Activiteit 14

Toeristen stad — Dominerende verzamelingen

Samenvatting

Veel situaties uit het dagelijks leven kunnen weergegeven worden in de vorm van een netwerk of 'graaf', zoals gebruikt bij de kleuropdracht van activiteit 13. Het weergeven van netwerken laat veel mogelijkheden zien voor het ontwikkelen van algoritmes die praktisch bruikbaar zijn. In deze activiteit willen we een aantal kruispunten, of 'knopen' markeren op zo'n manier dat alle andere knopen maximaal een stap verwijderd zijn van de gemarkeerde knopen. De vraag is, met hoe weinig gemarkeerde knopen kunnen we dit doen? Dit zal een verrassend moeilijk opdracht blijken.

Curriculum Links

- Rekenen vanaf groep 5: positie en oriëntatie

Vaardigheden

- Kaarten.
- Relaties.
- Probleem oplossen
- Iteratief doel benaderen.

Leeftijd

- 7 jaar en ouder

Materialen

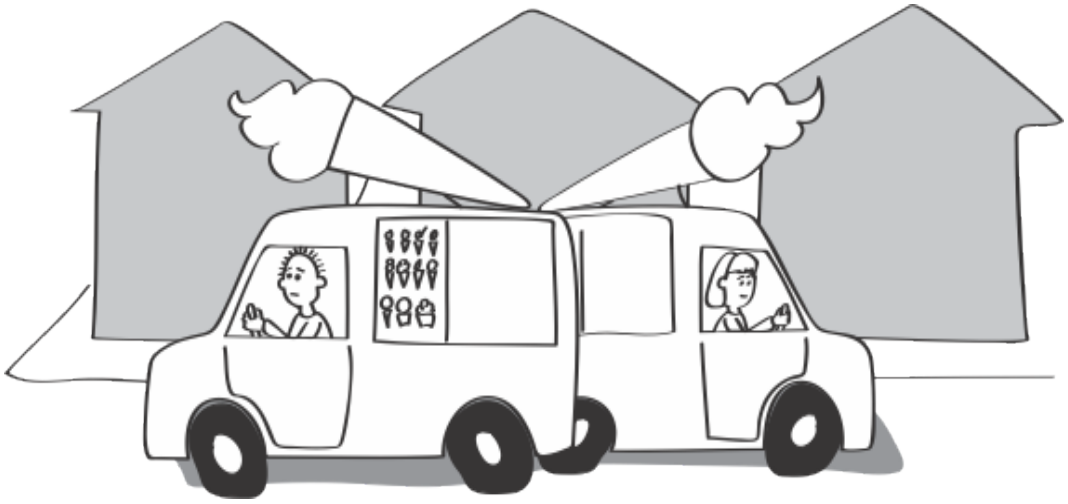
Iedere groep leerlingen heeft nodig:

- Een kopie van werkblad "ijscowagens"
- een aantal pionnen, munten of fiches van twee verschillenden kleuren.

De leerkracht heeft nodig

Een digitale versie van de oplossing van het werkblad "ijscowagens" voor op het digibord of een whiteboard om het op te tekenen.

Alleen te kopiëren voor educatieve doeleinden.
© 2012 Computer Science Unplugged (csunplugged.org)



Dominerende verzamelingen

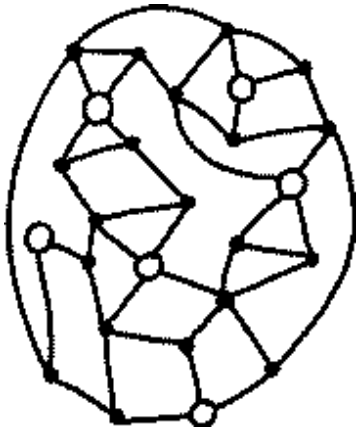
Introductie

Op het IJscowagentjes-werkblad staat een kaart van Toeristenstad. De lijnen zijn straten en de punten zijn straathoeken. De stad ligt in een belachelijk heet land, en in de zomer staan ijscowagens op straathoeken om ijsjes te verkopen aan toeristen. We willen de wagentjes zo neerzetten dat iedereen maximaal zijn of haar straat uit moet lopen en dan nog een huizenblok verder. (Het kan makkelijker zijn om je voor te stellen dat de mensen allemaal op een kruispunt wonen dan in de straten; dan moeten ze dus maximaal naar een andere straathoek moeten kunnen lopen voor een ijsje). De vraag is, hoeveel ijscowagens zijn er nodig en op welke hoeken moeten ze neergezet worden?

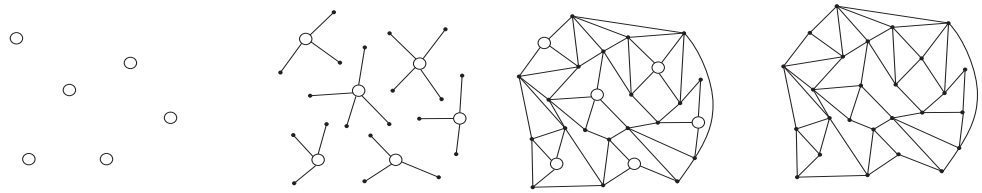
Discussie

- Verdeel de leerlingen in kleine groepjes, geef ze allemaal een kaart van Toeristenstad en wat fiches of iets dergelijks, en leg ze het probleem voor.
- Laat de leerlingen zien hoe je een fiche op een hoek neerlegt om een ijscowagen te markeren, en plaats dan anders gekleurde fiches op de volgende hoeken (knopen). Mensen die op deze kruispunten wonen (of in de straat die op dit kruispunt uitkomt) kunnen dus naar deze ijscowagen.
- Laat de leerlingen experimenteren met verschillende plekken voor de ijscowagens. Als ze oplossingen vinden om alle inwoners van ijs te voorzien, help ze dan herinneren dat een ijscowagen veel geld en energie kost en dat ze er zo min mogelijk van moeten gebruiken. Het is natuurlijk duidelijk dat je iedereen van ijs kan voorzien als je op iedere hoek een ijscowagen zet - de interessante vraag is hier natuurlijk, hoe kan je met zo min mogelijk wagens de stad van ijs voorzien.

Alleen te kopiëren voor educatieve doeleinden.
© 2012 Computer Science Unplugged (csunplugged.org)



- Het minimale aantal wagens in Toeristenstad is zes en de oplossing staat hierboven. Maar het is erg moeilijk om deze oplossing te vinden! Vertel na een tijdje puzzelen dat zes wagens genoeg zijn en daag ze uit om deze op de juiste manier te plaatsen. Dit blijft een moeilijk probleem: de meeste groepjes zullen het uiteindelijk opgeven. Zelfs de oplossing met acht of negen wagens kan heel lastig te vinden zijn.
- De kaart van Toeristenstad is gemaakt door (1) eerst zes punten op de kaart te tekenen, waar uiteraard allemaal ijscowagens op gezet werden. Deze werden verbonden met heel veel extra straten (2) en kruispunten om de oplossing te verbergen. Vervolgens willekeurig alle nieuwe kruispunten met elkaar verbinden (3). Het belangrijkste hierbij is om geen straten te tekenen tussen de kruispunten met ijscowagens (open cirkels) onderling, maar alleen tussen de extra kruispunten (dichte



1 *2* *3* *4*

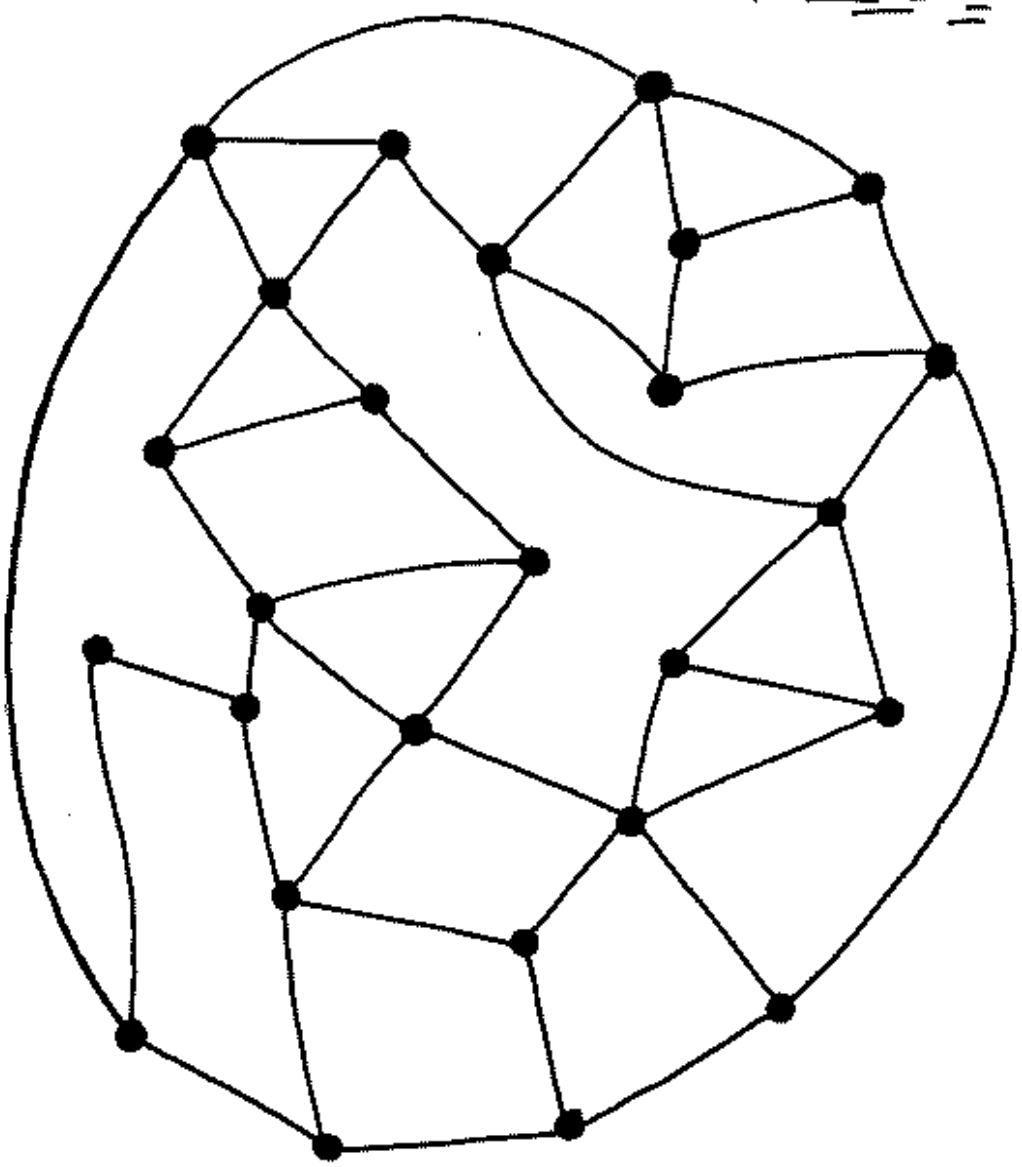
cirkels) en de kruispunten met een ijscowagen. Teken alle kruispunten en straten op dezelfde manier en je komt tot een ingewikkelde puzzel (4). Laat de klas deze techniek zien op het bord.

- Laat de leerlingen nu hun eigen ingewikkelde kaarten maken met deze strategie. Ze kunnen deze testen op vrienden of hun ouders en zullen zo uitvinden dat ze puzzels kunnen maken die zij kunnen oplossen, maar niemand anders. Dit zijn voorbeelden van wat we een "éénrichtingsfunctie" noemen: het is makkelijk om een puzzel te maken die voor anderen heel moeilijk is op te lossen - eigenlijk alleen als jij degene bent die hem heeft gemaakt. Eénrichtingfuncties spelen een cruciale rol in cryptografie (zie activiteiten 17 en 18).

Alleen te kopiëren voor educatieve doeleinden.
© 2012 Computer Science Unplugged (csunplugged.org)

Werkblad: IJscowagens

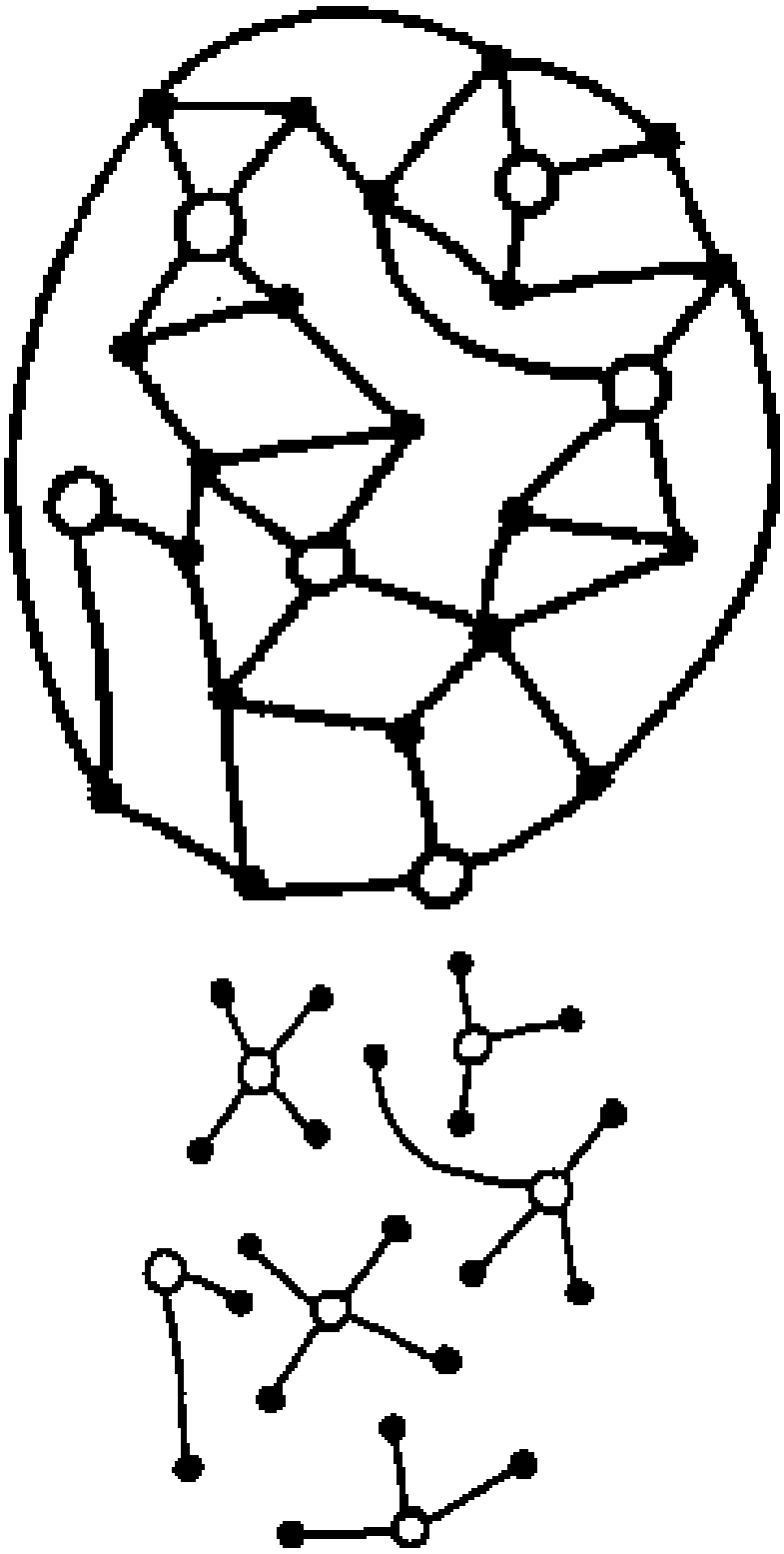
Vind uit hoe je ijscowagens op de kruispunten moet zetten zo dat ieder kruispunt maximaal een straat van een ijscowagen verwijderd is.



Alleen te kopiëren voor educatieve doeleinden.
© 2012 Computer Science Unplugged (csunplugged.org)

Oplossing: IJscowagens

Laat deze grafen aan de klas zien om te laten zien hoe de puzzel is gemaakt.



Alleen te kopiëren voor educatieve doeleinden.
© 2012 Computer Science Unplugged (csunplugged.org)

Variaties en uitbreidingen

Er zijn talloze situaties te bedenken waarin je dit soort problemen tegenkomt bij het bouwen en onderhouden van een stad: waar moeten de brievenbussen of prullenbakken, brandweerkazernes, winkels enzovoort. En in het dagelijkse leven zal de kaart niet gemaakt zijn met de truc die in deze activiteit is behandeld en hebben we daarmee geen oplossing bij de hand. Als je zo'n echt probleem zou moeten oplossen, hoe zou je dat doen?

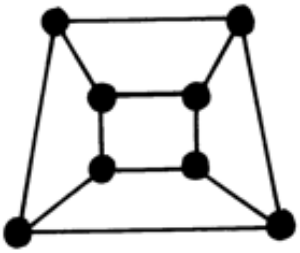
Er is een heel directe manier: ga alle mogelijke plekken van het plaatsen van een ijscowagen af en check welke oplossing het beste is. Met 26 straathoeken in Toeristenstad zijn er 26 manieren om een wagen te plaatsen. Het is makkelijk om alle 26 mogelijke plekken uit te proberen en het zal ook meteen duidelijk zijn dat geen een mogelijkheid een oplossing biedt aan het gestelde probleem. Met twee wagens zijn er 26 plekken voor de eerste wagen en welke er ook wordt gekozen, er blijven 25 plekken over voor de tweede wagen (je zal nooit twee ijscowagens op hetzelfde kruispunt zetten): 26 x 25 = 650 mogelijkheden om te checken. En opnieuw, het is makkelijk om iedere mogelijkheid te checken, maar het kost zo veel tijd. En beter nog, je hoeft eigenlijk maar de helft te checken (325) omdat het natuurlijk niet uitmaakt welke wagen er staat: als je wagen een op kruispunt A en wagen twee op kruispunt B hebt gecheckt, hoef je niet ook nog wagen een op kruispunt B en wagen twee op kruispunt A te checken. Je zou nog door kunnen gaan met drie wagens (2600 mogelijkheden), vier wagens (14950 mogelijkheden) en verder. Het zal duidelijk zijn dat 26 wagens sowieso genoeg is, omdat er 26 kruispunten zijn. Je kan het aantal mogelijkheden met alle mogelijke hoeveelheden wagens ook makkelijk uitrekenen. Er zijn 26 kruispunten en twee mogelijkheden voor iedere straathoek, er staat wel of er staat geen wagen, dat maakt 2^{26} (= 2 * 2 * 2 * ... * 2 [26 tweeën]), oftewel 67.108.864 mogelijkheden.

Deze manier van oplossen heet een "brute kracht" algoritme en duurt erg lang. Het is een misvatting dat computers zo snel zijn dat ze vrijwel ieder probleem zo kunnen oplossen, hoeveel werk het ook is. Dat is zeker niet het geval. Hoe lang een brute kracht algoritme duurt hangt af hoe lang het duurt om te controleren of een bepaalde keuze ook een oplossing is. Om dit te controleren moet de computer ieder kruispunt controleren om de afstand tot dichtstbijzijnde wagen te weten. Stel dat de hele setting in een seconde gecontroleerd kan worden. Hoe lang duurt het dan om alle 2^{26} mogelijkheden voor Toeristenstad te checken? (Antwoord: 2^{26} is ongeveer 67 miljoen; er zitten 86.400 seconden in een dag, dus 2^{26} seconden is ongeveer 777 dagen, ongeveer twee jaar.) En stel nu voor dat het geen seconde duurt, maar een honderdste van een seconde om iedere keuze te controleren. Dan zou het dezelfde twee jaar duren voor een computer om een stad met 36 kruispunten te controleren, omdat 2^{36} ongeveer 1000 keer zo groot is als 2^{26}. Zelfs als de computer een miljoen keer sneller zou zijn, zodat een miljoen mogelijkheden per seconde zouden kunnen worden gecontroleerd, dan zou het nog twee jaar duren bij een stad met

Alleen te kopiëren voor educatieve doeleinden.
© 2012 Computer Science Unplugged (csunplugged.org)

46 kruispunten. En dit zijn niet hele grote steden! (Hoeveel kruispunten telt jou stad?)

Nu het brute kracht algoritme te traag is, vragen we ons af of er andere oplossingen zijn om het probleem op te lossen? Nou, we kunnen de benadering gebruiken die succesvol was in Modderstad (activiteit 9). Maar nu moeten we niet de kortste weg vinden, maar juist het knooppunt met de meeste wegen. We beginnen met een ijscowagen te plaatsen op het 'drukste' kruispunt (waar de meeste wegen samenkomen), dan op het een-na-drukste kruispunt een wagen enzovoort. Hoewel dit niet zal betekenen dat dit ook zal leiden tot het gebruik van de minste ijscowagens, sterker nog het drukste kruispunt uit Toeristenstad, waar vijf wegen bij elkaar komen, is in dit geval helemaal geen goede plek (controleer dit met de klas).



Laten we kijken naar een makkelijker probleem. We worden nu niet gevraagd om de oplossing te vinden voor het minimale aantal ijscowagens. Maar we krijgen nu een oplossing en worden gevraagd of dit de minimale oplossing is of niet. In sommige gevallen is dit makkelijk. Bijvoorbeeld laat dit diagram een veel simpelere kaart zien waar de oplossing vrij voor de hand ligt. Als je de straten voorstelt als zijden van een kubus dan is het meteen duidelijk dat er twee ijscowagens op diagonaal tegenover elkaar liggende hoeken genoeg zijn. Hoe dan ook, je zal je nog steeds moeten afvragen of het niet mogelijk is om het probleem met minder ijscowagens op te lossen. Het is veel moeilijker, zo niet onmogelijk, om jezelf te overtuigen dat Toeristenstad niet met minder dan zes wagens toe kan. Voor de meeste kaarten is het heel erg moeilijk te bewijzen of een bepaalde oplossing met ijscowagens, de minimale oplossing is.

Waar gaat dit eigenlijk over?

Een van de meest interessante aspecten van het ijscowagenprobleem is dat niemand weet of er een algoritme bestaat om het minimale aantal locaties te vinden dat significant sneller is dan de brute-kracht-methode! De tijd die het kost om een minimaal aantal locaties te vinden met de brute-kracht methode groeit exponentieel met de toename van het aantal kruispunten - dit heet een exponentieel-tijd-algoritme. Een polynomiaal-tijd-algoritme is er een waarbij de tijd voor oplossen begrensd is door een kwadraat, een derde, zeventiende of iedere andere macht, van het aantal kruispunten.

Een polynomiaal-tijd-algoritme zal altijd sneller zijn voor grote kaarten - zelfs bij, laten we zeggen een zeventiende macht algoritme - omdat een exponentieel groeiende functie bij voldoende groot argument iedere polynomiaal groeiende functie de baas is (Als voorbeeld : voor n groter dan 117 geldt dat 2^{117} groter is dan n^{17}).

Alleen te kopiëren voor educatieve doeleinden.
© 2012 Computer Science Unplugged (csunplugged.org)

Bestaat er een polynomiaal-tijd-algoritme voor het vinden van een minimaal benodigde aantal locaties? Niemand weet het, hoewel genoeg mensen het hebben proberen te vinden. En hetzelfde is waar voor een beduidende kleinere taak zoals het controleren of een bepaald aantal locaties de minimale oplossing is: de brute-kracht methode om gewoon alle mogelijkheden na te gaan voor kleine hoeveelheden locaties blijft exponentiële tijd kosten ten opzichte van het aantal kruispunten, en polynomiaal-tijd algoritmes zijn nog niet gevonden. Het is onbekend of we verder moeten zoeken of dat ze gewoon niet bestaan.

Doet dit je herinneren aan het kaartenkleurprobleem (Activiteit 13)? Als het goed is wel. Het ijscowagenprobleem, dat officieel het minimaal dominerende verzamelingen probleem heet, is een van een groot aantal - wel duizenden - problemen waarvan wij niet weten of er een polynomiaal-tijd-algoritme bestaat. Deze problemen zijn er op het gebied van logica, bij legpuzzelachtige rangschikkingsproblemen als het kaartkleuren, bij het vinden van de optimale route op kaarten, en bij het maken van dienstregelingen of roosters. Verbazingwekkend genoeg zijn al deze op het oog verschillende problemen gelijk in de zin van dat als er een polynomiaal-tijd algoritme voor een van deze problemen wordt gevonden, deze relatief makkelijk omgezet kan worden tot een oplossingsalgoritme voor alle andere problemen - je kan zeggen dat ze samen sterk of zwak staan.

Dit soort problemen heten NP-volledig. NP staat voor “non-deterministisch polynominaal”. Dit jargon betekent dat het probleem kan worden opgelost in een redelijke tijd als je een computer zou hebben gehad die een willekeurig groot aantal oplossingen in één keer (dat is het non-deterministisch deel, niks ligt nog vast) uit zou kunnen proberen. Je zou kunnen denken dat dit een aardig onrealistische aanname is en dat is het ook. Het is niet mogelijk om zo’n soort computer te bouwen, omdat deze computer ook willekeurig groot zou moeten zijn! Toch is het concept van zo’n machine in principe belangrijk, omdat het laat zien dat NP-volledige problemen dus niet opgelost kunnen worden in een redelijke tijd zonder een non-deterministische computer.

Verder heet deze groep van problemen volledig omdat de problemen misschien wel heel verschillend lijken - bijvoorbeeld, het kaartkleurprobleem is een heel ander soort probleem dan het ijscowagenprobleem - maar het blijkt dat als er een efficiënte manier wordt gevonden om één van deze problemen op te lossen, dat deze methode gebruikt kan worden om al deze problemen op te lossen. Dat is wat we bedoelen dat ze samen sterk of zwak staan.

Er zijn duizenden van deze NP-volledige problemen en onderzoekers proberen al decennia deze problemen zonder succes aan te pakken. Als er een efficiënte oplossing gevonden zou worden voor één van hen, hebben we een oplossing voor allemaal. En omdat onderzoekers er zonder resultaat al zo lang mee bezig zijn, is het sterke vermoeden dat er geen

Alleen te kopiëren voor educatieve doeleinden.
© 2012 Computer Science Unplugged (csunplugged.org)

efficiënte oplossing bestaat. Maar het bewijs dat deze problemen alleen maar met een exponentiële tijd opgelost kunnen worden is vandaag de dag de meest prominente vraag in de theoretische computerwetenschap en waarschijnlijk van alle wiskunde.

Verder lezen

Harel's boek Algorithmics behandelt verschillende NP-volledige problemen en de vraag of er polynomiaal-tijd algoritmes bestaan. Dewdneys's Turing Omnibus gaat ook over NP-volledigheid. Het standaard informatica boek over het onderwerp is Garey & Johnson's Computers and Intractibility, hetgeen honderden NP-volledige problemen behandelt samen met technieken om NP-volledigheid van problemen te bewijzen. Wees gewaarschuwd, dit is zware kost en alleen aangeraden voor de informaticus in spé.

Alleen te kopiëren voor educatieve doeleinden.
© 2012 Computer Science Unplugged (csunplugged.org)