# Derde college algoritmiek

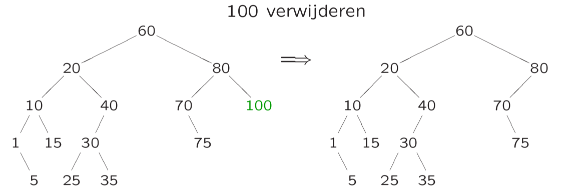
23 februari 2012

## Toestand-actie-ruimte





100 verwijderen

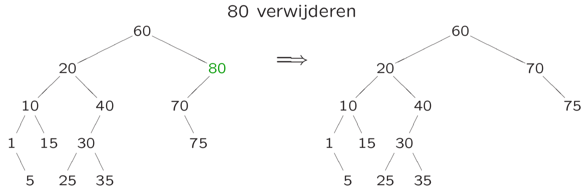



verwijderen van een blad





80 verwijderen

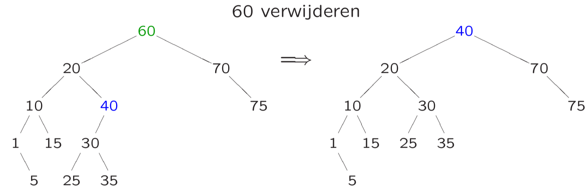



verwijderen van een knoop met 1 kind





60 verwijderen




verwijderen van een knoop met 2 kinderen
verwissel met de grootste kleinere of kleinste grotere





**Probleem ⟶ Toestand-actie-ruimte**

Een toestand-actie-ruimte (toestand-actie-diagram, state transition diagram, toestandsruimte, state space)

- *Bestaat uit* alle mogelijke toestanden en acties
- Begintoestand, eindtoestand(en)
- Een actie veroorzaakt een overgang van de ene (toegelaten) toestand naar een andere
- *Oplossing* van het probleem: een opeenvolging van acties die van de begintoestand naar een eindtoestand leiden





**Voorbeeld 1: NIM**

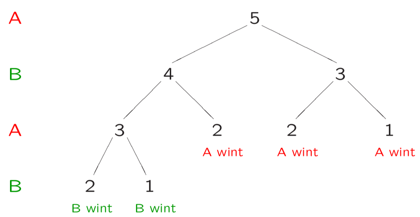
- We beginnen met één stapel van $n$ lucifers (begintoestand)
- Er zijn twee spelers: A en B
- De spelers pakken om de beurt 1 of 2 lucifers (acties)
- Het spel is afgelopen als er geen lucifers meer op de stapel liggen (eindtoestand)
- De speler die de laatste lucifer(s) pakt heeft gewonnen

**Gevraagd:** is het spel winnend voor degene die begint?





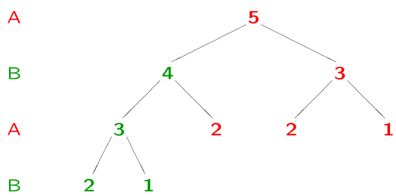
**toestand-actie-ruimte (spelboom)** $n = 5$



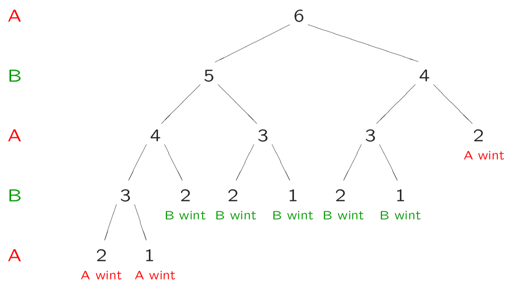





**Winnend voor A.**
Winnende zet: 2 lucifers wegnemen



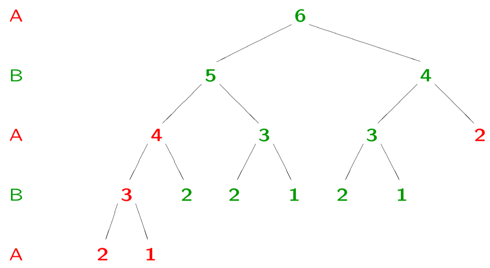
**toestand-actie-ruimte (spelboom)** $n = 6$








**A kan niet winnen** (bij perfect spel van B)








Een stand is winnend voor degene die aan de beurt is als een der directe (= in 1 zet te bereiken) vervolgstanden NIET winnend is voor de tegenstander.

Een algoritme dat bepaalt of een stand winnend is, ziet er dus ruwweg zo uit:

Loop alle mogelijke directe vervolgstanden af:
 kijk of je er een tegenkomt die not winnend is voor de tegenstander: recursie
 zo ja, dan is de oorspronkelijke stand winnend (en heb je meteen een winnende zet) en hoef je niet verder te kijken
 zo nee, dan de volgende vervolgstand proberen
Als alle vervolgstanden zijn geweest is de oorspronkelijke stand niet winnend.





```
winnend(stand)::

   if eindstand(stand) then
      //  makkelijk; bijv return false;
   else
      for alle mogelijke zetten i do
          kopie := stand;
          doezet(kopie,i);
          if not winnend(kopie) then
              return true;
      od
      return false;
   fi
```

Zie ook Programmeermethoden (college over recursie)





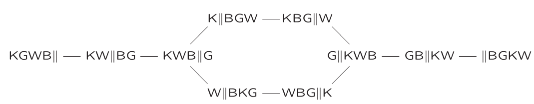
**Voorbeeld 2: Old world puzzle** (Ex. 1.2.1.) **(ook in tweede editie)**

We hebben een kool, een geit, een wolf en een boer. Deze moeten met een bootje van de ene kant van de rivier naar de andere. In het bootje kan alleen de boer met één ander iets. Als de boer er niet bij is zal de wolf de geit opeten en de geit de kool. De boer is de enige die de boot kan "besturen".

**Vraag:** Hoe kan alles naar de andere oever verplaatst worden?





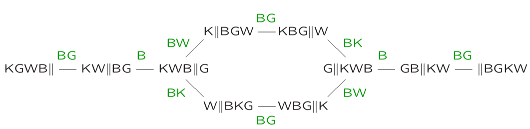



Merk op: de boot ligt altijd aan de oever waar de boer zich bevindt.

De oplossing is een kortste pad van de begintoestand naar de eindtoestand: hier zijn er twee, bij beide moet de boer 7 keer de rivier oversteken.








Merk op: de boot ligt altijd aan de oever waar de boer zich bevindt.

De oplossing is een kortste pad van de begintoestand naar de eindtoestand: hier zijn er twee, bij beide moet de boer 7 keer de rivier oversteken.





Een leuke variant op dit probleem is het volgende:

We hebben drie professoren en drie studenten. Deze moeten allemaal met een bootje van de ene kant van de rivier naar de andere. In het bootje kunnen hooguit twee personen. Op beide oevers mogen de professoren niet in de meerderheid zijn, anders worden de studenten nerveus.

**Vraag:** Hoe kan iedereen naar de andere oever verplaatst worden?

Merk op dat in tegenstelling tot het boer-wolf-geit-kool-probleem hier iedereen de boot kan "besturen". Er moet nu dus in een toestand worden aangegeven waar de boot ligt.

Er zijn 4 verschillende oplossingen, elk met 11 keer overvaren.

**Voorbeeld 3: Torens van Hanoi**

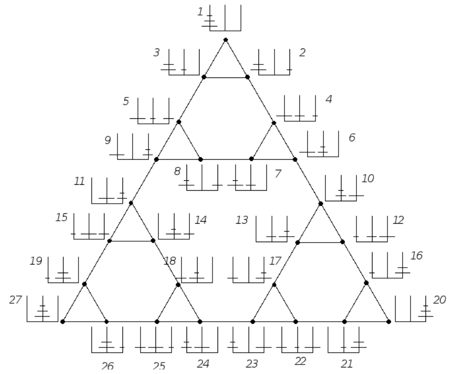
Gegeven $n$ ($n \geq 1$) schijven, alle verschillend in grootte, en 3 palen. In de beginsituatie liggen alle schijven boven op elkaar om één paal, waarbij er geen grotere schijf op een kleinere ligt. De andere 2 palen zijn leeg.

**Opdracht:** Breng de hele toren (zo snel mogelijk) naar een van de lege palen door het een voor een verplaatsen van schijven van de ene paal naar de andere. Alleen de *bovenste* schijf van een stapel kan verzet worden, en deze mag alleen *bovenop* een andere stapel gelegd worden. *Restrictie*: er mag nooit een grotere schijf op een kleinere gelegd worden.

Een **toestand** is in dit geval een verdeling van de schijven over de palen, waarbij (als gevolg van de restrictie) geen grotere schijf op een kleinere ligt. Een **actie** is het verplaatsen van een schijf volgens de spelregels.



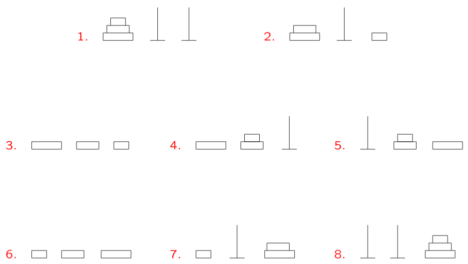






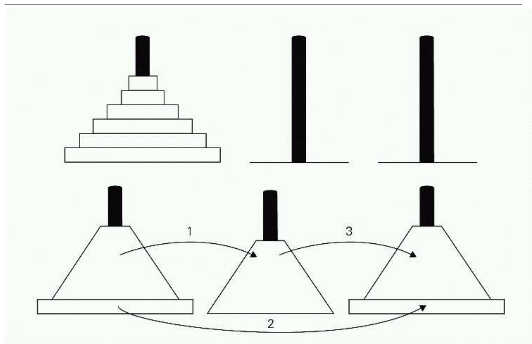



**Optimale oplossing voor $n = 3$.**








Recursieve oplossing van de Torens van Hanoi



# Een Intermezzo





**Voorbeeld 4: Kannenprobleem**

We hebben twee kannen: een grote met een inhoud van 8 liter, en een kleine met een inhoud van 5 liter. Op de kannen staat geen maatverdeling. Verder hebben we de beschikking over een waterkraan en een afvoer. Bij aanvang zijn beide kannen leeg.

**Vraag:** Hoe krijgen we precies 4 liter water in een van de twee kannen? En liefst zo snel mogelijk.





We onderscheiden toestanden en zinvolle (!) acties:

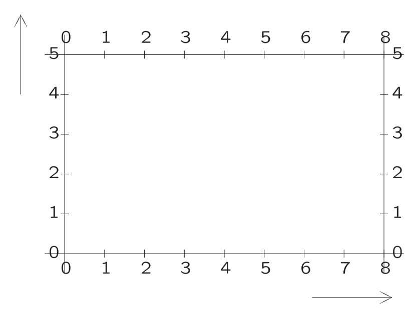
**Toestand**: Een paar $(x, y)$ met $0 \leq x \leq 8$ en $0 \leq y \leq 5$. Hierin is $x$ de inhoud van de grote kan en $y$ de inhoud van de kleine kan.
**Begintoestand**: beide kannen leeg, dus (0,0)
**Eindtoestanden**: alle toestanden met 4 liter in een van beide kannen, dus (4, y) en (x, 4)

**Acties**: vullen, legen en overgieten

- een kan geheel (aan)vullen
- een kan geheel leeggooien
- de ene kan leeggooien in de andere
- van de ene kan in de andere gieten totdat deze vol is

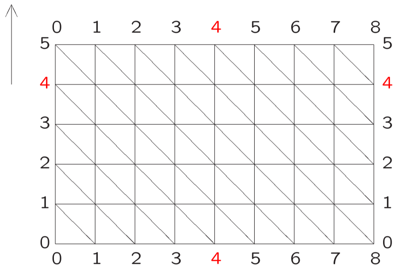






## … met mogelijke acties

inhoud kleine kan



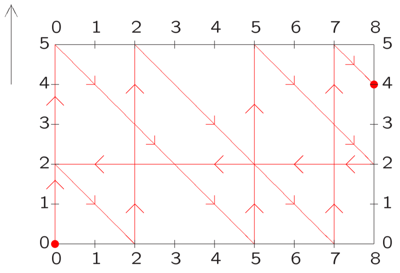

inhoud grote kan





## Oplossing

inhoud kleine kan




inhoud grote kan





## Snelste oplossing

De snelste oplossing gebruikt de volgende strategie en zorgt voor 4 liter in de kleine kan. Er is overigens ook een (iets) langere oplossing, die 4 liter in de grote kan achterlaat.

```
Herhaal
    Herhaal
        Vul de kleine kan;
        Giet over in de grote kan;
    totdat de grote kan vol is
    Grote kan leeggooien;
    Giet uit de kleine kan over in de grote kan;
totdat oplossing gevonden
```

Zie verder het college.





## (Werk)college

- **Lezen/leren bij dit college:**
  Paragraaf 6.6 (subparagraaf 'Reduction to Graph Problems')
  Paragraaf 4.5 (vanaf halverwege blz. 189)
- Week 9: aan de programmeeropdracht werken !!!!
- **Werkcollege:**
  donderdag 23 februari 2012, 13:45–15:30, in zaal Plein
- **Opgaven:**
  zie http://www.liacs.nl/home/rvvliet/algoritmiek/
- **Volgend college:**
  donderdag 1 maart 2012 (in zaal Archipel!)