Minimaal Opspannende Bomen

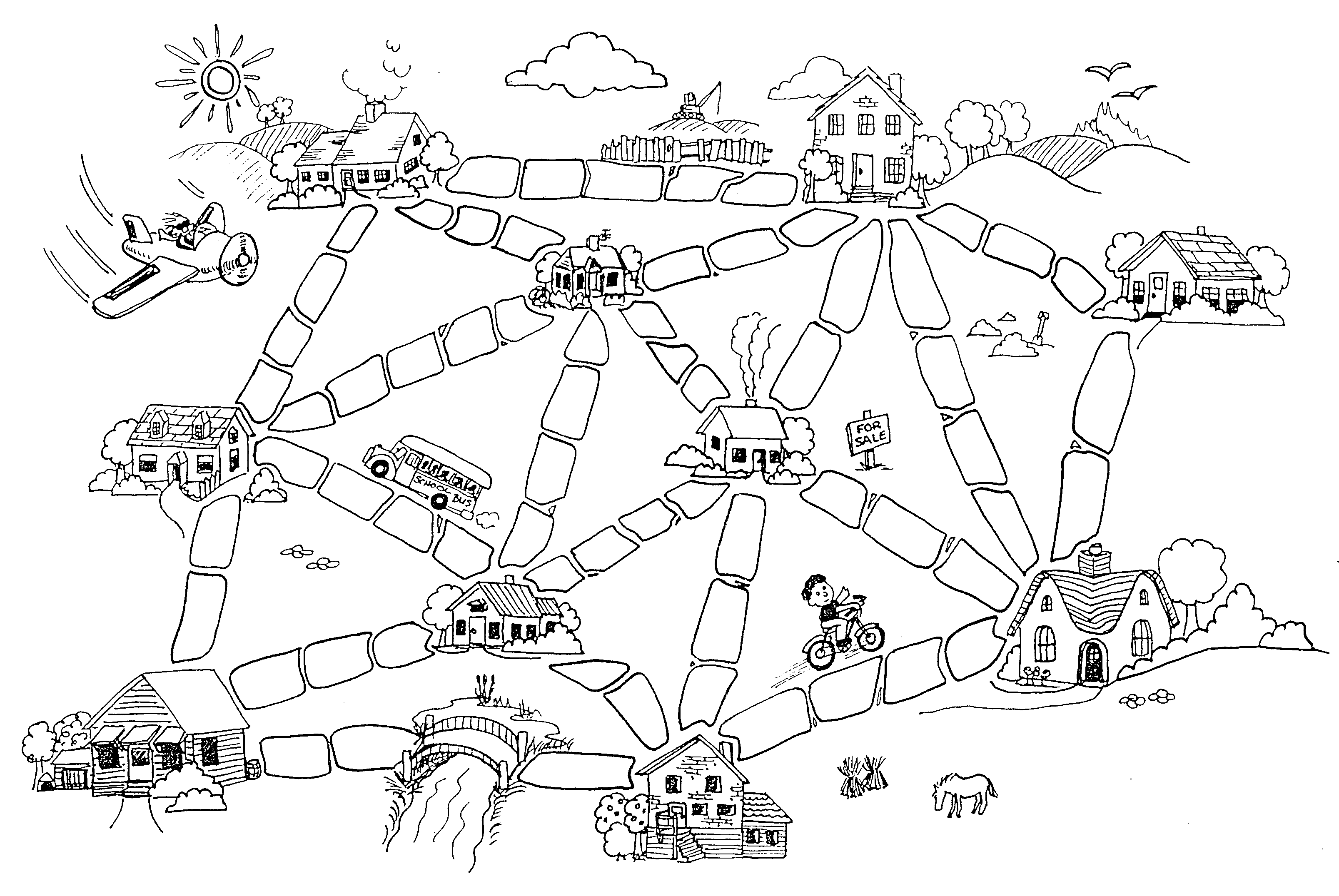
|  |
| --- |
| **Concepten:** complexiteit, minimaal opspannende bomen, algoritmen  **Doel:** Een aantal praktische problemen kan men oplossen door te zoeken naar een netwerk dat alle knooppunten verbindt waarbij de totale lengte van de verbindingen zo kort mogelijk is. Zo’n netwerk heet een minimaal opspannende boom. Door een geschikte representatie te kiezen (een graaf), kun je met verschillende algoritmes zo’n minimaal opspannende boom op een handige maken. In dit geval passen we de algoritmen van Kruskal en Prim toe.  **Leerdoelen:**  Na afloop kun je:   * beschrijven wat een minimaal opspannende boom is; * een oplossingsrichting voor een gegeven probleem ontwikkelen en weergeven als een algoritme; * een algoritme op een gestructureerde wijze weergeven met een stroomdiagram; * uitleggen wat een brute force algoritme is; * herkennen of een probleem op te lossen is met een minimaal opspannende boom algoritme; * minstens twee verschillende standaardoplossingen gebruiken voor het oplossen van hetzelfde probleem; * een probleem analyseren en de relevante informatie (abstractie) in een geschikte representatie (graaf, boom) weergeven; * een aantal standaard graafalgoritmen (zoals brute force, Kruskal en Prim) herkennen en gebruiken. * redeneren over de efficiëntie van een algoritme in relatie tot de invoer; * je kunt algoritmen (waaronder standaardalgoritmen), vergelijken met betrekking tot efficiëntie. * redeneren over de correctheid van een algoritme.   **Algoritmes en technieken:** Brute force algoritmes, gretige algoritmes, Kruskal, Prim. |

# Uitdaging: Modderdorp

Hieronder zie je een kaart van een dorp afgebeeld. Het dorp heeft geen geasfalteerde wegen. Als het regent krijgen alle dorpsbewoners modderige schoenen. Het dorp heeft niet genoeg geld om alle straten te asfalteren, want de burgemeester wil ook een zwembad bouwen.

Aan jou de uitdaging om te bepalen welke straten geasfalteerd moeten worden. Belangrijk is dat je net genoeg asfalteert zodat iedereen overal kan komen (het maakt niet uit dat het een omweg is), maar wel zo goedkoop mogelijk. De asfalteringskosten zijn 1000Euro per steen (het bruggetje kost ook 1000euro).

Kan jij de dorpsbewoners van Modderdorp helpen?

[[1]](#footnote-1)

*Geef aan hoeveel jouw oplossing gaat kosten.*

*Denk je dat jouw oplossing de optimale oplossing is, dus dat er geen oplossingen zijn die goedkoper zijn? Waarom denk je dat?*

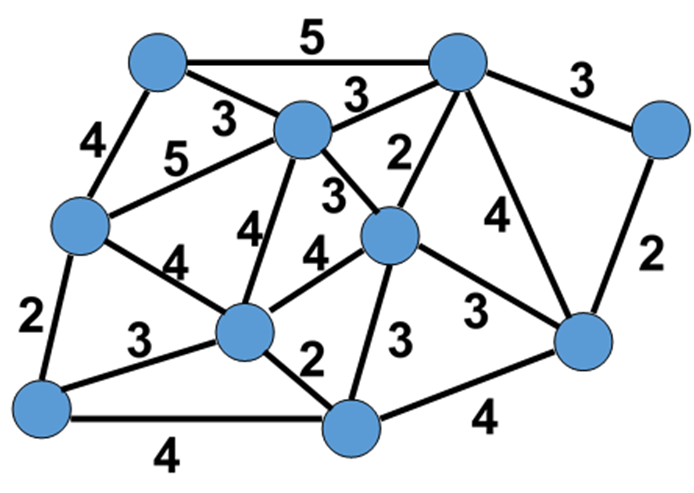
*Omschrijf je strategie. Hoe ben je te werk gegaan? Waar ben je begonnen? Hoe heb je besloten welke pad als eerst te asfalteren? En als tweede? Kun je dit in het algemeen omschrijven?*

*Teken van je strategie een stroomdiagram.*

# Verschillende strategieën vergelijken

**Representatie**

Om inzicht te krijgen in het probleem, kiezen we vaak voor een overzichtelijkere manier om dit te representeren. Een **graaf** kan vaak een handige representatie zijn. Hier zie je het modderdorp probleem als graaf weergegeven:



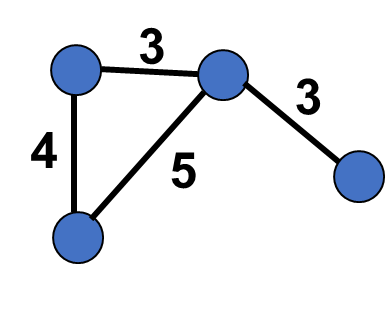
De huisjes worden de knopen (punten) en verbindingslijnen (kanten). Op de verbindingslijnen geven we de asfalteringsprijs aan. Onnodige details laat je weg (zoals hoe het huis eruit ziet). Dat heet **abstractie**.

Er zijn meerdere manieren om zo’n probleem op te lossen. We bekijken nu de volgende mogelijkheden:

1. Brute force
2. Dure straten weghalen
3. Kruskal: de goedkoopste straat aanleggen
4. Prim: de goedkoopste verbindende straat aanleggen

## Strategie A. Brute force

Met een brute force methode bereken je alle mogelijkheden en kiest dan de goedkoopste uit. We laten stap voor stap zien hoe je dat handig kan aanpakken. Hiervoor gebruiken we de volgende kleinere graaf:

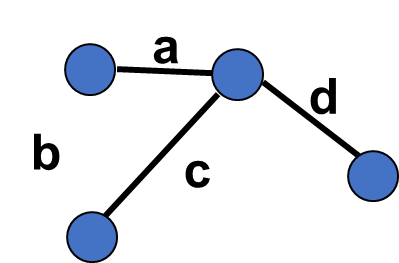


**Stap 1**: Geef elk lijnstuk een eigen letter, bijvoorbeeld met a, b, c en d.

**Stap 2**: Maak een tabel om alle mogelijkheden weer te geven.

|  |  |  |  |
| --- | --- | --- | --- |
| **a** | **b** | **c** | **D** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

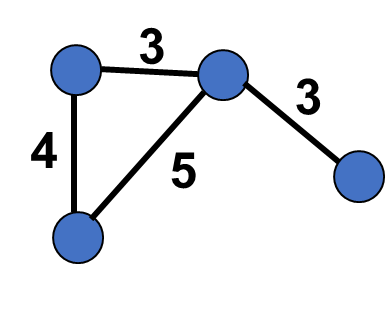
Met een 1 geven we aan dat de verbinding er is. Staat er een 0 dan is er geen verbinding. Dus, bij 1011 hoort het volgende graaf:



**Stap 3**: Streep in de tabel alle rijen door die niet voldoen, dus waarbij niet alle huizen verbonden zijn.

**Stap 4**: Bepaal van de rijen die voldoen welke het goedkoopste is.

*Voer de brute force methode uit bij deze graaf. Gebruik hiervoor de tabel hierboven.*



*Geef aan hoeveel de beste oplossing gaat kosten.*

|  |
| --- |
| Antwoord: 10 |

*Levert de brute force methode altijd de goedkoopste oplossing?*

Antwoord: Brute force is heel erg veel werk, zeker als het netwerk groot is. Maar, ja, als de berekeningen eindelijk klaar zijn, komt daar wel het beste resultaat (dus de goedkoopste oplossing) uit.

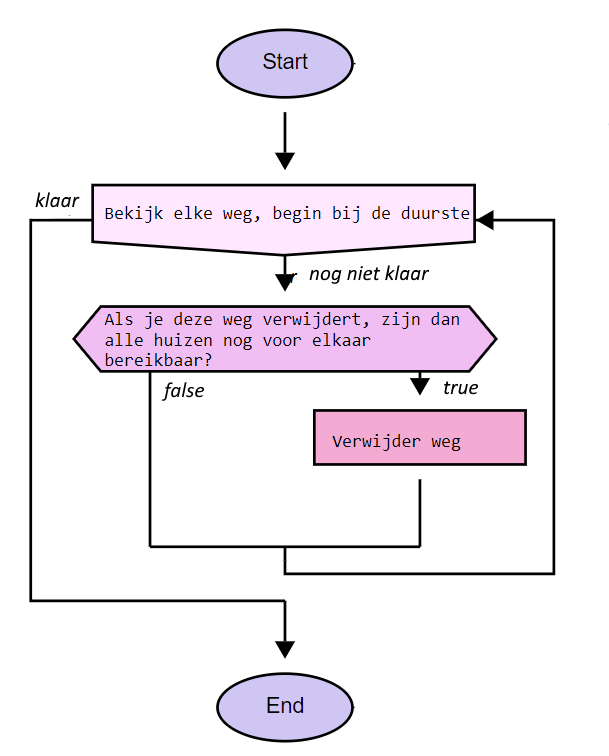
*Hoe veel groter wordt de tabel met mogelijkheden als je een huisje aan je graaf toevoegt?*

Antwoord: Een tabel met 4 knopen heeft 2^4=16 rijen. Een tabel met 5 knopen heeft 2^5 rijen. Het uitbreiden van de graaf met een knoop resulteert in een verdubbeling van het aantal rijen in de tabel.

|  |
| --- |
| **Brute force**  Met de brute force methode bereken je alle mogelijkheden en kiest dan de goedkoopste uit. Dat is vaak heel veel werk, zeker voor een grote graaf, maar het levert wel altijd de beste oplossing op. |

## Strategie B. Dure straten weghalen

Een andere strategie is om met een complete graaf te beginnen waarin alle verbindingen zijn aangegeven. Dan ga je, zolang geen huisje daarbij geïsoleerd raakt, één voor één de duurste straten wegstrepen. Het algoritme staat in de volgende stroomdiagram omschreven:

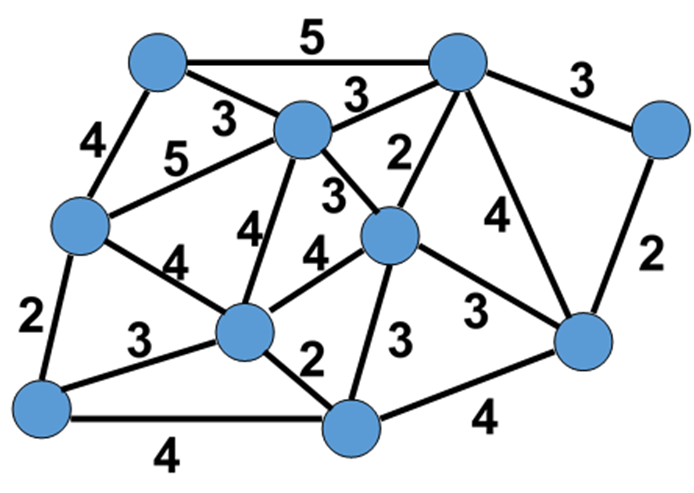
[[2]](#footnote-2)

**Stap 1**: Maak een lijst van alle waardes, van groot naar klein.

De totale lengte is:

Antwoord: 5, 5, 4, 4, 4, 4, 4, 4, 4, 3…., 3, 2, 2, 2, 2

**Stap 2**: Streep één voor één de duurste straat weg. Let op: je mag alleen een straat wegstrepen als daardoor geen huisje geïsoleerd raakt (anders kan die bewoner niet meer zijn huis uit).



Antwoord: eerst de 5-en, dan de 4-en enz.

**Stap 3**: Bereken de totale kosten*.*

Antwoord: 23

*Krijg je met dit algoritme altijd de goedkoopste oplossing?*

Antwoord: Ja, dit algoritme levert altijd de beste oplossing op.

*Is het mogelijk om met dit algoritme op twee verschillende netwerken uit te komen? Leg uit.*

|  |
| --- |
| Antwoord: Ja dat kan. Je kunt bij het wegstrepen van de eerste drie kiezen uit verschillende mogelijkheden. Dit leidt tot een andere boom. |

**Minimaal opspannende bomen**

We begonnen met een graaf: een netwerk met verbonden punten. We zijn toen op zoek gegaan naar een netwerk met alleen de hoogstnodige verbindingen. Wat overblijft van de graaf is een **opspannende boom**: een netwerk dat alle punten met elkaar verbindt en geen circuits (een circuit is een gesloten pad waarvan de begin- en eindpunt samenvallen) bevat.

Om zo min mogelijk asfalt te gebruiken zoeken we een **minimale** opspannende boom, dit wil zeggen een opspannende boom met de kleinst mogelijke totale lengte.

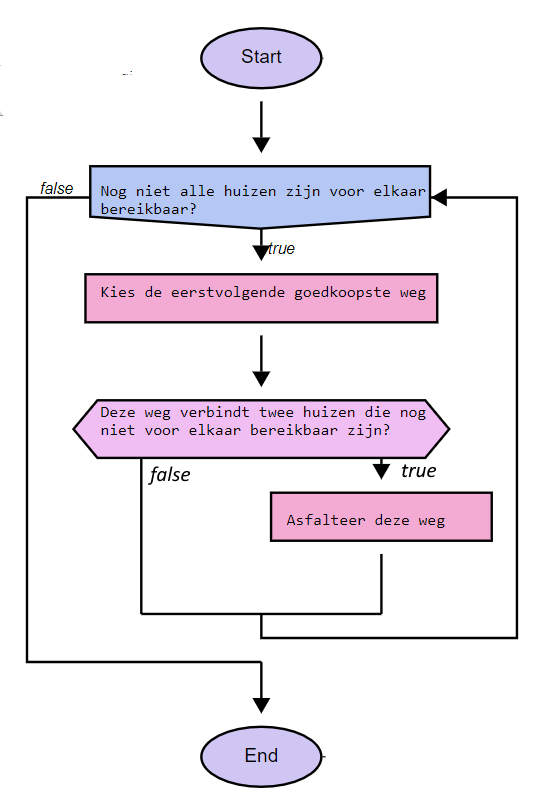
Hier zie je het modderdorp probleem als minimale opspannende boom weergegeven:



## Strategie C. Kruskal: de goedkoopste straat aanleggen

Een efficiënte strategie is om één voor één de goedkoopste straten aan te leggen. Je legt alleen een straat aan als deze een huis verbindt dat nog niet verbonden is, hiermee vermijd je een gesloten circuit. Dit algoritme heet het Kruskal algoritme.

Het algoritme van Kruskal staat in de volgende stroomdiagram omschreven:

[[3]](#footnote-3)

**Stap 1**: Maak een lijst van alle waardes, van goedkoopst naar duurst.

Antwoord: 2,2,2,2, 3, …, 4, 4, 4, 4, 4, 4, 4, 5, 5

**Stap 2**: Teken alle huizen als knopen.



**Stap 3**: Neem de goedkoopste verbinding. Alleen als deze geen gesloten circuit maakt, leg je deze verbinding aan. Streep de waarde in je lijst door (ook als je de verbinding niet hebt gelegd).

**Stap 4**: Herhaal de vorige stap totdat alle waarden uit je lijst zijn doorgestreept.

**Stap 5:** Bereken de totale kosten.

Antwoord: 23

*Krijg je met dit algoritme altijd de goedkoopste oplossing?*

Antwoord: Ja, dit algoritme levert altijd de beste oplossing op.

*Is het mogelijk om met dit algoritme op twee verschillende netwerken uit te komen? Leg uit.*

Antwoord: Ja dat kan. Je kunt bij het aanleggen van de eerste twee kiezen uit verschillende mogelijkheden. Dit leidt tot een andere boom.

## Strategie D. Prim: de goedkoopste verbindende straat aanleggen

Een andere efficiënte strategie is het Prim algoritme. Je kiest eerst een beginpunt. Dan kies je vanuit dat beginpunt de allergoedkoopste straat en je legt die aan. Hierna kies je steeds de allergoedkoopste straat waarmee je een niet verbonden huis toevoegt aan het netwerk dat je al hebt: je breidt dus elke stap de groep van huizen die al verbonden zijn met één huis uit. Omdat je legt alleen straten aanlegt naar een huis dat nog niet verbonden is, vermijd je het ontstaan van een gesloten circuit.

**Stap 1**: Teken alle huizen als knopen.



**Stap 2**: Kies een beginpunt en leg de eerste goedkoopste straat aan.

**Stap 3**: Neem de goedkoopste verbinding dat een nieuw huis met je netwerk verbindt. Alleen als deze geen gesloten circuit maakt, leg je deze verbinding aan.

**Stap 4**: Bereken de totale kosten.

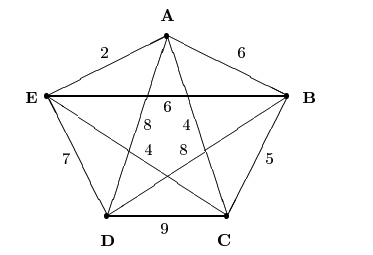
Antwoord: 23

*Teken een stroomdiagram van het Prim algoritme.*

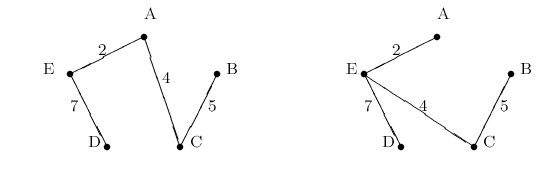
|  |
| --- |
| **Gretige algoritmen**  Een algoritme waarbij je steeds de goedkoopst mogelijke verbinding kiest, zonder vooruit te kijken, heet een **gretig** algoritme. Je maakt dus steeds een keuze die op dat moment het beste lijkt (de grootste directe winst oplevert). De vraag is of dat uiteindelijk leidt tot de allerbeste oplossing. |

# Opgaven

**Opgave 1.** Pas de algoritmen uit deel B, C, en D toe op deze graaf.



ANTWOORD:



**Opgave 2.**

Welk van de algoritmen beschreven in deel A, B, C en D zijn gretige algoritmen?

|  |
| --- |
| Antwoord:   1. Brute force is **niet gretig**, alle mogelijkheden worden bekeken. 2. Duurste straten weghalen is **wel gretig**, door de duurste straten weg te strepen houd je de goedkoopste straten over. 3. Kruskal is **wel gretig**. Omdat je steeds de goedkoopst mogelijke straat kiest heet ook dit een **gretig** algoritme. 4. Prim is **wel gretig**. Omdat je steeds de goedkoopst mogelijke straat kiest heet ook dit een **gretig** algoritme. |

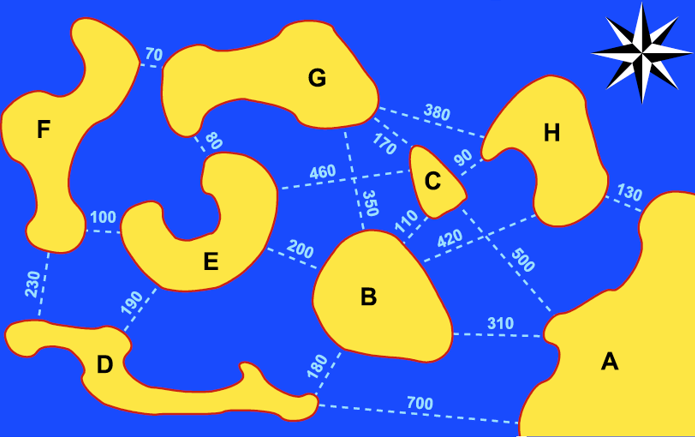
**Opgave 3.**

Een inbraakalarm verbindt sensoren met elkaar en gaat af zodra twee sensoren niet meer met elkaar verbonden zijn. Voor de verbindingen wordt kostbaar koperdraad gebruikt. De verbindingen hebben verschillende lengte en dus ook verschillende waarden. We geven zo’n alarm weer in de vorm van een graaf waarbij de kanten de kostbare koperdraden representeren. Wat kan de inbreker doen om zo veel mogelijk koperdraad te stelen? Hij heeft een kaart van het netwerk.

|  |
| --- |
| Antwoord: De inbreker maakt bij de graaf een kortste opspannende boom. Alle kanten die niet in de boom zitten kan hij weghalen. |

**Opgave 4.**

Hieronder zie je een kaart van de eilandenstelsel van Algos. Om te reizen tussen de eilanden of met het vaste land wordt gebruik gemaakt van veerponden. Als het stormt kan het voorkomen dat de veerponden kapseizen. Daarom wil Algos bruggen gaan bouwen. Het aanleggen van bruggen is wel kostbaar.

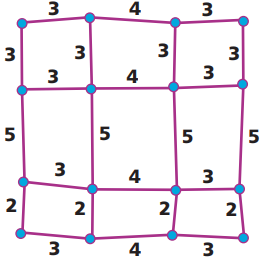
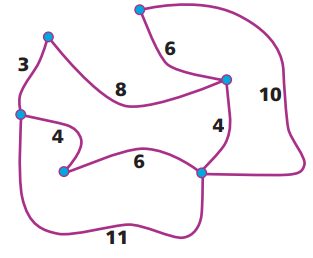
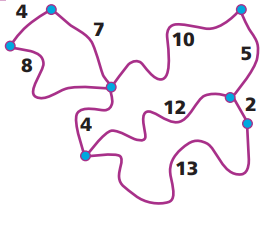
[[4]](#footnote-4)[[5]](#footnote-5)

*Hoe kun je alle eilanden bereikbaar maken, zodat de totale lengte van de bruggen zo kort mogelijk is? Op de kaart staan de afstanden (in meters) aangegeven.*

|  |
| --- |
| Antwoord: Totale lengte: 830m |

**Opgave 5.**

Bepaal de minimale lengte aan kabels die nodig is om alle huizen in elk van de drie kaarten van internet te voorzien.

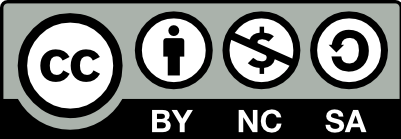
**[[6]](#footnote-6)**

|  |
| --- |
| Antwoord: |

|  |
| --- |
| **Waar gaat dit eigenlijk over?**  Een aantal praktische problemen kan men oplossen door te zoeken naar een netwerk dat alle knooppunten verbindt waarbij de totale lengte van de verbindingen zo kort mogelijk is. Zo’n netwerk heet een **minimaal opspannende boom** (in het Engels: minimal spanning tree). Er bestaan verschillende algoritmes waarmee je zo’n boom kunt maken.  Met brute force bereken je alle mogelijkheden en bepaalt dan de beste.  De algoritmen van Prim en Kruskal vertellen je hoe je zo’n minimale opspannende boom in een gegeven graaf kunt vinden. Met Prim’s algoritme bepaal je één voor één de goedkoopste mogelijke verbinding met het bestaande netwerk. Met Kruskal’s algoritme leg je steeds de kortste verbinding aan die twee nog niet verbonden plaatsen verbindt. Het verschil is dat er bij Prim steeds sprake is van één groter wordend netwerk en bij Kruskal van meerdere netwerken zijn die stapsgewijs met elkaar verbonden worden. Beide algoritmen zijn gretige algoritmen omdat ze steeds de kortste of goedkoopste verbinding leggen zonder vooruit te kijken naar de gevolgen van die beslissing.  Minimaal opspannende bomen kunnen voor veel meer problemen worden gebruikt. Je kunt ze ook inzetten om te bepalen welke bruggen te bouwen om eilanden te verbinden, hoe je goedkoop mogelijk riolering moet aanleggen in een nieuwe woonwijk, bij het plannen van een verkeersnetwerken (waar sporen of straten aan te leggen) en communicatienetwerken (zoals TV, telefoon, internet, etc.), en bij het ontwerpen van computer chips. |

|  |
| --- |
| **Docentenhandleiding**  Uitdaging: Modderdorp  Het doel van deze opdracht is om kennis te maken met het analyseren van een probleem en het weergeven van relevante gegevens in een graaf. Het bedenken van een eigen oplossingsrichting en deze weergeven als een algoritme (stroomdiagram) staat hierbij centraal.  Strategie A. Brute force  De leerling maakt kennis met een brute force algoritme en een strategische manier om alle mogelijkheden bij te houden en af te lopen. De leerling redeneert over de correctheid en efficiëntie van brute-force methode.  Strategie B. Straten weghalen  De leerling maakt kennis met een andere algoritme op hetzelfde doel te bereiken. De leerling leert een algoritme, beschreven in een stroomdiagram, uit te voeren. Daarnaast wordt over correctheid geredeneerd.  Strategie C. Kruskal  De leerling maakt kennis met een standaard graafalgoritme om hetzelfde doel te bereiken. De algoritme, beschreven in een stroomdiagram , wordt uitgevoerd. Daarnaast wordt over correctheid geredeneerd.  Strategie D. Prim  De leerling maakt kennis met een standaard graafalgoritme om hetzelfde doel te bereiken. De algoritme, beschreven in woorden, wordt uitgevoerd. De leerling oefent met het tekenen van een stroomdiagram. Daarnaast wordt over correctheid geredeneerd.  Opgaven.  Bij de opgaven past te leerling de kennis toe in nieuwe contexten. Zo moeten ze het probleem analyseren, in een geschikte representatie weergeven en bepalen of het probleem met een minimaal opspannende boom algoritme is op te lossen. Verschillende standaard algoritmen worden vervolgens toegepast. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Toetsmatrijs**  **Leerdoelen:**   1. uitleggen wat een minimaal opspannende boom is; 2. een oplossingsrichting voor een gegeven probleem ontwikkelen en weergeven als een algoritme; 3. een algoritme op een gestructureerde wijze weergeven met een stroomdiagram; 4. uitleggen wat een brute force algoritme is; 5. herkennen of een probleem op te lossen is met een minimaal opspannende boom algoritme; 6. minstens twee verschillende standaardoplossingen gebruiken voor het oplossen van hetzelfde probleem; 7. een probleem analyseren en de relevante informatie (abstractie) in een geschikte representatie (graaf, boom) weergeven; 8. een aantal standaard graafalgoritmen (zoals brute force, Kruskal en Prim) herkennen en gebruiken. 9. redeneren over de efficiëntie van een algoritme in relatie tot de invoer; 10. redeneren over de correctheid van een algoritme.  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |  | | Modderdorp |  | Cre | Cre |  |  |  |  |  |  | Eval |  | | A. Brute Force |  | App |  |  |  |  |  | App | Eval | Eval |  | | B. Straten weghalen |  | App | App |  |  |  | App | Eval |  | (Eval) |  | | C. Kruskal |  | App | App |  |  |  |  | Eval |  | (Eval) |  | | D. Prim |  | App | Cre |  |  |  |  |  |  |  |  | | Opg 1 |  |  |  |  |  | App |  | App |  |  |  | | Opg 2 |  |  |  | Und |  |  |  | Rem/Und |  |  |  | | Opg 3 | App |  |  |  | Eval |  | Ana |  |  |  |  | | Opg 4 |  |  |  |  | Eval |  | Cre | App |  |  |  | | Opg 5 |  |  |  |  | Eval |  |  | App |  |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |   **Beoordelingsrubrics:**  Voor elk van de uitdagingen en/of opgaven wordt een van de volgende beoordelingen bepaald:  1. *Correct*: juiste antwoord.  2. *Gedeeltelijk bekwaam*: de leerling komt tot een verkeerd antwoord, maar de leerling was wel op de goede weg (De gedachtegang is inzichtelijk en er was sprake van een duidelijke misvatting of er is een rekenfout gemaakt.  3. *Onvoldoende*: antwoorden waarbij de student het probleem niet probeerde of waar het antwoord onjuist was.   |  |  | | --- | --- | | Opdract | Beoordeling (1/2/3) | | Modderdorp |  | | A. Brute Force |  | | B. Straten weghalen |  | | C. Kruskal |  | | D. Prim |  | | Opg 1 |  | | Opg 2 |  | | Opg 3 |  | | Opg 4 |  | | Opg 5 |  | |

1. Bron: csunplugged.org  [↑](#footnote-ref-1)
2. Bron: [www.informaticaunplugged.nl](http://www.informaticaunplugged.nl)  [↑](#footnote-ref-2)
3. Bron: [www.informaticaunplugged.nl](http://www.informaticaunplugged.nl)  [↑](#footnote-ref-3)
4. Bron: Vöcking, B., Alt, H., Dietzfelbinger, M., Reischuk, R., Scheideler, C., Vollmer, H., & Wagner, D. (Eds.). (2010). *Algorithms unplugged*. Springer Science & Business Media. [↑](#footnote-ref-4)
5. [↑](#footnote-ref-5)
6. Bron: http://www.cre8atemaths.org.uk/resources [↑](#footnote-ref-6)