mei 1990

experimentele versie

W 12 16

Algoritmiek

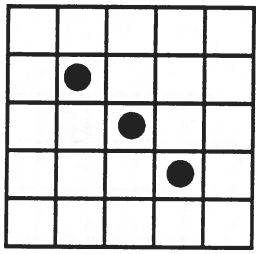
Uitgave in het kader van het team W12-16
Ontwerp: Gerrit van den Heuvel en Hans Krabbendam.

Hoofdstuk 1: Het begrip algoritme

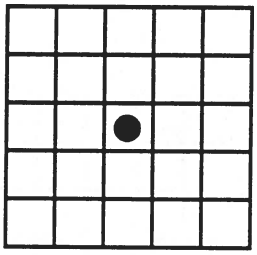
Dit hoofdstuk gaat over het werken volgens bepaalde vaste patronen, algoritmes genaamd. Eerst komt er een stukje over wat je je daarbij eigenlijk moet voorstellen.
Een voorbeeld:

Welkom in life

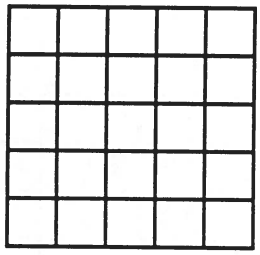
LIFE is een spel dat zich afspeelt in een bedachte wereld. In die wereld heersen bepaalde regels over geboren worden, sterfte en doorleven. Het speelt zich af op een rooster.



1e generatie



2e generatie



3e generatie

Hierboven zie je een bepaald patroon opgebouwd uit fiches die in een vakje, de cel staan. Het patroon ontwikkelt zich in een nieuw patroon volgens de volgende regels:

Regel 1 (over overleven): Een cel met een fiche die twee of drie buurcellen met fiches heeft blijft in leven. Het fiche blijft dan liggen.

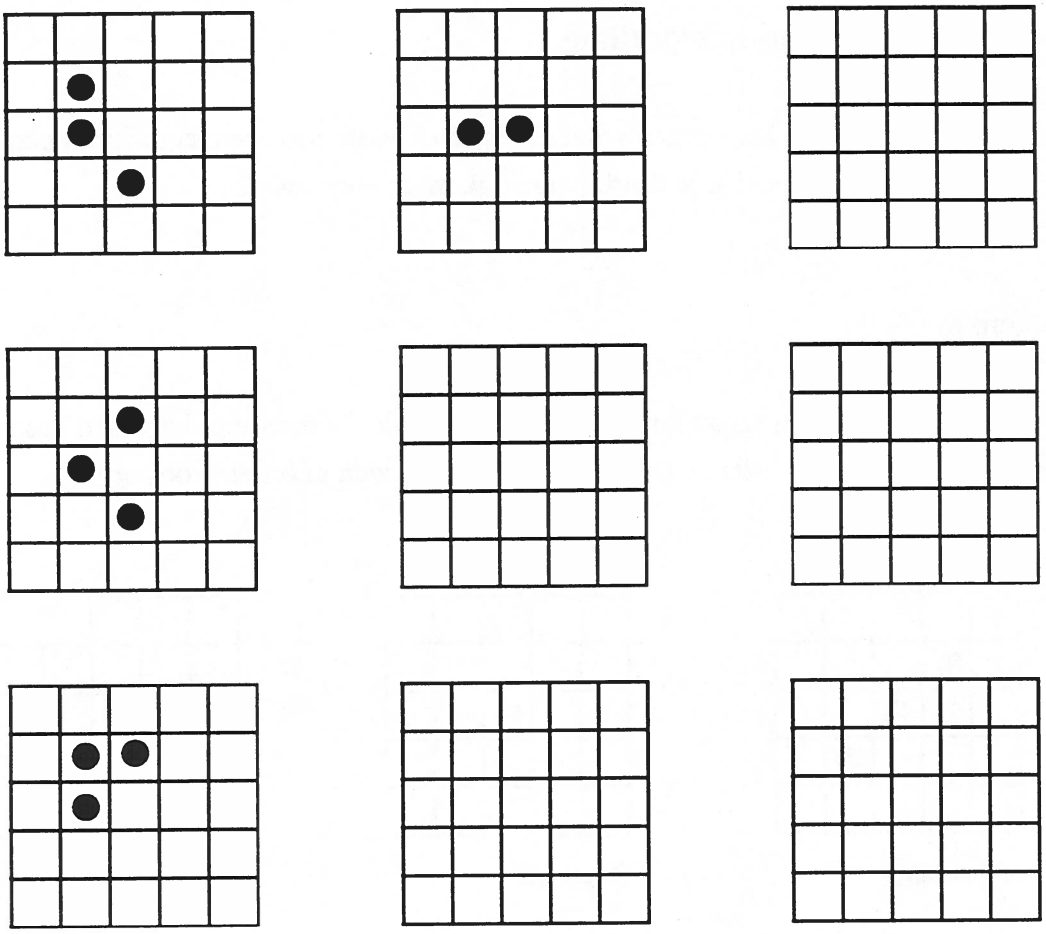
Regel 2 (over doodgaan):
a. Een fiche met geen of slechts één buurfiche sterft van eenzaamheid.
b. Een fiche met vier of meer buurfiches sterft wegens overbevolking.

Regel 3 (over geboren worden): In een leeg vakje wordt een fiche geboren als dit vakje omgeven is door precies drie buurfiches.

> Vul nu het rooster hierboven in om te zien hoe de derde generatie er uit ziet.

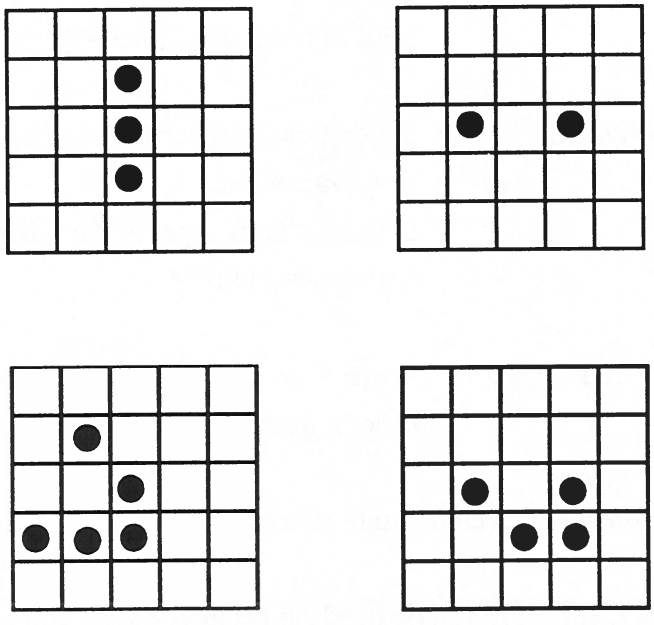
Van elk patroon is met behulp van de regels te bepalen wat de volgende generatie wordt.

> Maak van de patronen hieronder de volgende generatie.



1e generatie 2e generatie 3e generatie

> Bepaal of de patronen in de tweede generatie kunnen zijn ontstaan uit de eerste. Zo nee, waarom niet?



Als je volgens de regels werkt kom je vanzelf bij de volgende generatie. Het werken volgens die regels heet het werken volgens een algoritme.

In de volgende opdrachten kom je er nog meer tegen.

Een spel

Neem 31 lucifers. Je speelt met z'n tweeën. Ieder mag om de beurt een paar lucifers weghalen, minstens 1, maar niet meer dan 4.
Er ontbreekt nog een regel aan het spel: wie wordt de winnaar?

> Verzin zelf een regel hiervoor.

> Speel het spelletje eens een paar keer.

Hier zie je een stukje van zo'n spelletje:

	aantal gepakt	over
		--
speler 1	1	30
speler 2	3	27
speler 1	2	25
speler 2	3	22
speler 1	2	20
speler 2	4	16
speler 1	1	15
speler 2	1	14
speler 1	4	10
speler 2	2	8
speler 1	3	5

Mijn regel is: wie de laatste kan wegpakken wint het spel.

> Wie wint er nu?

Je kunt ook een andere regel kiezen, bijvoorbeeld: wie de laatste moet wegpakken heeft verloren.

> Wie wint er nu?

Kies deze laatste regel maar eens. Dus wie de laatste lucifer moet wegpakken heeft verloren.

Je ziet hier het laatste stukje van een spelletje:

	aantal gepakt	over
		10
speler 1	3	7

> Hoeveel moet je er nu wegpakken om er zeker van te zijn dat je wint?

Iemand beweert dat hij altijd kan winnen. Maar dan moet hij wel mogen beginnen.

> Klopt die bewering?

> Schrijf eens een manier op waarbij je zeker bent van winst.

De wolf, de geit en de kool.

Een boer had een wolf, een geit en een kool. Die konden elkaar, zoals je begrijpt, wel opvreten. De wolf vond de geit wel lekker en de geit zou de kool wel een kopje kleiner willen maken. Je kon ze dan ook niet met elkaar alleen laten. Dat kon wel met de wolf en de kool, want eeen wolf lust geen kolen. De boer moest zijn drie bezittingen van de ene kant van de rivier naar de andere kant brengen. Hij had daarvoor een roeibootje waar behalve hijzelf nog maar een ding in kon.

Dus eerst deze situatie, alles aan de ene kant:

deze kant		andere kant
wolf		--
geit	bootje --	
kool		--

Uiteindelijk moet het er zo uit zien, alles aan de andere kant:

deze kant	andere kant
--	wolf
--	geit
--	kool

> Hoe zou de boer dat moeten doen? Goed opletten dat ze elkaar niet opeten, ook niet aan de andere kant, als de boer weer op de terugtocht is.

> Beschrijf de manier (de strategie) die je gebruikt om dit spel tot een goed einde te brengen.

Als je eenmaal weet hoe dit spel en het vorige werken, dan kun je ze altijd tot een goed einde brengen. Je hebt een algoritme geleerd, zoals dat heet. Dat betekent een aantal vaste handelingen achter elkaar die steeds hetzelfde opleveren als je ze precies in de goede volgorde doet.

Nog meer algoritmen

Reken eens uit:

2391338 : 3178 =

Vergelijk de manieren van uitrekenen eens met anderen. Doen die het op dezelfde manier?
Als je volgens een vast patroon werkt bij dit soort sommen, kom je vanzelf bij het antwoord. Dit heet dus een delingsalgoritme. Alweer een algoritme dus.

Nog een paar voorbeelden:

> Probeer eens te vertellen hoe je je veters van je schoenen strikt.

Dat valt niet mee. Je kunt waarschijnlijk niet precies vertellen hoe je het doet, maar je doet het en het lukt altijd, als je de verschillende handelingen maar achter elkaar uitvoert.

> Neem een telefoonboek. Zoek eens op: J. de Bruin. Hoe doe je dat?

Een priemgetal is een getal dat je niet door een ander getal kunt delen. Dus bijvoorbeeld het getal 7 en het getal 23.

> Is 11 een priemgetal? en 39? en 71?

> Schrijf alle priemgetallen onder de 100 op.

> Hoe heb je dat gedaan? Kan het nog systematischer?

Een manier om dit heel systematisch te doen staat hieronder:

Stap 1 Schrijf alle getallen onder de 100 op. In een 10x10 vierkant is het handigst.
Stap 2 Streep alle getallen die deelbaar zijn door 2 eruit.
Stap 3 Streep nu alle getallen die deelbaar zijn door 3 eruit.
Stap 4 Streep alle getallen die deelbaar zijn door 5 eruit.

> Waarom hoef je niet meer alle getallen die deelbaar zijn door 4 eruit te schrappen?

> Welke getallen zullen er nu weggestreept moeten worden? Dus vul aan:
Stap 5 Streep alle getallen die deelbaar zijn door .... eruit.

> Waarom slaan we 6 over? Hoe komt het dat alle getallen die deelbaar zijn door 6 er al uit zijn?

Dit gaat net zo lang door tot alle getallen die door een ander getal gedeeld kunnen worden op zijn. Dan blijven de priemgetallen over.

> Je hebt gezien dat de getallen die deelbaar waren door 7 de laatste waren die weggestreept werden. Waarom zijn ze daarna allemaal op en blijven alleen de priemgetallen over?

Dit algoritme voor het vinden van de priemgetallen heet de zeef van *Eratosthenes*.
Eratosthenes was een Griek. Hij heeft ca. 250 voor Chr. deze methode verzonnen.

> Waarom wordt dit de **zeef** van Eratosthenes genoemd?

Nog een laatste algoritme:

> In het jaar 325 is afgesproken dat eerste paasdag zal vallen op de eerste zondag na de eerste volle maan na het begin van de lente. Met die regel kun je aardig de goede paasdatum berekenen, als je tenminste weet wanneer het volle maan is. In de middeleeuwen was de berekening van de paasdatum, *de computistiek*, het moeilijkste vak op school.

Tegenwoordig is er een betrekkelijk eenvoudig algoritme om de datum van eerste Paasdag te bepalen, dat op het eerste gezicht niets met de volle maan te maken heeft.
Het gaat zo:
Even een opmerking vooraf: bij delingen hieronder hoef je steeds alleen maar het deel voor de comma te nemen, dus als uit de deling 19,9 komt neem je 19.

Nu het algoritme:

stap 1 Deel het jaartal door 19, neem de rest van die deling en tel er 1 bij op.
stap 2 Deel het jaartal door 100 en tel er 1 bij op.
stap 3 Vermenigvuldig de uitkomst van stap 2 met 3, deel door 4 en trek er 12 af.
stap 4 Vermenigvuldig de uitkomst van 2 met 8, tel er 5 bij, deel die som door 25 en trek er 5 af.

stap 5 Vermenigvuldig het jaartal met 5, deel door 4, trek daar eerst de uitkomst van stap 3 en dan nog 10 af.

stap 6 Vermenigvuldig de uitkomst van stap 1 met 11, tel daar 20 bij op, tel de uitkomst van stap 4 er bij op en trek daar de uitkomst van stap 3 van af. Deel dat getal door 30 en neem de rest. Als die rest 24 is, of als de rest 25 is en de uitkomst van stap 1 was 11, tel dan 1 op bij de rest.

stap 7 Trek nu de uitkomst van stap 6 af van 44. Blijft er minder dan 21 over, tel er dan 30 bij.

stap 8 Tel de uitkomsten van stap 5 en stap 7 op, deel deze som door 7 en bepaal de rest. Trek die af van de uitkomst van g vermeerderd met 7.

Als de uitkomst van stap 8 kleiner is dan 31, valt Pasen op zoveel maart als de uitkomst van stap 8 aangeeft. Is die uitkomst van stap 8 groter dan 31 dan valt Pasen op zoveel april als de uitkomst van stap 8 min 31.

Een hele klus om dat allemaal te doen, maar gegarandeerd dat je de goede datum vindt.

> Op welke datum valt Pasen dit jaar?

> En in het jaar 2000?

Terugblik

Hierboven staan een heleboel voorbeelden van algoritmen genoemd.

> Noem zelf nog 3 andere voorbeelden van algoritmen die je kent.

> Waarom zijn het algoritmen?

1. .....
2. .....
3. .....

> Is er een algoritme om een voetbalwedstrijd te kunnen winnen?

> Is er een algoritme om een voldoende te halen voor je wiskundeproefwerk?

> Is er een algoritme om de weg terug te vinden in een doolhof?

plaatje van een doolhof

> Is er een algoritme om een schaakpartij te winnen?

> Kan een computer schaken?

> Schrijf eens een voordeel op van een algoritme. Kun je ook een nadeel noemen?

Nog even dit:

Een algoritme levert altijd het bedoelde resultaat als je het goed toepast. Dat betekent ook dat je ze meestal door een computer kunt laten uitvoeren. Sommige algoritmen zijn wel langzaam. Het is van belang algoritmen te vinden die snel en zuinig werken. Dan kan een computer snel het goede antwoord geven. Dan is er niet veel dure rekentijd nodig.
In het vervolg kijken we eens naar het **effect, de snelheid en de zuinigheid** van algoritmen.

Hoofdstuk 2: Alle rondreizen uitrekenen

Inleiding

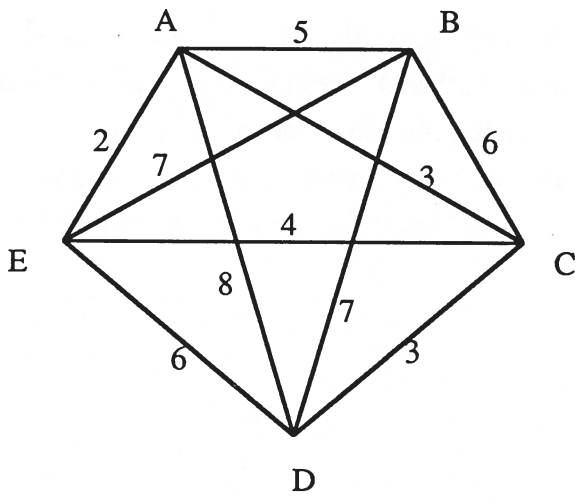
In een eerder pakket heb je kennisgemaakt met het probleem van de Kortste Rondreis langs een aantal plaatsen. Daar speelden algoritmen ook een belangrijke rol. Je hebt er twee van gezien met de computer:

- Het Dichtste-Buur-algoritme
- Het Deukalgoritme

Die komen later in dit pakket nog weer terug.

Maar je kunt zo'n rondreisprobleem natuurlijk ook anders aanpakken. Namelijk door alle mogelijke rondreizen uit te rekenen en dan te kijken wat de kortste is. In deze paragraaf gaan we na, hoeveel verschillende rondreizen er eigenlijk te maken zijn. En hoeveel tijd het zou kosten om al die rondreizen uit te rekenen. Dat zijn er heel veel, zoals we zullen zien. Ja zelfs zoveel dat de computer er zich haast in verslikt. Je krijgt dan ook een beter idee waarom men in de praktijk zo blij is met bijvoorbeeld het Deukalgoritme.

1. Vijf punten zijn met elkaar verbonden. De onderlinge afstanden staan in de graaf aangegeven.



a. Geef eens een voorbeeld van een rondreis en bereken de lengte ervan.

b. De rondreizen ABCDEA en CDEABC zijn dezelfde. Waarom?

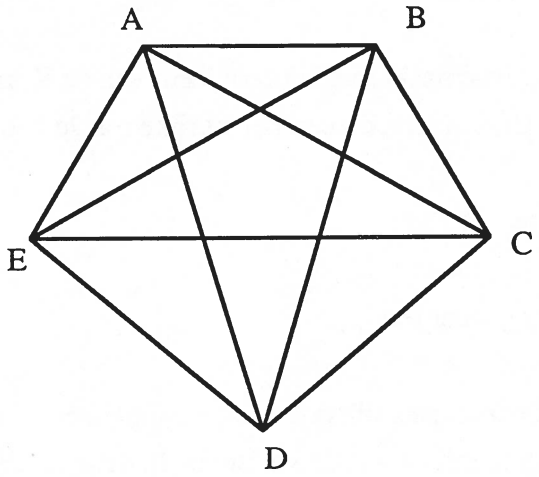
c. Hoeveel verschillende rondreizen zijn er?

d. Wat is volgens jou de kortste?

Systematisch tellen

Om het aantal verschillende rondreizen te bepalen, is het handig om systematisch te tellen. We bekijken nog eens de situatie met vijf punten. Voor het gemak laten we even de afstanden erbij weg.

Hieronder zie je een systematische aanpak van de vraag beschreven. Probeer het verhaal te volgen, zodat je het straks zelf kunt toepassen.



stap 0 Het begin- en eindpunt. Wat we kiezen, doet er niet veel toe. Daarvoor nemen we in dit voorbeeld A.

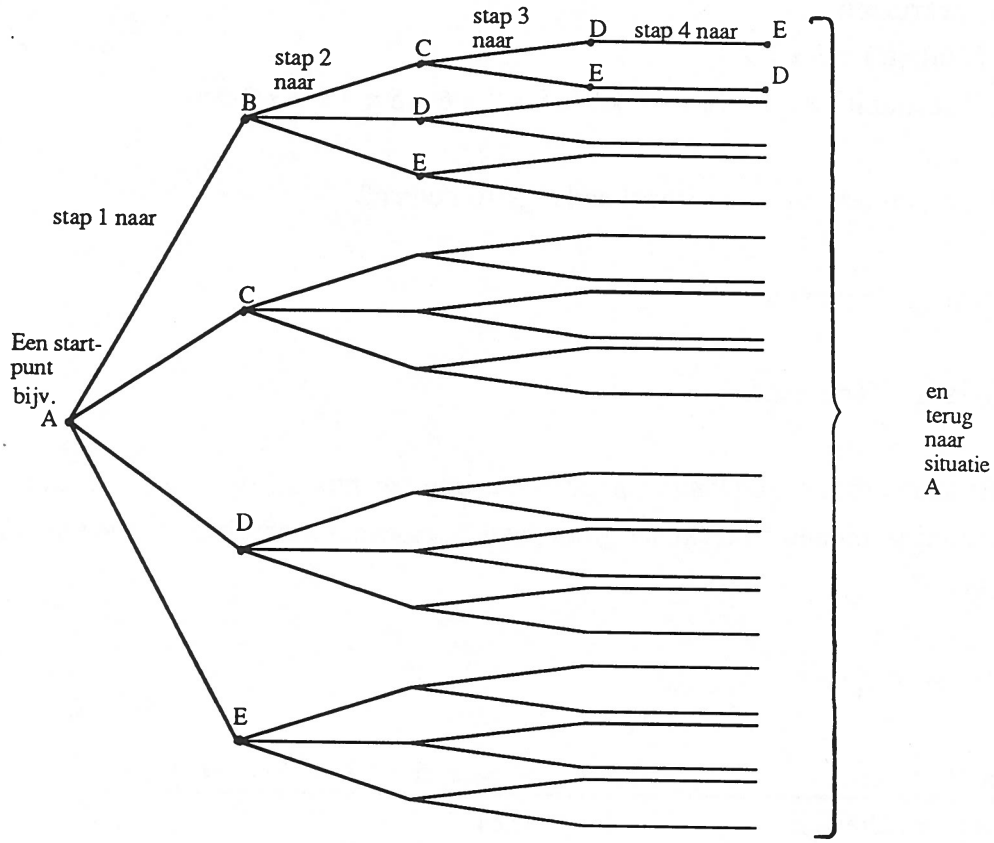
stap 1 Vanuit A kun je 4 kanten op, naar B, C, D of E.

stap 2 Als je in B, C, D of E bent aangeland, heb je steeds nog 3 mogelijkheden, om een nieuw punt te kiezen. Zodoende heb je voor de eerste twee stappen in totaal 4 x 3 mogelijkheden.

stap 3 Als je twee stappen hebt gemaakt, kun je steeds nog 2 nieuwe punten kiezen, om verder te gaan. Na ABC kun je bijvoorbeeld kiezen uit D of E als volgende. Zo krijg je in totaal 4x3x2 mogelijkheden voor de eerste drie stappen.

stap 4 Hierna is de keus niet groot meer: er is nog 1 punt over waar je niet bent geweest, en dan moet je weer terug naar A. --> Totaal aantal rondreizen tussen 5 punten wordt 4x3x2x1= 24.

Je kunt deze manier van systematisch tellen, als je wilt, ook met een boom uitschrijven. Kijk maar:

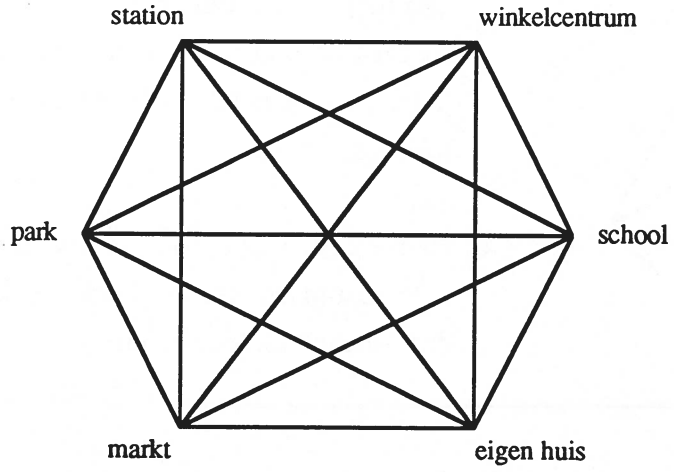


aantal takken = aantal rondreizen = 4 x 3 x 2 x 1

Oefenen met systematisch tellen

2. Hoeveel rondreizen kun je maken langs 4 punten?

3. Een ijsverkoper heeft 6 plaatsen in de stad waar hij zijn ijs wil uitventen. (zie graaf).



Hij neemt elke dag een andere route langs deze 6 plaatsen. En hij komt overal precies één keer. Start en finish liggen steeds bij zijn eigen huis.
Zou hij in de maanden juli en augustus alle mogelijke verschillende routes langs die 6 plaatsen kunnen nemen?

Opmerking: Op je rekenmachine zit een knopje [x!] (x!: spreek uit: 'x-faculteit'). Dat kun je hier handig gebruiken.
3! ('3 faculteit') = 3 x 2 x 1
12! ('12 faculteit') = 12 x 11 x 10 x 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1

4. Hoeveel rondreizen kun je maken langs 10 punten?

5. En langs 25 punten?

6. En langs 50 punten?

7. Het aantal rondreizen langs 100 punten is immens groot. We noemen het even x rondreizen, waarbij x een onuitspreekbaar groot getal is. Hoeveel rondreizen kun je nu maken langs 101 punten?

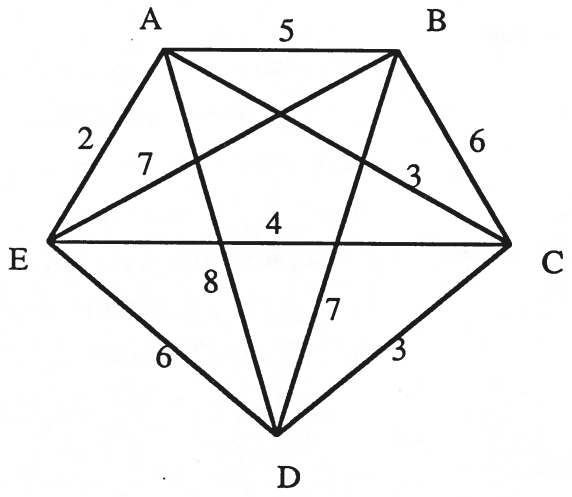
8. Vul de tabel in:

aantal punten	2	3	4	5	6	7	8	9	10
aantal rondreizen				24					

Rekentijd

Je weet nu hoeveel verschillende rondreizen er zijn langs een aantal punten. We gaan nu eens kijken, hoeveel tijd het kost om al die rondreizen uit te rekenen. Voor jezelf, en voor de computer.

9. We gaan nog eens terug naar de situatie van vraag 1.



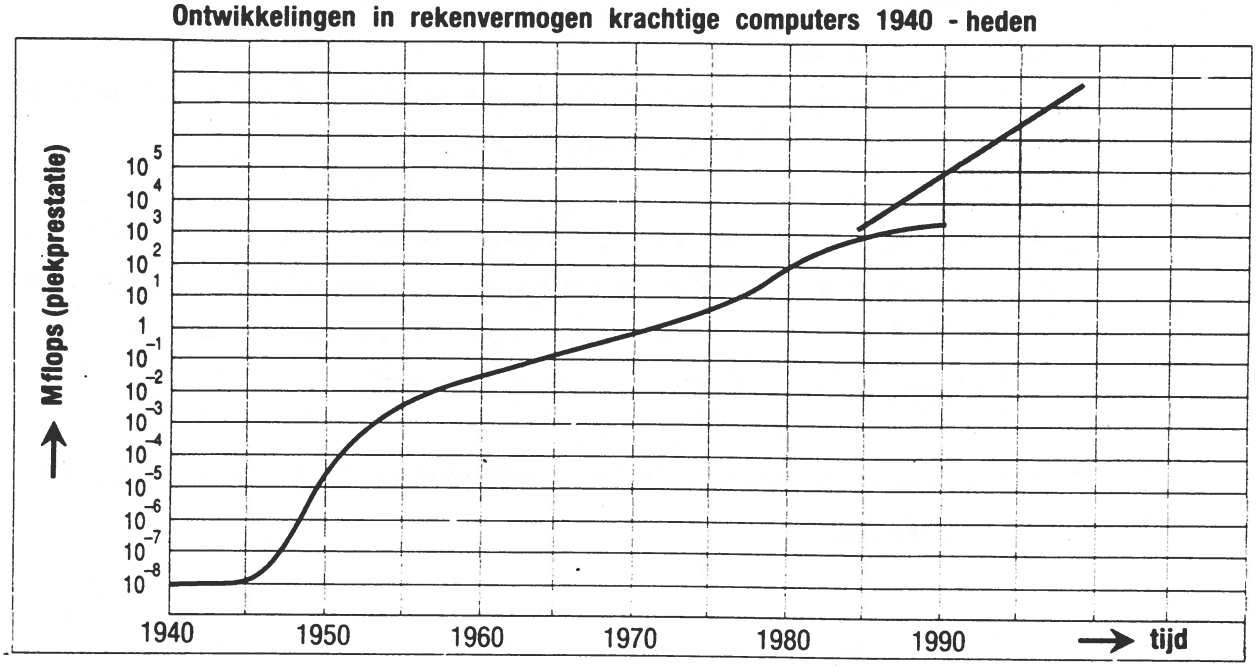
a. Een mogelijke rondreis hier is bijvoorbeeld ACDBEA.Neem eens op hoe lang je erover doet, om de lengte van deze rondreis uit te rekenen.

b. Hoeveel rekentijd zou jij ongeveer nodig hebben, om alle verschillende rondreizen in deze situatie uit te rekenen?

c. Als je de kortste rondreis wilt bepalen door alle mogelijke rondreizen na te rekenen en te

vergelijken, dan ben je er nog niet met het antwoord uit vraag b. Wat komt er nog aan extra activiteit bij?

10. Een computer rekent veel sneller dan een mens.
In de grafiek hieronder zie je hoe snel.



a. a1. In vergelijking met 1980 zijn de computers ..... keer zo snel geworden.
a2. Men verwacht dat ze in 2000 nog ..... keer zo snel zijn als in 1990.

b. Met een snelle computer uit 1990 wordt een probleem van een rondreis langs 50 plaatsen opgelost.
b1. Hoeveel verschillende rondreizen zijn er langs 50 plaatsen?
b2. Hoeveel optellingen moet de computer makenom één rondreis uit te rekenen?
b3. Hoeveel optellingen moet de computer makenom alle mogelijke rondreizen na te rekenen?
b4. Hoeveel tijd kost dat?

c. En hoeveel tijd kosten de berekeningen bij een rondreis langs 60 plaatsen?

In de praktijk kun je vereenvoudigingen aanbrengen in computerprogramma's die alle mogelijke rondreizen berekenen. Je kunt daarbij denken aan:
- Rondreizen in omgekeerde volgorde niet uitrekenen, want daar komt toch hetzelfde uit.
- Niet de hele berekening maken, als dat overbodig is.

enz.

Toch blijft het gigantisch veel tijd kosten, om voor veel punten, ook met een heel snelle computer, alles te berekenen.

11. Een computer lost een rondreisprobleem voor 30 punten in 1989 op in een half uur.
    a. Hoeveel tijd heeft deze computer ongeveer nodig om alle rondreizen voor 31 punten uit te rekenen?
    b. En voor 40 punten?
    c. 'Over 25 jaar zijn de computers wel 1000 keer zo snel als nu', zegt iemand. Hoeveel punten kan zo'n computer dan aan in een half uur?

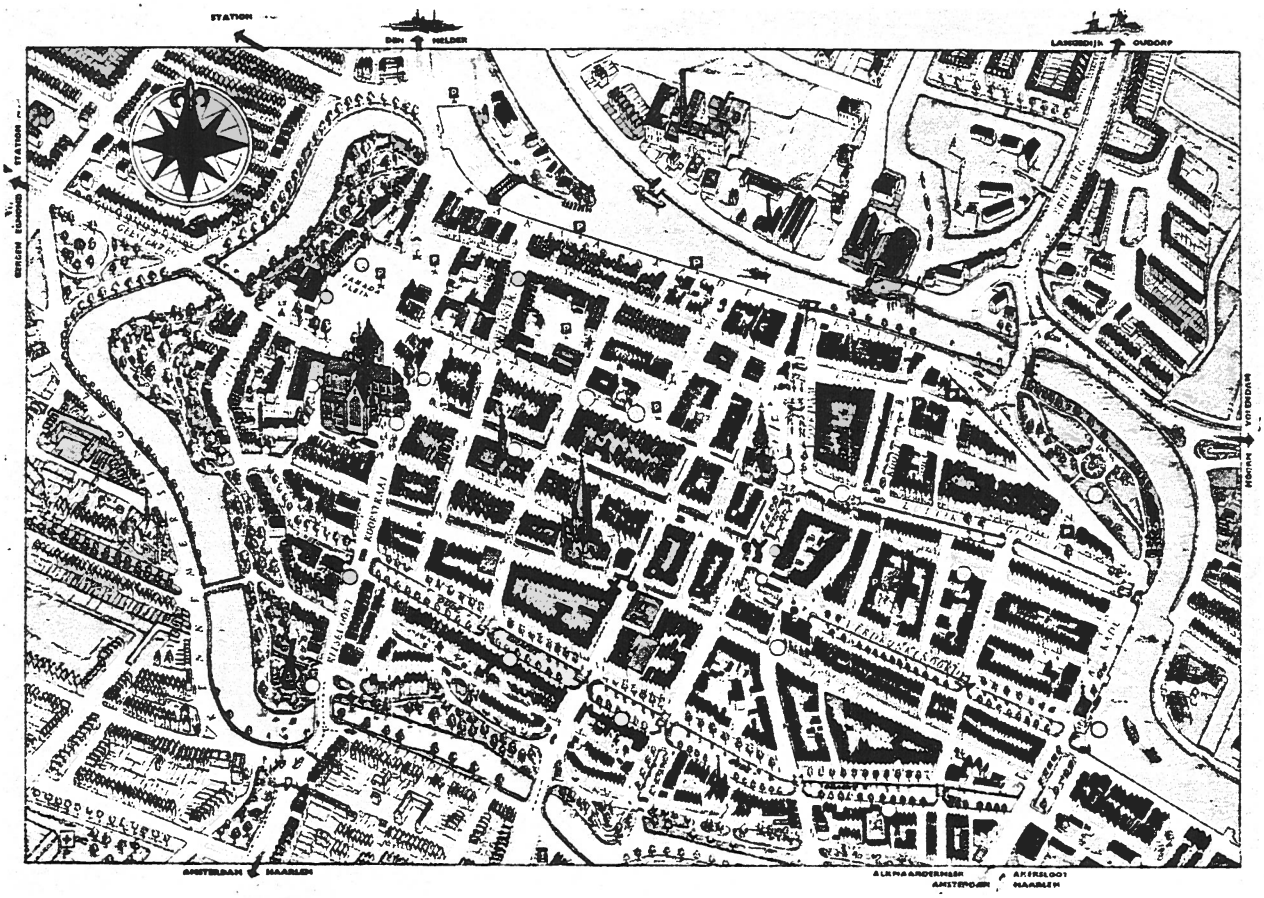
12. Wiskundeleraar XYZ geeft een demonstratie voor de klas over het rondreisprobleem. Zijn computer rekent alle mogelijke rondreizen langs 20 plaatsen uit in 5 minuten. 'Ach', zegt Saskia, 'Probeer het nou ook eens voor 25 punten, dat zijn er maar 5 meer'. En ze lacht daarbij heel venijnig. Wat is het gemene in Saskia's voorstel?

13. Het deukalgoritme geeft niet altijd de kortste rondreis. Toch wordt het vaak gebruikt in de praktijk. Waarom?

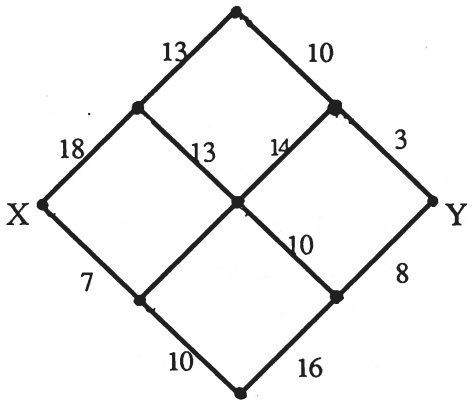
Hoofdstuk 3: Routes in een graaf

Inleiding

In dit hoofdstuk bekijken we de zuinigheid van het kortste-weg-algoritme. Dat is, de naam zegt het al, het algoritme om de kortste weg tussen twee punten in een graaf te bepalen.
We bepalen hoeveel berekeningen er nodig zijn om alle wegen tussen twee punten te berekenen en hoeveel berekeningen het algoritme vraagt. Zoals we zullen zien, levert het algoritme een gigantische besparing op als de problemen wat groter worden.



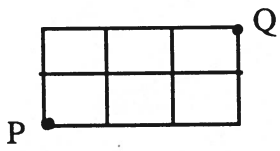
Je bent als tourist in Alkmaar. Je wilt op zeker moment van X naar Y. Hoeveel gevels zou je onderweg kunnen zien als je zonder omwegen loopt? In de graaf hieronder, staan de aantallen per straatdeel.



a. Noteer de getallen ook in de plattegrond op de vorige bladzijde
b. Op hoeveel manieren kun je, zonder omwegen, van X naar Y?
c. Hoeveel gevels kom je op die verschillende routes tegen?
d. Wat is de route met de meeste gevels?
e. Had je deze vraag ook met het-kortste-weg-algoritme kunnen beantwoorden??

Systematisch tellen

Om de zuinigheid van het kortste-weg-algoritme te bepalen, moet je weten hoeveel wegen er zijn tussen twee punten van een graaf. Hieronder zie je twee manieren om dit te bepalen beschreven. Ze gaan over deze situatie:

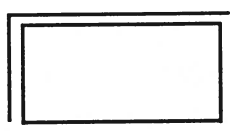


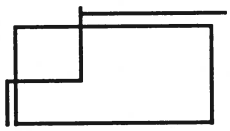
2. Hoeveel wegen (zonder omweg) zijn er van P naar Q?

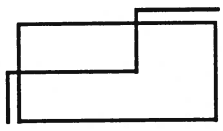
Bestudeer beide manieren, tot je ze goed snapt.

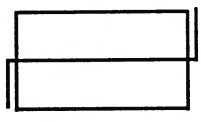
<u>Manier 1: wegen tekenen</u>

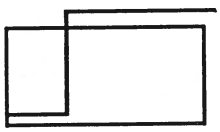
We tekenen systematisch alle wegen (kun je het systeem ontdekken?).

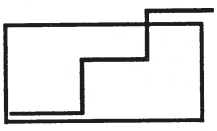

stap 1
| | - - -

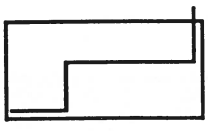

stap 2
| - | - -

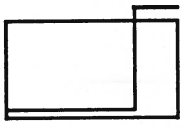

stap 3
| - - | -

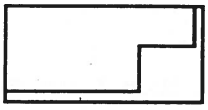

stap 4
| - - - |


stap 5
- | | - -

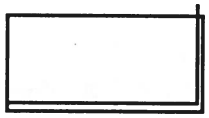

stap 6
- | - | -


stap 7
- | - - |


stap 8
- - | | -

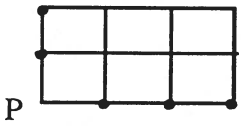


stap 9
- - | - |

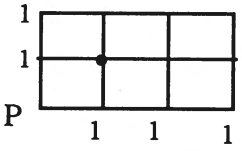


stap 10
- - - | |

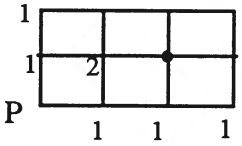
Manier 2: Tellen en redeneren



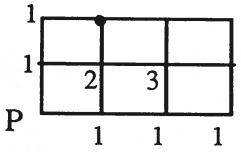
Vanuit P kun je op precies één kortste manier in de punten met een stip komen.



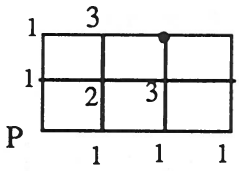
Vanuit P kun je op 1+1=2 manieren bij de stip komen.



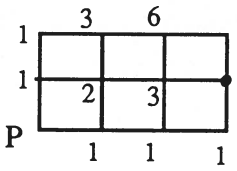
Vanuit P kun je op 1+2=3 manieren bij de stip komen.



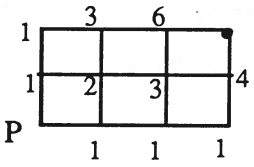
Vanuit P kun je op 2+1=3 manieren bij de stip komen.



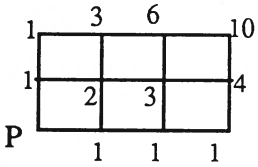
Vanuit P kun je op 3+3=6 manieren bij de stip komen.



Vanuit P kun je op 3+1=4 manieren bij bij de stip komen.



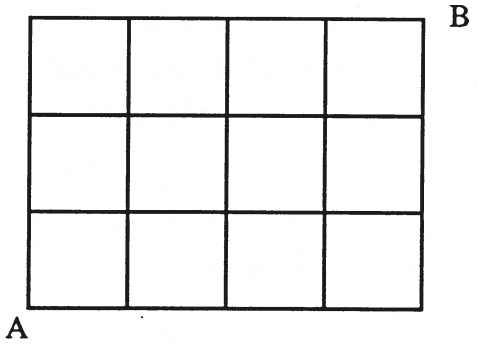
Vanuit P kun je op 6+4=10 manieren bij bij de stip komen



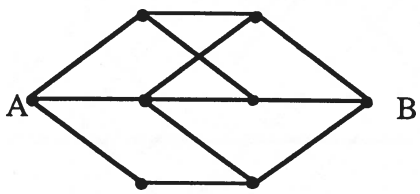
Overzicht tellen en redeneren: 10 wegen van P naar Q.

Oefenen met systematisch tellen

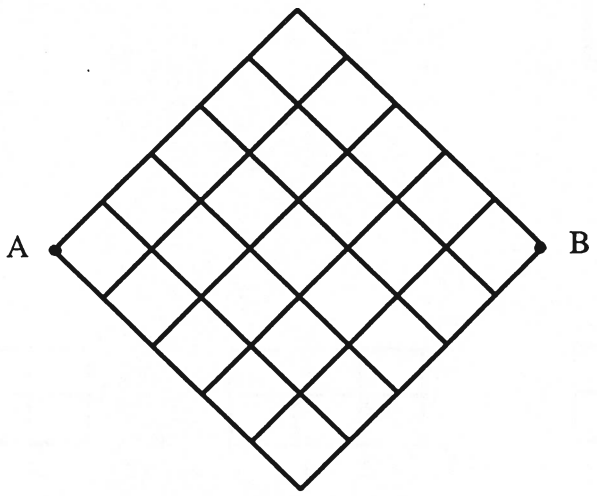
3. Op hoeveel manieren kun je van A naar B gaan, als je alleeen vooruit mag reizen in de graaf? Schrijf je oplosmanier bij je antwoord op.



van A naar B



van A naar B



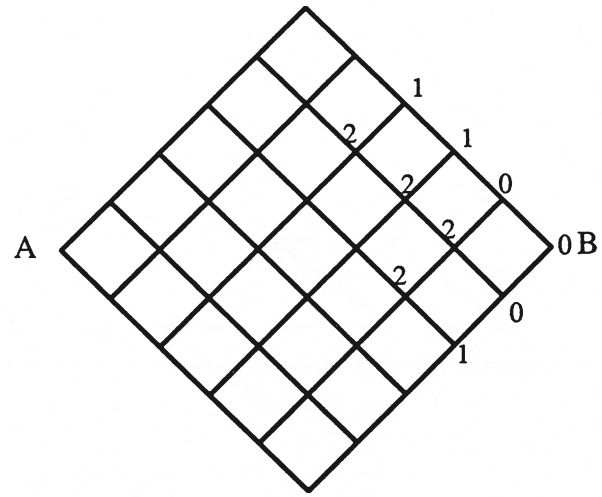
4. Hoeveel verschillende wegen zijn er van A naar B?

Rekentijd

Je kunt nu uitrekenen hoeveel routes er zijn tussen twee punten van een graaf. Om de kortste weg te bepalen kun je de lengte van al die routes uitrekenen en met elkaar vergelijken. We gaan na hoeveel berekeningen dit vraagt. Verder bepalen we het aantal berekeningen, als je met het kortste-weg-algoritme werkt.

5. Bekijk de situatie van opgave 4.
   a. Hoeveel wegen waren er van A naar B?
   b. Hoeveel optellingen moet je maken, om de lengte van één zo'n weg van A naar B uit te rekenen?
   c. Hoeveel optellingen heb je nodig om alle wegen van A naar B uit te rekenen?

6. In de graaf hieronder staat het aantal optellingen dat je maakt in een bepaald punt, bij toepassen van het kortste-weg-algoritme.



a. Leg de getallen uit.
b. Vul het schema verder aan
c. Hoeveel optellingen vraagt het kortste-weg-algorimte in totaal?

7. In vraag 5 en 6 heb je het aantal optellingen bepaald bij de kortste weg in een 5x5-rooster. Doe hetzelfde voor andere roosters en vul de resultaten in in de tabel hieronder.

grootte rooster	aantal roosterpunten	aantal berekeningen bij alle routes	aantal berekeningen bij algoritmes
1 x 1	2 x 2	2	2
2 x 2	3 x 3	18	10
3 x 3	4 x 4	.....	22
4 x 4	5 x 5	.....	.....
5 x 5	6 x 6	.....	.....
6 x 6	7 x 7	10164	.....
7 x 7	8 x 8	.....	.....
8 x 8	9 x 9	193050	.....
9 x 9	10 x 10	826540	.....
10 x 10	11 x 11	3510364	.....

8. De rekensnelheid van een computer is ..... berekeningen/seconde. Hoe lang rekent die computer over het kortste-weg-probleem
   a. als alle routes worden berekend?
   b. als het algoritme wordt gebruikt?

Je ziet dat de verschillen steeds groter worden. Bij heel grote getallen kun je je voorstellen dat de verschillen enorm zijn en dat je dus altijd manier II zult gebruiken.

Voor heel grote getallen worden computers gebruikt. De tijd die computers moeten rekenen is heel kostbaar. Het is daarom van belang snelle rekenmethoden te gebruiken. Nu dat kortste-weg-algoritme is er een, en wordt heel veel gebrukt bij situaties waar een rijroute moet worden bepaald, van vrachtauto's bijvoorbeeld.

Hoofdstuk 4: Hoe goed is een algoritme (computerpracticum)

Het kortste weg algoritme is, zoals we gezien hebben, een heel goed algoritme. Het geeft een duidelijk voorschrijft om op een snelle manier inderdaad de kortste weg te vinden.
Bij rondreizen ligt de zaak ingewikkelder.

1. a. Welke twee algoritmen voor rondreizen ken je?
   b. Geven deze algoritmen duidelijke oplosvoorschriften?
   c. Werken de algoritmen snel?
   d. Geven de algoritmen steeds de kortste rondreis?

In het computerpracticum hierna, gaan we eens kijken hoe snel en hoe goed de algoritmen zijn. We vergelijken dat met uitputtend onderzoek, maar ook met hoe snel en hoe goed we zelf zijn.

Computerpracticum

1. Start het programma 'De Handelsreiziger'. Haal het bestand 'OU.PNT' op en toets de eigen rondreis in, waarvan je zelf denkt dat die de kortste is. Bepaal ook de kortste rondreis met de buurmethode en de deukmethode. En tot slot met de exacte methode. Dit laatste staat voor uitputtend onderzoek. Vul op de vraag of je een bovengrens weet 'n' (nee) in.
   Vul de tabel hieronder in (gebruik de F2-toets):

methode	lengte	route	tijd
eigen rondreis			
dichtste buur			
grootste hoek			
exacte methode			

2. Kies een paar keer een random aantal punten en maak daarbij een tabel als bij opgave 1.
3. Wat kun je zeggen over de waarde van de verschillende manieren? Wanneer ben je zelf beter, wanneer zijn de algoritmen beter, enz.?

Conclusie

De conclusie is, dat het ondoenlijk is om alle mogelijkheden te proberen en met elkaar te vergelijken. Toch kon je met de exacte methode uit het computerprogramma gegarandeerd de allerkortste rondreis vinden. Dat komt omdat bij die methode flink bespaard wordt op het systematisch proberen van álle

mogelijke rondreizen. De bovengrens speelt bij dat zoeken een belangrijke rol. Zodra blijkt dat (het begin van) een rondreis de bovengrens zal gaan overschrijden, wordt met dat beginstuk niet verder meer gezocht. Dat wordt dan immers niks meer. Maar zelfs met deze besparingen, blijft het erg veel werk om écht de kortste rondreis te vinden. En dat terwijl het op het oog toch niet zo moeilijk is om een rondreis 'te zien'. Daarvoor is wél nodig dat de punten op een rooster liggen of dat je een goede kaart hebt met alles netjes op schaal getekend.

archief FI 02.01.20

Algoritmiek

Heuvel, G. van den , H. Krabbendam