

- Een bedrijf kan 5 miljoen euro besteden om zijn fabrieken uit te breiden.
- Elke fabriek overlegt een aantal voorstellen hoe het het geld zal investeren.
- De uitvoering van elk voorstel heeft bepaalde kosten (c) en geeft bepaalde winst (r).
- In elke fabriek mag slechts één voorstel worden uitgevoerd.

Voorstel	Fabriek 1		Fabriek 2		Fabriek 3	
	c_1	r_1	c_2	r_2	c_3	r_3
1	0	0	0	0	0	0
2	1	5	2	8	1	4
3	2	6	3	9	-	-
4	-	-	4	12	-	-

Welke voorstellen moet het bedrijf kiezen zodat de totale winst het grootst is?

We kunnen het probleem oplossen door alle mogelijkheden langs te gaan en de beste te kiezen. Probleem met deze manier van oplossen: teveel rekenwerk bij grote problemen.

Splitsen het probleem in 3 stappen (stages).

- stap 1: fabriek 1
- stap 2: fabriek 2
- stap 3: fabriek 3

We ordenen de stappen volgens deze nummering.

Bij elke stap hebben **toestanden (states)**.

voor stap 1: de hoeveelheid geld dat besteed wordt aan fabriek 1.

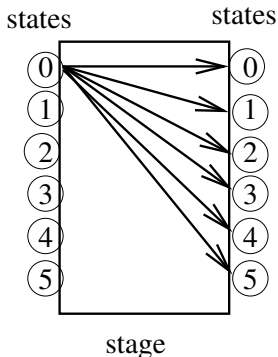
$$s_1 \in \{0, 1, 2, 3, 4, 5\}.$$

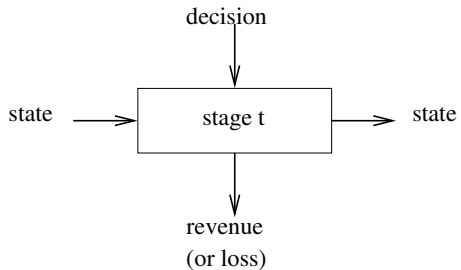
voor stap 2: de hoeveelheid geld dat besteed wordt aan fabrieken 1 en 2.

$$s_2 \in \{0, 1, 2, 3, 4, 5\}.$$

voor stap 3: de hoeveelheid geld dat besteed wordt aan fabrieken 1,2 en 3.

$$s_3 \in \{0, 1, 2, 3, 4, 5\}.$$





voorstel voor fabriek 1						
s_1	1	2	3	4	optimale voorstel	winst fabriek 1
0	0	-	-	-	1	0
1	0	5	-	-	2	5
2	0	5	6	-	3	6
3	0	5	6	-	3	6
4	0	5	6	-	3	6
5	0	5	6	-	3	6
voorstel voor fabriek 2						
s_2	1	2	3	4	optimale voorstel	winst fabrieken 1 en 2
0	0	-	-	-	1	0
1	5	-	-	-	1	5
2	6	8	-	-	2	8
3	6	13	9	-	2	13
4	6	14	14	12	2,3	14
5	6	14	15	17	4	17

s_3	voorstel voor fabriek 3		optimale voorstel	winst fabrieken 1,2 en 3
	1	2		
0	0	-	1	0
1	5	4	1	5
2	8	9	2	9
3	13	12	1	13
4	14	17	2	17
5	17	18	2	18

(Omdat we de totale winst willen maximaliseren hadden we $s_3 \in \{0, 1, 2, 3, 4\}$ niet hoeven uit te rekenen.)

- $r(k_j) =$ winst van voorstel k_j in stap j
- $c(k_j) =$ kosten van voorstel k_j in stap j
- $f_j(s_j) =$ winst van toestand s_j in stap j

Dan

$$f_1(s_1) = \max_{k_1: c(k_1) \leq s_1} \{r(k_1)\}$$

en

$$f_j(s_j) = \max_{k_j: c(k_j) \leq s_j} \{r(k_j) + f_{j-1}(s_j - c(k_j))\}$$

voor $j = 2, 3$.

Berekeningen worden recursief gedaan.

- stap 2 hangt alleen af van stap 1
- stap 3 hangt alleen af van stap 2

Principe van optimaliteit

Als je in een bepaalde toestand bent, worden alle toekomstige beslissingen genomen onafhankelijk van hoe je in kwam in die toestand.

Een dynamisch programmerings model (dynamisch programma) bestaat uit 3 basiselementen:

- 1 definitie van de **stappen**
- 2 definitie van de **keuzemogelijkheden** in elke stap
- 3 definitie van de **toestanden** in elke stap

Bij elke stap maken we

s_i	keuzemogelijkheden	optimale keuzemogelijkheden	winst

We gebruikten voorwaartse recursie (forward recursion).

voorwaartse recursie

stap 1 \rightarrow stap 2 \rightarrow stap 3

terugwaartse recursie

stap 1 \leftarrow stap 2 \leftarrow stap 3

Oplossen probleem met terugwaartse recursie

De toestanden zijn

- stage 3:** de overgebleven hoeveelheid geld dat beschikbaar is voor fabriek 3 (nadat we geld besteed hebben aan fabrieken 1 en 2). $s_3 \in \{0, 1\}$.
- stage 2:** de overgebleven hoeveelheid geld dat beschikbaar is voor fabrieken 2 en 3. $s_2 \in \{0, 1, 2, 3, 4, 5\}$.
- stage 1:** de (overgebleven) hoeveelheid geld dat beschikbaar is voor fabrieken 1,2 en 3. $s_1 \in \{5\}$.

	voorstel voor fabriek 3			
s_3	1	2	optimale voorstel	winst bij fabriek 3
0	0	-	1	0
1	0	4	2	4

		voorstel voor fabriek 2					
s_2	1	2	3	4	optimale voorstel	winst bij fabrieken 2 en 3	
0	0	-	-	-	1	0	
1	4	-	-	-	1	4	
2	4	8	-	-	2	8	
3	4	12	9	-	2	12	
4	4	12	13	12	3	13	
5	4	12	13	16	4	16	
		voorstel voor fabriek 1					
s_1	1	2	3	4	optimale voorstel	winst bij fabrieken 1,2 en 3	
5	16	18	18	-	2,3	18	

De recursie formules voor dit dynamische programmering model zijn:

$$f_3(s_3) = \max_{k_3: c(k_3) \leq s_3} \{r(k_3)\}$$

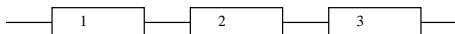
en

$$f_j(s_j) = \max_{k_j: c(k_j) \leq s_j} \{r(k_j) + f_{j+1}(s_j - c(k_j))\}$$

voor $j = 1, 2$.

De recursie formules kunnen ook een andere vorm hebben.

Een elektronische apparaat bestaat uit drie componenten.



Het stukgaan van één van de componenten leidt tot het niet-functioneren van het apparaat.

- Door "standby" onderdelen parallel te installeren in elke component kunnen we de betrouwbaarheid van het apparaat verbeteren.
- De betrouwbaarheid van het apparaat is het product van de betrouwbaarheden van de componenten.
- Het apparaat mag niet meer kosten dan 10 euro.
- We willen dat de betrouwbaarheid zo hoog mogelijk is.

aantal "standby" onderdelen	component 1		component 2		component 3	
	r_1	c_1	r_2	c_2	r_3	c_3
1	0.6	1	0.7	3	0.5	2
2	0.8	2	0.8	5	0.7	4
3	0.9	3	0.9	6	0.9	5

r_i = betrouwbaarheid component i , c_i = kosten component i .

Wat zijn de **stappen**, **keuzemogelijkheden** en **toestanden**?

- stap 1 wordt gerepresenteerd door component 1
- stap 2 wordt gerepresenteerd door component 2
- stap 3 wordt gerepresenteerd door component 3

De keuzemogelijkheden in elke stap zijn de aantal "standby" onderdelen.
De toestanden in

- stap 3:** De overgebleven hoeveelheid geld beschikbaar voor component 3. $s_3 \in \{2, 3, 4, 5, 6\}$.
- stap 2:** De overgebleven hoeveelheid geld beschikbaar voor componenten 2 en 3. $s_2 \in \{5, 6, 7, 8, 9\}$.
- stap 1:** De (overgebleven) hoeveelheid geld beschikbaar voor componenten 1,2 en 3. $s_1 \in \{6, 7, 8, 9, 10\}$.

Welke recursie formules hebben we?

- k_j = aantal "standby" onderdelen voor stap (component) i
- $c_j(k_j)$ = kosten om k_j "standby" onderdelen te hebben in component j
- $r_j(k_j)$ = betrouwbaarheid door k_j "standby" onderdelen te hebben in component j

De recursie formules zijn:

$$f_3(s_3) = \max_{k_3: c_3(k_3) \leq s_3} \{r_3(k_3)\}$$

en

$$f_j(s_j) = \max_{k_j: c_j(k_j) \leq s_j} \{r_j(k_j) \cdot f_{j+1}(s_j - c_j(k_j))\}$$

	aantal "standby" onderdelen				
s_3	1	2	3	x_3^*	betrouwbrhd component 3
2	0.5	-	-	1	0.5
3	0.5	-	-	1	0.5
4	0.5	0.7	-	2	0.7
5	0.5	0.7	0.9	3	0.9
6	0.5	0.7	0.9	3	0.9
	aantal "standby" onderdelen				
s_2	1	2	3	x_2^*	betrouwbrhd componenten 2 en 3
5	0.35	-	-	1	0.35
6	0.35	-	-	1	0.35
7	0.49	0.4	-	1	0.49
8	0.63	0.4	0.45	1	0.63
9	0.63	0.56	0.45	1	0.63

s_1	aantal "standby" onderdelen			x_1^*	betrouwbrhd component 1,2 en 3
	1	2	3		
6	0.21	-	-	1	0.21
7	0.21	0.28	-	2	0.28
8	0.294	0.28	0.315	3	0.315
9	0.378	0.392	0.315	2	0.392
10	0.378	0.504	0.441	2	0.504

We zien dat

- component 1 twee standby onderdelen moet hebben
- component 2 één standby onderdeel moet hebben
- component 3 drie standby onderdelen moet hebben