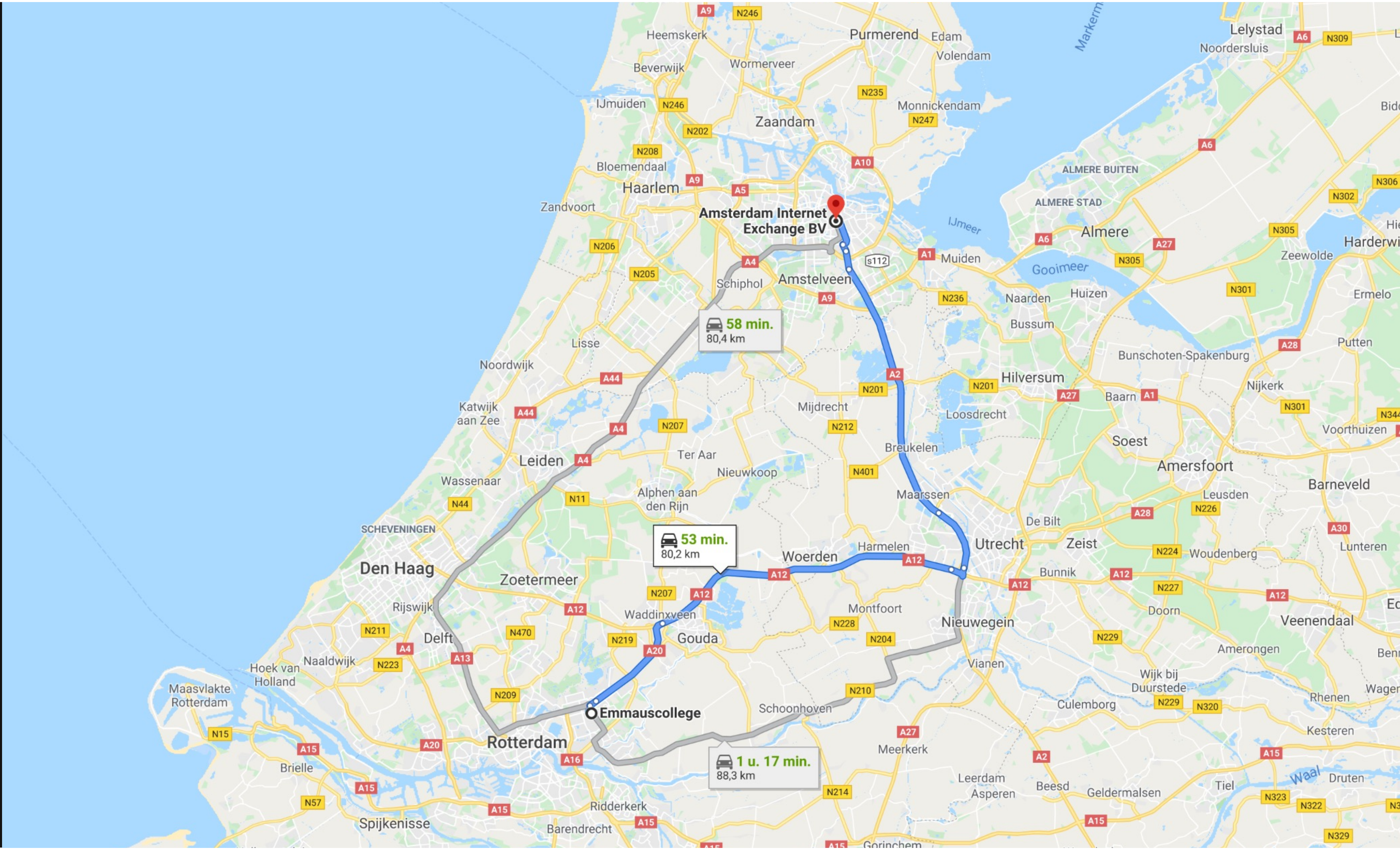
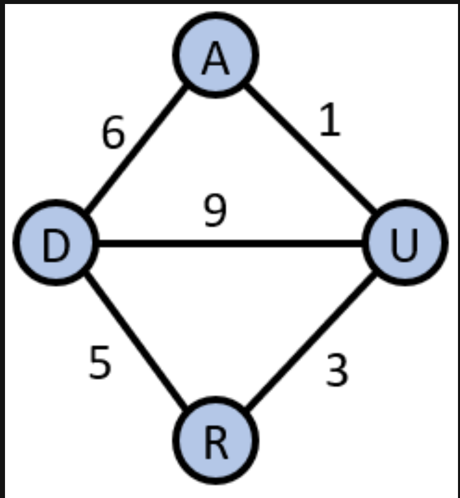
Het kortste-pad-algoritme


Amsterdam Internet Exchange BV
Emmauscollege
58 min.
80,4 km
53 min.
80,2 km
1 u. 17 min.
88,3 km

Hoe vindt Google Maps de kortste route?

Verschillende routes in een schematische voorstelling

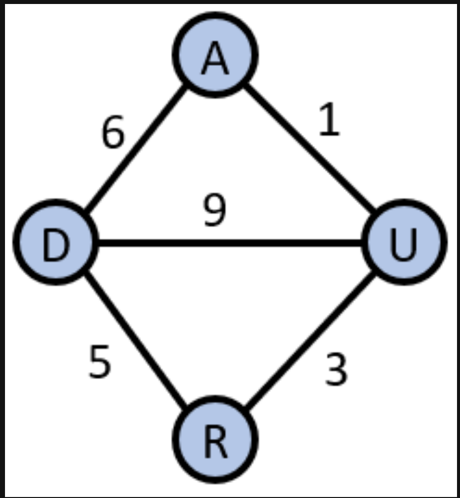


Startpunt	Eindpunt	Afstand
Amsterdam	Den Haag	6
Amsterdam	Utrecht	1
Den Haag	Rotterdam	5
Den Haag	Utrecht	9
Rotterdam	Utrecht	3

Een routekaart kun je modelleren als een **graaf**. Een graaf bevat:

- **Knopen** (denk aan steden of kruispunten)
- **Takken** (Denk aan wegen tussen steden of kruispunten)
- **Gewicht** (Denk aan reistijd of afstand per weg)
- **Pad** (Denk aan route over 1 of meerder takken van een startpunt naar een eindpunt)

Verschillende routes in een schematische voorstelling

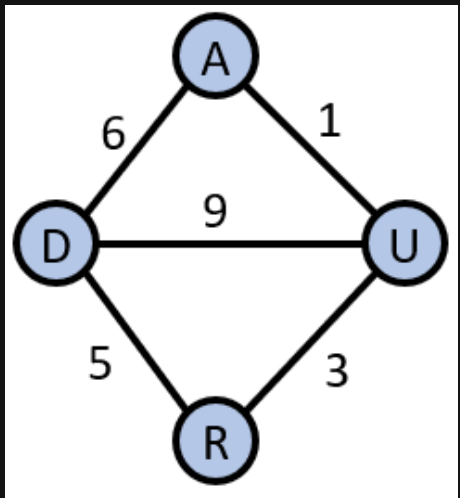


Startpunt	Eindpunt	Afstand
Amsterdam	Den Haag	6
Amsterdam	Utrecht	1
Den Haag	Rotterdam	5
Den Haag	Utrecht	9
Rotterdam	Utrecht	3

In ons eenvoudige voorbeeld geldt:

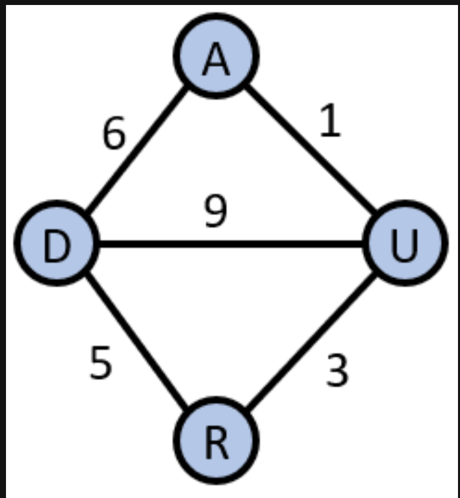
- Maximaal 1 directe weg tussen twee steden
- Heenweg duurt net zolang als terugweg
- Geen informatie over de soort weg, de locatie van de stad, enzovoorts.

1e mogelijke oplossing:



- Kies vanuit het startpunt steeds de kortste weg naar een volgende plaats;

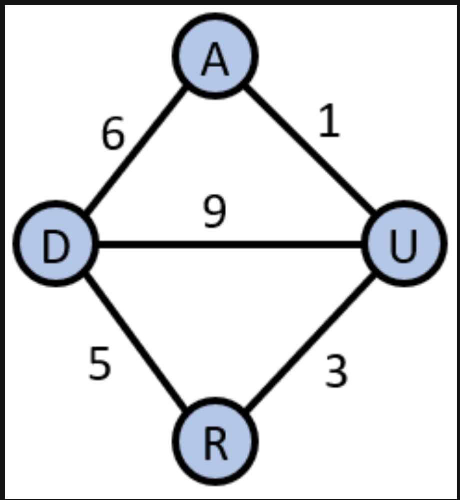
1e mogelijke oplossing:



- Kies vanuit het startpunt steeds de kortste weg naar een volgende plaats;

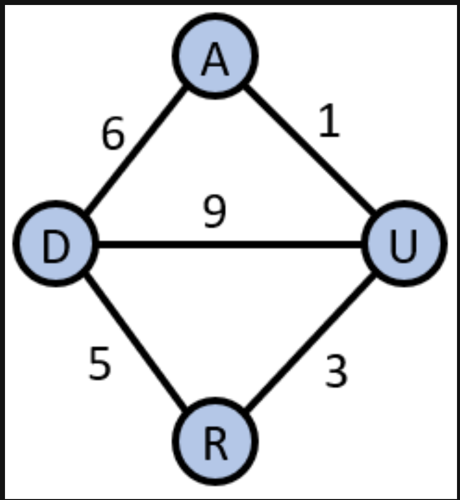
- Als je in alle naastliggende plaatsen geweest bent en het eindpunt is nog niet gevonden, ga je steeds een stapje terug, totdat je weer naar een naastliggende stad kunt waar je nog nooit bent geweest;

1e mogelijke oplossing:



- Kies vanuit het startpunt steeds de kortste weg naar een volgende plaats;
- Als je in alle naastliggende plaatsen geweest bent en het eindpunt is nog niet gevonden, ga je steeds een stapje terug, totdat je weer naar een naastliggende stad kunt waar je nog nooit bent geweest;
- Als je het eindpunt gevonden hebt, ben je klaar.

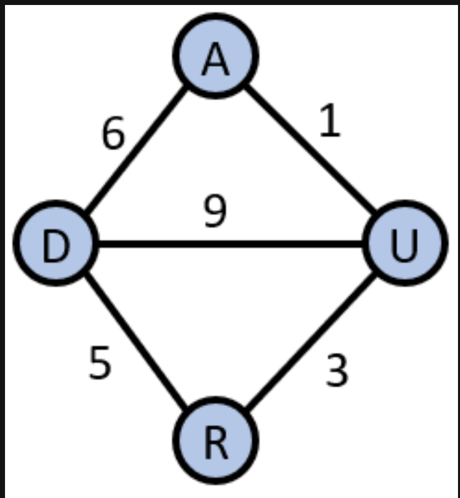
1e mogelijke oplossing:



- Kies vanuit het startpunt steeds de kortste weg naar een volgende plaats;
- Als je in alle naastliggende plaatsen geweest bent en het eindpunt is nog niet gevonden, ga je steeds een stapje terug, totdat je weer naar een naastliggende stad kunt waar je nog nooit bent geweest;
- Als je het eindpunt gevonden hebt, ben je klaar.

Maar dit algoritme is niet goed, want het vindt niet altijd het kortste pad.

1e mogelijke oplossing:

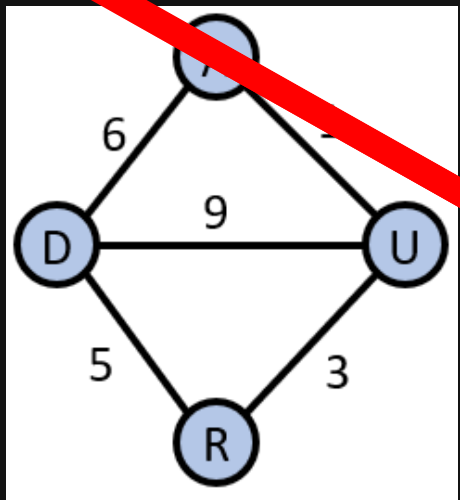


- Kies vanuit het startpunt steeds de kortste weg naar een volgende plaats;
- Als je in alle naastliggende plaatsen geweest bent en het eindpunt is nog niet gevonden, ga je steeds een stapje terug, totdat je weer naar een naastliggende stad kunt waar je nog nooit bent geweest;
- Als je het eindpunt gevonden hebt, ben je klaar.

Maar dit algoritme is niet goed, want het vindt niet altijd het kortste pad.

- Probeer het algoritme maar eens met de route van Den Haag naar Utrecht. Het algoritme kiest dan voor D-R-U, terwijl D-A-U korter is.

1e mogelijke oplossing:



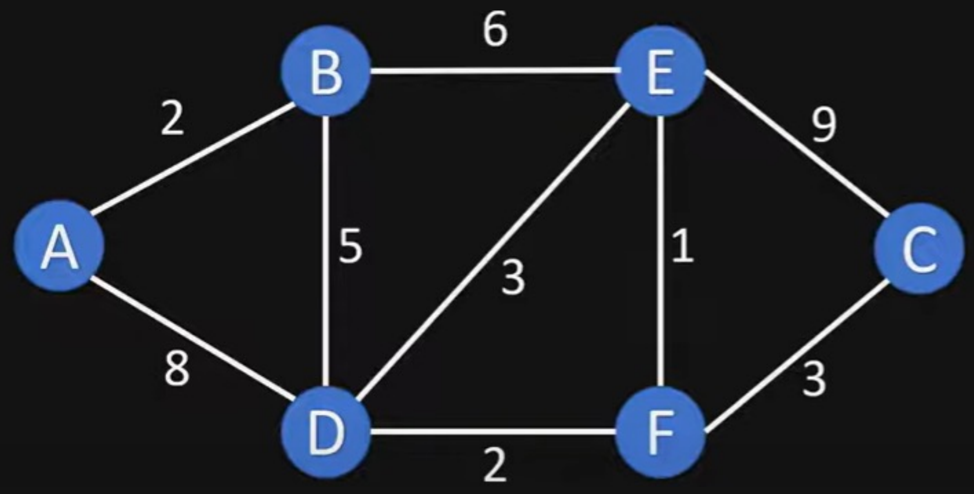
- Kies vanuit het startpunt steeds de kortste weg naar een volgende plaats;

- Als je in alle naastliggende plaatsen geweest bent en het eindpunt is nog niet gevonden, ga je steeds een stapje terug, totdat je weer naar een naastliggende stad kunt waar je nog nooit bent geweest;

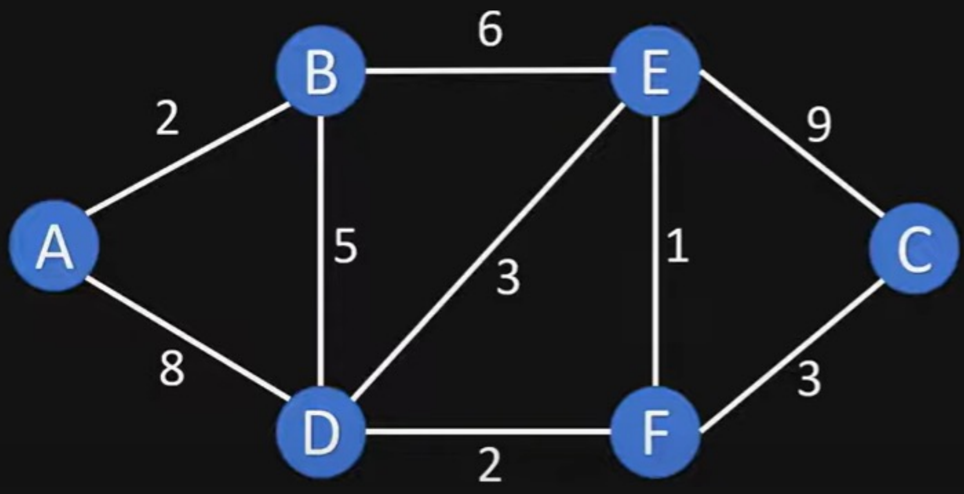
- Als je het eindpunt gevonden hebt, ben je klaar.

Maar dit algoritme is niet goed, want het vindt niet altijd het kortste pad.

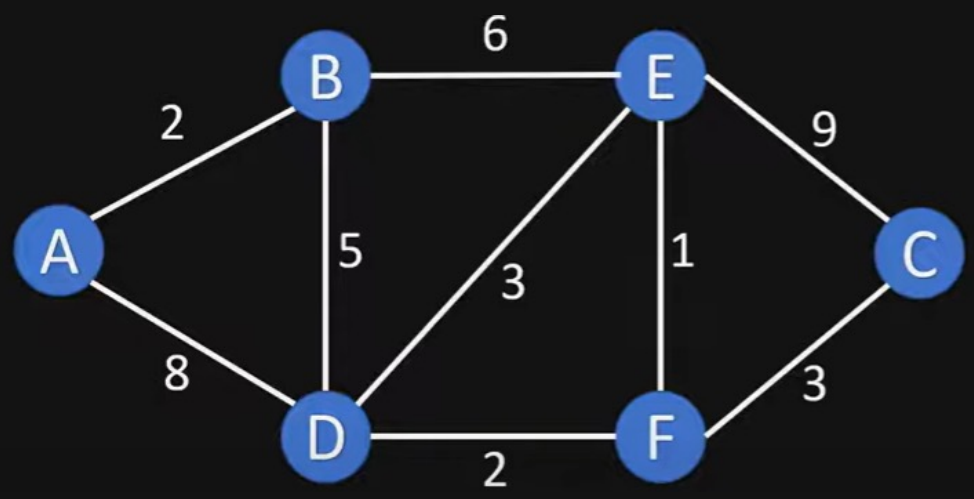
- Probeer het algoritme maar eens met de route van Den Haag naar Utrecht. Het algoritme kiest dan voor D-R-U, terwijl D-A-U korter is.


A
B
C
D
E
F
2
8
5
6
3
2
1
9
3

2e mogelijke oplossing: Probeer alle mogelijke takken

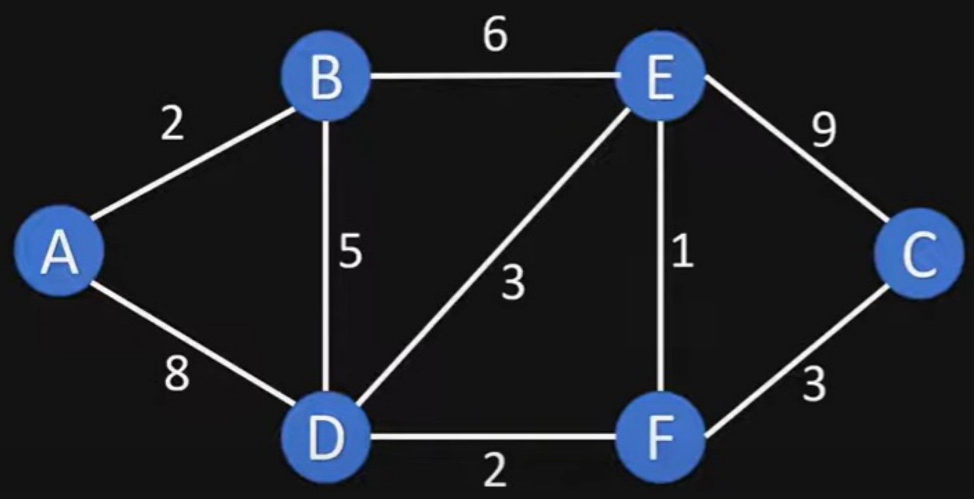


2e mogelijke oplossing: Probeer alle mogelijke takken



- **Ook dit algoritme is niet goed**, want je moet alle mogelijke takken tussen de verschillende knopen aflopen.

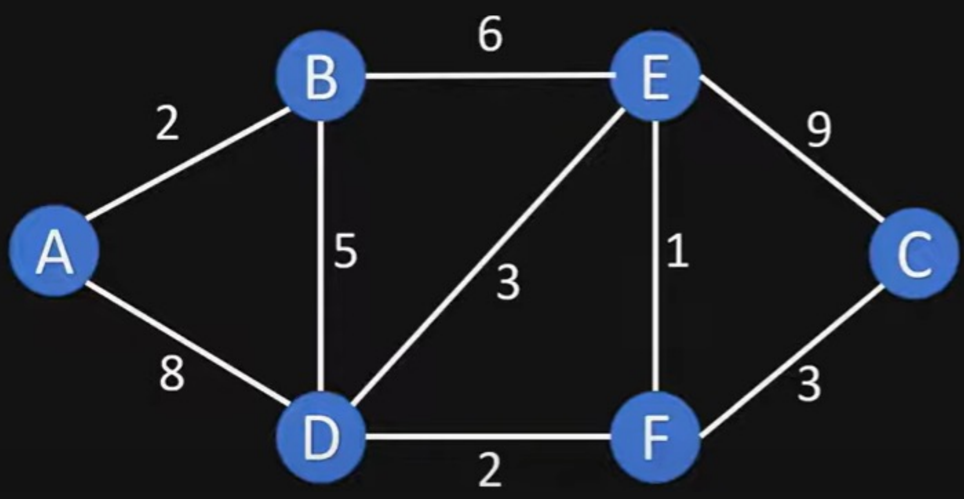
2e mogelijke oplossing: Probeer alle mogelijke takken



- **Ook dit algoritme is niet goed**, want je moet alle mogelijke takken tussen de verschillende knopen aflopen.

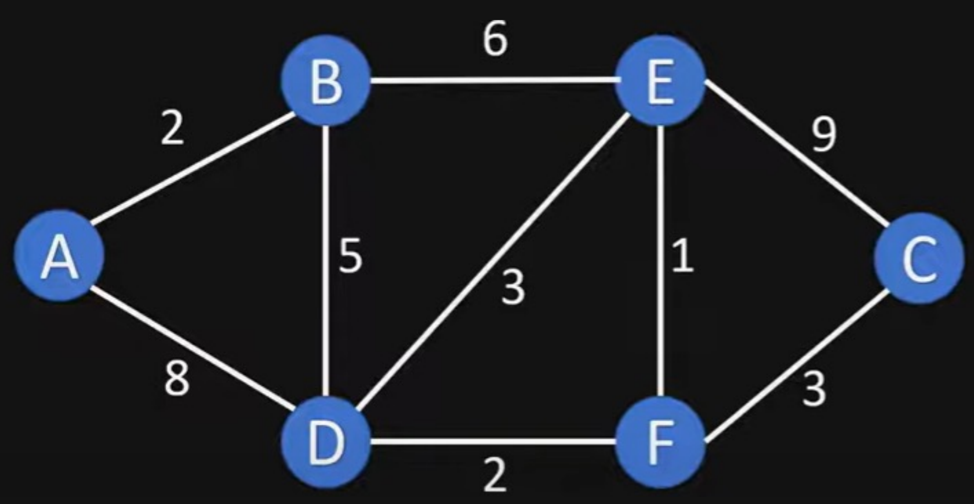
- Dit kost zeer veel rekenkracht en computergeheugen.

2e mogelijke oplossing: Probeer alle mogelijke takken



- **Ook dit algoritme is niet goed**, want je moet alle mogelijke takken tussen de verschillende knopen aflopen.
- Dit kost zeer veel rekenkracht en computergeheugen.
- Zo’n algoritme is niet efficiënt.

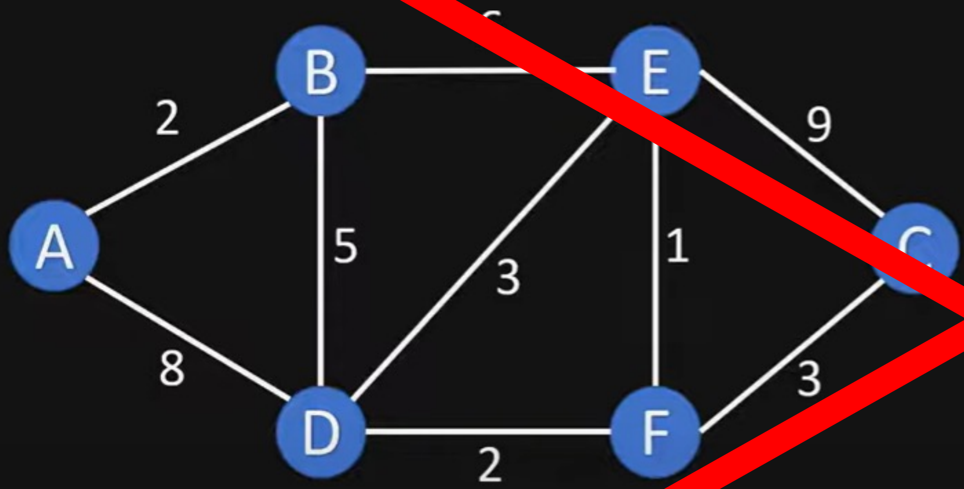
2e mogelijke oplossing: Probeer alle mogelijke takken



- **Ook dit algoritme is niet goed**, want je moet alle mogelijke takken tussen de verschillende knopen aflopen.
- Dit kost zeer veel rekenkracht en computergeheugen.
- Zo’n algoritme is niet efficiënt.

- Zo kunnen we hier vele unieke paden vinden, die elk uit een unieke set takken bestaan.

2e mogelijke oplossing: Probeer alle mogelijke takken



- **Ook dit algoritme is niet goed**, want je moet alle mogelijke takken tussen de verschillende knopen aflopen.
- Dit kost zeer veel rekenkracht en computergeheugen.
- Zo'n algoritme is niet efficiënt.

- Zo kunnen we hier vele unieke paden vinden, die elk uit een unieke set takken bestaan.

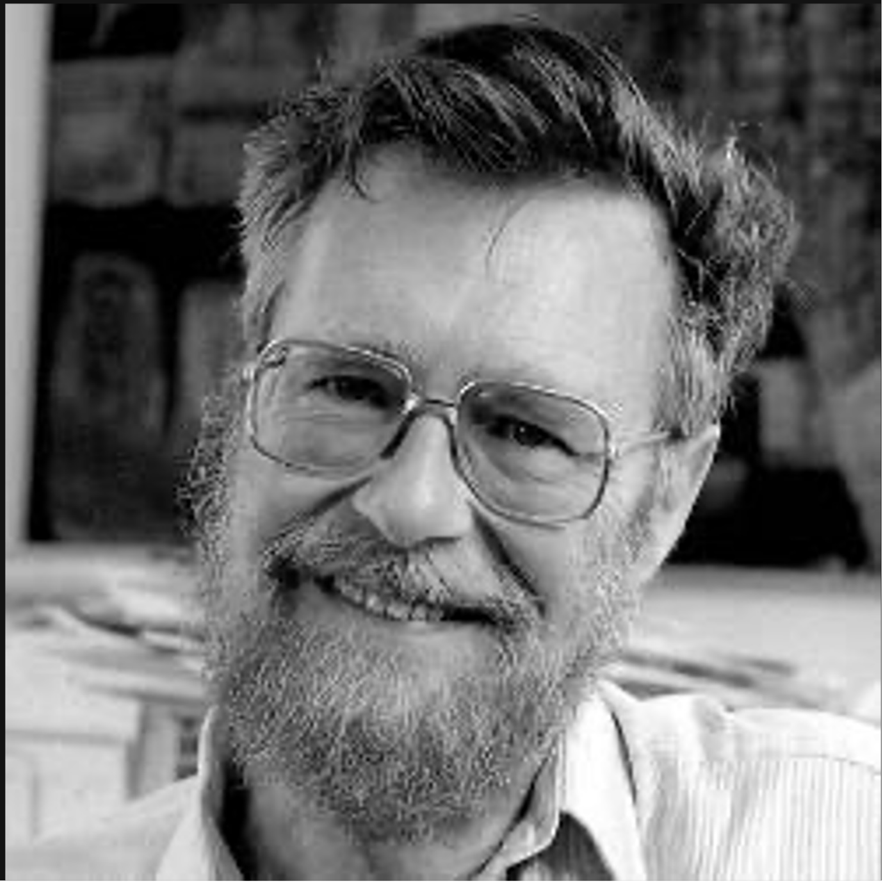
Juiste oplossing: Het algoritme van Dijkstra

Edsger Wybe Dijkstra

(1930 - 2002)

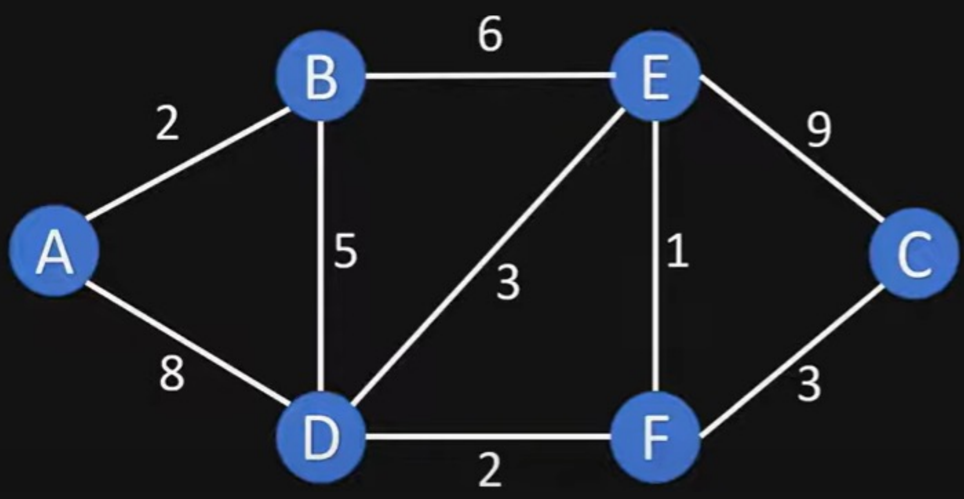
- Rotterdamse wiskundige en informaticus
- Bedacht het kortste-pad-algoritme in 1959 in 20 minuten met pen en papier.

Bron: https://cacm.acm.org/magazines/2010/8/96632-an-interview-with-edsger-w-dijkstra/fulltext

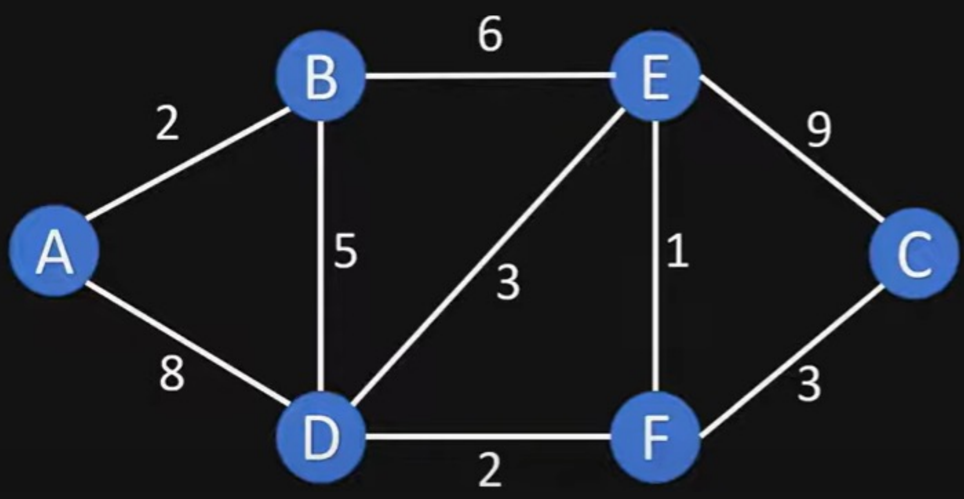


Het algoritme van Dijkstra

Het algoritme van Dijkstra

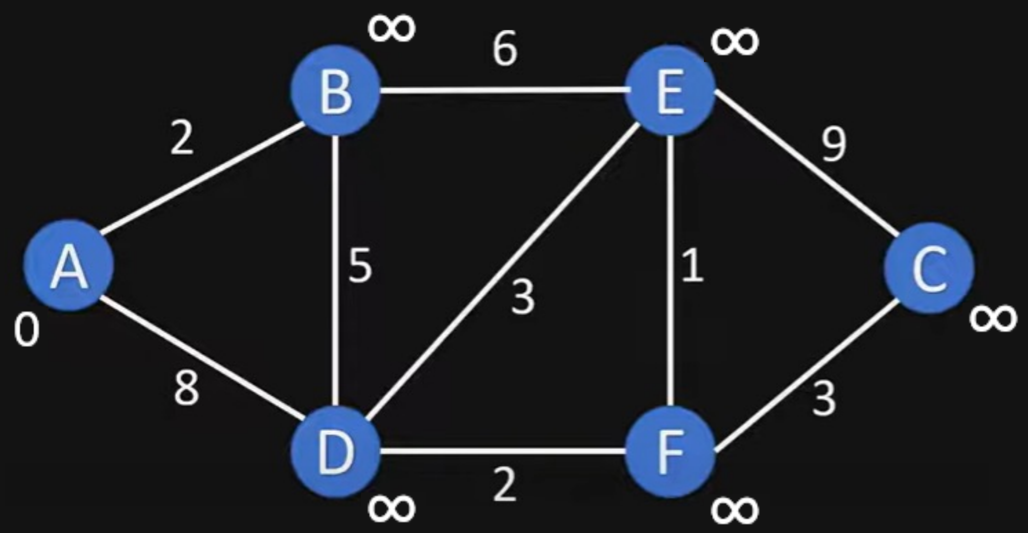


Het algoritme van Dijkstra



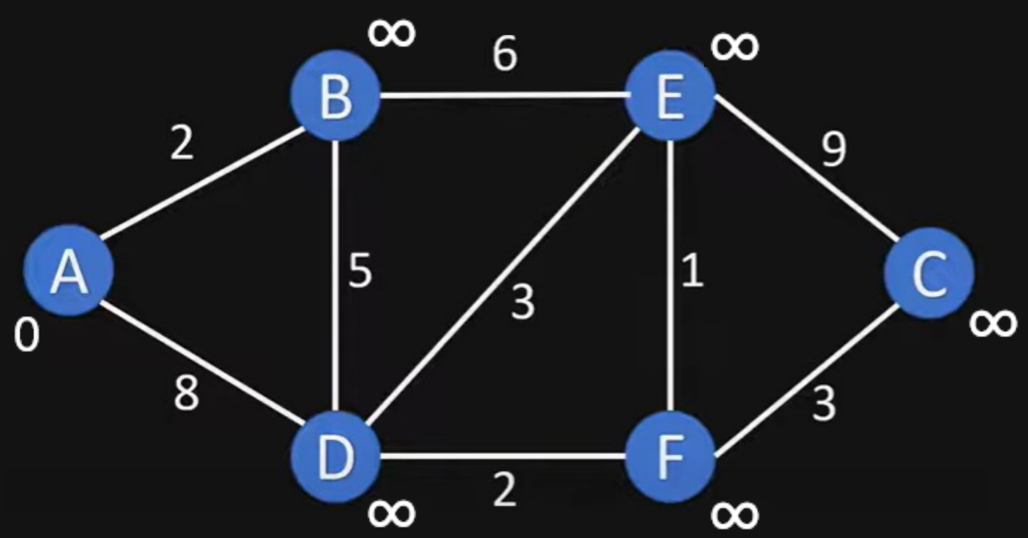
1. Stel dat A onze startknop is en C onze eindknoop.

Het algoritme van Dijkstra



1. Stel dat A onze startknop is en C onze eindknoop.
2. We markeren alleen de startknoop met de waarde 0 en alle andere knopen met oneindig (∞), doordat deze knopen nog niet bezocht zijn en de afstanden tot deze knopen nog niet bekend zijn.

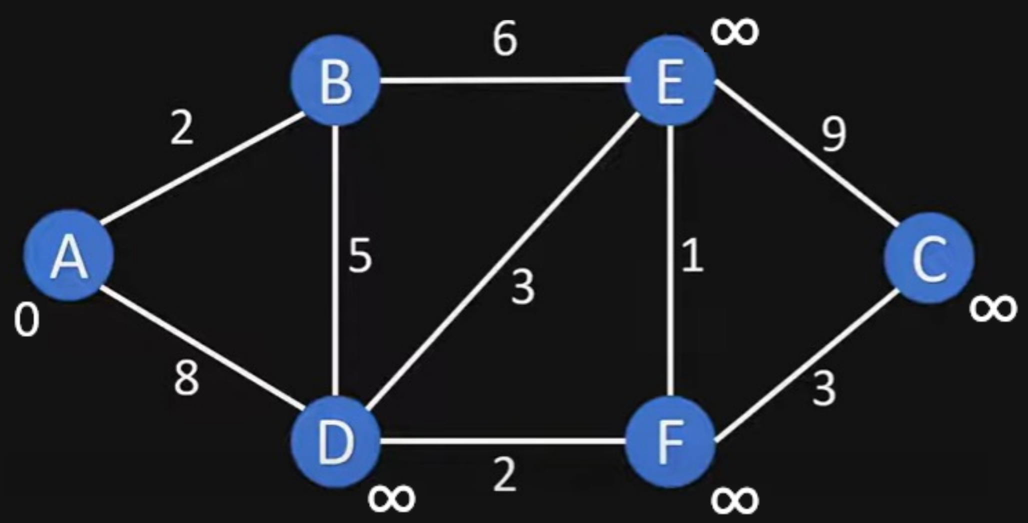
Het algoritme van Dijkstra



3. Vat alle afstanden in een tabel en laat open wat niet bekend is.

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	∞	
C	∞	
D	∞	
E	∞	
F	∞	

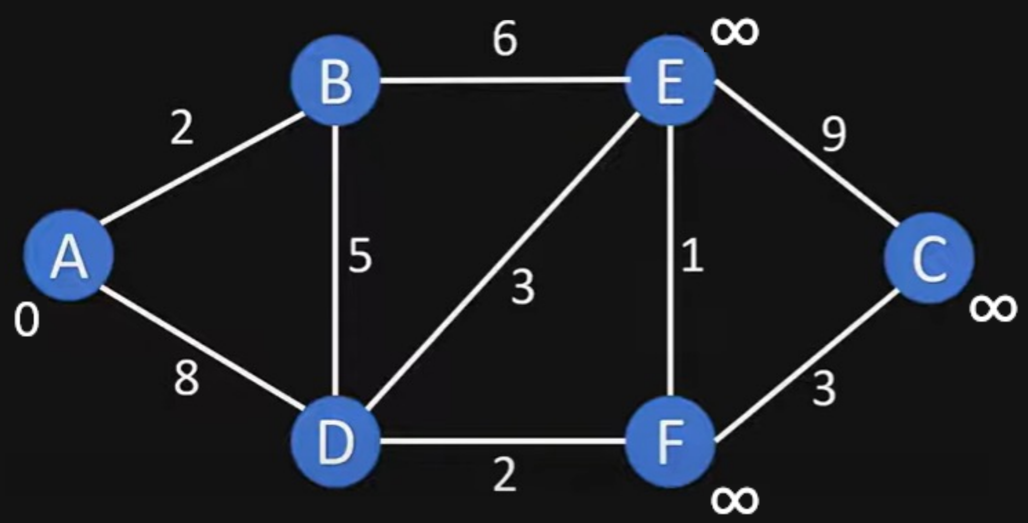
Het algoritme van Dijkstra



4. We zien dat het pad naar B, vanuit A, het kortste is en we noteren:

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	∞	
D	∞	
E	∞	
F	∞	

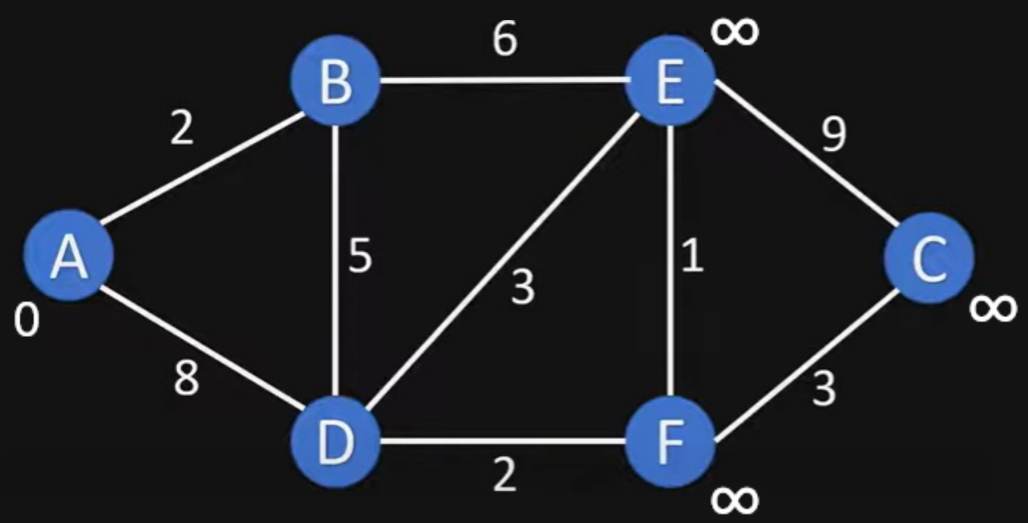
Het algoritme van Dijkstra



5. We zien dat het pad naar D, vanuit A, vervolgens 8 is en we noteren:

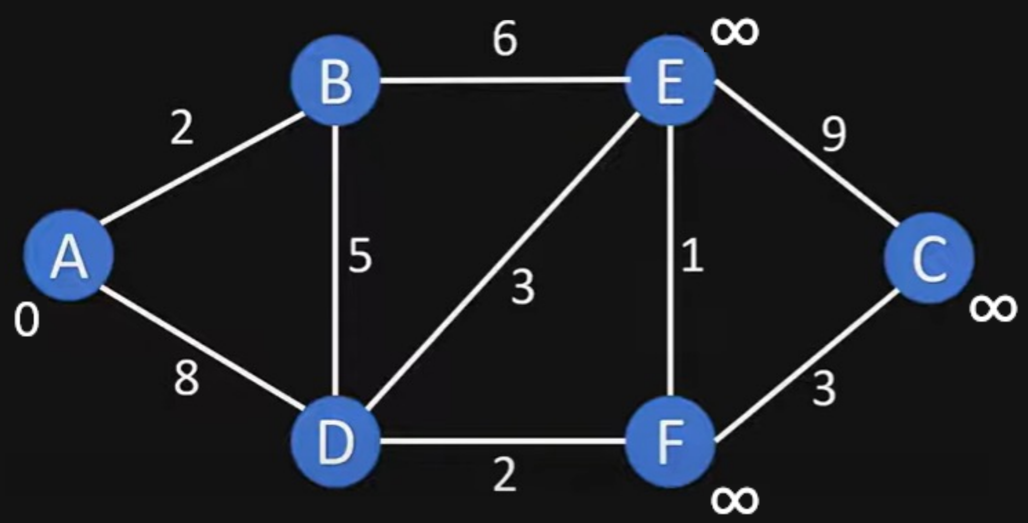
Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	∞	
D	8	A
E	∞	
F	∞	

Het algoritme van Dijkstra



6. Nu we alle takken van A gehad hebben, nemen we de knoop met de kortste afstand tot A, in dit geval knoop B, onder de loep.

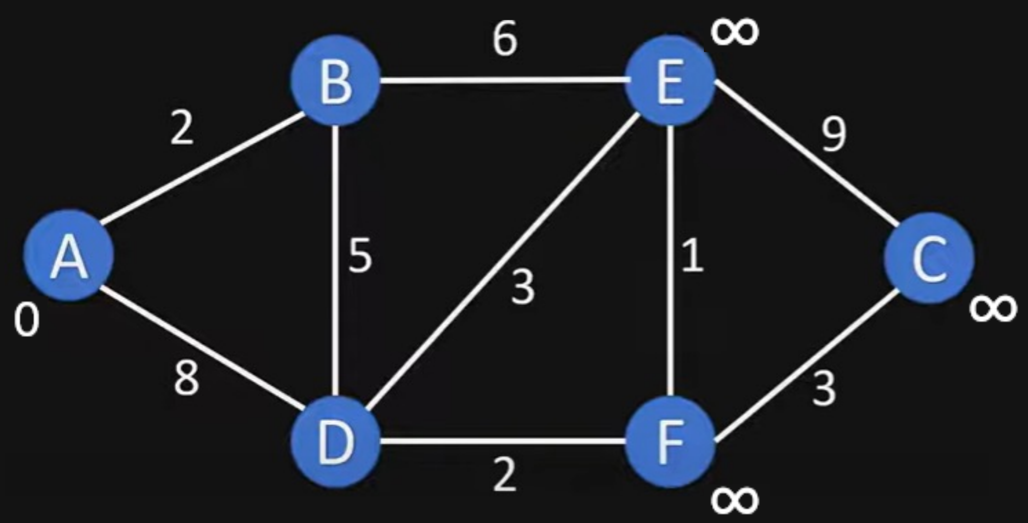
Het algoritme van Dijkstra



7. We zien dat het pad naar D, vanuit A, over B, lager is en we updaten D:

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	∞	
D	7	B
E	∞	
F	∞	

Het algoritme van Dijkstra

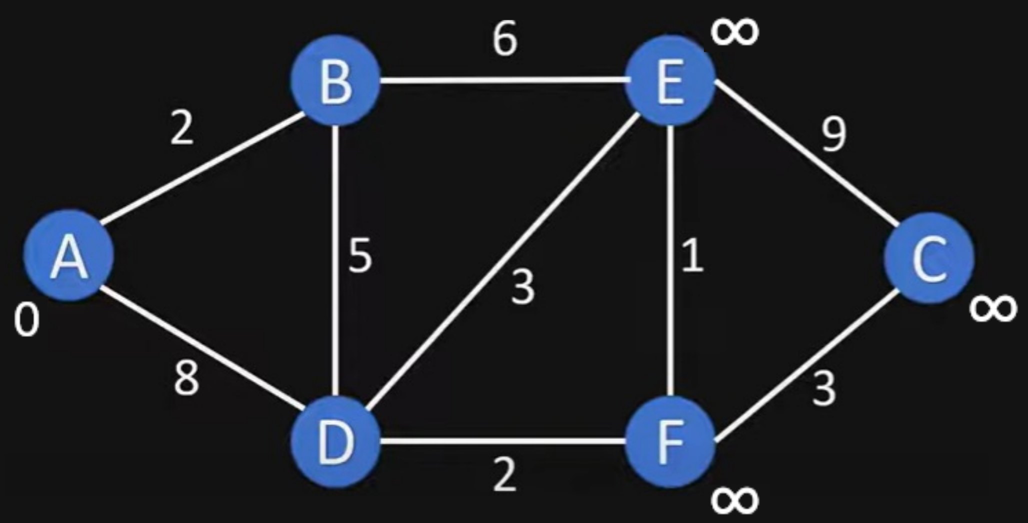


7. We zien dat het pad naar D, vanuit A, over B, lager is en we updaten D:

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	∞	
D	7	B
E	∞	
F	∞	

De kortste afstand van het pad naar D, vanuit A, over B, wordt geüpdate van 8 naar 7.

Het algoritme van Dijkstra

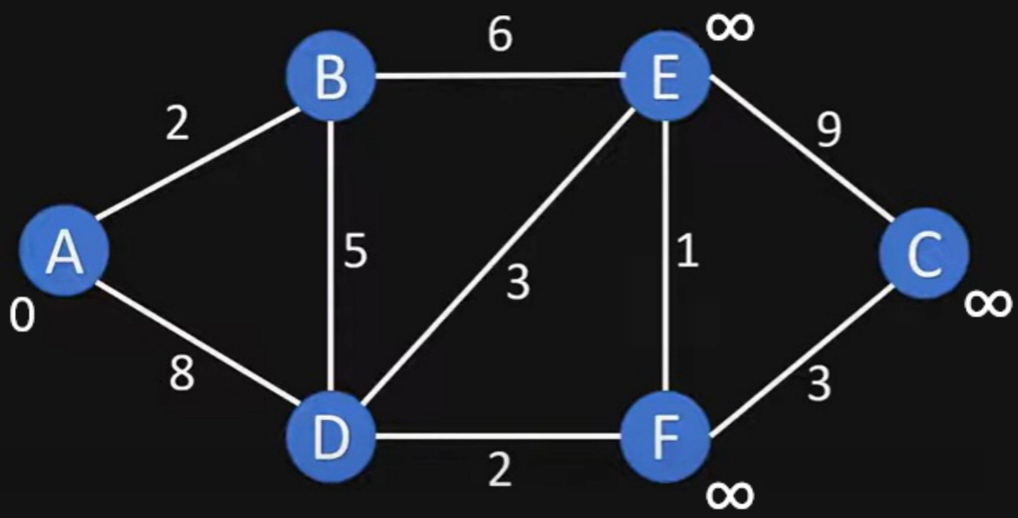


7. We zien dat het pad naar D, vanuit A, over B, lager is en we updaten D:

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	∞	
D	7	B
E	∞	
F	∞	

We kunnen tot zover uit de tabel aflezen dat het kortste pad tussen A en D via B loopt.

Het algoritme van Dijkstra

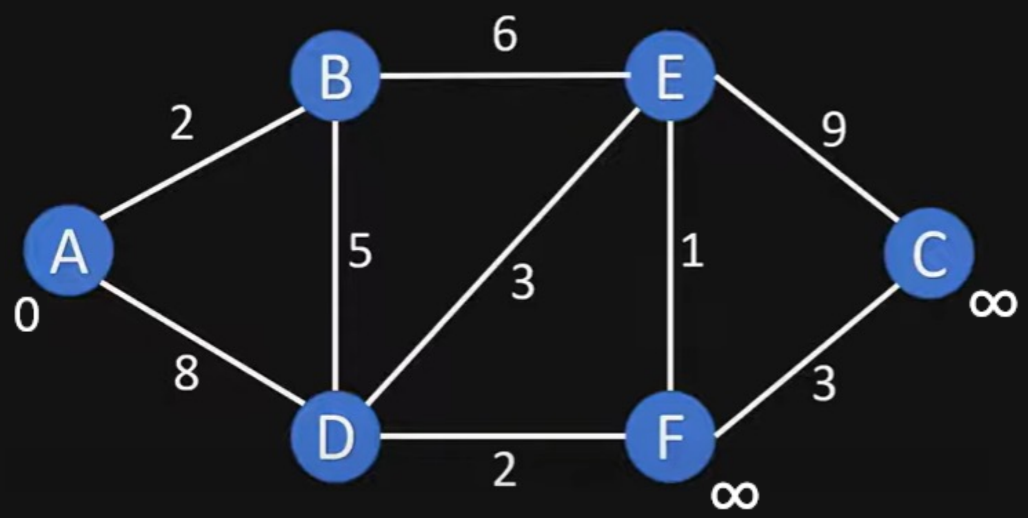


7. We zien dat het pad naar D, vanuit A, over B, lager is en we updaten D:

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	∞	
D	7	B
E	∞	
F	∞	

We kunnen tot zover uit de tabel aflezen dat het kortste pad tussen A en D via B loopt.

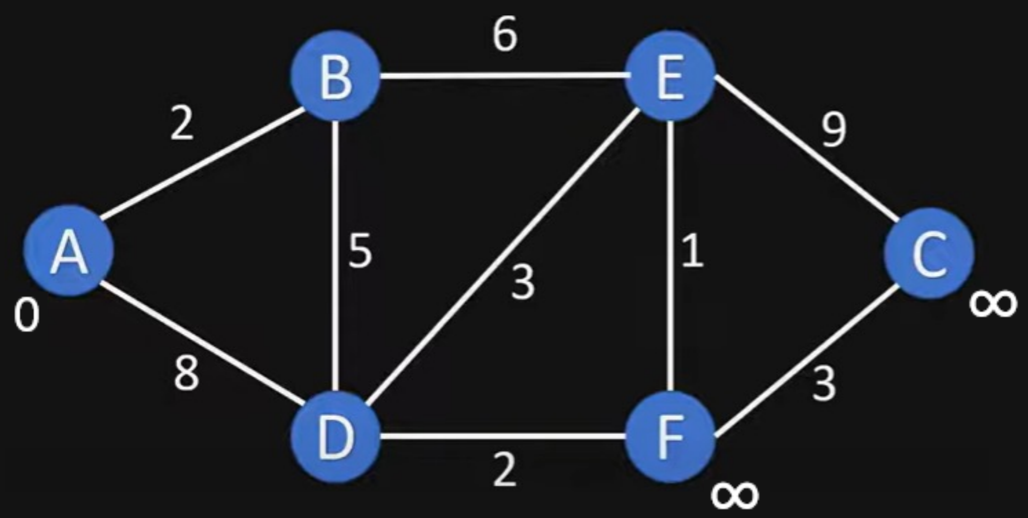
Het algoritme van Dijkstra



8. We zien vervolgens dat het pad naar E, vanuit A, over B, 2 + 6 = 8 is:

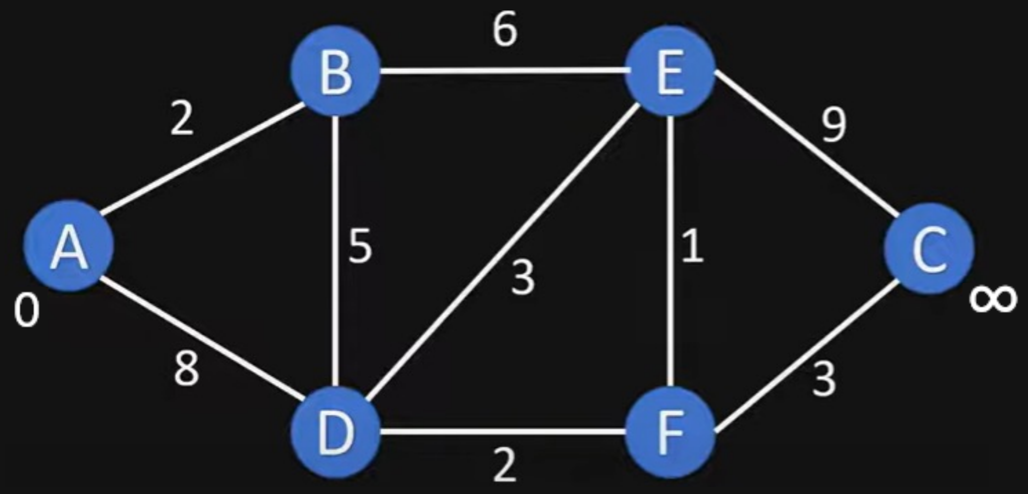
Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	∞	
D	7	B
E	8	B
F	∞	

Het algoritme van Dijkstra



9. Alle takken van B zijn bezocht, waardoor we eerst de volgende knoop onder de loep nemen met de kortste afstand tot A, in dit geval D met een afstand 7.

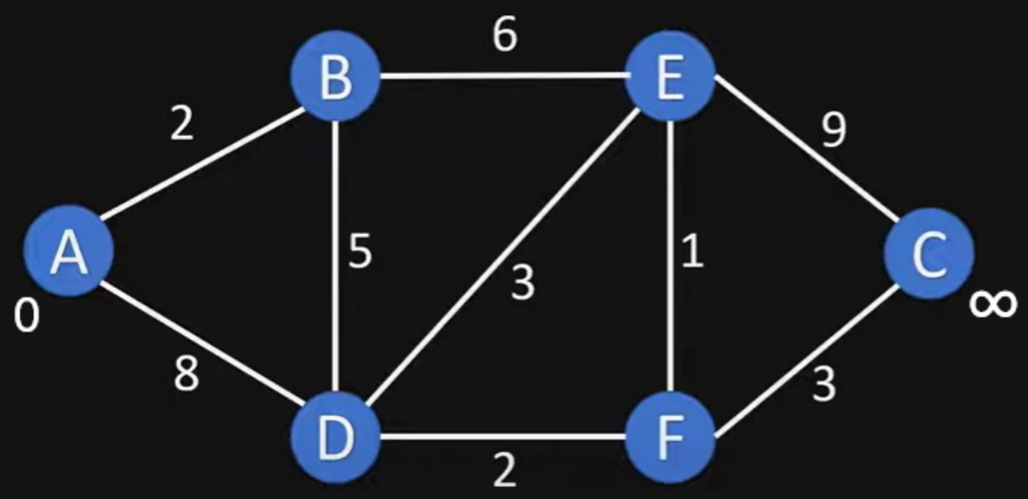
Het algoritme van Dijkstra



10. We zien dat het kortste pad, naar F, vanuit A, over D, 7 + 2 = 9 is:

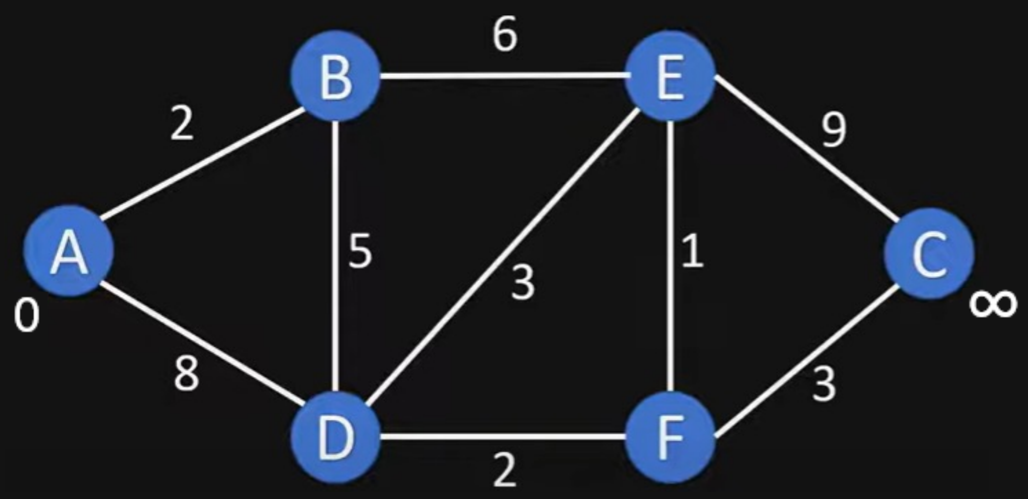
Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	∞	
D	7	B
E	8	B
F	9	D

Het algoritme van Dijkstra



11. We zien vervolgens dat het pad naar E, vanuit A, over D, 7 + 3 = 10 is, maar het kortste pad naar E, vanuit A, is 8, lager, waardoor we de tabel niet updaten.

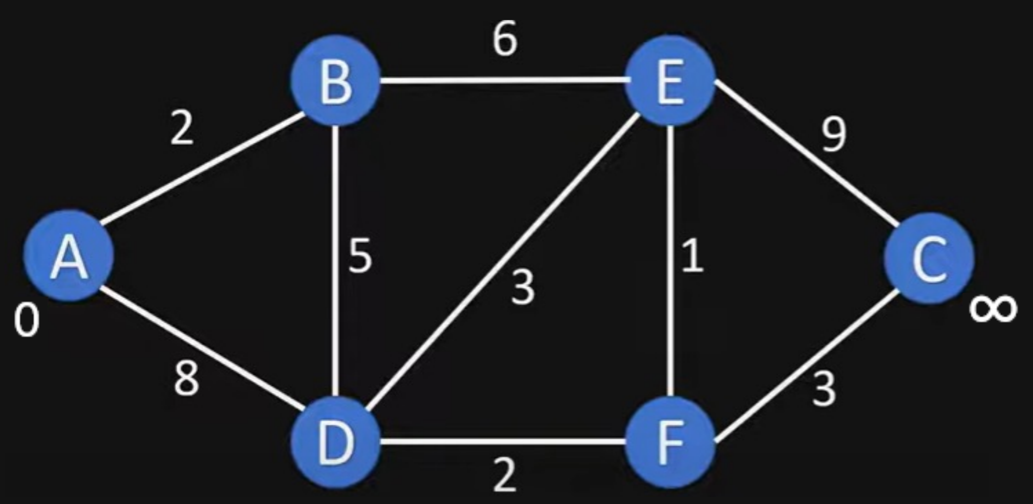
Het algoritme van Dijkstra



11. We zien vervolgens dat het pad naar E, vanuit A, over D, $7 + 3 = 10$ is, maar het kortste pad naar E, vanuit A, is 8, lager, waardoor we de tabel niet updaten.

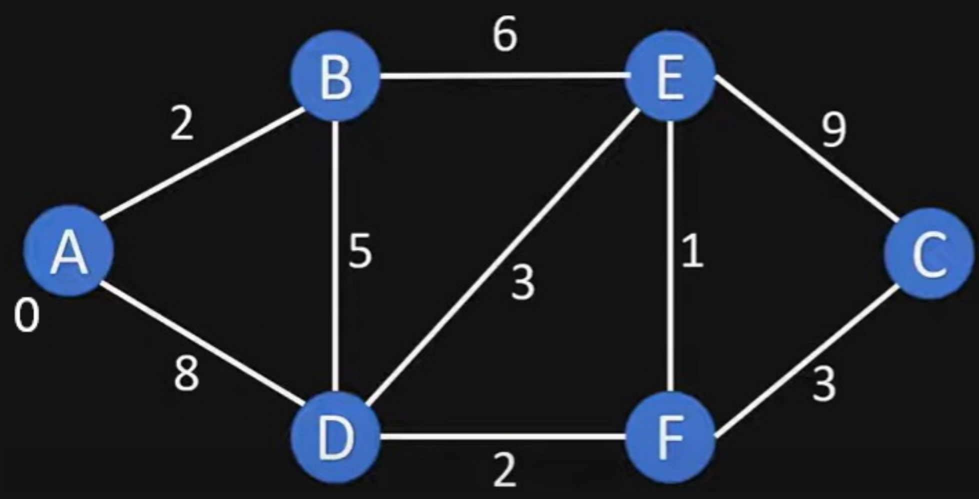
12. Alle takken van D zijn bezocht, waardoor we eerst de volgende knoop onder de loep nemen met de kortste afstand tot A, in dit geval E met een afstand 8.

Het algoritme van Dijkstra



13. We zien dat het kortste pad, naar F, vanuit A, over E, 8 + 1 = 9 is, maar doordat het kortste pad naar F, vanuit A, over E gelijk is aan de kortste afstand die we hiervoor al hadden gevonden, namelijk 9 over D (en B), hoeven we de tabel niet te updaten.

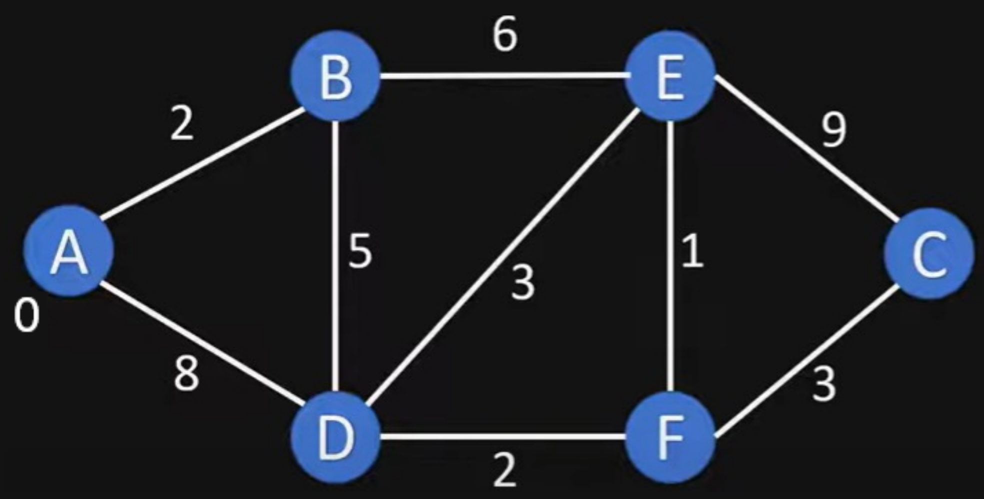
Het algoritme van Dijkstra



14. We zien vervolgens dat het pad naar C, vanuit A, over E, 8 + 9 = 17 is:

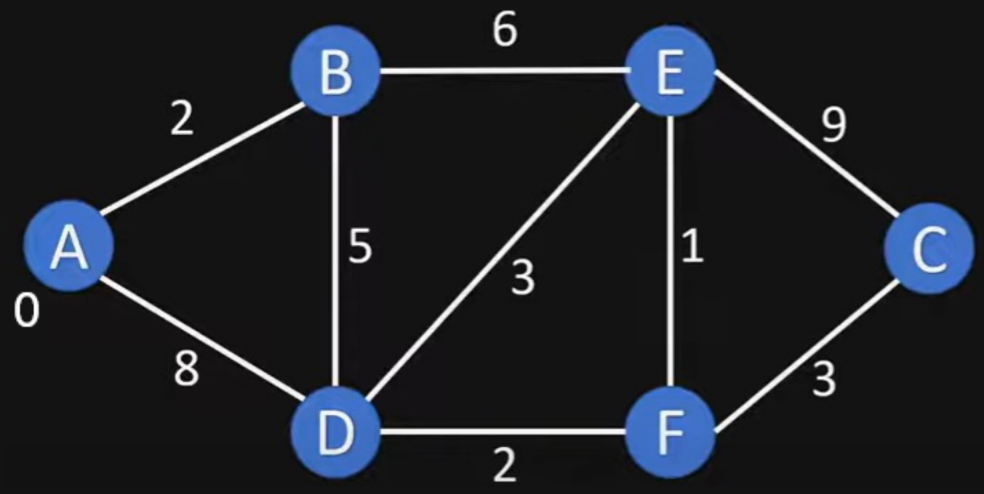
Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	17	E
D	7	B
E	8	B
F	9	D

Het algoritme van Dijkstra



15. Alle takken van E zijn bezocht, waardoor we eerst de volgende knoop onder de loep nemen met de kortste afstand tot A, in dit geval F met een afstand 9.

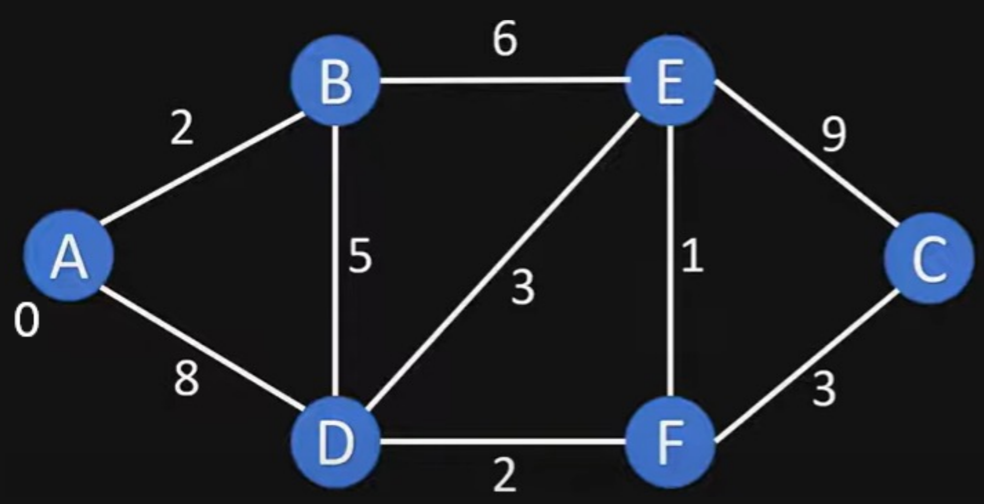
Het algoritme van Dijkstra



15. Alle takken van E zijn bezocht, waardoor we eerst de volgende knoop onder de loep nemen met de kortste afstand tot A, in dit geval F met een afstand 9.

16. De afstand tussen A en F, via E, hebben we reeds bekeken, waardoor we de afstand tussen A en C, via F, bekijken, welke 9 + 3 = 12 is, waarmee we onze tabel updaten.

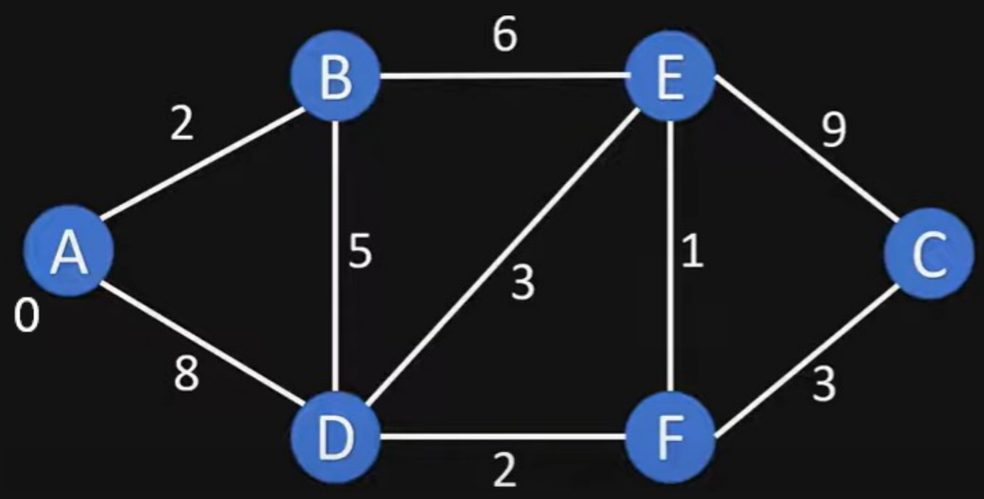
Het algoritme van Dijkstra



17. Dat leidt tot onze eindtabel:

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	12	F
D	7	B
E	8	B
F	9	D

Het algoritme van Dijkstra

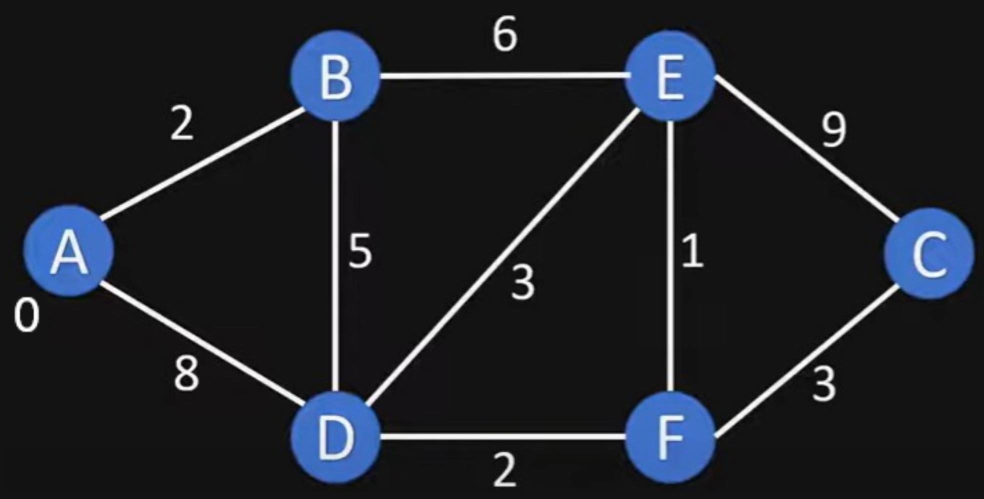


17. Dat leidt tot onze eindtabel:

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	12	F
D	7	B
E	8	B
F	9	D

De kortste afstand van het pad naar C, vanuit A, over F, is geüpdate van 17 naar 12.

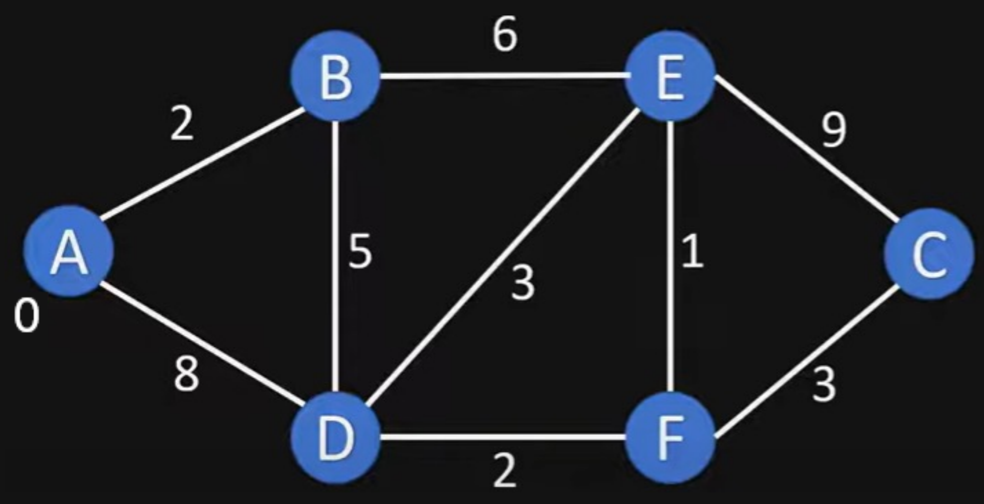
Het algoritme van Dijkstra



18. Hieruit vinden we de kortste route:

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	12	F
D	7	B
E	8	B
F	9	D

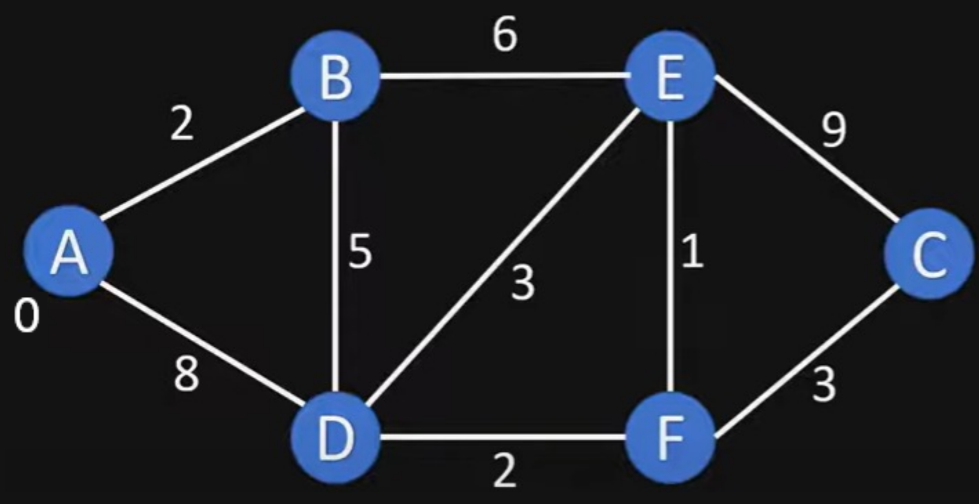
Het algoritme van Dijkstra



18. Hieruit vinden we de kortste route:

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	12	F
D	7	B
E	8	B
F	9	D

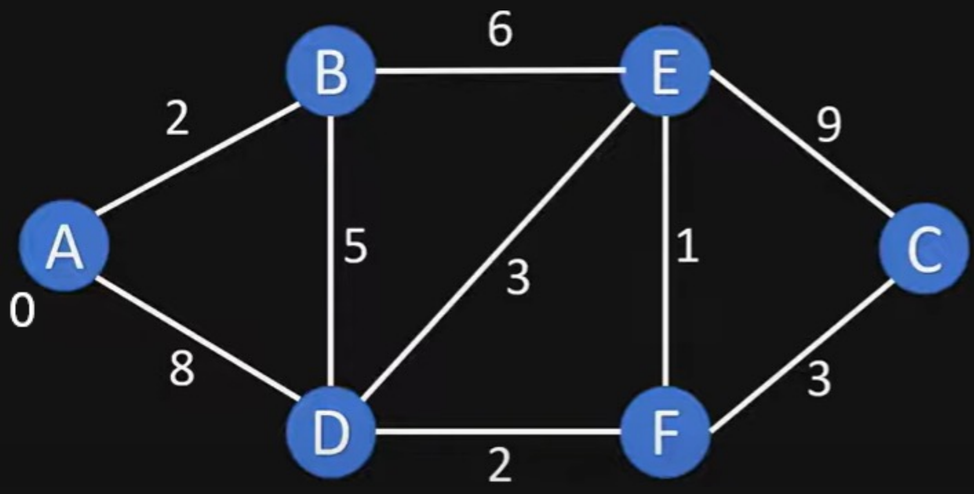
Het algoritme van Dijkstra



18. Hieruit vinden we de kortste route:

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	12	F
D	7	B
E	8	B
F	9	D

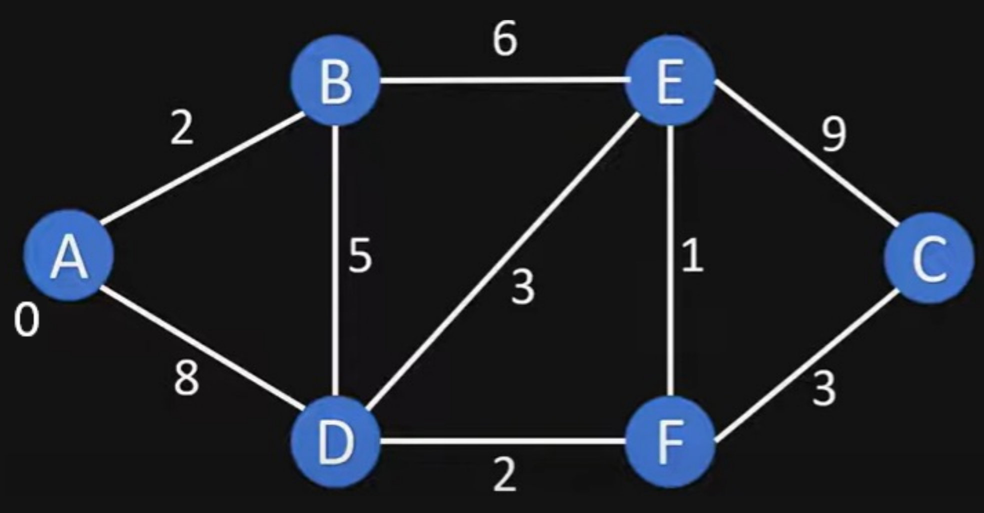
Het algoritme van Dijkstra



18. Hieruit vinden we de kortste route:

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	12	F
D	7	B
E	8	B
F	9	D

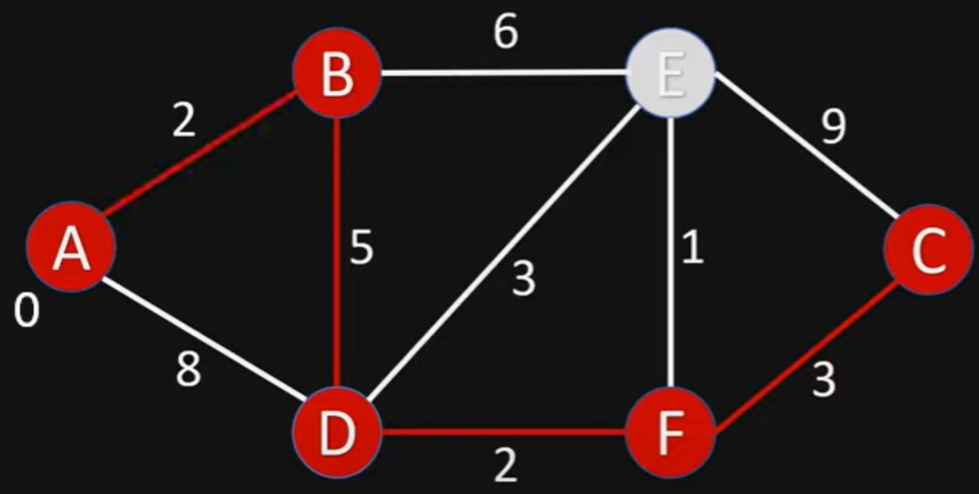
Het algoritme van Dijkstra



18. Hieruit vinden we de kortste route:

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	12	F
D	7	B
E	8	B
F	9	D

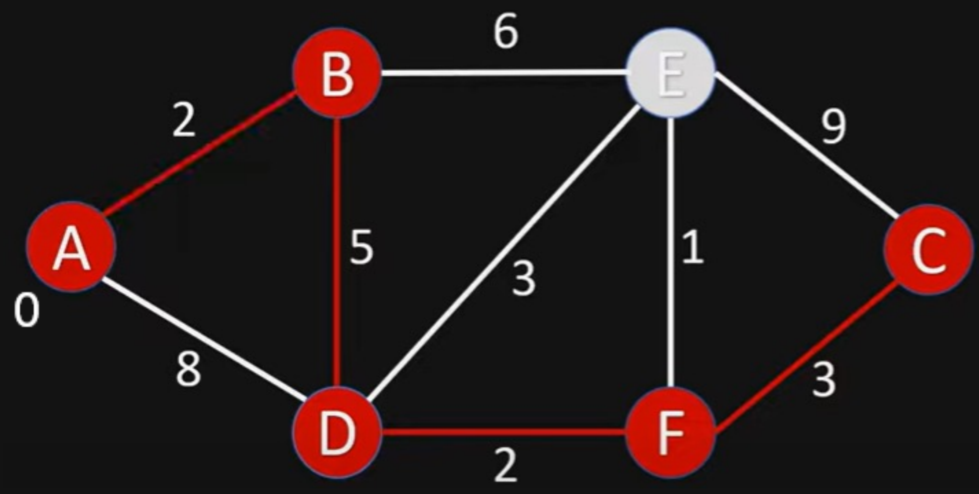
Het algoritme van Dijkstra



18. Hieruit vinden we de kortste route:

Knoop	Kortste afstand	Voorgaande knoop
A	0	A
B	2	A
C	12	F
D	7	B
E	8	B
F	9	D

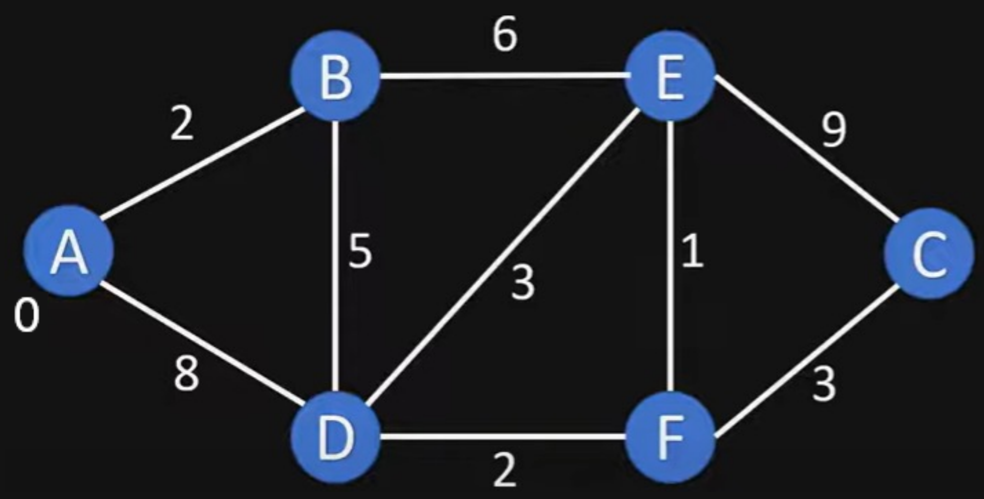
Het algoritme van Dijkstra



Samenvattend:

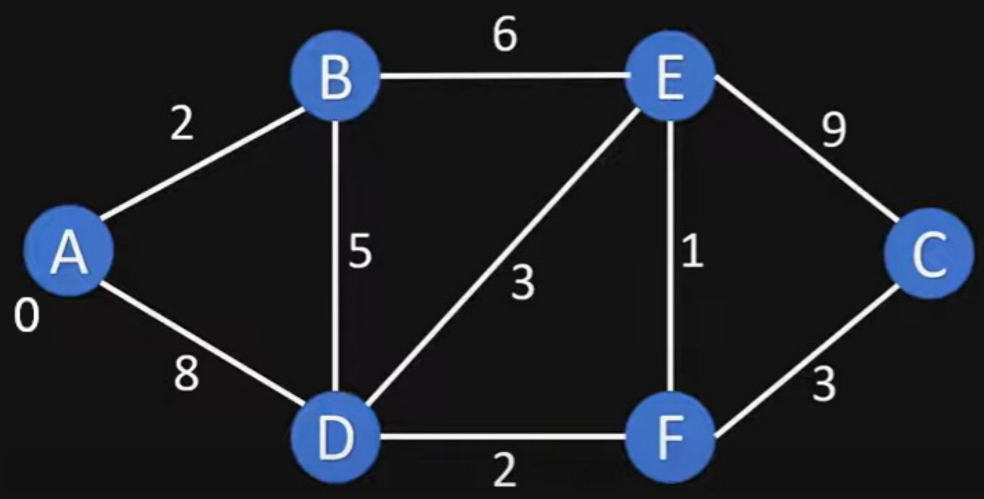
- Start met de tak met de laagste waarde en loop ze in opeenvolgende volgorde af;
- Neem vervolgens de knoop met de kortste afstand tot de startknoop en herhaal het procedé;
- Werk onderwijl de tabel bij, waar nodig.
- Houd bij welke knopen je afgewerkt hebt.

Het algoritme van Dijkstra



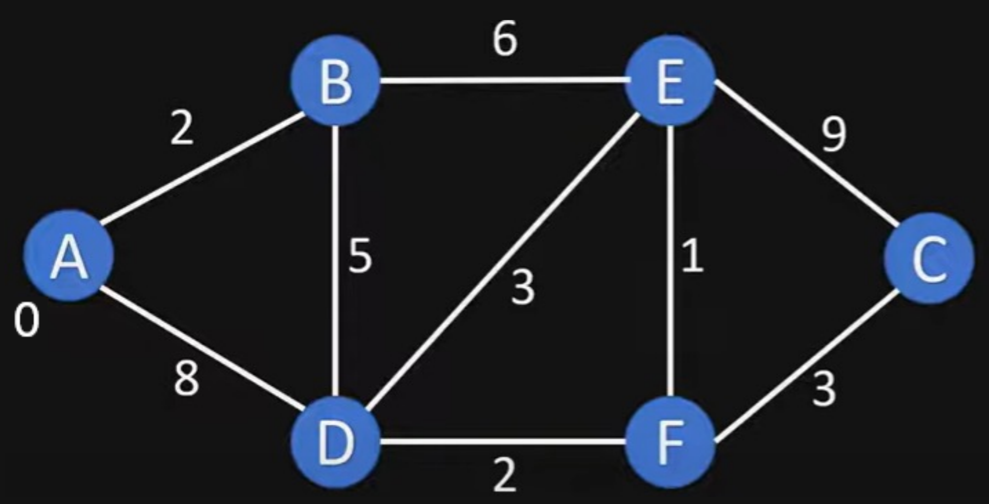
- Je kan je voorstellen dat een computer alle variaties tussen verschillende start- en eindknopen kan doorrekenen.

Het algoritme van Dijkstra



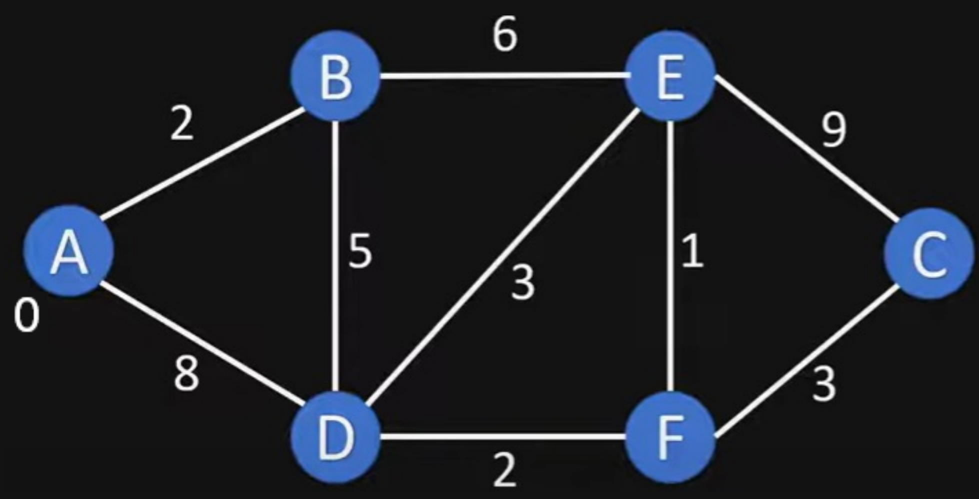
- Je kan je voorstellen dat een computer alle variaties tussen verschillende start- en eindknopen kan doorrekenen.
- Gelukkig hoef je daar niet zelf een algoritme of verschillende functies voor te schrijven, want deze bestaan vaak al, ook in Javascript.

Het algoritme van Dijkstra



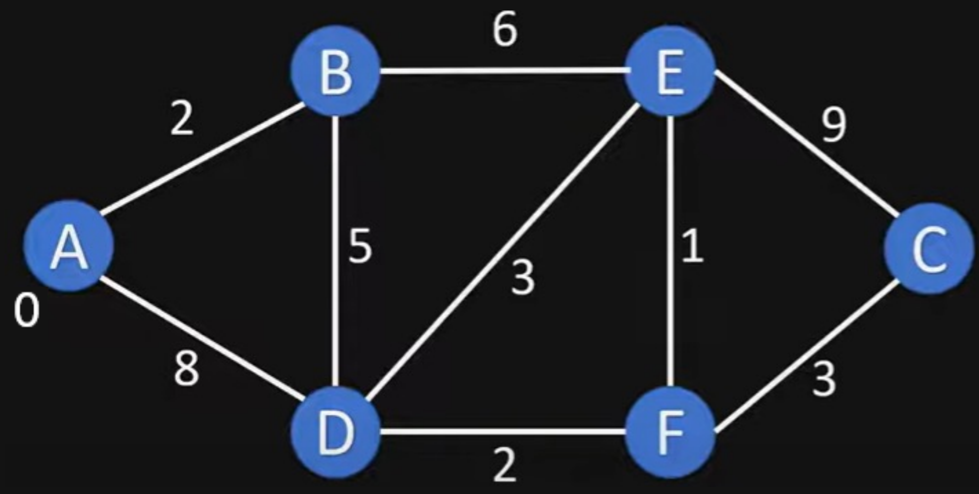
- Je kan je voorstellen dat een computer alle variaties tussen verschillende start- en eindknopen kan doorrekenen.
- Gelukkig hoef je daar niet zelf een algoritme of verschillende functies voor te schrijven, want deze bestaan vaak al, ook in Javascript.
- Deze Javascript-functie krijg je uitgereikt.

Een graaf in een matrix



- Zo stoppen computers grafen in een matrix, voordat ze doorgerekend worden.

Een graaf in een matrix

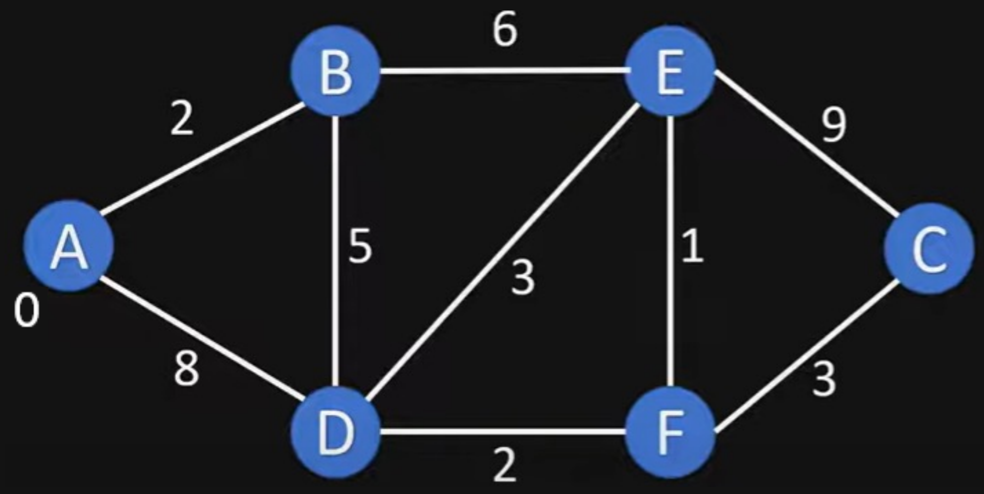


- Zo stoppen computers grafen in een matrix, voordat ze doorgerekend worden.

- Een voorbeeld van de behandelde graaf:

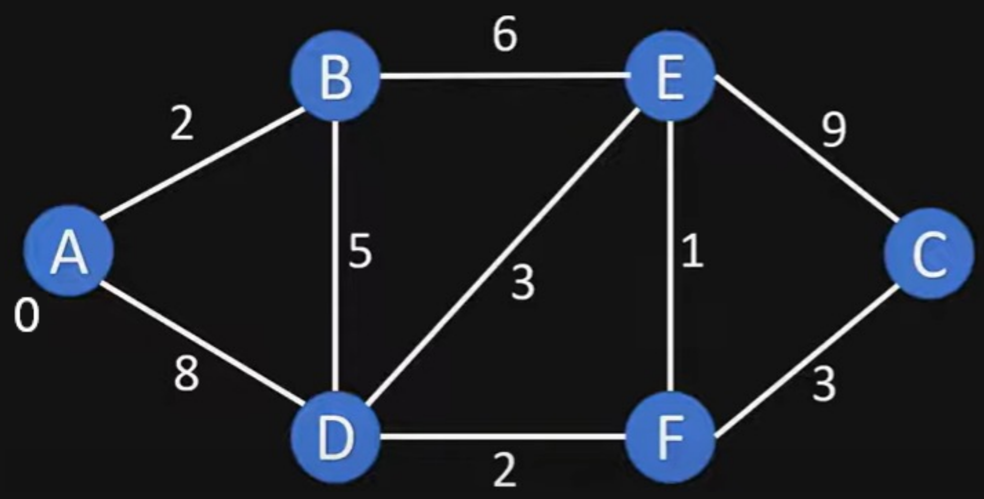
	A	B	C	D	E	F
A	0	2	•	8	•	•
B	2	0	•	5	6	•
C	•	•	0	•	9	3
D	8	5	•	0	3	2
E	•	6	9	3	0	1
F	•	•	3	2	1	0

Een graaf in een matrix



- Voor het uitgereikte Javascript-algoritme ziet dit er vanaf regel 4 als volgt uit:

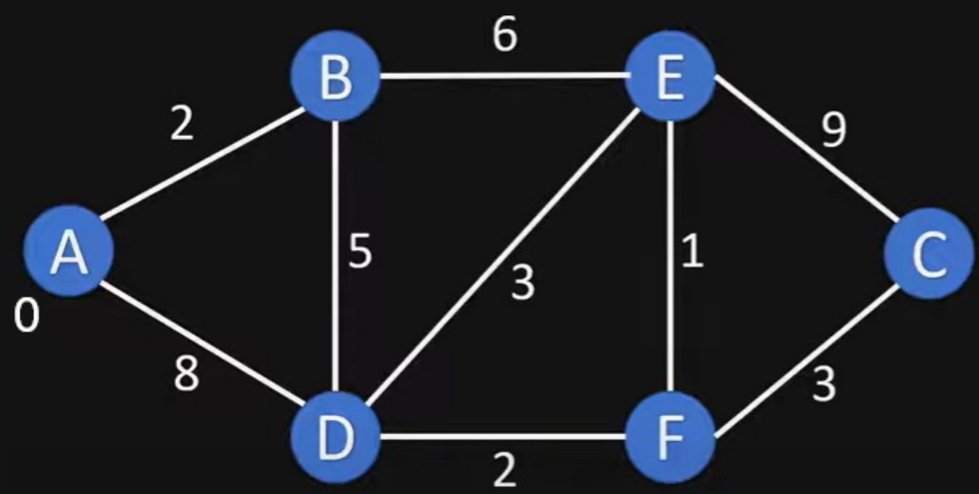
Een graaf in een matrix



- Voor het uitgereikte Javascript-algoritme ziet dit er vanaf regel 4 als volgt uit:

```
const graph = {
        A: {B: 2, D: 8},
        B: {A: 2, D: 5, E: 6},
        C: {E: 9, F: 3},
        D: {A: 8, B: 5, E: 3, F: 2},
        E: {B: 6, C: 9, D: 3, F: 1},
        F: {C: 3, D: 2, E: 1}
};
```

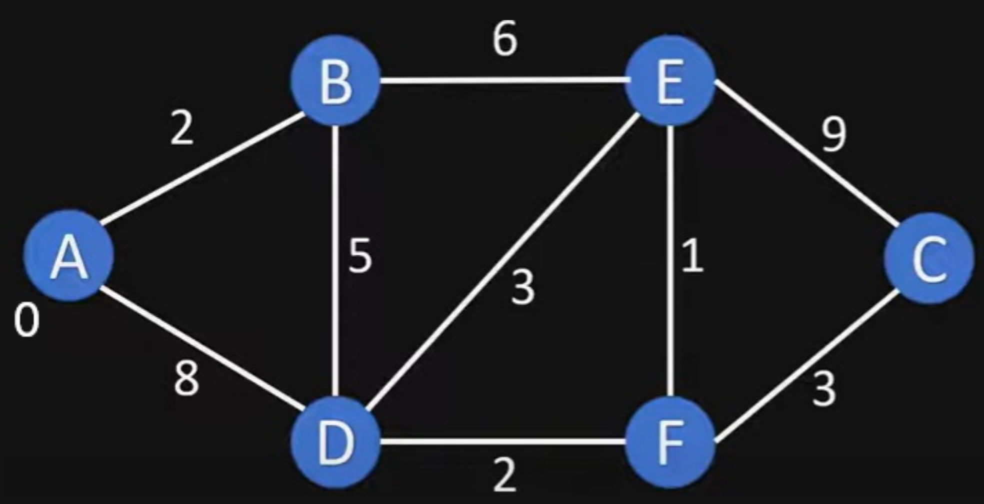
Een graaf in een matrix



- De startknoop wordt bepaald in regel 15 met bijvoorbeeld:

  let firstNode = "A";

Een graaf in een matrix

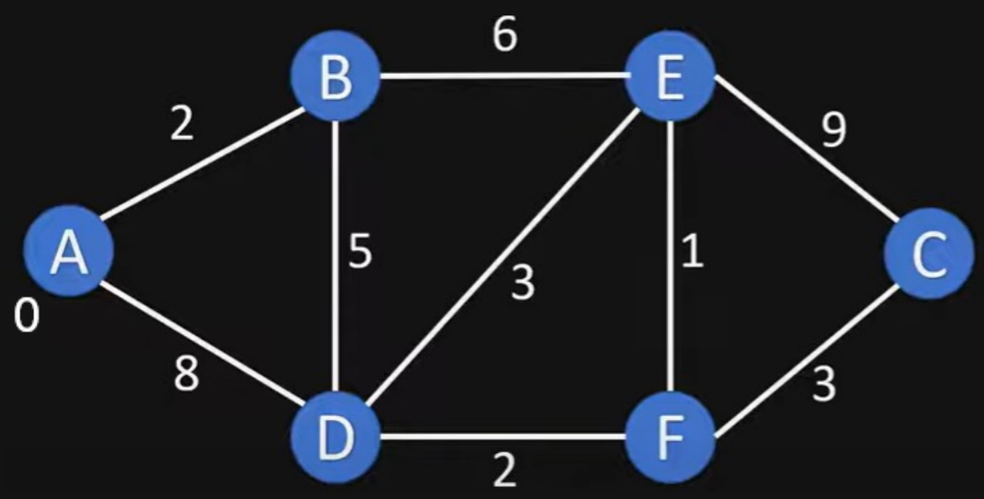


- De startknoop wordt bepaald in regel 15 met bijvoorbeeld:

  `let firstNode = "A";`

- Vervolgens wordt van alle mogelijke (eind-)knopen het kortste pad doorgerekend en bepaald;

Een graaf in een matrix



- De startknoop wordt bepaald in regel 15 met bijvoorbeeld:

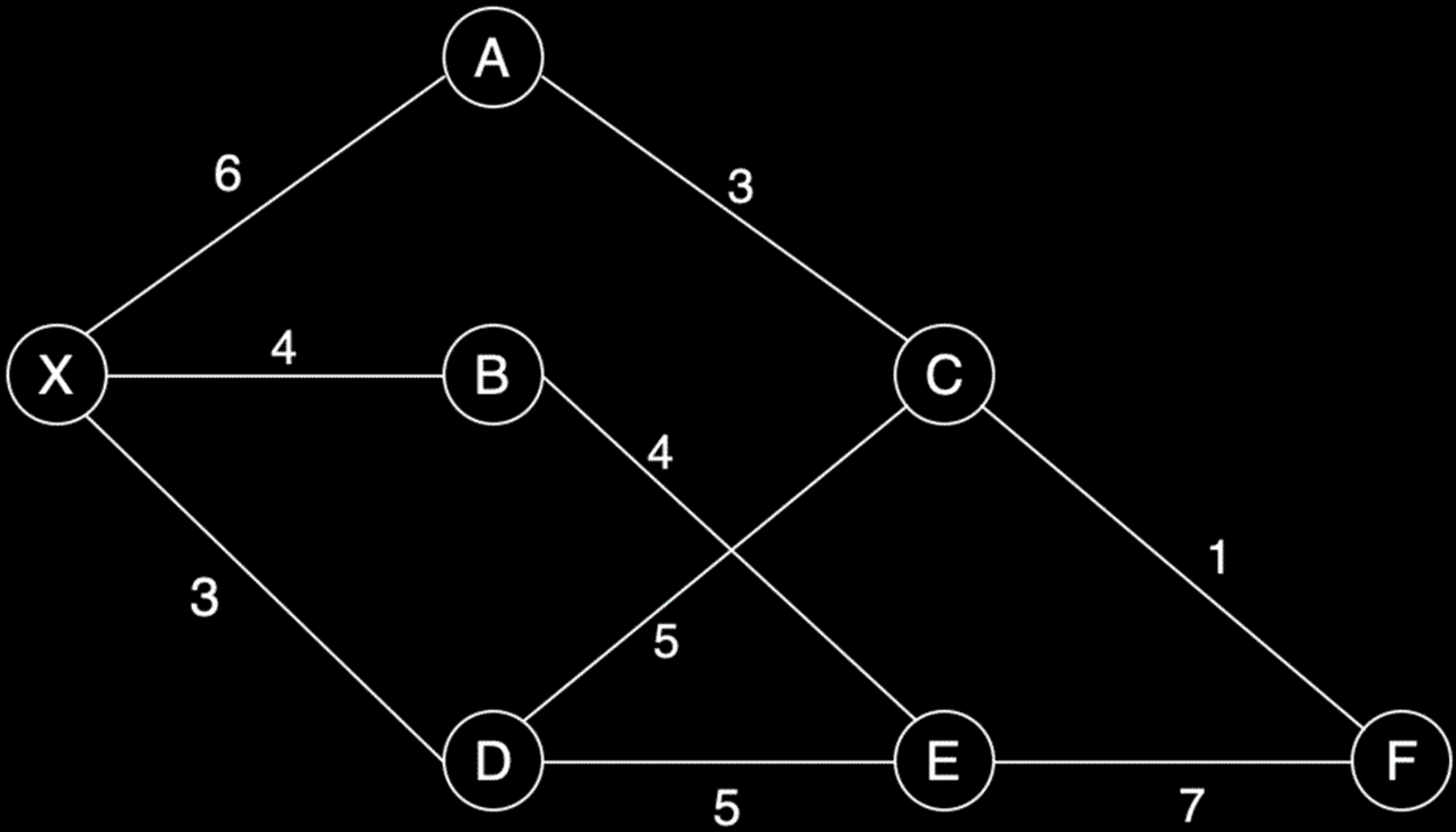
  let firstNode = "A";

- Vervolgens wordt van alle mogelijke (eind-)knopen het kortste pad doorgerekend en bepaald;

- Dit wordt door de uitvoer weergegeven in een overzicht, waaruit dit te herleiden valt.

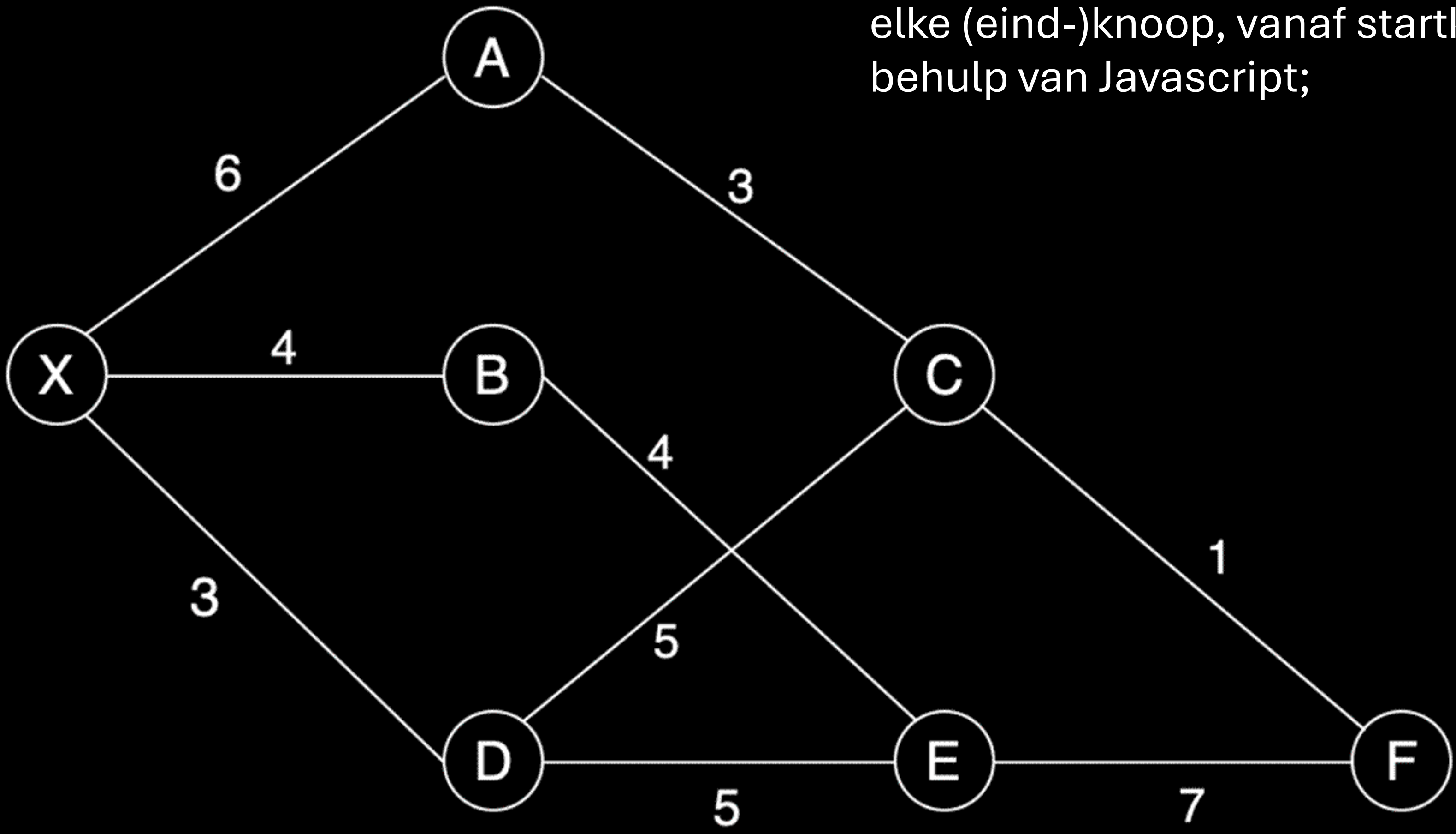
Huiswerk

Huiswerk

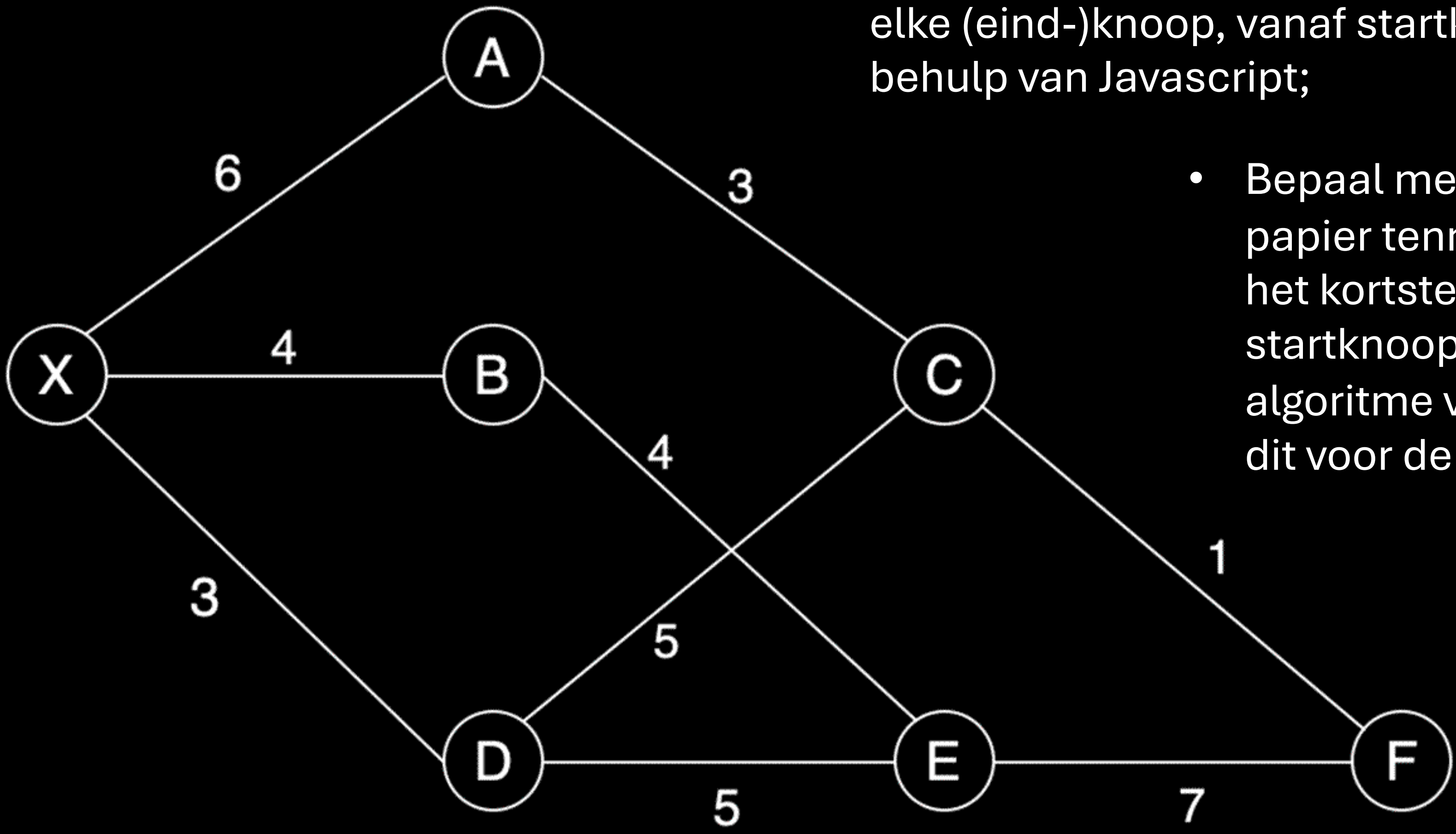


Huiswerk

- Bepaal van deze graaf het kortste pad naar elke (eind-)knoop, vanaf startknoop X, met behulp van Javascript;



Huiswerk



- Bepaal van deze graaf het kortste pad naar elke (eind-)knoop, vanaf startknoop X, met behulp van Javascript;

- Bepaal met gum, potlood en papier tenminste voor knoop E het kortste pad, vanaf startknoop X, volgens het algoritme van Dijkstra, en lever dit voor de volgende keer in.