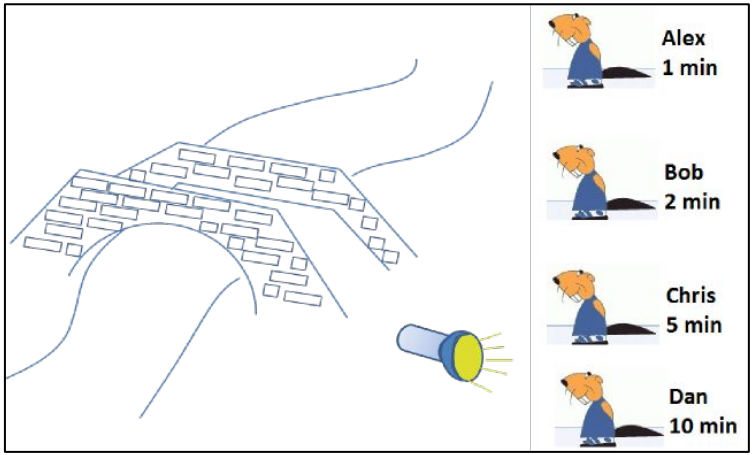
# UGent UniMath 2023
# - algoritmen aan het werk -

Veerle Fack, Universiteit Gent

# De bevers en de brug

# De bevers en de brug




Vier bevers willen in het donker een brug oversteken. Ze kunnen de brug **slechts alleen of met twee tegelij**k oversteken, en omdat het donker is, hebben ze daarbij ook nog de enige **zaklamp** nodig die ze hebben.

De zaklamp over het water teruggooien naar de overkant is veel te riskant, dus moet er telkens één van de bevers de zaklamp terug naar de overkant brengen.

Alex kan de brug oversteken in één minuut, Bob heeft twee minuten nodig, Chris doet er vijf minuten over en Dan tien minuten.

**Wat is de minimum tijd die nodig is om alle bevers aan de overkant te krijgen?**

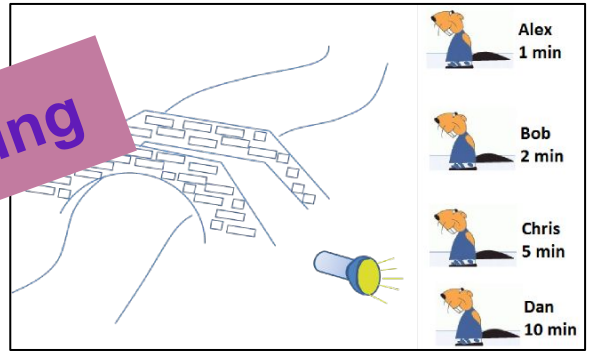
# De bevers en de brug

**Eerste idee:**
➔ snelste telkens meesturen

Dit geeft:
- A + B heen (2 minuten)
- A terug (1 minuut)
- A + C heen (5 minuten)
- A terug (1 minuut)
- A + D heen (10 minuten)

In totaal: 19 minuten

**Gretige benadering**




**Ander idee**
➔ twee traagste samen sturen
➔ dit moet voorbereid worden

Dit geeft:
- A + B heen (2 minuten)
- A terug (1 minuut)
- C + D heen (10 minuten)
- B terug (2 minuten)
- A + B heen (2 minuten)

In totaal: 17 minuten

# Hoger-lager

# Een raadspelletje


Annie en Bea spelen een raadspelletje

- ➔ Annie kiest een geheel getal tussen 1 en 512
- ➔ Bea probeert dit getal te raden, door gokken te doen
- ➔ Annie geeft telkens feedback:
  - ◆ “hoger”: als haar getal groter is dan de gok van Bea
  - ◆ “lager”: als haar getal kleiner is
  - ◆ “juist”: als Bea haar getal geraden heeft

**Geef een strategie waarmee Bea het getal kan raden in zo weinig mogelijk gokken**

**Waarom werkt je strategie?**

**Hoeveel stappen heb je hoogstens nodig?**

# Raden getal ts. 1 en 512: mogelijk scenario

| | | |
|---|---|---|
| 256 | lager | tussen 1 en 512 |
| 128 | hoger | tussen 1 en 255 |
| 192 | lager | tussen 129 en 255 |
| 160 | hoger | tussen 129 en 191 |
| 176 | juist | tussen 161 en 191 |

waarom strategie werkt

hoeveel stappen?

- mogelijk interval telkens halveren
- hoe vaak halveren tot 1 getal over?
- 512 = 2^9 = 2*2*2*2*2*2*2*2*2
- hoogstens ~ 9 = log_2(512) stappen

# Binaire zoekmethode

Als rij niet gesorteerd

- alle getallen bekijken (in het slechtste geval)
- 999 582 34 645 2 599 534 954 712 852 123 967 476 823 912 78 498 875 345 898 900



Als rij gesorteerd

- op basis van middelste getal, beslissen in welke helft getal kan zitten
- 2 34 78 123 345 476 498 534 582 599 **645** 712 823 852 875 898 900 912 954 967 999

## Opzoeken van item in rij data

- vb. zoek ***878*** in gegeven rij getallen

| # getallen | lineair | binair |
|---|---|---|
| 512 | 512 | 9 |
| 1024 | 1024 | 10 |
| 2048 | 2048 | 11 |
| 4096 | 4096 | 12 |
| ... | ... | ... |
| 1.048.576 | 1.048.576 | 20 |
| 2.097.152 | 2.097.152 | 21 |

# Handen schudden

# Handen schudden

In een groep van 9 personen, is het mogelijk dat iedereen precies drie andere personen de hand drukt?

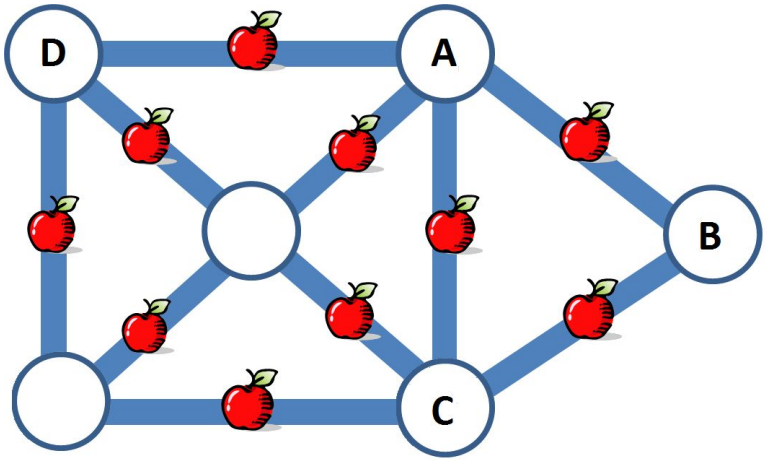
**Eigenschap**
som van graden over alle toppen
= 2 x # bogen

**Grafen**

- toppen verbonden door bogen
- buren van een top
- graad van een top

# Appels rapen / Euleriaanse grafen

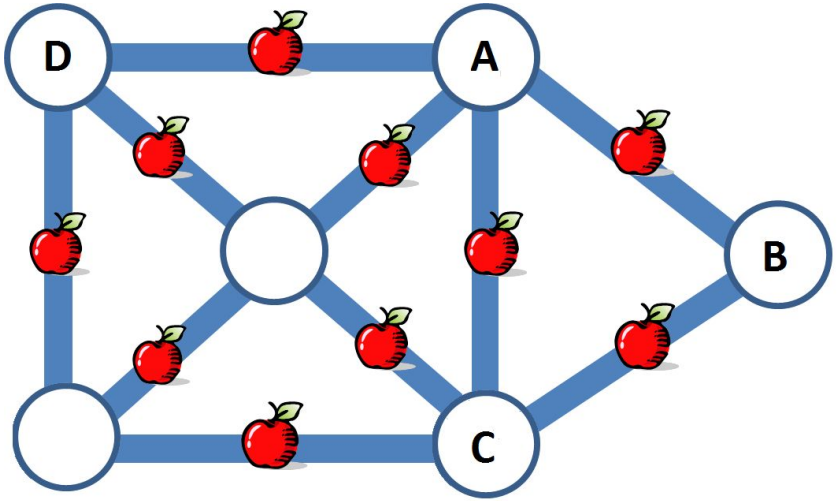
# Appels rapen



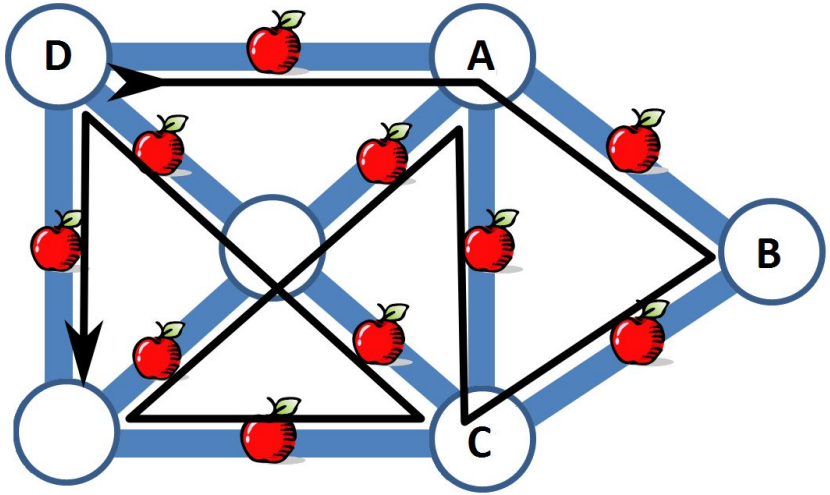

Het appelseizoen is aangebroken. Ward de bever is appels gaan plukken, maar is ze onderweg verloren.
Om zo snel mogelijk zijn appels terug te kunnen oprapen, beslist hij een rondgang te maken die elke weg precies één keer (volledig) doorloopt.

Op welke van de plaatsen A-D moet hij beginnen om alle appels op te rapen zonder een weg meer dan één keer te moeten doorlopen?

# Appels rapen

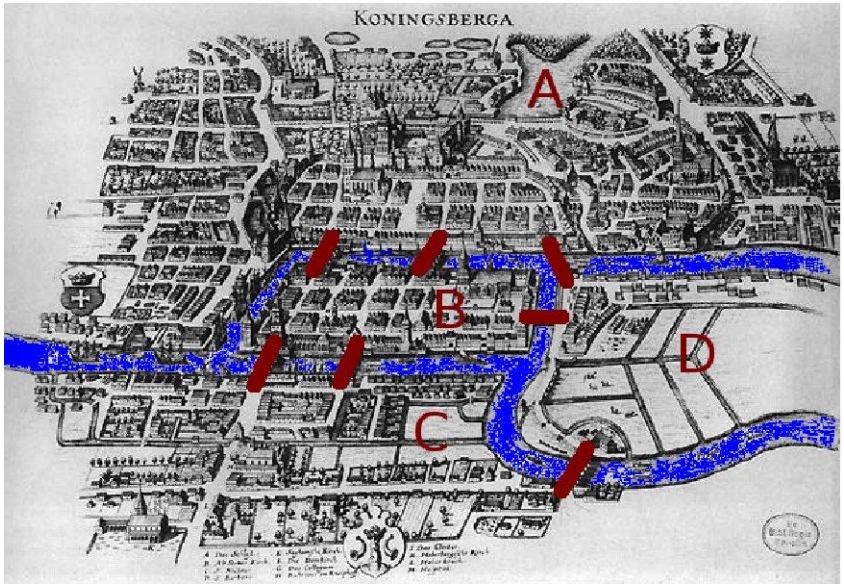



graaf heeft toppen en bogen

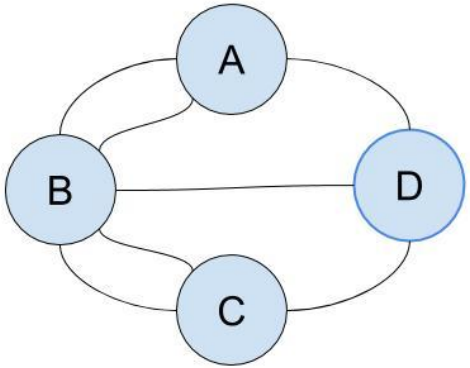



wandeling die elke boog
precies eenmaal gebruikt

# Euleriaanse grafen




**Bruggen van Königsberg (Euler, 1736)**

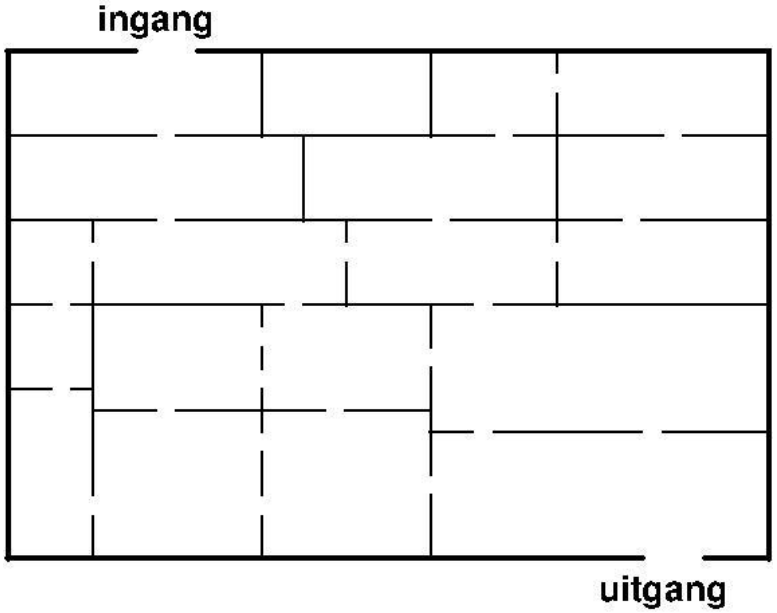



## Karakterisatie

- ➔ gesloten Euleriaanse wandeling a.s.a. alle toppen even graad
- ➔ Euleriaanse wandeling a.s.a. twee toppen met oneven graad, alle andere toppen even graad

Bewijs: door constructie

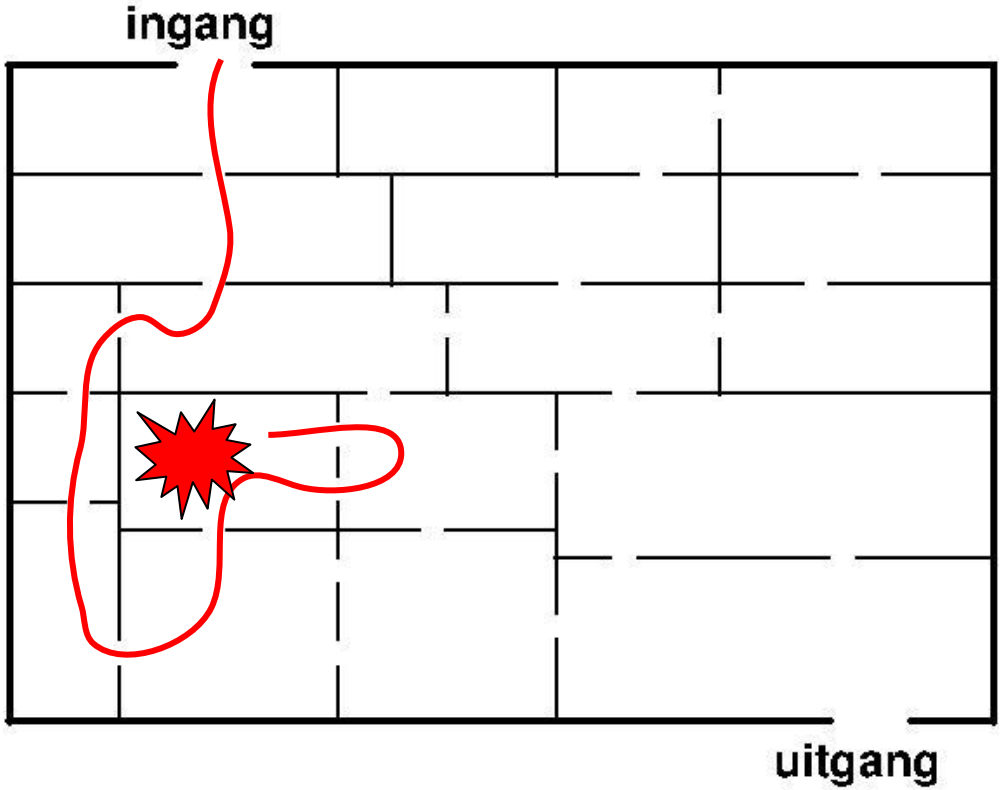
# Ontsnappen uit de spiegelhal




Hiernaast het plattegrond van een spiegelhal. Als bezoeker start je bij de ingangsdeur en passeer je dan door opeenvolgende deuren, totdat je de uitgangsdeur bereikt. Wanneer je een deur gepasseerd bent, sluit deze deur automatisch. Veronderstel dat je de weg uit een kamer steeds kunt vinden zolang nog niet alle deuren gesloten zijn.

**Bestaat het risico dat je voor altijd in de spiegelhal opgesloten geraakt?**

# Ontsnappen uit de spiegelhal




als de graaf een euleriaanse wandeling van ingang naar uitgang heeft, is ontsnappen altijd mogelijk

➔ we controleren of elke kamer een even aantal deuren heeft

# Vergaderingen plannen

| | Anna | Bart | Chris | Dina | Eric | Fons |
|---|---|---|---|---|---|---|
| Anna | | X | | X | X | |
| Bart | X | | X | X | | X |
| Chris | | X | | | X | |
| Dina | X | X | | | X | X |
| Eric | X | | X | X | | X |
| Fons | | X | | X | X | |

Gegeven

➔ gewenste vergaderingen per twee personen (X)

Gevraagd

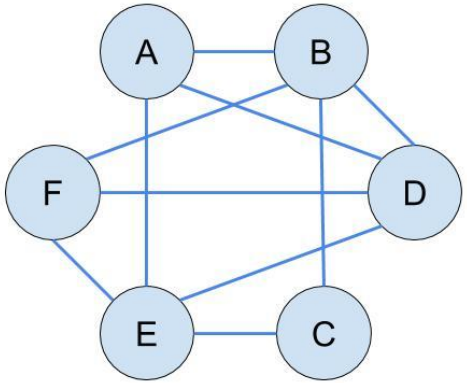
➔ vergaderschema waarbij elke vergadering gepland

➔ bij overgang naar volgende vergadering blijft 1 persoon zitten

➔ niemand meer dan 2 vergaderingen na elkaar

# Vergaderingen plannen

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | | X | | X | X | |
| B | X | | X | X | | X |
| C | | X | | | X | |
| D | X | X | | | X | X |
| E | X | | X | X | | X |
| F | | X | | X | X | |



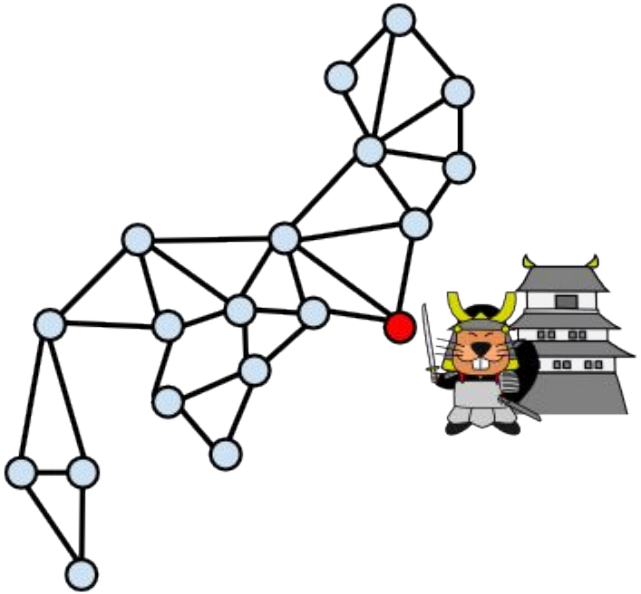

Voorstellen als graaf
- ➔ toppen zijn personen
- ➔ bogen zijn gewenste vergaderingen

Gezochte volgorde van vergaderingen
- ➔ wandeling in graaf die elke boog precies eenmaal bevat
- ➔ d.i. Euleriaanse wandeling

**Graaf heeft twee oneven toppen, dus schema is mogelijk**

# Rooksignalen / BFS

# Rooksignalen



In Japan, een heel lange tijd geleden, dienden de Ninjas de Shoguns. In geval van nood gebruikten ze rooksignalen om met elkaar te communiceren. In de figuur hiernaast geeft het rode punt aan waar de Shogun-regering zich bevindt. Alle andere (blauwe) punten komen overeen met plaatsen waarop een rooksignaal kan worden aangestoken. Twee punten zijn in dit schema met elkaar verbonden als ze elkaars rooksignalen kunnen zien.
Op elk punt staan er dag en nacht Ninjas klaar. Zodra ze een rooksignaal zien (van een punt dat met het hunne is verbonden), laten ze precies één minuut later hun eigen rooksignaal zien.

**Wanneer er bij de de Shogun-regering een rooksignaal wordt aangestoken, na hoeveel minuten zal het signaal dan op alle andere punten ook zijn aangestoken?**

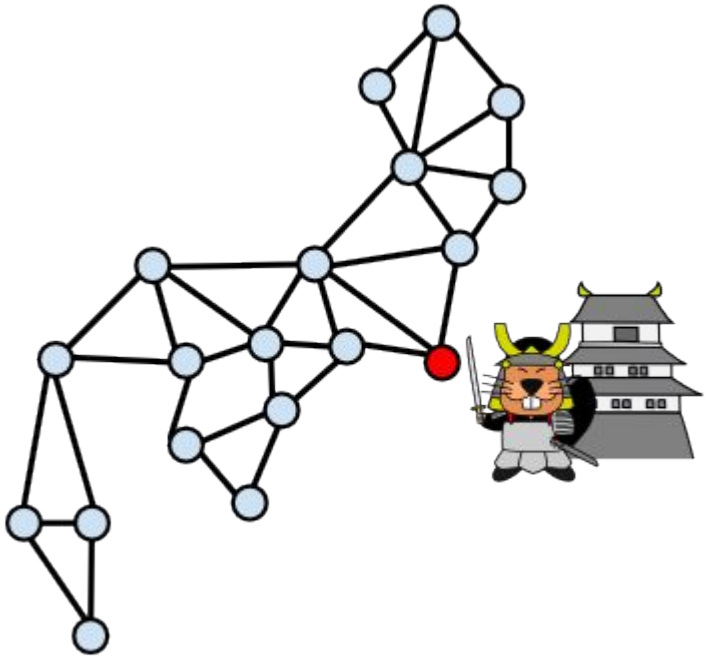
# Rooksignalen

Gegeven:

- ➔ rode punt: regering
- ➔ andere punten: rooksignaalplaatsen
- ➔ verbindingen: zichtbaarheid
- ➔ na opmerken rooksignaal: 1 minuut later eigen rooksignaal geven

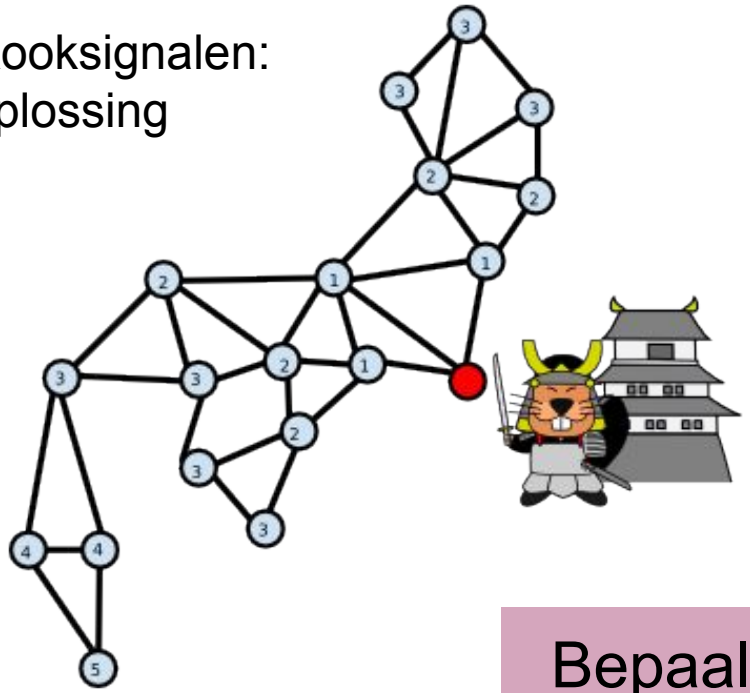
Gevraagd:

- ➔ signaal start bij regering
- ➔ na hoeveel minuten branden alle rooksignalen?



# Breedte-eerst doorlopen van graaf

Rooksignalen:
oplossing




Strategie:
- ➔ begin in starttop
- ➔ markeer alle buren met 1
- ➔ markeer alle buren hiervan met 2
- ➔ enz

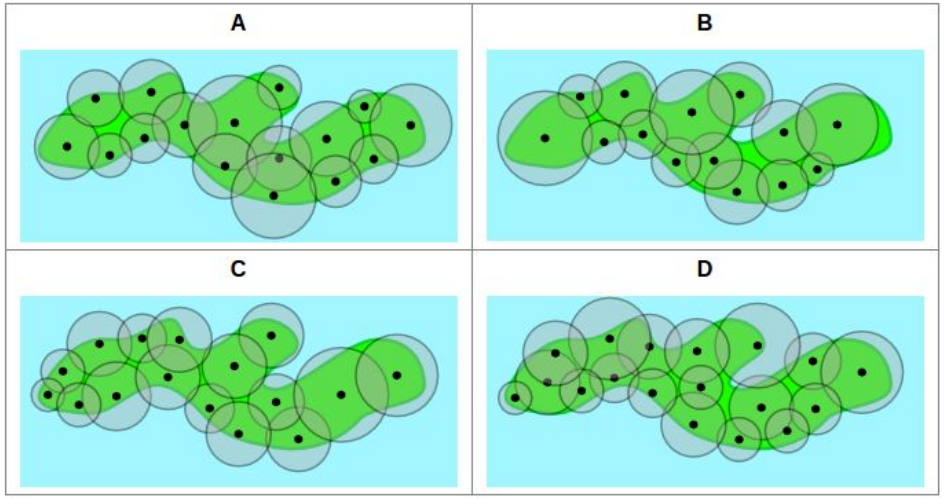
Systematisch alle toppen bezoeken

Bepaalt kortste pad van starttop naar elke andere top

# Stormbestendig netwerk

# Stormbestendig netwerk

De GSM-maatschappij Bever Telecom wil GSM-masten plaatsen op Windeneiland. Het dekkingsgebied van een mast is een cirkel die errond is gecentreerd. Twee masten heten verbonden met elkaar als hun dekkingsgebieden overlappen. Twee masten kunnen met elkaar communiceren als er een rij tussenliggende masten bestaat zodat elke mast met elke volgende is verbonden.

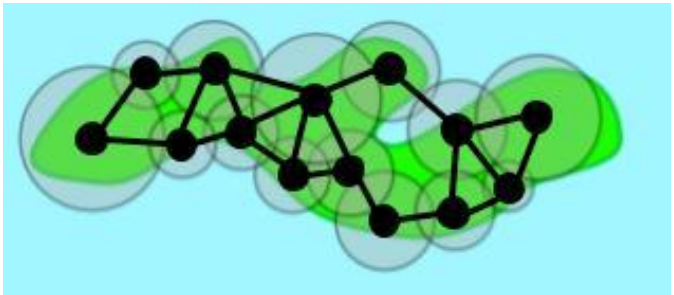

Door de sterke wind op het eiland gebeurt het af en toe dat een mast breekt. Als er ergens één mast niet meer functioneert, willen we toch nog dat elke twee van de overblijvende torens met elkaar kunnen blijven communiceren.

**Welke van de gegeven opstellingen moeten we hiervoor gebruiken?**

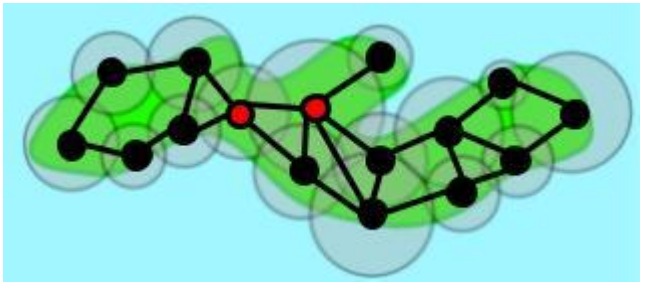
# Connectiviteit van netwerk / Scharniertoppen

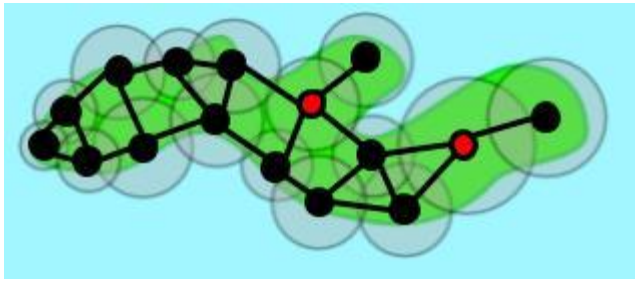
Voorstellen als graaf
➔ toppen = masten
➔ verbonden als bereik overlapt

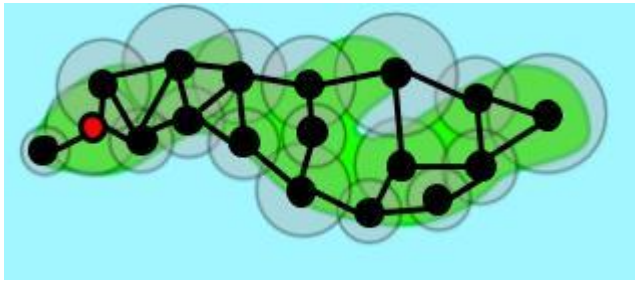
Sterk verbonden (B)



Niet sterk verbonden







# Batman kleedt zich aan / Topologisch sorteren

# Batman kleedt zich aan

hij moet volgende kledingstukken aantrekken

- collant
- pakje
- laarzen
- short
- handschoenen
- gordel
- hoed
- cape

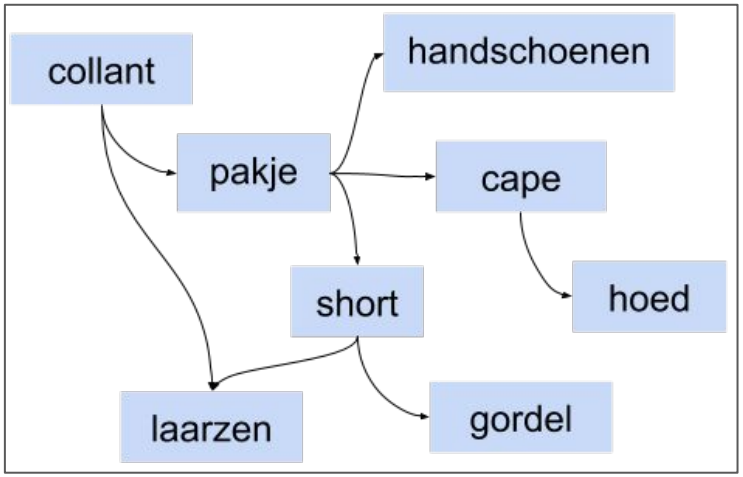
rekening houdend met volgende “voorrangsregels”:

- collant voor pakje
- pakje voor short
- pakje voor handschoenen
- short voor gordel
- pakje voor cape
- cape voor hoed
- short voor laarzen
- collant voor laarzen

geef een volgorde waarin hij zich kan aankleden

# Batman kleedt zich aan

voorstellen als graaf: pijlen geven voorrang aan

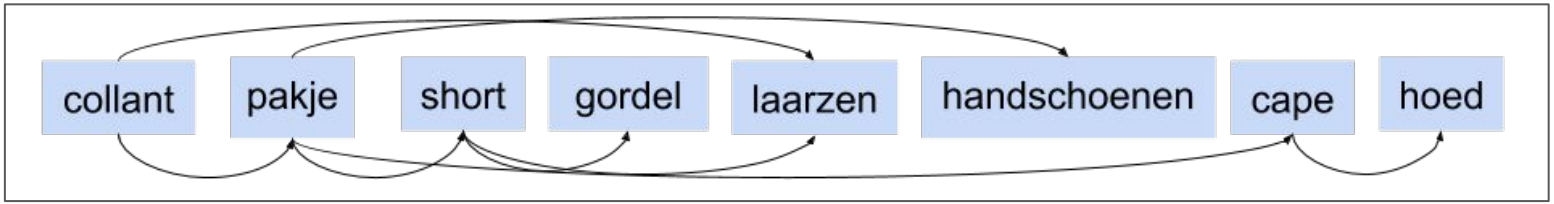



"voorrangsregels":

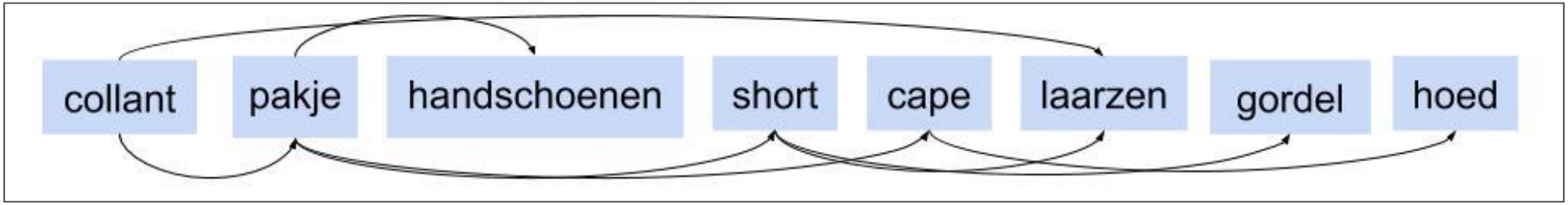
- collant voor pakje
- pakje voor short
- pakje voor handschoenen
- short voor gordel
- pakje voor cape
- cape voor hoed
- short voor laarzen
- collant voor laarzen

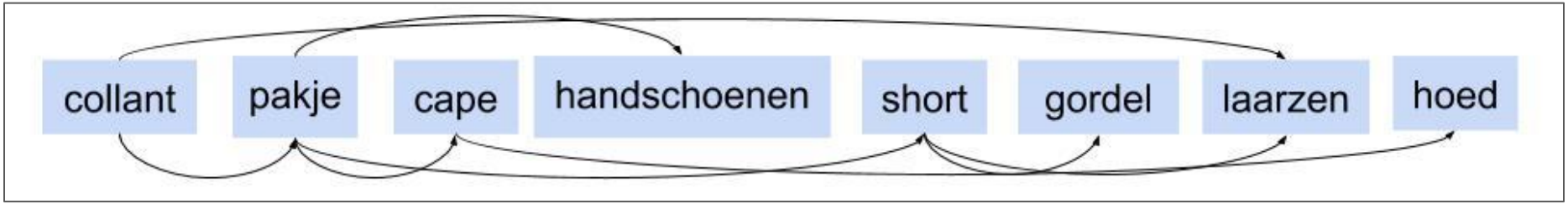
**gerichte acyclische graaf**
(DAG, directed acyclic graph)

**topologische ordening**
➔ volgorde van toppen zodat alle pijlen voorwaarts wijzen

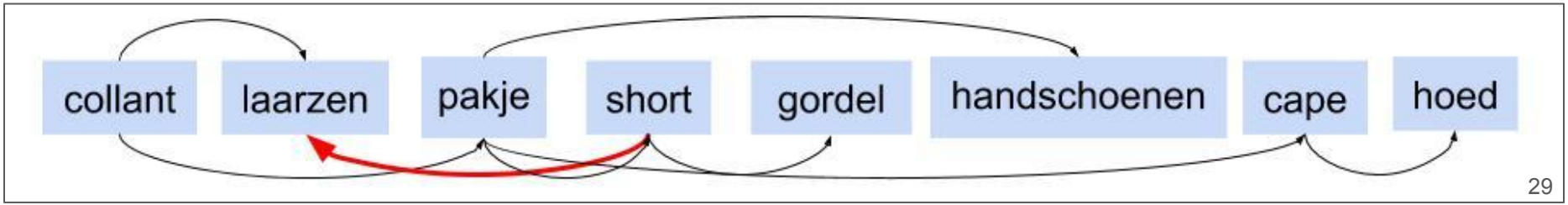


# Andere ordeningen




## Probleem



# Topologisch sorteren

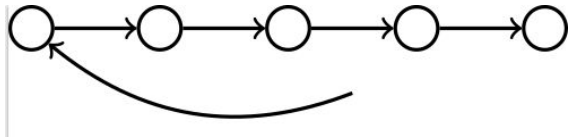
## Algoritme

Input
- een gerichte acyclische graaf (DAG)

Output
- een lijst met toppen in topologische volgorde

Strategie:
- kies een top zonder inkomende pijlen
- voeg deze top toe aan de outputlijst
- “verwijder” deze top uit DAG
- herhaal tot alle toppen behandeld

Waarom is er altijd een top zonder binnenkomende pijlen?

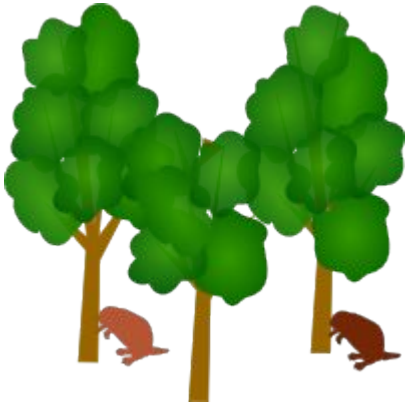


# Planningsproblemen

# Bomen doorknagen

De bevers willen drie bomen doorknagen. Eén bever heeft precies 10 minuten nodig om één boom door te knagen, maar hij mag ook meer tijd nemen en hoeft de boom niet in één keer door te knagen. Twee bevers mogen echter niet samen aan dezelfde boom knagen, want dat is te gevaarlijk: ze zouden elkaar kunnen kwetsen met hun tanden.

**Wat is de minimale tijd die twee bevers nodig hebben om de drie bomen door te knagen?**

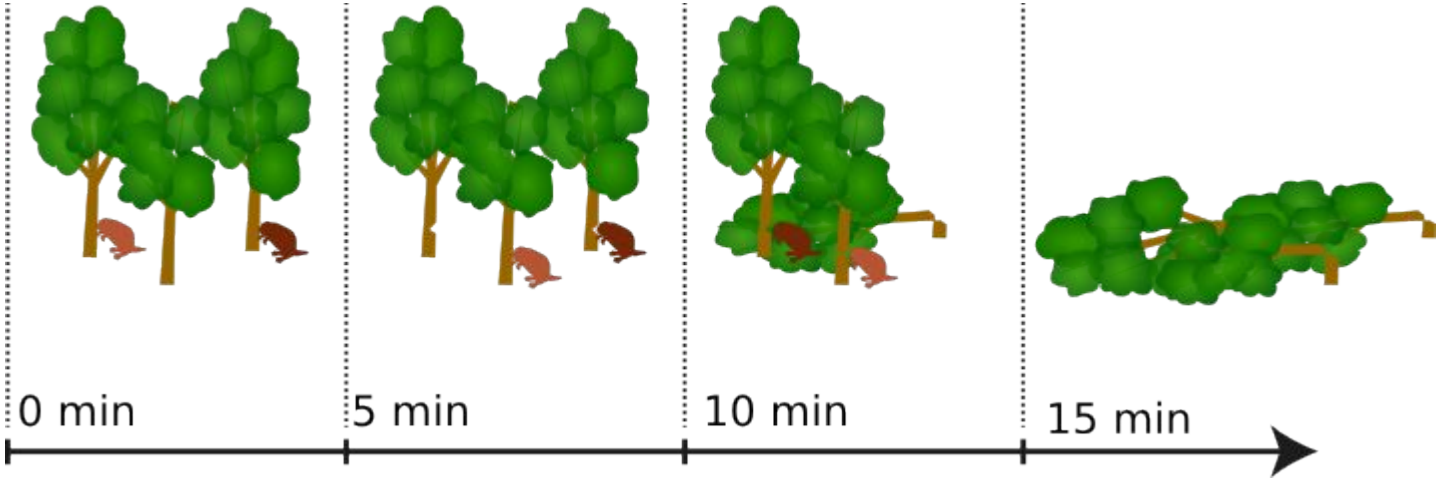


Mogelijkheden

- 30 minuten: kan beter
- 20 minuten: kan
- 15 minuten: kan???
- 10 minuten: kan niet

# Bomen doorknagen

**Oplossing in 15 minuten:**



# Planningsproblemen

**Inplannen van taken (*scheduling*)**

- ➔ plannen van taken voor CPU's
- ➔ plannen van invoer/uitvoer naar printers, harde schijven, e.d.

**Veelgebruikte strategieën**

- ➔ "wie eerst komt, eerst maalt" (*"first come first served"*)
  - ◆ zou hier 20 minuten vragen
- ➔ taken onderverdelen
  - ◆ levert hier optimale oplossing van 15 minuten

# Klanten bedienen

Je moet vier klanten bedienen. Voor Anna heb je 8 minuten nodig, voor Bart 3 minuten, voor Chris 15 minuten, voor Dina 10 minuten.

Om deze klanten allemaal zo tevreden mogelijk te houden, wil je ze bedienen in zodanige volgorde dat het gemiddeld aantal minuten dat een klant op zijn resultaat moet wachten zo klein mogelijk is.

**Welke volgorde laat een klant gemiddeld zo kort mogelijk wachten op de (afgewerkte) bediening?**

1. Anna - Bart - Chris - Dina
2. Chris - Dina - Anna - Bart
3. Dina - Bart - Chris - Anna
4. Bart - Dina - Anna - Chris
5. Bart - Anna - Dina - Chris

# Algemeen planningsprobleem

Gegeven

➔ taken met uitvoeringstijden

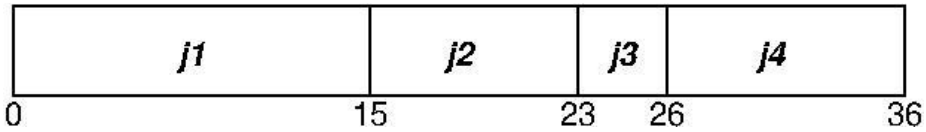
Gevraagd

➔ volgorde van uitvoering zodat gemiddelde eindtijd zo klein mogelijk

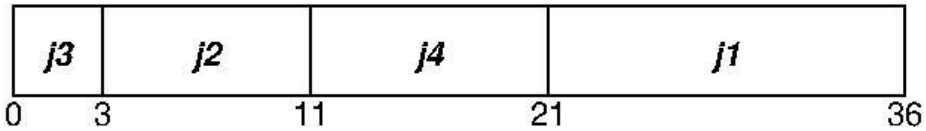
| Taak | J1 | J2 | J3 | J4 |
|---|---|---|---|---|
| Tijd | 15 | 8 | 3 | 10 |

Kan het nog beter?

In gegeven volgorde:
gemiddelde eindtijd =
(15 + 23 + 26 + 36) / 4 = 25




In gesorteerde volgorde:
gemiddelde eindtijd =
(3 + 11 + 21 + 36) / 4 = 17.75



# Waarom best kortste taak eerst?

**Volgorde:** 15, 8, 3, 10

**Eindtijden:**

- 15
- 15 + 8 = 23
- 15 + 8 + 3 = 26
- 15 + 8 + 3 + 10 = 36

**Gemiddelde:**

- (15 + 23 + 26 + 36) / 4

**Volgorde:** 3, 8, 10, 15

**Eindtijden:**

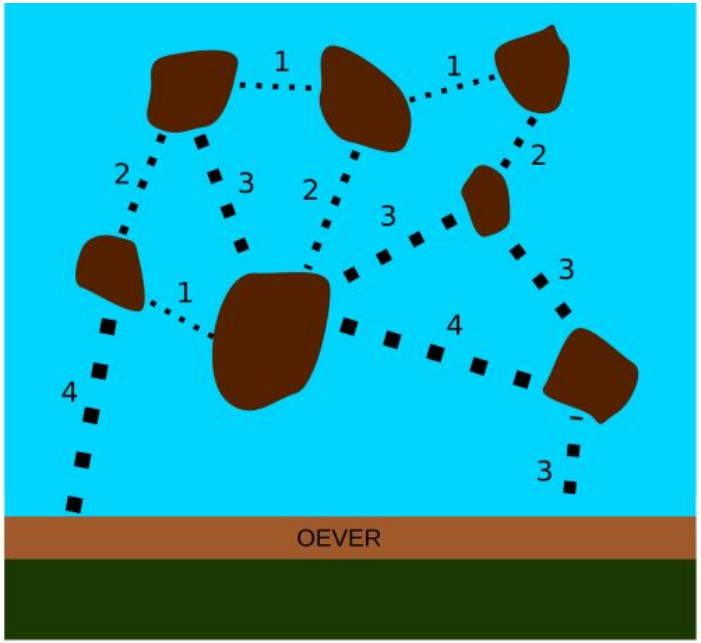
- 3
- 3 + 8 = 11
- 3 + 8 + 10 = 21
- 3 + 8 + 10 + 15 = 36

**Gemiddelde:**

- (3 + 11 + 21 + 36) / 4

# Beverburchten verbinden (MST)

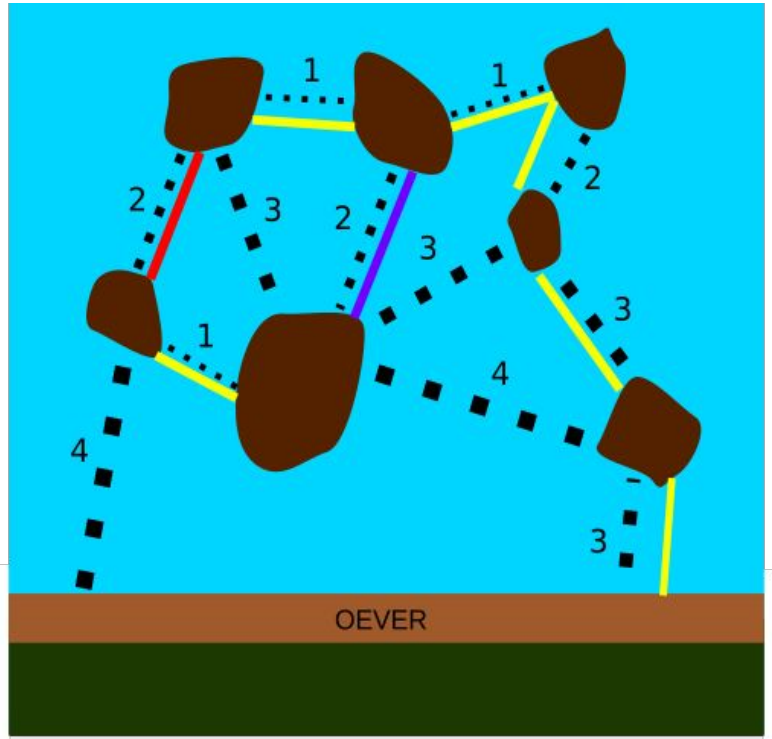
# Beverburchten verbinden




Er liggen zeven beverburchten in een vijver niet ver van de oever. Tussen de burchten kunnen de bevers bruggen bouwen zoals door de stippellijnen is aangegeven. De getallen bij de lijnen geven aan hoeveel bomen je nodig hebt voor elke brug.
De bevers willen genoeg bruggen bouwen zodat elke burcht vanop de oever kan bereikt worden zonder te zwemmen.

**Wat is het kleinste aantal bomen dat ze hiervoor nodig hebben?**

# Beverburchten verbinden




**Oplossing: 13 bomen**

- ➔ alle gele bruggen
- ➔ ofwel rode ofwel paarse brug
- ➔ 1+1+1+2+2+3+3=13

**Kan dit met minder bomen?**

- ➔ 7 burchten, dus minstens 7 bruggen nodig
- ➔ kan 1+1+1+2+2+2+3=12 ?
- ➔ paarse en rode brug maken een lus, dus niet allebei te nemen

**Minimale-kost opspannende boom**

# Minimale-kost opspannende boom

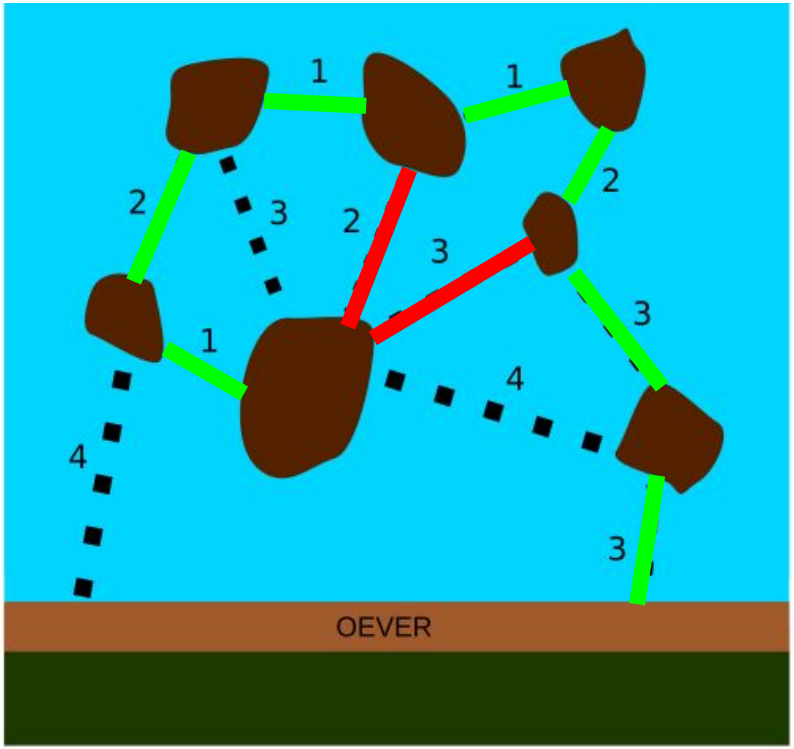
**Algoritme van Kruskal (1956)**

- overloop bogen volgens stijgend gewicht
- voeg telkens volgende kleinste boog toe, tenzij die lus maakt

**Ontwerpstrategie: gretig algoritme**

➔ bewijs van correctheid

➔ implementatie van algoritme in programmeertaal



# Minimale-kost opspannende boom

**Algoritme van Prim (1957)**

- laat boom groeien
- voeg telkens kleinste boog toe die boom met een top uitbreidt