zal beschrijven hoe deze omzetting gebeurt met enkele aannames die logisch zijn bij een evacuatieprobleem.

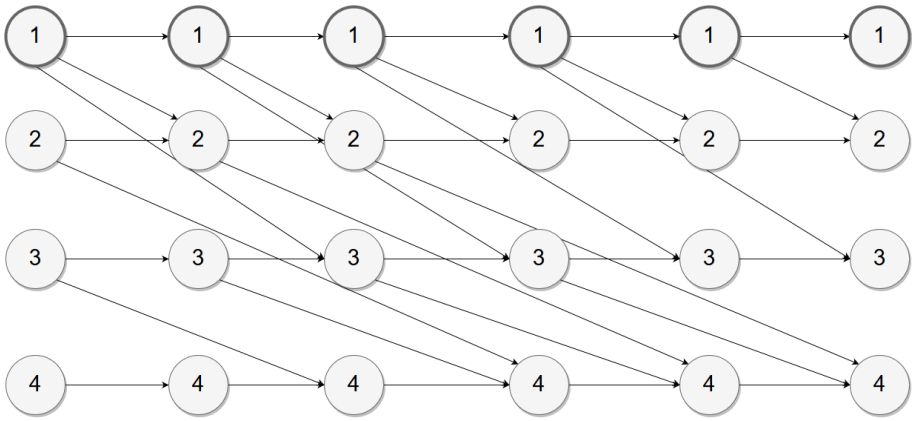
De capaciteit en reistijd kunnen tijdsafhankelijk gemaakt worden. Dan worden beide de capaciteit en de reistijd functies van t. Bij deze tijdsafhankelijke reistijd kan naar twee mogelijkheden gekeken worden. Het "frozen arc model" en het "elastic arc model"[4]. Bij het frozen arc model staat de reistijd vast en is deze dus $\lambda_{ij}(t)$ met t de vertrektijd. Bij het elastic arc model worden veranderingen in de reistijd meegenomen en dus iedere stap herberekend. Er zal in dit deze paragraaf alleen naar het frozen arc model gekeken worden. Verder zal aangenomen worden dat het first-in first-out(FIFO) principe geldt, wat logisch is bij evacuatieproblemen. Dit houdt in dat het onmogelijk is dat later vertrekken een eerdere aankomsttijd oplevert. Dit betekent dat de functies $\lambda_{ij}(t)$ niet dusdanig dalend zijn dat dit mogelijk is. Dus $\lambda'_{ij}(t) \geq -t_1$ met t_1 het eerste tijdsinterval en dus ook de grootte van de tijdsintervallen. Hierdoor is het nooit voordelig later te vertrekken omdat de vermindering in reistijd maximaal zo groot is als de verstreken tijd en dus netto geen verbetering op kan leveren.

Dit alles samen kan omgezet worden in een statische stroomgraaf die het dynamische probleem oplost. Hiervoor wordt voor ieder tijdsinterval ieder punt getekend. Alle pijlen worden van punten in een tijdsinterval t getekend naar het punt waar ze heen gaan in het tijdinterval $t + \lambda_{ij}(t)$. In figuur 9.1 staat een voorbeeldgraaf met daarin bij iedere pijl eerst de capaciteit en daarna de reistijd. In figuur 9.2 is deze dynamische graaf omgeschreven in een statische graaf voor een tijdspanne van 5. Omdat alle pijlen capaciteit 2 hebben zijn de capaciteiten na het eerste figuur weggelaten.

Dit statische stroomprobleem is direct erg groot geworden en veel van de punten zullen niet gebruikt worden omdat de sink/afvoer niet meer bereikt kan worden of het punt niet bereikt kan worden vanaf de source. Deze punten weghalen maakt het probleem significant makkelijker en overzichtelijker. Het gereduceerde probleem wordt dan figuur 10.3.



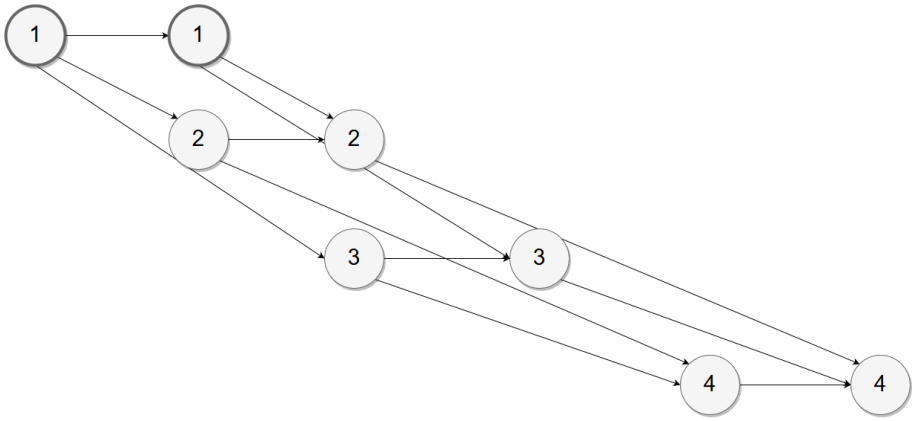
Figuur 10.1: Een voorbeeld van een dynamisch stroomprobleem



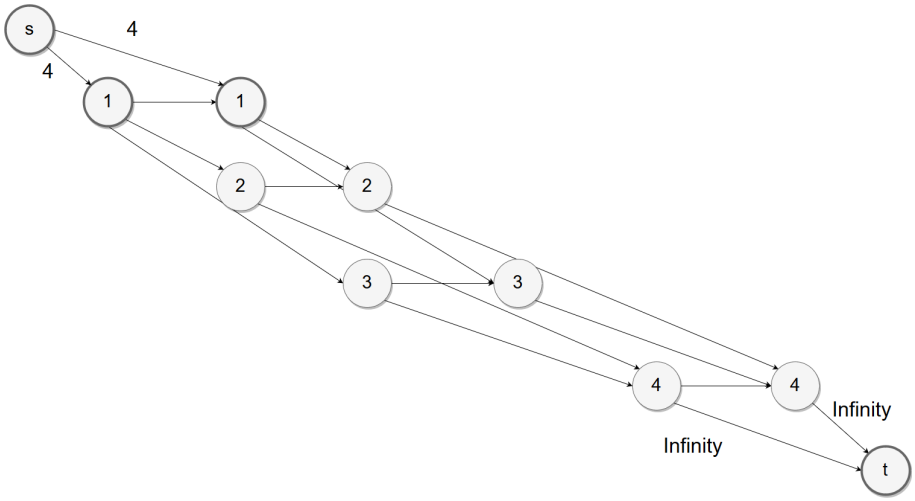
Figuur 10.2: Een voorbeeld van een dynamisch stroomprobleem omgezet in een statisch stroomprobleem

Met deze notatie zijn er meerdere sinks en sources ontstaan. Alle 1 punten zijn sources en alle 4 punten zijn sinks geworden. Dit is echter een probleem aangezien dit niet goed werkt met een statisch stroomprobleem. Hiervoor moeten we een super-source en super-sink toevoegen. De super-source heeft pijlen afhankelijk van het probleem, in deze variant van het probleem krijgen alle pijlen vanaf de super sink 4 als een continue draaiend riool waar altijd 4 binnenkomt, bij een evacuatieprobleem zou voor iedere kamer in de eerste tijdstap een set mensen toegevoegd worden. Hier wordt van een ander voorbeeld gebruik gemaakt omdat anders het meerdere sources probleem niet geïllustreerd kan worden. De super sink krijgt pijlen vanuit alle sinks die oneindig veel capaciteit hebben. Zie figuur 10.4.
Dit is het concept achter discrete netwerken met veel veranderende variabelen en grootteafhankelijke stroomsnelheden. Doordat ieder stukje van de graaf expliciet gemaakt is kunnen deze ook allemaal los aangepast worden en gemodelleerd naar de werkelijkheid. Bij een onderwerp als evacuatie waar mensenlevens mee gemoeid zijn moet veralgemenisering natuurlijk zo veel mogelijk vermeden worden. Dit kost echter voor veel problemen veel te veel rekenkracht voor de simpele problemen die het vaak zijn. In de volgende paragraaf zal een algoritme behandeld worden wat laat zien hoe simpelere problemen sneller en zelfs beter opgelost kunnen worden.

Mensen met meer interesse in hoe een evacuatiemodel exact in elkaar zit wordt aangeraden de masterthesis te lezen waar deze paragraaf een stuk van behandelt, specifiek hoofdstuk 3 en delen van 1 en 7.[4]



Figuur 10.3: Een voorbeeld van een dynamisch stroomprobleem omgezet in een statisch stroomprobleem met de niet nodige punten weggehaald



Figuur 10.4: Een voorbeeld van een dynamisch stroomprobleem omgezet in een statisch stroomprobleem met de niet nodige punten weggehaald en een super sink en super source toegevoegd

10.4 Simpele dynamische stroomproblemen

In deze paragraaf zal een algoritme behandeld worden om dynamische stroomproblemen op te lossen die minder complex zijn dan de in de vorige paragraaf genoemde problemen. Deze dynamische stroomproblemen zullen geen veranderende capaciteit of reistijd hebben. Ze zullen opgelost worden met het universele maximale stroomalgoritme van Wilkinson[5]. Universele maximale stroom in een dynamische stroomgraaf is een maximale stroom over alle tijdseenheden $t = t_0, t_1, ..., t_n$ die als de totale tijd verkort word nog steeds maximaal is.

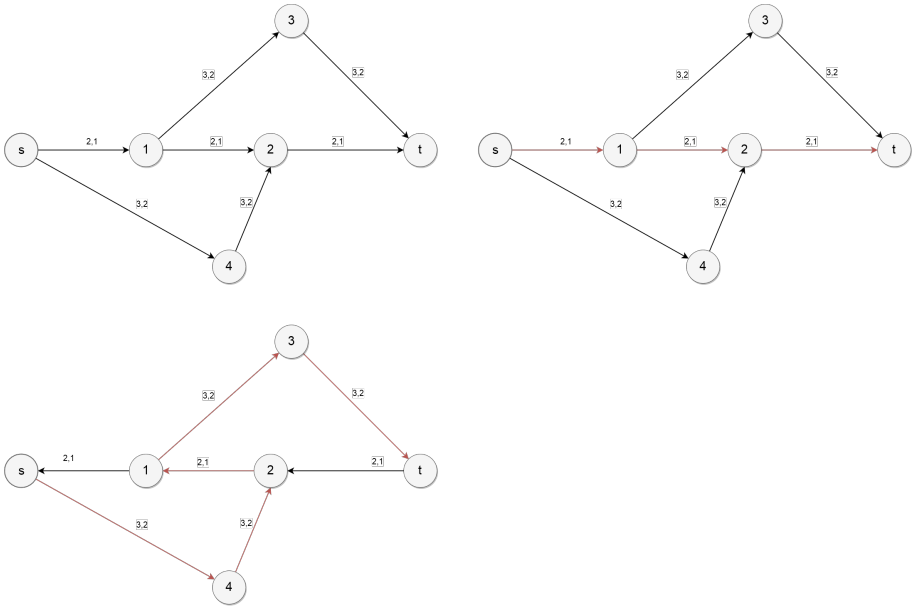
Eerst wordt er kort een algoritme besproken om een normale maximale dynamische flow te vinden. Daarna wordt een aanpassing besproken om met een zelfde soort algoritme een universele maximale flow te vinden. Het algoritme dat voor normale maximale dynamische flow gebruikt wordt komt uit een artikel van Ford en Fulkerson[6]. De aanpassing komt uit een artikel van Wilkinson[5].

Voor het vinden van een normale maximale dynamische flow wordt gebruik gemaakt van tijdelijk herhalende stromen. Dit zijn stromen die zich herhalen voor een bepaalde tijdsduur. Stel er is een stroom van s naar t die 3 tijdseenheden duurt en de totale tijdsspanne is 5 dan kan deze stroom 3 keer herhaald worden. De afstand $l(x, y)$ tussen twee punten wordt ook opnieuw gedefinieerd als het aantal tijdseenheden die het kost om van het ene naar het andere punt te komen. Nu wordt voor $t = t_0$, het eerste moment dat er stroom binnen kan komen de maximale stroom gezocht die aan kan komen voor dat tijdstip. Deze wordt vervolgens tijdelijk herhaald tot het einde van de tijdsspanne. Hier hoort ook een residuele graaf bij, in deze graaf wordt vervolgens voor tijdstip $t = t_1$ gezocht in de residuele graaf naar de maximale stroom die vervolgens tot 1 tijdseenheid voor het einde herhaald wordt. Van de combinatie van deze stromen wordt weer een residuele stroomgraaf gemaakt. In deze nieuwe residuele stroomgraaf wordt berekend wat de maximale stroom is voor tijdstap $t = 2$ enzovoorts. Aan het einde worden al deze herhalende stromen gecombineerd om de volledige maximale flow te maken.

Tot zover een werkend maximaal flow algoritme wat gebruik maakt van het principe van het statische flow algoritme van het vorige hoofdstuk. Merk op dat in het algoritme de maximale flow voor een tussentijdstap berekend is op het moment dat de stroom met die tijdsduur berekend is. Alles wat het algoritme namelijk doet is het berekenen van de aanpassingen die gedaan kunnen worden met de nieuwe beschikbare tijd. Die worden dan vervolgens uitgevoerd voor de tijdstippen waarop dat kan om de nieuwe maximale flow te maken. Er zal een kleine aanpassing aan het algoritme gemaakt worden zodat de flow op een tijdstip als deze maximaal is niet meer minder wordt.

Als een terugpijl uit de residuele graaf gebruikt wordt kan het zo zijn dat daardoor de flow op een eerder tijdstip minder wordt omdat die flow daar niet meer stroomt maar op een later tijdstip nog wel aankomt. De totale flow wordt niet minder maar de flow voor een bepaald tijdstip wel. Hier is omheen te werken door te bepalen wanneer de laatste mogelijkheid is om een terugpijl in te zetten of "te stoppen met de stroom die kant op sturen". Hierdoor wordt de stroom pas omgestuurd als het aangevuld wordt door de stroom die later op dat punt aankomt. Een stroom die van een punt x naar t stroomt zou als er minder in punt x aankomt minder kunnen worden maar doordat alle stromen die al lopen bij het toevoegen van een nieuwe stroom nu niet meer minder kunnen worden kan dit niet meer. Ook zal de stroom van x naar t nooit terugstromen omdat zodra t bereikt is het kortste pad algoritme stopt, dan is het pad gevonden. Om dit te illustreren een klein voorbeeld.

In figuur 10.5 is te zien hoe twee paden gevonden worden één met tijdsduur 3 en een met tijdsduur 7. Nu wordt het plan gemaakt door eerst de stroom van tijdsduur 3 te herhalen en daarna de stroom van tijdsduur 7 er over heen te zetten waarbij de stroom teruggestuurd wordt op het allerlaaste moment zoals geillustreerd in figuur 10.6 (na het einde van het hoofdstuk!). Hierbij zijn de pijlen waar twee cijfers bij staan de pijlen met reistijd 2. Hierbij is het eerste getal de hoeveelheid die er net ingekomen is en het tweede getal de hoeveelheid die er een stap geleden in zat. Zoals



Figuur 10.5: Een stroomgraaf probleem met de twee bijbehorende mogelijke paden.

te zien is er nu niet alleen aan het einde maximale opbrengst maar ook

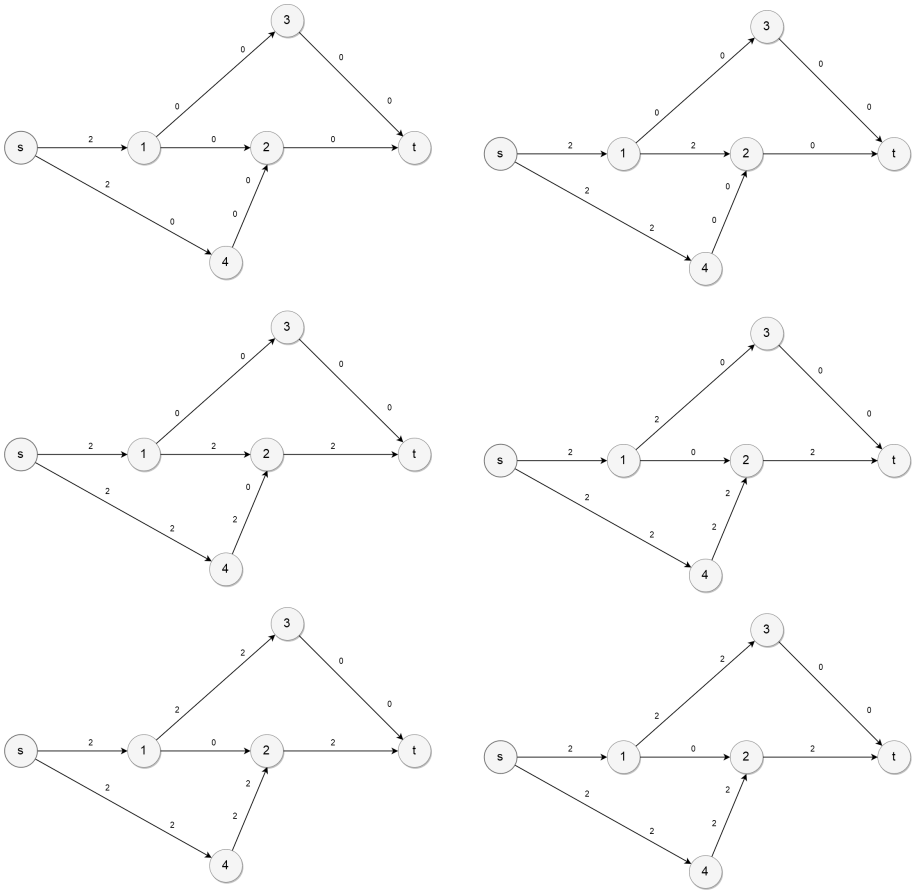
10.5 Toepassingen, discussie en conclusie

Zoals eerder genoemd kunnen dynamische stromen gebruikt worden om evacuaties te simuleren. Hier is veel onderzoek naar gedaan en er zijn ook een hoop artikelen die dieper gaan dan dit. Wilkinson's aanpassing op het algoritme van Ford en Fulkerson heeft veel minder onderzoek gekregen, voor zover er gezocht is is dit de enige Nederlands talige uitleg van de werking van het algoritme. Dit algoritme heeft ook beperkte toepassingen. Een toepassing die te bedenken is, is een netwerk wat niet stabiel is en waar zo veel mogelijk over verstuurd moet worden. Een praktijkvoorbeeld zou een bevoorrading van een leger zijn. Gegeven dat de voorraad in overvloed is zou een universele maximale stroom er voor zorgen dat ook als er problemen optreden in het netwerk er nog steeds voor eerdere tijdstippen zoveel mogelijk aankomt. Een interessante vraag kan zijn in hoeverre een universele dynamische stroom ook gemaakt kan worden als de capaciteiten tijdsafhankelijk zijn.

Uiteindelijk is de bedoeling van dit dictaat dat het gebruikt kan worden om meer mensen met succes het vak discrete wiskunde aan de Universiteit Utrecht te leren.

10.6 Bronverantwoording

In dit hoofdstuk is de bronvermelding direct gedaan.



Figuur 10.6: Het tijdsverloop van de universeel maximale stroom.

Bibliografie

[1] Discrete wiskunde, *Een vak aan de Universiteit Utrecht*, Han Hoogeveen, Utrecht, 3de periode, 2015.

[2] Fred S. Roberts, Barry Tesman, *Applied Combinatorics*, Chapman & Hall 2nd edition, 2009.

[3] Thomas H.Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to algorithms*, The MIT Press, 3rd edition, 2009.

[4] Stevanus Adrianto Tjandra, *Dynamic Network Optimalisation with Application to the Evacuation Problem*, https://kluedo.ub.uni-kl.de/frontdoor/index/index/year/2003/docId/1407, Universität Keiserslautern, 2003.

[5] W.L. Wilkinson, *An Algorithm for Universal Maximal Dynamic Flows in a Network*, INFORMS, Operations Research, Vol. 19, No. 7 (Nov. - Dec., 1971), pp. 1602-161.

[6] L.R.Ford, Jr. And D.R.Fulkerson, *Constructing Maximal Dynamic Flows from Static Flows*, Opns. Res. 6, 419-433 1958.