Derde college algoritmiek

20 februari 2009

Toestand-actie-ruimte en Complexiteit

Probleem $\longrightarrow$ Toestand-actie-ruimte

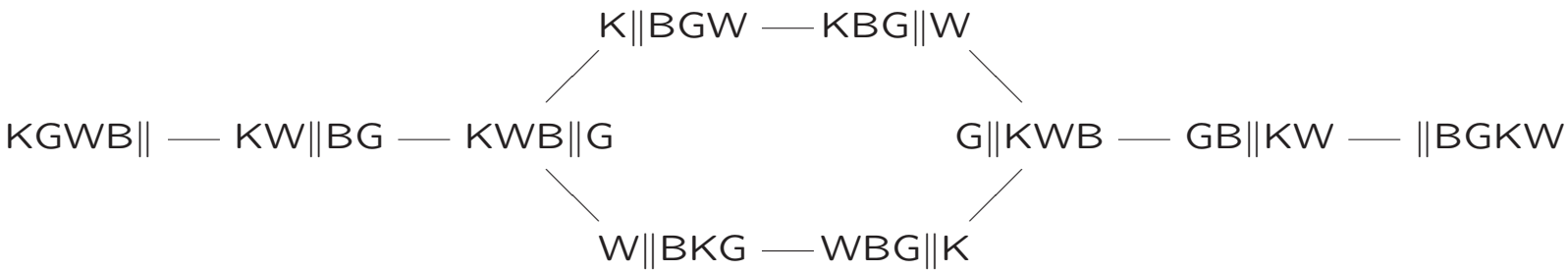
Een toestand-actie-ruimte (toestand-actie-diagram, state transition diagram, toestandsruimte, state space)

- *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 2: Old world puzzle (Ex. 1.2.1.)

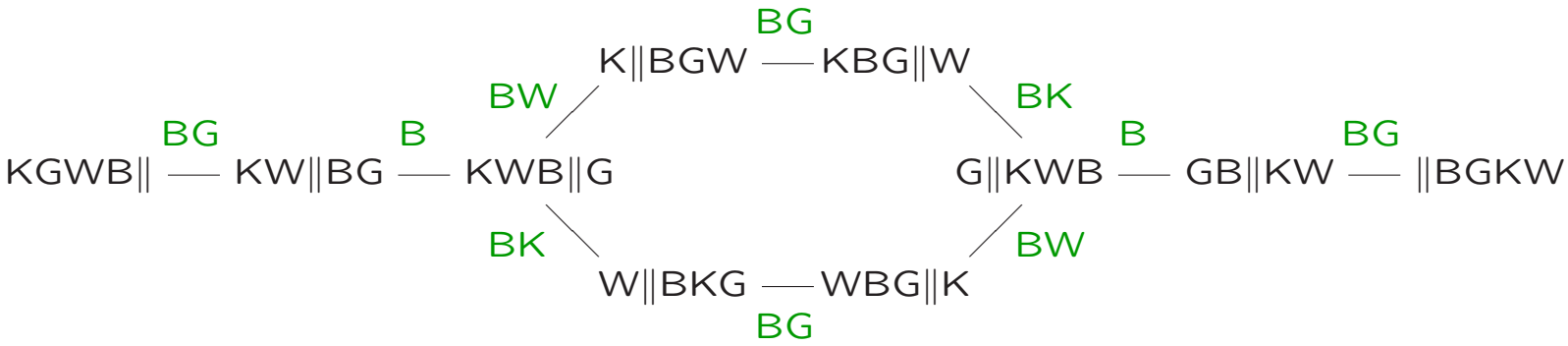
We hebben een kool, een geit, een wolf en een boer. Deze moeten met een bootje van de ene kant van de rivier naar de andere. In het bootje kan alleen de boer met één ander iets. Als de boer er niet bij is zal de wolf de geit opeten en de geit de kool. De boer is de enige die de boot kan "besturen".

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.

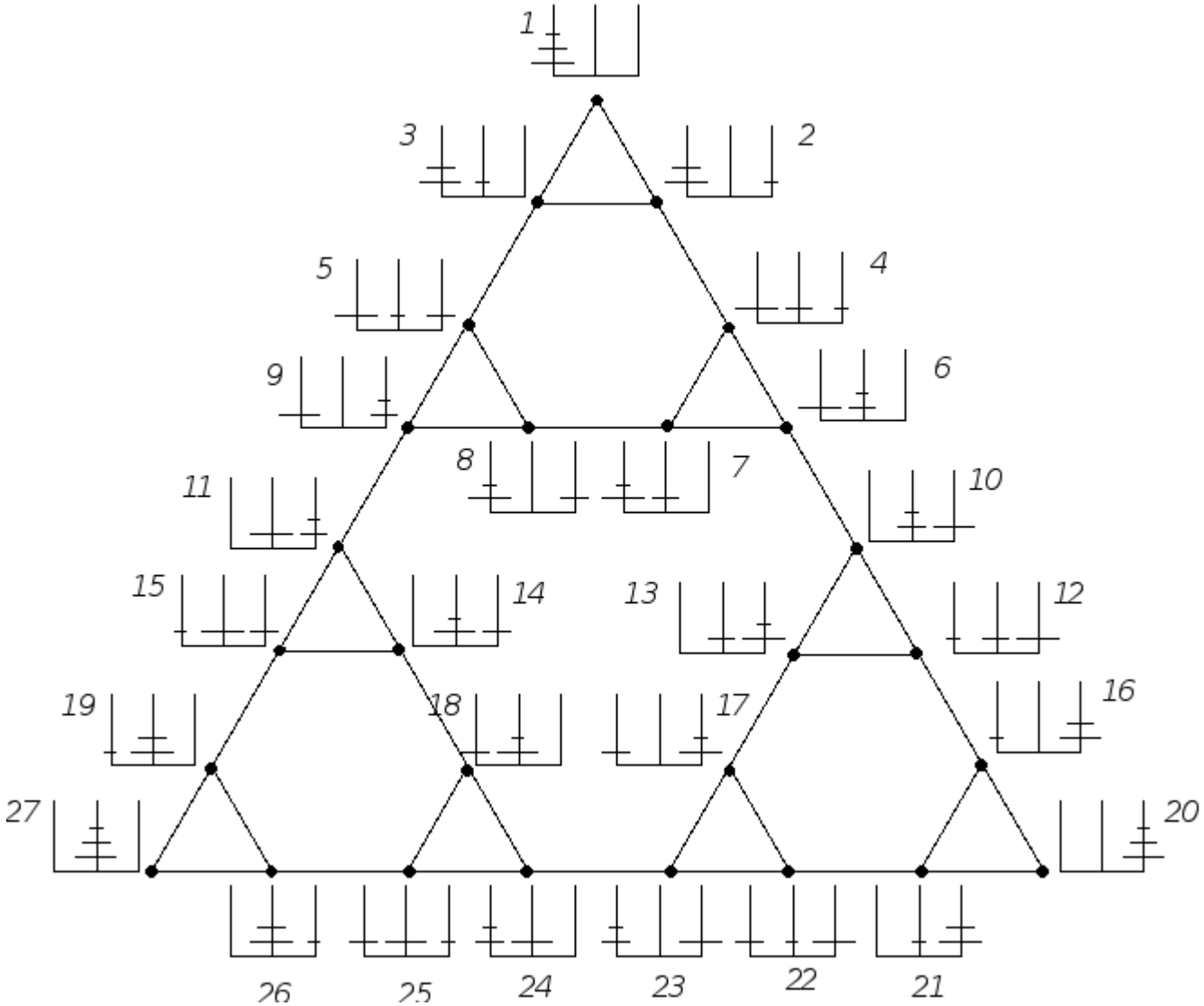
Voorbeeld 3: Torens van Hanoi

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.

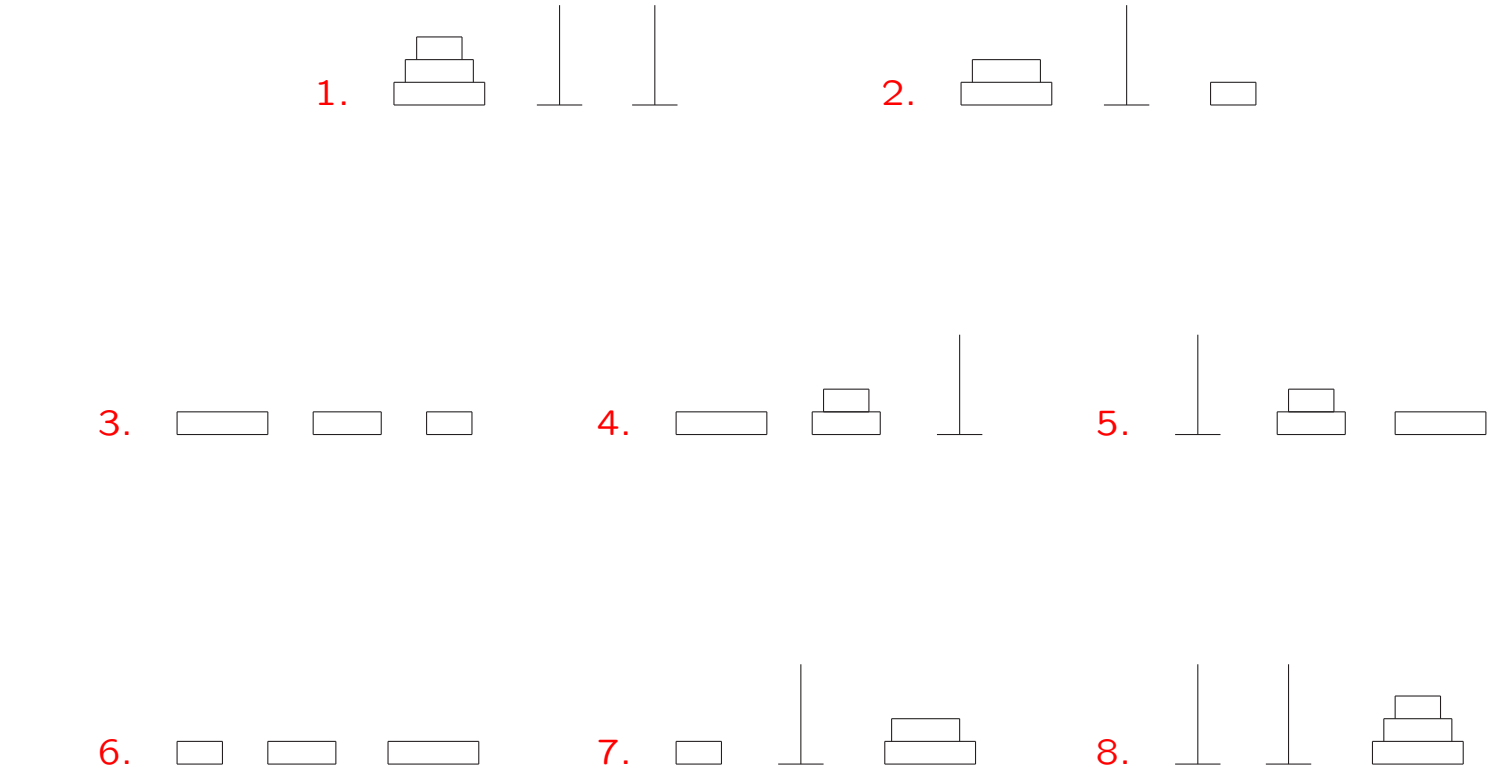
Opdracht: Breng de hele toren (zo snel mogelijk) naar een van de lege palen door het een voor een verplaatsen van schijven van de ene paal naar de andere. Alleen de *bovenste* schijf van een stapel kan verzet worden, en deze mag alleen *bovenop* een andere stapel gelegd worden. *Restrictie*: er mag nooit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

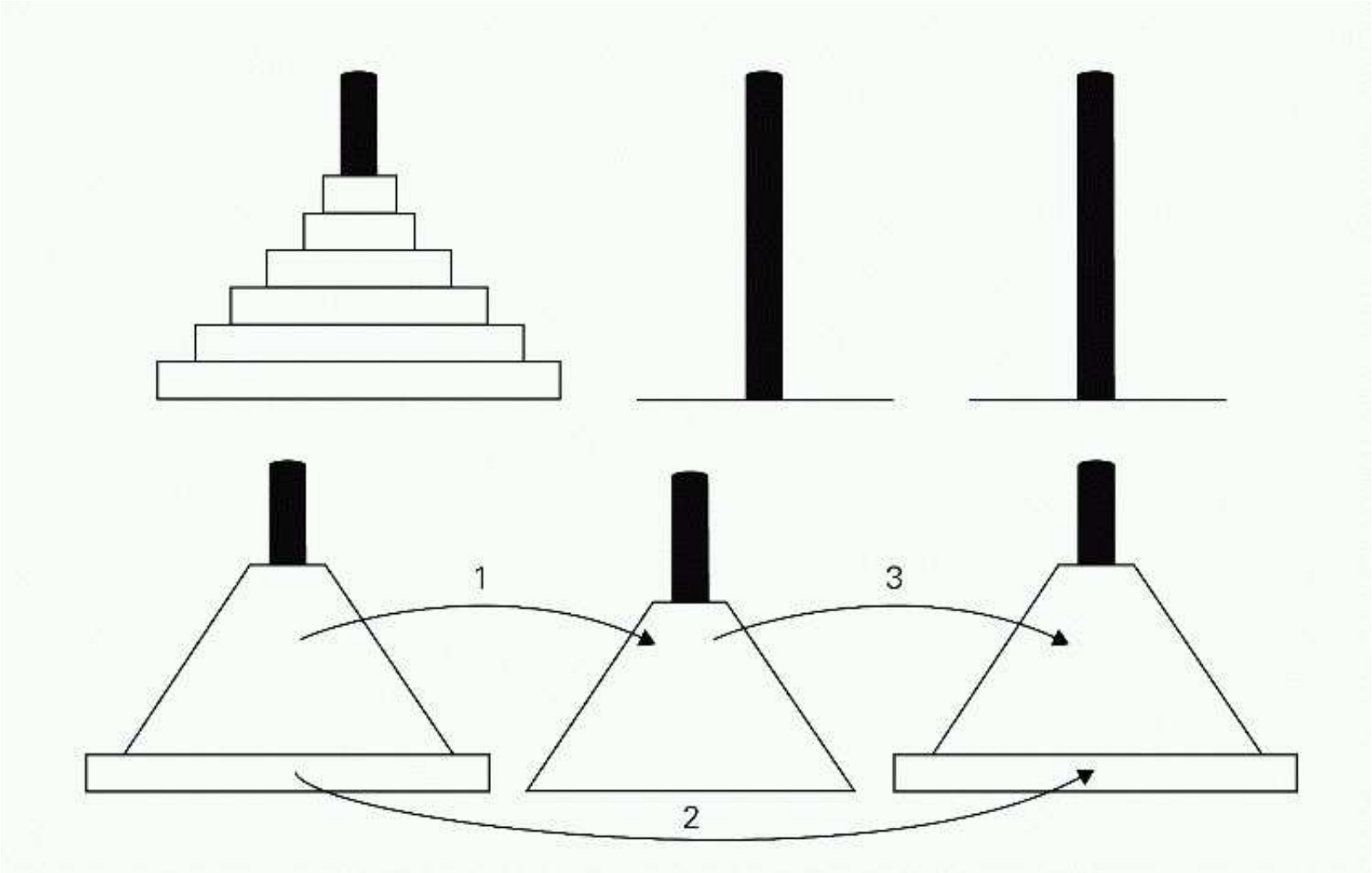
Toestand-actie-ruimte $n = 3$



Optimale oplossing voor $n = 3$.



Oplossing Hanoi algemeen



Recursieve oplossing van de Torens van Hanoi

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.

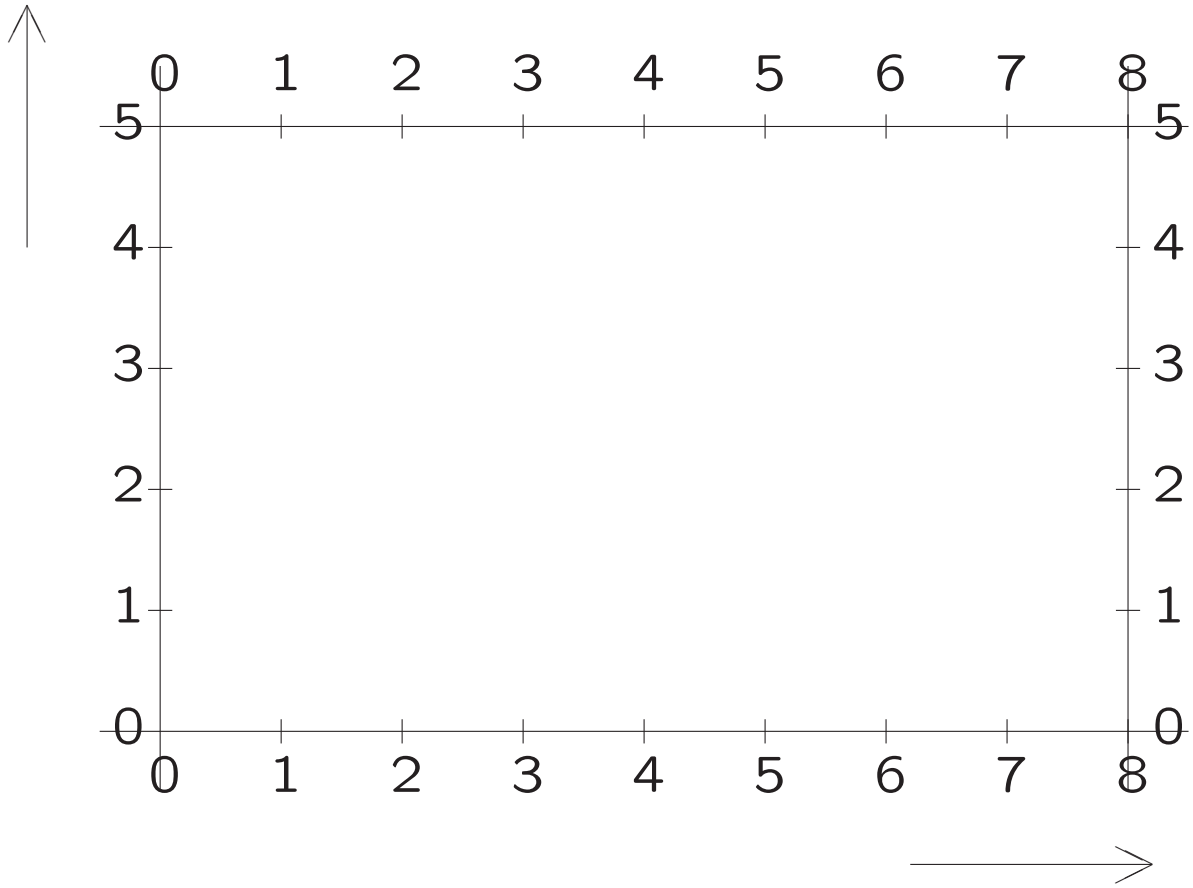
We onderscheiden toestanden en zinvolle (!) acties:

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

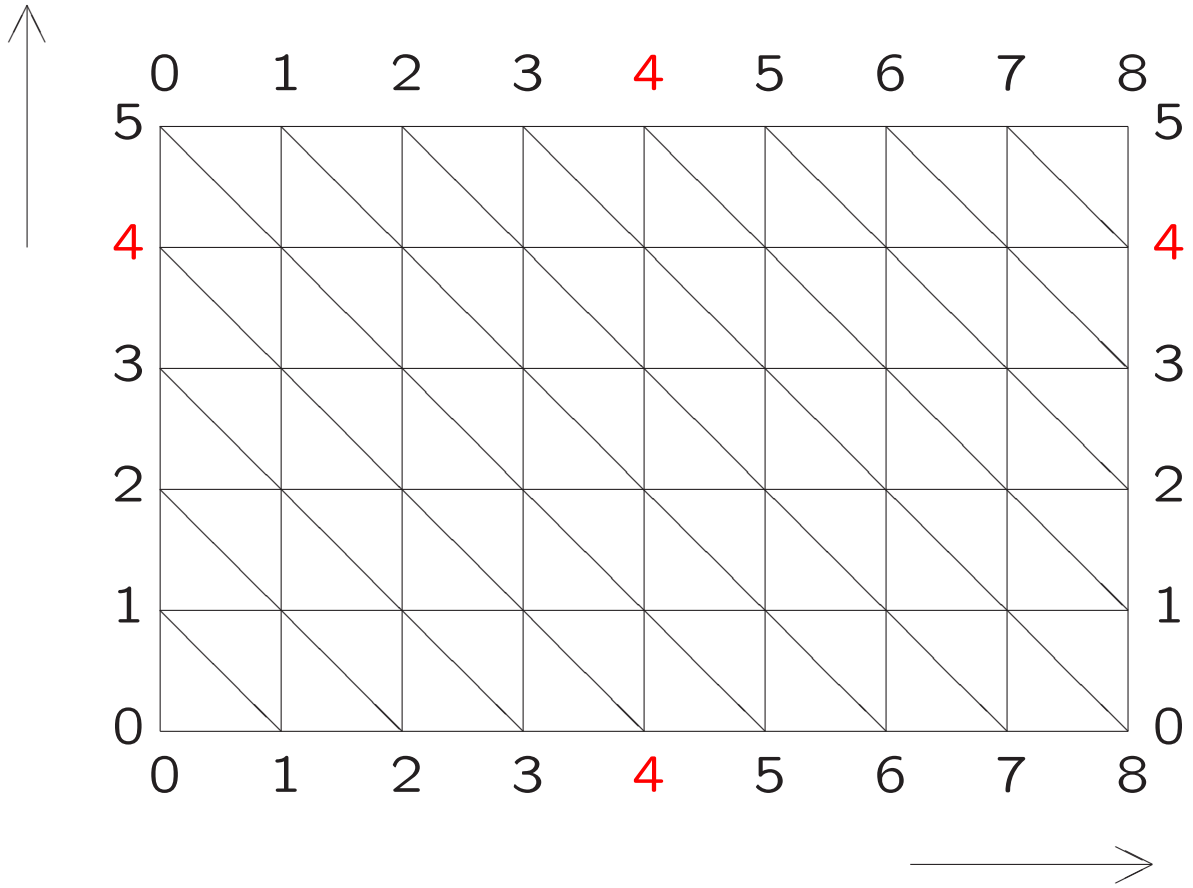
inhoud kleine kan



inhoud grote kan

... met mogelijke acties

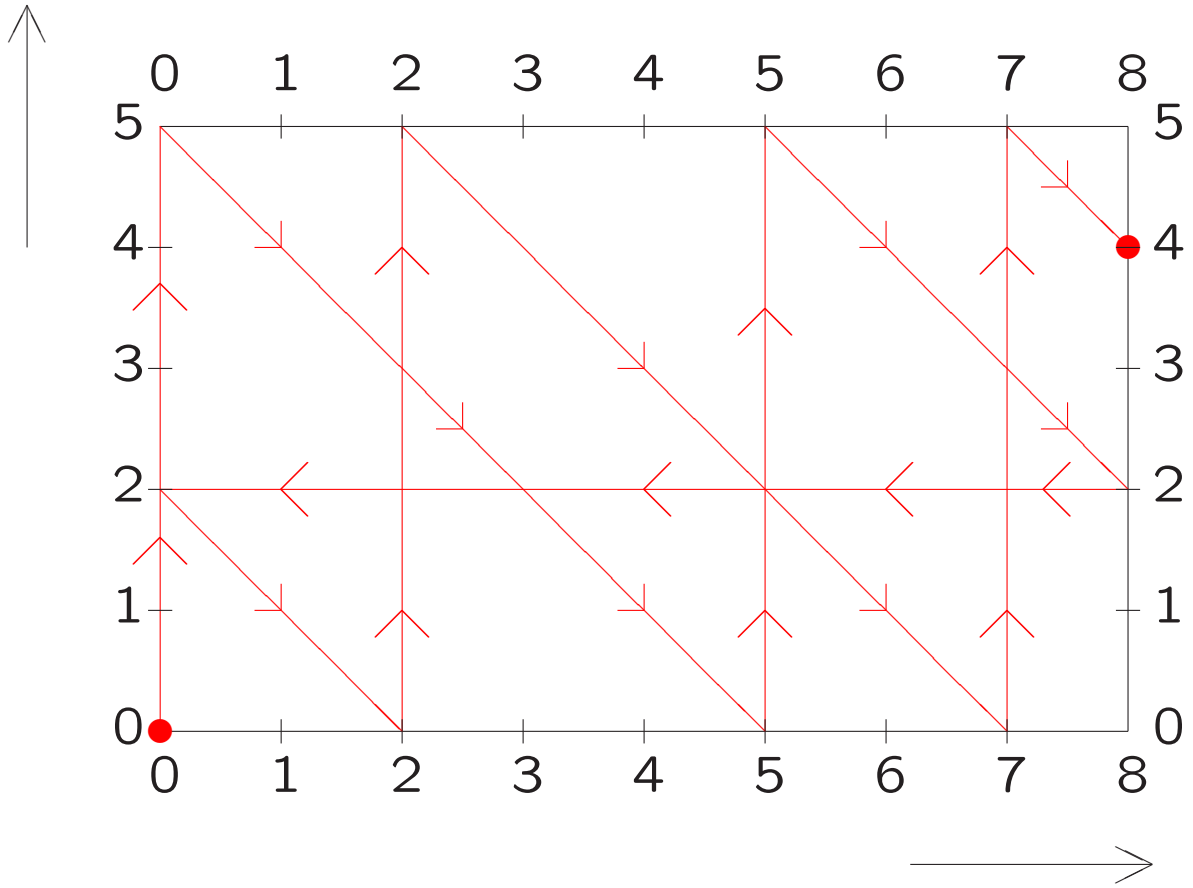
inhoud kleine kan



inhoud grote kan

Oplossing

inhoud kleine kan



inhoud grote kan

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.

Complexiteit (= tijdcomplexiteit) van een algoritme:

- = hoeveelheid werk verricht door het algoritme
- hangt meestal af van de grootte van de invoer: hoe groter de invoer, hoe groter de complexiteit
- wordt bepaald door het aantal keer dat de basisoperatie wordt uitgevoerd
- het belangrijkst is de (asymptotische) groei
- wordt vaak uitgedrukt in O-notatie (orde van grootte)
- hangt vaak ook af van het soort invoer: worst case, best case, average case

Voorbeeld 1:

```
// invoer: array a[0···n − 1] bestaande uit n reals
// uitvoer: de grootste waarde
```

```
max := a[0];
for i := 1 to n − 1 do
    if ( a[i] > max ) then ⟵ basisoperatie
        max := a[i];
    fi
od
return max;
```

Complexiteit: $C(n) = n - 1 \in \Theta(n)$

Voorbeeld 2:

```
// invoer: array a[0···n − 1] bestaande uit n reals
// uitvoer: true als alle n waarden verschillen
```

```
for i := 0 to n − 2 do
    for j := i + 1 to n − 1 do
        if ( a[i] = a[j] ) then ⟵ basisoperatie
            return false; fi
    od
od
return true;
```

Best case complexiteit: $B(n) = 1 \in \Theta(1)$

Worst case complexiteit:

$$W(n) = \sum_{i=0}^{n-2} (n-1-i) = \frac{1}{2} n(n-1) \in \Theta(n^2)$$

- **Werkcollege:**

  donderdag 26 februari 2009 in zaal 174

- **Opgaven:**

  zie http://www.liacs.nl/home/graaf/ALGO/

- **Volgend college:**

  vrijdag 27 februari 2009