Elfde college algoritmiek

30 april 2025

Gretige Algoritmen,
Algoritme van Dijkstra

- nakijken opdracht 1
- deadline opdracht 2

Goedkoop Tanken

```
60
4
10 10 15
15 100 20
40 30 18
80 35 25
125
```

Goedkoop Tanken

```
60
4
10  10  15
15  100  20
40  30  18
80  35  25
125
```

Minimale kosten = 2315
Tank: 10, 50, 30, 25

Goedkoop Tanken

```
60
4
10 10 15
15 100 20
40 30 18
80 35 25
125
```

Minimale kosten = 2315
Tank: 10, 50, 30, 25

Dynamisch programmeren

`kosten[i][j] = ...`

`kosten[3][25] = ...` `kosten[3][10] = ...`

Goedkoop Tanken

```
60
4
10 10 15
15 100 20
40 30 18
80 35 25
125
```

Met twee jokers:
Minimale kosten = 1383
Tank: 5, 60 (joker), 15, 35 (joker)

Dynamisch programmeren

`kosten[i][j][k] = ...`

Gegeven een verzameling $A = \{1, 2, \ldots, n\}$ met activiteiten die allemaal gebruik willen maken van een of andere "resource" (bijvoorbeeld een collegezaal). Deze resource kan maar door één activiteit tegelijk gebruikt worden. Activiteit i vindt plaats gedurende het tijdsinterval $[b_i, e_i)$. De begin- en eindtijden b_i en e_i zijn voor alle activiteiten bekend.

Definitie: activiteiten i en j heten compatibel als $[b_i, e_i)$ en $[b_j, e_j)$ niet overlappen, dus als $e_i \leq b_j$ of $e_j \leq b_i$.

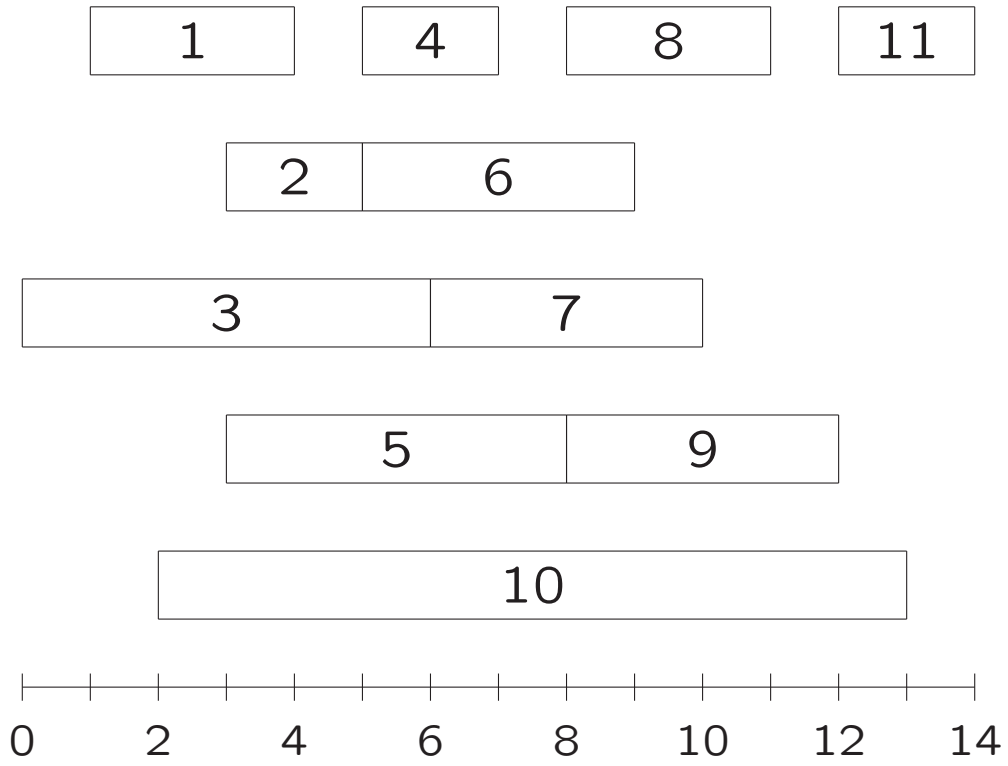
Opdracht: vind een zo groot mogelijke deelverzameling van paarsgewijs compatibele activiteiten uit A.

Alle deelverzamelingen aflopen...

i	b_i	e_i
1	1	4
2	3	5
3	0	6
4	5	7
5	3	8
6	5	9
7	6	10
8	8	11
9	8	12
10	2	13
11	12	14

Optimale oplossing: . . .

i	b_i	e_i
1	1	4
2	3	5
3	0	6
4	5	7
5	3	8
6	5	9
7	6	10
8	8	11
9	8	12
10	2	13
11	12	14



Optimale oplossing: . . .

Gretig algoritme: in elke stap

- wordt een activiteit i gekozen die compatibel is met de reeds gekozen activiteiten en die op dat moment het beste lijkt (gretige keuze)

Mogelijke gretige keuzes voor het kiezen van activiteit i: . . .

Gretig algoritme: in elke stap

- wordt een activiteit i gekozen die compatibel is met de reeds gekozen activiteiten en die op dat moment het beste lijkt (gretige keuze)

Mogelijke gretige keuzes voor het kiezen van activiteit i:

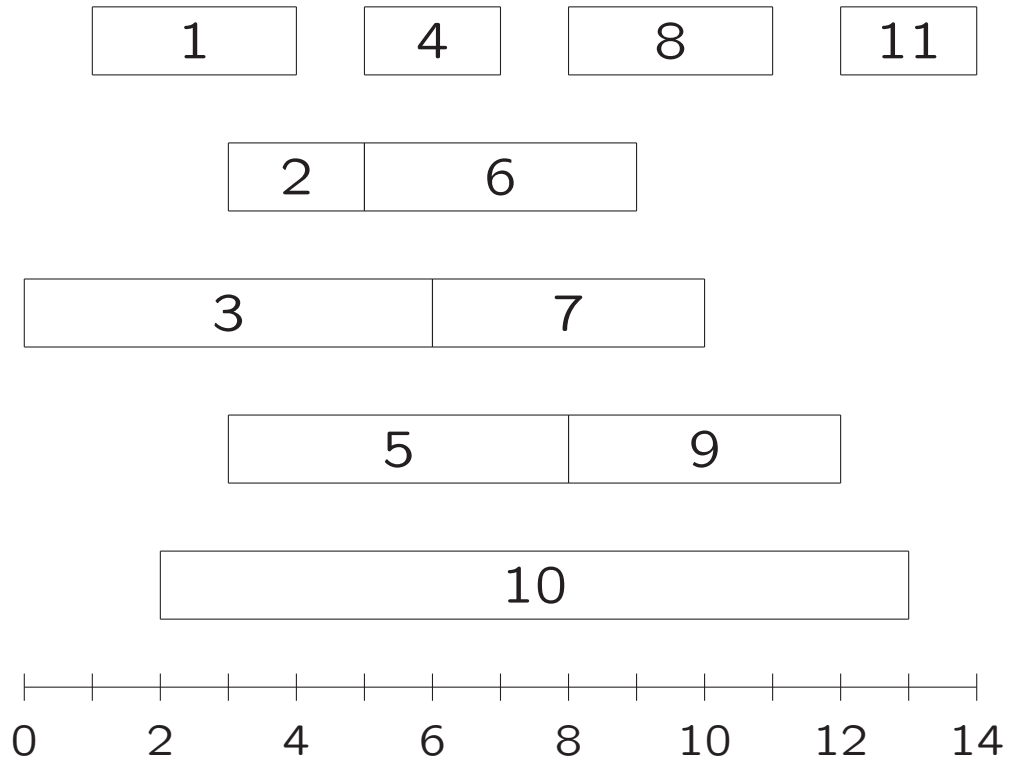
1. selecteer steeds de activiteit die het eerst begint
2. selecteer steeds de activiteit die het kortste duurt
3. selecteer steeds de activiteit die overlapt met zo min mogelijk andere activiteiten
4. selecteer steeds de activiteit die het eerst eindigt

Welke gretige keuzes voor het kiezen van activiteit i leiden altijd tot een optimale oplossing?

1. selecteer de activiteit die het eerst begint: werkt niet
2. selecteer de activiteit die het kortste duurt: werkt niet
3. selecteer de activiteit die overlapt met zo min mogelijk andere activiteiten: werkt niet
4. selecteer de activiteit die het eerst eindigt: werkt wel

```
// neem aan dat A oplopend gesorteerd is op eindtijd e_i,
// anders eerst even sorteren: O(n lg n)
A' := {1};
j := 1;
// de laatst aan A' toegevoegde activiteit
for i := 2 to n do
// loop activiteiten af in volgorde van eindtijd
    if i is compatibel met A' then (*)
        A' := A' ⋃ {i}; j := i;
    fi
od
// A' bevat nu een optimale paarsgewijs
// compatibele deelverzameling van A
```

i	b_i	e_i
1	1	4
2	3	5
3	0	6
4	5	7
5	3	8
6	5	9
7	6	10
8	8	11
9	8	12
10	2	13
11	12	14



Optimale oplossing: $\{1, 4, 8, 11\}$

```
// neem aan dat A oplopend gesorteerd is op eindtijd e_i,
// anders eerst even sorteren: O(n lg n)
A' := {1};
j := 1;
// de laatst aan A' toegevoegde activiteit
for i := 2 to n do
// loop activiteiten af in volgorde van eindtijd
    if i is compatibel met A' then (*)
        A' := A' ⋃ {i}; j := i;
    fi
od
// A' bevat nu een optimale paarsgewijs
// compatibele deelverzameling van A
```

Correctheid: . . .

De **correctheid** van het gretige algoritme volgt uit de volgende twee observaties:

1. Er bestaat een optimale oplossing die met activiteit 1 begint (die met de kleinste eindtijd dus).

2. . . .

De **correctheid** van het gretige algoritme volgt uit de volgende twee observaties:

1. Er bestaat een optimale oplossing die met activiteit 1 begint (die met de kleinste eindtijd dus).

2. Stel A' is een optimale oplossing van het oorspronkelijke probleem, dus met activiteitenverzameling A, die 1 bevat. Dan is $B' = A' \setminus \{1\}$ een optimale oplossing van het probleem met activiteitenverzameling $B = \{i \in A : b_i \geq e_1\}$.

```
// neem aan dat A oplopend gesorteerd is op eindtijd e_i,
// anders eerst even sorteren: O(n lg n)
A' := {1};
j := 1;
// de laatst aan A' toegevoegde activiteit
for i := 2 to n do
// loop activiteiten af in volgorde van eindtijd
    if i is compatibel met A' then (*)
        A' := A' ⋃ {i}; j := i;
    fi
od
// A' bevat nu een optimale paarsgewijs
// compatibele deelverzameling van A
```

(*) Merk op: i is compatibel met A' als hij compatibel is met de laatst toegevoegde activiteit.

Dus (*) wordt: `if` $b_i \geq e_j$ `then`

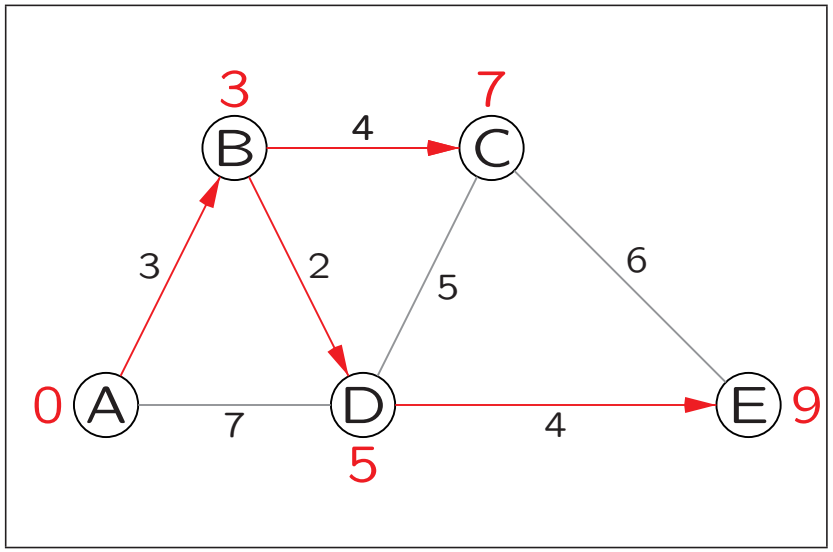
Correctheid: OK (gezien)

Complexiteit: $O(n)$ als A reeds gesorteerd is

Gegeven een graaf G met gewichten op de takken, en een beginknoop s. We nemen aan dat alle gewichten ≥ 0 zijn.

Gevraagd: voor elke willekeurige knoop v in de graaf (de lengte van) het/een kortste pad van s naar v.

Merk op dat al deze kortste paden vanuit s samen een boomstructuur vormen.



De kortste paden vanuit A zijn:

A $\longrightarrow$ B: lengte 3

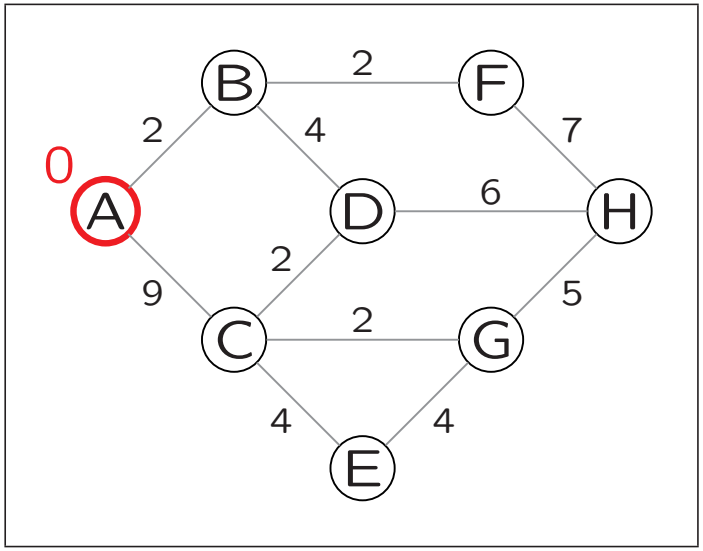
A $\longrightarrow$ B $\longrightarrow$ D: lengte 5

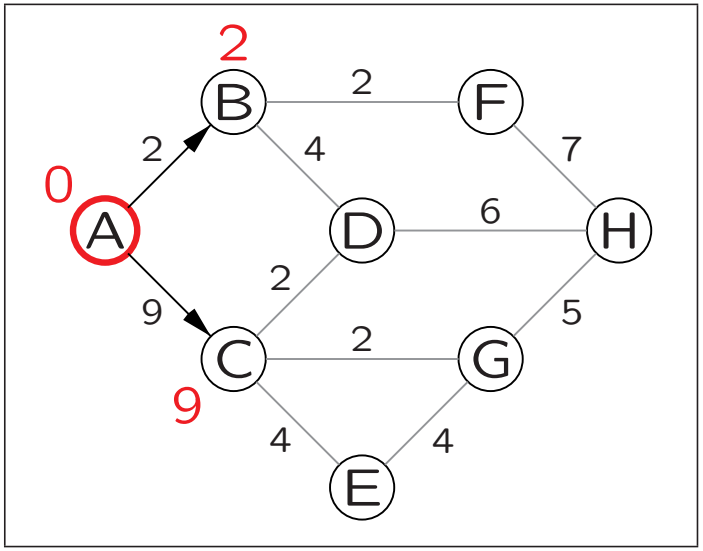
A $\longrightarrow$ B $\longrightarrow$ C: lengte 7

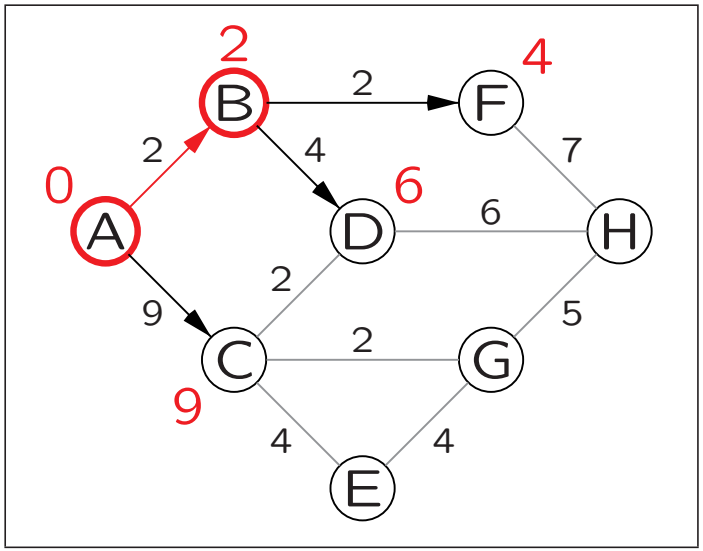
A $\longrightarrow$ B $\longrightarrow$ D $\longrightarrow$ E: lengte 9

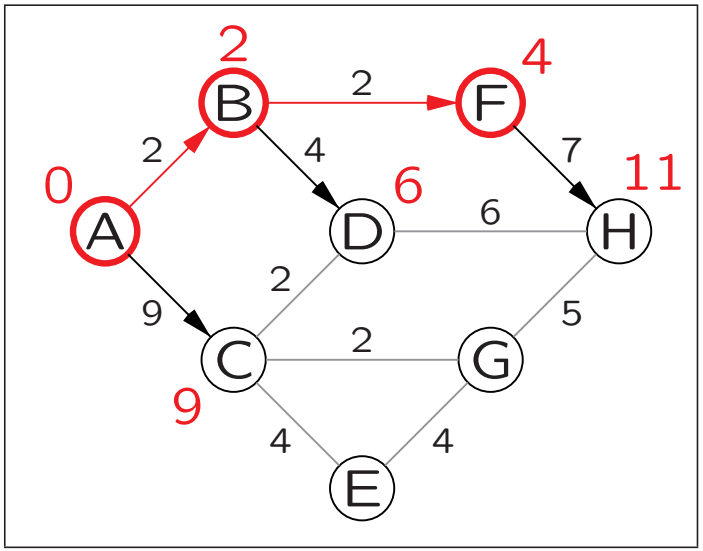
Oplossing: het algoritme van Dijkstra is een gretig algoritme, dat de kortste paden van s naar elk van de andere knopen vindt in volgorde van hun lengte. In elke stap wordt een knoop (en daarmee het pad van s naar die knoop) gekozen waarvoor het tot nu toe bekende pad vanaf s zo kort mogelijk is.

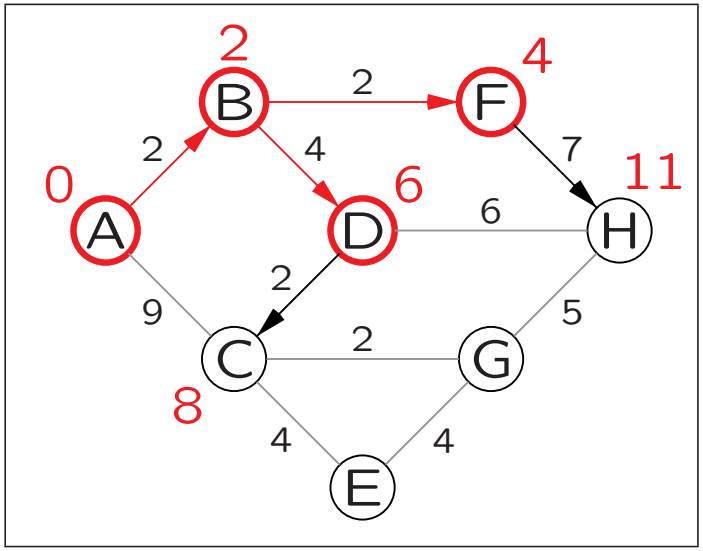
Edsger Wybe Dijkstra (Rotterdam, 11 mei 1930 — Nuenen, 6 augustus 2002) was een Nederlandse wiskundige en informaticus die veel voor de informatica heeft gedaan, met name op het gebied van gestructureerd programmeren. In 1972 werd hij onderscheiden met de Turing Award.

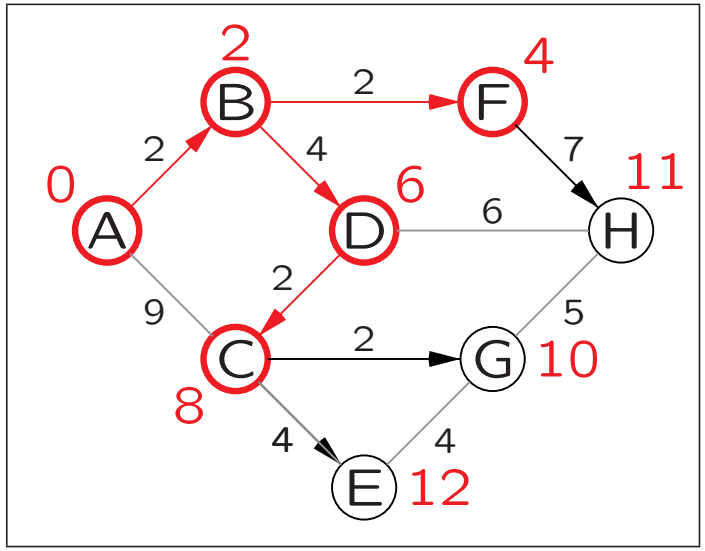

0
A
B
C
D
E
F
G
H
2
2
9
4
2
2
4
6
7
2
5
4

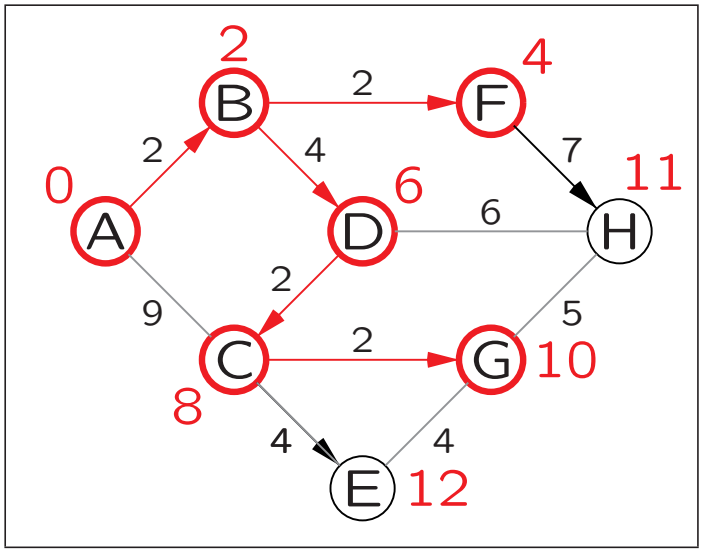

0
2
9
A
B
C
D
E
F
G
H
2
2
4
6
7
2
5
4
2
4

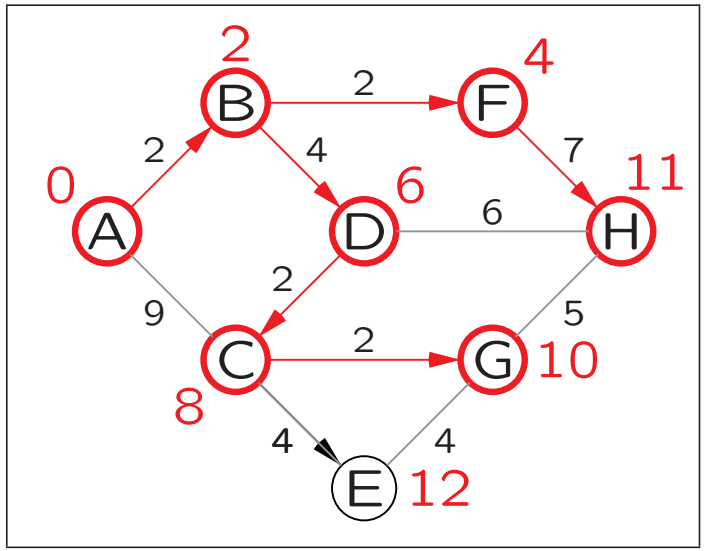

0
2
6
9
A
B
C
D
E
F
G
H
2
2
4
6
7
2
5
4
2
4
4

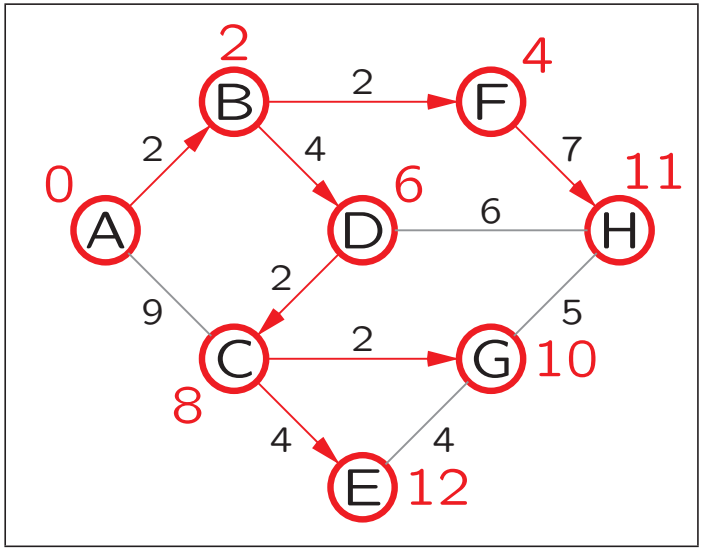

0
2
6
9
11
A
B
C
D
E
F
G
H
2
2
4
6
7
2
5
4
2
4
4


0
2
4
6
11
8
A
B
F
D
H
C
G
E
2
2
4
7
2
6
9
2
5
4
4

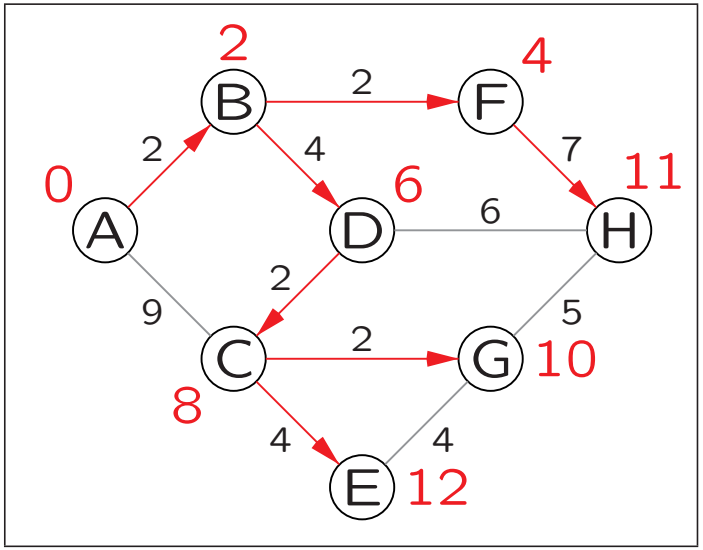

0
2
4
6
11
10
8
12
A
B
F
D
H
C
G
E
2
2
4
7
2
6
9
2
5
2
4
4


0
2
4
6
11
10
8
12
A
B
F
D
H
C
G
E
2
2
4
7
2
6
9
2
5
2
4
4


0
2
4
6
11
10
8
12
A
B
F
D
H
C
G
E
2
2
4
7
2
6
9
2
5
2
4
4



Het algoritme is klaar: alle knopen gehad



Alle kortste paden vanuit A met hun lengtes

Als je gewoon wilt doortekenen in één plaatje. . .

Begin met A, afstand 0.

A $\rightarrow$ B: $0 + 2 = 2$. OK.

A $\rightarrow$ C: $0 + 9 = 9$. OK.

Kies B, afstand 2, vanaf A.

B $\rightarrow$ D: $2 + 4 = 6$. OK.

B $\rightarrow$ F: $2 + 2 = 4$. OK.

Kies F, afstand 4, vanaf B.

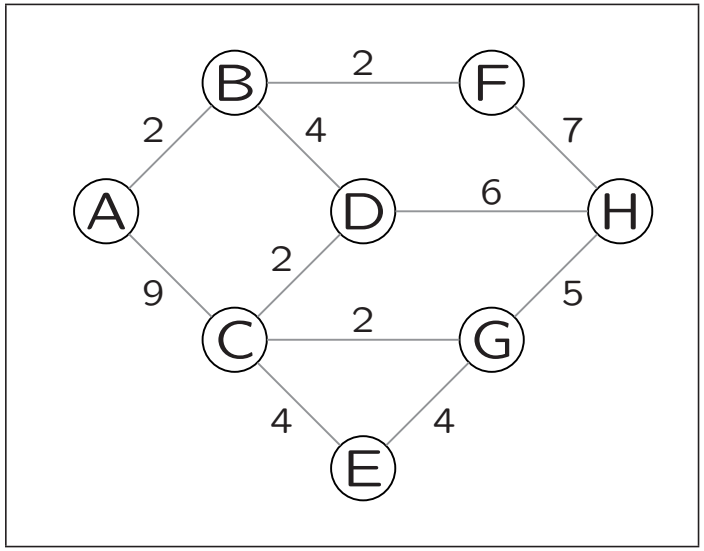
F $\rightarrow$ H: $4 + 7 = 11$. OK.

Kies D, afstand 6, vanaf B.

D $\rightarrow$ C: $6 + 2 = 8 < 9$. OK.

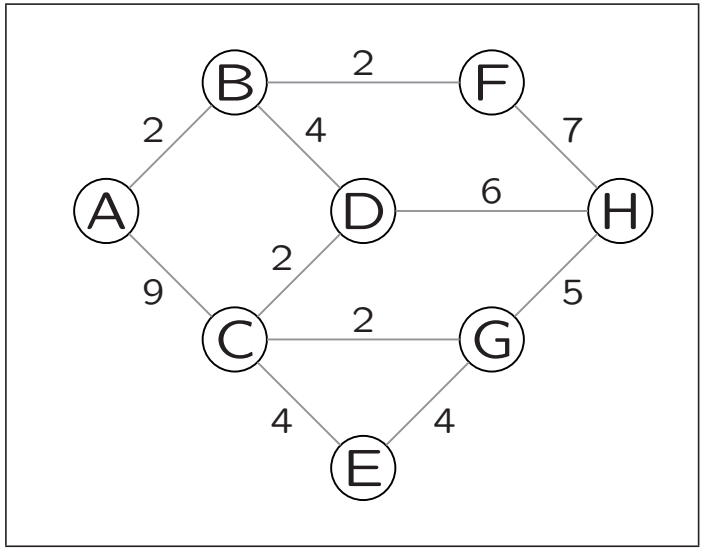
D $\rightarrow$ H: $6 + 6 = 12 > 11$. X.

Enzovoort.



Als je gewoon wilt doortekenen in één plaatje. . . (alternatief)

A	B	C	D	E	F	G	H	Actie
0	∞	∞	∞	∞	∞	∞	∞	Begin met A
–	2	9	∞	∞	∞	∞	∞	Kies B, vanaf A
–	–	9	6	∞	4	∞	∞	Kies F, vanaf B
–	–	9	6	∞	–	∞	11	Kies D, vanaf B
–	–	8	–	∞	–	∞	11	Kies C, vanaf D
–	–	–	–	12	–	10	11	Kies G, vanaf C
–	–	–	–	12	–	–	11	Kies H, vanaf F
–	–	–	–	12	–	–	–	Kies E, vanaf C



Een rij in de tabel komt overeen met het array `pad` in pseudo-code verderop.

- gewichten ≥ 0
- kortste pad $A - B$ vs alle kortste paden vanuit A

// invoer: samenhangende gewogen graaf $G = (V, E)$ en startknoop s
// uitvoer: array dat de lengtes van de kortste paden vanuit s bevat;
// na afloop is pad[v] = de lengte van een kortste pad van s naar v

for $v \in V$ **do**
    pad[v] := ∞;
od
pad[s] := 0;
$U := \emptyset$;

while ($U \neq V$) **do**
    vind knoop $v^* \in V \setminus U$ met pad[v^*] minimaal;
    $U := U \cup \{v^*\}$;
    for alle knopen v aangrenzend aan v^* **do**
        if pad[v^*] + gewicht(v^*, v) < pad[v] **then**
            pad[v] := pad[v^*] + gewicht(v^*, v);
        fi
    od
od

Complexiteit (met adjacency matrix): . . .

// invoer: samenhangende gewogen graaf $G = (V, E)$ en startknoop s
// uitvoer: array dat de lengtes van de kortste paden vanuit s bevat;
// na afloop is pad[v] = de lengte van een kortste pad van s naar v

for $v \in V$ **do**
    pad[v] := ∞;
od
pad[s] := 0;
$U := \emptyset$;

while ($U \neq V$) **do**
    vind knoop $v^* \in V \setminus U$ met pad[v^*] minimaal;
    $U := U \cup \{v^*\}$;
    for alle knopen v aangrenzend aan v^* **do**
        if pad[v^*] + gewicht(v^*, v) < pad[v] **then**
            pad[v] := pad[v^*] + gewicht(v^*, v);
        fi
    od
od

Complexiteit (met adjacency matrix):
$\Theta(n + 1 + n(n + 1 + n)) = \Theta(n^2)$

// invoer: samenhangende gewogen graaf $G = (V, E)$ en startknoop s
// uitvoer: array dat de lengtes van de kortste paden vanuit s bevat;
// na afloop is pad[v] = de lengte van een kortste pad van s naar v

for $v \in V$ **do**
    pad[v] := ∞;
od
pad[s] := 0;
$U := \emptyset$;

while ($U \neq V$) **do**
    vind knoop $v^* \in V \setminus U$ met pad[v^*] minimaal;
    $U := U \cup \{v^*\}$;
    for alle knopen v aangrenzend aan v^* **do**
        if pad[v^*] + gewicht(v^*, v) < pad[v] **then**
            pad[v] := pad[v^*] + gewicht(v^*, v);
        fi
    od
od

Complexiteit (met adjacency list en priority queue): . . .

Bij Dijkstra:
kies knoop v^* buiten boom met laagste kandidaatwaarde pad[v^*]

Bij Branch & Bound (volgende week):
kies deeloplossing met beste ondergrens/bovengrens

Priority Queue:

- objecten met prioriteit (en info), b.v. { (C,9), (D,6), (H,11) }
- insert
- findmax (findmin)
- deletemax (deletemin)
- (optioneel) changepriority

Bij Dijkstra: kies knoop buiten boom met laagste kandidaatwaarde

Bij Branch & Bound (volgende week):
kies deeloplossing met beste ondergrens/bovengrens

Priority Queue:

- objecten met prioriteit (en info), b.v. $\{(C, 9), (D, 6), (H, 11)\}$
- insert: heap: $O(\log n)$
- findmax (findmin): heap: $O(1)$
- deletemax (deletemin): heap: $O(\log n)$
- (optioneel) changepriority: heap: $O(\log n)$

// invoer: samenhangende gewogen graaf $G = (V, E)$ en startknoop s
// uitvoer: array dat de lengtes van de kortste paden vanuit s bevat;
// na afloop is pad[v] = de lengte van een kortste pad van s naar v

for $v \in V$ **do**
    pad[v] := ∞;
od
pad[s] := 0;
$U := \emptyset$;

while ($U \neq V$) **do**
    vind knoop $v^* \in V \setminus U$ met pad[v^*] minimaal;
    $U := U \cup \{v^*\}$;
    for alle knopen v aangrenzend aan v^* **do**
        if pad[v^*] + gewicht(v^*, v) < pad[v] **then**
            pad[v] := pad[v^*] + gewicht(v^*, v);
        fi
    od
od

Complexiteit (met adjacency list en heap): . . .

// invoer: samenhangende gewogen graaf $G = (V, E)$ en startknoop s
// uitvoer: array dat de lengtes van de kortste paden vanuit s bevat;
// na afloop is pad[v] = de lengte van een kortste pad van s naar v

for $v \in V$ **do**
    pad[v] := ∞;
od
pad[s] := 0;
$U := \emptyset$;

while ($U \neq V$) **do**
    vind knoop $v^* \in V \setminus U$ met pad[v^*] minimaal;
    $U := U \cup \{v^*\}$;
    for alle knopen v aangrenzend aan v^* **do**
        if pad[v^*] + gewicht(v^*, v) < pad[v] **then**
            pad[v] := pad[v^*] + gewicht(v^*, v);
        fi
    od
od

Complexiteit (met adjacency list en heap):
$O(n + 1 + n + n \log n + m \log n) = O(m \log n)$

- In het algoritme bevat U steeds alle knopen waarvan de definitieve kortste afstand vanaf s reeds bepaald is. Voor deze knopen geeft het label pad[v] al de definitieve kortste afstand aan. Moet bewezen worden.
- Voor de andere knopen w geldt na elke ronde (= doorgang door de while):

  $$(\#)\ \text{pad}[w] = \min_{u \in U}\{\text{pad}[u] + \text{gewicht}(u, w)\}^*$$

  Dit volgt direct uit het algoritme.
- De volgende dichtstbijzijnde knoop v^* wordt gekozen uit de knopen uit $V \setminus U$ die direct grenzen aan U. Nadat deze gekozen is worden de labels aangepast, zodat (#) ook geldt voor de nieuwe U.
- Het is niet zo moeilijk dit algoritme aan te passen zodat ook de kortste paden zelf worden berekend. Sla direct na het aanpassen van het label van knoop v de nieuwe kandidaattak (v^*, v) op:

  if pad[v^*] + gewicht(v^*, v) < pad[v] **then**
  pad[v] := pad[v^*] + gewicht(v^*, v);
  nieuwe kandidaattak voor v: (v^*, v)
  fi

*Dit betekent dat pad[w] voor deze knopen $w \notin U$ de lengte van een kortste pad van s naar w aangeeft via uitsluitend knopen van U.

Na elke ronde (dus ook na de laatste, wanneer $U = V$) van het algoritme van Dijkstra geldt:

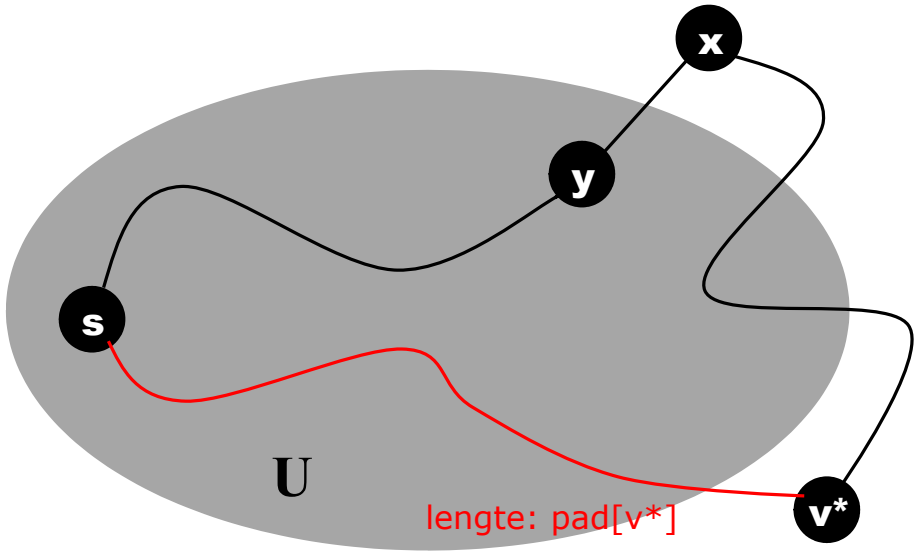
- U bevat alle knopen waarvan de definitieve kortste afstand vanaf s reeds bepaald is. Voor elke $v \in U$ geeft het label pad[v] die kortste afstand aan.

Om dit te bewijzen moet je laten zien dat:

- wanneer v^* wordt toegevoegd aan U, pad[v^*] inderdaad de lengte van het kortste pad van s naar v^* bevat

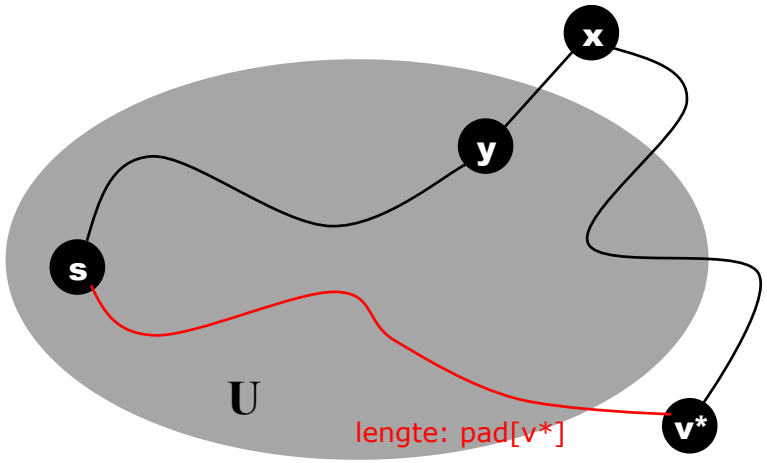
pad$[v^*]$ is daadwerkelijk lengte van een pad van s naar v^*:

$$(\#)\ \text{pad}[v^*] = \min_{u \in U}\{\text{pad}[u] + \text{gewicht}(u, v^*)\}^*$$



*Dit betekent dat pad$[v^*]$ de lengte van een kortste pad van s naar v^* aangeeft via uitsluitend knopen van U.

Bekijk, behalve het pad ter lengte pad[v^*] van s naar v^* via U een willekeurig ander pad van s naar v^*. Stel dat x de eerste knoop op dat pad is buiten U, en y de laatste knoop daarvóór. (Deze x kán gelijk zijn aan v^*.) Zij $d(s, y, x, v^*)$ de lengte van dat pad.



Dan geldt: $d(s, y, x, v^*) \geq d(s, y, x) = d(s, y) + \text{gewicht}(y, x) \geq \text{pad}[y] + \text{gewicht}(y, x) \geq \text{pad}[x] \geq \text{pad}[v^*]$ omdat respectievelijk alle gewichten van de takken ≥ 0 zijn, pad[y] volgens inductiehypothese kortste afstand is naar y, (#) geldt voor x, en v^* gekozen was in deze ronde als 'minimale' knoop.

QED

- **Lezen/leren bij dit college:**
  paragraaf 9.3, slides

- **Werkcollege** gretige algoritmen / Dijkstra:
  vrijdag 2 mei 2025, 11.00–12.45, BW.0.08

- **Opgaven:**
  zie `https://liacs.leidenuniv.nl/~vlietrvan1/algoritmiek/`

- **Practicumbijeenkomst** programmeeropdracht 3:
  Dinsdag 6 mei 2025, 09.00–10.45, DM.0.09, DM.0.13

- **Volgend (laatste!) hoorcollege:**
  Woensdag 7 mei 2025, 15.15–17.00

- **Daarna:**
  14 mei 2025: tentamen oefenen