Tussentoets

- 26 november, tijdens de instructies (10:45–11:30)
- Bij de tussentoets mag een eenvoudige (niet grafische; niet programmeerbare) rekenmachine meegenomen worden, en 1 tweezijdig A4-tje met daarop handgeschreven aantekeningen die men nodig denkt te hebben bij het maken van de toets
- geen boeken, geen kopieën

Zaal: paviljoen study hub
Time: 90min

Waar waren we ook al weer gebleven?

Vorige keer

- Transportproblemen
- Toewijzingsproblemen

Vandaag

- Dynamische programmering

Dynamische programmeringsproblemen (1)

- Bedrijf kan 5 M euro investeren om fabrieken uit te breiden.
- Elke fabriek presenteert aantal voorstellen hoe het geld te investeren.
- Uitvoering van elk voorstel brengt bepaalde kosten (c) met zich mee en levert bepaalde winst (r) op.
- In elke fabriek mag slechts één voorstel worden uitgevoerd.

	Fabriek 1		Fabriek 2		Fabriek 3	
Voorstel	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	_	_

Probleem Welke voorstellen moet bedrijf selecteren om totale winst te maximaliseren?

Dynamische programmeringsproblemen (2)

- Dom algoritme ('enumeratie'): alle mogelijkheden aflopen en oplossing nemen die grootste waarde geeft.
- Veel te veel werk, we kunnen het slimmer aanpakken $\Rightarrow$ Dynamisch programmeren

Dynamische programmeringsproblemen (3)

We splitsen probleem in drie stappen (stages):

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

Dynamische programmeringsproblemen (4)

Bij elke stap hebben we toestanden (states):

voor stap 1: hoeveelheid geld we besteden aan fabriek 1;
$s_1 \in \{0, 1, 2, 3, 4, 5\}$

voor stap 2: hoeveelheid geld we besteden aan fabrieken 1 en 2;
$s_2 \in \{0, 1, 2, 3, 4, 5\}$

voor stap 3: hoeveelheid geld we besteden aan fabrieken 1,2 en 3;
$s_3 \in \{0, 1, 2, 3, 4, 5\}$

	fabriek 1	
Voorstel	kosten	winst
1	0	0
2	1	5
3	2	6
4	–	–

Stap 1

winst voor fabriek 1 bij voorstel k_1 voor fabriek 1

s_1	voorstel $k_1 = 1$	voorstel $k_1 = 2$	voorstel $k_1 = 3$	optimale voorstel	winst fabriek 1
0	0	–	–	1	0

	fabriek 1	
Voorstel	kosten	winst
1	0	0
2	1	5
3	2	6
4	–	–

Stap 1

winst voor fabriek 1 bij voorstel k_1 voor fabriek 1

s_1	voorstel $k_1 = 1$	voorstel $k_1 = 2$	voorstel $k_1 = 3$	optimale voorstel	winst fabriek 1
0	0	–	–	1	0
1	0	5	–	2	5

	fabriek 1	
Voorstel	kosten	winst
1	0	0
2	1	5
3	2	6
4	–	–

Stap 1

winst voor fabriek 1 bij voorstel k_1 voor fabriek 1

s_1	voorstel $k_1 = 1$	voorstel $k_1 = 2$	voorstel $k_1 = 3$	optimale voorstel	winst fabriek 1
0	0	–	–	1	0
1	0	5	–	2	5
2	0	5	6	3	6

	fabriek 1	
Voorstel	kosten	winst
1	0	0
2	1	5
3	2	6
4	–	–

Stap 1

winst voor fabriek 1 bij voorstel k_1 voor fabriek 1

s_1	voorstel $k_1 = 1$	voorstel $k_1 = 2$	voorstel $k_1 = 3$	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

	fabriek 1	
Voorstel	kosten	winst
1	0	0
2	1	5
3	2	6
4	–	–

Stap 1

winst voor fabriek 1 bij voorstel k_1 voor fabriek 1

s_1	voorstel $k_1 = 1$	voorstel $k_1 = 2$	voorstel $k_1 = 3$	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

	fabriek 1	
Voorstel	kosten	winst
1	0	0
2	1	5
3	2	6
4	–	–

Stap 1

winst voor fabriek 1 bij voorstel k_1 voor fabriek 1

s_1	voorstel $k_1 = 1$	voorstel $k_1 = 2$	voorstel $k_1 = 3$	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

	fabriek 1	
Voorstel	kosten	winst
1	0	0
2	1	5
3	2	6
4	–	–

Stap 1

winst voor fabriek 1 bij voorstel k_1 voor fabriek 1

$\mathbf{s_1}$	voorstel $k_1 = 1$	voorstel $k_1 = 2$	voorstel $k_1 = 3$	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**

	fabriek 2	
voorstel	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_1	winst fabriek 1
0	0
1	5
2	6
3	6
4	6
5	6

Stap 2

winst fabrieken 1 en 2 bij voorstel k_2 voor fabriek 2

s_2	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	winst fab. 1+2
0	0	–	–	–	1	0

	fabriek 2	
voorstel	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_1	winst fabriek 1
0	0
1	5
2	6
3	6
4	6
5	6

Stap 2

winst fabrieken 1 en 2 bij voorstel k_2 voor fabriek 2

s_2	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	winst fab. 1+2
0	0	–	–	–	1	0
1	5	–	–	–	1	5

	fabriek 2	
voorstel	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_1	winst fabriek 1
0	0
1	5
2	6
3	6
4	6
5	6

Stap 2

winst fabrieken 1 en 2 bij voorstel k_2 voor fabriek 2

s_2	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	winst fab. 1+2
0	0	–	–	–	1	0
1	5	–	–	–	1	5
2	6	8	–	–	2	8

	fabriek 2	
voorstel	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_1	winst fabriek 1
0	0
1	5
2	6
3	6
4	6
5	6

Stap 2

winst fabrieken 1 en 2 bij voorstel k_2 voor fabriek 2

s_2	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	winst fab. 1+2
0	0	–	–	–	1	0
1	5	–	–	–	1	5
2	6	8	–	–	2	8
3	6	13	9	–	2	13

	fabriek 2	
voorstel	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_1	winst fabriek 1
0	0
1	5
2	6
3	6
4	6
5	6

Stap 2

winst fabrieken 1 en 2 bij voorstel k_2 voor fabriek 2

s_2	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	winst fab. 1+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 of 3	14

	fabriek 2	
voorstel	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_1	winst fabriek 1
0	0
1	5
2	6
3	6
4	6
5	6

Stap 2

winst fabrieken 1 en 2 bij voorstel k_2 voor fabriek 2

s_2	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	winst fab. 1+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 of 3	14
5	6	14	15	17	4	17

	fabriek 2	
voorstel	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_1	winst fabriek 1
0	0
1	5
2	6
3	6
4	6
5	6

Stap 2

winst fabrieken 1 en 2 bij voorstel k_2 voor fabriek 2

$\mathbf{s_2}$	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	**winst fab. 1+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 of 3	**14**
5	6	14	15	17	4	**17**

	fabriek 3	
voorstel	kosten	winst
1	0	0
2	1	4
3	–	–
4	–	–

s_2	winst fabrieken 1 en 2
0	0
1	5
2	8
3	13
4	14
5	17

Stap 3

winst fabrieken 1, 2 en 3 bij voorstel k_3 voor fabriek 3

s_3	bij voorstel $k_3 = 1$	bij voorstel $k_3 = 2$	optimale voorstel	winst fabrieken 1,2 en 3
0	0	–	1	0

	fabriek 3	
voorstel	kosten	winst
1	0	0
2	1	4
3	-	-
4	-	-

s_2	winst fabrieken 1 en 2
0	0
1	5
2	8
3	13
4	14
5	17

Stap 3

winst fabrieken 1, 2 en 3 bij voorstel k_3 voor fabriek 3

s_3	bij voorstel $k_3 = 1$	bij voorstel $k_3 = 2$	optimale voorstel	winst fabrieken 1,2 en 3
0	0	-	1	0
1	5	4	1	5

	fabriek 3	
voorstel	kosten	winst
1	0	0
2	1	4
3	-	-
4	-	-

s_2	winst fabrieken 1 en 2
0	0
1	5
2	8
3	13
4	14
5	17

Stap 3

winst fabrieken 1, 2 en 3 bij voorstel k_3 voor fabriek 3

s_3	bij voorstel $k_3 = 1$	bij voorstel $k_3 = 2$	optimale voorstel	winst fabrieken 1,2 en 3
0	0	-	1	0
1	5	4	1	5
2	8	9	2	9

	fabriek 3	
voorstel	kosten	winst
1	0	0
2	1	4
3	-	-
4	-	-

s_2	winst fabrieken 1 en 2
0	0
1	5
2	8
3	13
4	14
5	17

Stap 3

winst fabrieken 1, 2 en 3 bij voorstel k_3 voor fabriek 3

s_3	bij voorstel $k_3 = 1$	bij voorstel $k_3 = 2$	optimale voorstel	winst fabrieken 1,2 en 3
0	0	-	1	0
1	5	4	1	5
2	8	9	2	9
3	13	12	1	13

	fabriek 3	
voorstel	kosten	winst
1	0	0
2	1	4
3	-	-
4	-	-

s_2	winst fabrieken 1 en 2
0	0
1	5
2	8
3	13
4	14
5	17

Stap 3

winst fabrieken 1, 2 en 3 bij voorstel k_3 voor fabriek 3

s_3	bij voorstel $k_3 = 1$	bij voorstel $k_3 = 2$	optimale voorstel	winst fabrieken 1,2 en 3
0	0	-	1	0
1	5	4	1	5
2	8	9	2	9
3	13	12	1	13
4	14	17	2	17

	fabriek 3	
voorstel	kosten	winst
1	0	0
2	1	4
3	-	-
4	-	-

s_2	winst fabrieken 1 en 2
0	0
1	5
2	8
3	13
4	14
5	17

Stap 3

winst fabrieken 1, 2 en 3 bij voorstel k_3 voor fabriek 3

s_3	bij voorstel $k_3 = 1$	bij voorstel $k_3 = 2$	optimale voorstel	winst fabrieken 1,2 en 3
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

	fabriek 3	
voorstel	kosten	winst
1	0	0
2	1	4
3	-	-
4	-	-

s_2	winst fabrieken 1 en 2
0	0
1	5
2	8
3	13
4	14
5	17

Stap 3

winst fabrieken 1, 2 en 3 bij voorstel k_3 voor fabriek 3

s_3	bij voorstel $k_3 = 1$	bij voorstel $k_3 = 2$	optimale voorstel	winst fabrieken 1,2 en 3
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**

Optimale oplossing gevonden:

- Winst is 18 miljoen.
- Fabriek 3 kiest voorstel 2.
- Fabriek 2 kiest voorstel 2 of 3.
- Fabriek 1 kiest voorstel 3.

Merk op: we hadden bij stap 3 eigenlijk alleen $s_3 = 5$ hoeven uit te rekenen.

Dynamische programmeringsproblemen (8)

Introduceer variabelen:

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

Dan geldt:

$$f_1(s_1) = \max_{k_1 : c_1(k_1) \leq s_1} \{r_1(k_1)\}$$

en

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

voor $j = 2, 3$

Dynamische programmeringsproblemen (9)

Berekeningen worden recursief uitgevoerd:

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

Principe van optimaliteit

Als je in bepaalde toestand bent, kunnen alle toekomstige beslissingen worden genomen onafhankelijk van hoe je in die toestand terecht bent gekomen.

Dynamische programmeringsproblemen (10)

Dynamisch programmeringsprobleem bestaat uit drie basiselementen:

1. definitie van stappen
2. definitie van keuzemogelijkheden in elke stap
3. definitie van toestanden in elke stap

Dynamische programmeringsproblemen (11)

Bij elke stap bepalen we tabel met

s_j	winst voor elke keuzemogelijkheid			optimale keuze-mogelijkheden	winst voor optimale keuzemogelijkheden

Dynamische programmeringsproblemen (12)

We gebruikten voorwaartse recursie (forward recursion):
stap 1 $\rightarrow$ stap 2 $\rightarrow$ stap 3

Alternatief is terugwaartse recursie (backward recursion):
stap 1 $\leftarrow$ stap 2 $\leftarrow$ stap 3

Dynamische programmeringsproblemen (13)

Toestanden in eerdere voorbeeld probleem voor terugwaartse recursie:

stap 3: overgebleven hoeveelheid geld beschikbaar voor fabriek 3 (nadat we geld besteed hebben aan fabrieken 1 en 2); $s_3 \in \{0, 1\}$

stap 2: overgebleven hoeveelheid geld beschikbaar voor fabrieken 2 en 3; $s_2 \in \{0, 1, 2, 3, 4, 5\}$

stap 1: (overgebleven) hoeveelheid geld beschikbaar voor fabrieken 1,2 en 3; $s_1 \in \{5\}$

	fabriek 3	
voorstel k_3	kosten	winst
1	0	0
2	1	4
3	–	–
4	–	–

Stap 3

	winst fabriek 3 bij			
s_3	voorstel $k_3 = 1$	voorstel $k_3 = 2$	optimale voorstel	winst fabriek 3
0	0	–	1	0
1	0	4	2	4

	fabriek 3	
voorstel k_3	kosten	winst
1	0	0
2	1	4
3	–	–
4	–	–

Stap 3

	winst fabriek 3 bij			
$\mathbf{s_3}$	voorstel $k_3 = 1$	voorstel $k_3 = 2$	optimale voorstel	**winst fabriek 3**
0	0	–	1	**0**
1	0	4	2	**4**

	fabriek 2	
voorstel k_2	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_3	winst fabriek 3
0	0
1	4

Stap 2

	winst fabrieken 2 en 3 bij voorstel k_2 voor fabriek 2					
s_2	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	winst fabrieken 2 en 3
0	0	–	–	–	1	0

	fabriek 2	
voorstel k_2	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_3	winst fabriek 3
0	0
1	4

Stap 2

	winst fabrieken 2 en 3 bij voorstel k_2 voor fabriek 2					
s_2	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	winst fabrieken 2 en 3
0	0	–	–	–	1	0
1	4	–	–	–	1	4

	fabriek 2	
voorstel k_2	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_3	winst fabriek 3
0	0
1	4

Stap 2

	winst fabrieken 2 en 3 bij voorstel k_2 voor fabriek 2					
s_2	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	winst fabrieken 2 en 3
0	0	–	–	–	1	0
1	4	–	–	–	1	4
2	4	8	–	–	2	8

	fabriek 2	
voorstel k_2	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_3	winst fabriek 3
0	0
1	4

Stap 2

	winst fabrieken 2 en 3 bij voorstel k_2 voor fabriek 2					
s_2	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	winst fabrieken 2 en 3
0	0	–	–	–	1	0
1	4	–	–	–	1	4
2	4	8	–	–	2	8
3	4	12	9	–	2	12

	fabriek 2	
voorstel k_2	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_3	winst fabriek 3
0	0
1	4

Stap 2

	winst fabrieken 2 en 3 bij voorstel k_2 voor fabriek 2					
s_2	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	winst 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

	fabriek 2	
voorstel k_2	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_3	winst fabriek 3
0	0
1	4

Stap 2

	winst fabrieken 2 en 3 bij voorstel k_2 voor fabriek 2					
s_2	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	winst 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

	fabriek 2	
voorstel k_2	kosten	winst
1	0	0
2	2	8
3	3	9
4	4	12

s_3	winst fabriek 3
0	0
1	4

Stap 2

	winst fabrieken 2 en 3 bij voorstel k_2 voor fabriek 2					
$\mathbf{s_2}$	voorstel $k_2 = 1$	voorstel $k_2 = 2$	voorstel $k_2 = 3$	voorstel $k_2 = 4$	optimale voorstel	**winst 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**

	fabriek 1	
voorstel k_1	kosten	winst
1	0	0
2	1	5
3	2	6
4	–	–

s_2	winst fabrieken 2 en 3
0	0
1	4
2	8
3	12
4	13
5	16

Stap 1

	winst fabrieken 1, 2 en 3 bij voorstel k_1 voor fabriek 1				
s_1	voorstel $k_1 = 1$	voorstel $k_1 = 2$	voorstel $k_1 = 3$	optimale voorstel	winst fabrieken 1, 2 en 3
5	16	18	18	2, 3	**18**

	fabriek 1	
voorstel k_1	kosten	winst
1	0	0
2	1	5
3	2	6
4	–	–

s_2	winst fabrieken 2 en 3
0	0
1	4
2	8
3	12
4	13
5	16

Stap 1

	winst fabrieken 1, 2 en 3 bij voorstel k_1 voor fabriek 1				
s_1	voorstel $k_1 = 1$	voorstel $k_1 = 2$	voorstel $k_1 = 3$	optimale voorstel	winst fabrieken 1, 2 en 3
5	16	18	18	2, 3	**18**

Optimale winst: 18 miljoen

Dynamische programmeringsproblemen (15)

Recursie formules voor dit voorbeeld probleem voor terugwaartse recursie zijn:

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

en

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

voor $j = 1, 2$

Recursie formules kunnen in het algemeen andere vorm hebben.

Dynamisch Programmeren: Tweede Voorbeeld (1)

- Elektronisch apparaat bestaat uit drie componenten
- Apparaat functioneert niet wanneer één van componenten defect raakt
- We kunnen betrouwbaarheid van apparaat verbeteren door "standby" onderdelen parallel te installeren in elke component
- Betrouwbaarheid van apparaat is product van betrouwbaarheden van afzonderlijke componenten
- Apparaat mag niet meer kosten dan 10 euro
- We willen dat betrouwbaarheid zo hoog mogelijk is

Dynamisch Programmeren: Tweede Voorbeeld (2)

	component 1		component 2		component 3	
aantal parallelle onderdelen	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_j = betrouwbaarheid component j

c_j = kosten component j

Wat zijn stappen, keuzemogelijkheden en toestanden?

Dynamisch Programmeren: Tweede Voorbeeld (2)

	component 1		component 2		component 3	
aantal parallelle onderdelen	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_j = betrouwbaarheid component j

c_j = kosten component j

Wat zijn stappen, keuzemogelijkheden en toestanden?

- stap 1: component 1
- stap 2: component 2
- stap 3: component 3

Dynamisch Programmeren: Tweede Voorbeeld (3)

	component 1		component 2		component 3	
aantal parallelle onderdelen	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

Keuzemogelijkheden in elke stap zijn aantal parallelle onderdelen

Toestanden in

- stap 3: overgebleven hoeveelheid geld beschikbaar voor component 3 (na besteden van geld aan componenten 1 en 2); $s_3 \in \{2, 3, 4, 5, 6\}$
- stap 2: overgebleven hoeveelheid geld beschikbaar voor componenten 2 en 3; $s_2 \in \{7, 8, 9\}$
- stap 1: (overgebleven) hoeveelheid geld beschikbaar voor componenten 1, 2 en 3; $s_1 = 10$

Dynamisch Programmeren: Tweede Voorbeeld (3)

Introduceer variabelen:

- k_j = aantal parallelle onderdelen voor stap (component) j
- $c_j(k_j)$ = kosten om k_j parallelle onderdelen te hebben in component j
- $r_j(k_j)$ = betrouwbaarheid component j met k_j parallelle onderdelen
- $f_j(s_j)$ = optimale betrouwbaarheid in stap j met toestand (budget) s_j

Dynamisch Programmeren: Tweede Voorbeeld (3)

Introduceer variabelen:

- k_j = aantal parallelle onderdelen voor stap (component) j
- $c_j(k_j)$ = kosten om k_j parallelle onderdelen te hebben in component j
- $r_j(k_j)$ = betrouwbaarheid component j met k_j parallelle onderdelen
- $f_j(s_j)$ = optimale betrouwbaarheid in stap j met toestand (budget) s_j

Recursie formules:

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

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

voor $j = 1, 2$

Dynamisch Programmeren: Tweede Voorbeeld (4)

component 3		
k_3	r_3	c_3
1	0.5	2
2	0.7	4
3	0.9	5

k_3	aantal parallelle onderdelen comp. 3
r_3	betrouwbaarheid component 3
c_3	kosten component 3
s_3	beschikbare geld voor component 3

Stap 3

	betrouwbaarheid component 3 bij k_3 parallelle onderdelen				
s_3	$k_3 = 1$	$k_3 = 2$	$k_3 = 3$	k_3^*	betrouwbaarheid
2	0.5	–	–	1	0.5

Dynamisch Programmeren: Tweede Voorbeeld (4)

component 3		
k_3	r_3	c_3
1	0.5	2
2	0.7	4
3	0.9	5

k_3	aantal parallelle onderdelen comp. 3
r_3	betrouwbaarheid component 3
c_3	kosten component 3
s_3	beschikbare geld voor component 3

Stap 3

	betrouwbaarheid component 3 bij k_3 parallelle onderdelen				
s_3	$k_3 = 1$	$k_3 = 2$	$k_3 = 3$	k_3^*	betrouwbaarheid
2	0.5	–	–	1	0.5
3	0.5	–	–	1	0.5

Dynamisch Programmeren: Tweede Voorbeeld (4)

component 3		
k_3	r_3	c_3
1	0.5	2
2	0.7	4
3	0.9	5

k_3	aantal parallelle onderdelen comp. 3
r_3	betrouwbaarheid component 3
c_3	kosten component 3
s_3	beschikbare geld voor component 3

Stap 3

	betrouwbaarheid component 3 bij k_3 parallelle onderdelen				
s_3	$k_3 = 1$	$k_3 = 2$	$k_3 = 3$	k_3^*	betrouwbaarheid
2	0.5	–	–	1	0.5
3	0.5	–	–	1	0.5
4	0.5	0.7	–	2	0.7

Dynamisch Programmeren: Tweede Voorbeeld (4)

component 3		
k_3	r_3	c_3
1	0.5	2
2	0.7	4
3	0.9	5

k_3	aantal parallelle onderdelen comp. 3
r_3	betrouwbaarheid component 3
c_3	kosten component 3
s_3	beschikbare geld voor component 3

Stap 3

	betrouwbaarheid component 3 bij k_3 parallelle onderdelen				
s_3	$k_3 = 1$	$k_3 = 2$	$k_3 = 3$	k_3^*	betrouwbaarheid
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

Dynamisch Programmeren: Tweede Voorbeeld (4)

component 3		
k_3	r_3	c_3
1	0.5	2
2	0.7	4
3	0.9	5

k_3	aantal parallelle onderdelen comp. 3
r_3	betrouwbaarheid component 3
c_3	kosten component 3
s_3	beschikbare geld voor component 3

Stap 3

	betrouwbaarheid component 3 bij k_3 parallelle onderdelen				
s_3	$k_3 = 1$	$k_3 = 2$	$k_3 = 3$	k_3^*	betrouwbaarheid
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

Dynamisch Programmeren: Tweede Voorbeeld (5)

component 2		
k_2	r_2	c_2
1	0.7	3
2	0.8	5
3	0.9	6

s_3	betrouwbaarheid component 3
2	0.5
3	0.5
4	0.7
5	0.9
6	0.9

Stap 2

	betrouwbaarheid comp. 2 en 3 bij k_2 par. ond. in comp. 2				
s_2	$k_2 = 1$	$k_2 = 2$	$k_2 = 3$	k_2^*	betrouwbaarheid
7	0.49	0.4	_	1	0.49

Dynamisch Programmeren: Tweede Voorbeeld (5)

component 2		
k_2	r_2	c_2
1	0.7	3
2	0.8	5
3	0.9	6

s_3	betrouwbaarheid component 3
2	0.5
3	0.5
4	0.7
5	0.9
6	0.9

Stap 2

	betrouwbaarheid comp. 2 en 3 bij k_2 par. ond. in comp. 2				
s_2	$k_2 = 1$	$k_2 = 2$	$k_2 = 3$	k_2^*	betrouwbaarheid
7	0.49	0.4	_	1	0.49
8	0.63	0.4	0.45	1	0.63

Dynamisch Programmeren: Tweede Voorbeeld (5)

component 2		
k_2	r_2	c_2
1	0.7	3
2	0.8	5
3	0.9	6

s_3	betrouwbaarheid component 3
2	0.5
3	0.5
4	0.7
5	0.9
6	0.9

Stap 2

	betrouwbaarheid comp. 2 en 3 bij k_2 par. ond. in comp. 2				
s_2	$k_2 = 1$	$k_2 = 2$	$k_2 = 3$	k_2^*	betrouwbaarheid
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

Dynamisch Programmeren: Tweede Voorbeeld (6)

component 1		
k_1	r_1	c_1
1	0.6	1
2	0.8	2
3	0.9	3

s_2	betrouwbaarheid component 2 en 3
7	0.49
8	0.63
9	0.63

Stap 1

	betrouwbaarheid comp. 1, 2 en 3 bij k_1 par. ond. in comp. 1				
s_1	$k_1 = 1$	$k_1 = 2$	$k_1 = 3$	k_1^*	betrouwbaarheid
10	0.378	0.504	0.441	**2**	**0.504**

Dynamisch Programmeren: Tweede Voorbeeld (6)

component 1		
k_1	r_1	c_1
1	0.6	1
2	0.8	2
3	0.9	3

s_2	betrouwbaarheid component 2 en 3
7	0.49
8	0.63
9	0.63

Stap 1

	betrouwbaarheid comp. 1, 2 en 3 bij k_1 par. ond. in comp. 1				
s_1	$k_1 = 1$	$k_1 = 2$	$k_1 = 3$	k_1^*	betrouwbaarheid
10	0.378	0.504	0.441	**2**	**0.504**

- de optimale betrouwbaarheid van het apparaat is 0.504
- component 1 moet twee parallelle onderdelen hebben

Dynamisch Programmeren: Tweede Voorbeeld (7)

Terugkijkend naar stappen 2 en 3 kunnen we uitvinden hoe veel parallelle onderdelen componenten 2 en 3 moeten hebben.

- Component 1 moet twee parallelle onderdelen hebben;
  - dit kost 2 euro;
  - er is 8 euro over voor componenten 2 en 3.

Dynamisch Programmeren: Tweede Voorbeeld (7)

Terugkijkend naar stappen 2 en 3 kunnen we uitvinden hoe veel parallelle onderdelen componenten 2 en 3 moeten hebben.

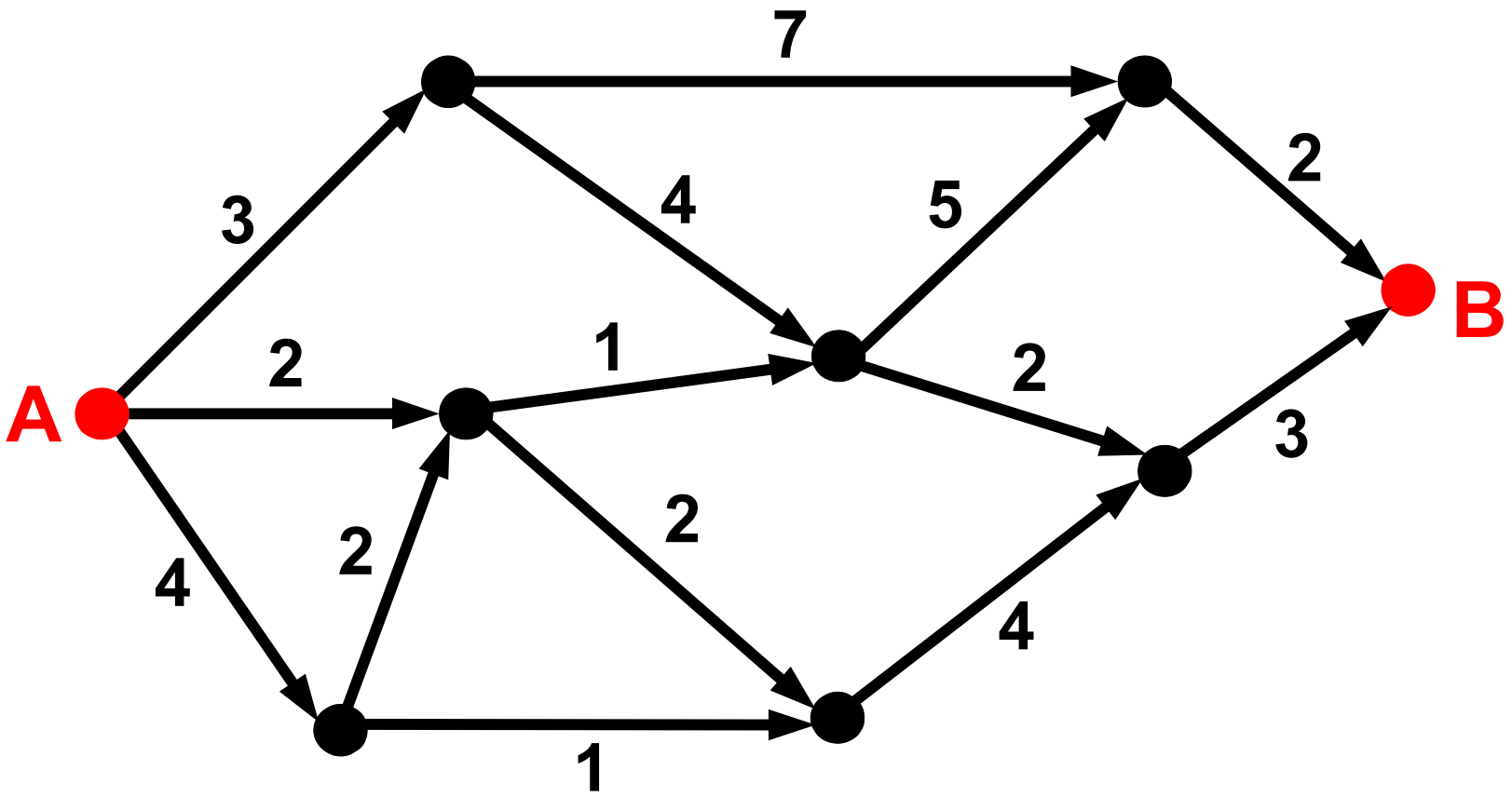
- Component 1 moet twee parallelle onderdelen hebben;
  - dit kost 2 euro;
  - er is 8 euro over voor componenten 2 en 3.
- In Stap 2 met $s_2 = 8$ zien we dat component 2 één parallel onderdeel moet hebben;
  - dit kost 3 euro;
  - er is 5 euro over voor component 3.

Dynamisch Programmeren: Tweede Voorbeeld (7)

Terugkijkend naar stappen 2 en 3 kunnen we uitvinden hoe veel parallelle onderdelen componenten 2 en 3 moeten hebben.

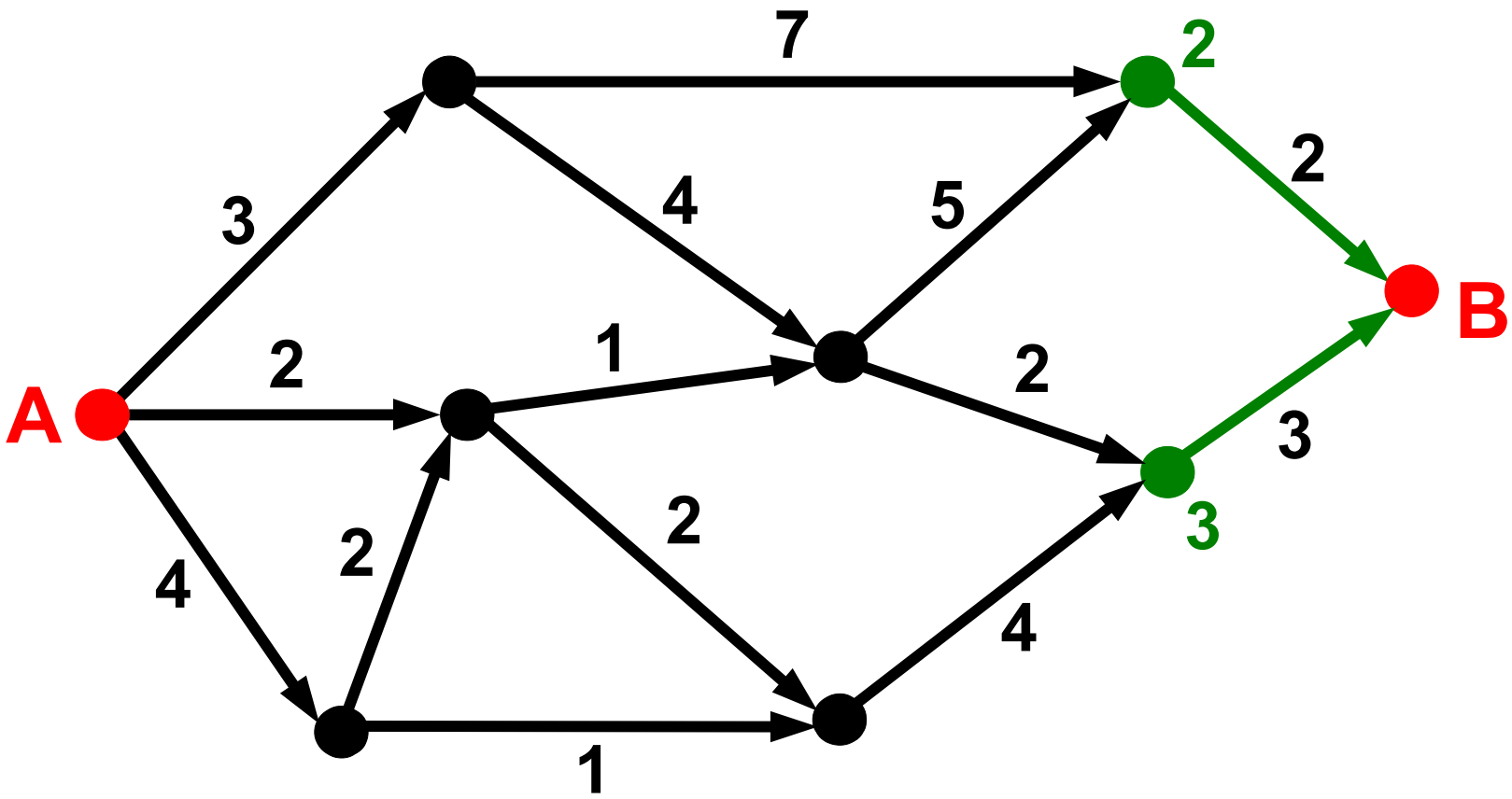
- Component 1 moet twee parallelle onderdelen hebben;
  - dit kost 2 euro;
  - er is 8 euro over voor componenten 2 en 3.
- In Stap 2 met $s_2 = 8$ zien we dat component 2 één parallel onderdeel moet hebben;
  - dit kost 3 euro;
  - er is 5 euro over voor component 3.
- In Stap 3 met $s_3 = 5$ zien we dat component 3 drie parallelle onderdelen moet hebben.

Kortste paden met dynamisch programmeren



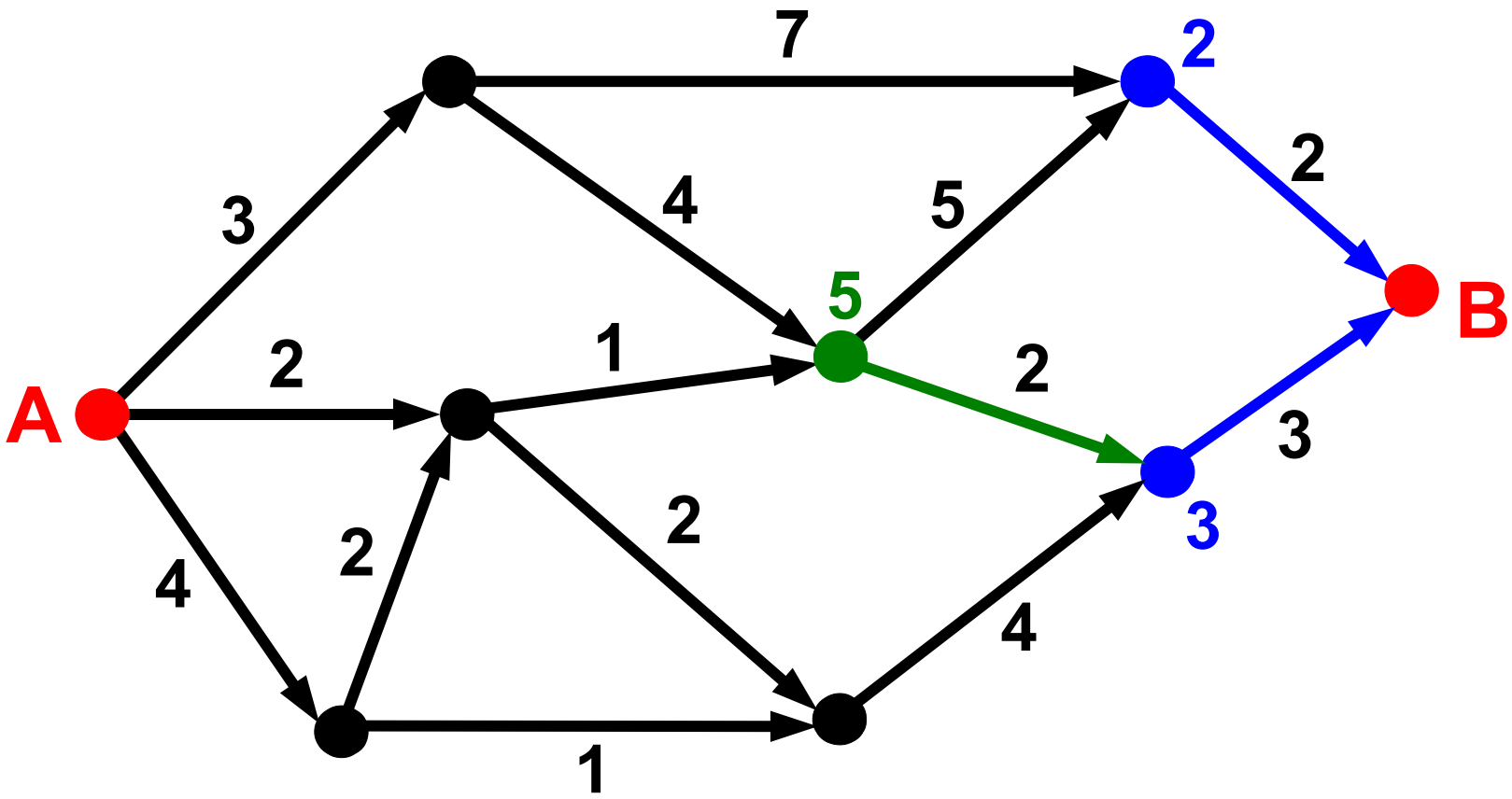
Vind het kortste pad van A naar B

Kortste paden met dynamisch programmeren



Stap 1: label punten die één stap van B vandaan liggen met de afstand naar B (over een kortste pad)

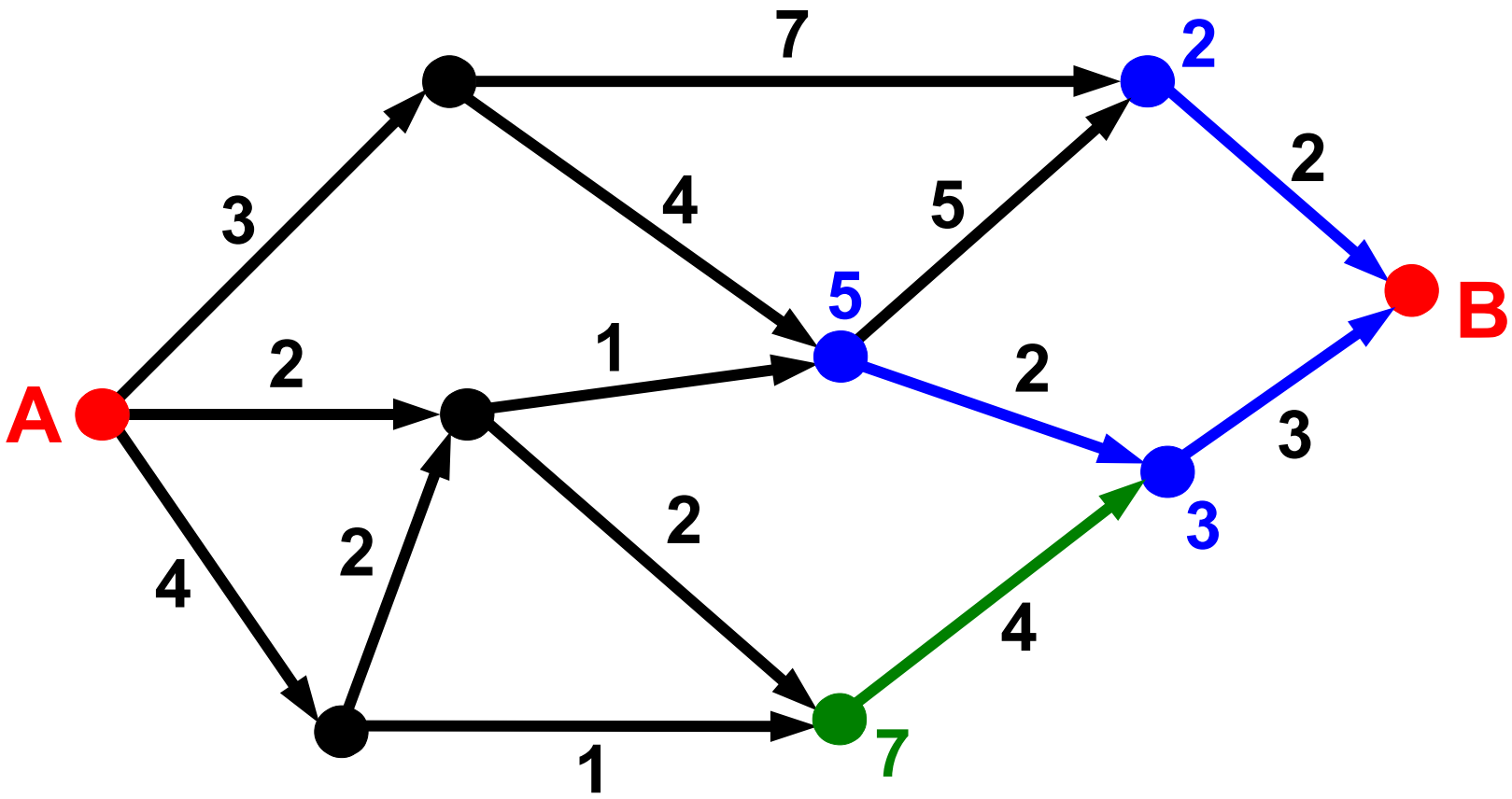
Kortste paden met dynamisch programmeren



De afstand van het groene punt naar B (over een kortste pad) is:

$$\min\{5+2, 2+3\} = 5$$

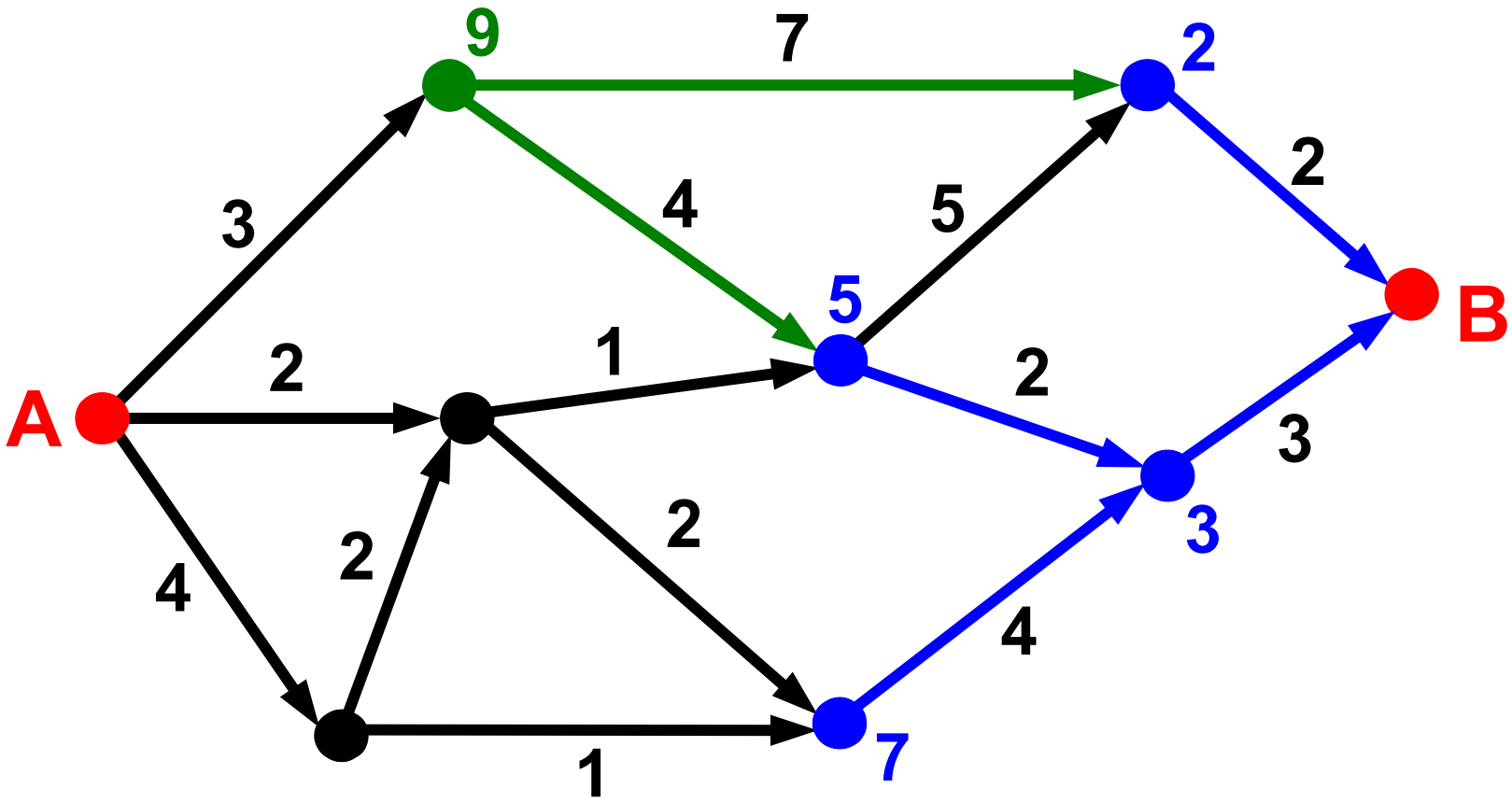
Kortste paden met dynamisch programmeren



De afstand van het groene punt naar B (over een kortste pad) is:

$$\min\{4+3\} = 7$$

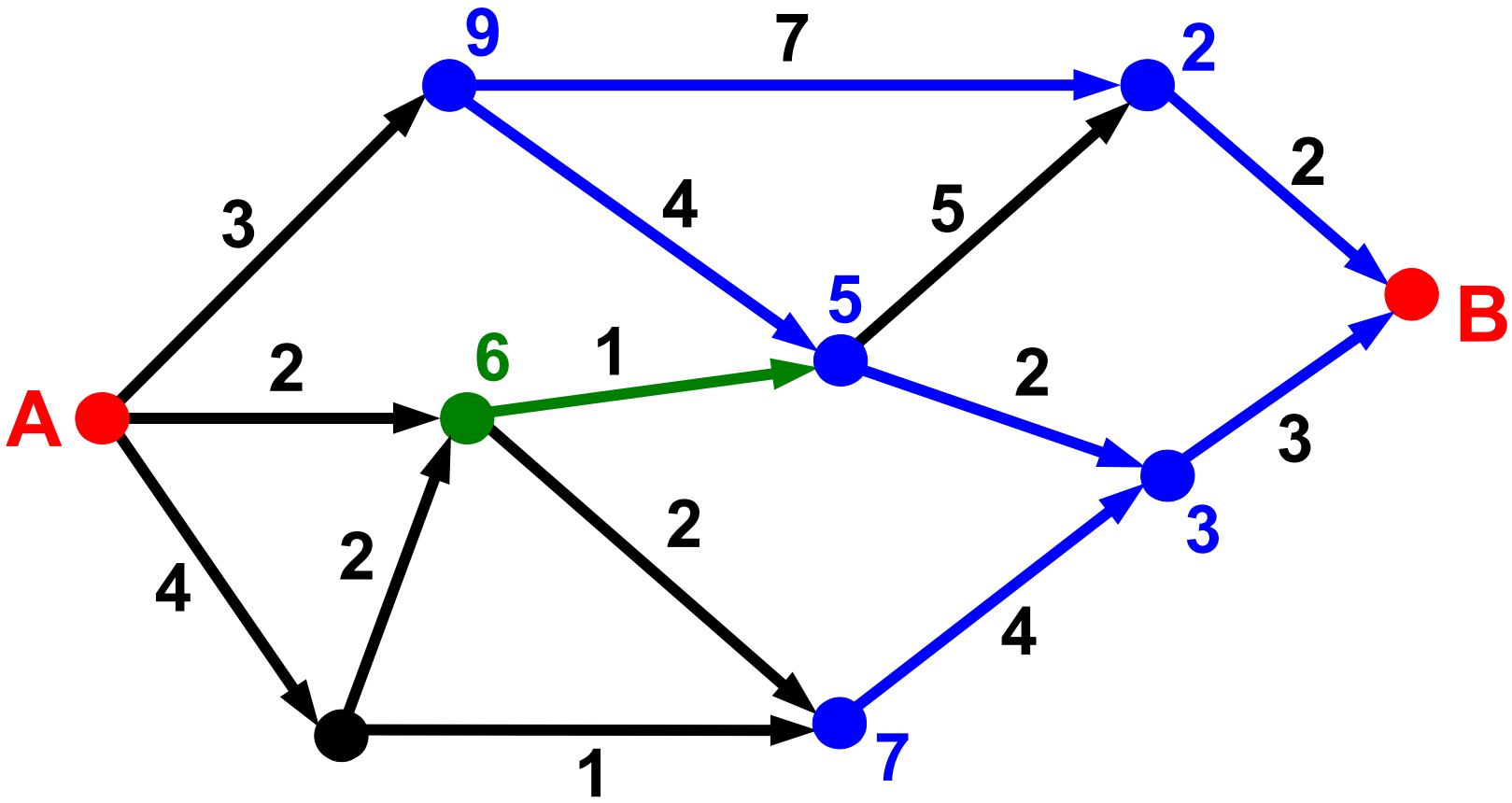
Kortste paden met dynamisch programmeren



De afstand van het groene punt naar B (over een kortste pad) is:

$$\min\{7+2, 4+5\} = 9$$

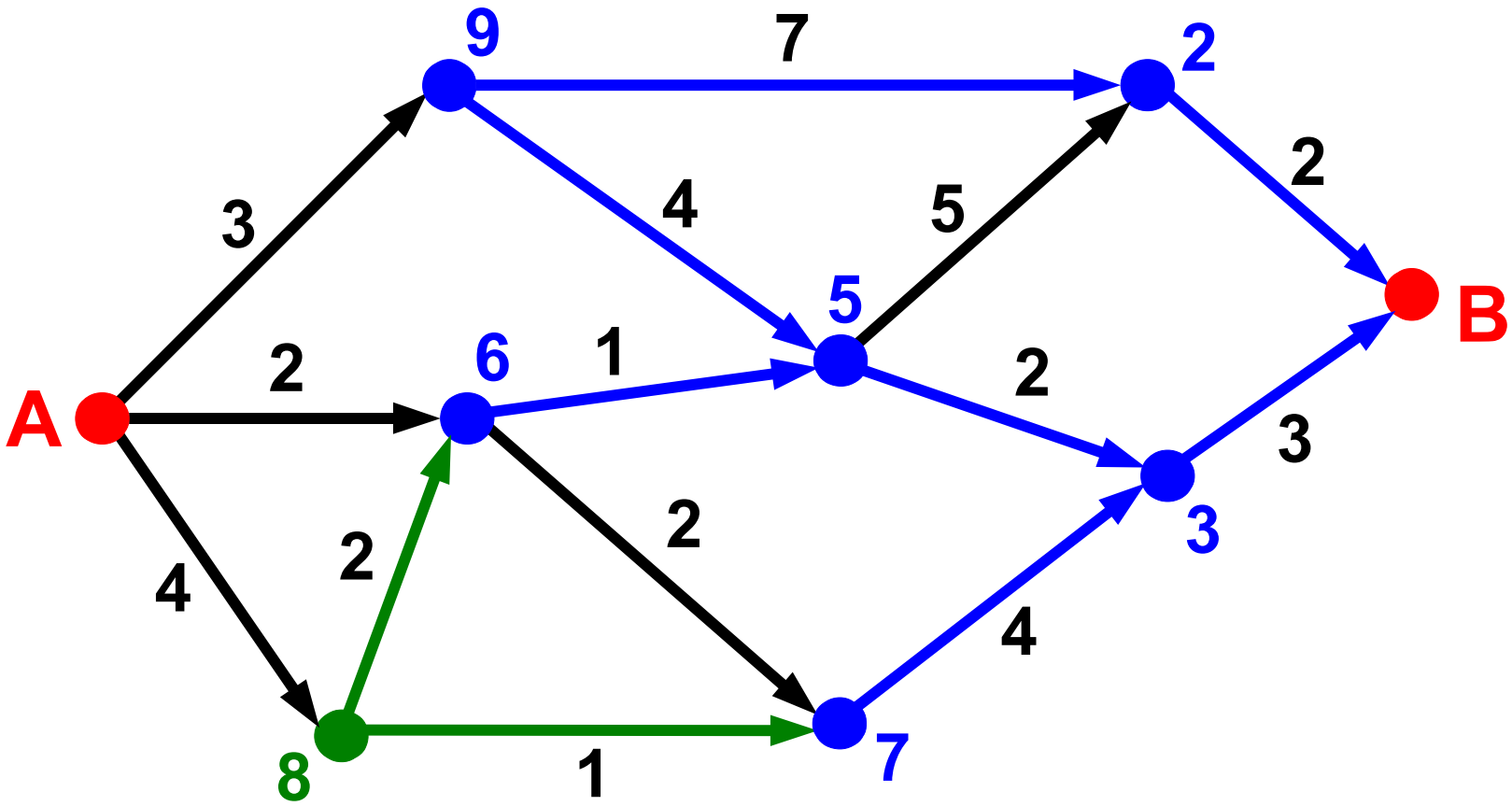
Kortste paden met dynamisch programmeren



De afstand van het groene punt naar B (over een kortste pad) is:

$$\min\{1+5, 2+7\} = 6$$

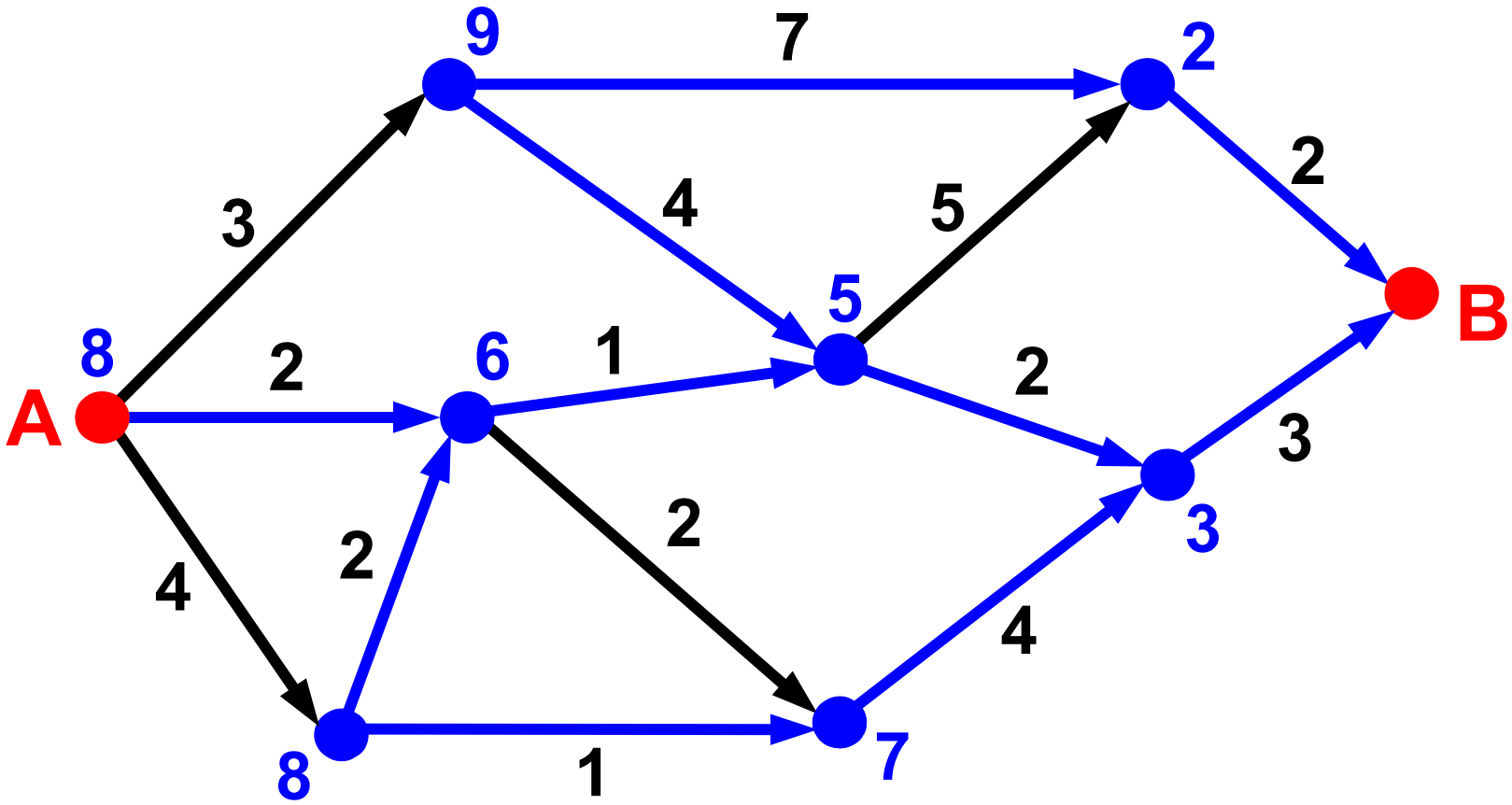
Kortste paden met dynamisch programmeren



De afstand van het groene punt naar B (over een kortste pad) is:

$$\min\{2+6, 1+7\} = 8$$

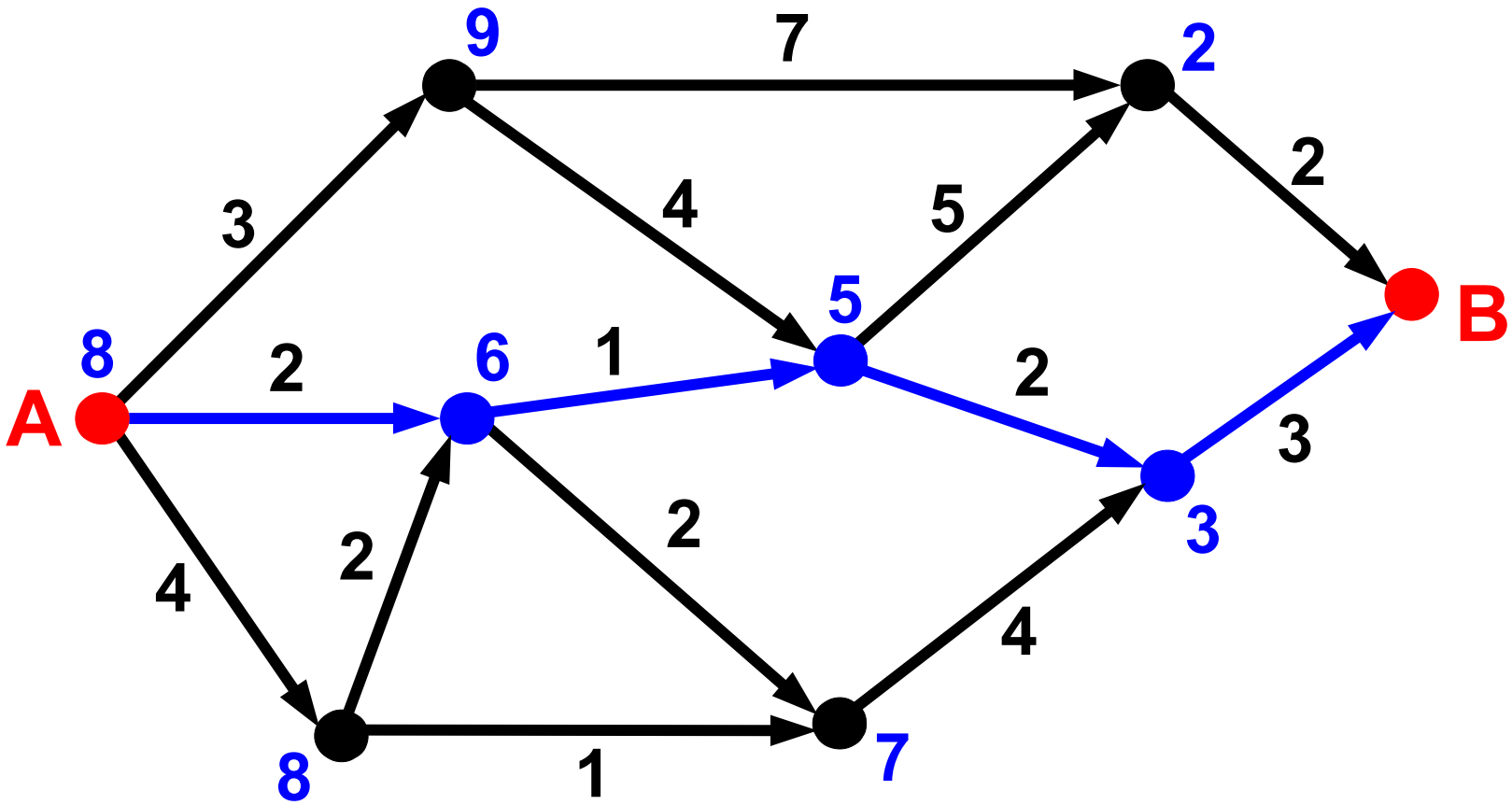
Kortste paden met dynamisch programmeren



De afstand van A naar B (over een kortste pad) is:

$$\min\{3+9, 2+6, 4+8\} = 8$$

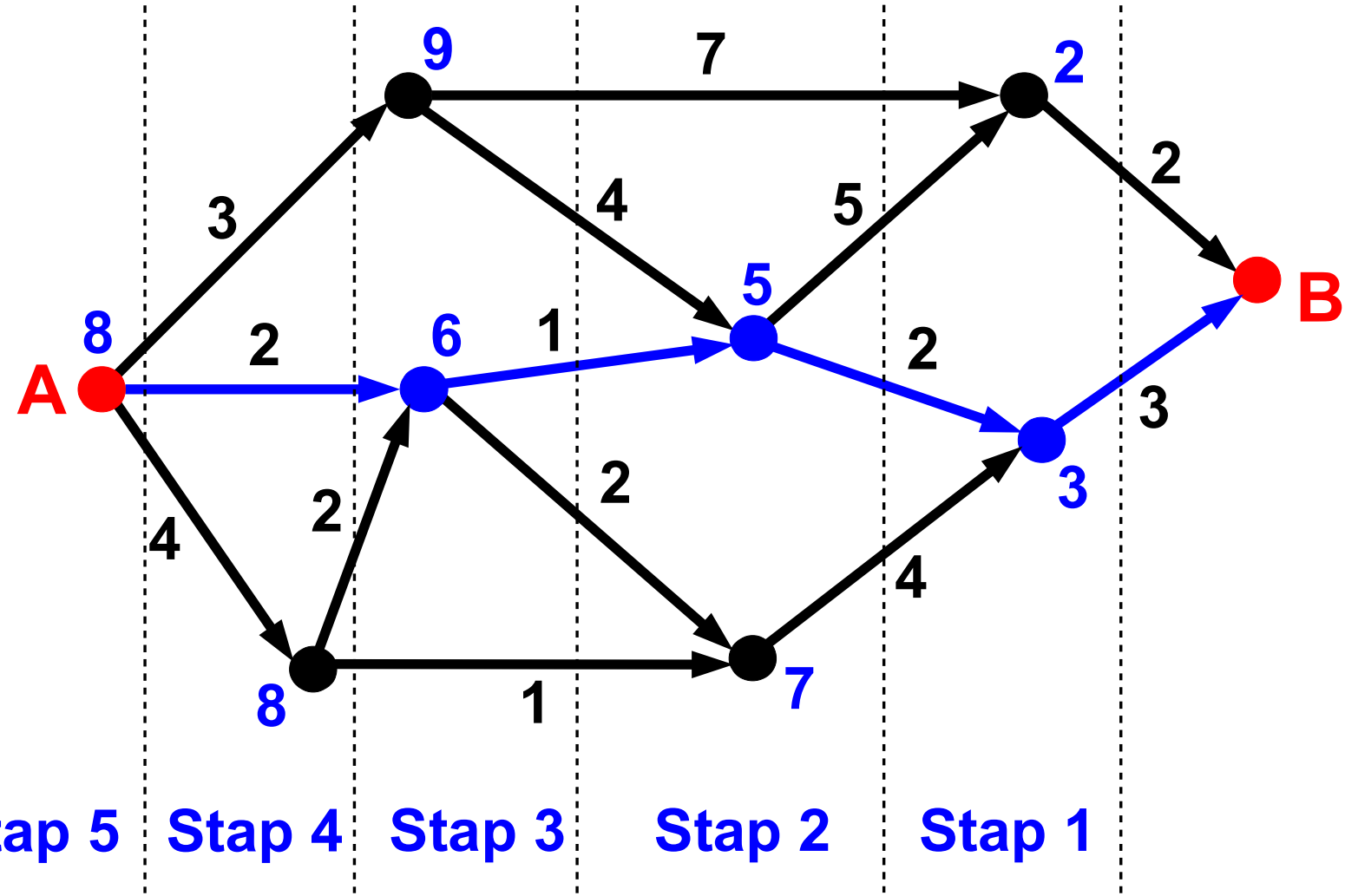
Kortste paden met dynamisch programmeren



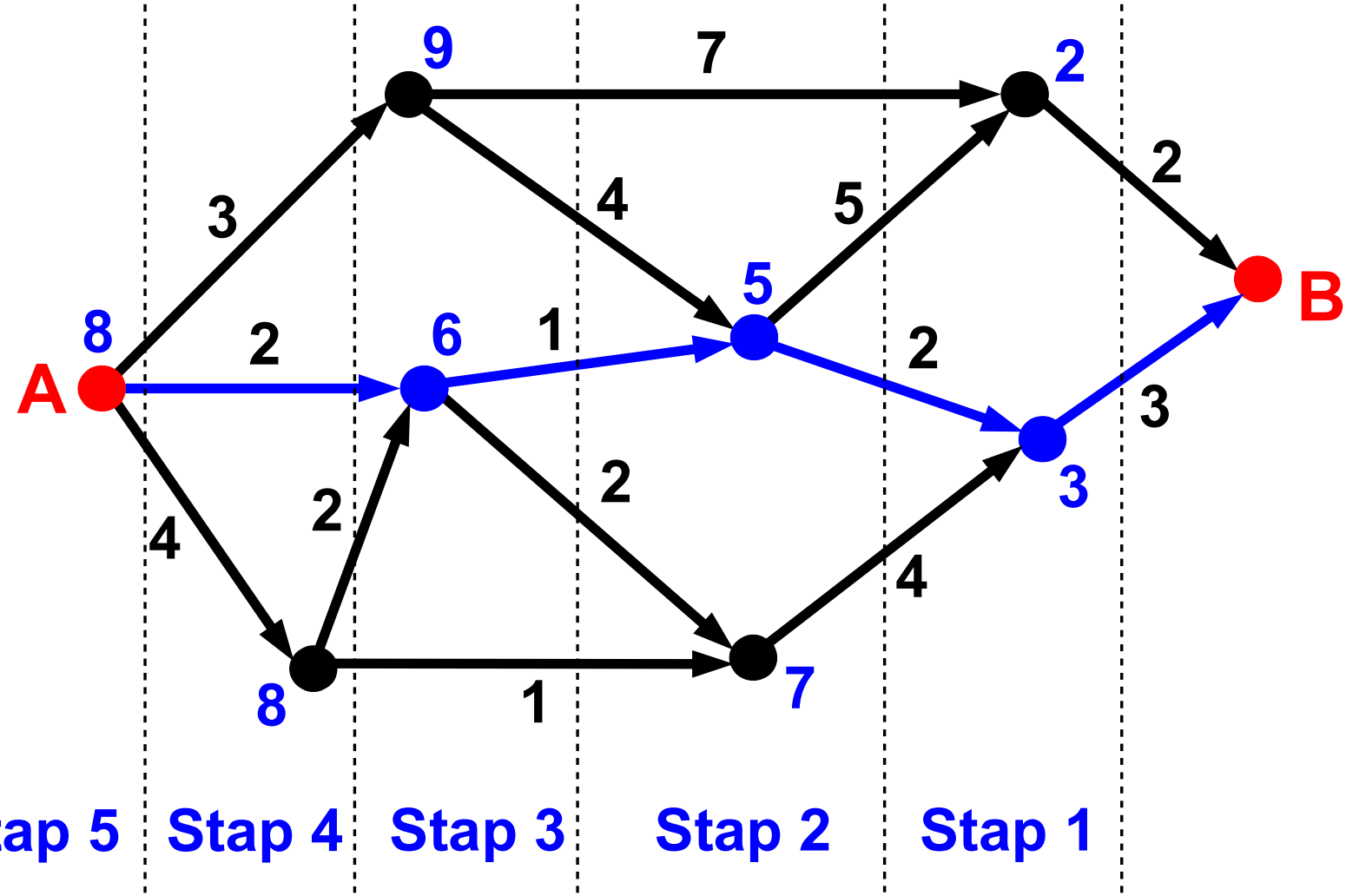
De lengte van een kortste pad van A naar B is 8. We kunnen zo'n pad vinden door bij A te beginnen en steeds een optimale (blauwe) pijl te volgen tot aan B.

Kortste paden met dynamisch programmeren

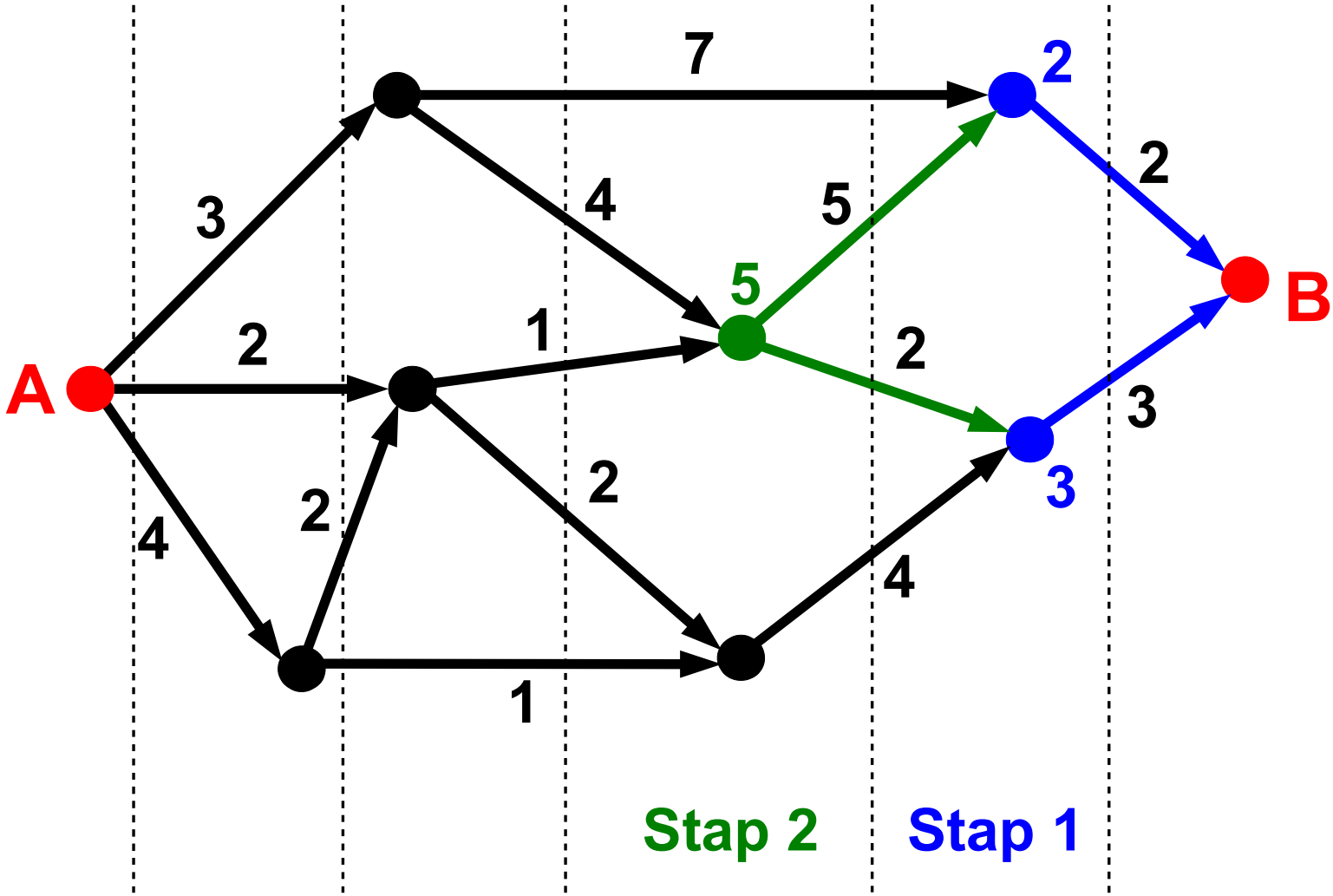
- We hebben het kortste pad van A naar B berekend door middel van dynamisch programmeren
- Deze methode werkt in een gericht acyclisch netwerk



We hebben het probleem opgedeeld in **stappen**. In elke stap zijn er een aantal **toestanden** (punten) en voor elke toestand zijn er verschillende **keuzemogelijkheden** (uitgaande pijlen)



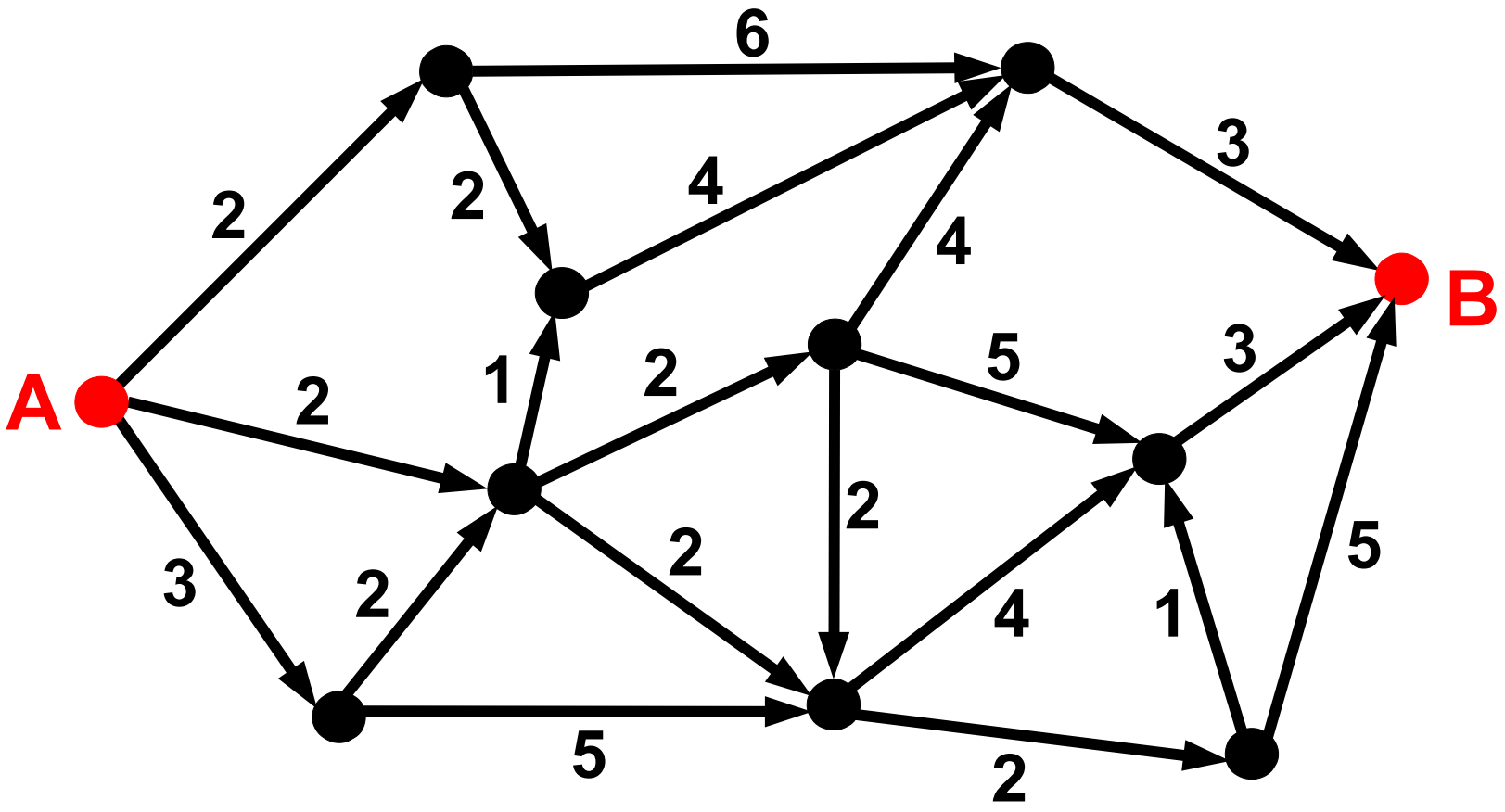
Voor elke stap en elke toestand hebben we de optimale keuzemogelijkheden bepaald en de optimale waarde (afstand)



De optimale oplossing in stap i kunnen we berekenen uit de optimale oplossing uit stap $i-1$. Bijvoorbeeld:

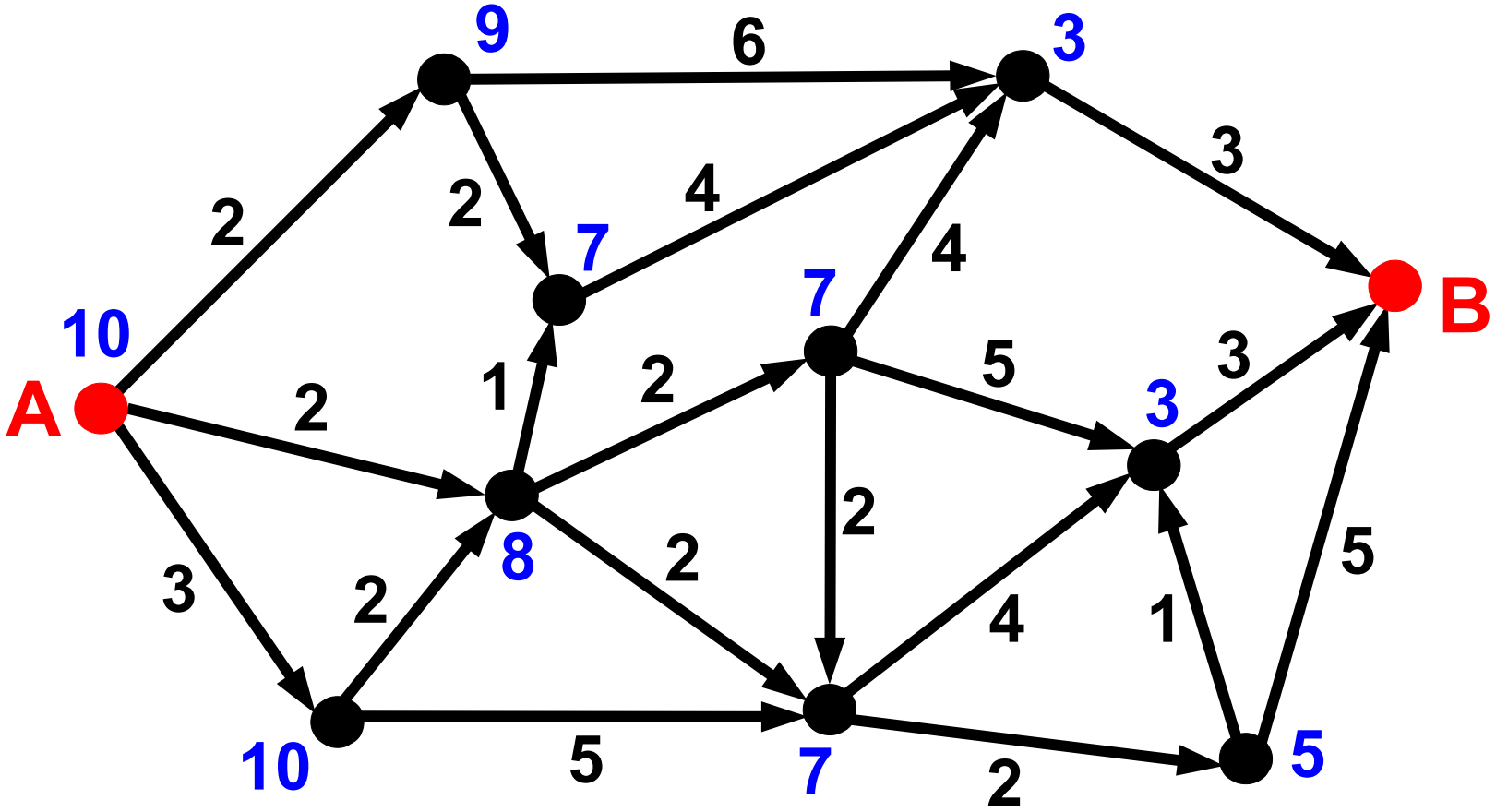
$$\min\{5+2, 2+3\} = 5$$

Oefening



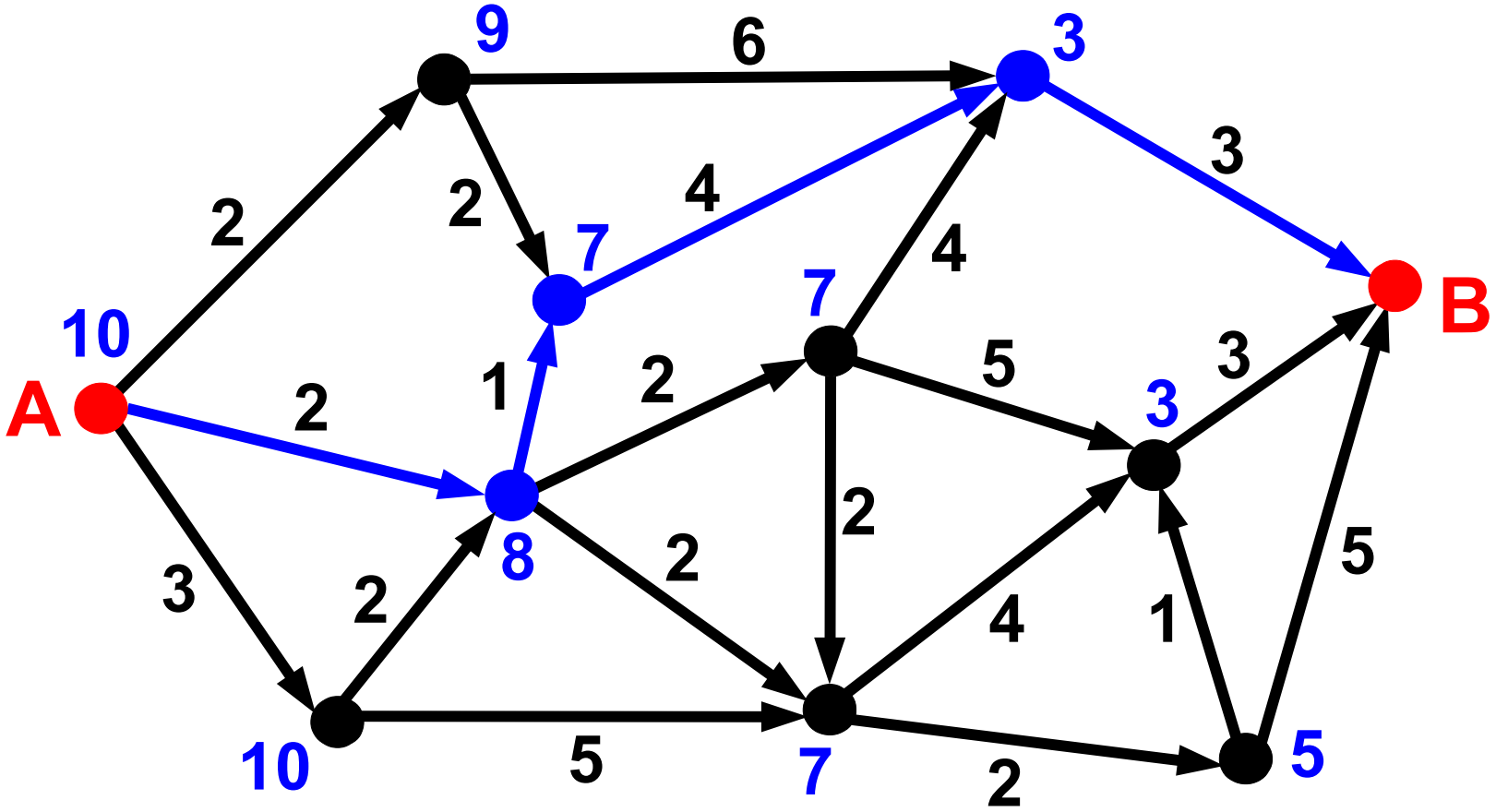
Vind een kortste pad van A naar B

Oplossing (afstanden)



De afstand van A naar B over een kortste pad is 10

Oplossing (kortste pad)



Zijn er nog andere mogelijkheden?

Als we deze notatie gebruiken:

- $d(u, v)$: de afstand van u naar v (via een kortste pad)
- $\delta^+(u)$: de verzameling punten die uit u te bereiken zijn (via één pijl)
- K: de verzameling punten waarvoor de afstand tot B al bekend is

Als we deze notatie gebruiken:

- $d(u, v)$: de afstand van u naar v (via een kortste pad)
- $\delta^+(u)$: de verzameling punten die uit u te bereiken zijn (via één pijl)
- K: de verzameling punten waarvoor de afstand tot B al bekend is

Dan kunnen we de methode zo schrijven:

1. Zet $d(B, B) := 0$
2. Zet $K := \{B\}$

Als we deze notatie gebruiken:

- $d(u, v)$: de afstand van u naar v (via een kortste pad)
- $\delta^+(u)$: de verzameling punten die uit u te bereiken zijn (via één pijl)
- K: de verzameling punten waarvoor de afstand tot B al bekend is

Dan kunnen we de methode zo schrijven:

1. Zet $d(B, B) := 0$
2. Zet $K := \{B\}$
3. Voor elk punt u met $\delta^+(u) \subseteq K$, zet

$$d(u, B) := \min\{d(u, v) + d(v, B) \mid v \in \delta^+(u)\}$$

Als we deze notatie gebruiken:

- $d(u, v)$: de afstand van u naar v (via een kortste pad)
- $\delta^+(u)$: de verzameling punten die uit u te bereiken zijn (via één pijl)
- K: de verzameling punten waarvoor de afstand tot B al bekend is

Dan kunnen we de methode zo schrijven:

1. Zet $d(B, B) := 0$
2. Zet $K := \{B\}$
3. Voor elk punt u met $\delta^+(u) \subseteq K$, zet
$$d(u, B) := \min\{d(u, v) + d(v, B) \mid v \in \delta^+(u)\}$$
4. Voeg u toe aan K

Als we deze notatie gebruiken:

- $d(u, v)$: de afstand van u naar v (via een kortste pad)
- $\delta^+(u)$: de verzameling punten die uit u te bereiken zijn (via één pijl)
- K: de verzameling punten waarvoor de afstand tot B al bekend is

Dan kunnen we de methode zo schrijven:

1. Zet $d(B, B) := 0$
2. Zet $K := \{B\}$
3. Voor elk punt u met $\delta^+(u) \subseteq K$, zet
$$d(u, B) := \min\{d(u, v) + d(v, B) \mid v \in \delta^+(u)\}$$
4. Voeg u toe aan K
5. Herhaal stappen 3 en 4 totdat A in K zit

Alle afstanden zijn nu bekend. We kunnen een kortste pad als volgt vinden:

1. Zet $u := A$
2. Ga van u naar een punt v waarvoor

$$d(u, B) = d(u, v) + d(v, B)$$

3. Zet $u := v$
4. Herhaal stappen 2 en 3 totdat B bereikt is

Samenvatting Dynamisch Programmeren

- Splits het probleem op in stappen
- In elke stap zijn er keuzemogelijkheden en toestanden
- Bereken voor elke stap, voor elke toestand en voor elke keuzemogelijkheid de waarde van de optimale oplossing
- Je kunt voorwaartse of terugwaartse recursie gebruiken
  - Bij voorwaartse recursie berekenen we de oplossingen in stap i uit de oplosingen van stap $i-1$
  - Bij terugwaartse recursie berekenen we de oplossingen in stap i uit de oplosingen van stap $i+1$

Samenvatting van vandaag

- Dynamische programmering
- Principe van optimaliteit
- stappen, keuzemogelijkheden en toestanden
- voorwaartse en terugwaartse recursie