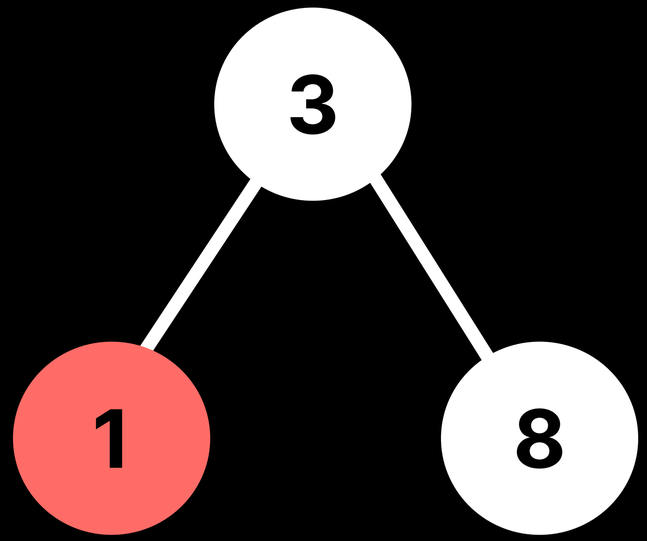
IN2010

Repetisjon

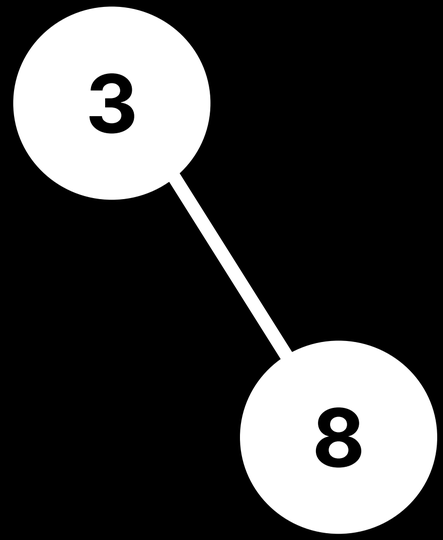
Slette barn i binære søketrær

Slette node uten barn



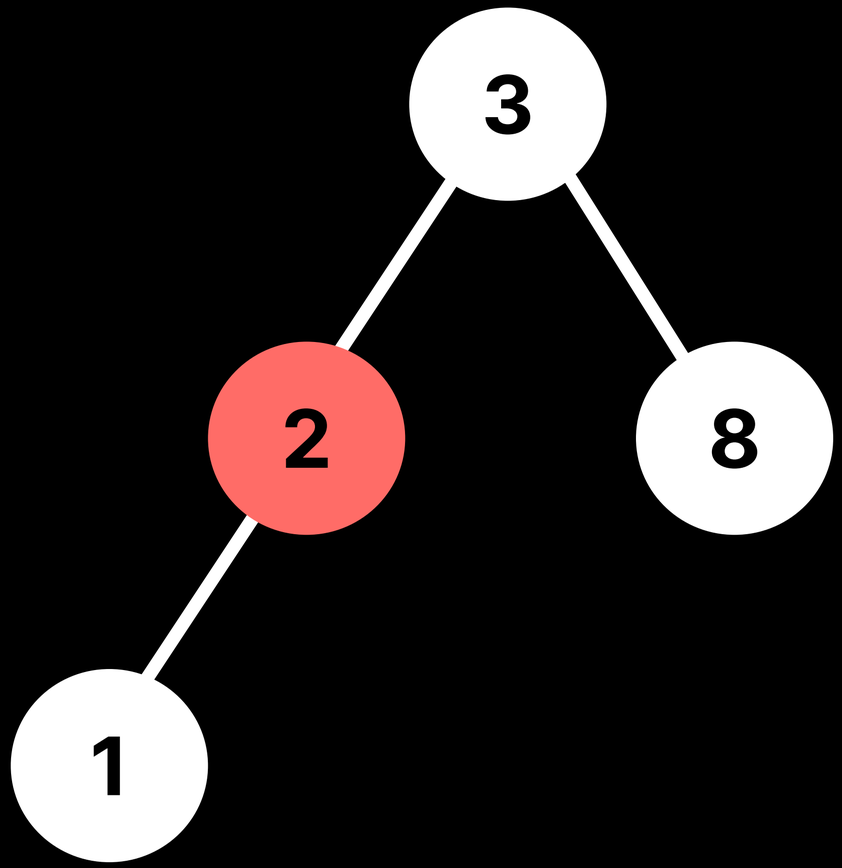
Slette barn i binære søketrær

Slette node uten barn



Slette barn i binære søketrær

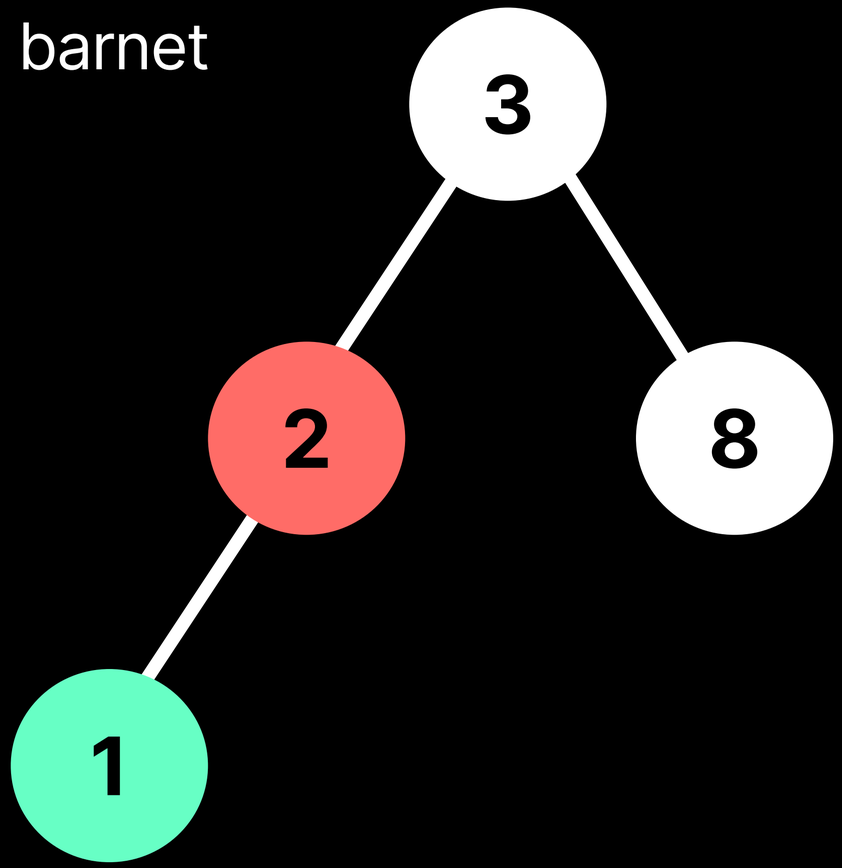
Slette node med ett barn



Slette barn i binære søketrær

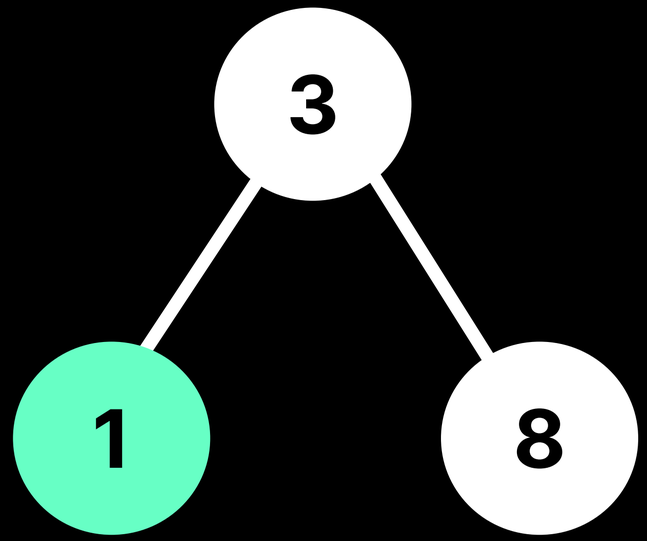
Slette node med ett barn

- Erstatt noden med barnet



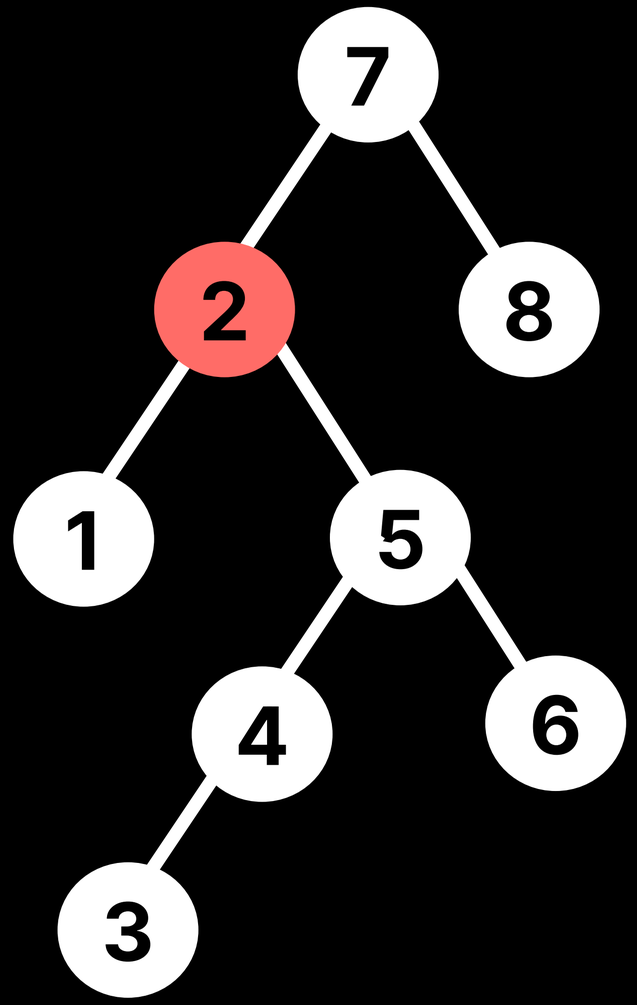
Slette barn i binære søketrær

Slette node med ett barn



Slette barn i binære søketrær

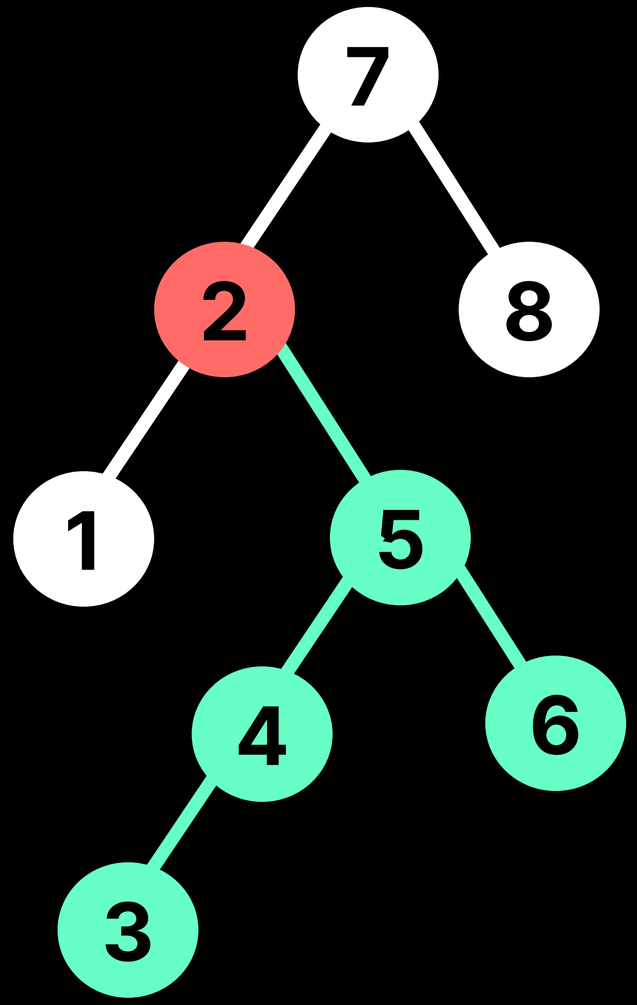
Slette node med ett barn



Slette barn i binære søketrær

Slette node med ett barn

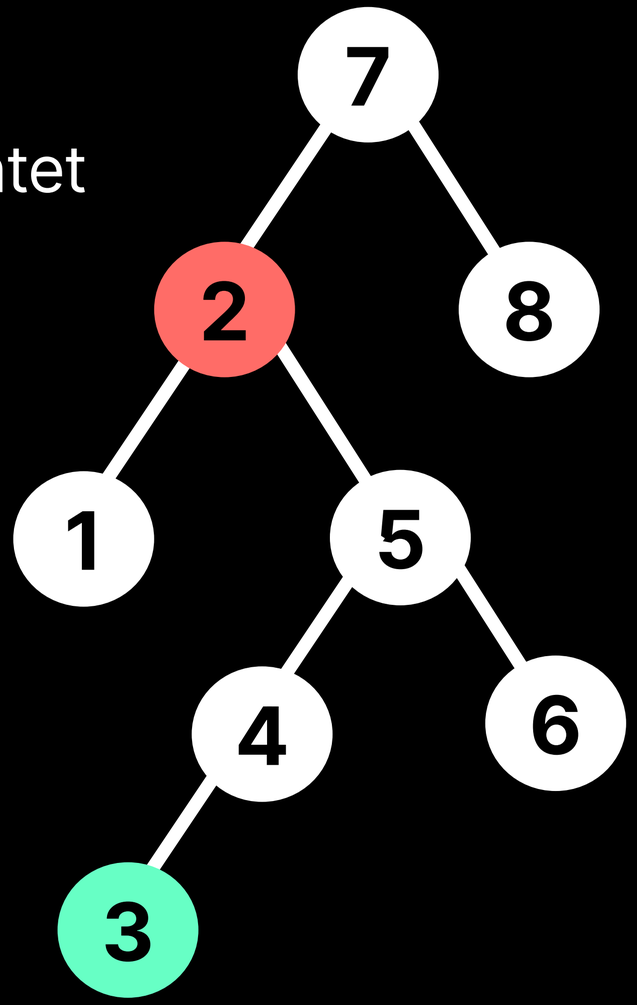
- Se på høyre subtre



Slette barn i binære søketrær

Slette node med ett barn

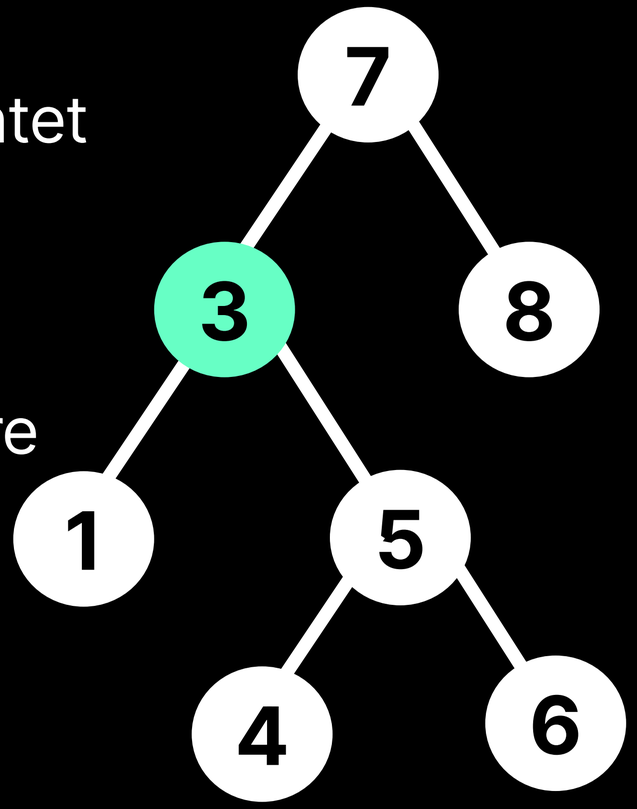
- Se på høyre subtre
- Finn det minste elementet i høyre subtre



Slette barn i binære søketrær

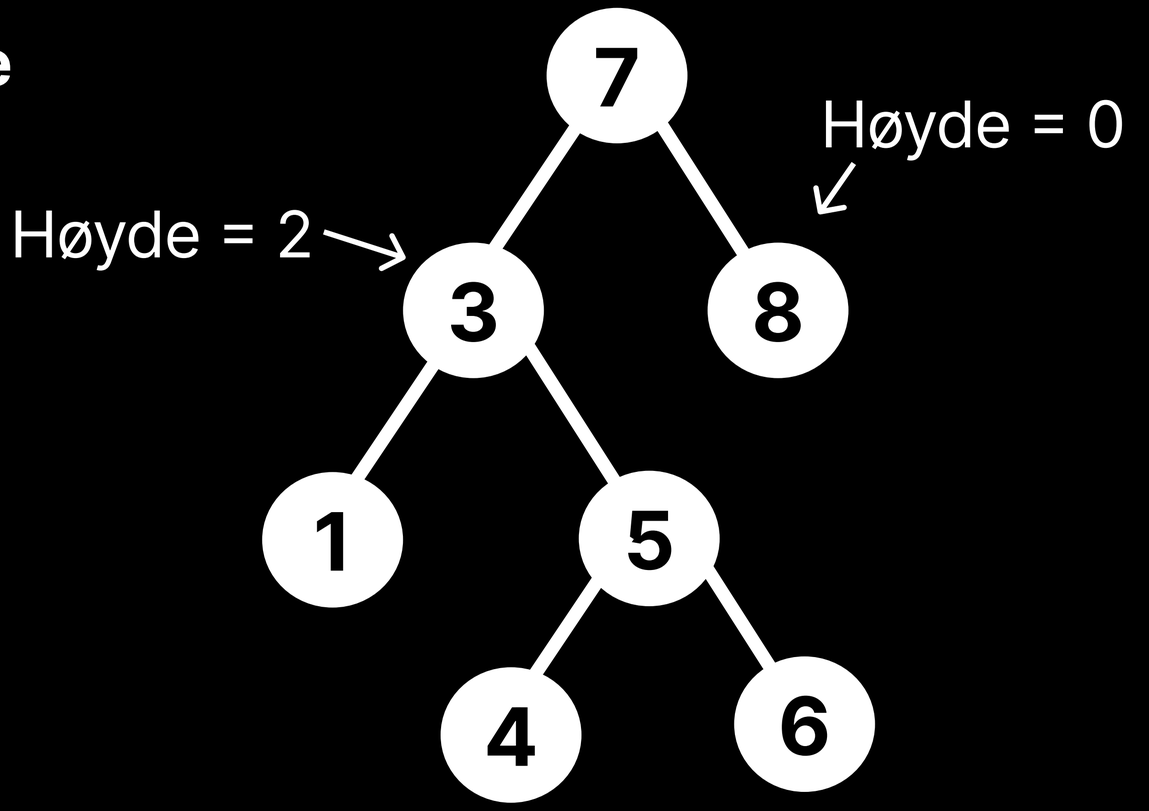
Slette node med ett barn

- Se på høyre subtre
- Finn det minste elementet i høyre subtre
- Erstatt noden du ville slette med det minste elementet i høyre subtre

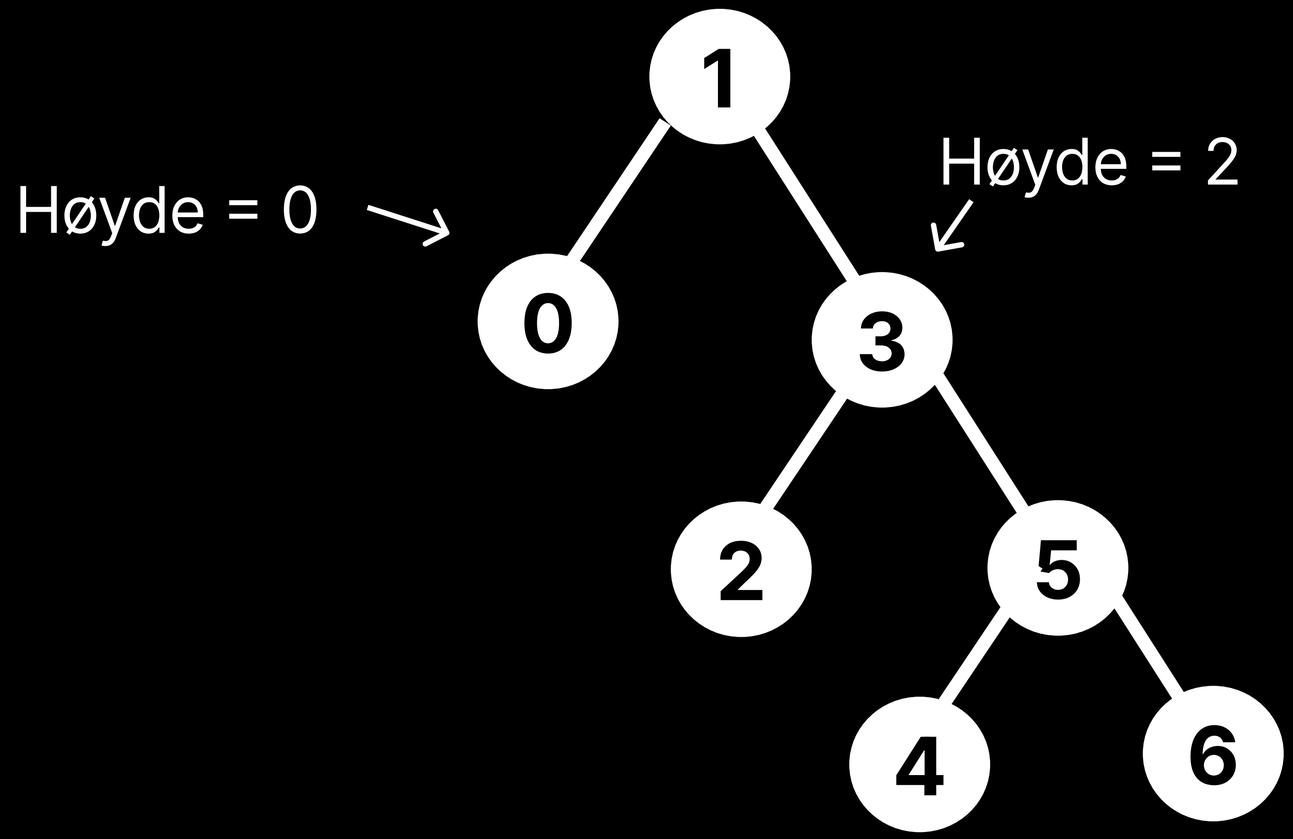


Rotasjoner

Rotasjoner skjer i AVL trær når høydeforskjellen mellom venstre og høyre subtre er større enn 1

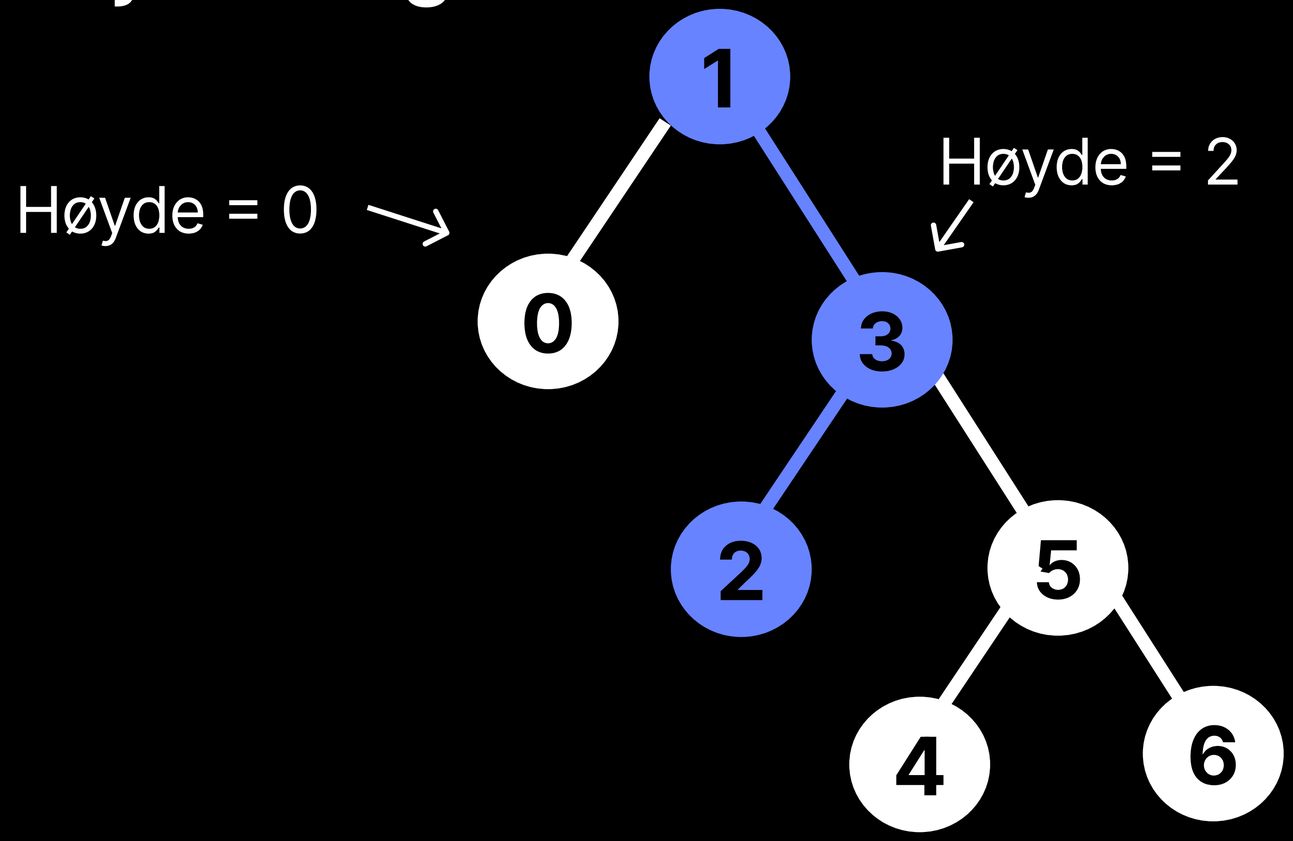


Venstrerotasjon



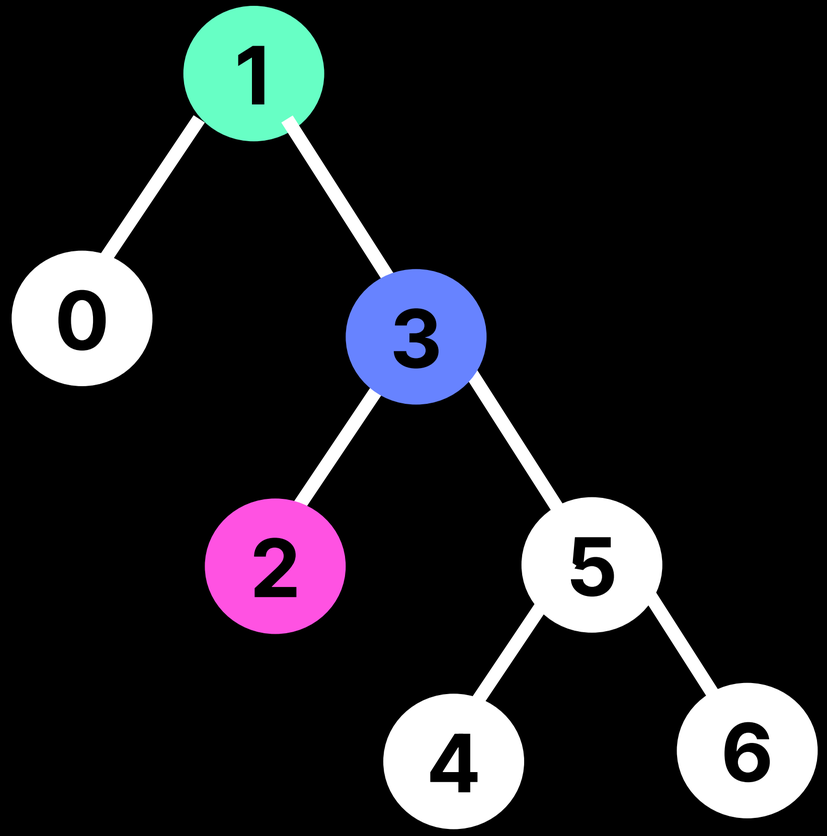
Venstrerotasjon

Kjennetegnet:



Venstrerotasjon

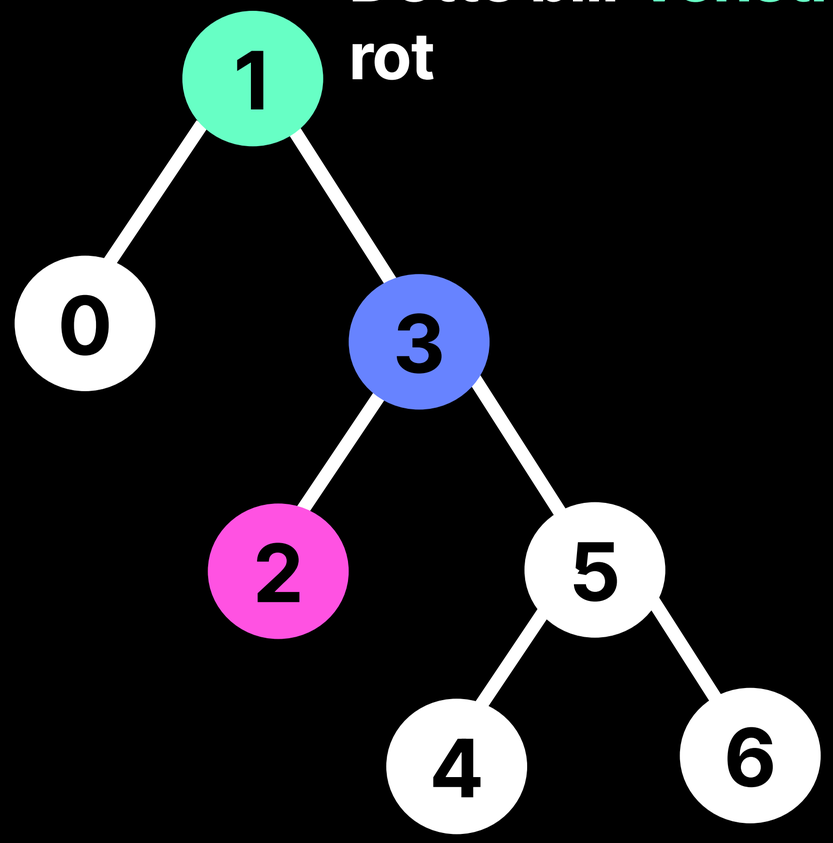
Det er disse tre nodene vi fokuserer på



Venstrerotasjon

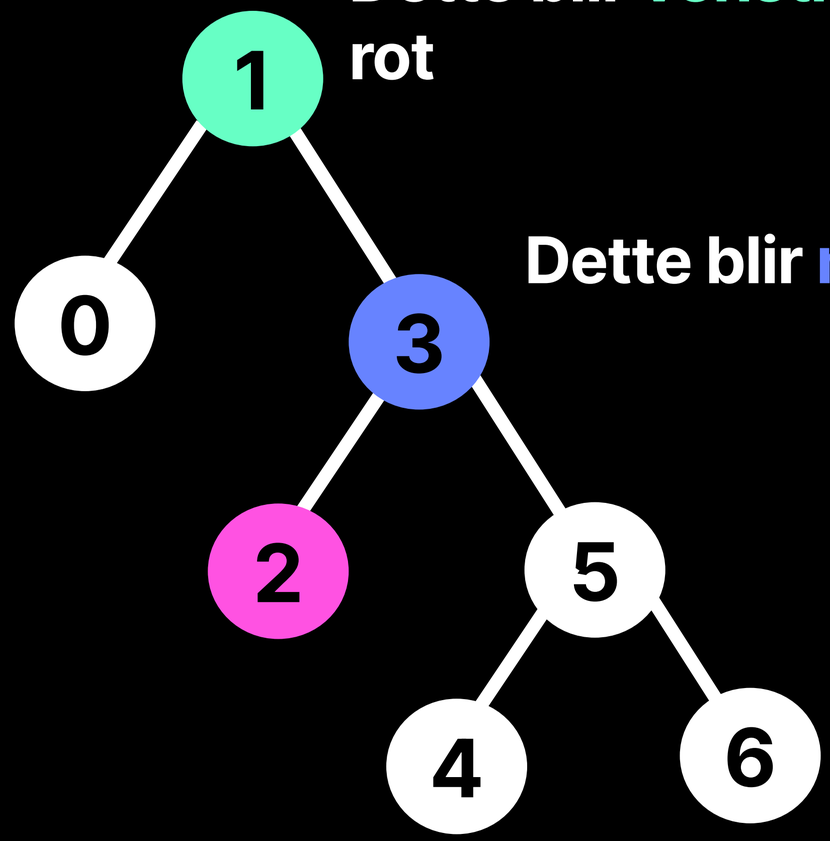
Det er disse tre nodene vi fokuserer på

Dette blir venstre barn til ny rot



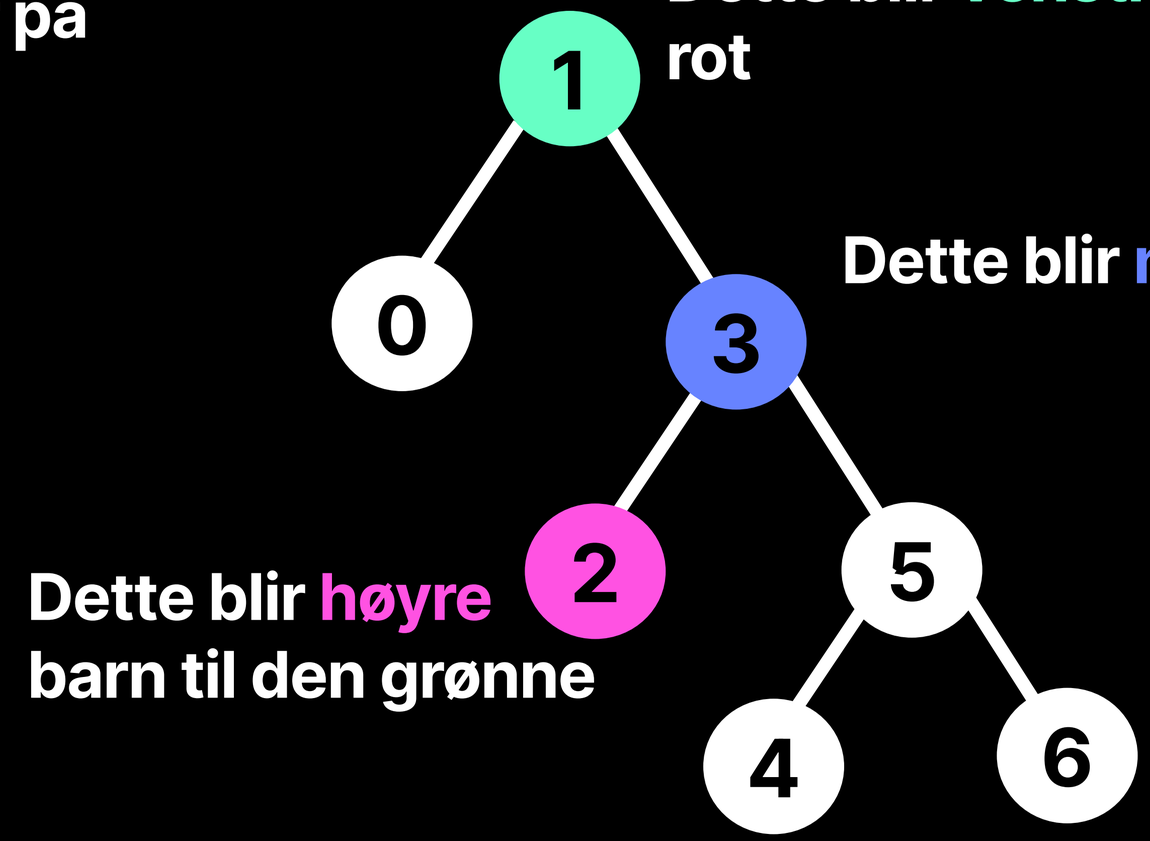
Venstrerotasjon

Det er disse tre nodene vi fokuserer på



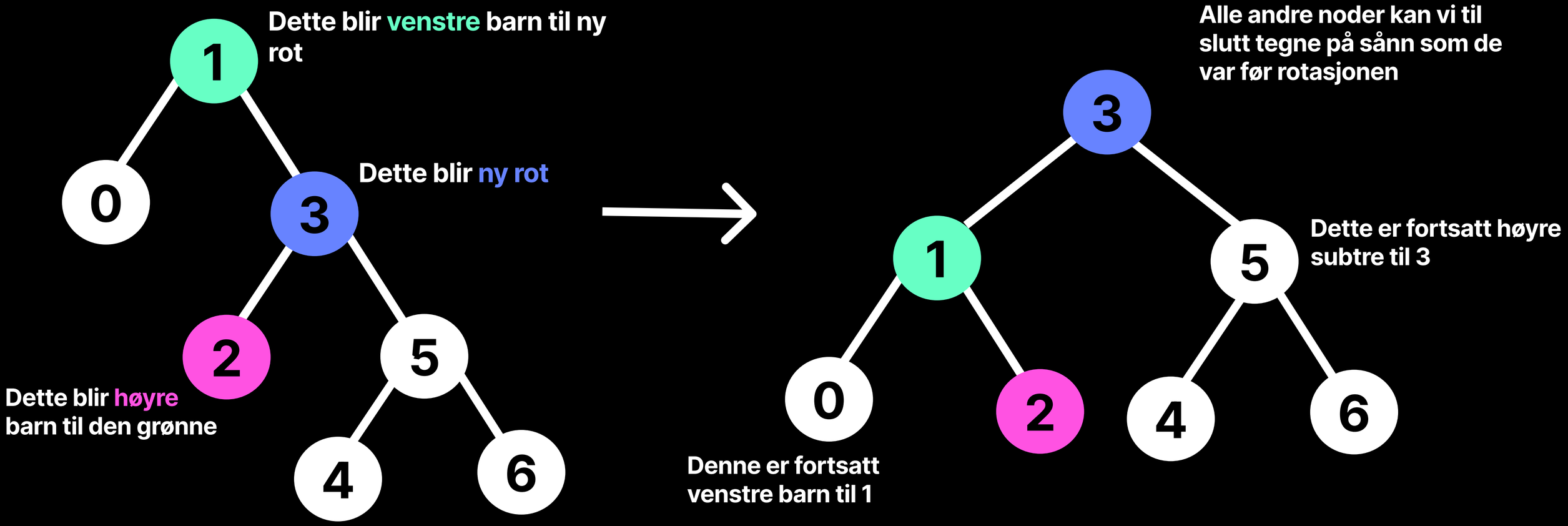
Venstrerotasjon

Det er disse tre nodene vi fokuserer på



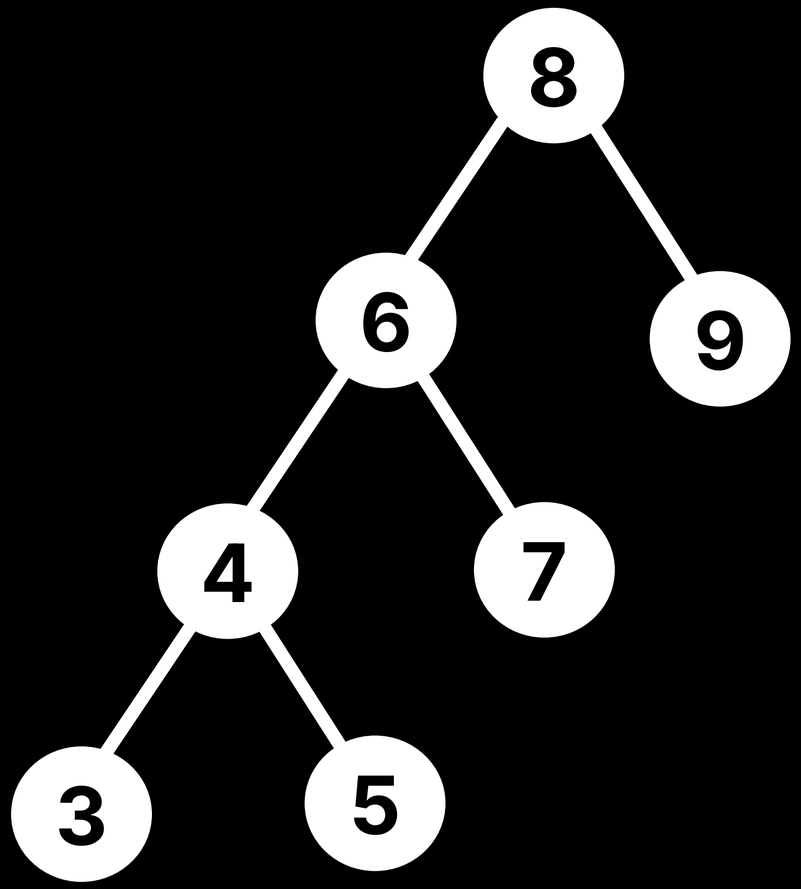
Venstrerotasjon

Det er disse tre nodene vi fokuserer på



Høyrerotasjon

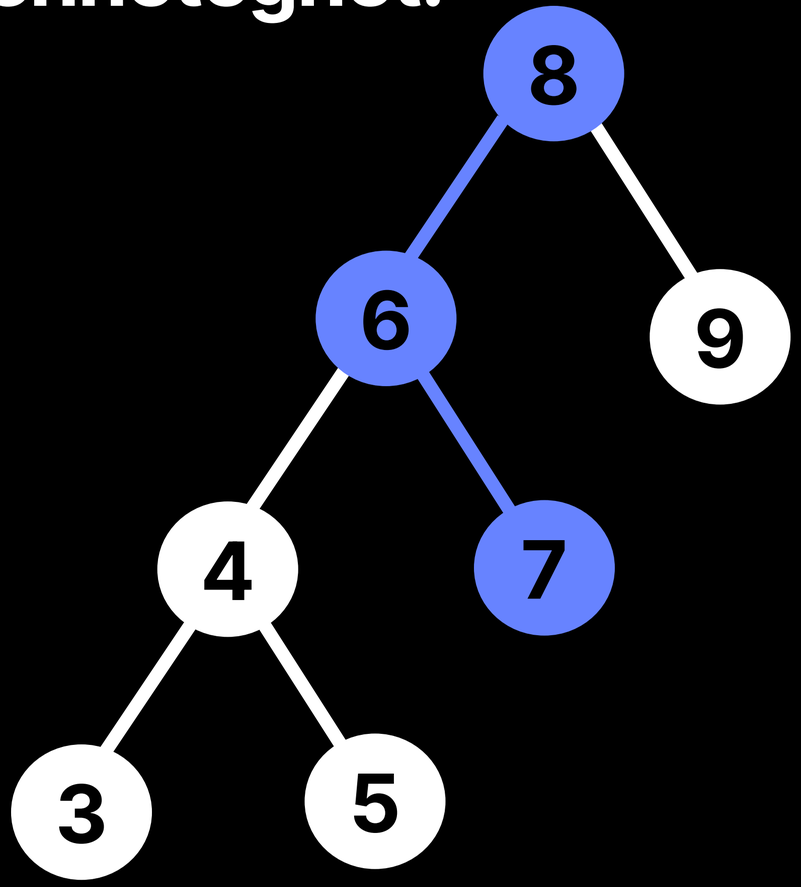
Speilvendt av venstrerotasjon!



Høyrerotasjon

Speilvendt av venstrerotasjon!

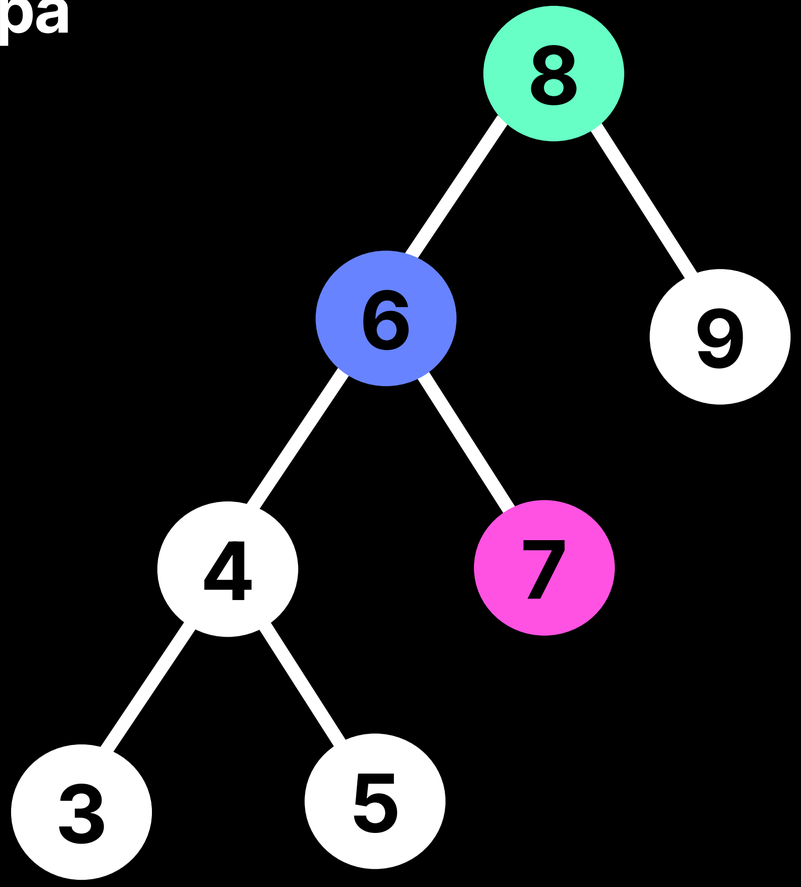
Kjennetegnet:



Høyrerotasjon

Speilvendt av venstrerotasjon!

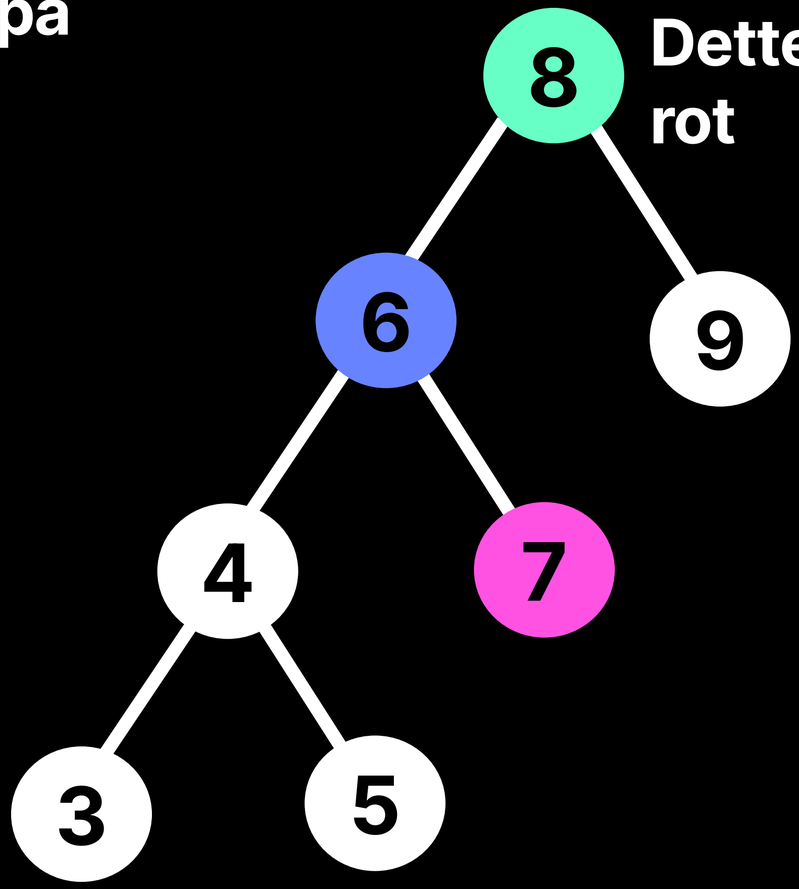
Det er disse tre nodene vi fokuserer på



Høyrerotasjon

Speilvendt av venstrerotasjon!

Det er disse tre nodene vi fokuserer på

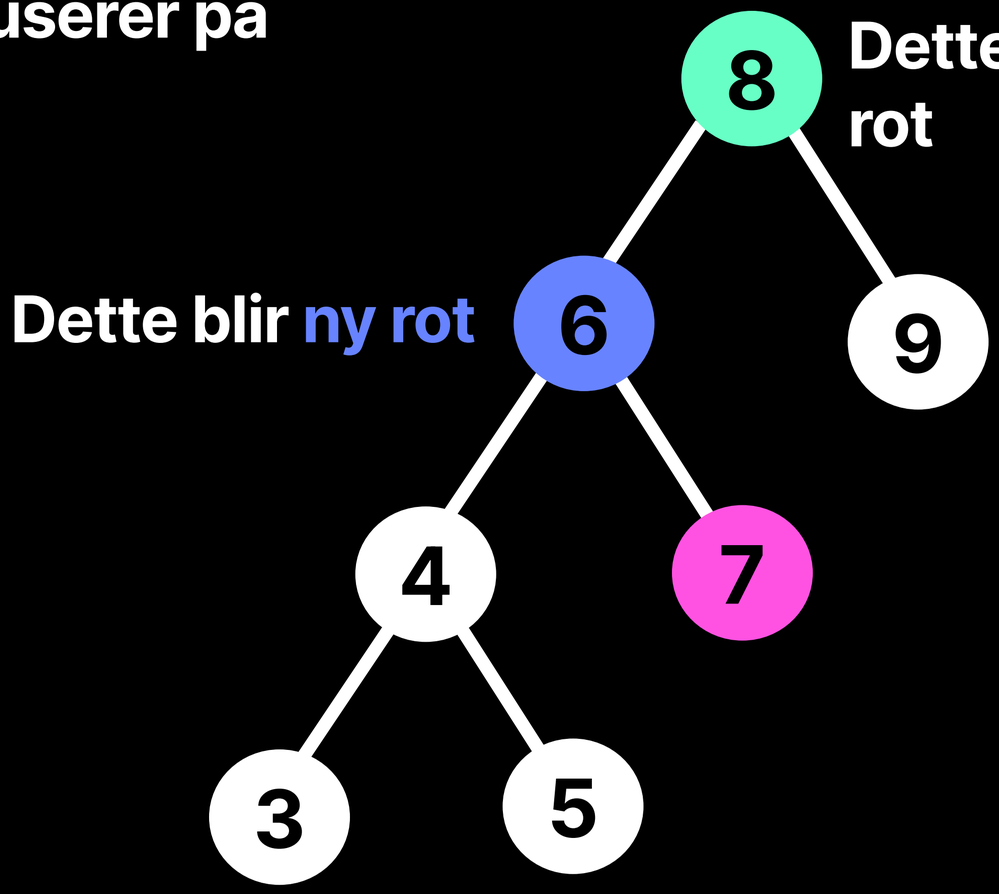


Dette blir høyre barn til ny rot

Høyrerotasjon

Speilvendt av venstrerotasjon!

Det er disse tre nodene vi fokuserer på



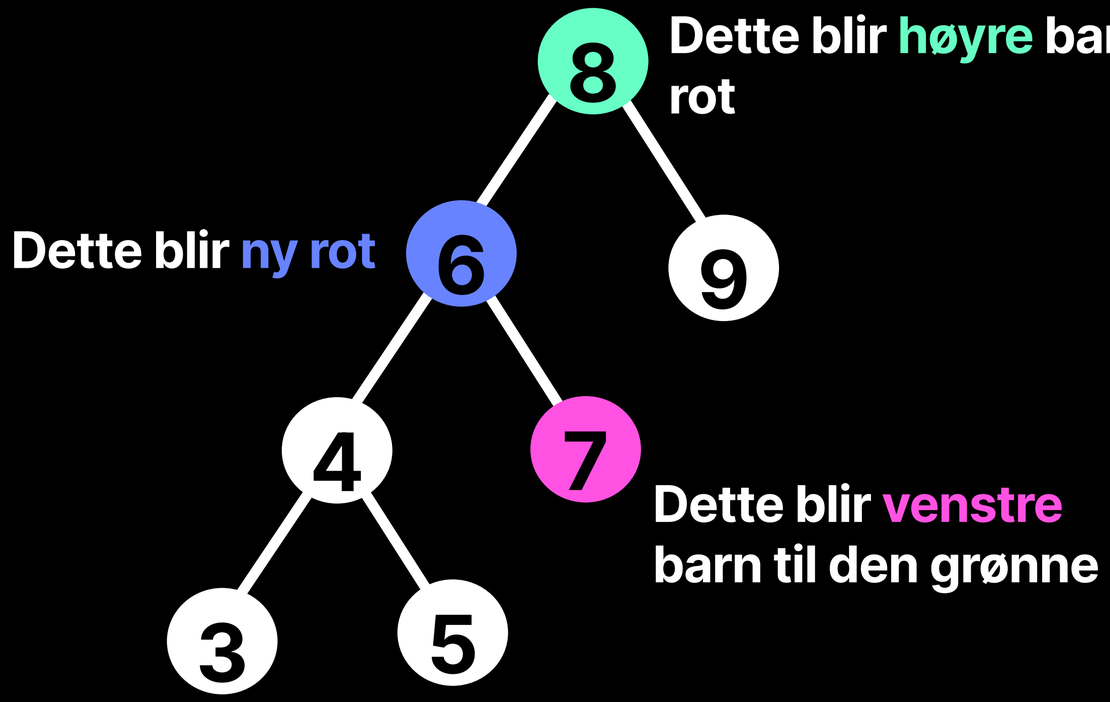
Dette blir høyre barn til ny rot

Dette blir ny rot

Høyrerotasjon

Speilvendt av venstrerotasjon!

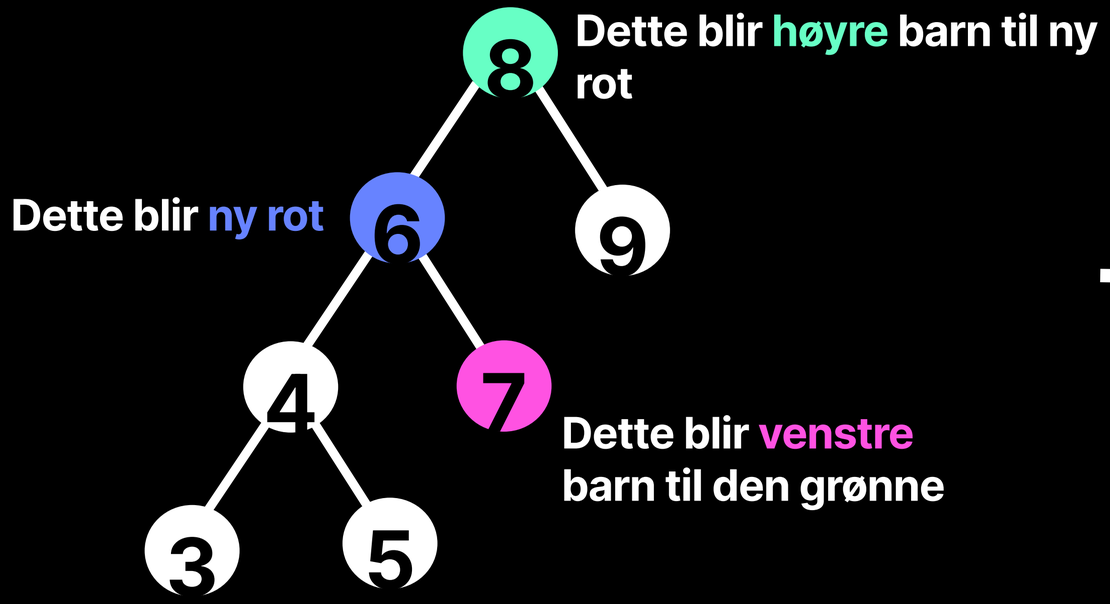
Det er disse tre nodene vi fokuserer på

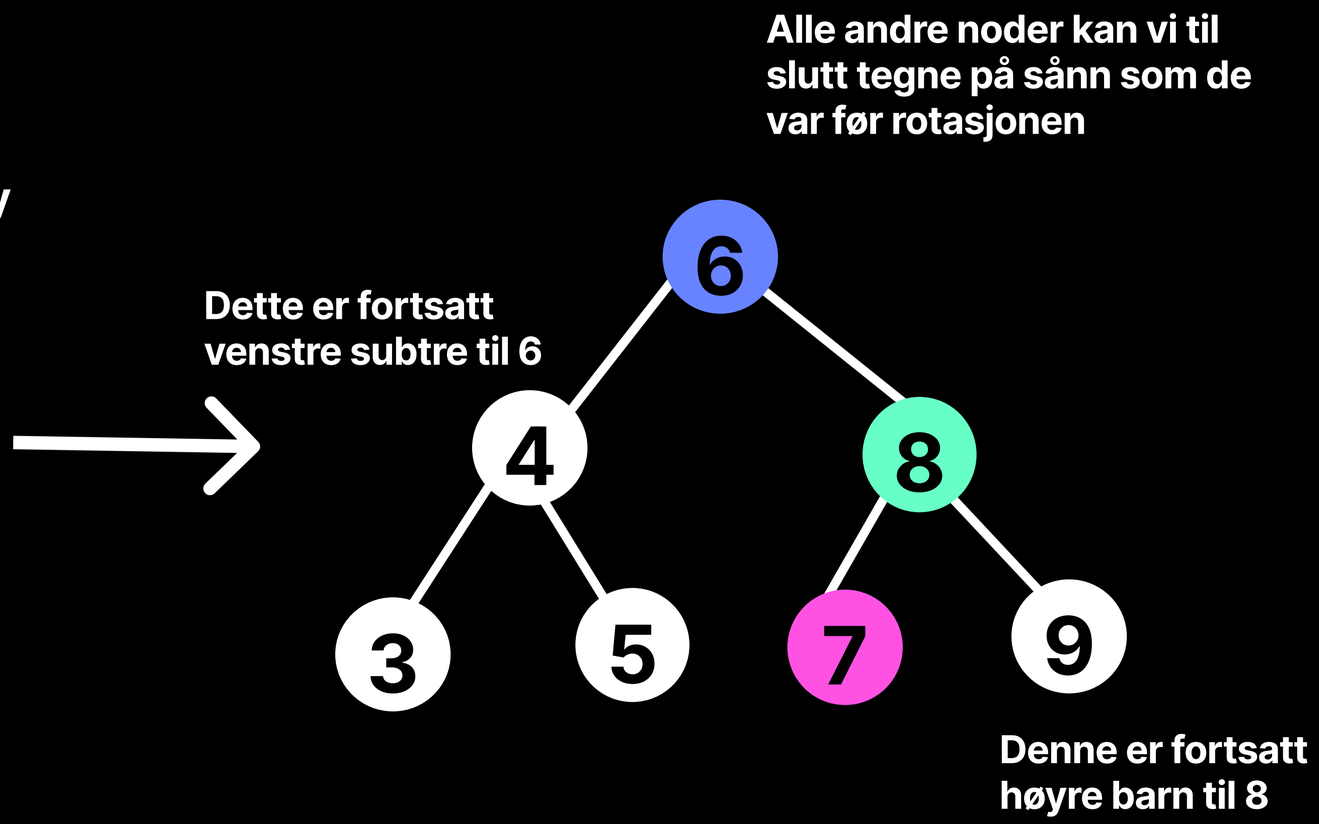


Høyrerotasjon

Speilvendt av venstrerotasjon!

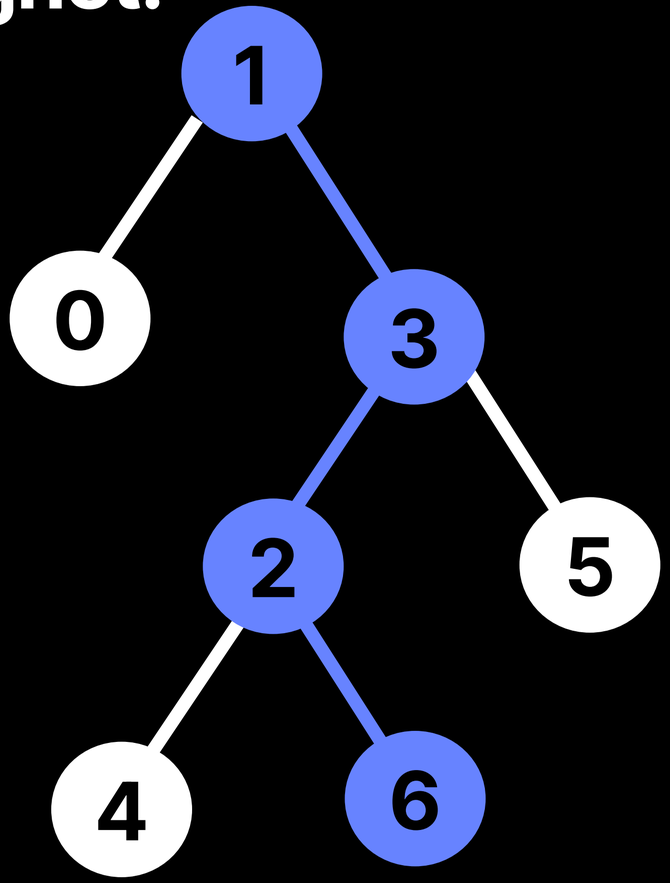
Det er disse tre nodene vi fokuserer på





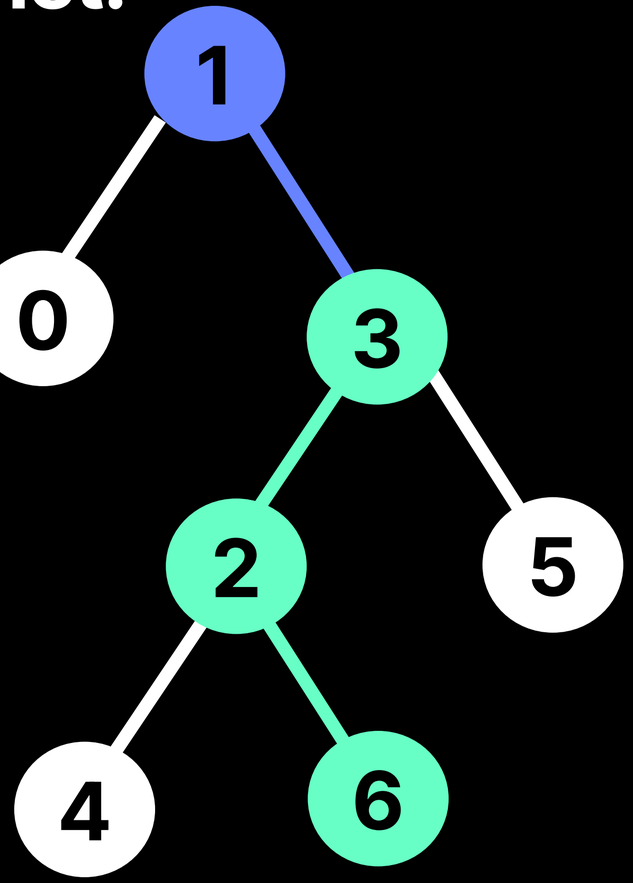
Doble rotasjoner

Kjennetegnet:



Doble rotasjoner

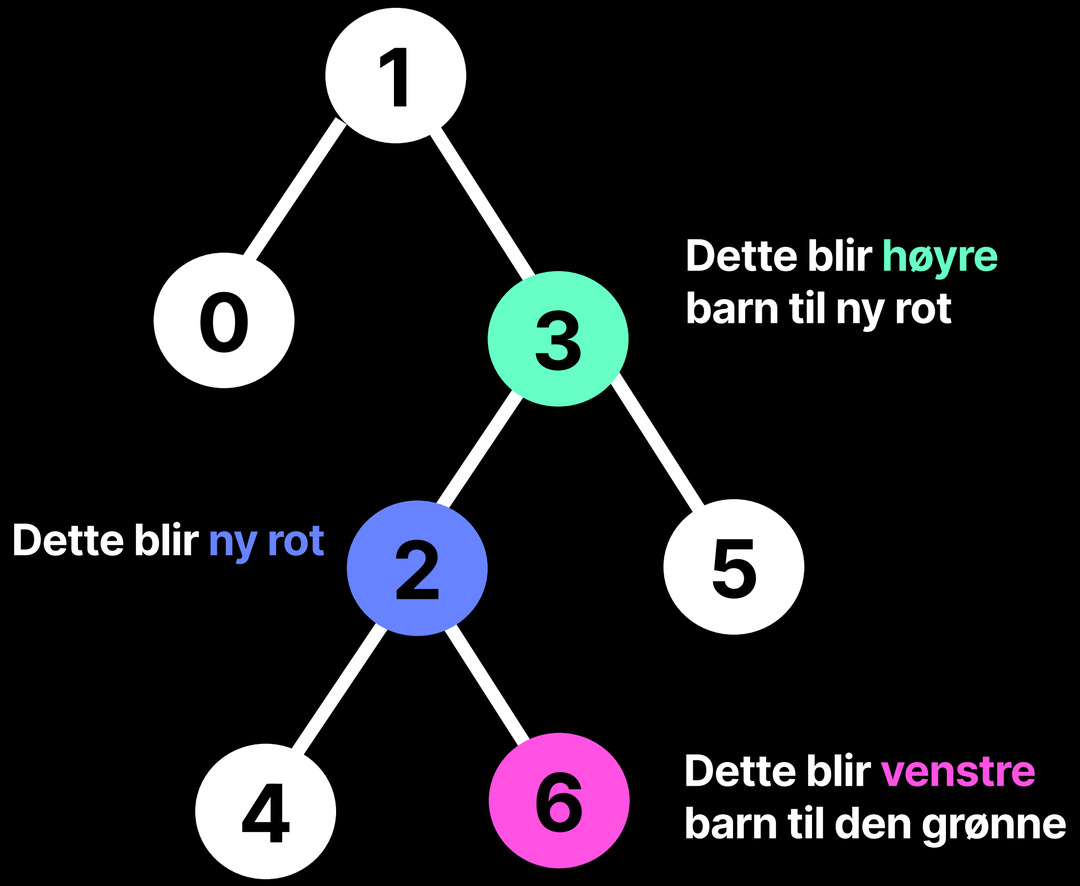
Kjennetegnet:



Kjennetegnet på høyre rotasjon er det lavest nede, så vi starter med en høyre rotasjon!

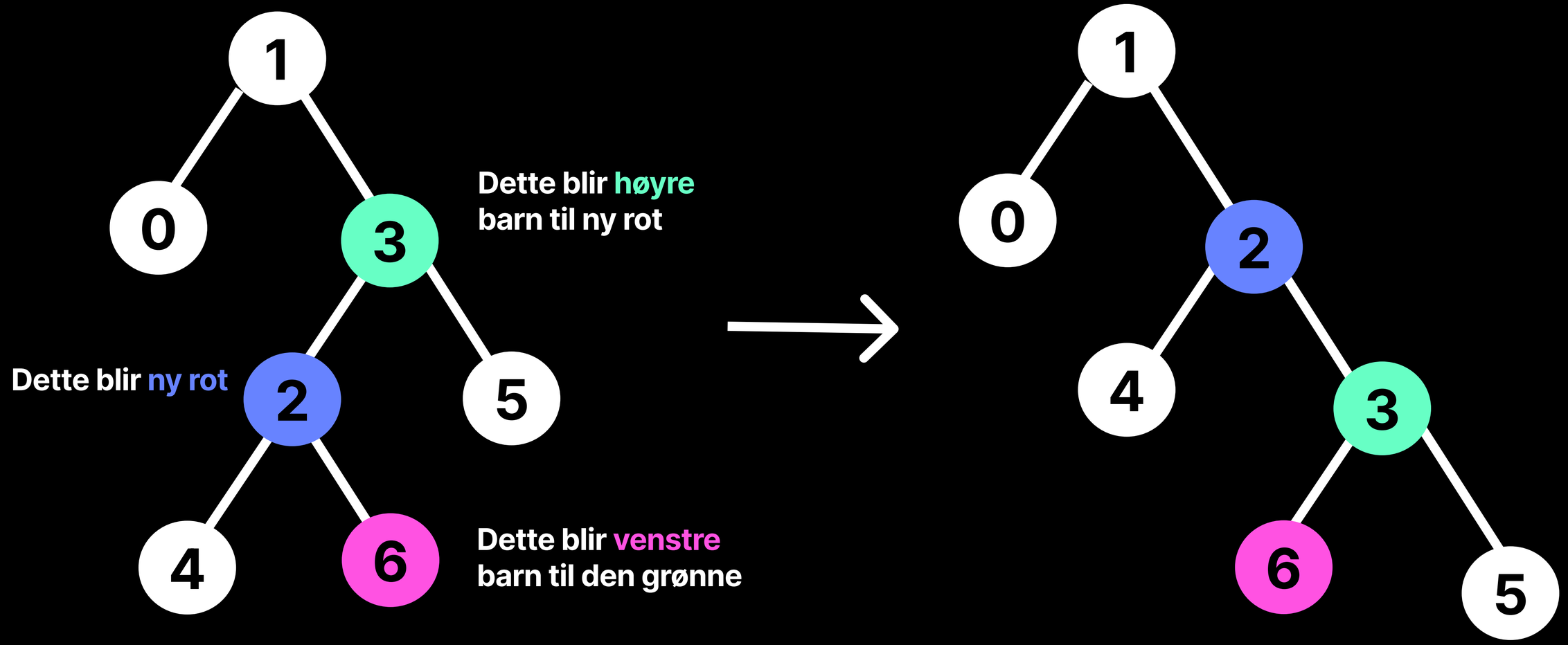
Doble rotasjoner

Starter med en høyrerotasjon



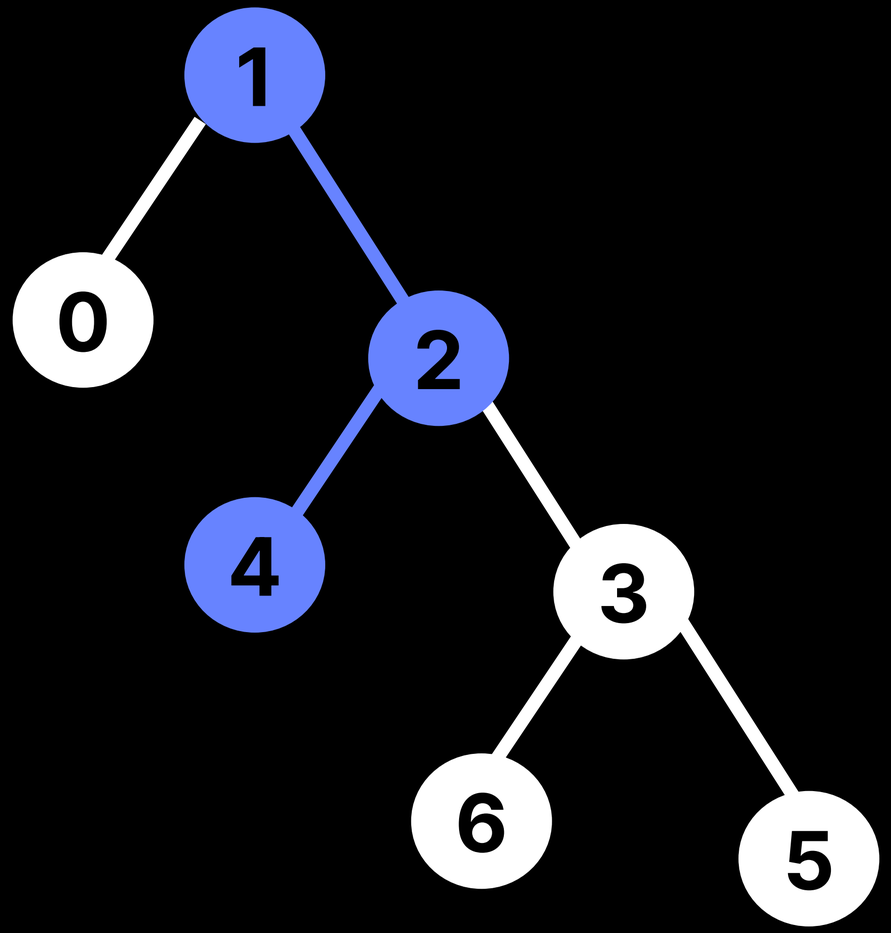
Doble rotasjoner

Starter med en høyrerotasjon



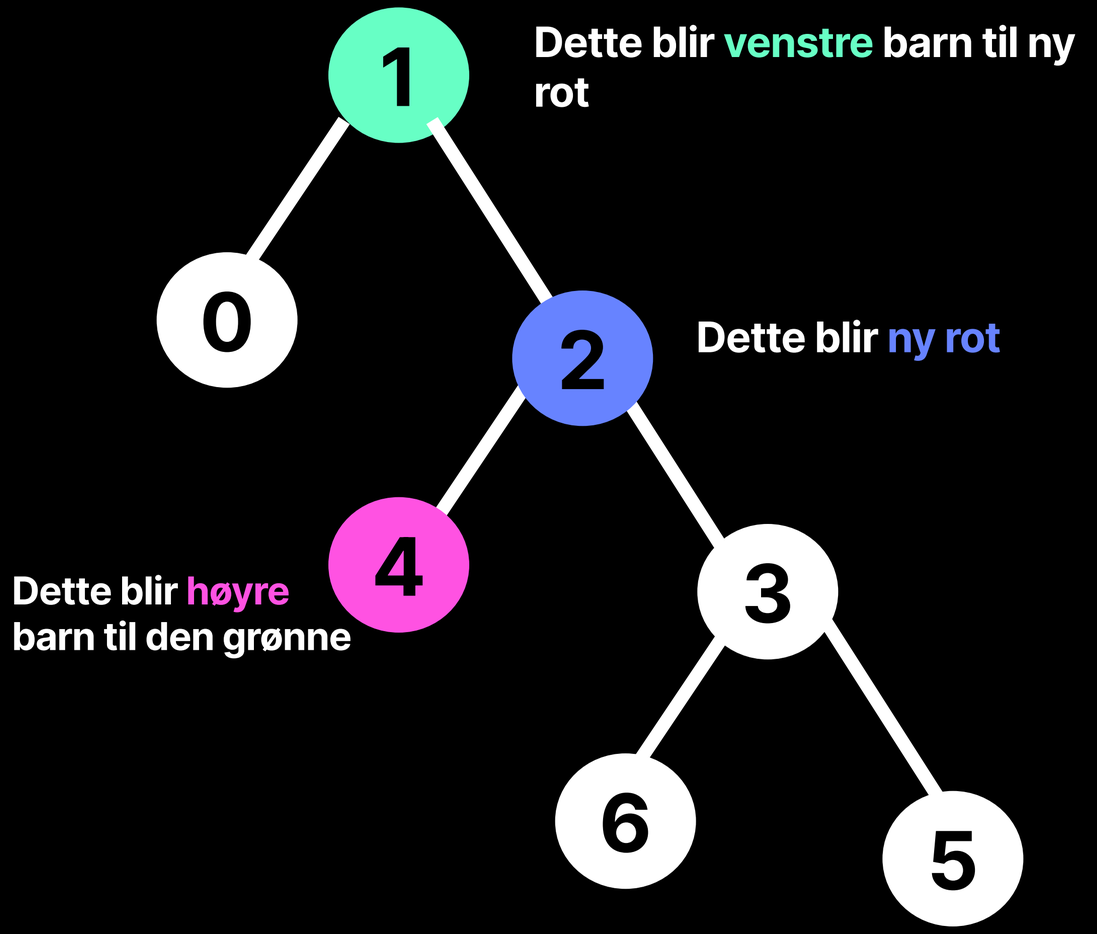
Doble rotasjoner

Nå ser vi kjennetegnet på en venstrerotasjon!



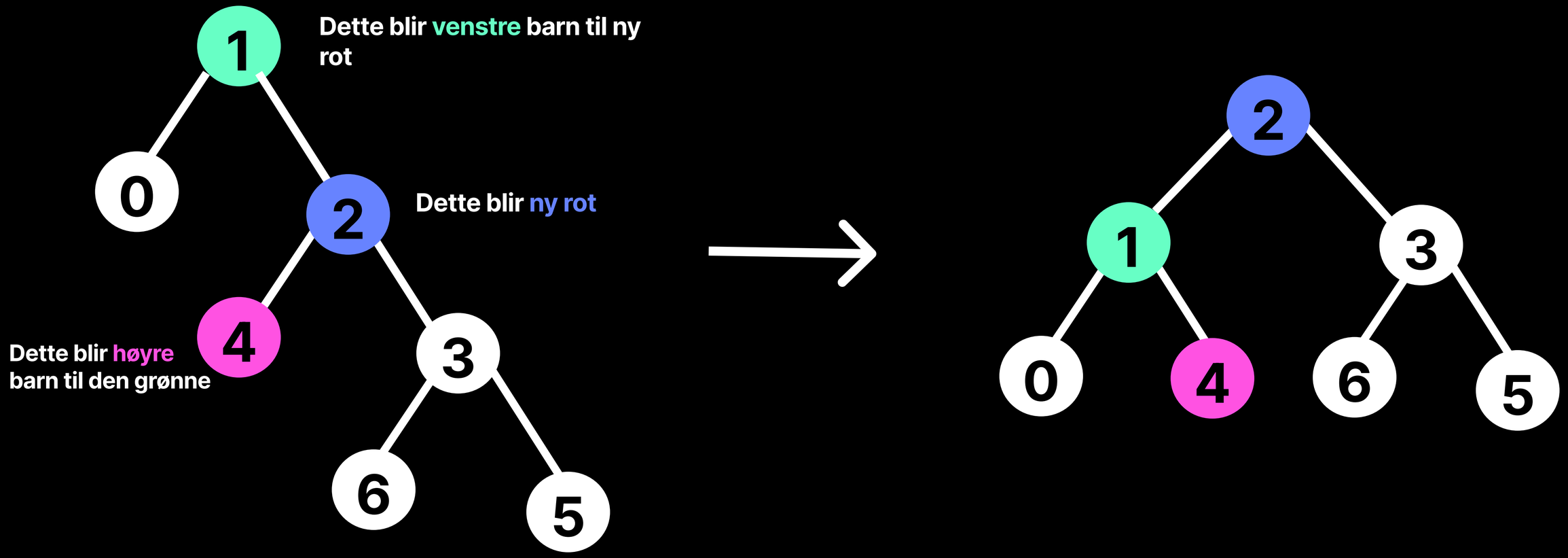
Doble rotasjoner

Vi gjør en venstrerotasjon



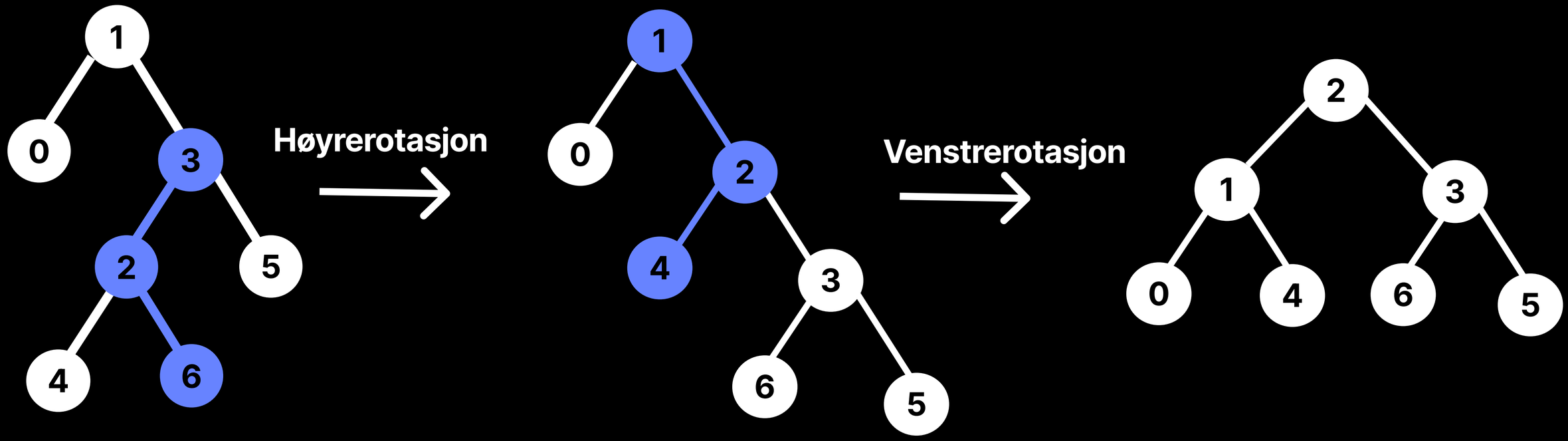
Doble rotasjoner

Vi gjør en venstrerotasjon



Doble rotasjoner

Hele operasjonen:



Komplette binær trær

Disse er komplette

Bare en rot



Komplette binær trær

- **Alle nivåer utenom siste skal være fulle.** 2^n noder i nivå n. Obs: starter på 0, så rotnivå er nivå 0.

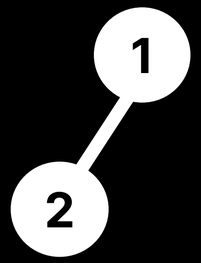
- Siste nivå er **fylt fra venstre uten hull.**

Komplette binær trær

Disse er komplette

Bare en rot



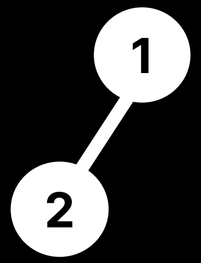


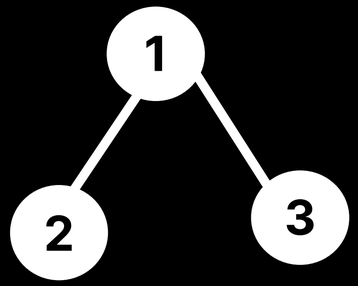
Komplette binær trær

Disse er komplette

Bare en rot





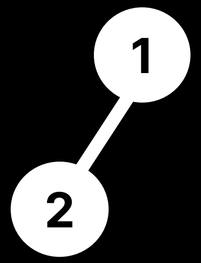


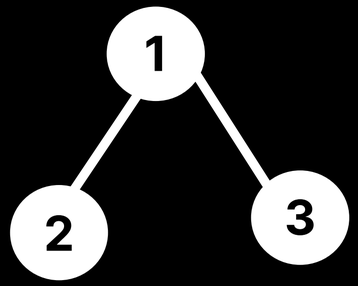
Komplette binær trær

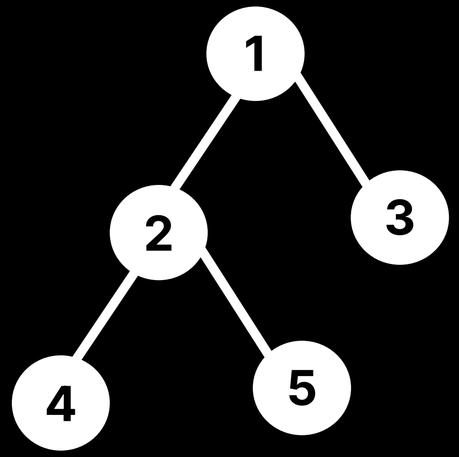
Disse er komplette

Bare en rot







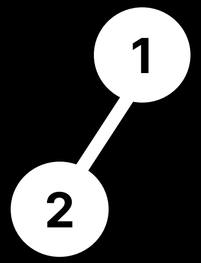


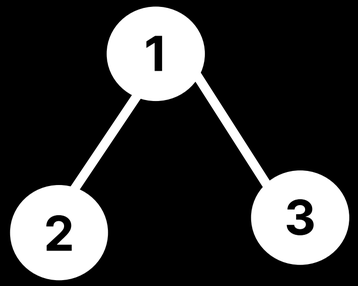
Komplette binær trær

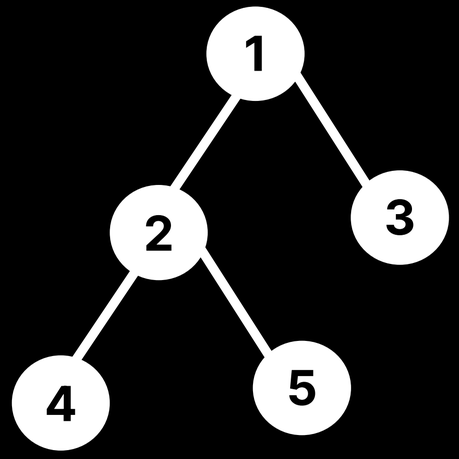
Disse er komplette

Bare en rot

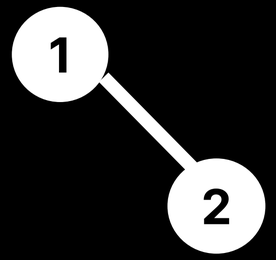








Disse er IKKE komplette

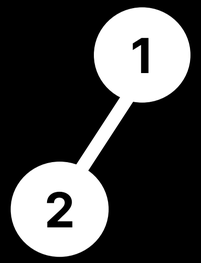


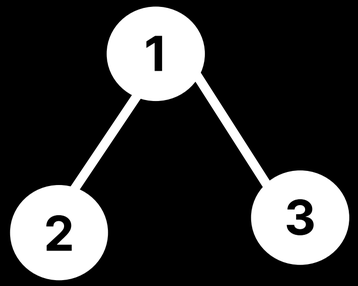
Komplette binær trær

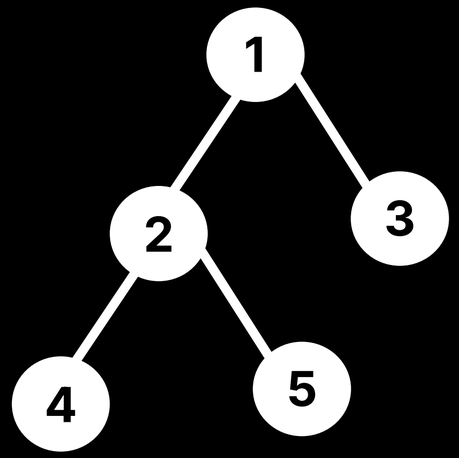
Disse er komplette

Bare en rot



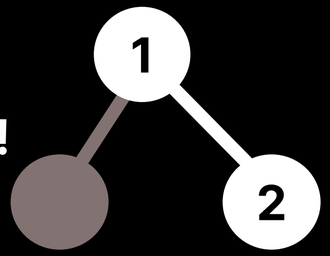






Disse er IKKE komplette

Hull!

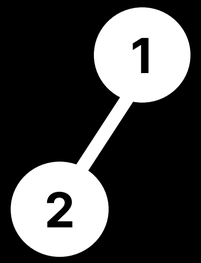


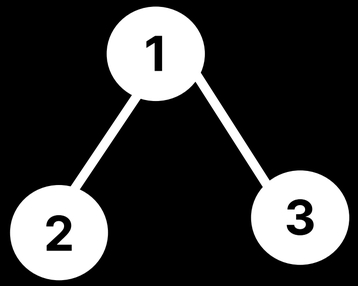
Komplette binær trær

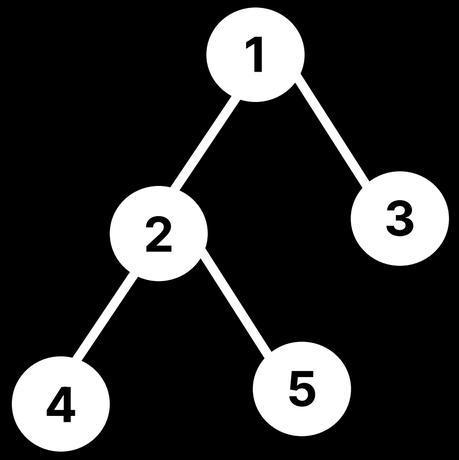
Disse er komplette

Bare en rot



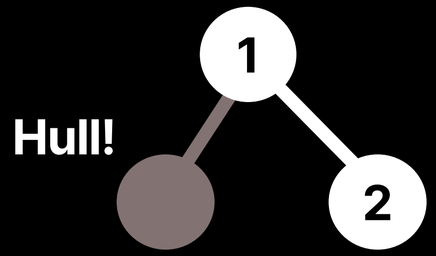


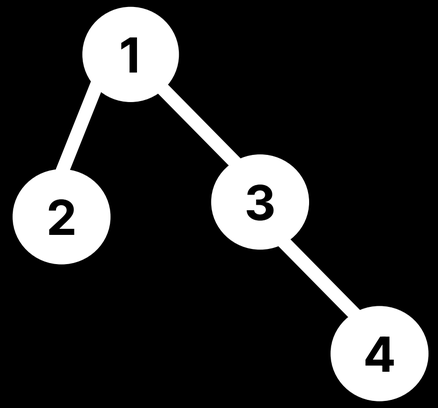




Disse er IKKE komplette

Hull!



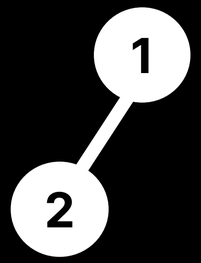


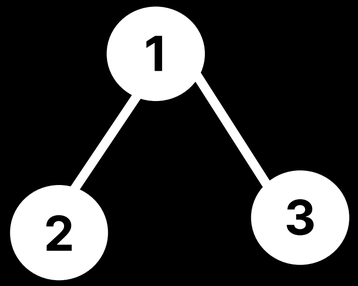
Komplette binær trær

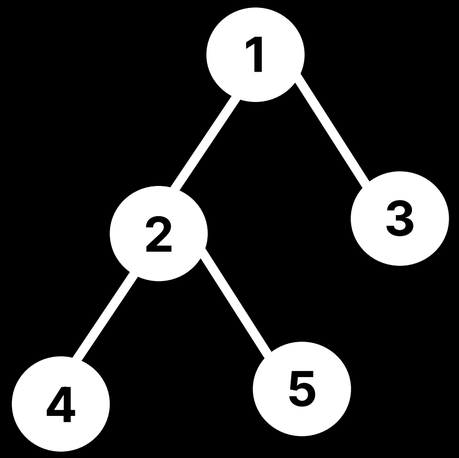
Disse er komplette

Bare en rot

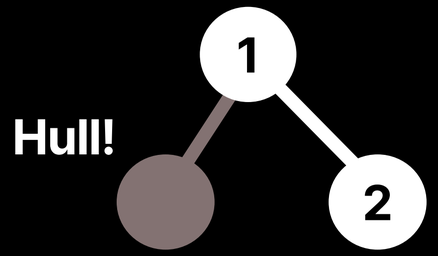


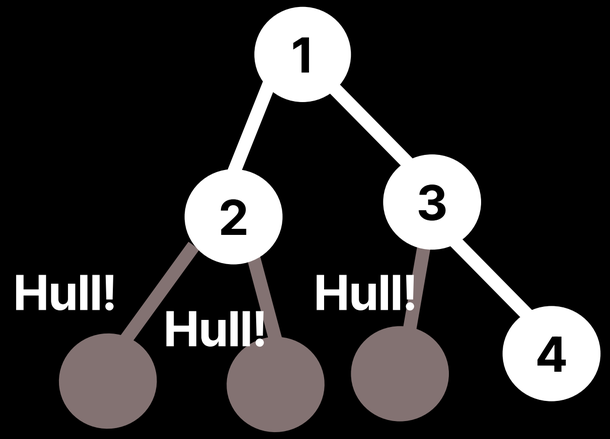






Disse er IKKE komplette



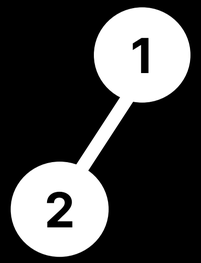


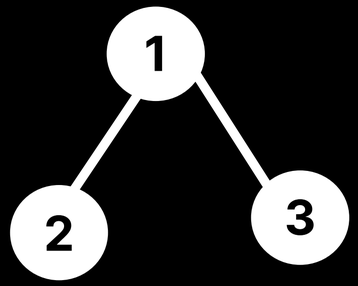
Komplette binær trær

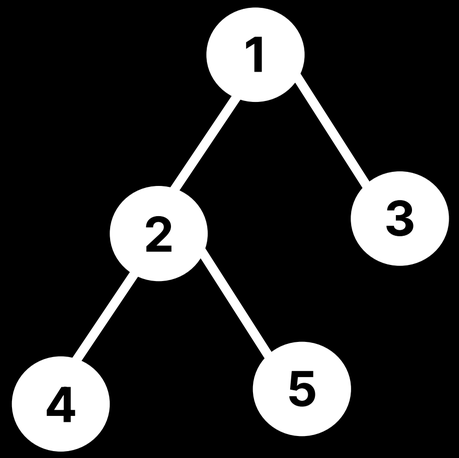
Disse er komplette

Bare en rot

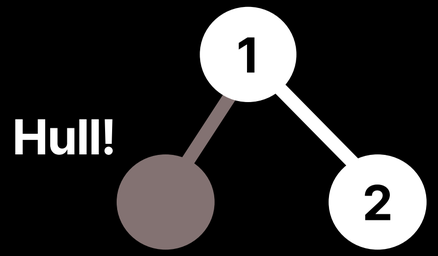


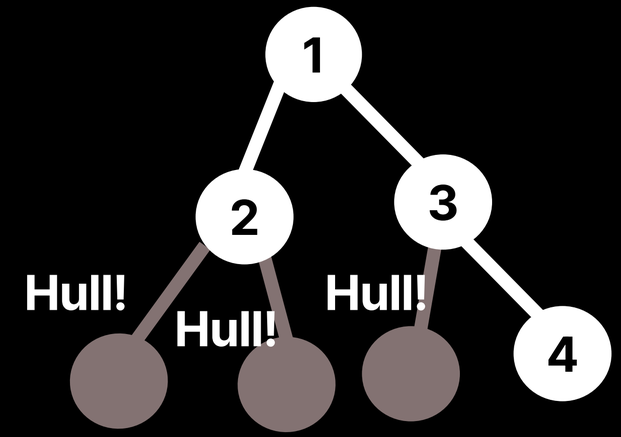


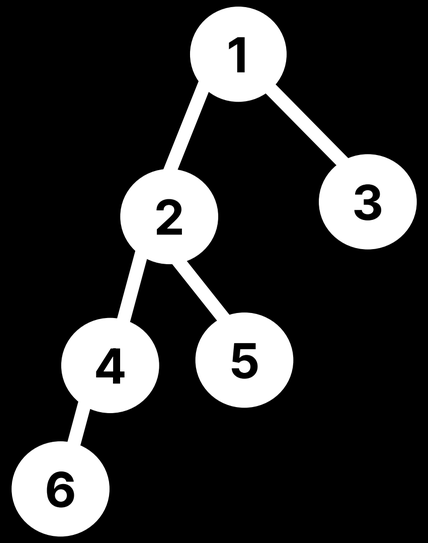




Disse er IKKE komplette





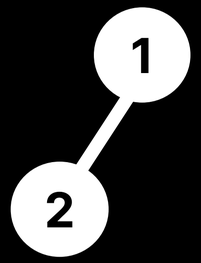


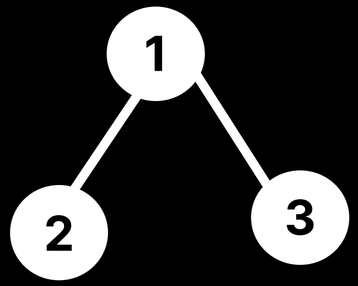
Komplette binær trær

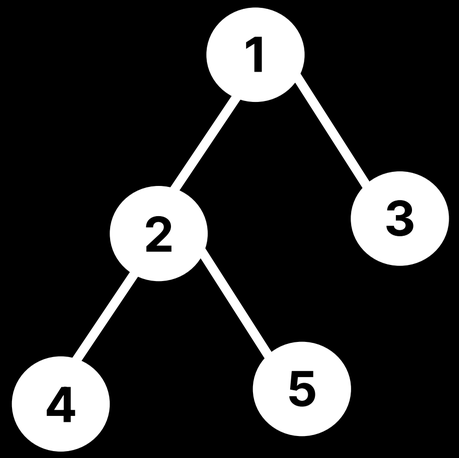
Disse er komplette

Bare en rot

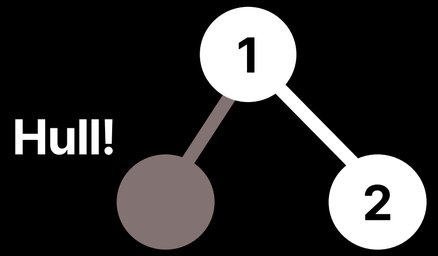


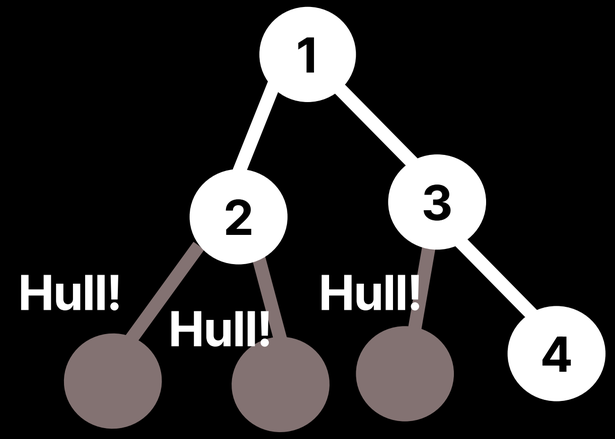


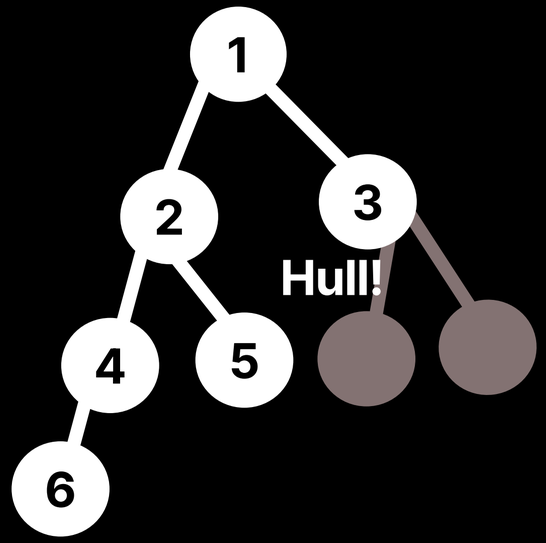




Disse er IKKE komplette







Binære heaps

Min-heap

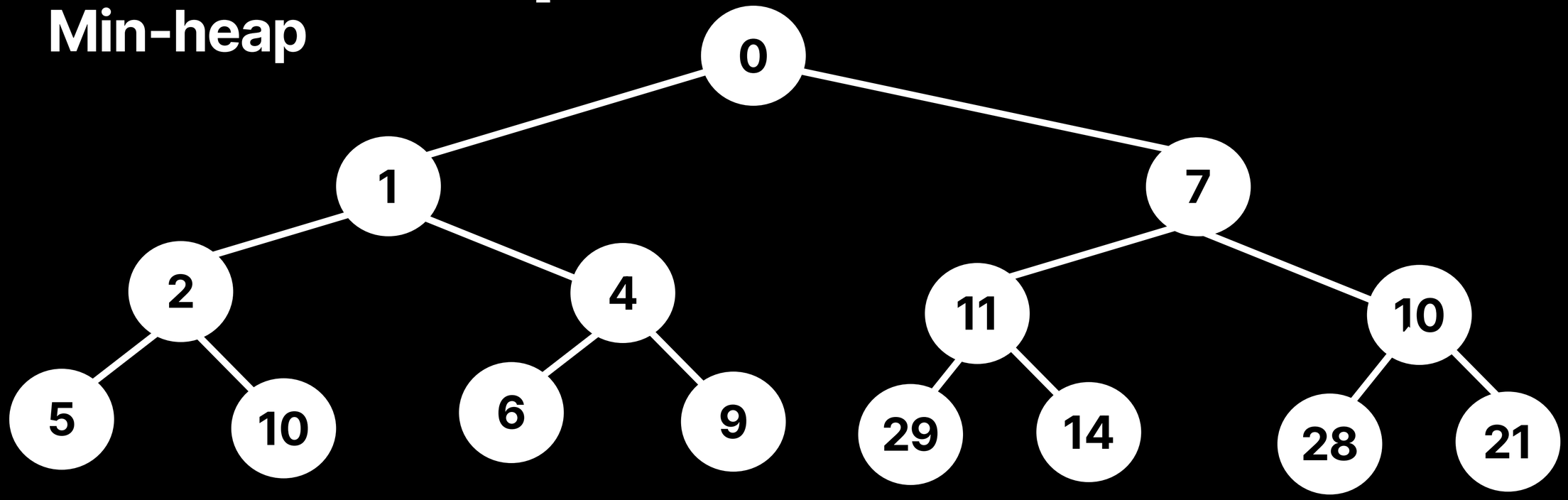
- Hver node har en verdi som er **mindre enn eller lik verdien til dens barn**
- **Rroten har alltid den minste verdien.**

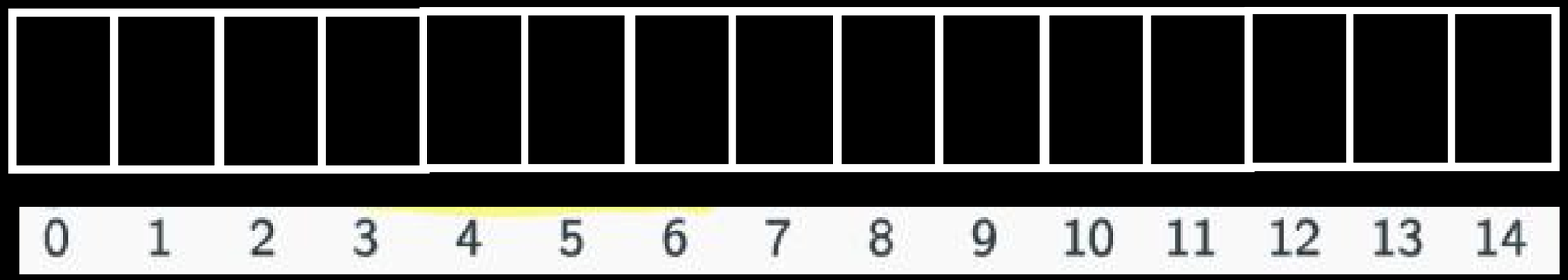
Maks-heap

- Hver node har en verdi som er **større enn eller lik verdien til dens barn**
- **Roten har alltid den største verdien).**

Binære heaps

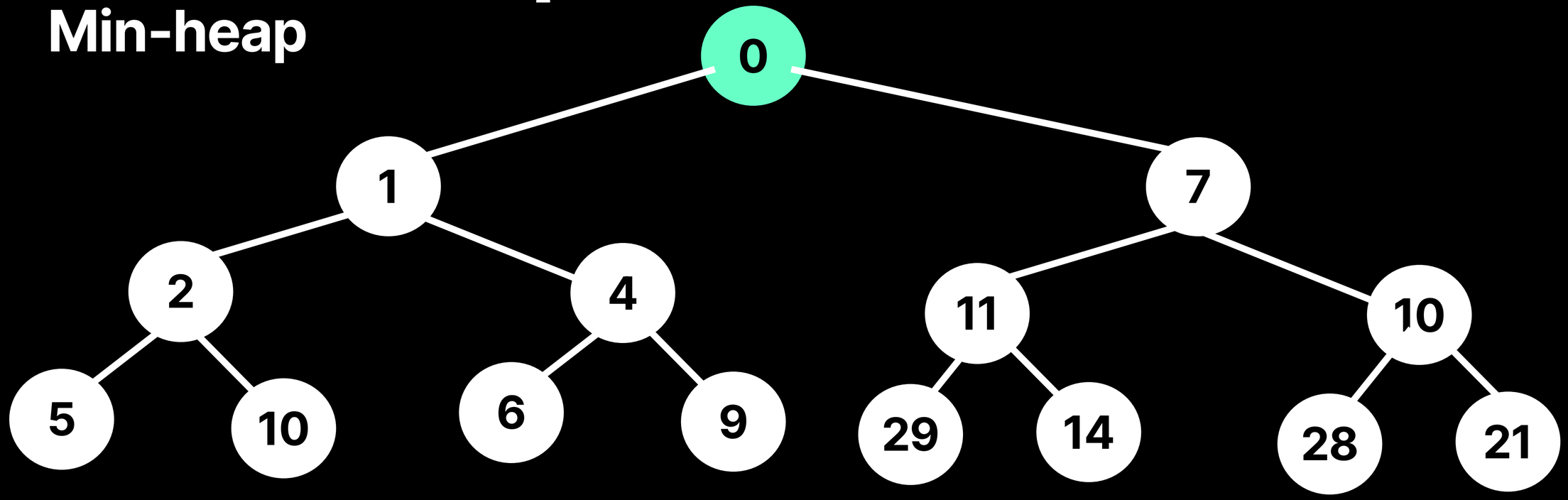
Min-heap

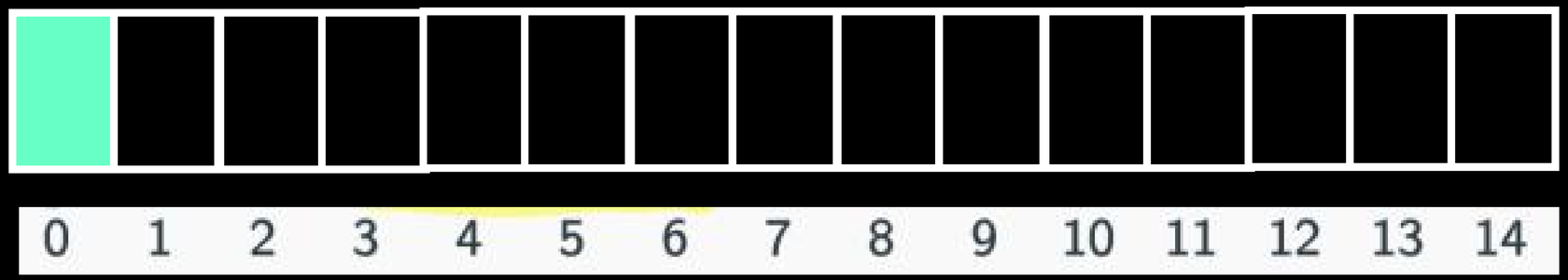




Binære heaps

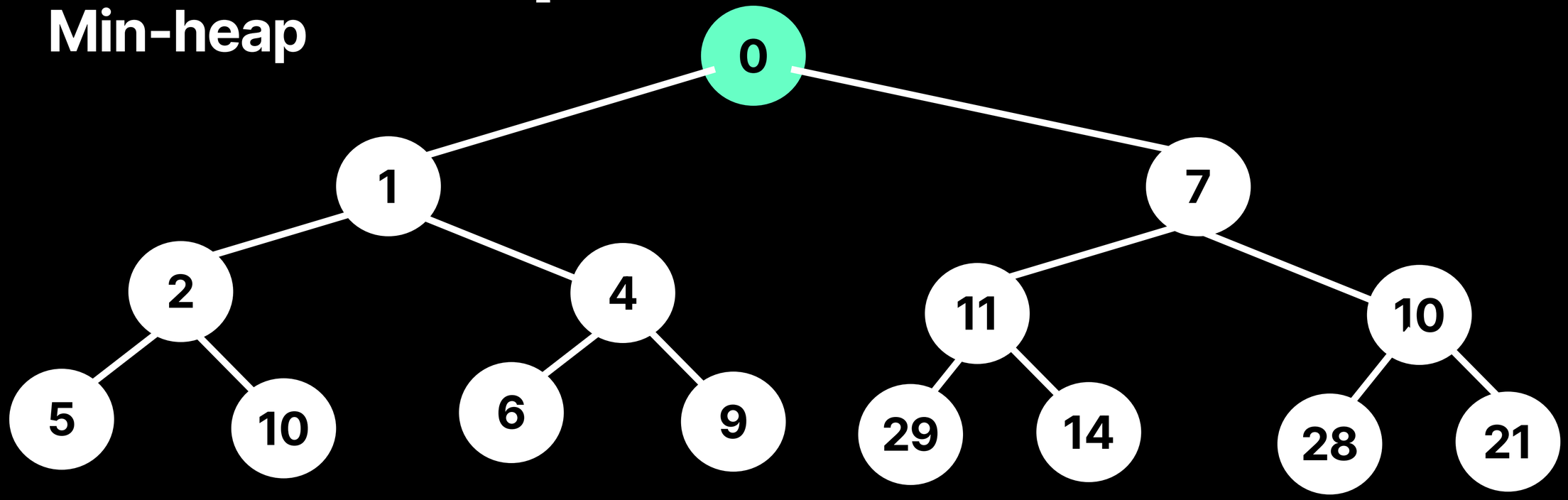
Min-heap





Binære heaps

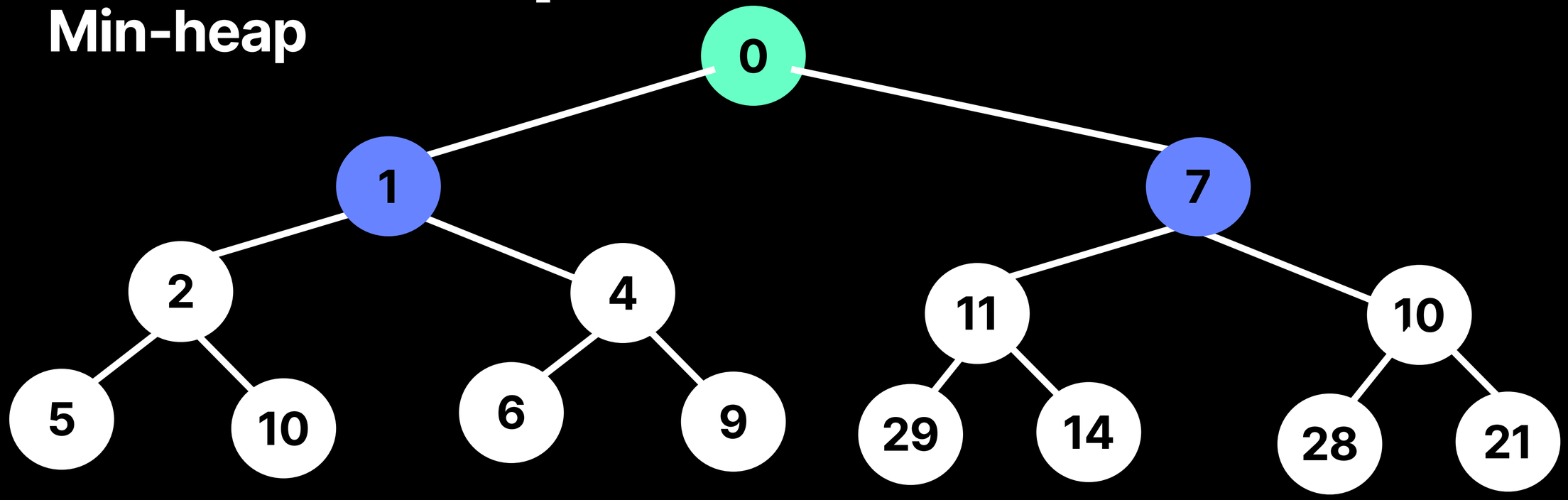
Min-heap

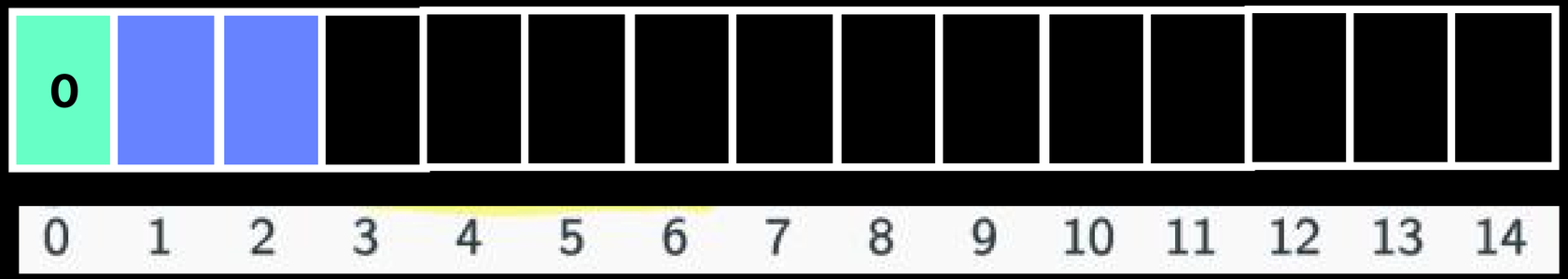




Binære heaps

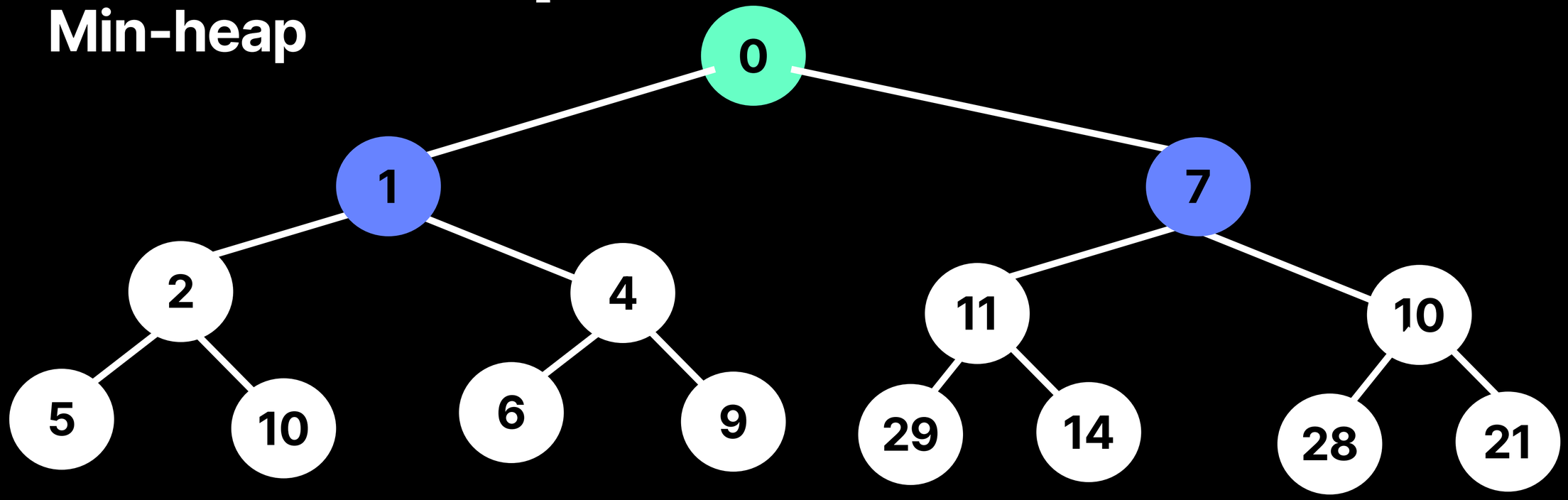
Min-heap





Binære heaps

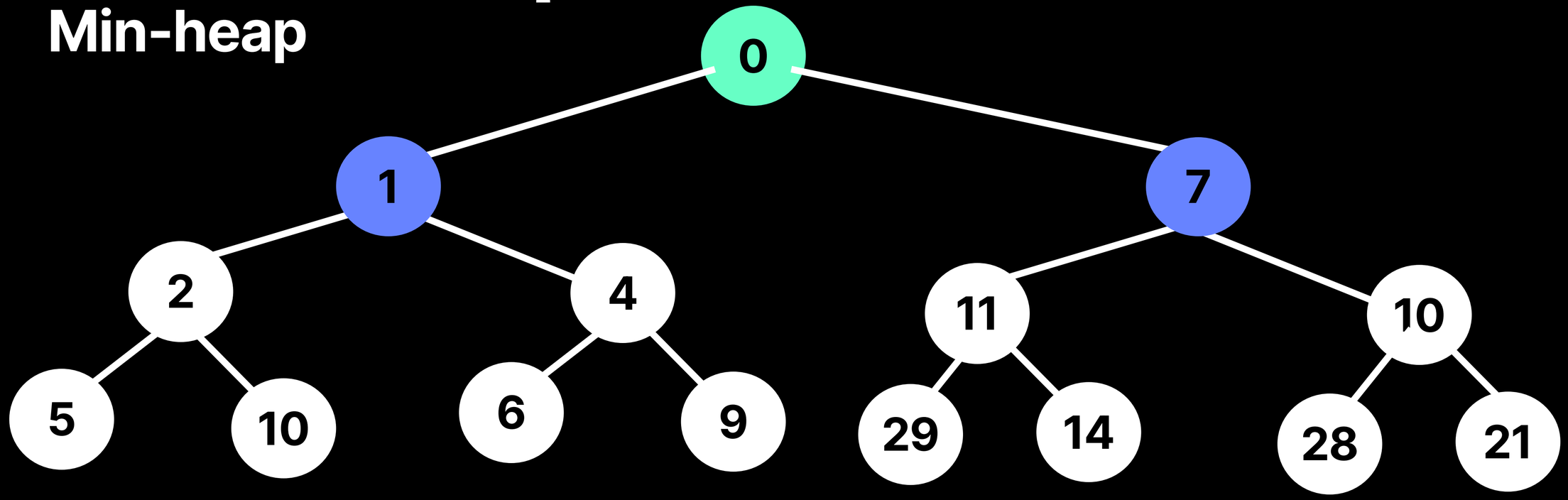
Min-heap

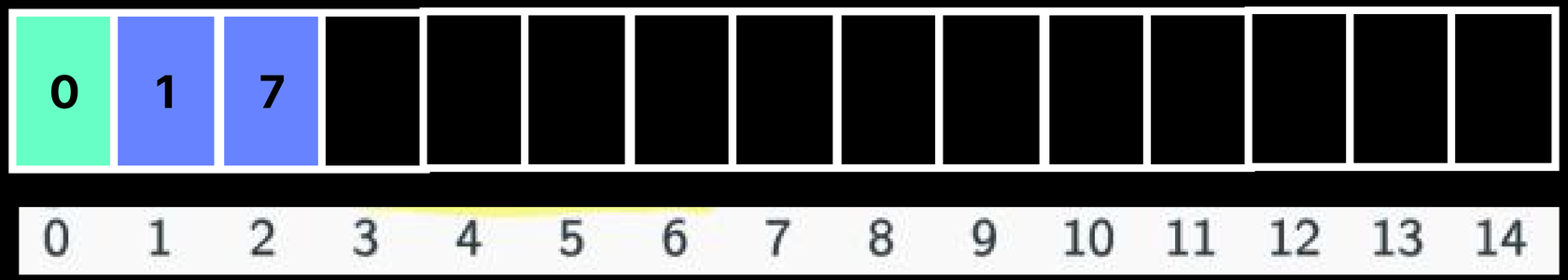




Binære heaps

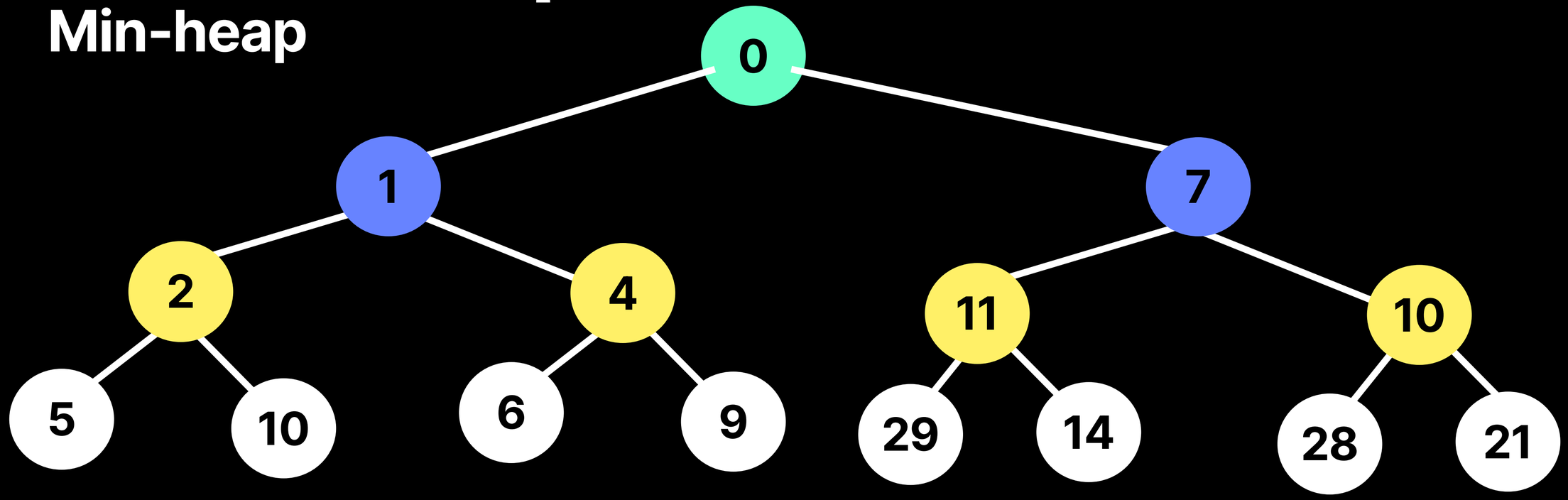
Min-heap

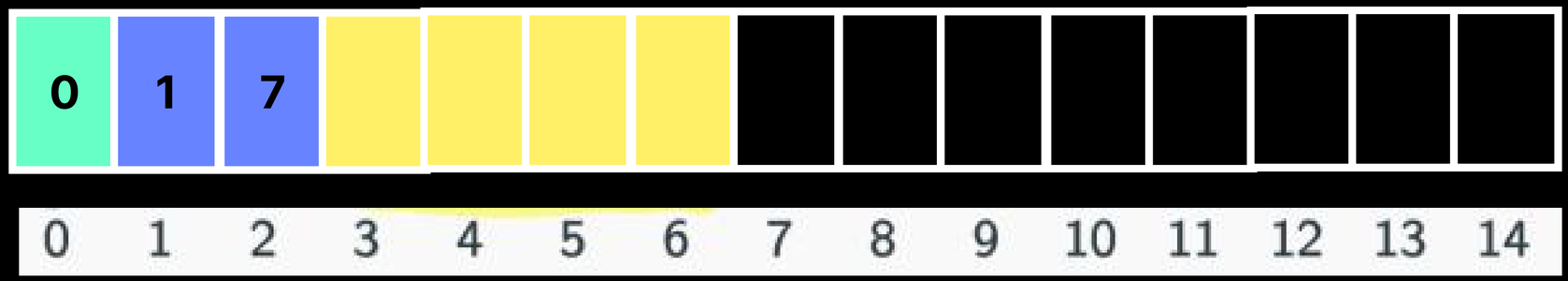




Binære heaps

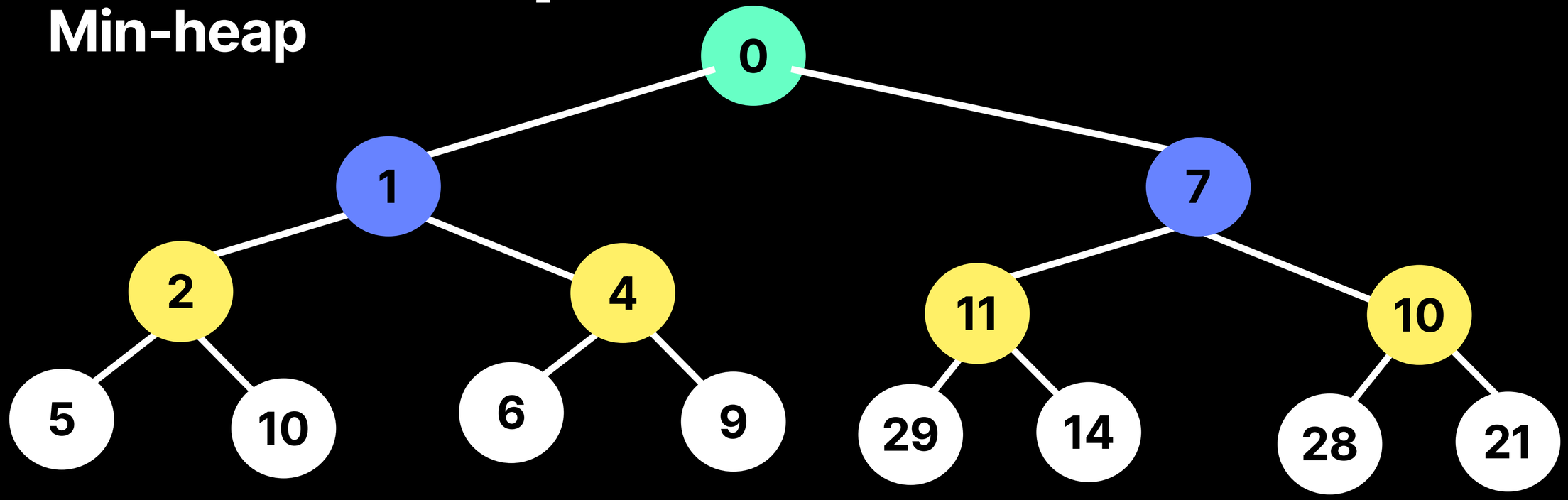
Min-heap

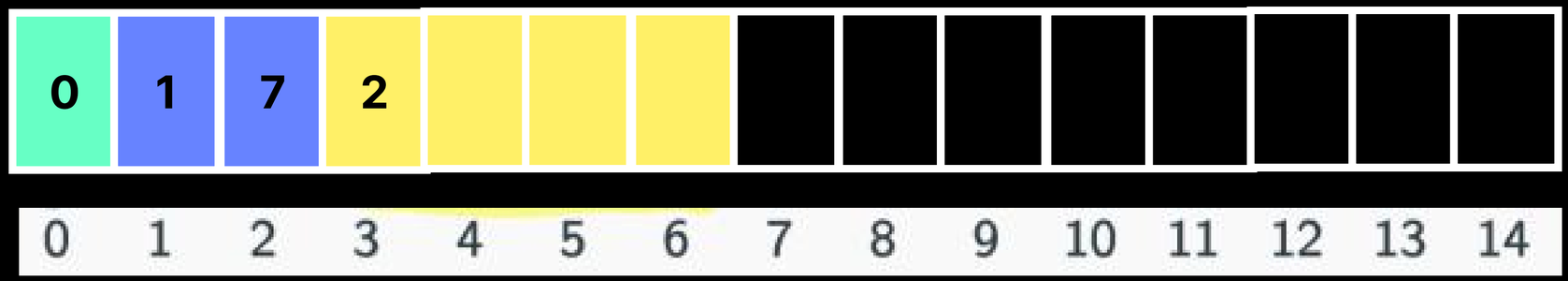




Binære heaps

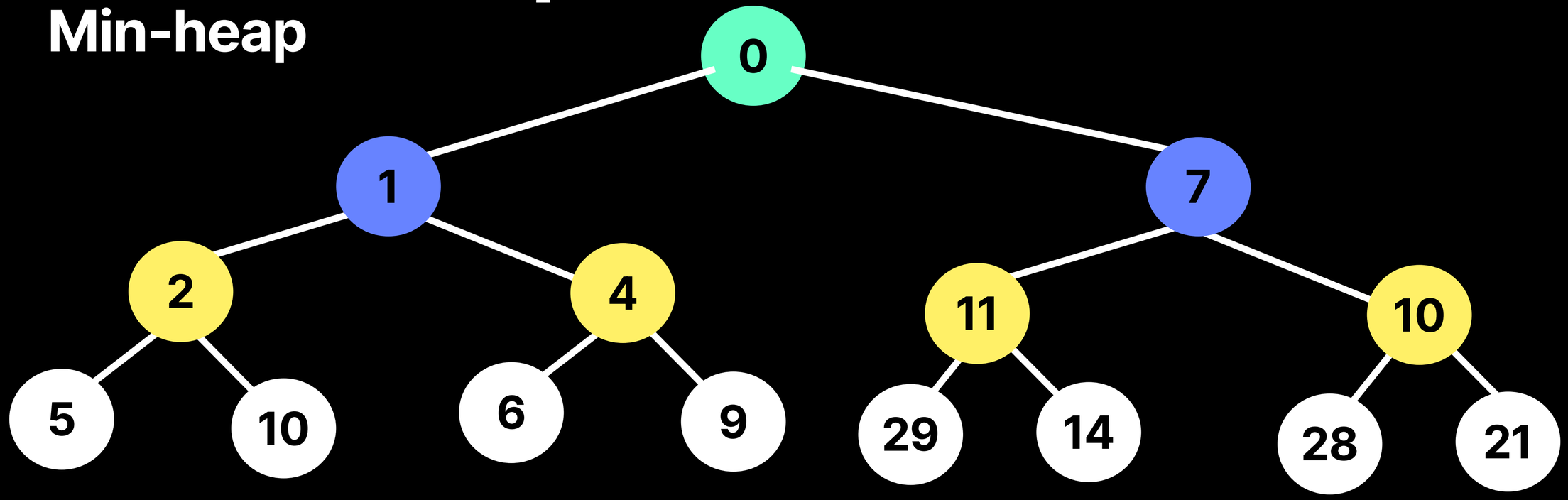
Min-heap

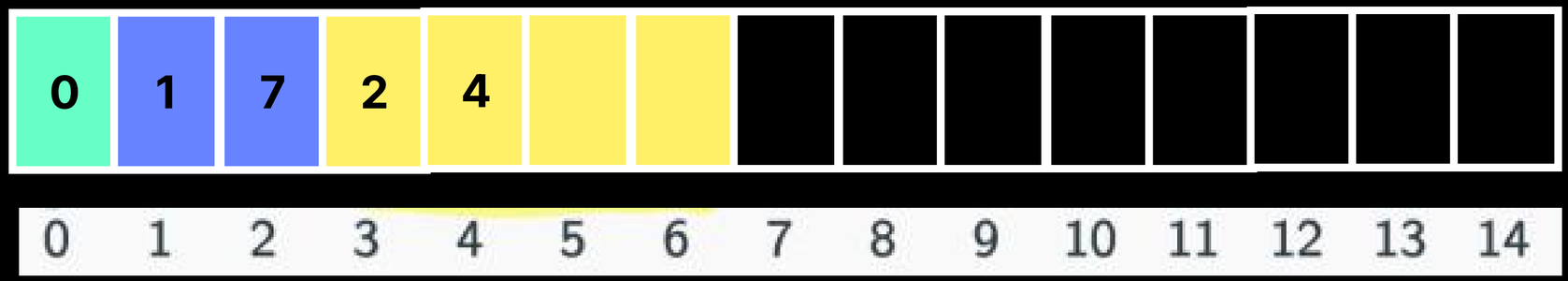




Binære heaps

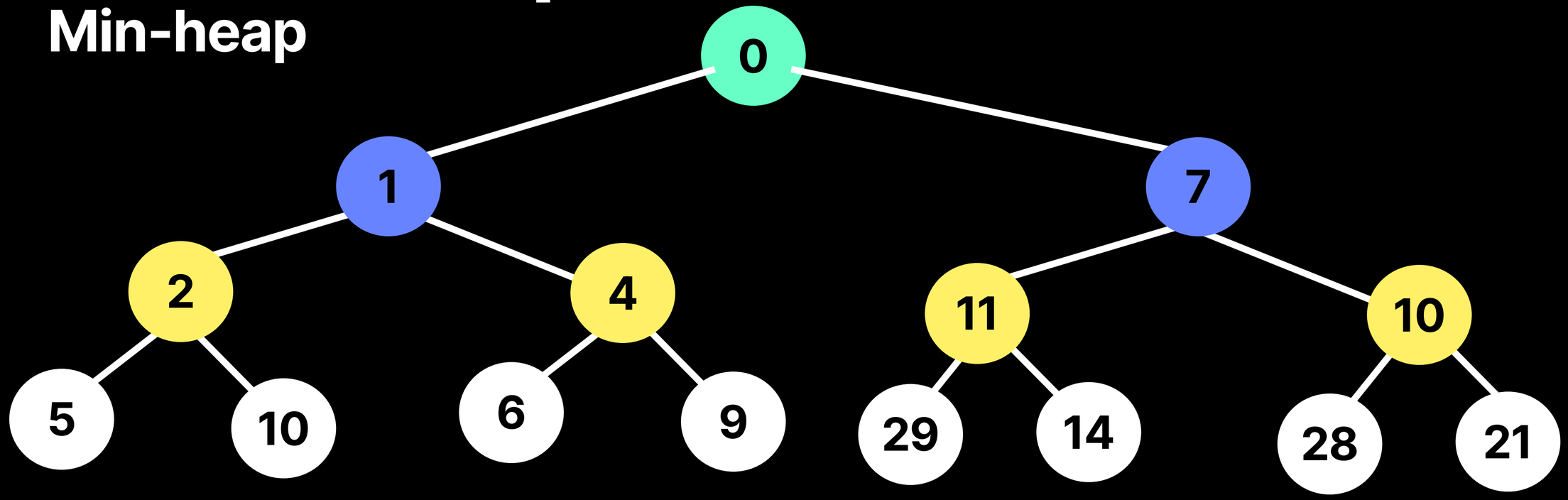
Min-heap

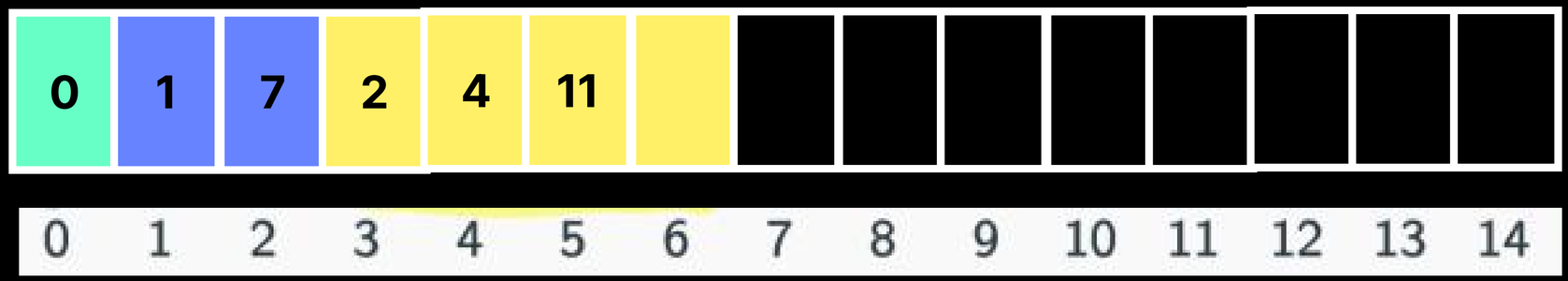




Binære heaps

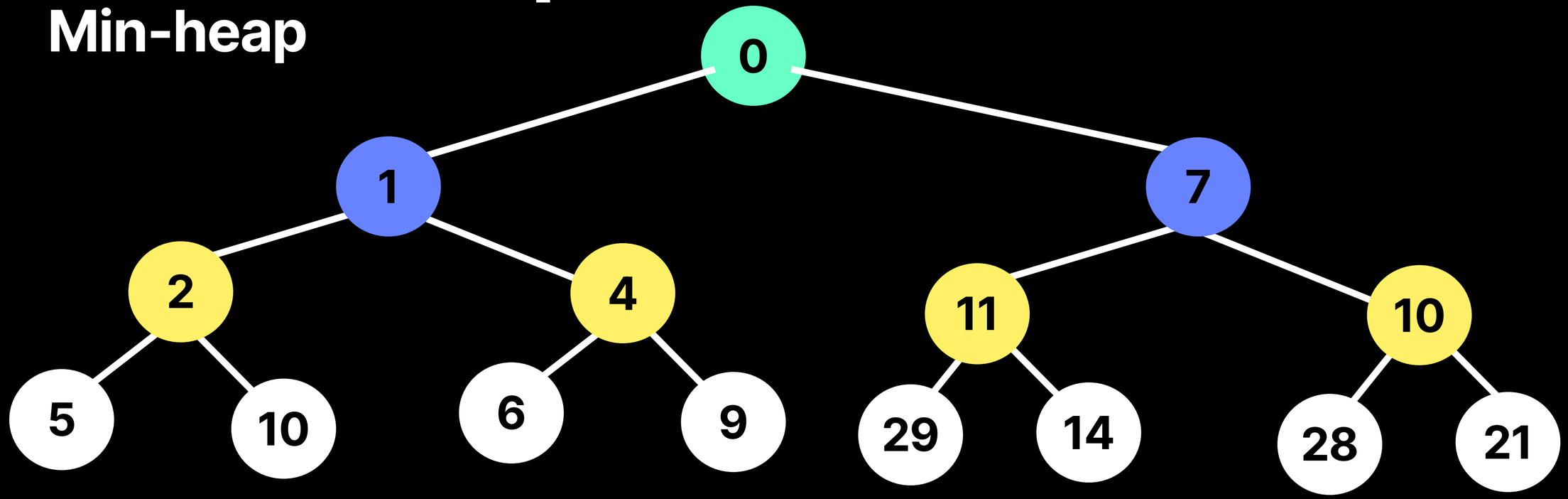
Min-heap

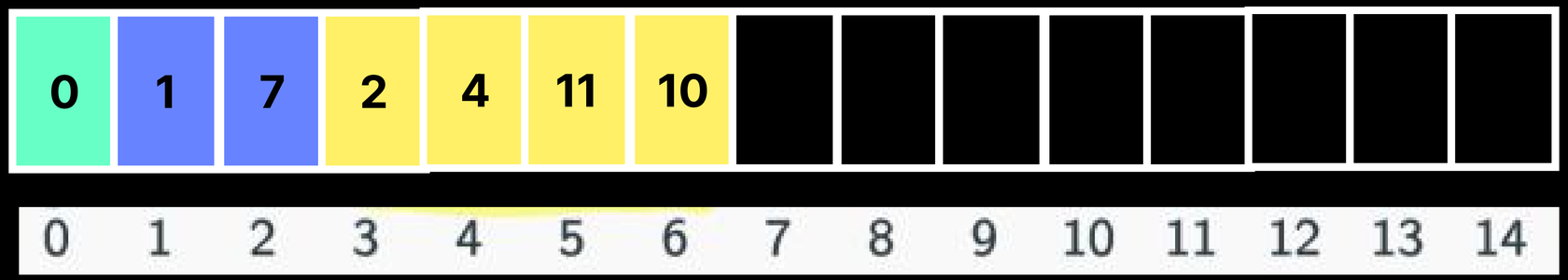




Binære heaps

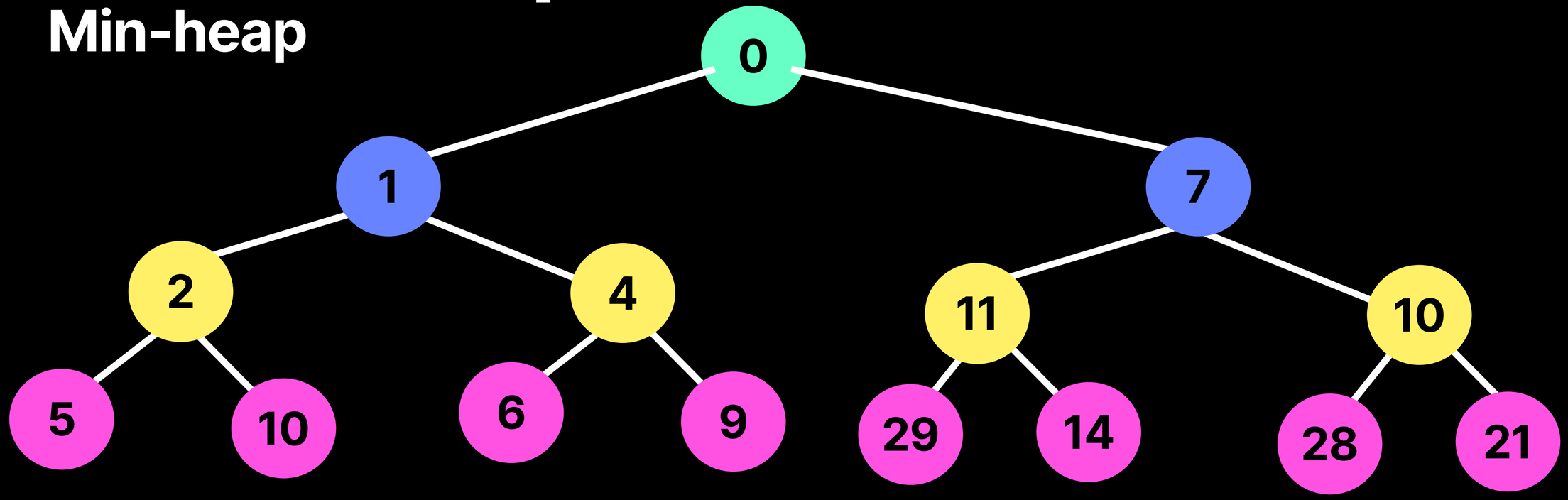
Min-heap

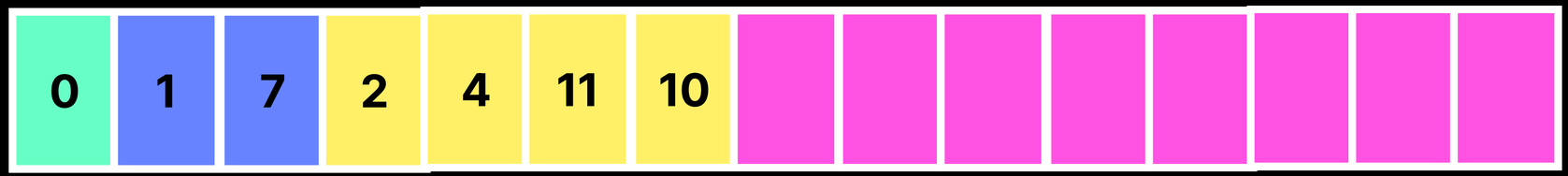




Binære heaps

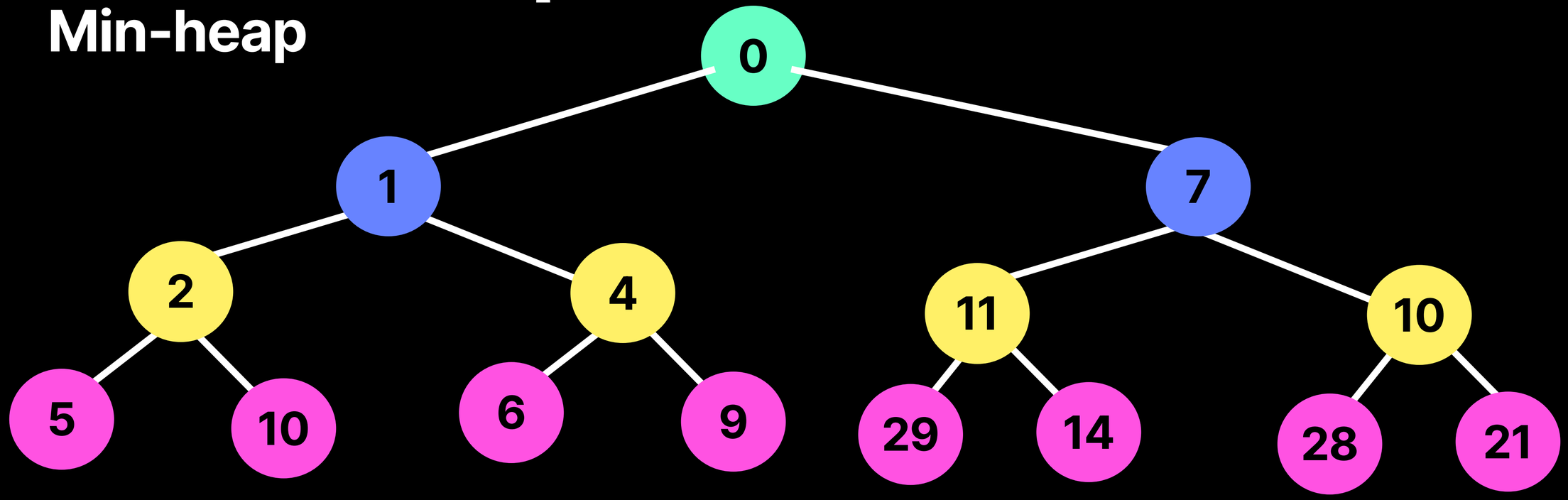
Min-heap

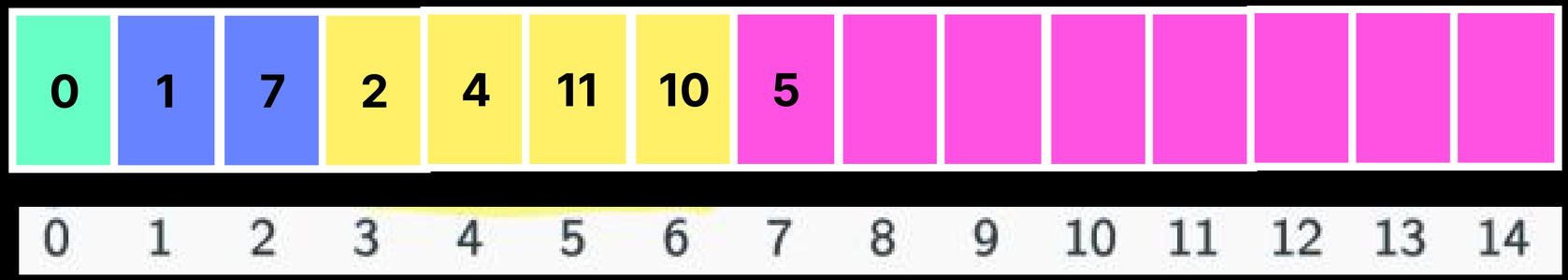




Binære heaps

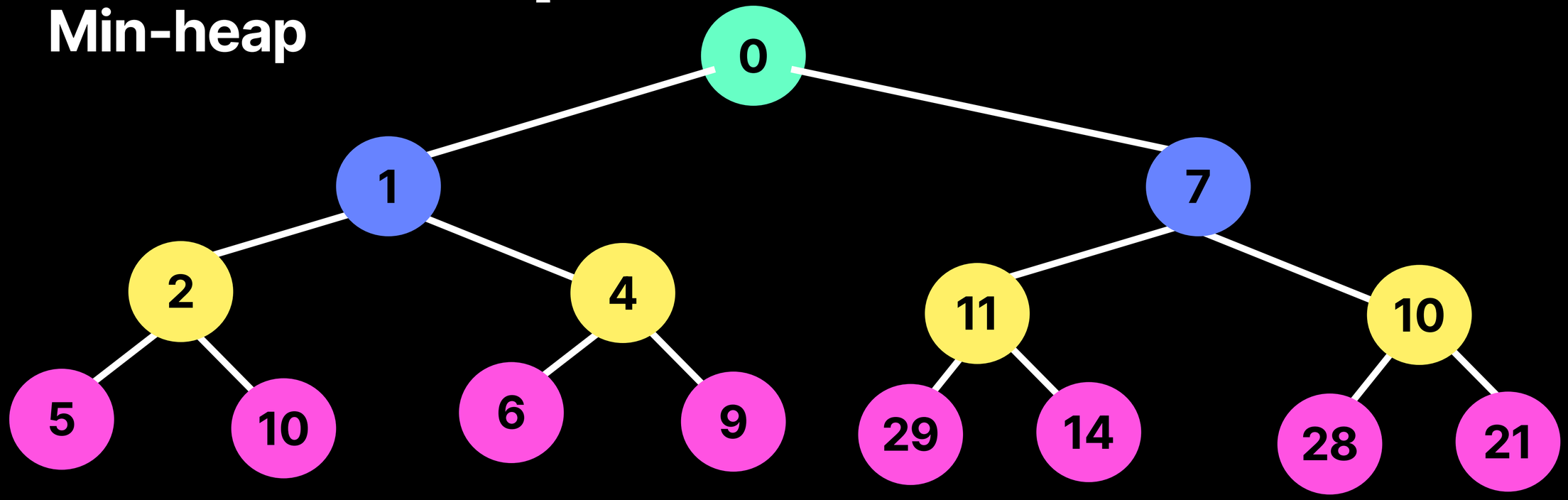
Min-heap

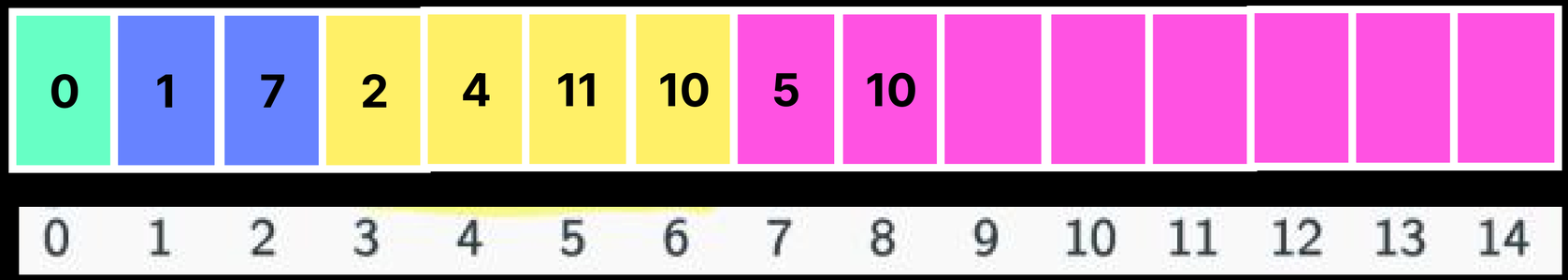




Binære heaps

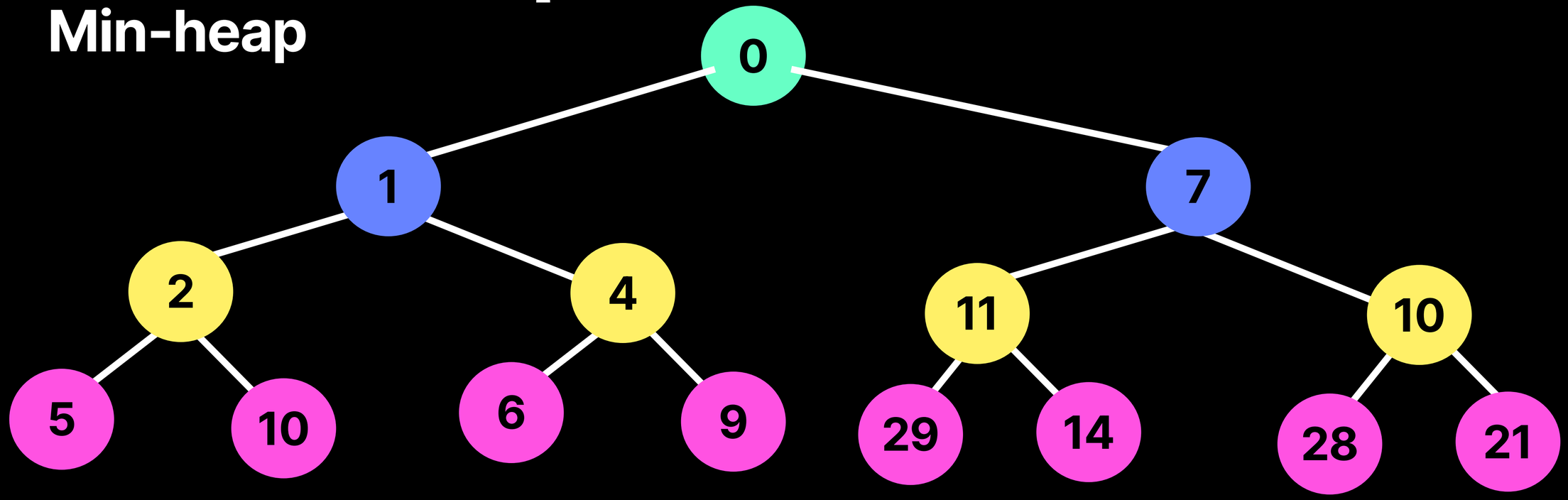
Min-heap

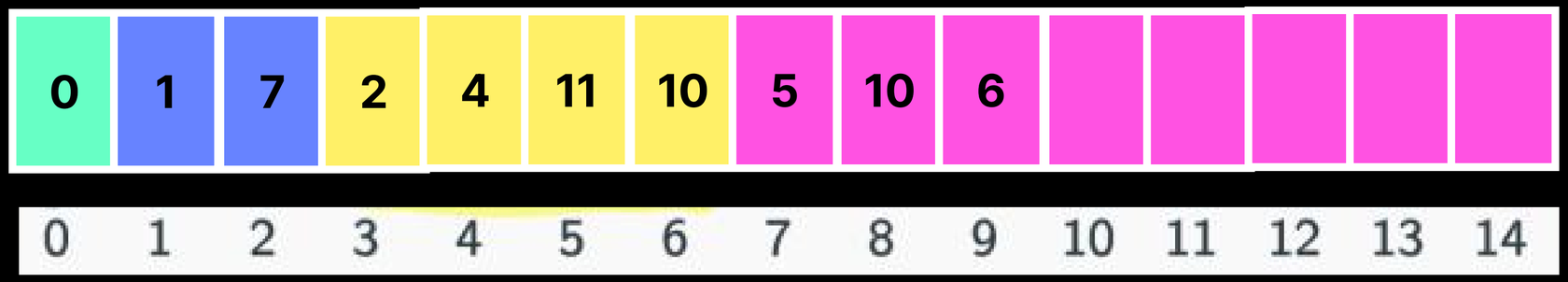




Binære heaps

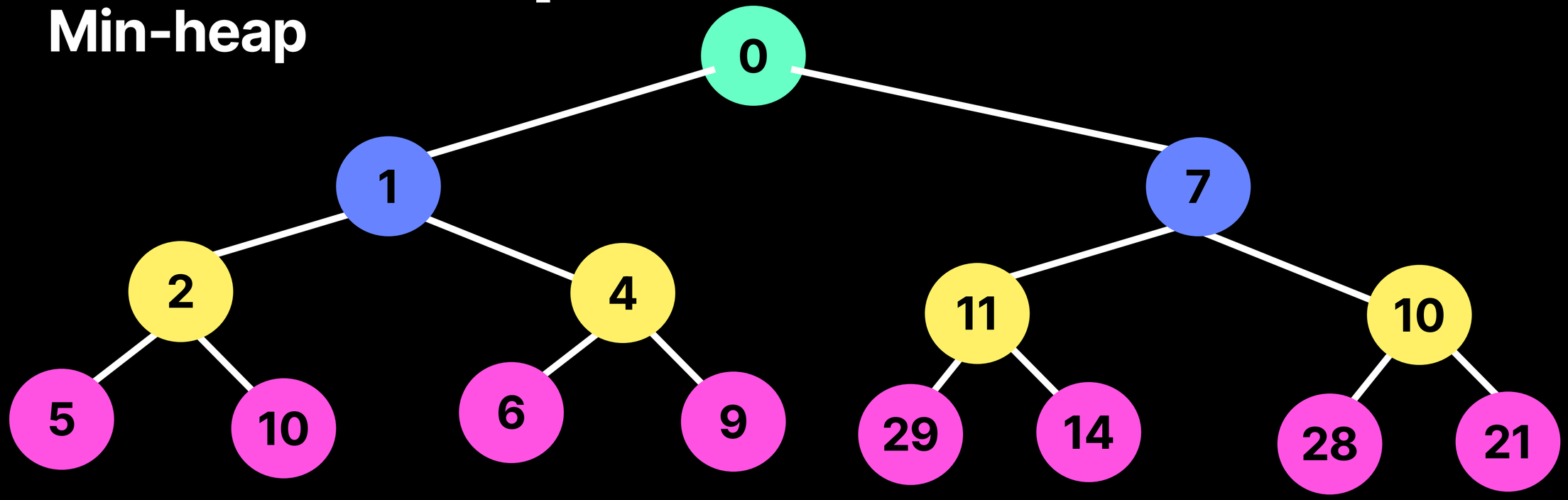
Min-heap

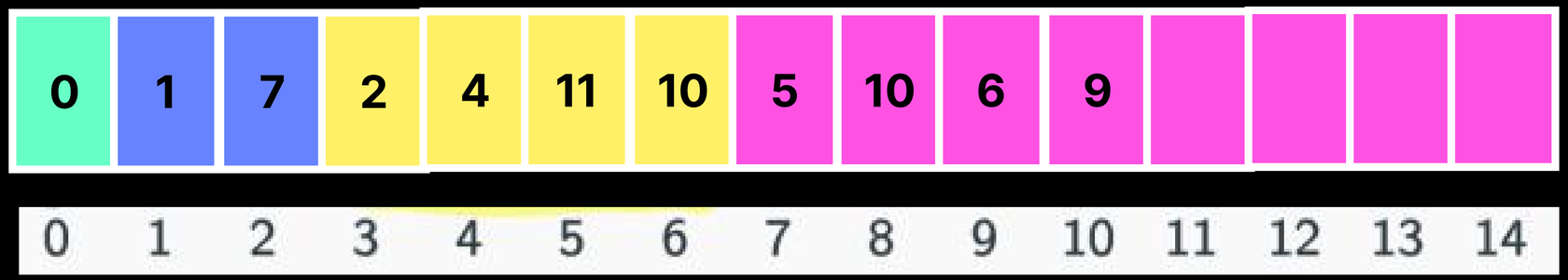




Binære heaps

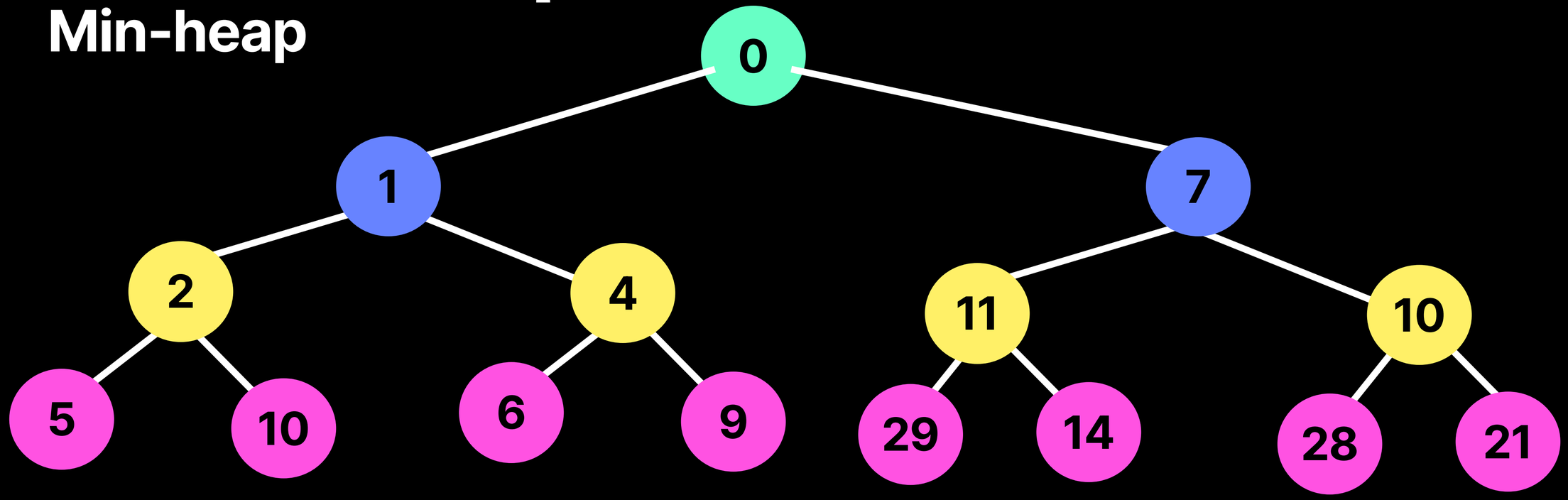
Min-heap

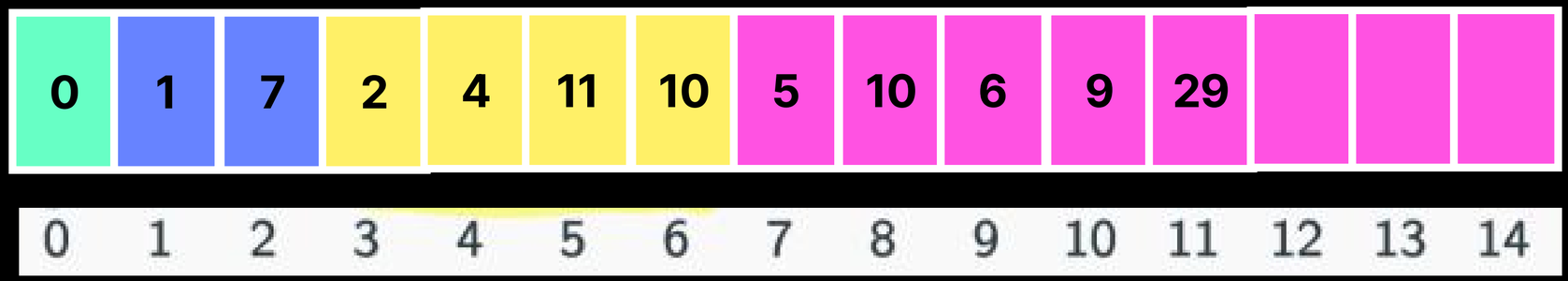




Binære heaps

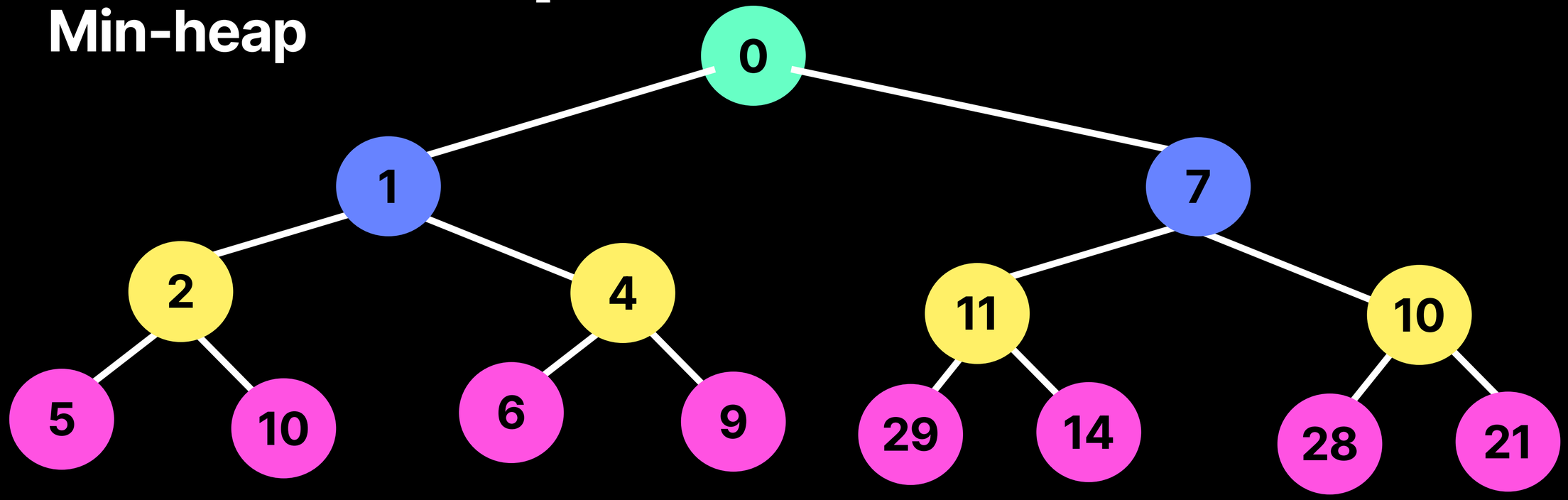
Min-heap

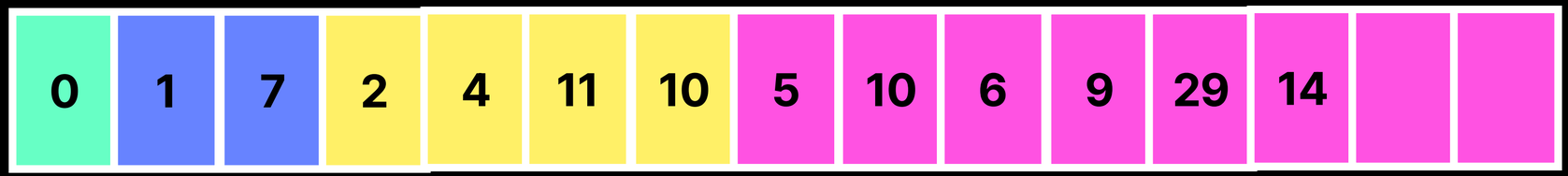




Binære heaps

Min-heap

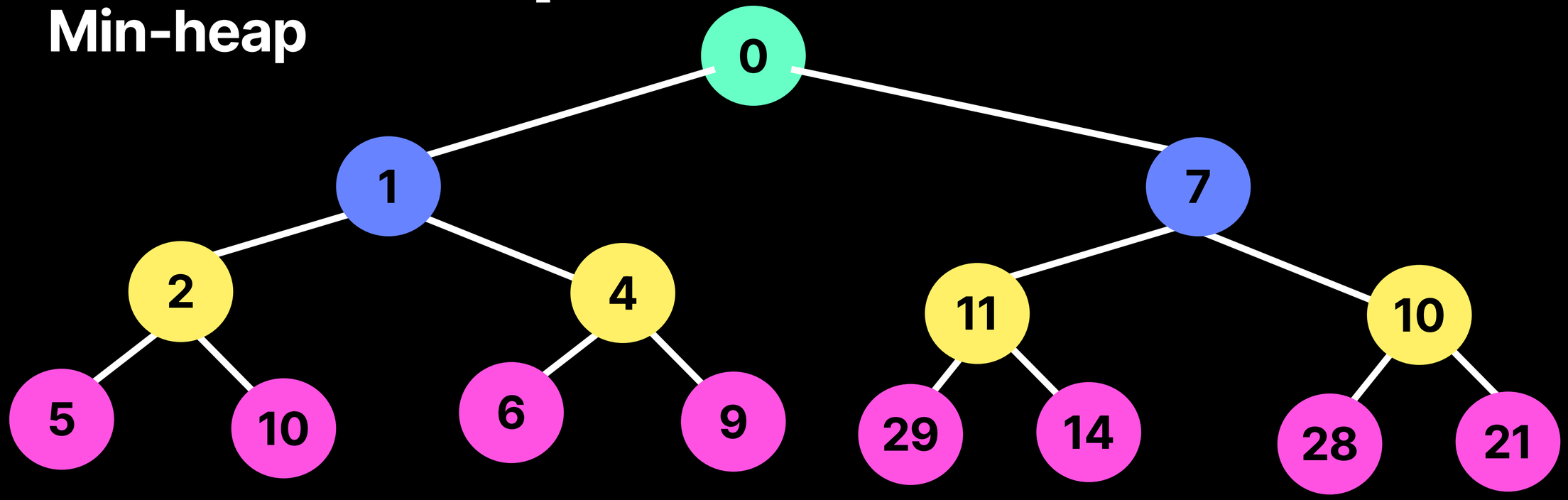


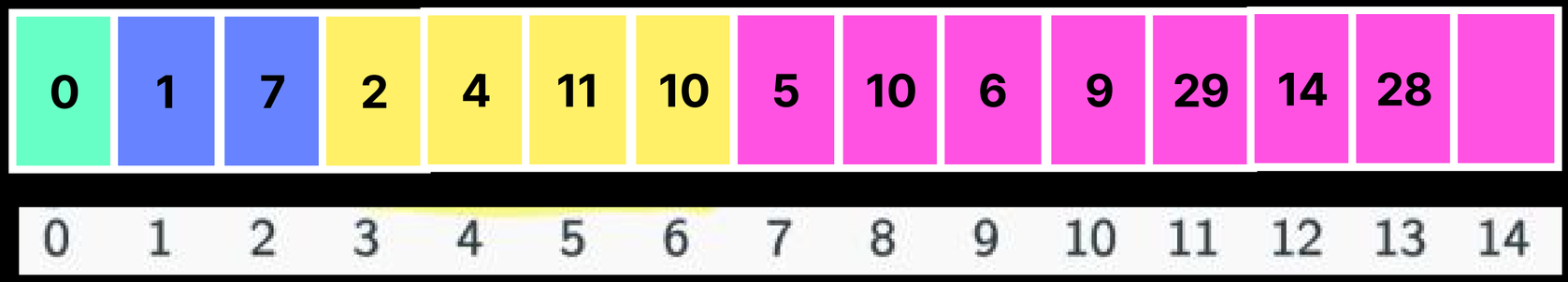




Binære heaps

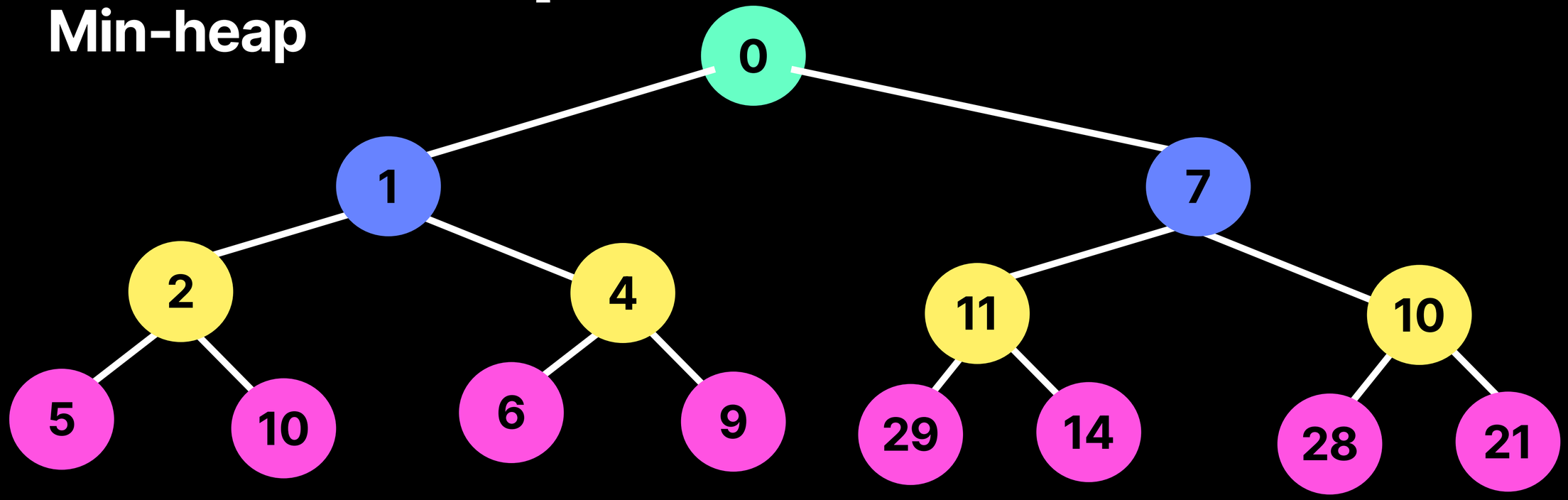
Min-heap





Binære heaps

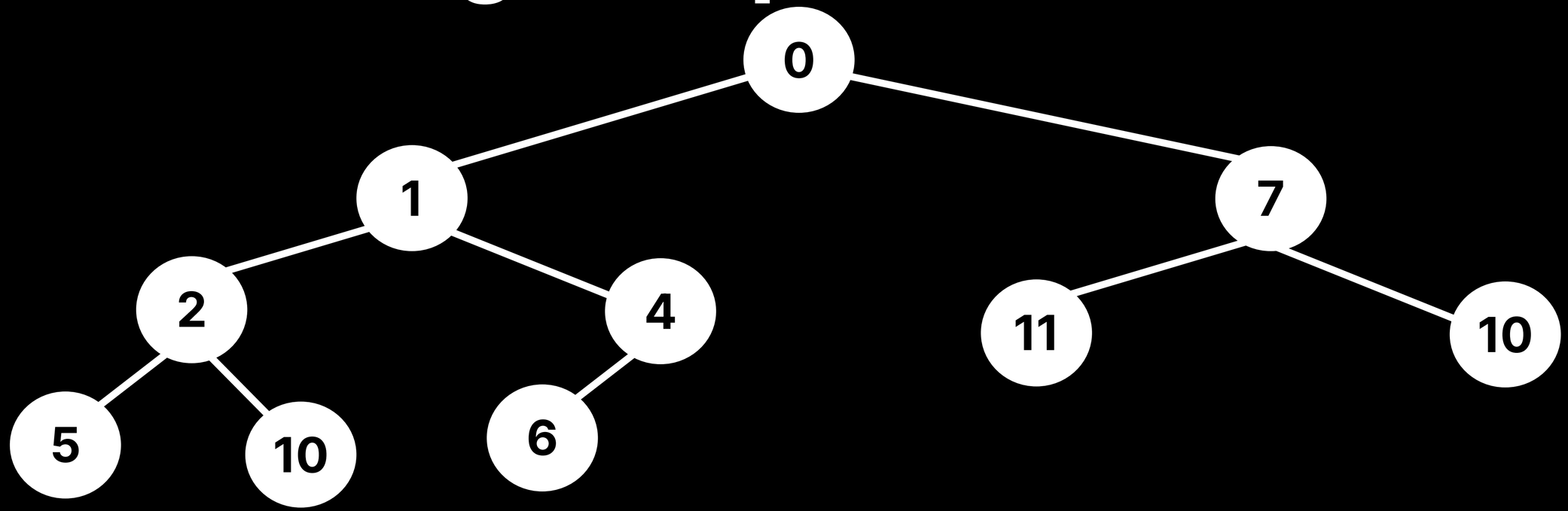
Min-heap



0	1	7	2	4	11	10	5	10	6	9	29	14	28	21

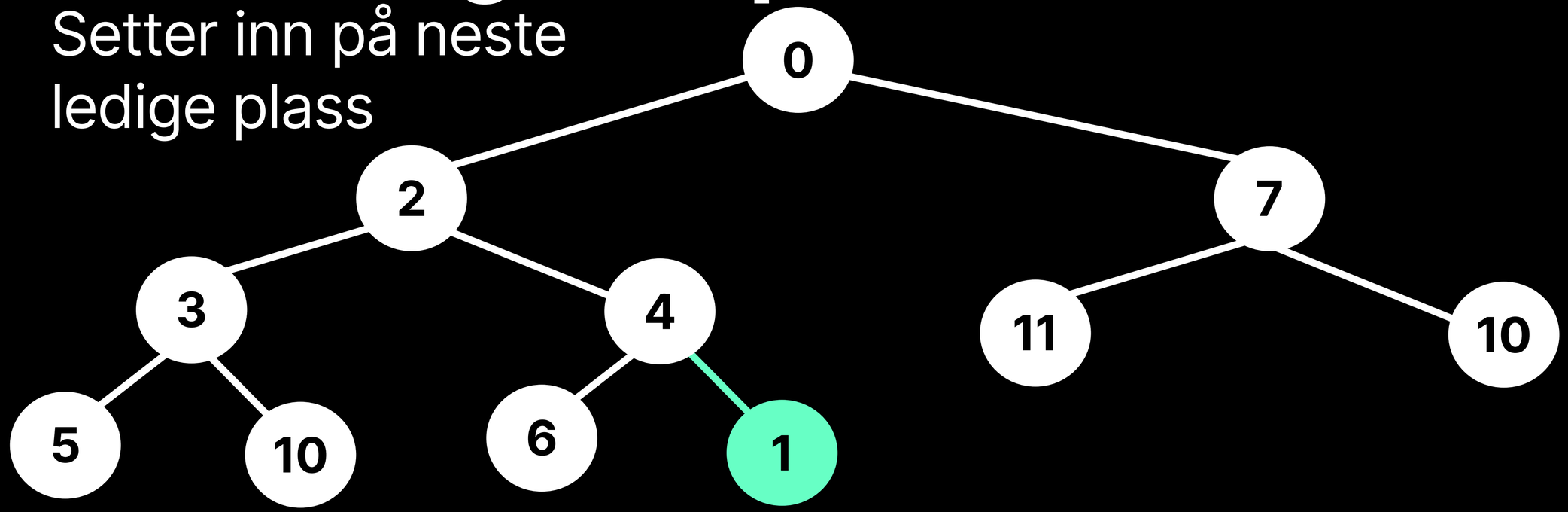
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Innsetting i heap



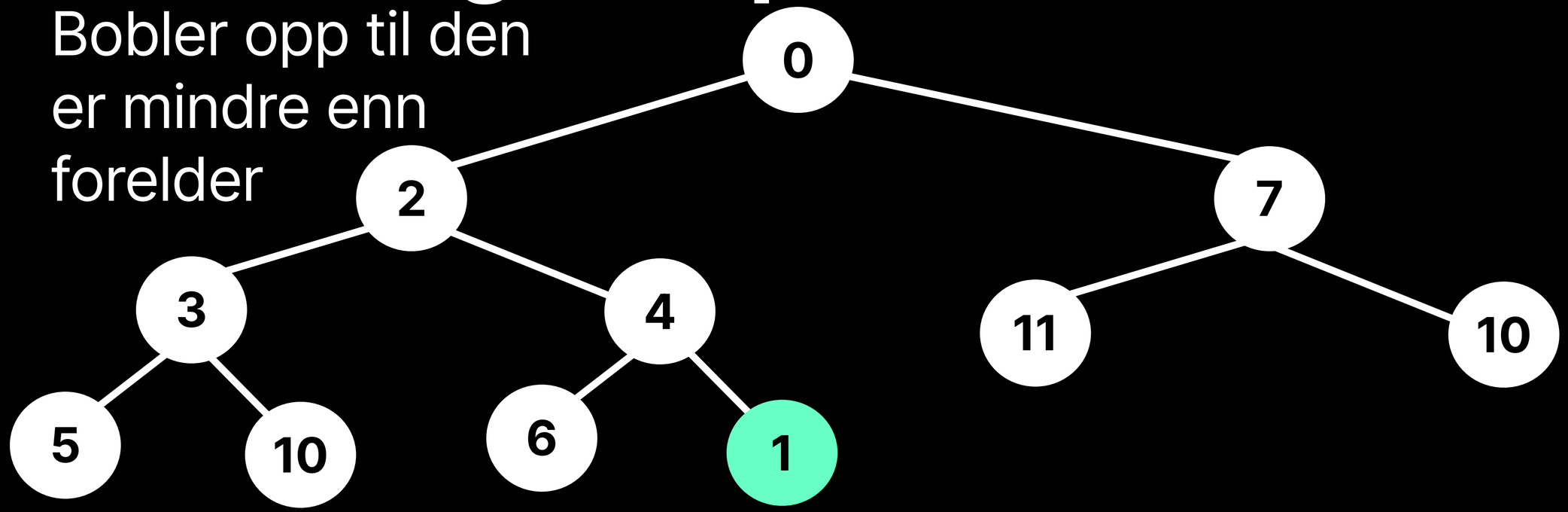
Innsetting i heap

Setter inn på neste ledige plass



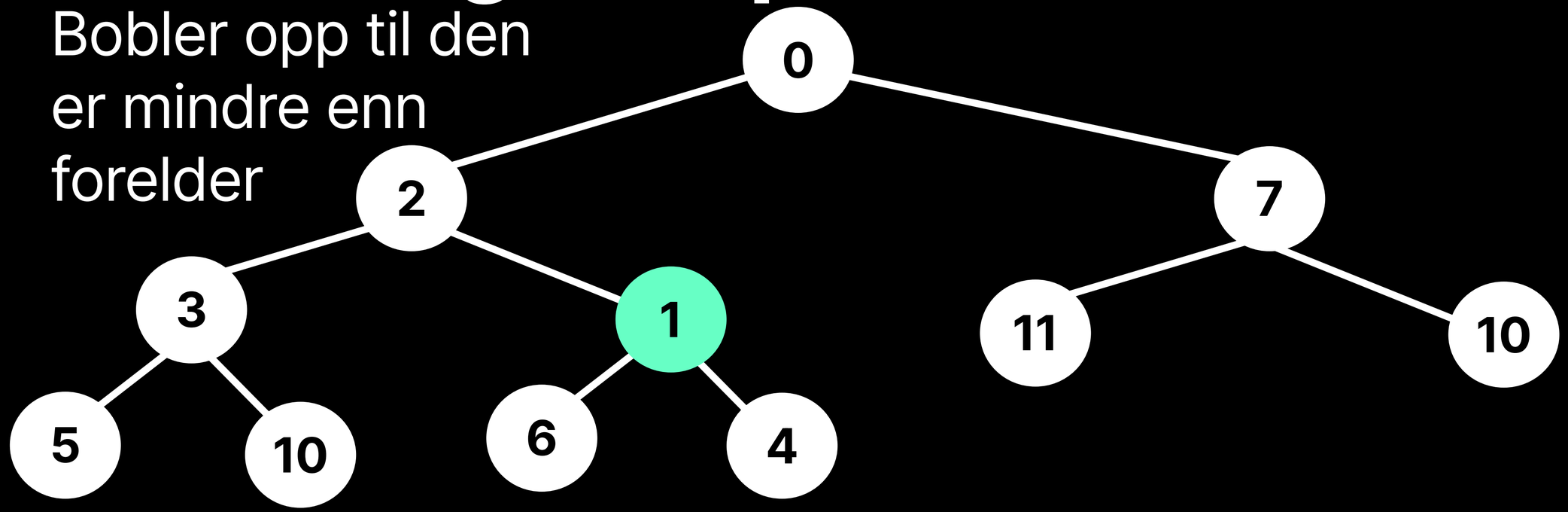
Innsetting i heap

Bobler opp til den er mindre enn forelder



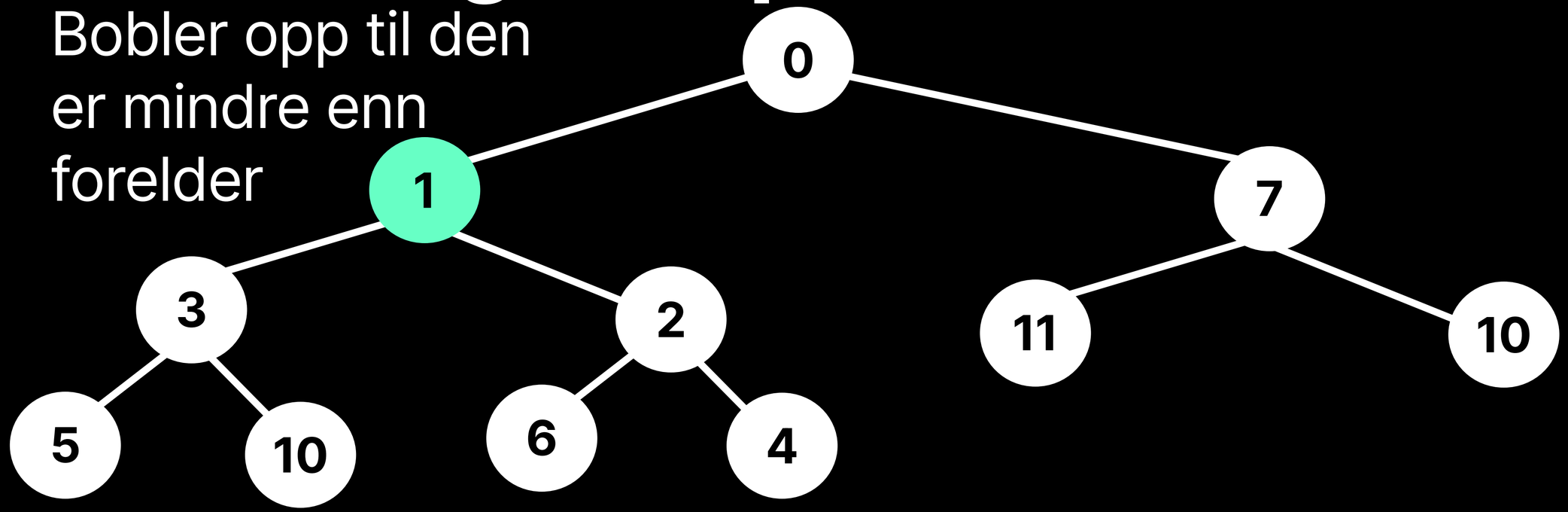
Innsetting i heap

Bobler opp til den er mindre enn forelder



Innsetting i heap

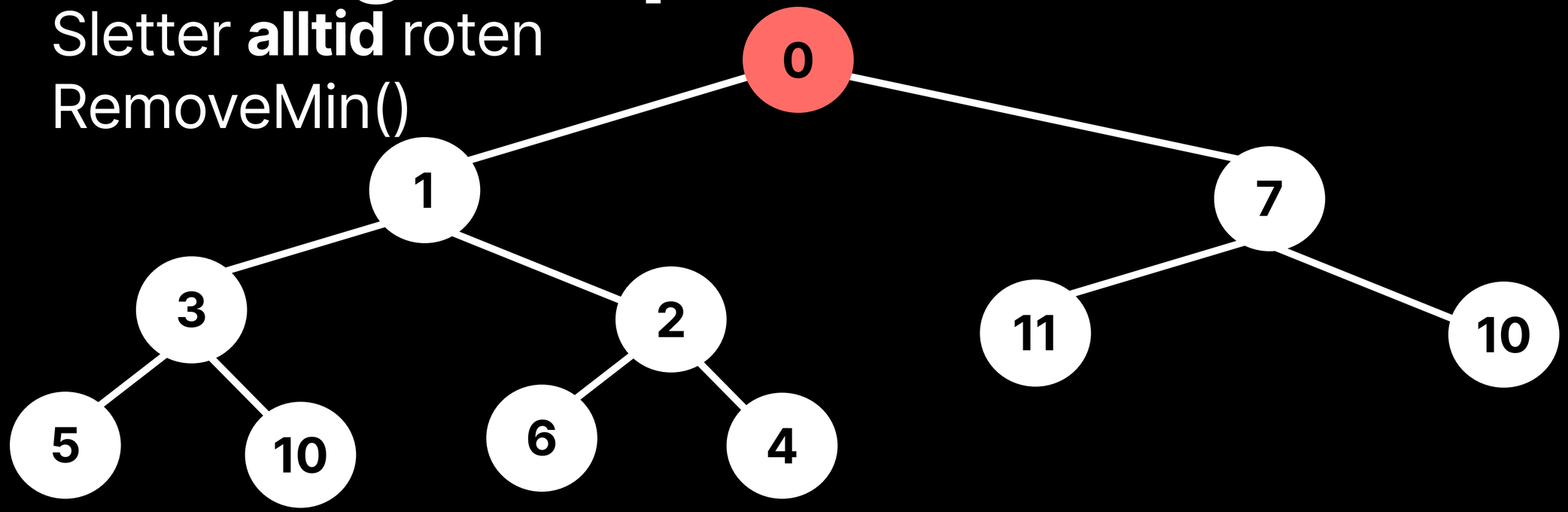
Bobler opp til den er mindre enn forelder



Sletting i heap

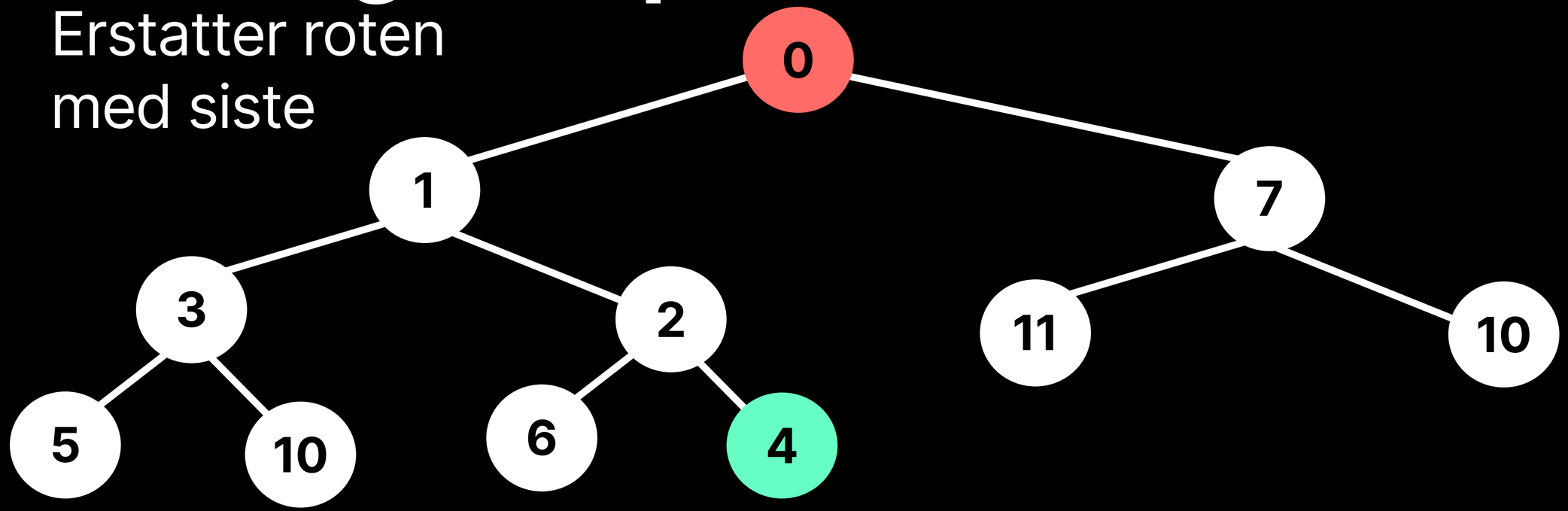
Sletter **alltid** roten

RemoveMin()



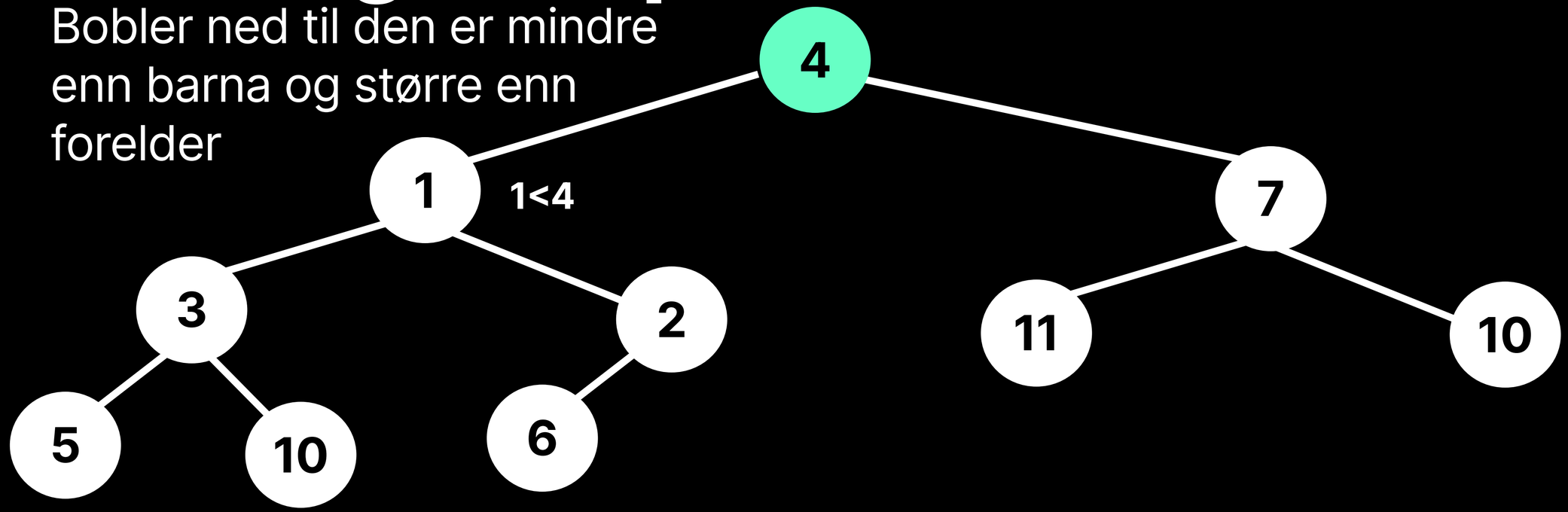
Sletting i heap

Erstatter roten
med siste



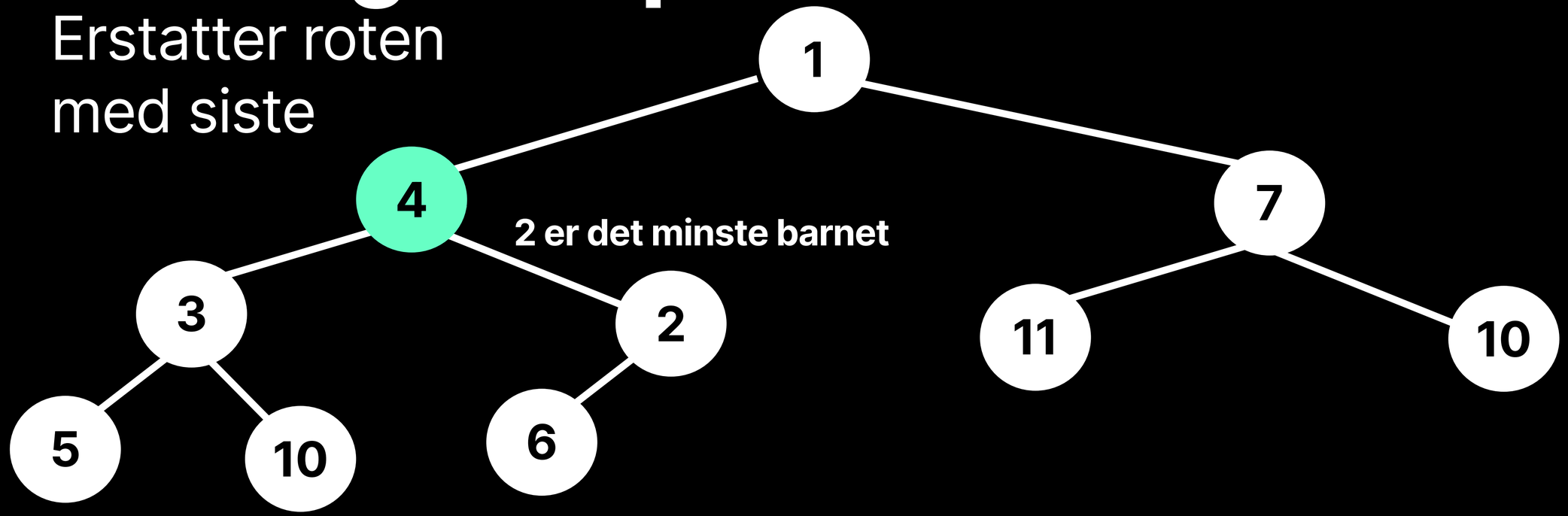
Sletting i heap

Bobler ned til den er mindre enn barna og større enn forelder



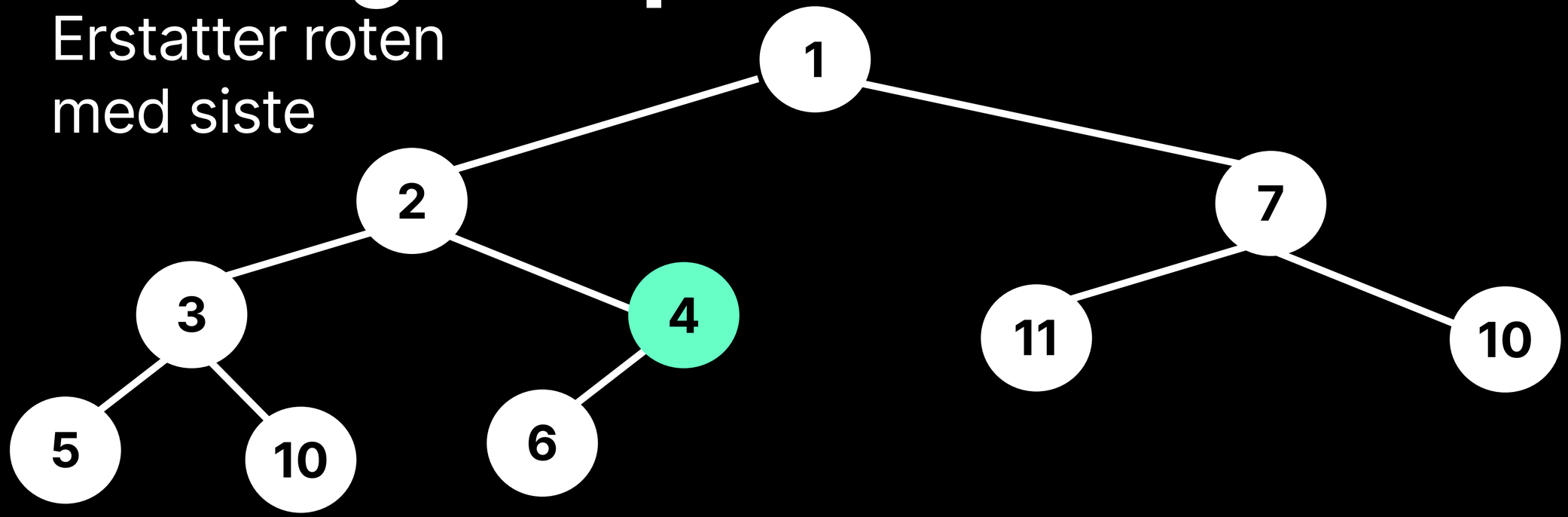
Sletting i heap

Erstatter roten
med siste



Sletting i heap

Erstatter roten
med siste



Sortering

HeapSort

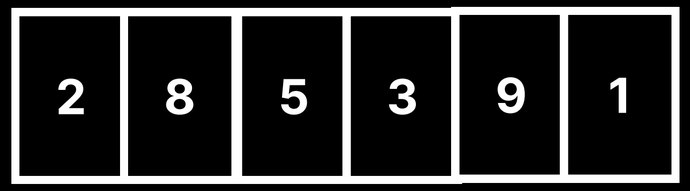
og

RadixSort

HeapSort

Maks-heap

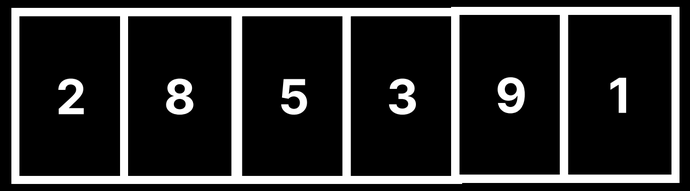
1. Lag en maks-heap

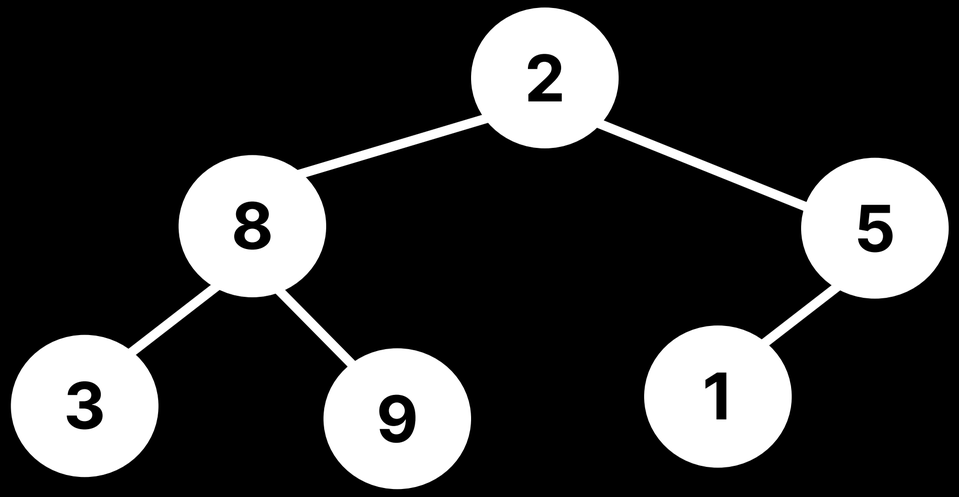


HeapSort

Maks-heap

1. Lag en maks-heap

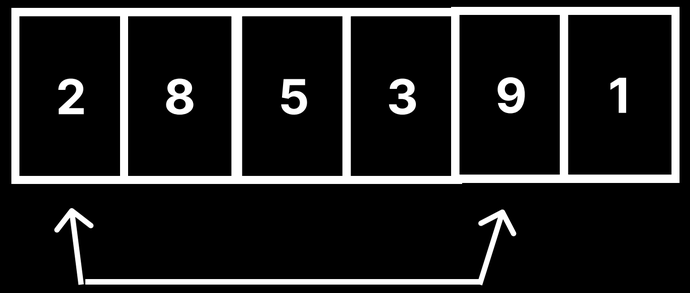


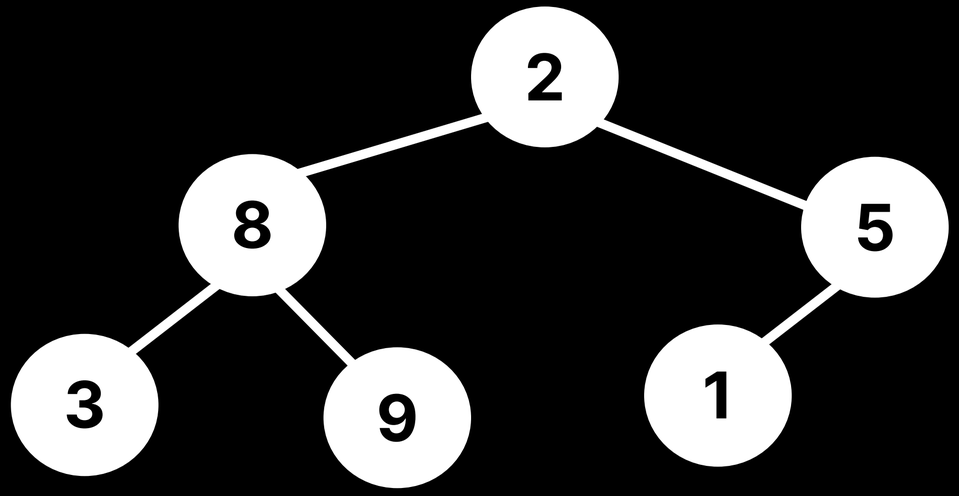


HeapSort

Maks-heap

1. Lag en maks-heap

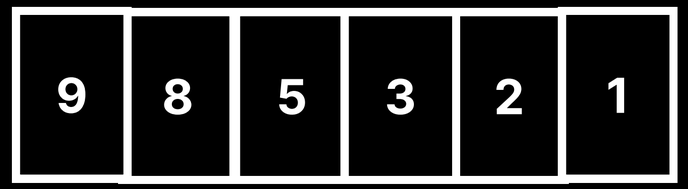


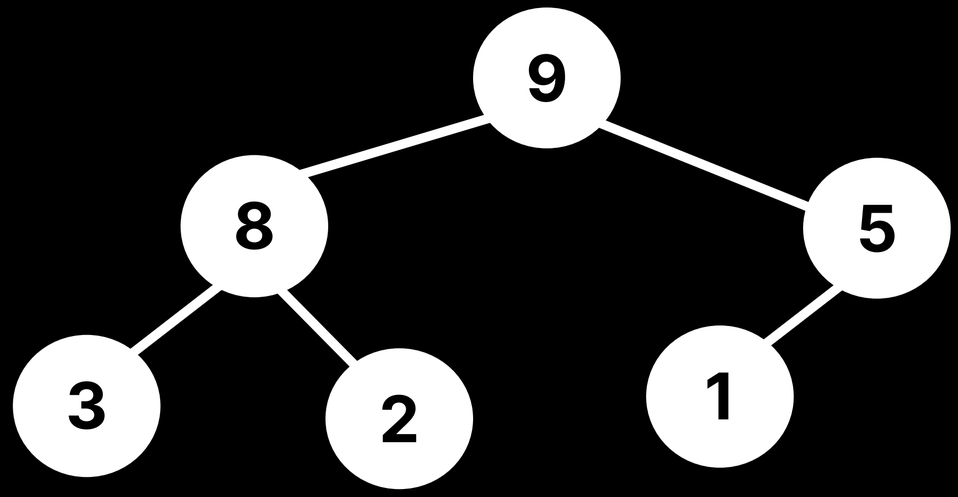


HeapSort

Maks-heap

1. Lag en maks-heap
2. Slett roten, og putt det bakerst i arrayet

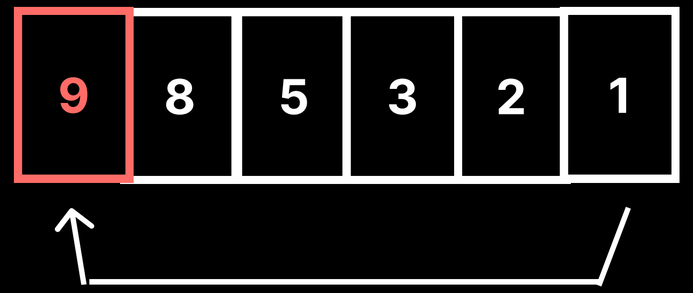


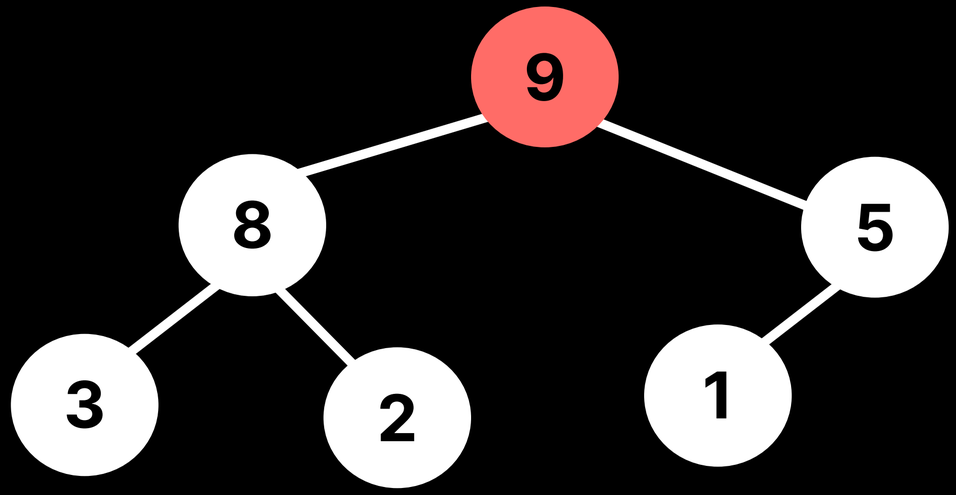


HeapSort

Maks-heap

1. Lag en maks-heap
2. Slett roten

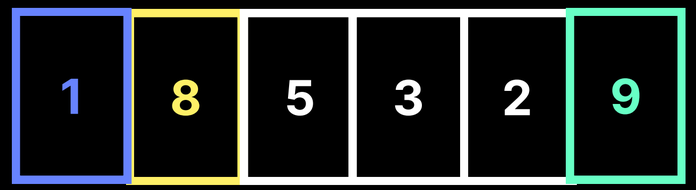


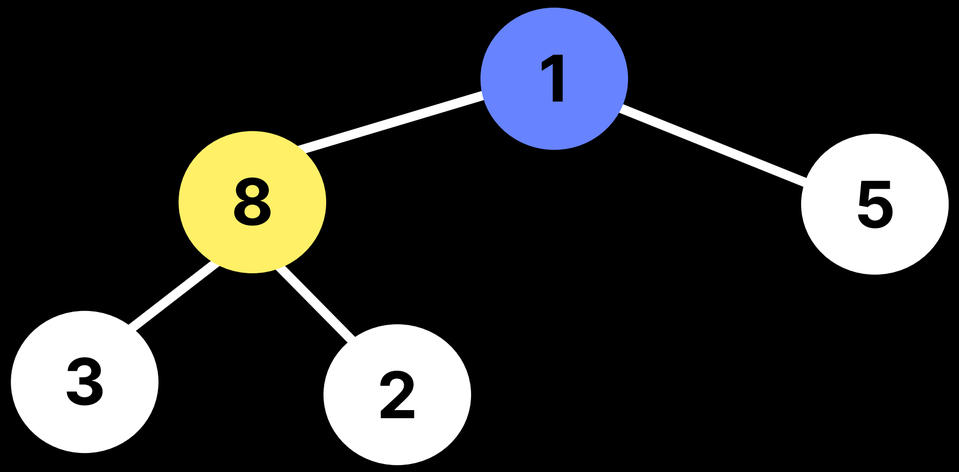


HeapSort

Maks-heap

1. Lag en maks-heap
2. Slett roten, husk bubble down
3. Putt roten bakerst i arrayet

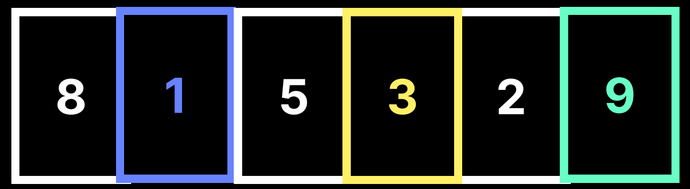


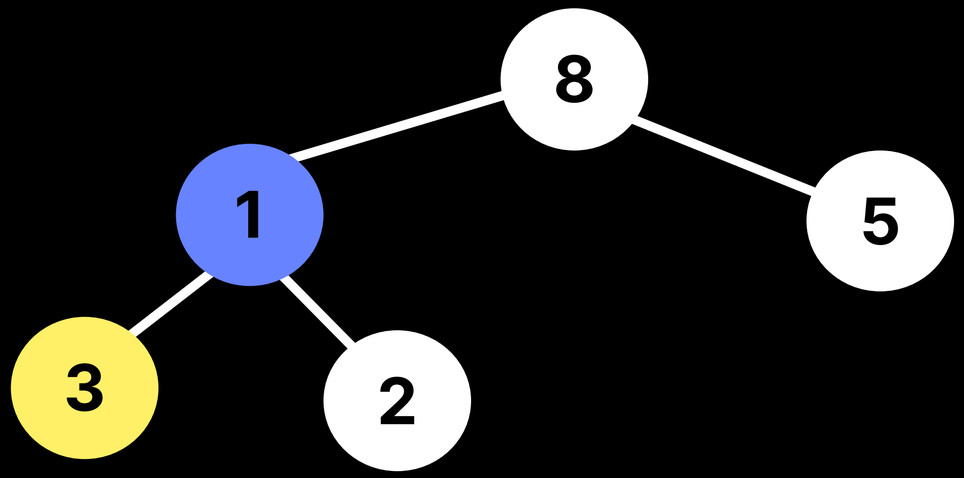


HeapSort

Maks-heap

1. Lag en maks-heap
2. Slett roten, husk bubble down
3. Putt roten bakerst i arrayet

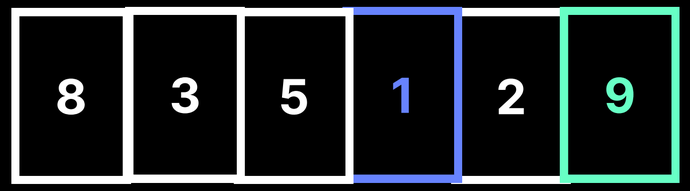


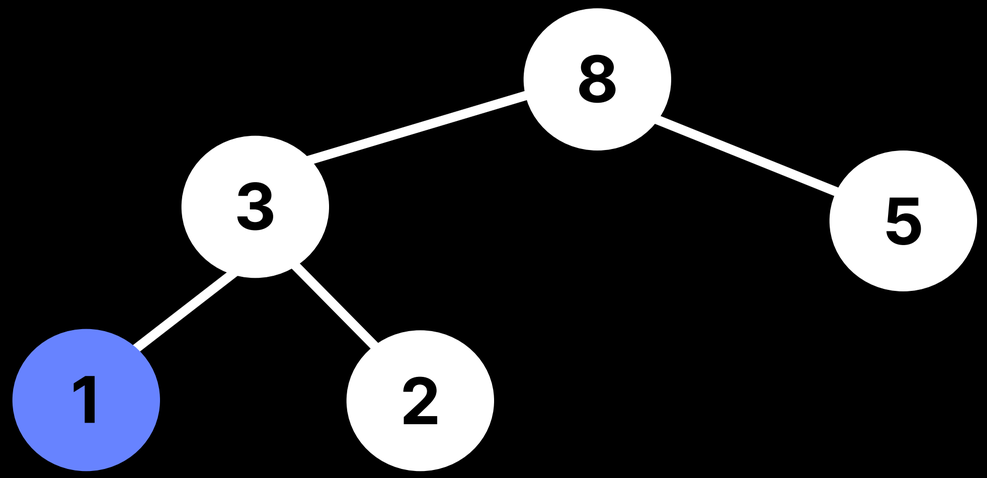


HeapSort

Maks-heap

1. Lag en maks-heap
2. Slett roten, husk bubble down
3. Putt roten bakerst i arrayet

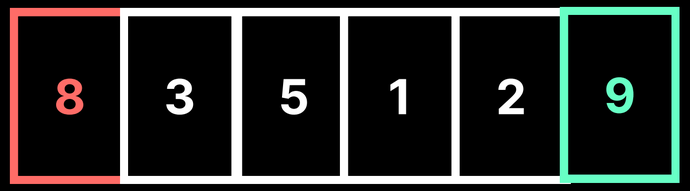


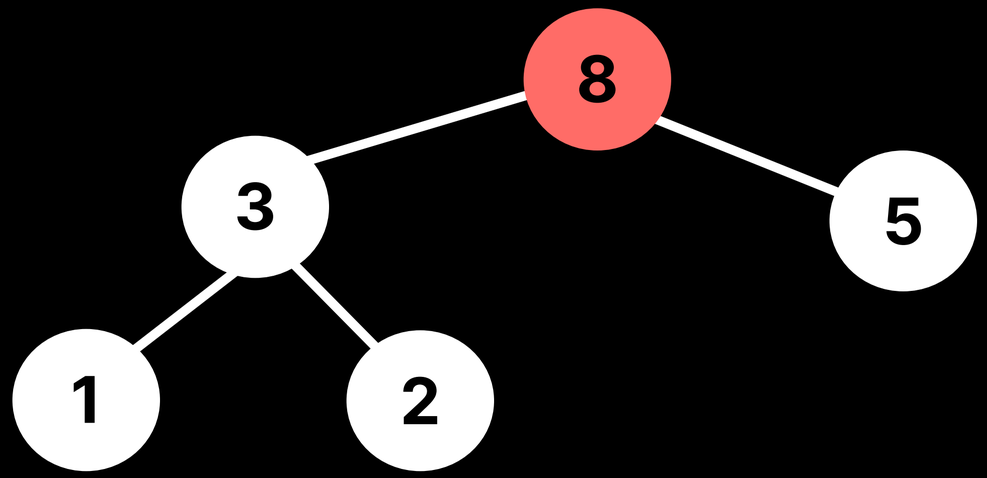


HeapSort

Maks-heap

1. Lag en maks-heap
2. Slett roten, husk bubble down
3. Putt roten bakerst i arrayet

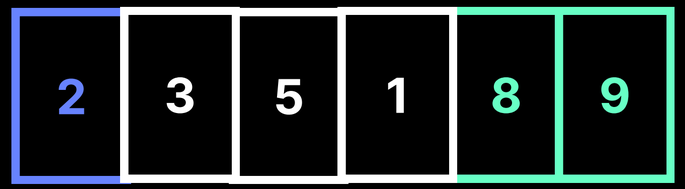


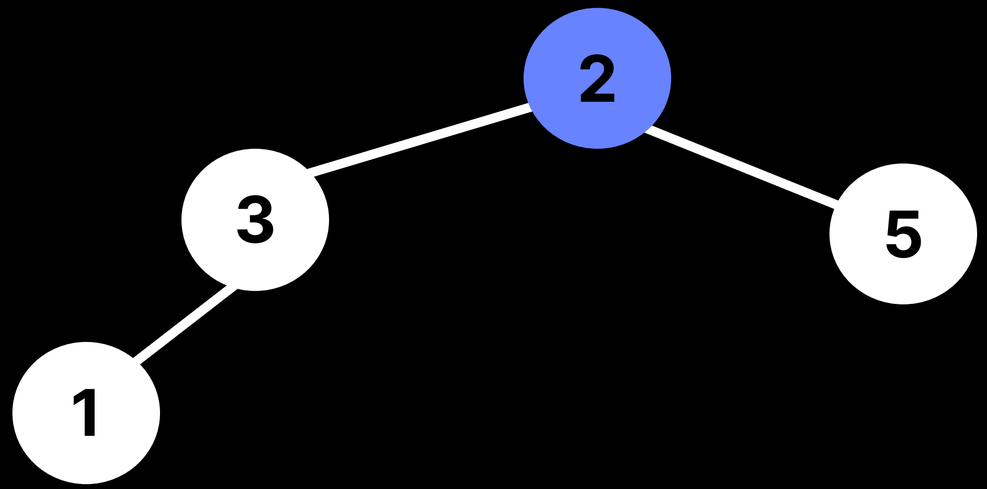


HeapSort

Maks-heap

1. Lag en maks-heap
2. Slett roten, husk bubble down
3. Putt roten bakerst i arrayet

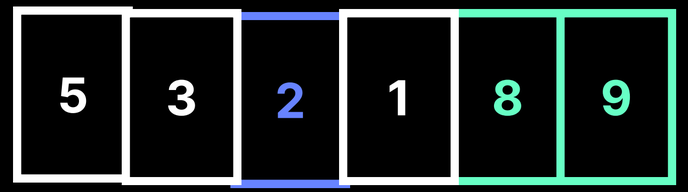


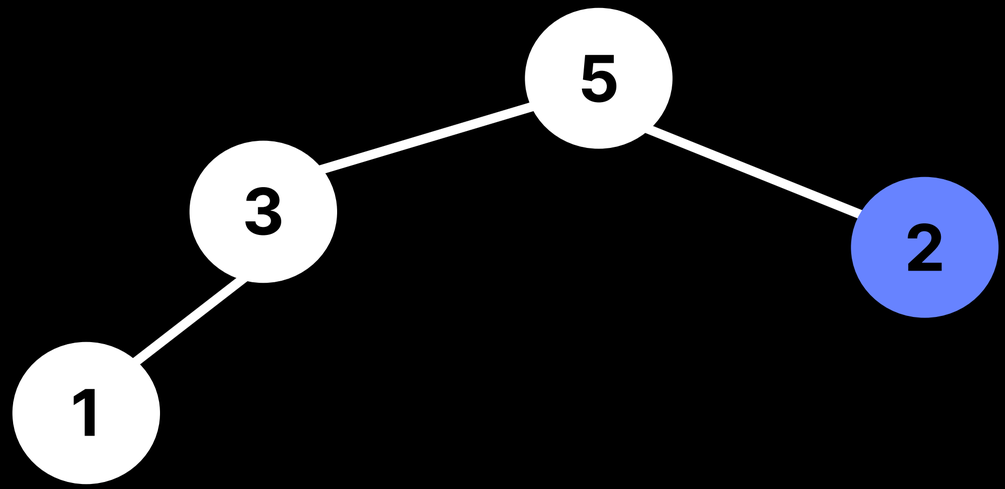


HeapSort

Maks-heap

1. Lag en maks-heap
2. Slett roten, husk bubble down
3. Putt roten bakerst i arrayet

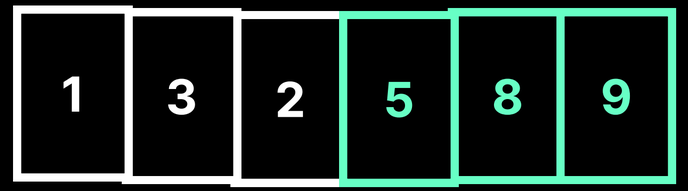


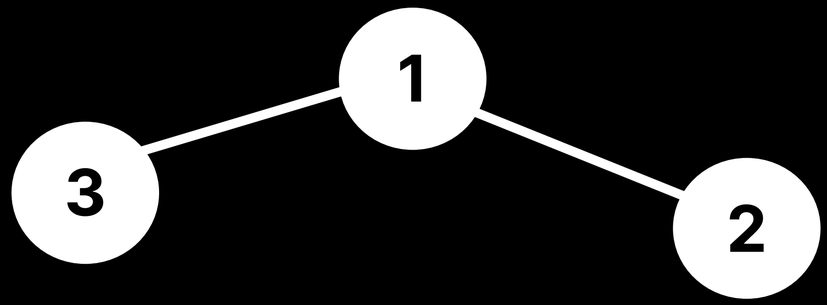


HeapSort

Maks-heap

1. Lag en maks-heap
2. Slett roten, husk bubble down
3. Putt roten bakerst i arrayet

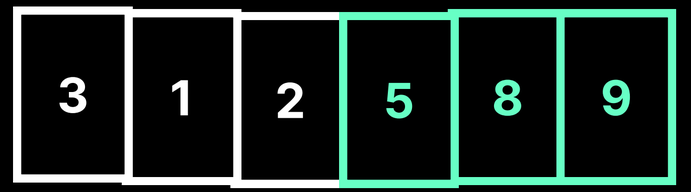


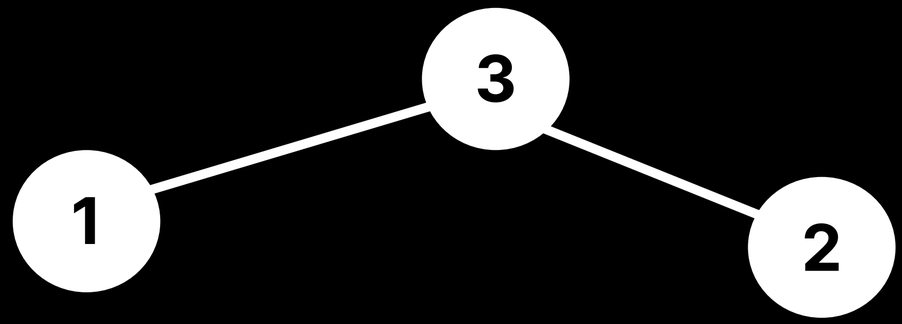


HeapSort

Maks-heap

1. Lag en maks-heap
2. Slett roten, husk bubble down
3. Putt roten bakerst i arrayet

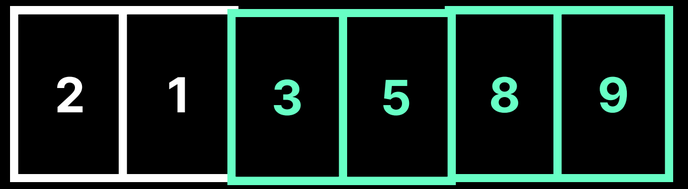


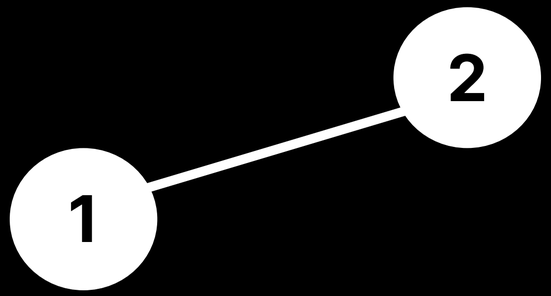


HeapSort

Maks-heap

1. Lag en maks-heap
2. Slett roten, husk bubble down
3. Putt roten bakerst i arrayet

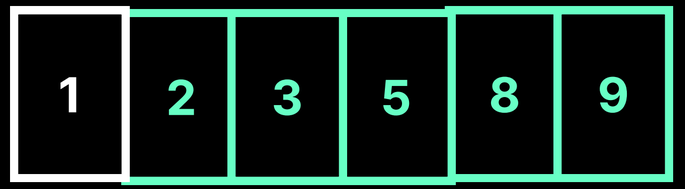




HeapSort

Maks-heap

1. Lag en maks-heap
2. Slett roten, husk bubble down
3. Putt roten bakerst i arrayet

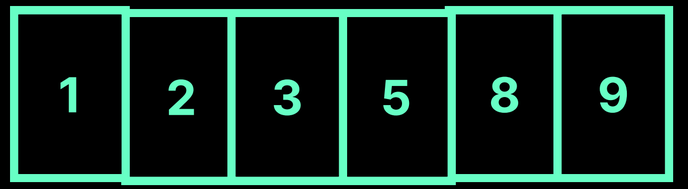




HeapSort

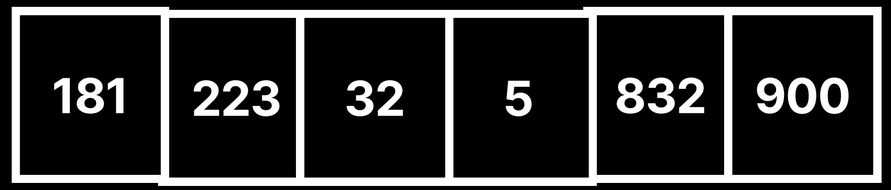
Maks-heap

1. Lag en maks-heap
2. Slett roten, husk bubble down
3. Putt roten bakerst i arrayet



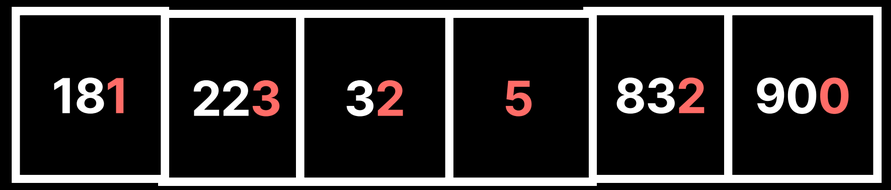
RadixSort

1. Se først på ener plassen



RadixSort

1. Se først på ener plassen



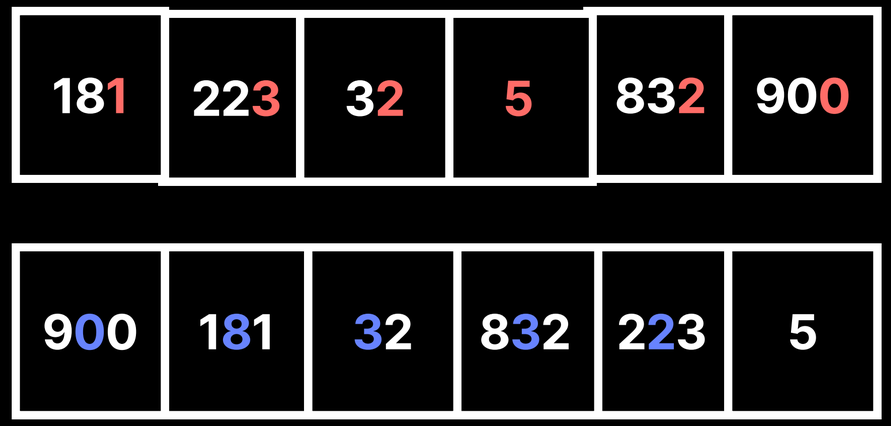
RadixSort

1. Se først på ener plassen



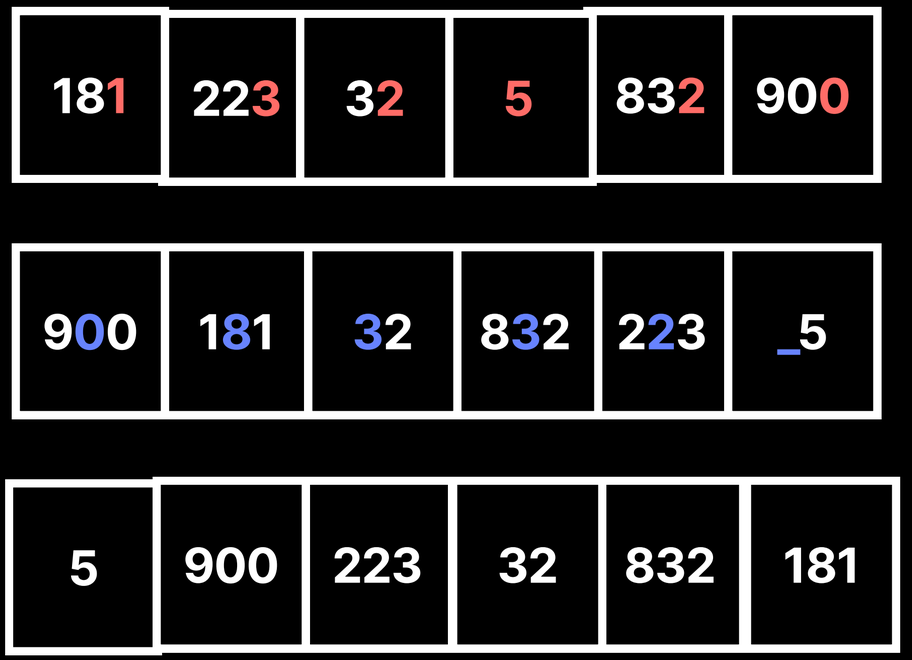
RadixSort

1. Se først på ener plassen
2. Så se på tier plassen



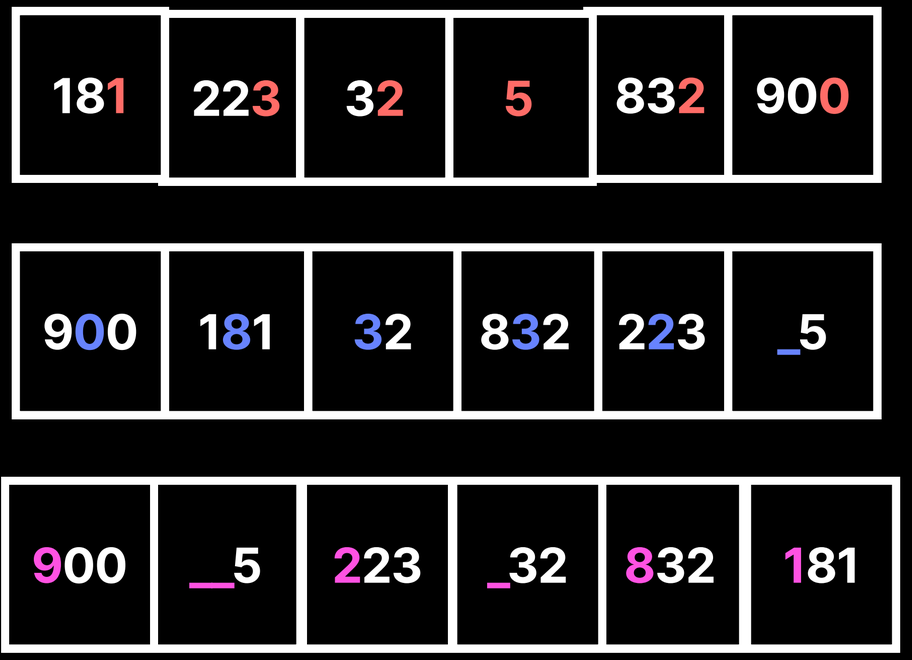
RadixSort

1. Se først på ener plassen
2. Så se på tier plassen



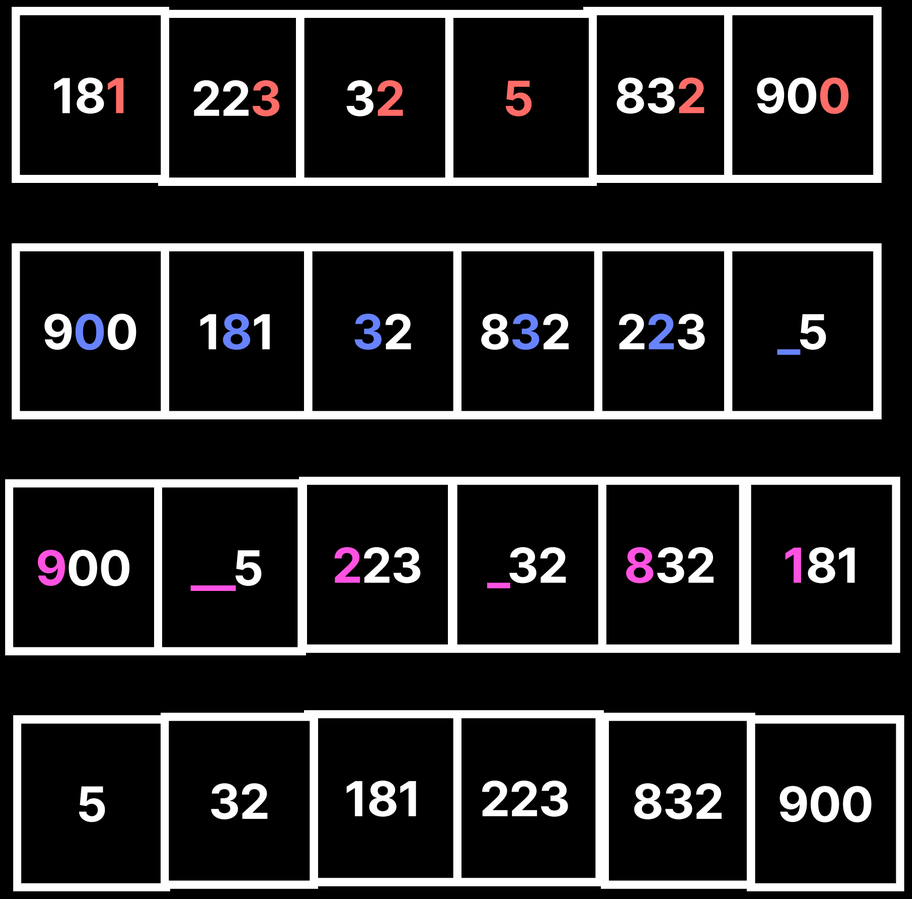
RadixSort

1. Se først på ener plassen
2. Så se på tier plassen
3. Se til slutt på 100 plassen



RadixSort

1. Se først på ener plassen
2. Så se på tier plassen
3. Se til slutt på 100 plassen



Sti vs. vei

Hva er en sti?

Sti vs. vei

Hva er en sti?
Sekvens av noder, forbundet av kanter, der ingen **noder** gjentas

Sti vs. vei

Hva er en sti?
Sekvens av noder, forbundet av kanter, der ingen **noder** gjentas

Hva er en vei?

Sti vs. vei

Hva er en sti?

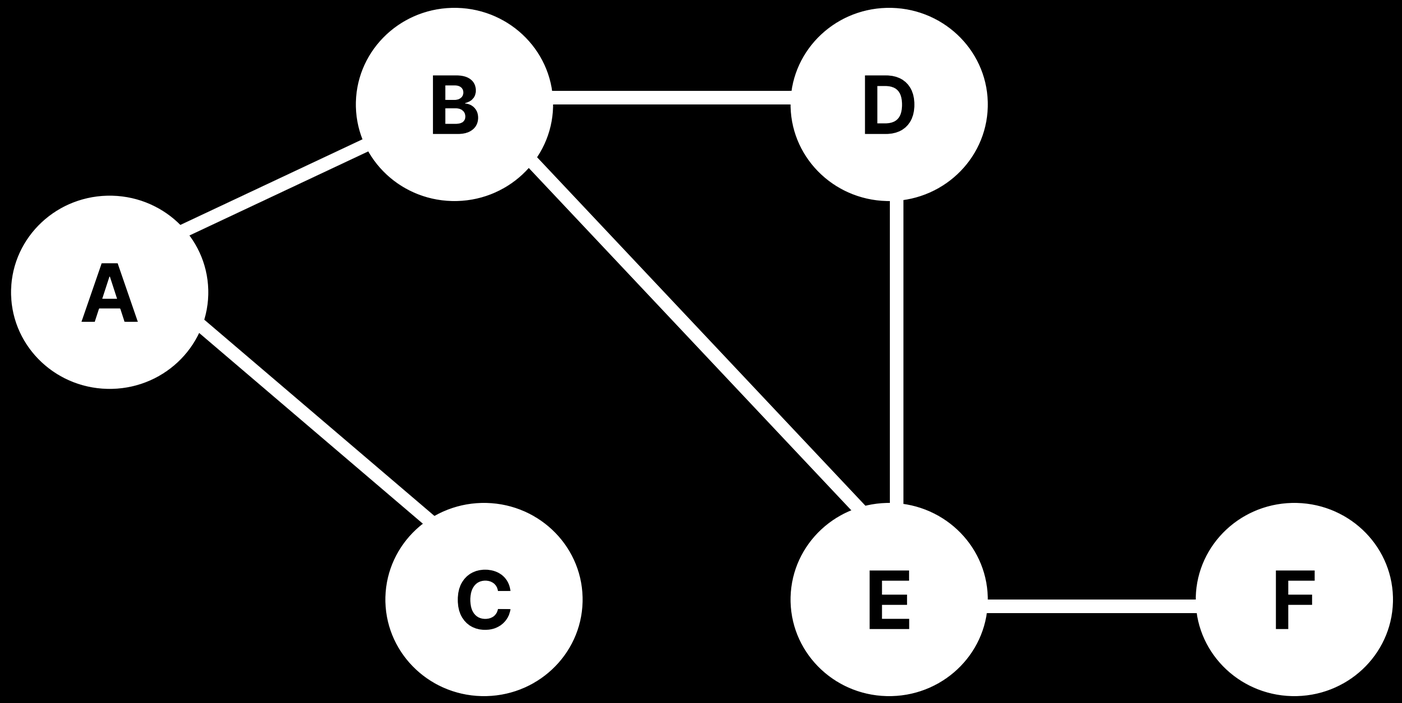
Sekvens av noder, forbundet av kanter, der ingen **noder** gjentas

Hva er en vei?

Sekvens av noder, forbundet av kanter, der ingen **kanter** gjentas

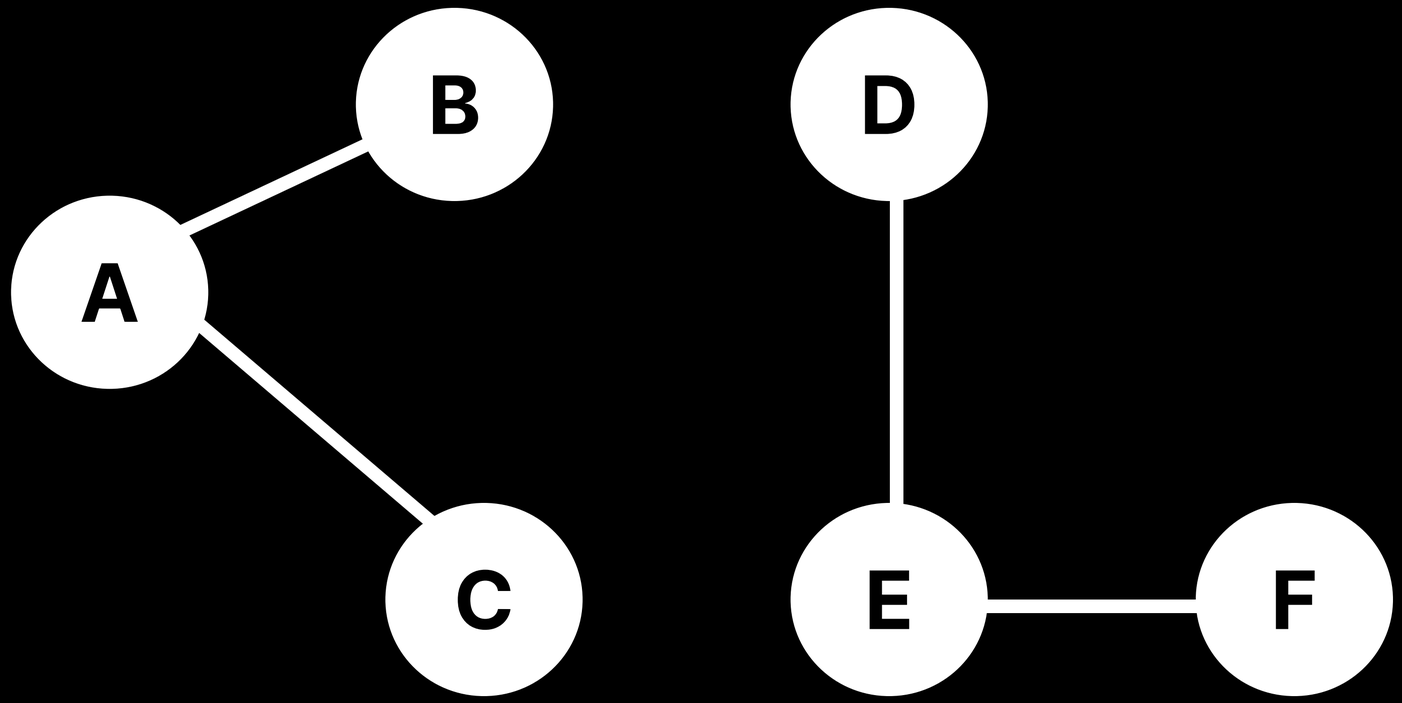
Sammenhengende grafer

det finnes en sti mellom alle par av noder



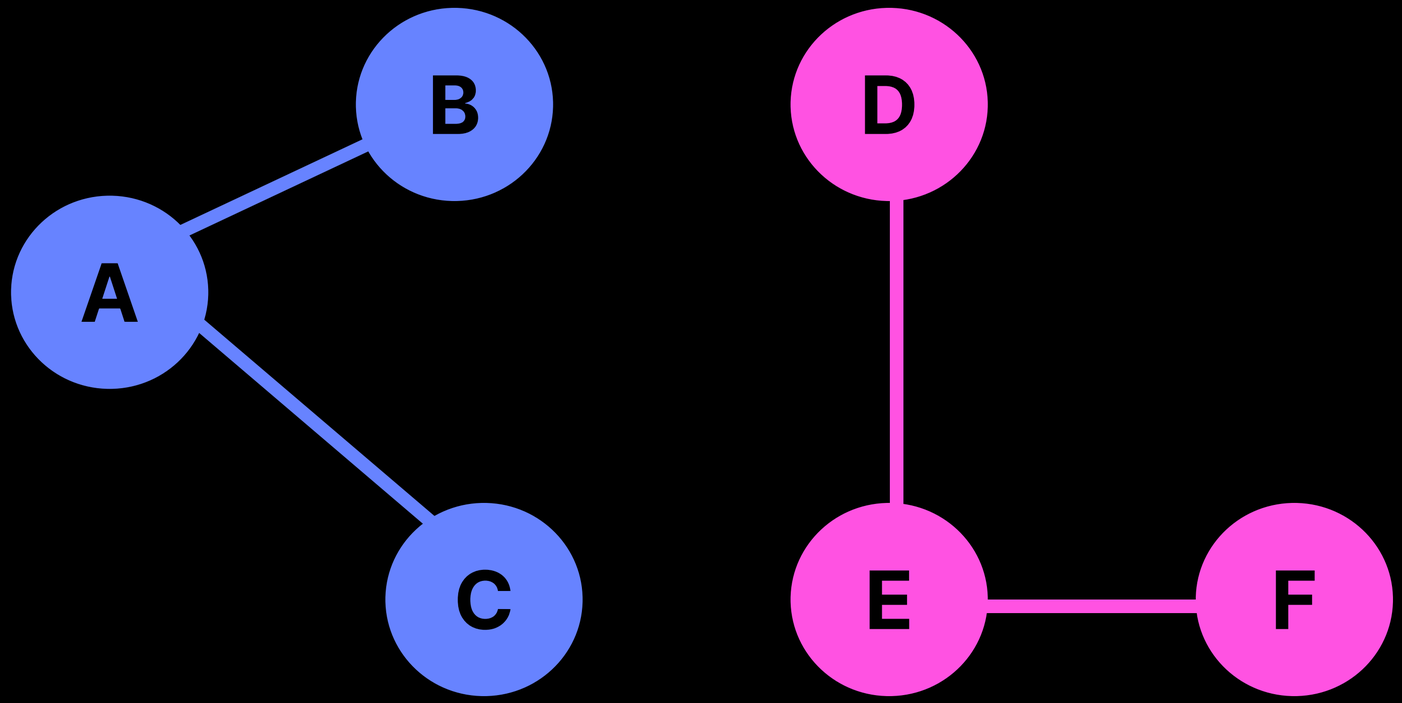
Ikke-sammenhengende grafer

har komponenter



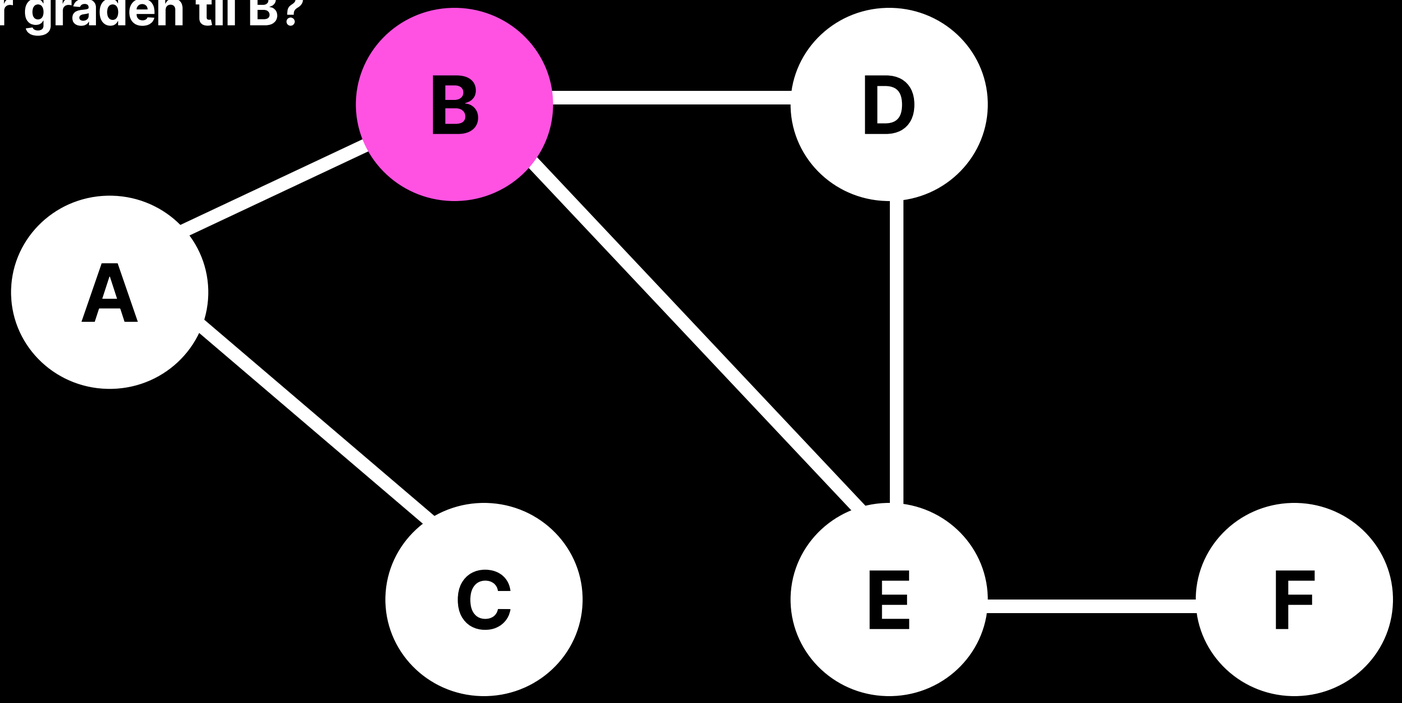
Ikke-sammenhengende grafer

har komponenter



Graden til en node

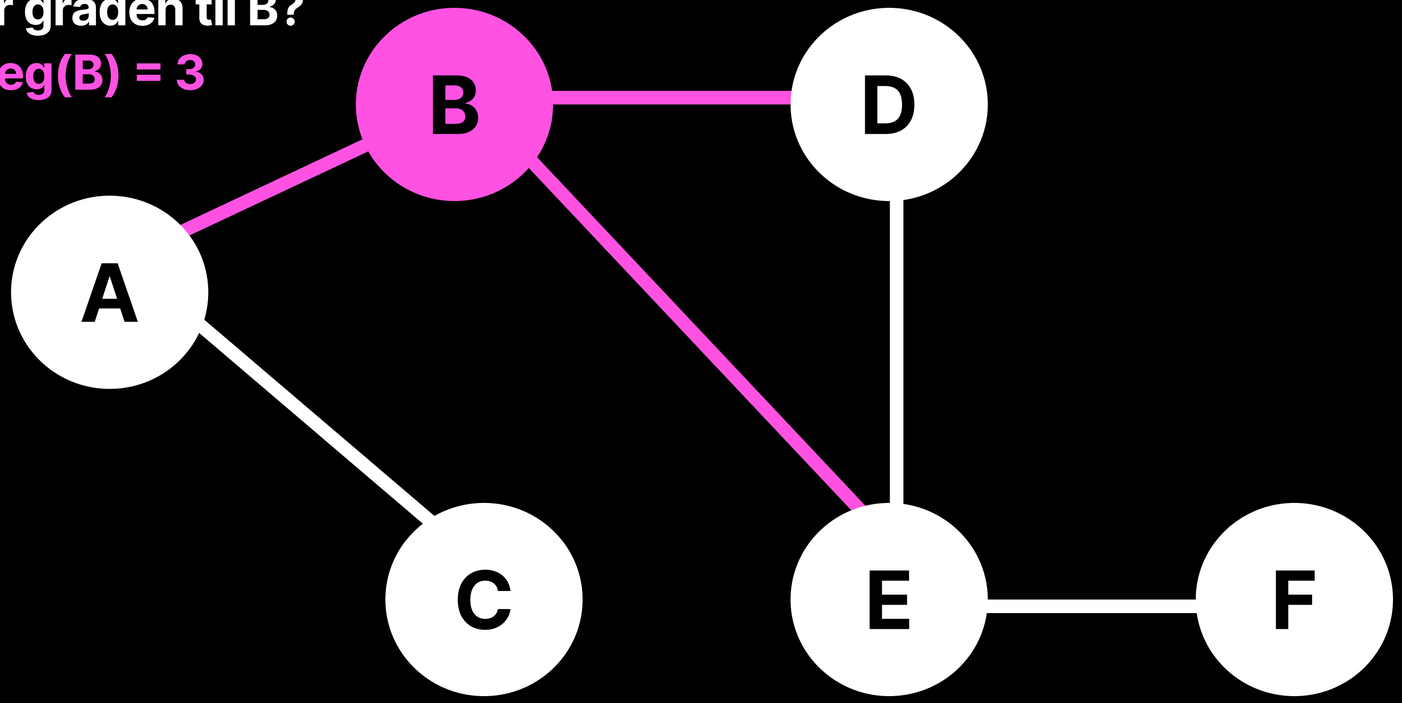
Hva er graden til B?



Graden til en node

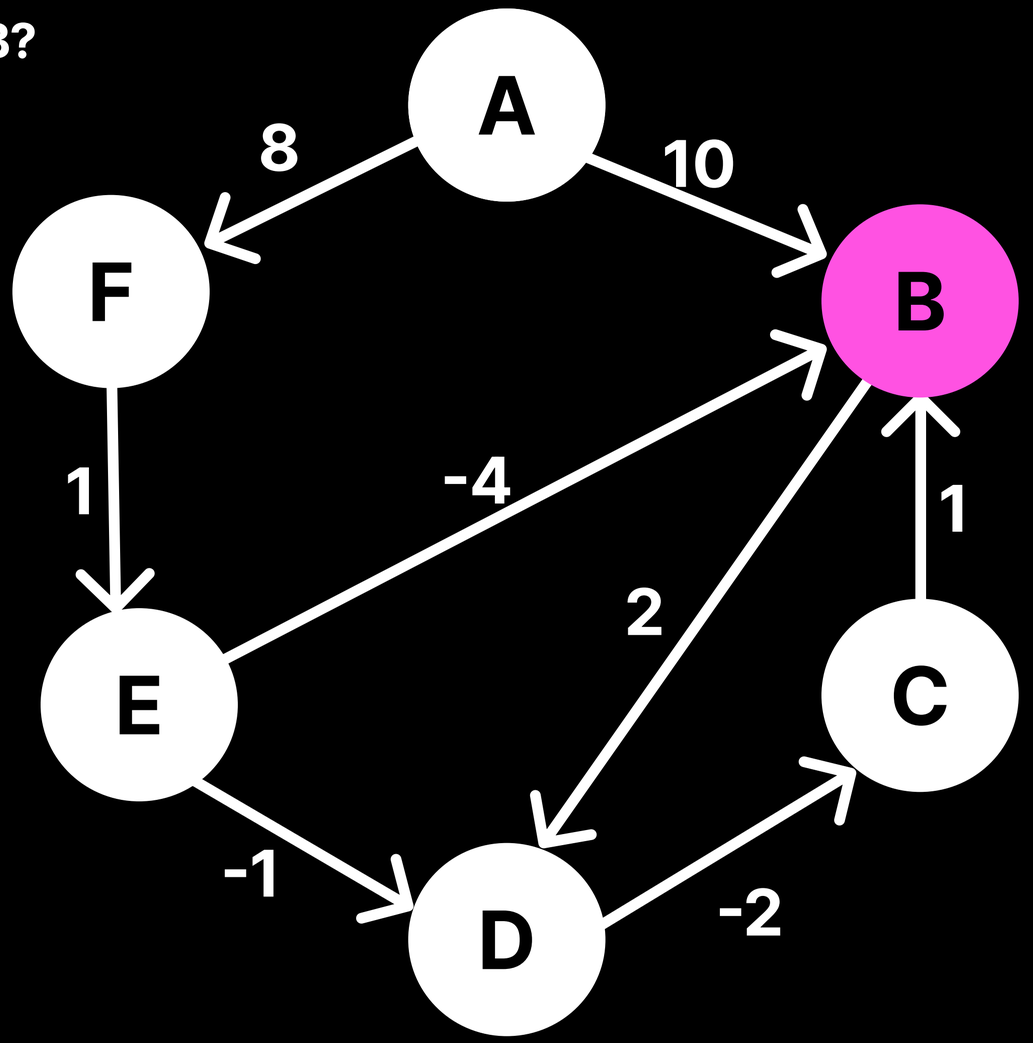
Hva er graden til B?

deg(B) = 3



Graden til en node

Hva er graden til B?

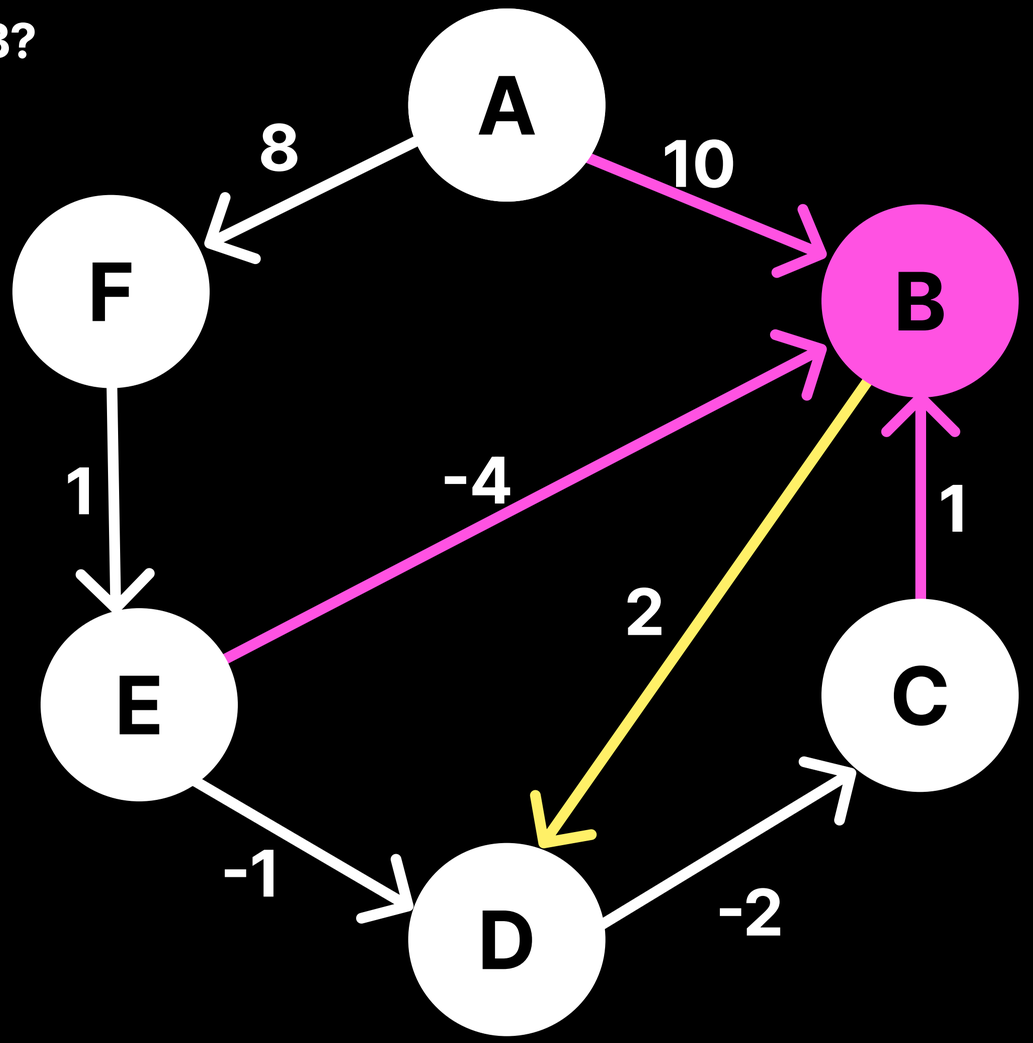


Graden til en node

Hva er graden til B?

inngrad: 3

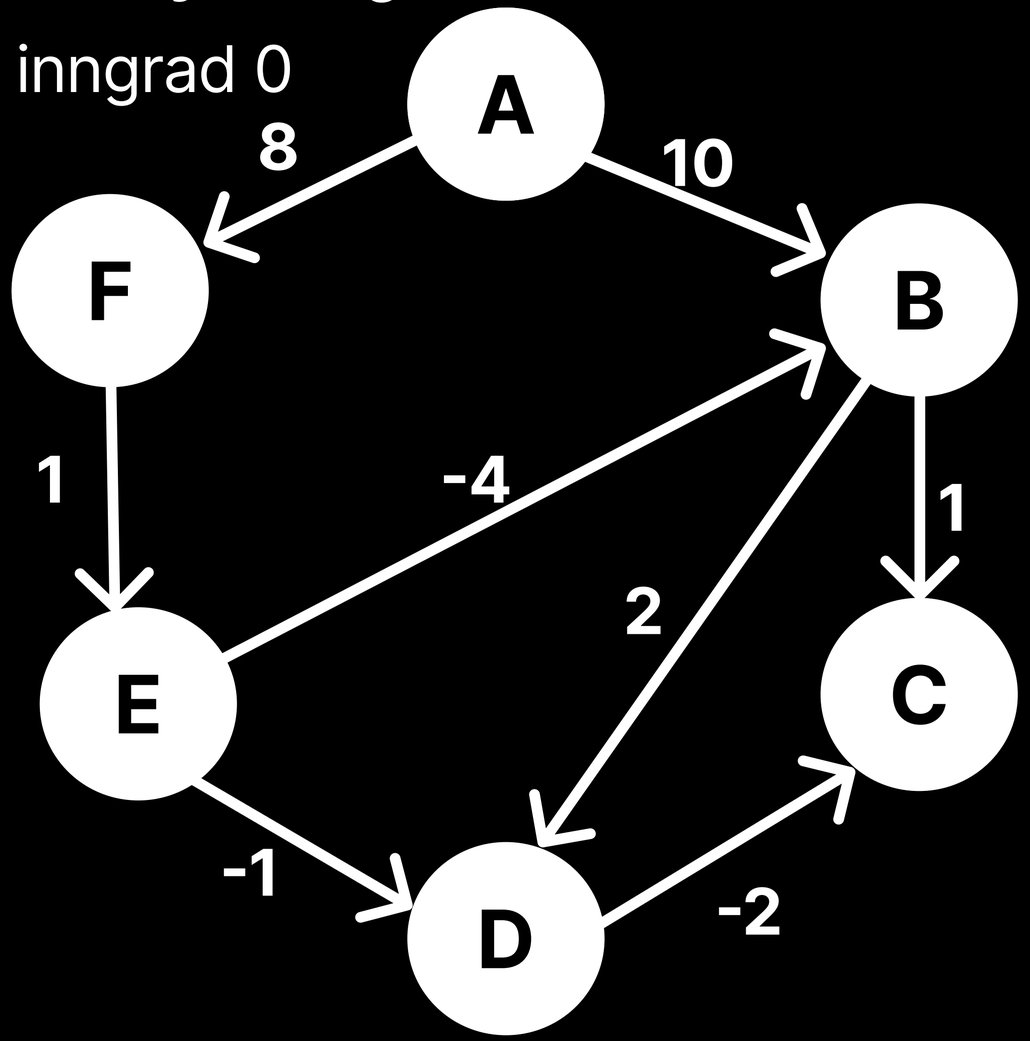
utgrad: 1



Topologisk sortering (TopSort)

Ordner nodene i en **rettet asyklisk** graf (DAG).

Starter i noden(e) med inngrad 0

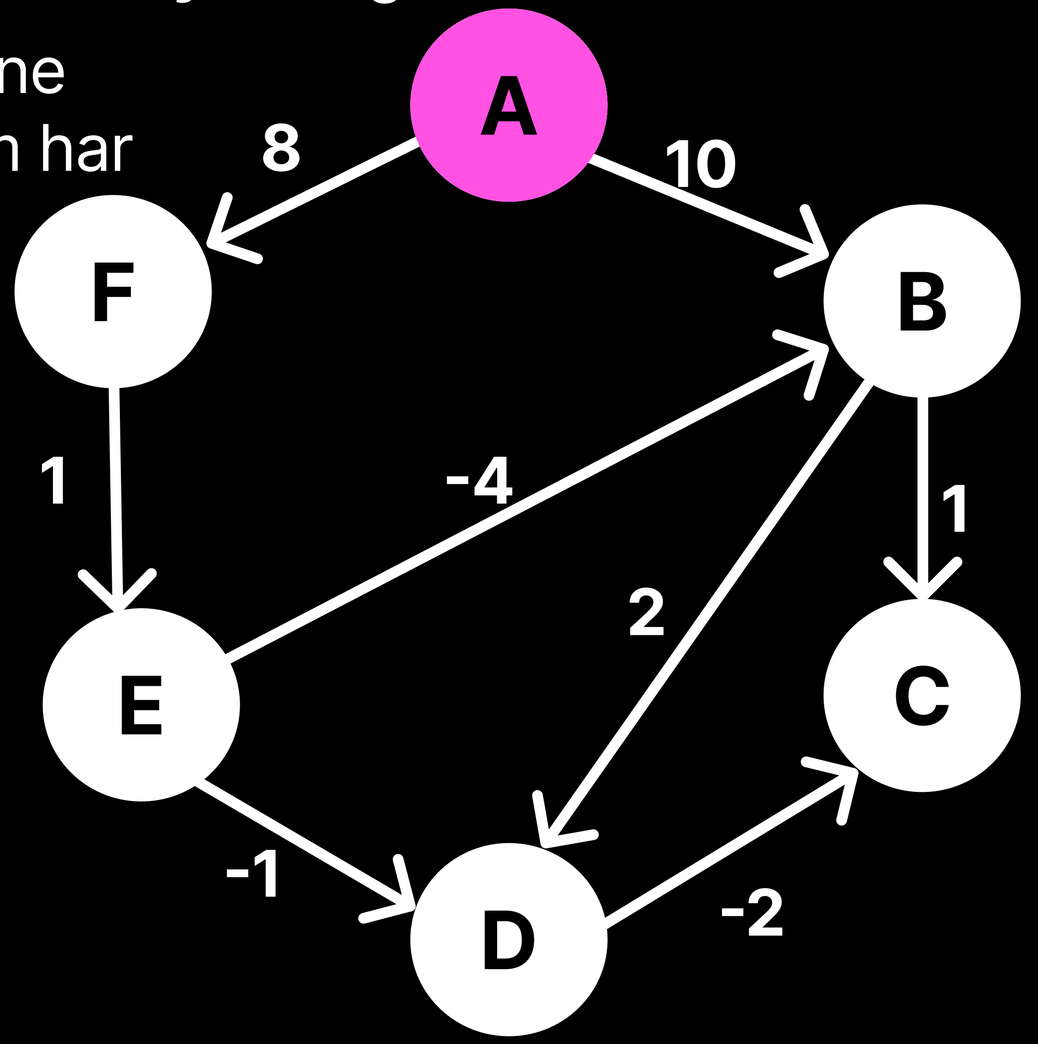


Output:

Topologisk sortering (TopSort)

Ordner nodene i en **rettet asyklisk** graf (DAG).

Fjerner kantene til denne sjekker igjen hvem som har inngrad 0 nå

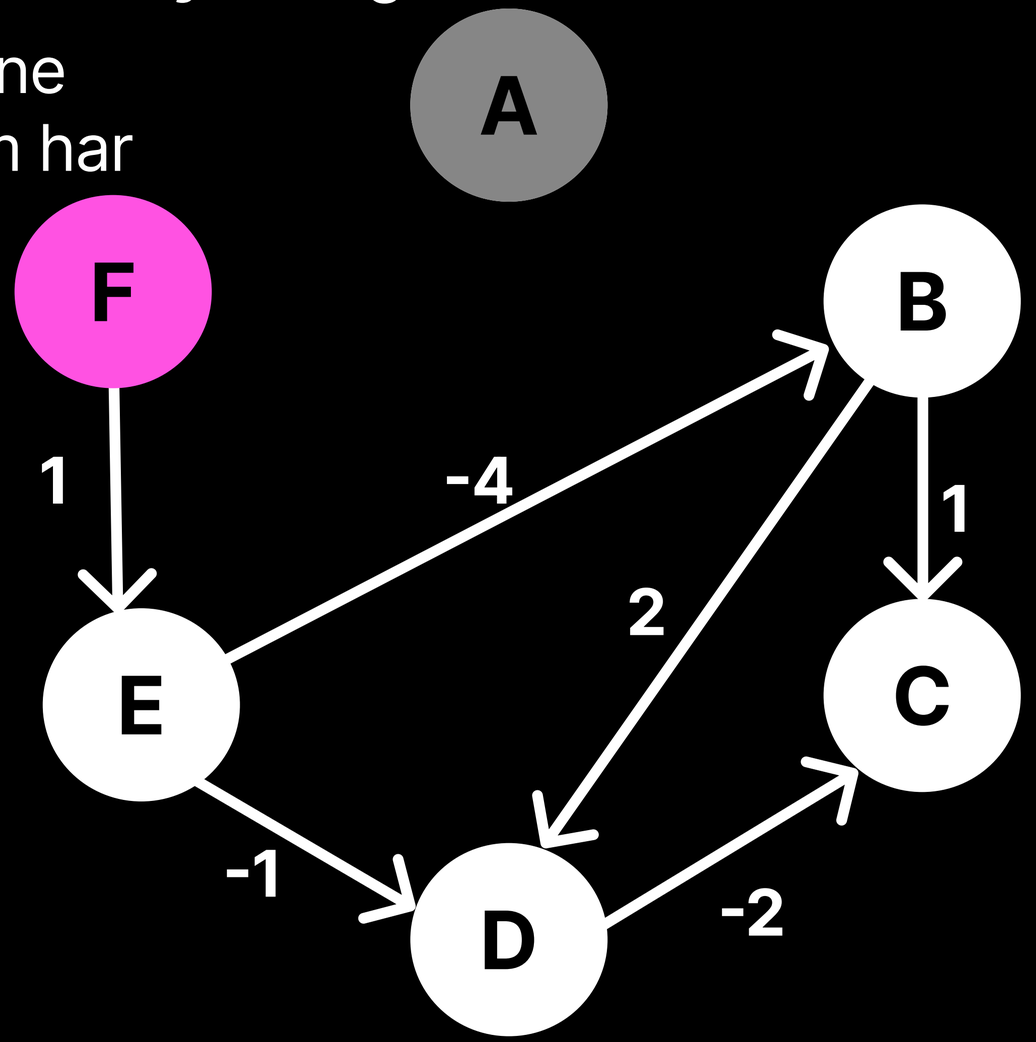


Output: A

Topologisk sortering (TopSort)

Ordner nodene i en **rettet asyklisk** graf (DAG).

Fjerner kantene til denne sjekker igjen hvem som har inngrad 0 nå

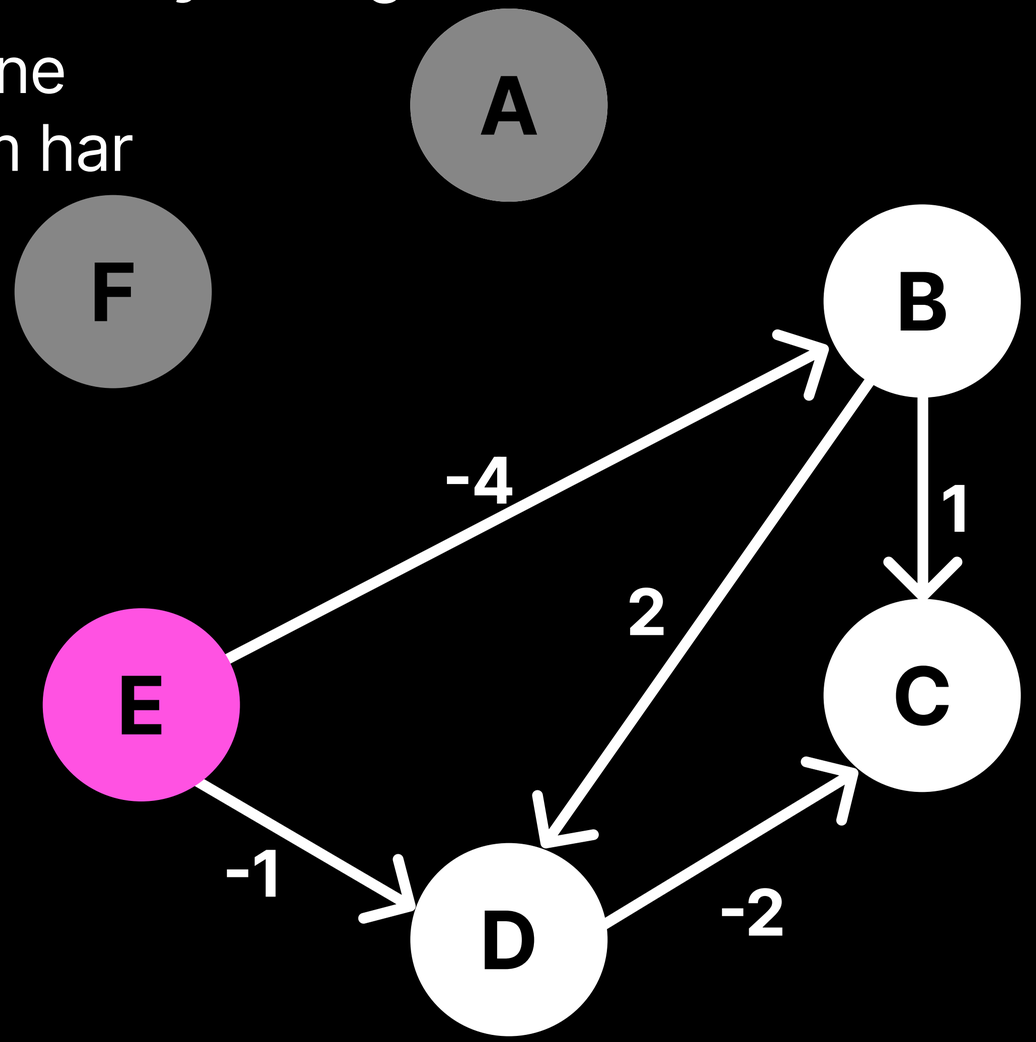


Output: A F

Topologisk sortering (TopSort)

Ordner nodene i en **rettet asyklisk** graf (DAG).

Fjerner kantene til denne sjekker igjen hvem som har inngrad 0 nå

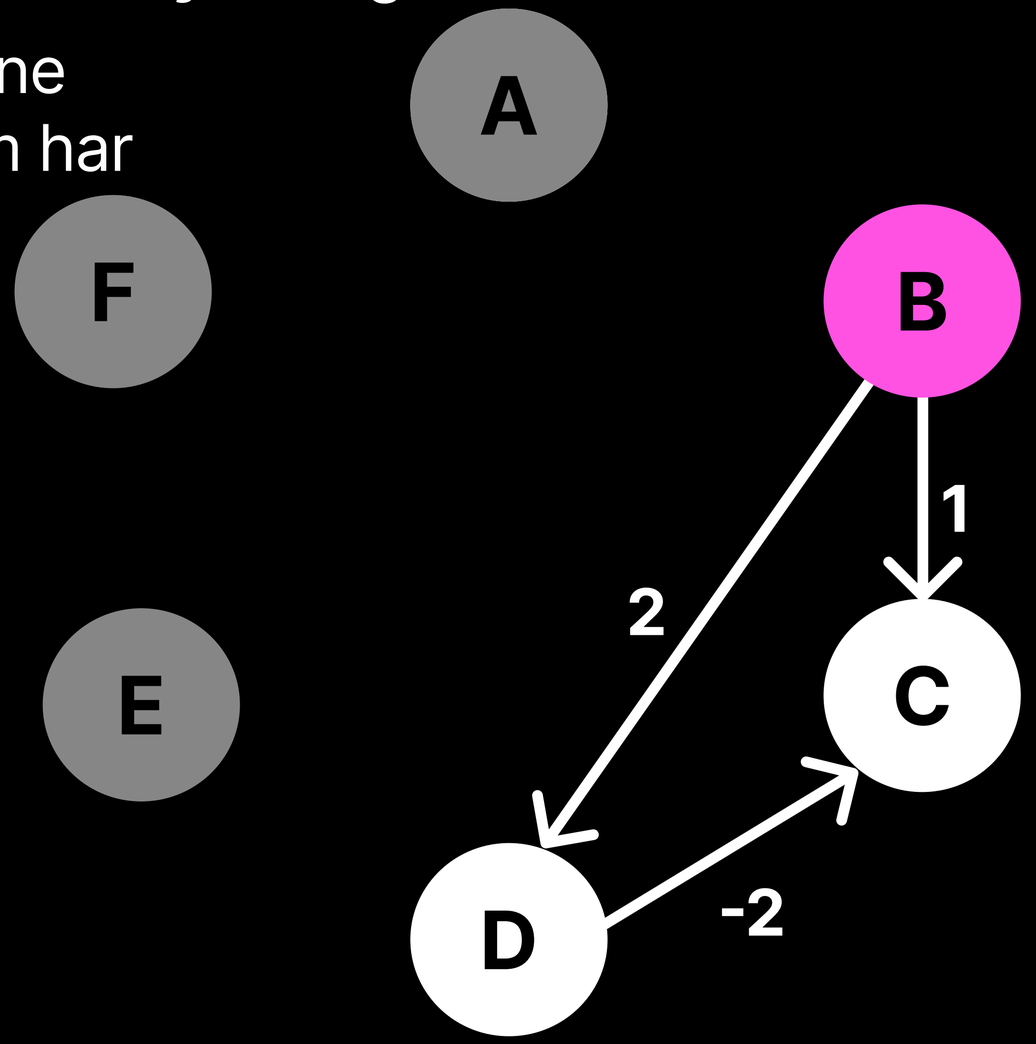


Output: A F E

Topologisk sortering (TopSort)

Ordner nodene i en **rettet asyklisk** graf (DAG).

Fjerner kantene til denne sjekker igjen hvem som har inngrad 0 nå

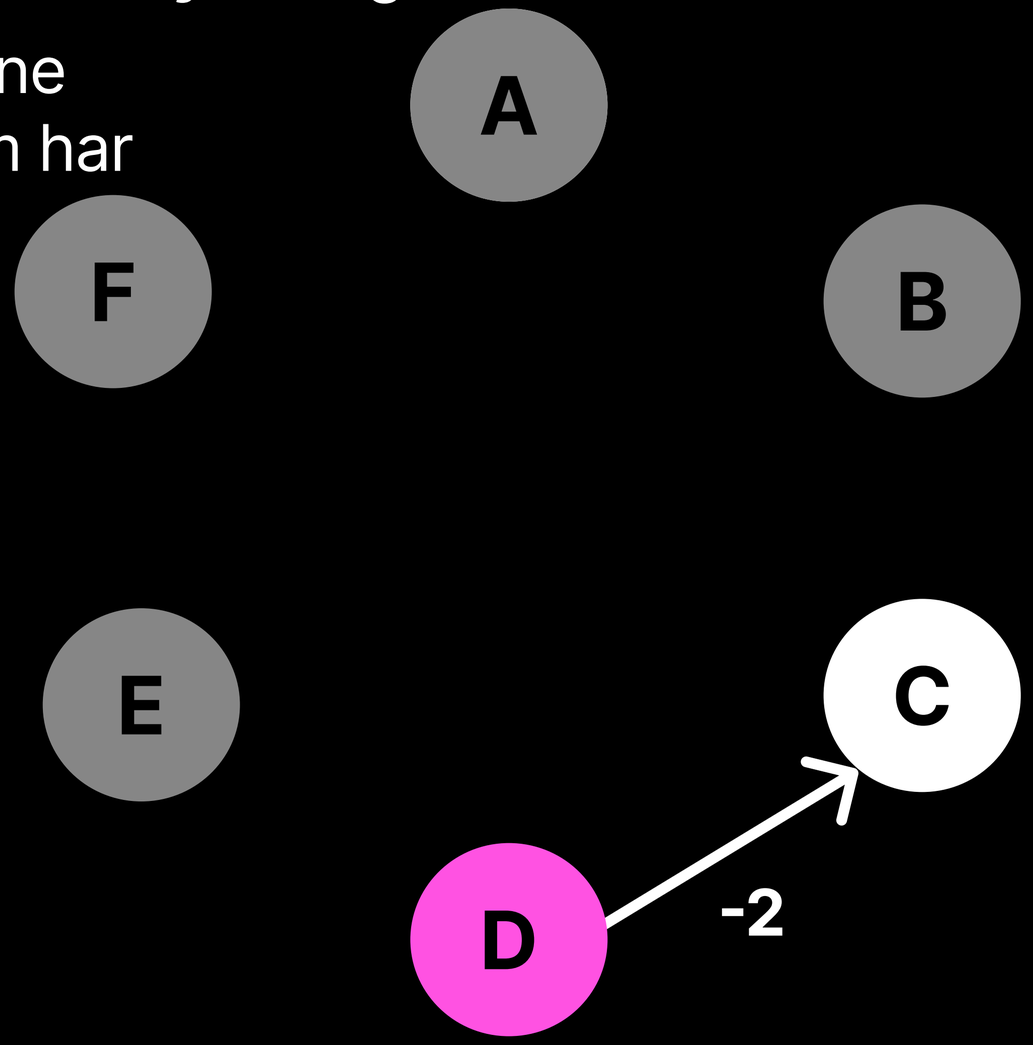


Output: A F E B

Topologisk sortering (TopSort)

Ordner nodene i en **rettet asyklisk** graf (DAG).

Fjerner kantene til denne sjekker igjen hvem som har inngrad 0 nå

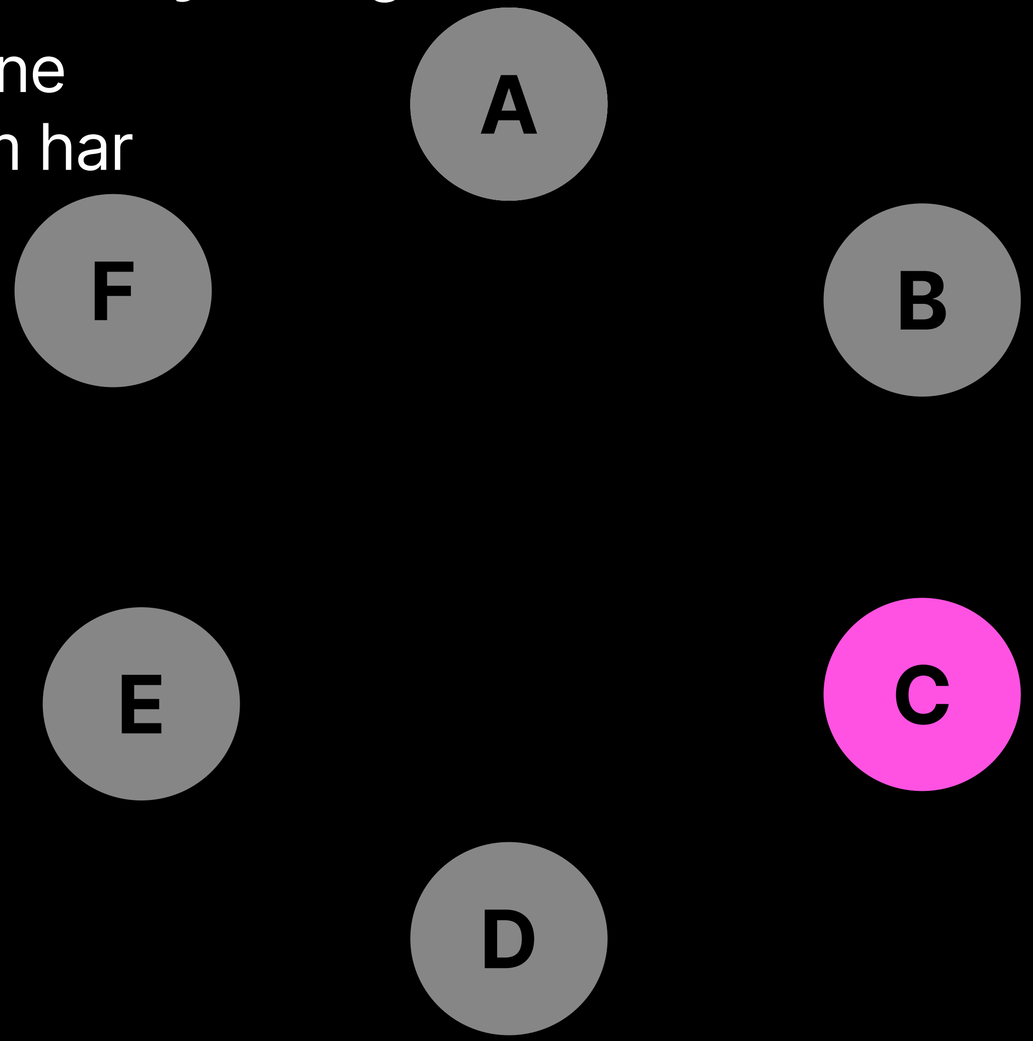


Output: A F E B D

Topologisk sortering (TopSort)

Ordner nodene i en **rettet asyklisk** graf (DAG).

Fjerner kantene til denne sjekker igjen hvem som har inngrad 0 nå

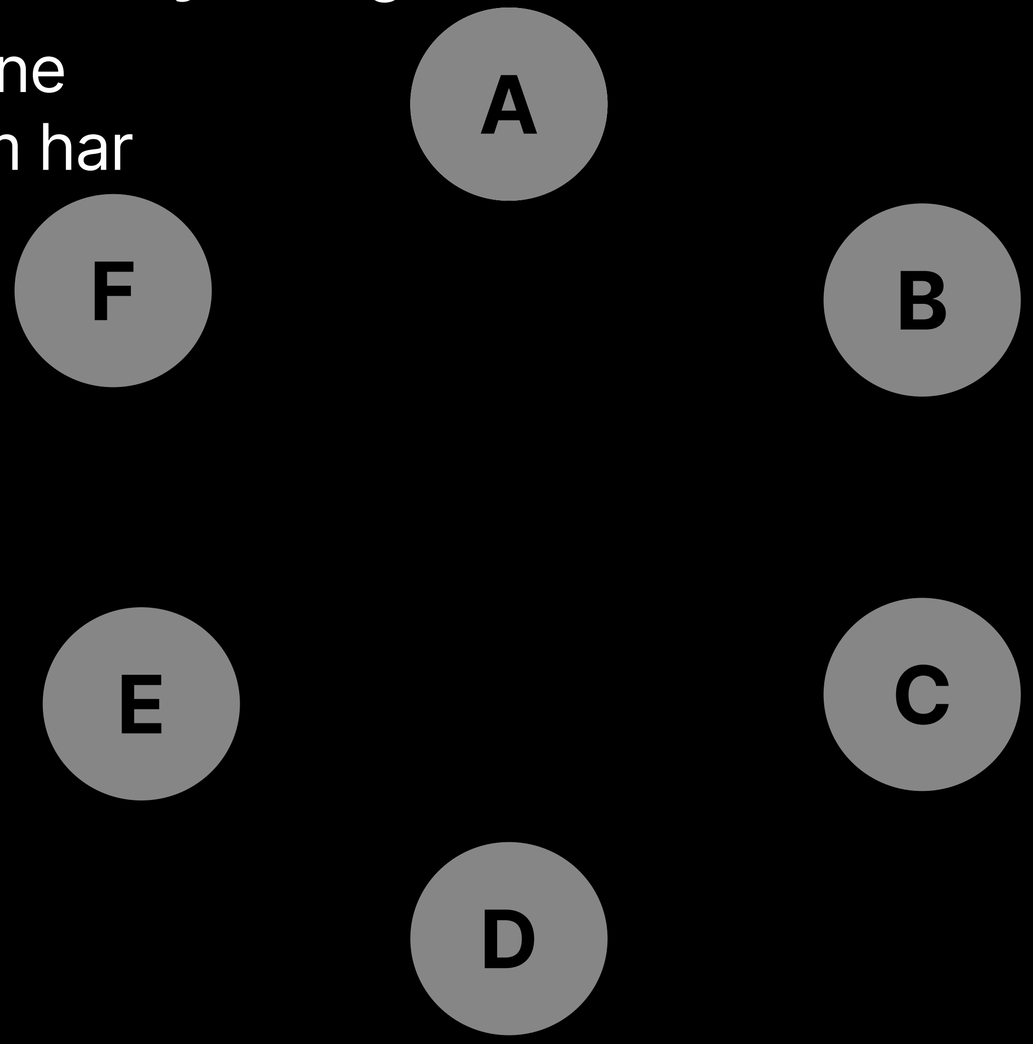


Output: A F E B D C

Topologisk sortering (TopSort)

Ordner nodene i en **rettet asyklisk** graf (DAG).

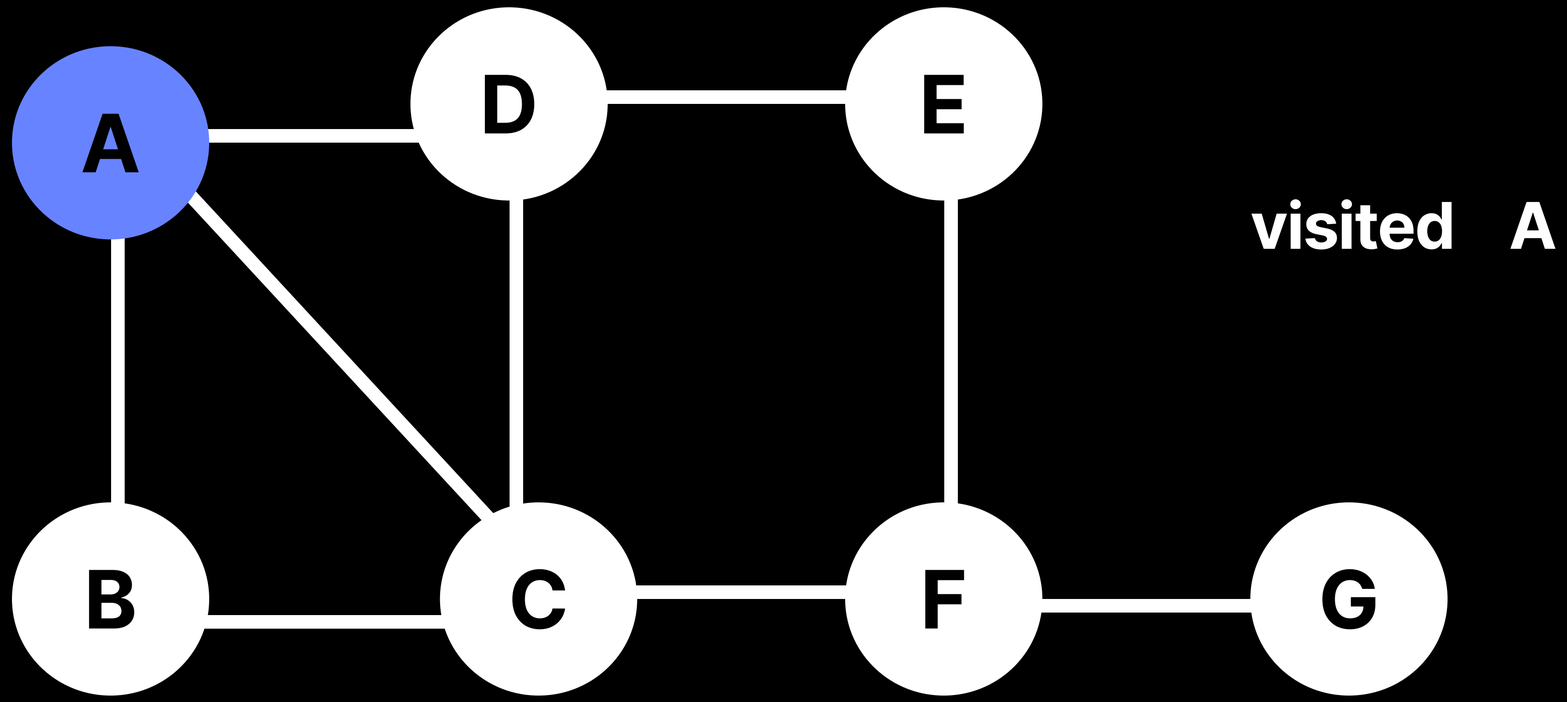
Fjerner kantene til denne sjekker igjen hvem som har inngrad 0 nå



Output: A F E B D C

Dybde-først søk (DFS)

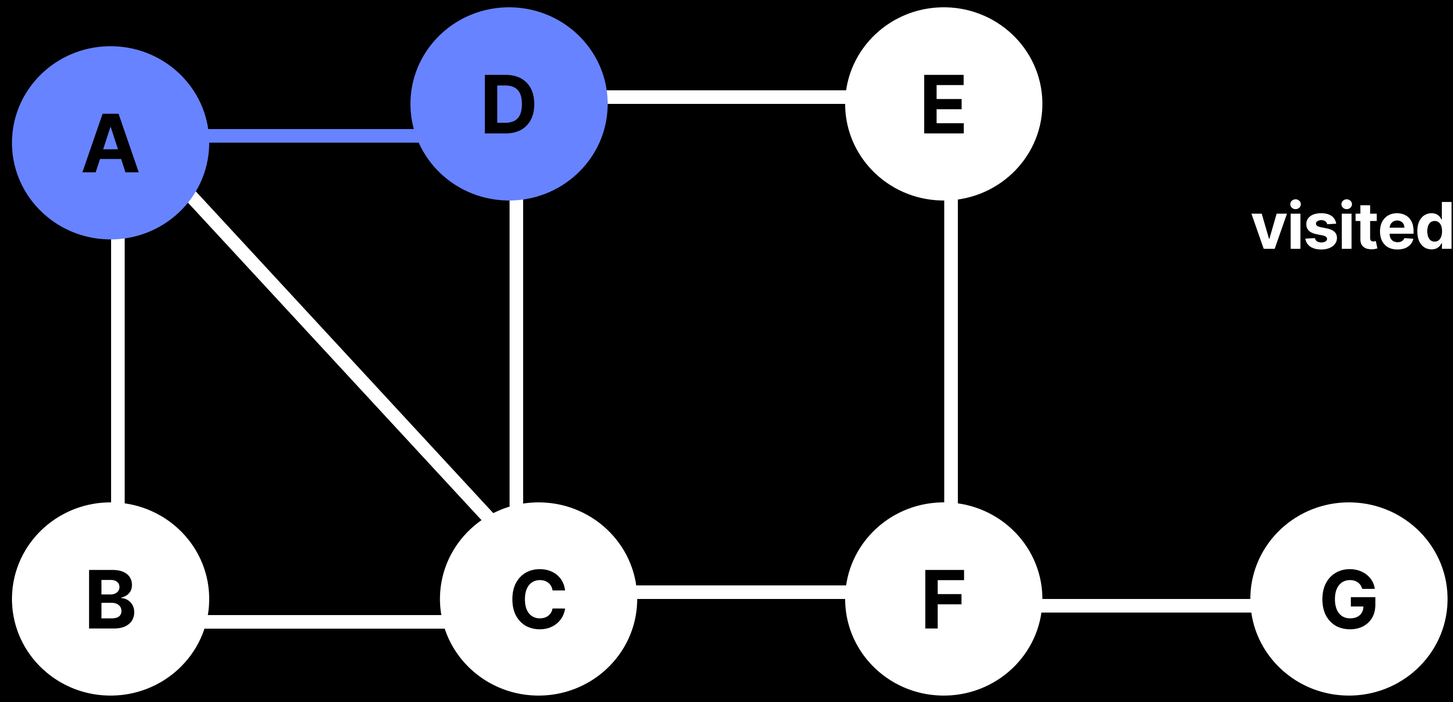
Beveger oss i dybden, dvs. vi prøver å gå så langt bort fra startnoden som mulig



visited A

Dybde-først søk (DFS)

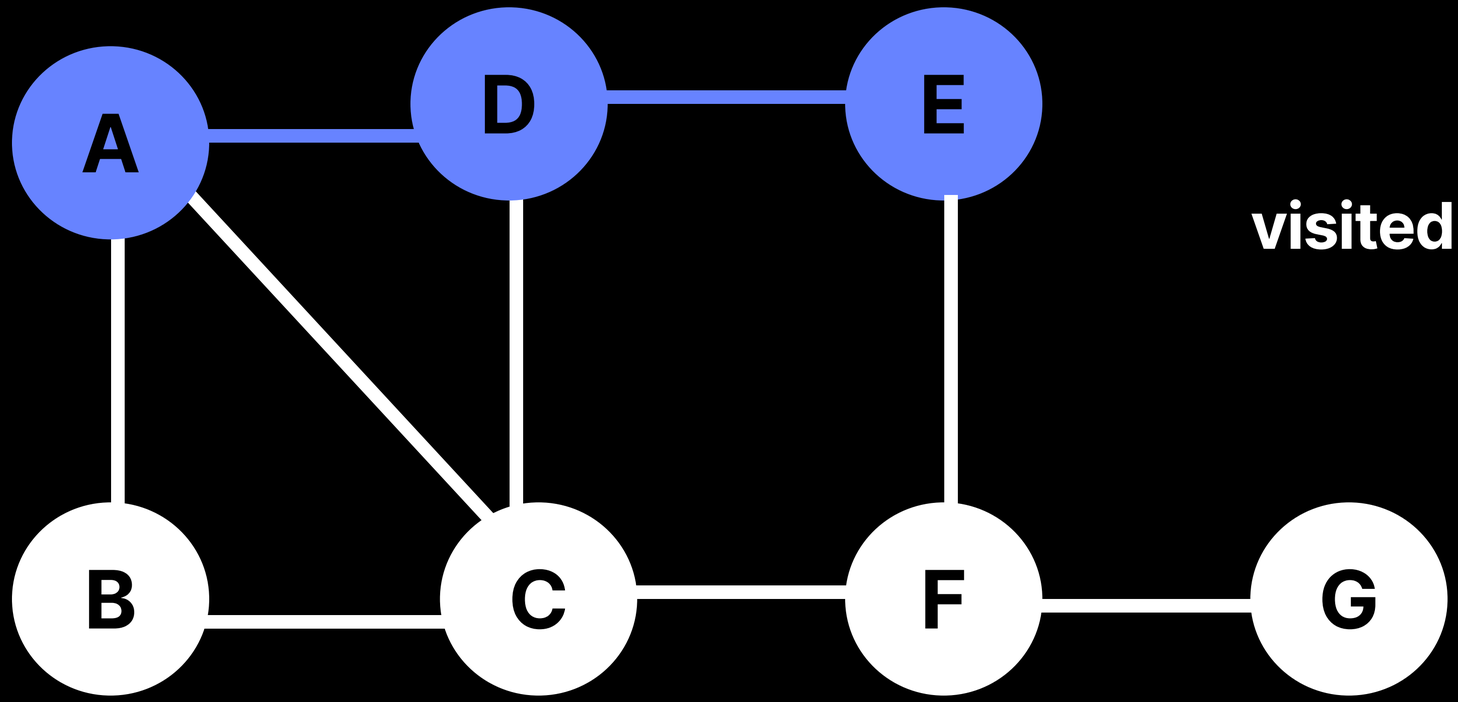
Beveger oss i dybden, dvs. vi prøver å gå så langt bort fra startnoden som mulig



visited A D

Dybde-først søk (DFS)

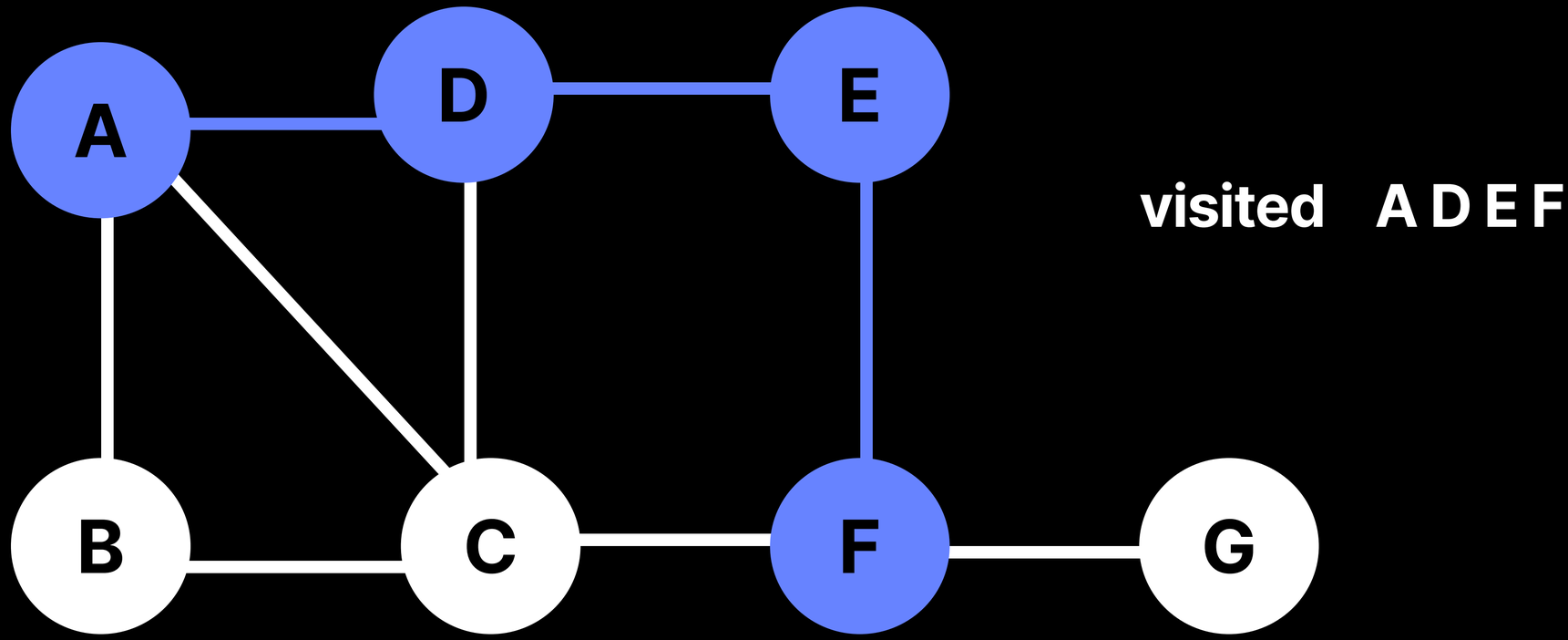
Beveger oss i dybden, dvs. vi prøver å gå så langt bort fra startnoden som mulig



visited **A D E**

Dybde-først søk (DFS)

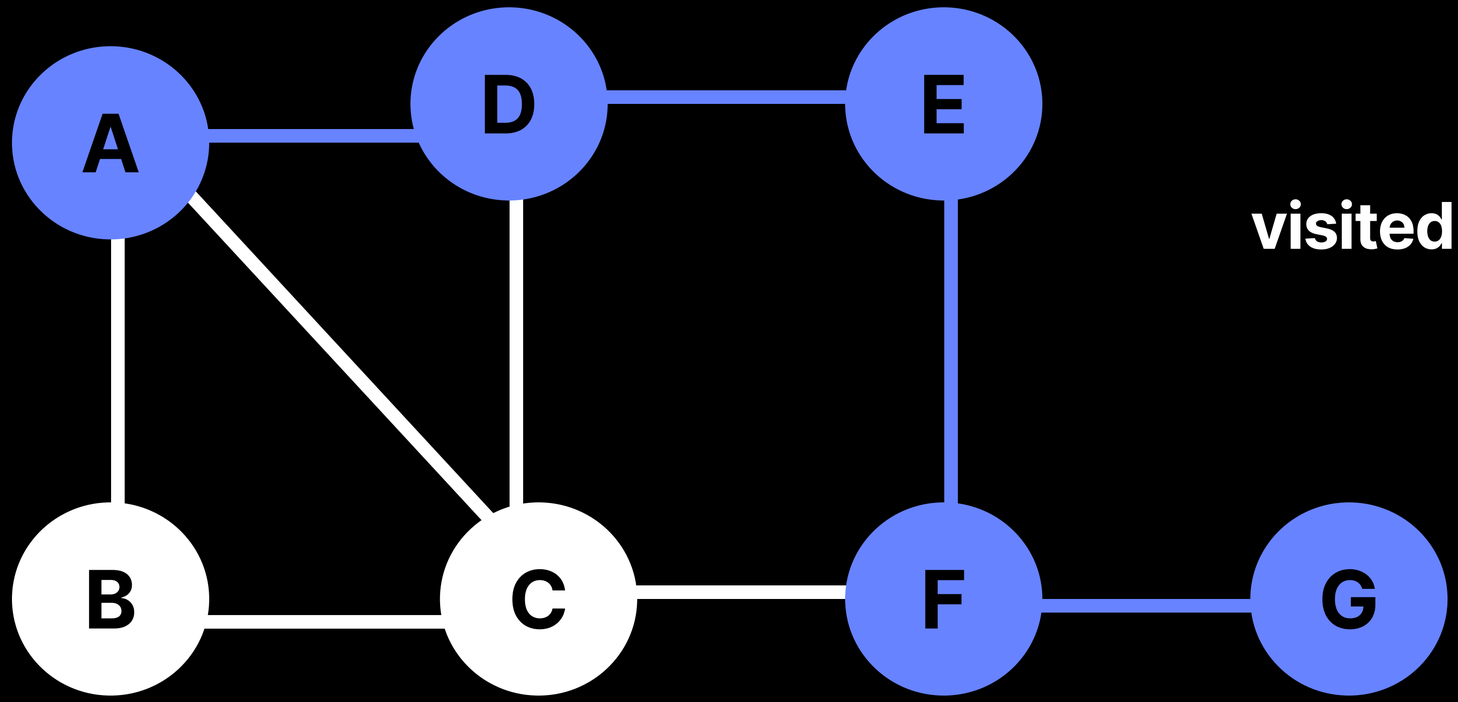
Beveger oss i dybden, dvs. vi prøver å gå så langt bort fra startnoden som mulig



visited A D E F

Dybde-først søk (DFS)

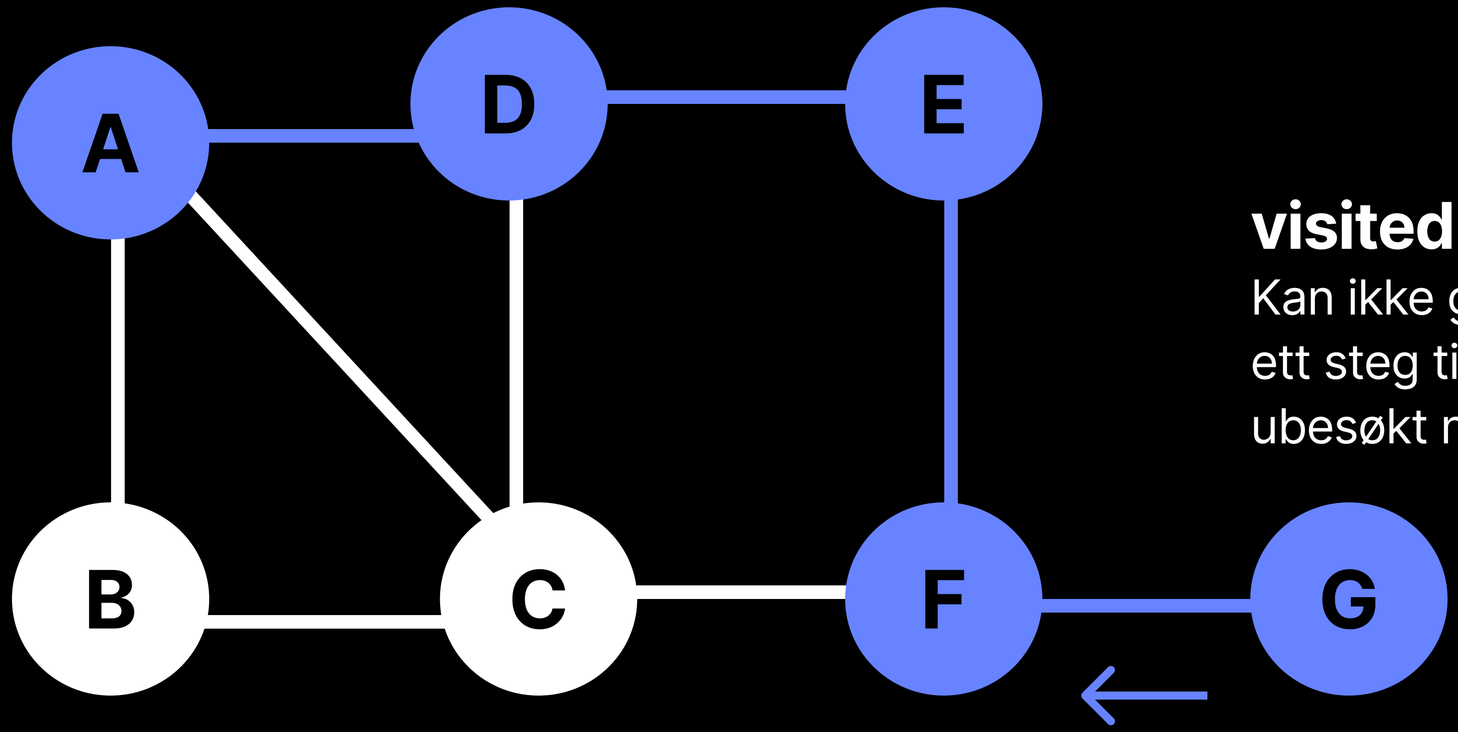
Beveger oss i dybden, dvs. vi prøver å gå så langt bort fra startnoden som mulig



visited **A D E F G**

Dybde-først søk (DFS)

Beveger oss i dybden, dvs. vi prøver å gå så langt bort fra startnoden som mulig

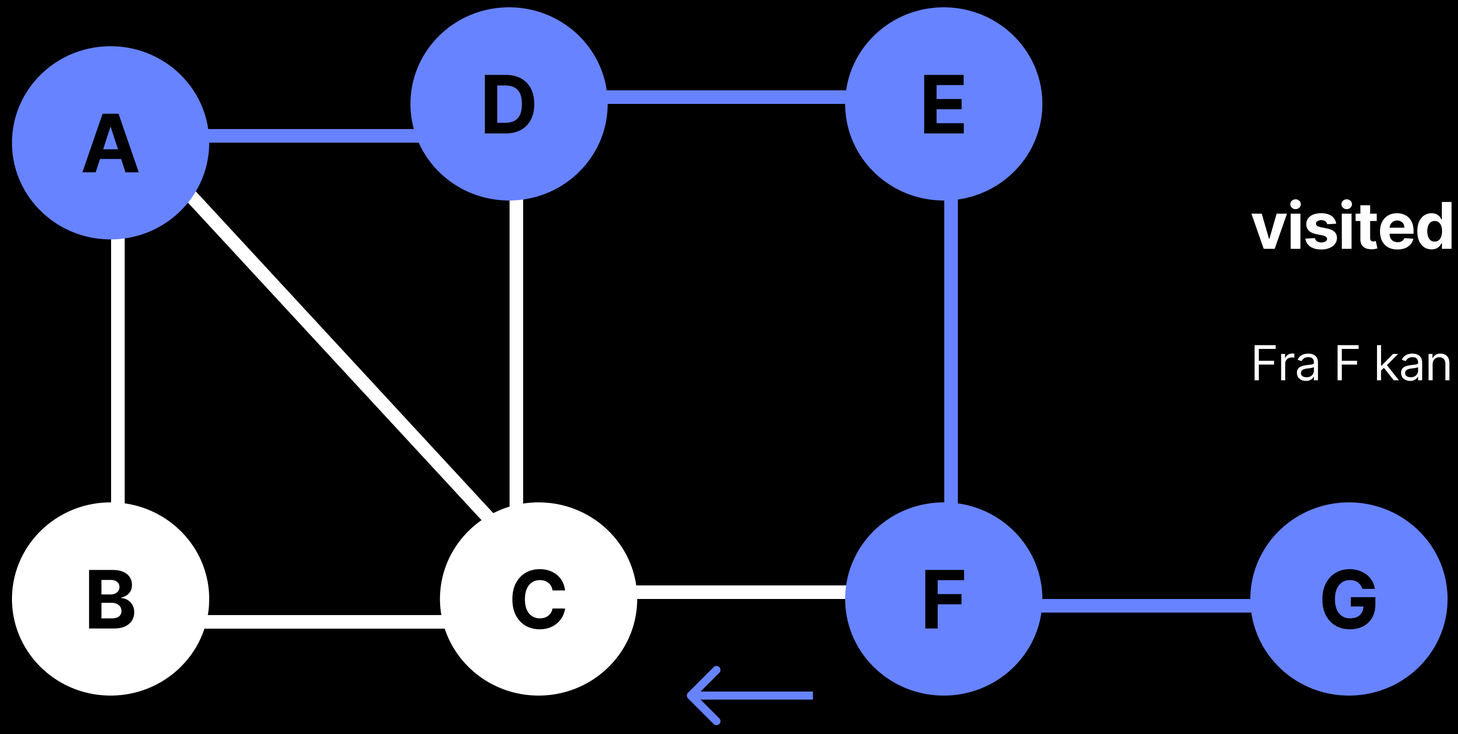


visited **A D E F G**

Kan ikke gå lenger bort fra A, så vi hopper ett steg tilbake og ser om vi kan gå til en ubesøkt node

Dybde-først søk (DFS)

Beveger oss i dybden, dvs. vi prøver å gå så langt bort fra startnoden som mulig

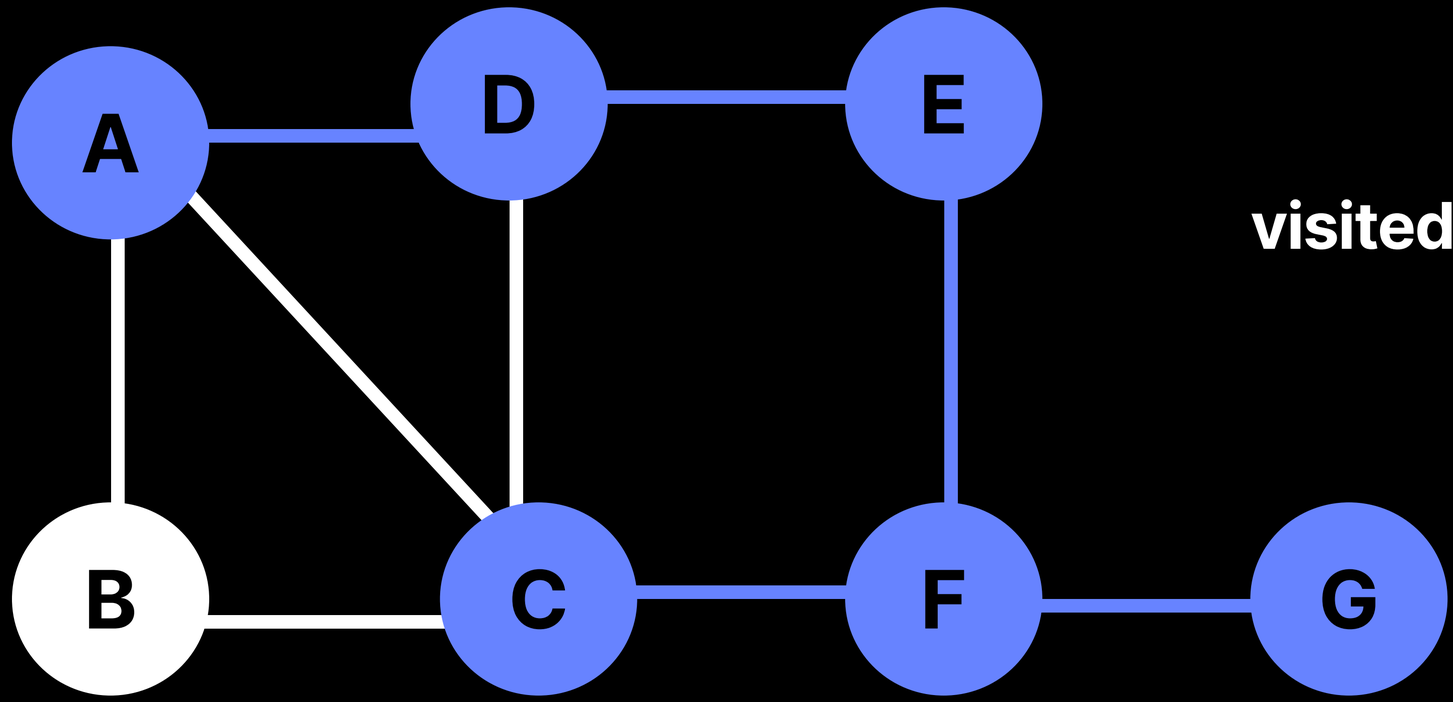


visited **A D E F G**

Fra F kan vi gå videre til C!

Dybde-først søk (DFS)

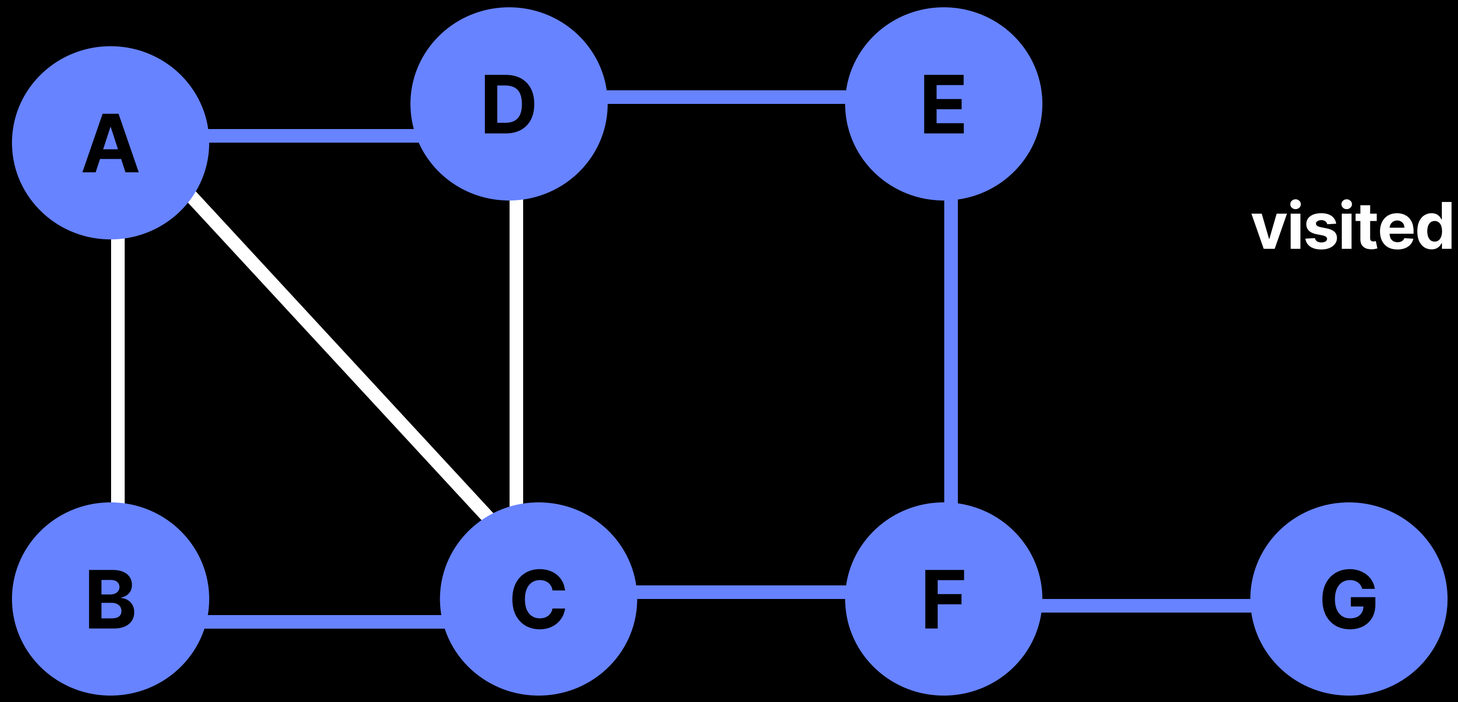
Beveger oss i dybden, dvs. vi prøver å gå så langt bort fra startnoden som mulig



visited **A D E F G C**

Dybde-først søk (DFS)

Beveger oss i dybden, dvs. vi prøver å gå så langt bort fra startnoden som mulig

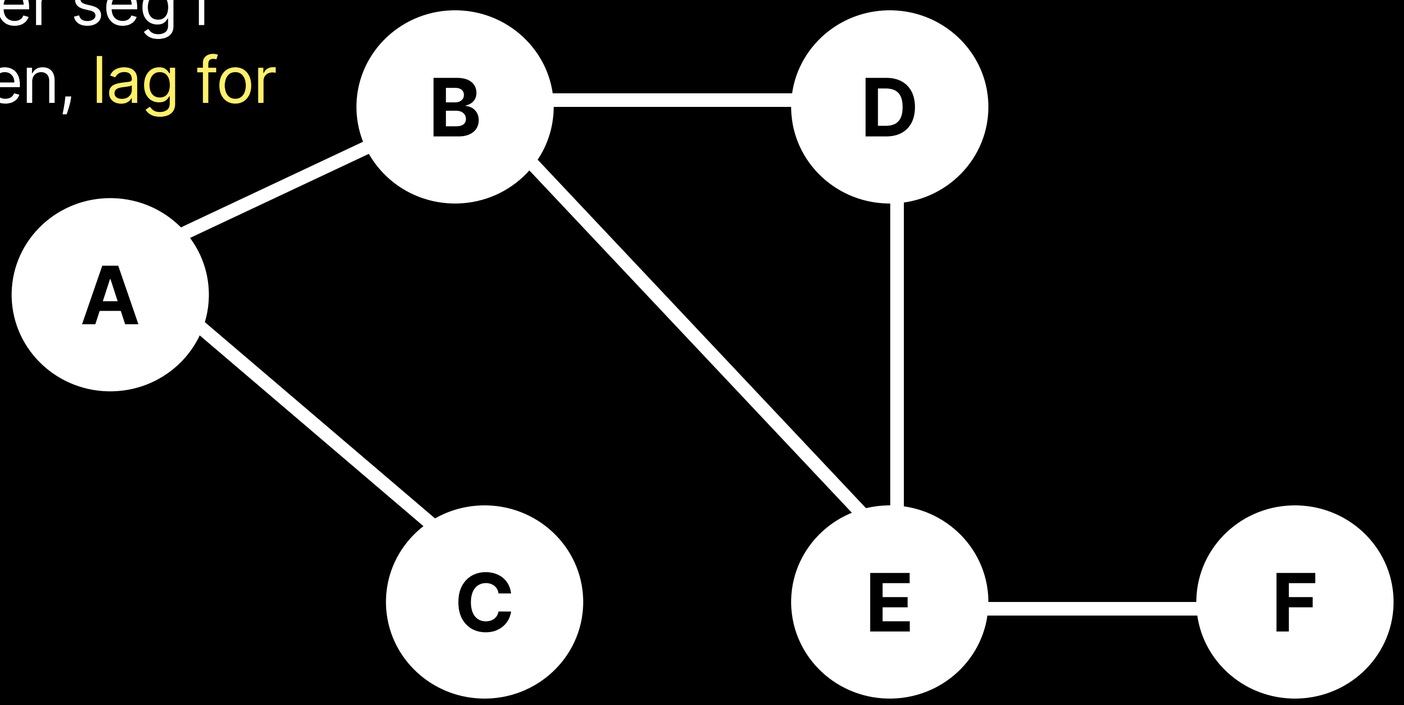


visited **A D E F G C B**

Korteste sti i en uvekta graf

Bredde først søk

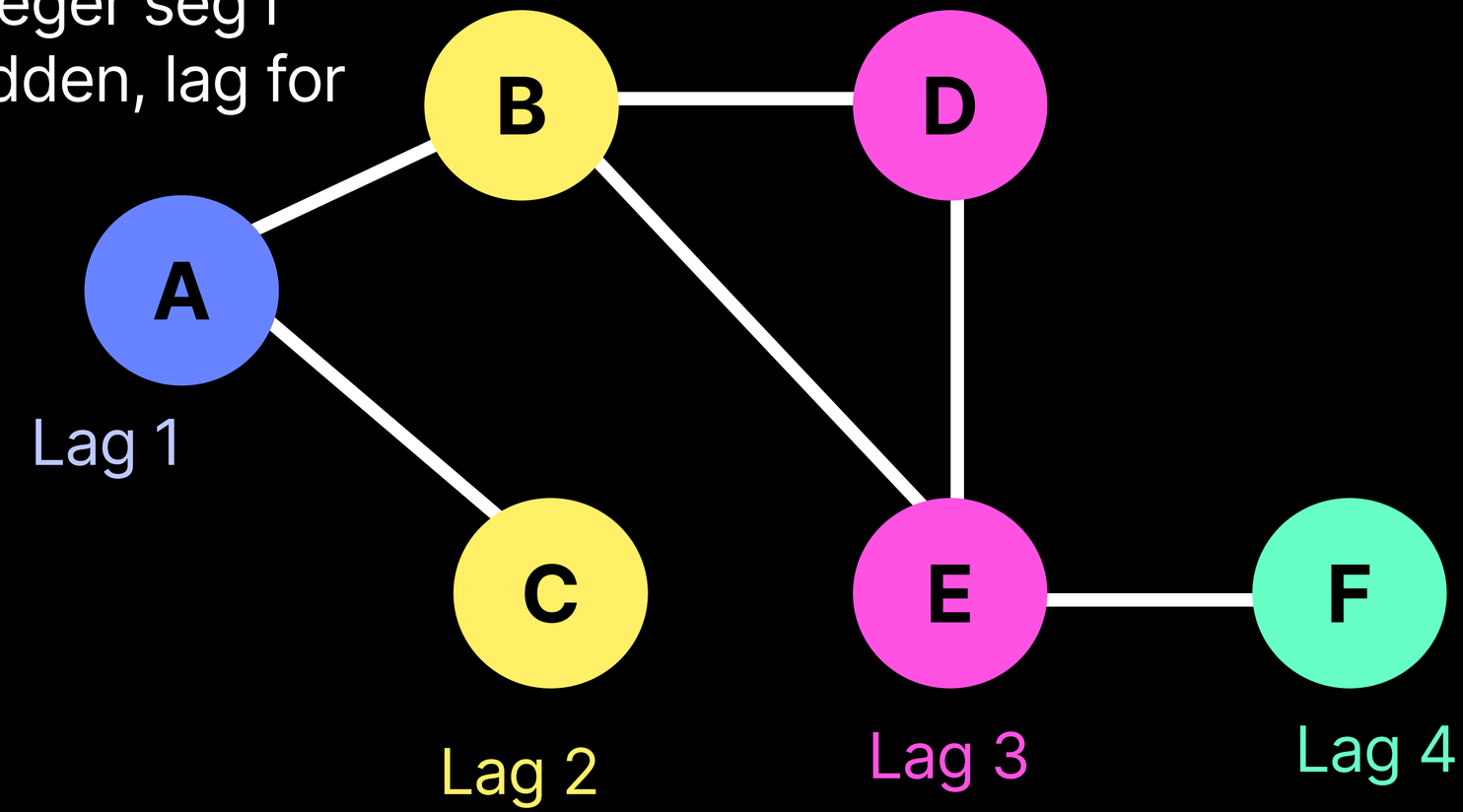
Beveger seg i bredden, lag for lag



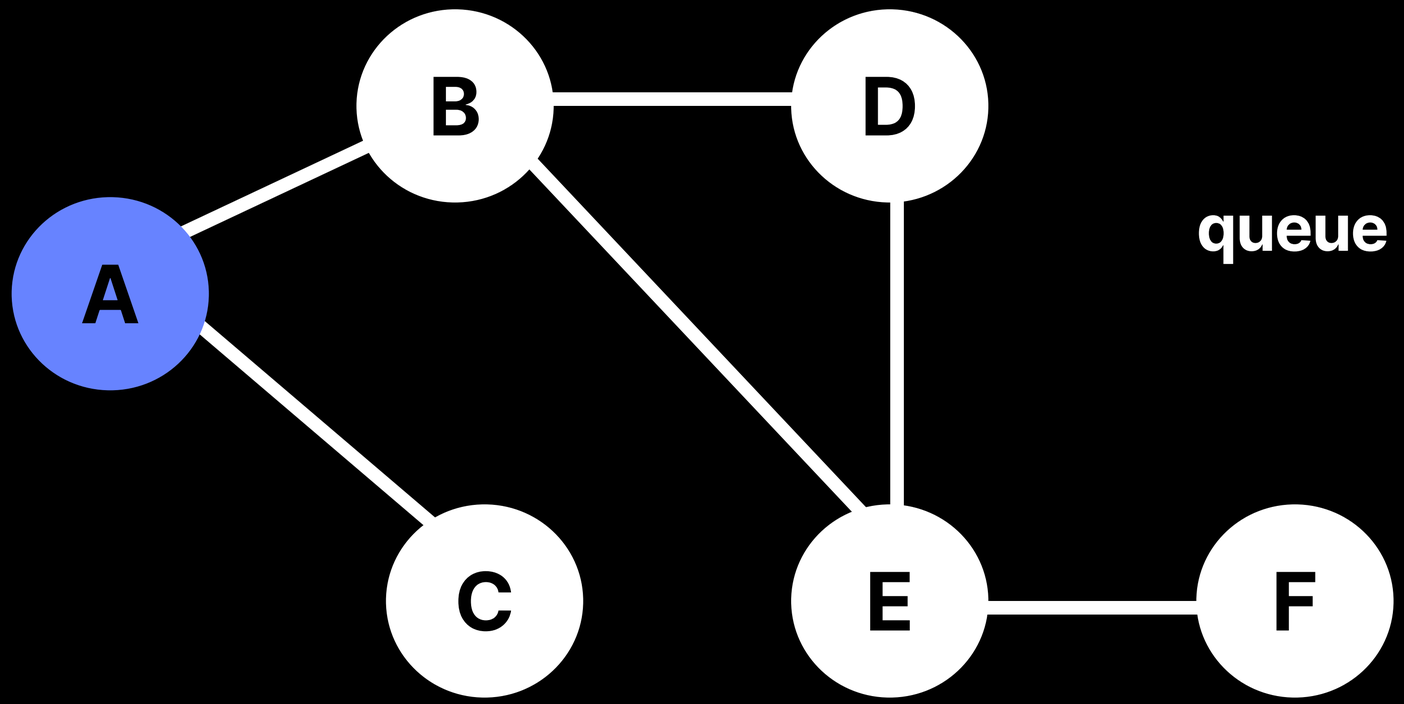
Korteste sti i en uvekta graf

Bredde først søk

Beveger seg i bredden, lag for lag

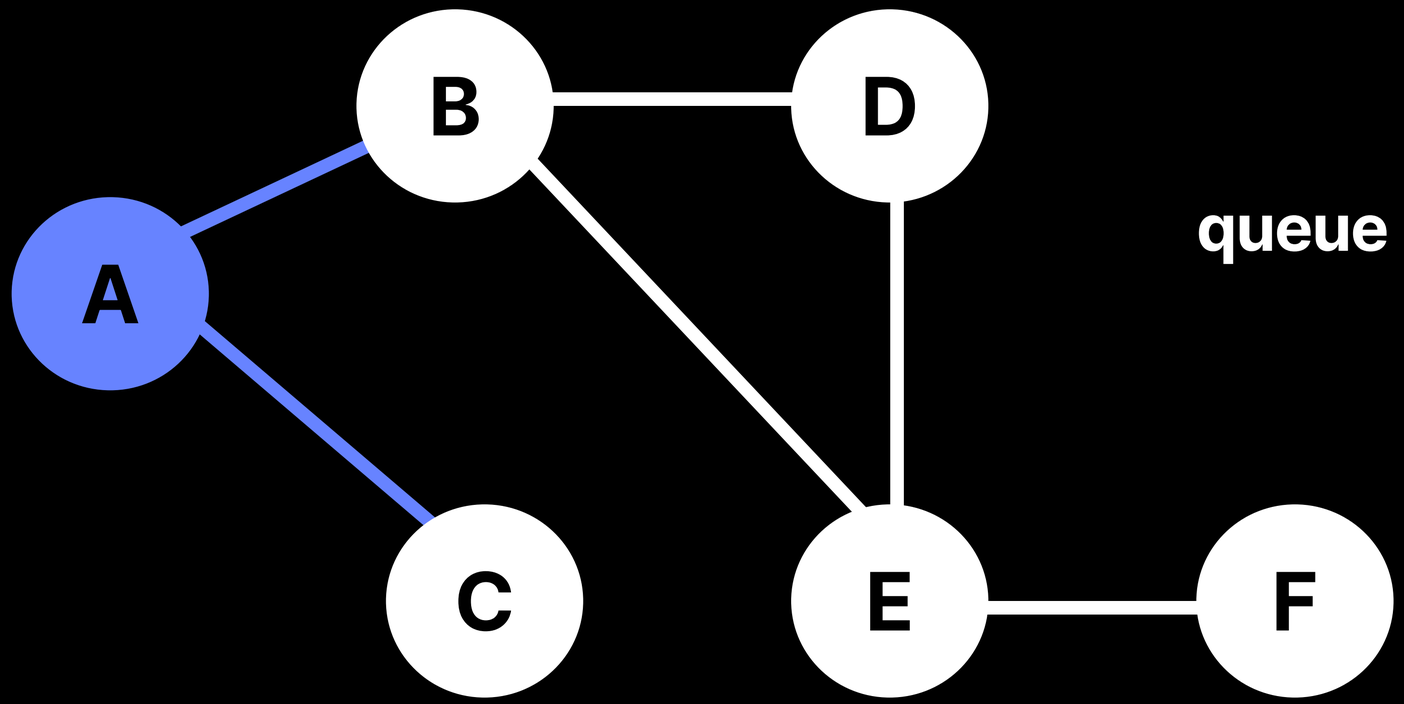


Bredde først søk (BFS)



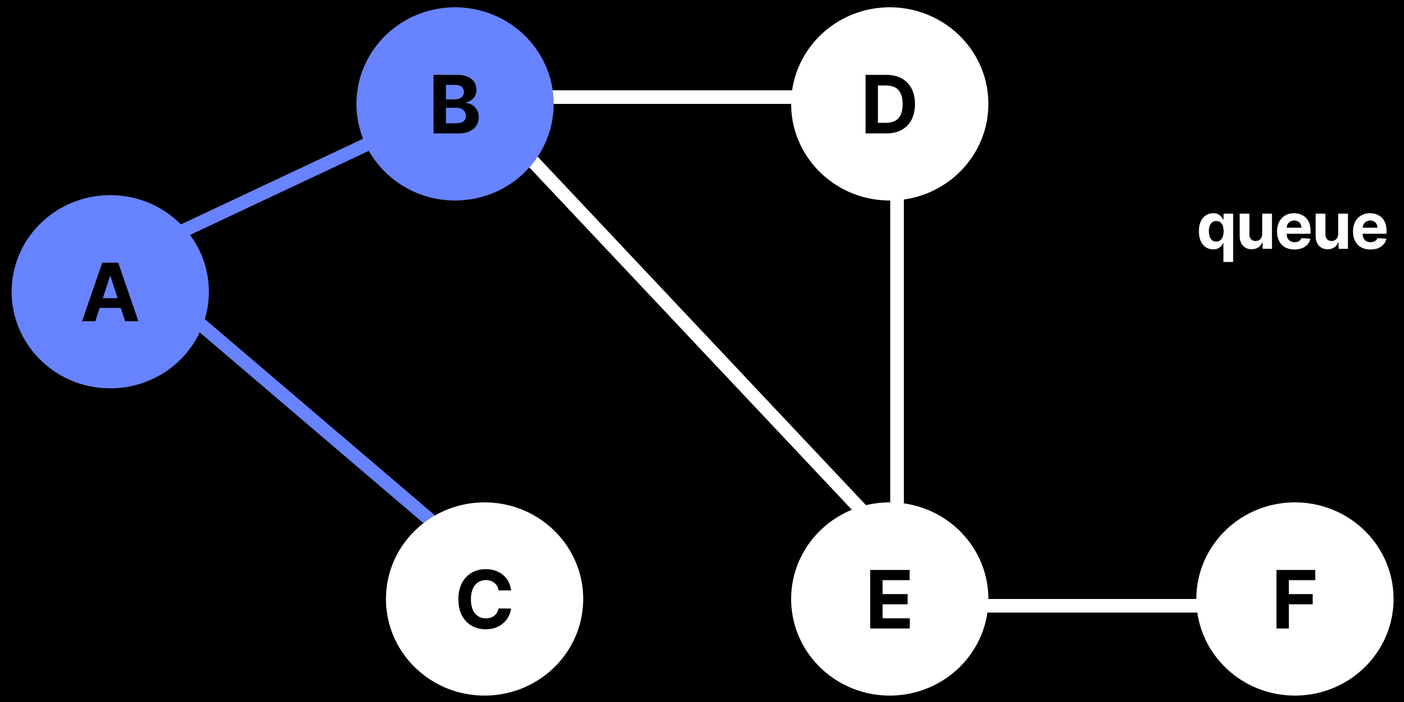
queue A

Bredde først søk (BFS)



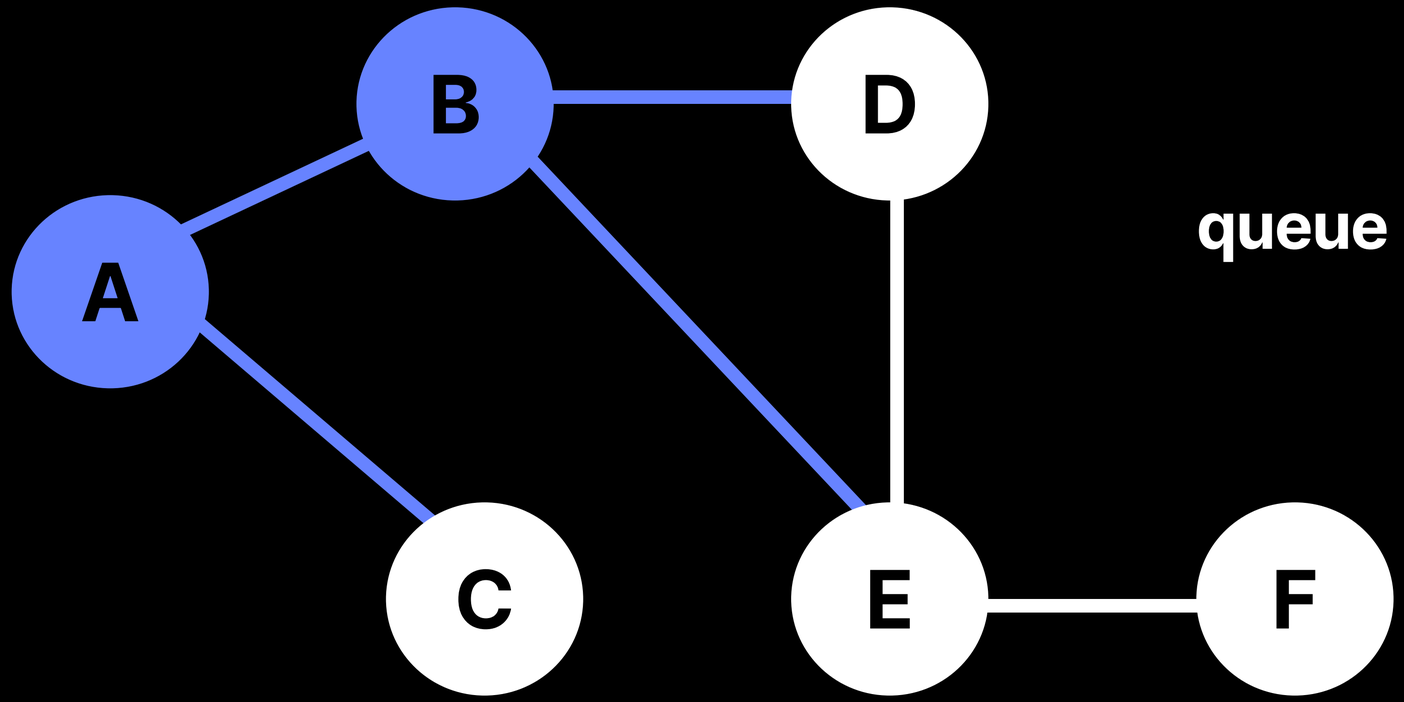
queue **A** B C

Bredde først søk (BFS)



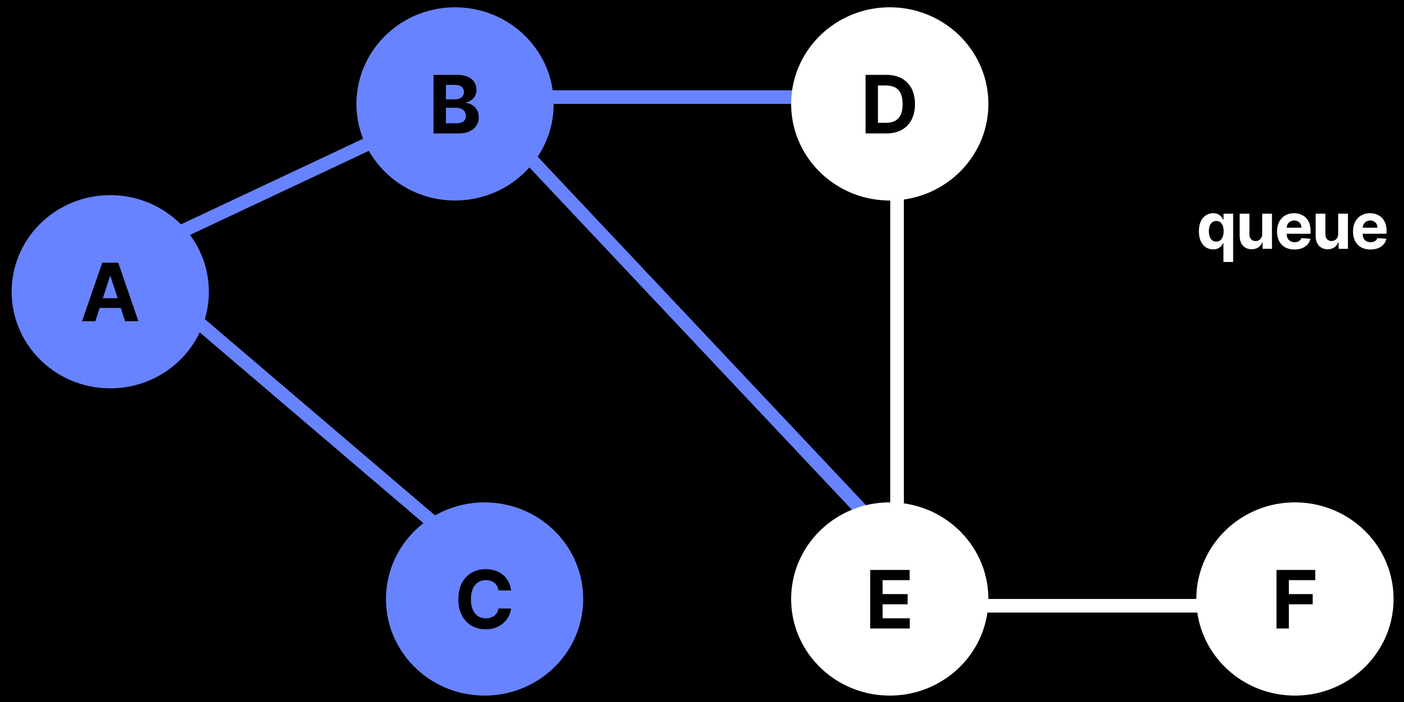
queue A **B** C

Bredde først søk (BFS)



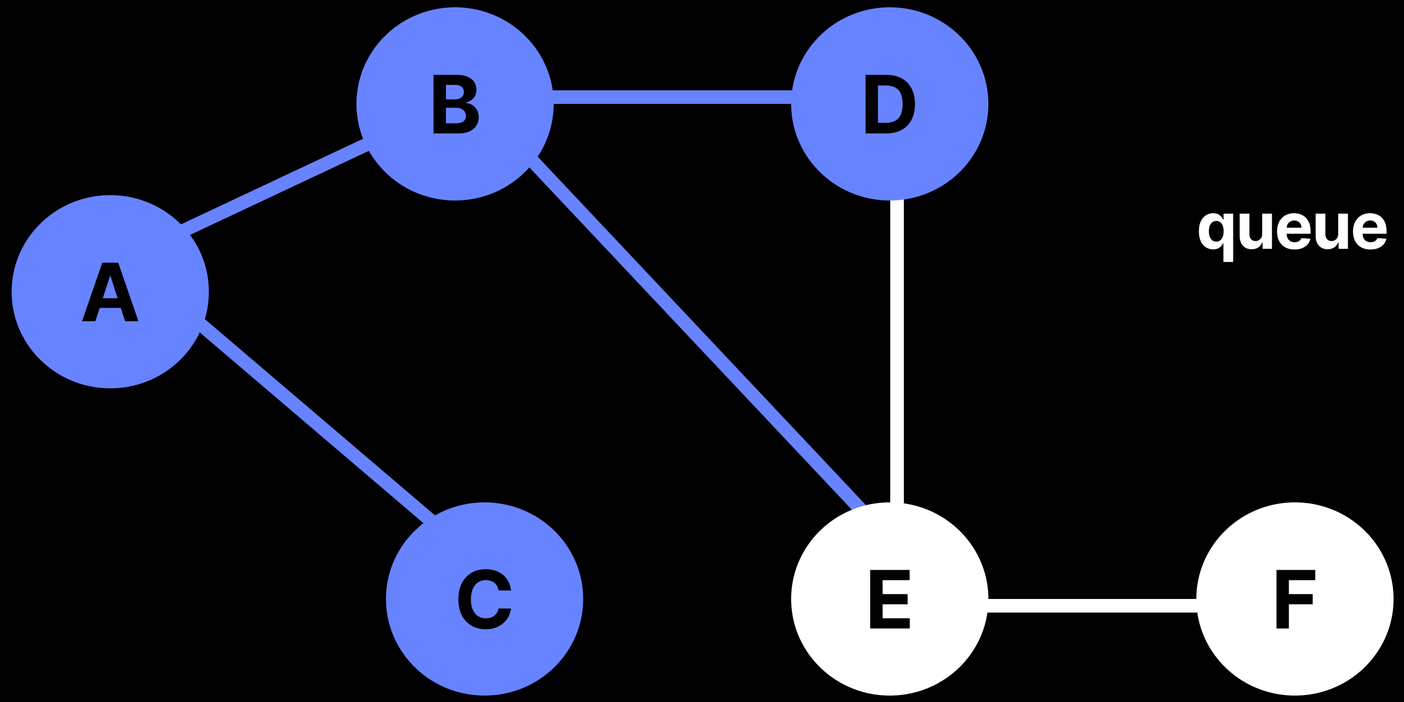
queue A **B** C D E

Bredde først søk (BFS)



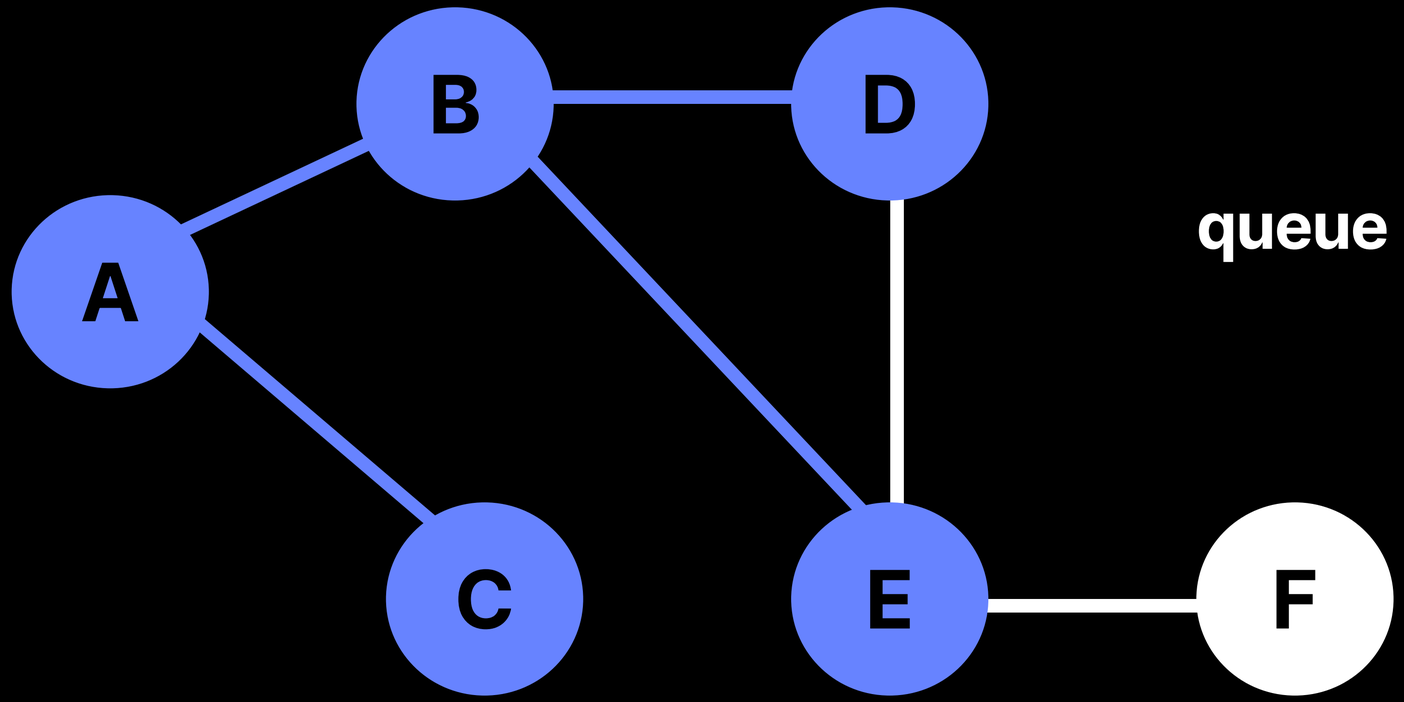
queue A B **C** D E

Bredde først søk (BFS)



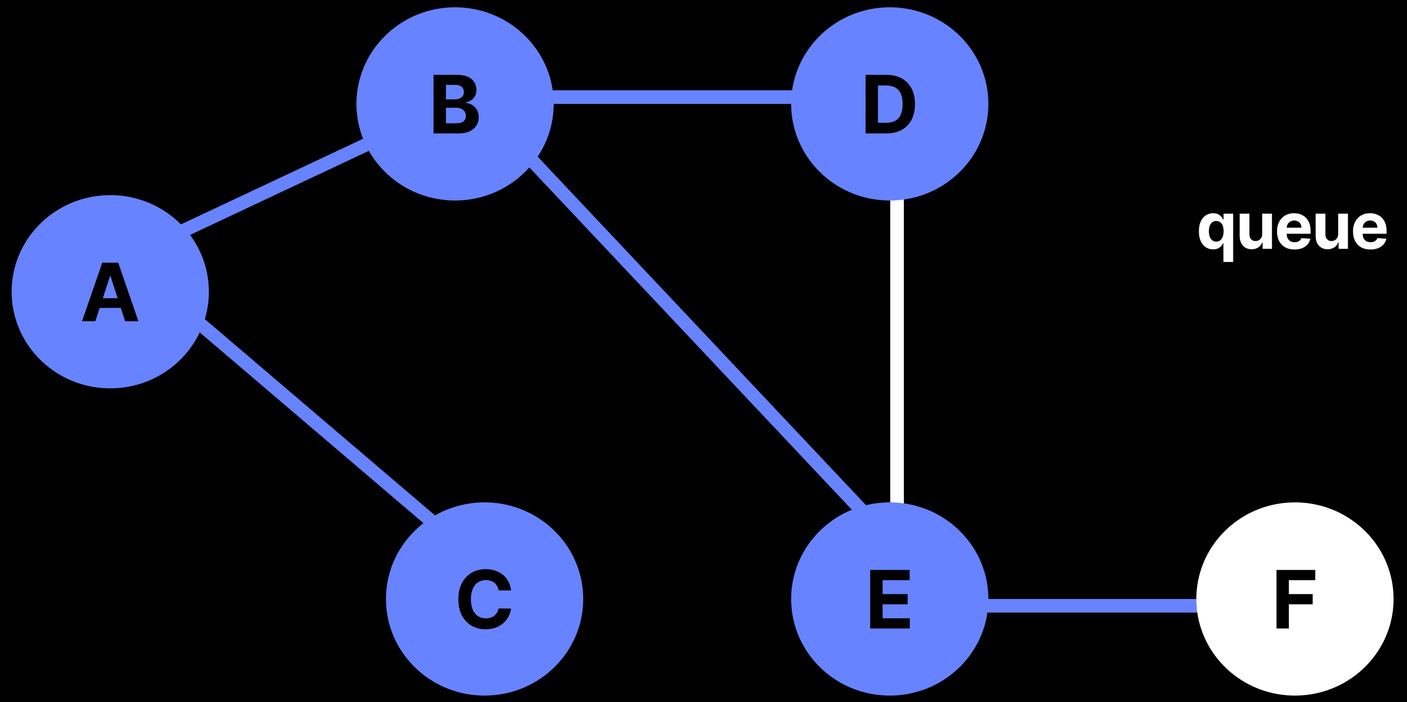
queue A B C **D** E

Bredde først søk (BFS)



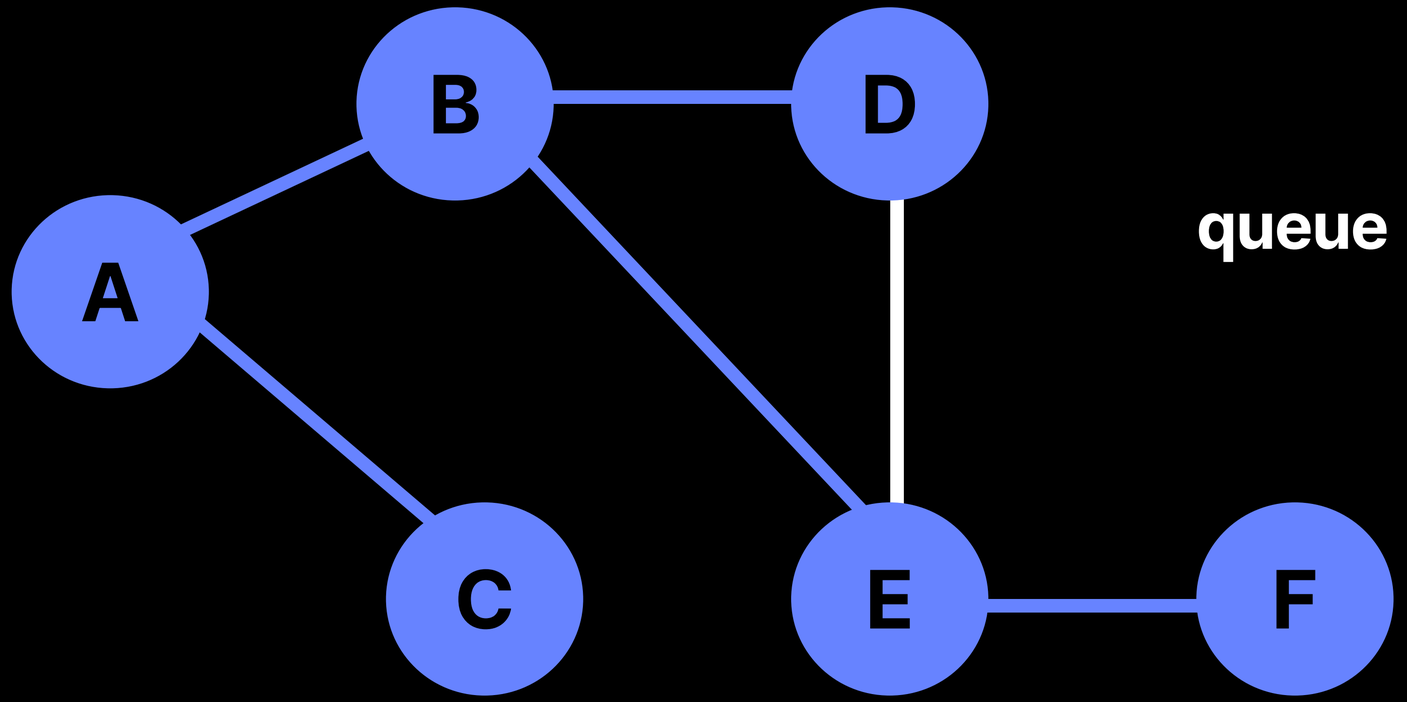
queue A B C D **E**

Bredde først søk (BFS)



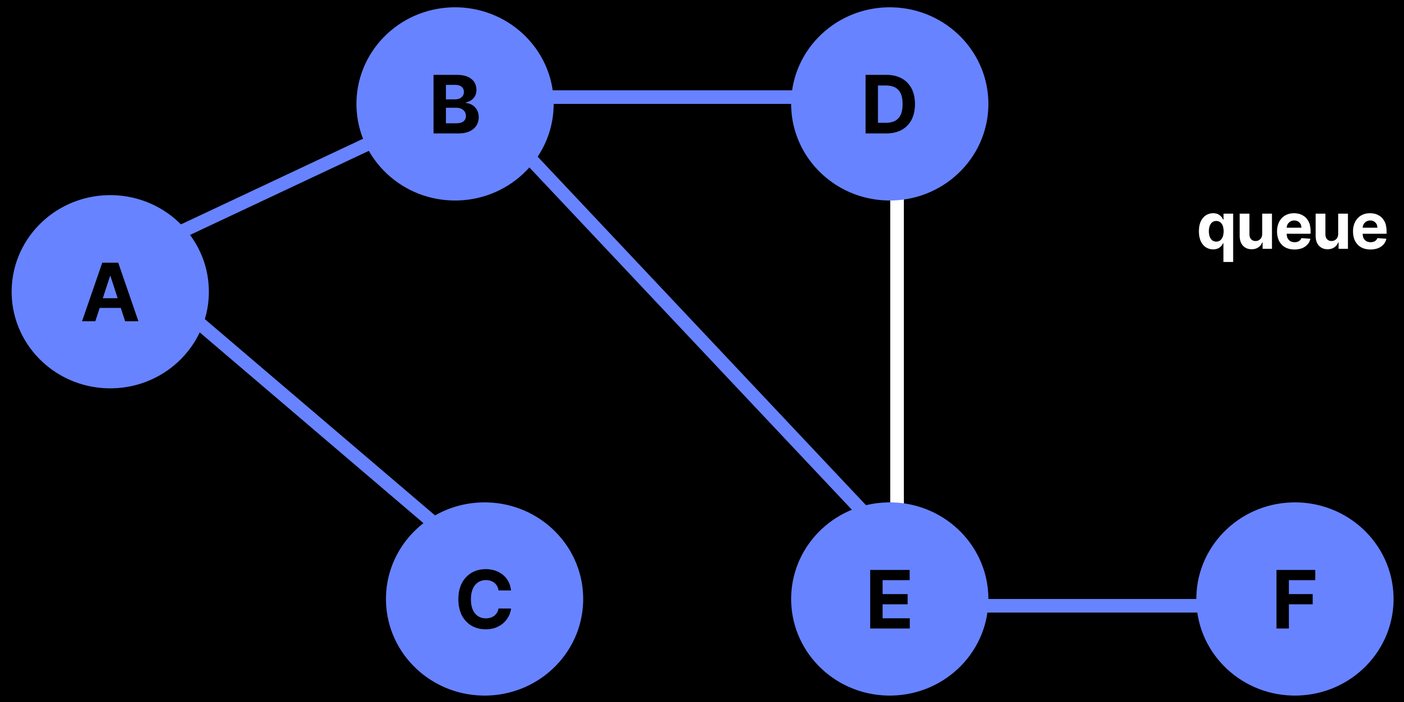
queue A B C D **E** F

Bredde først søk (BFS)



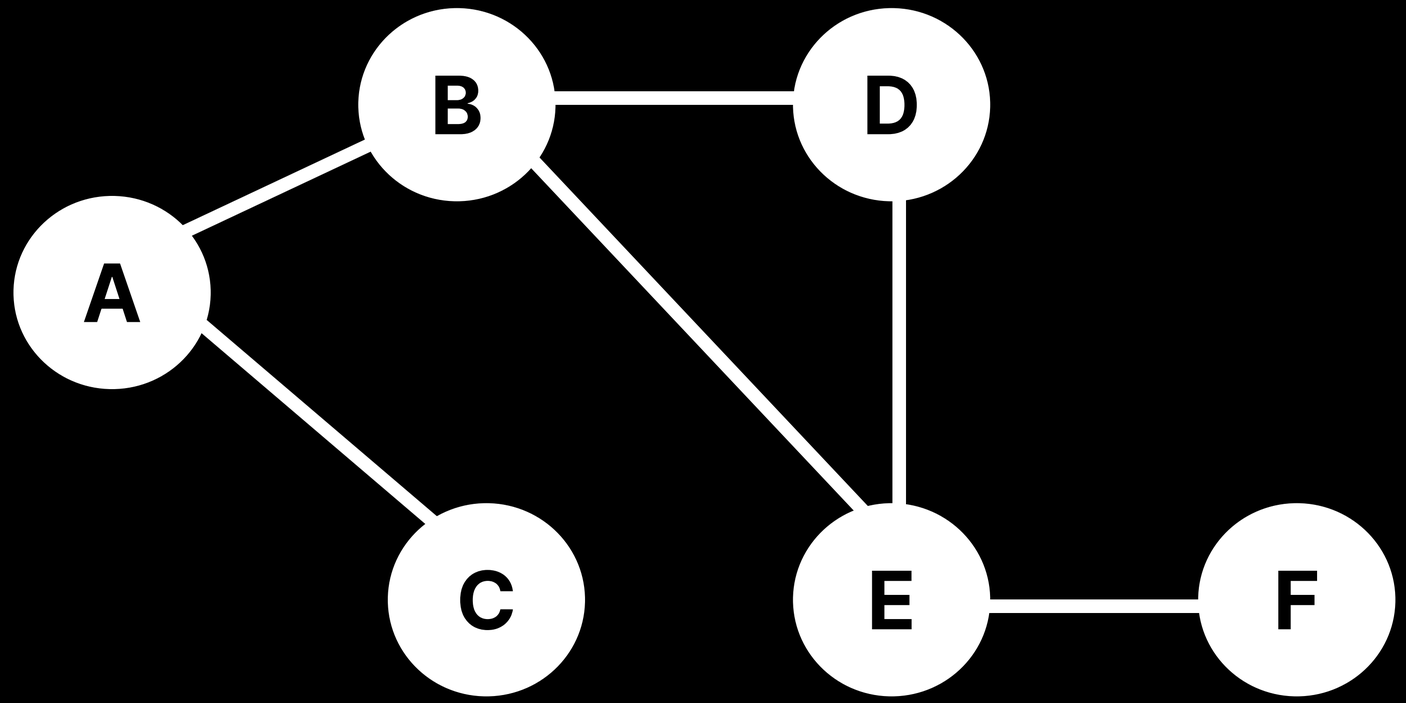
queue A B C D E **F**

Bredde først søk (BFS)

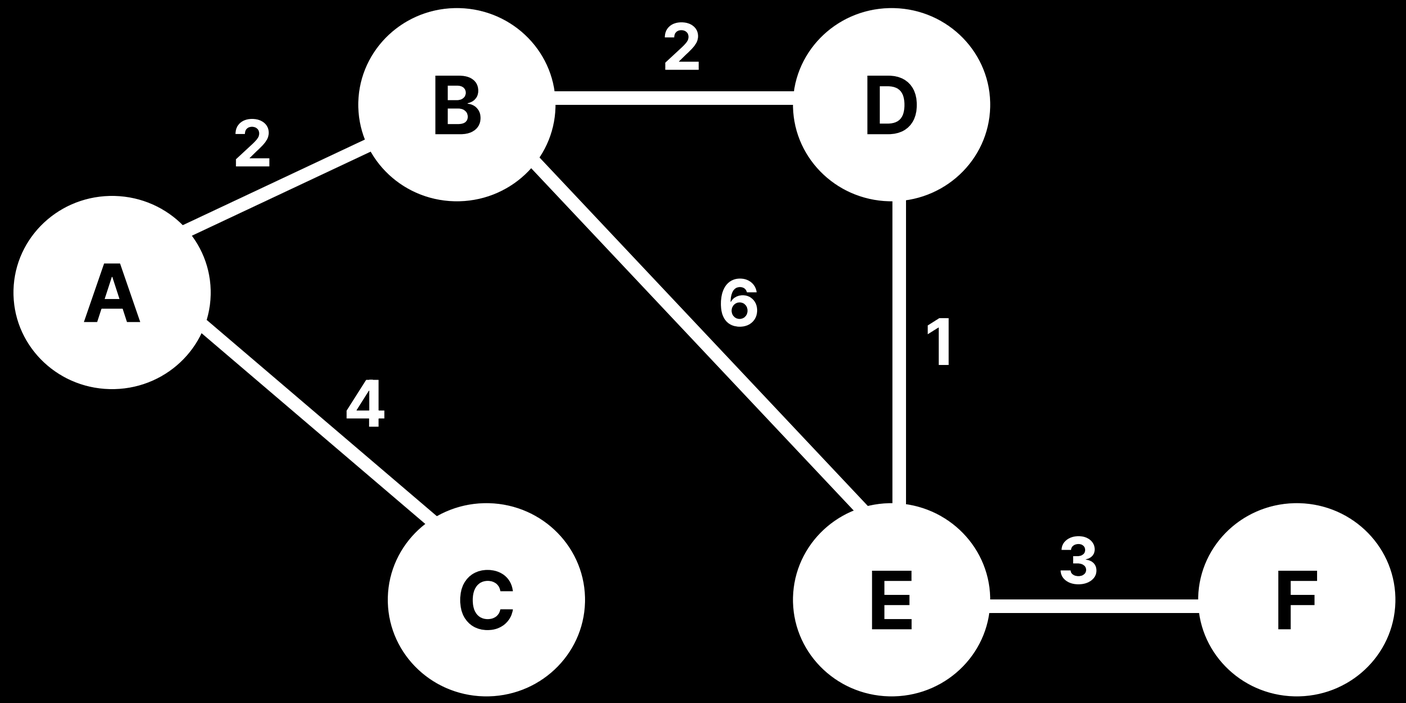


queue A B C D E F

Hvordan gjør vi denne grafen om til en vekted graf?



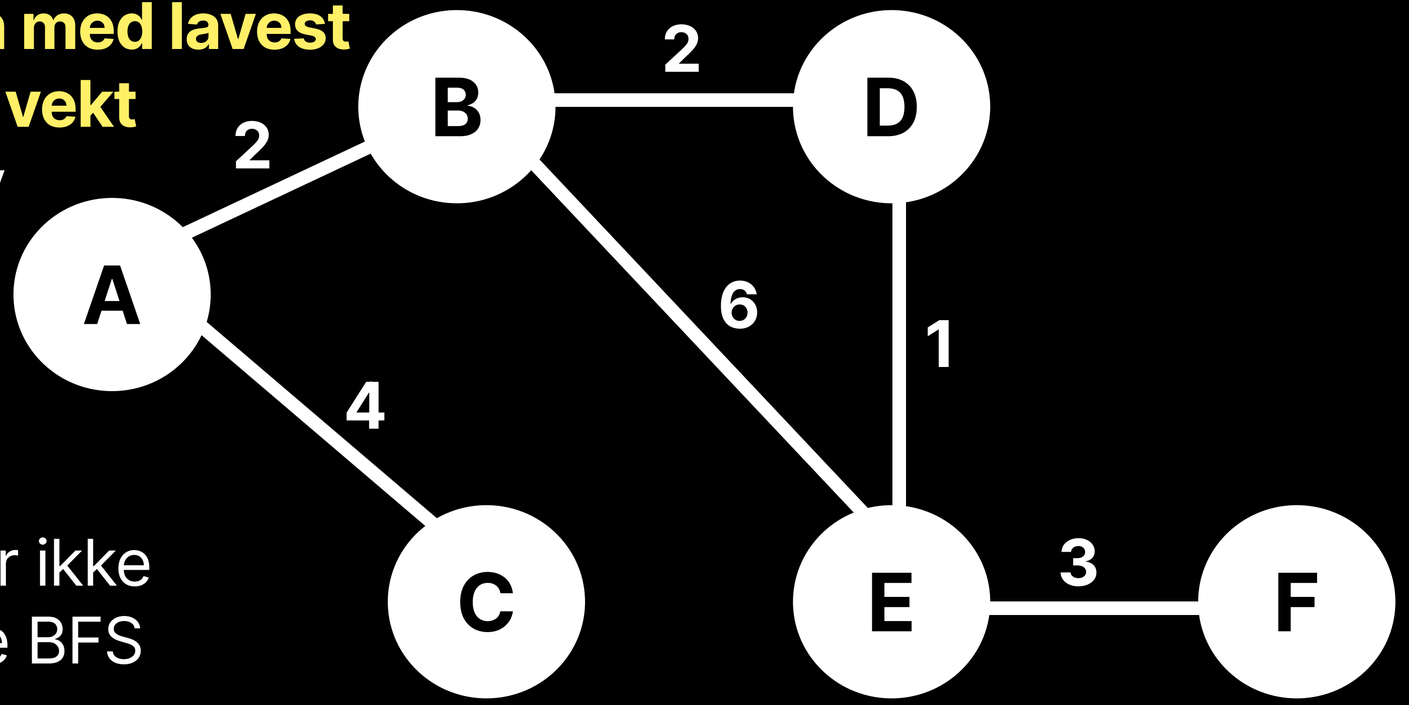
Vektede grafer



Korteste sti i vektede grafer

Korteste sti i en vekted graf er stien med lavest akkumulert vekt

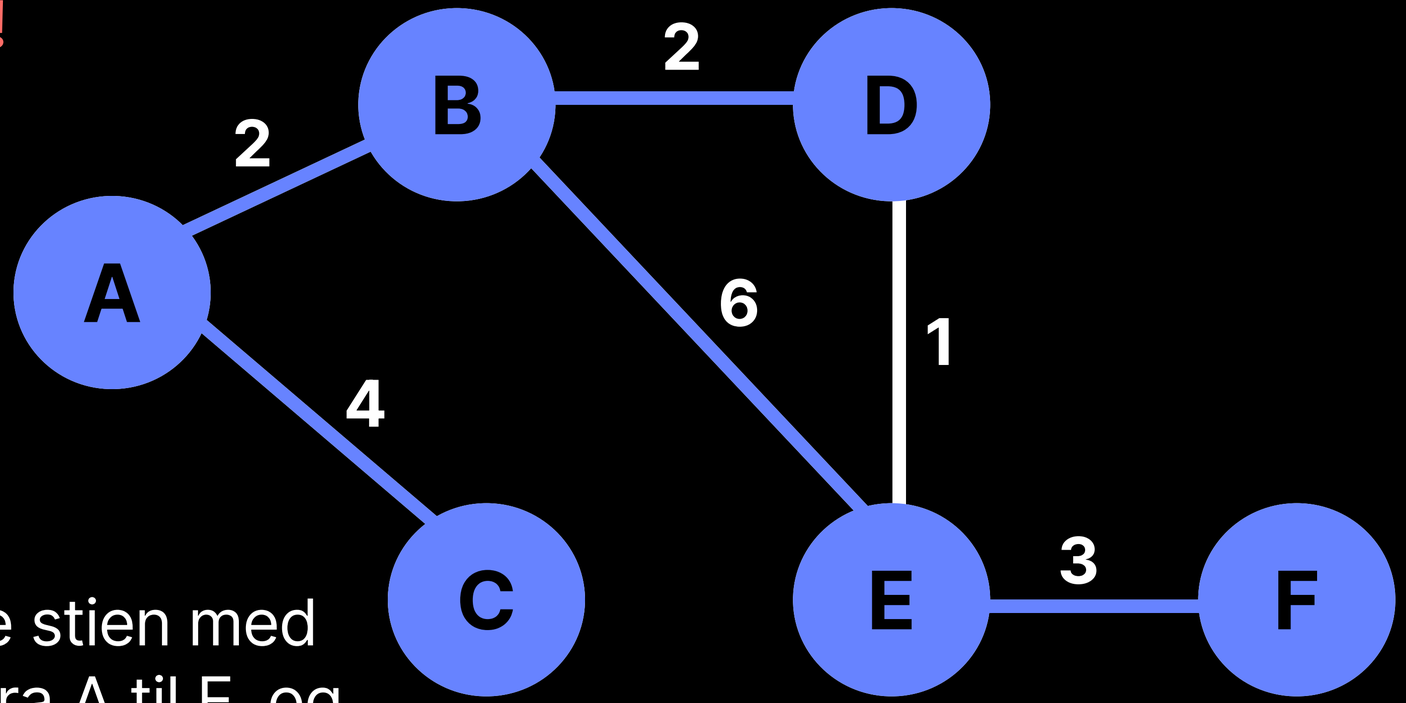
dvs. summen av verdiene i stien skal være minst mulig



Vi kan derfor ikke lenger bruke BFS

Korteste sti i vektede grafer

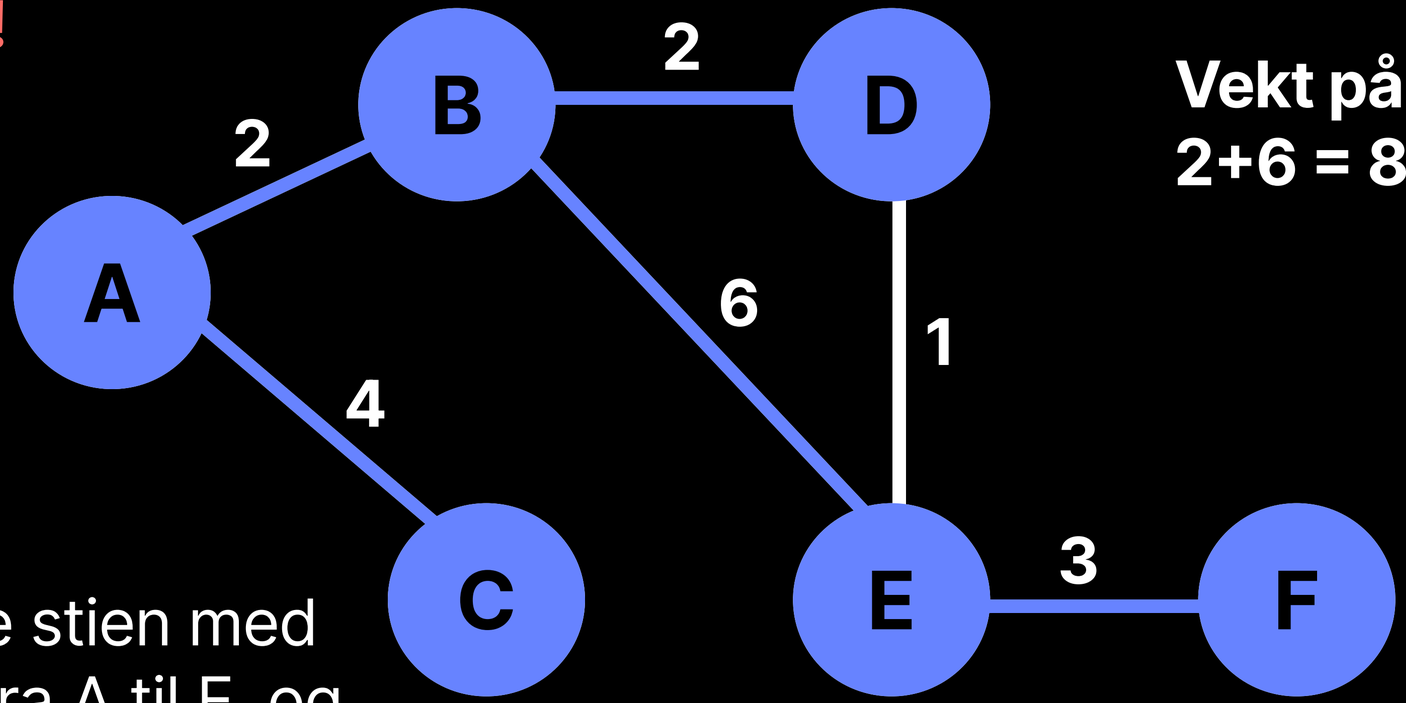
BFS ville ikke gitt oss korteste sti!



Dette er ikke stien med lavest vekt fra A til E, og dermed ikke korteste sti

Korteste sti i vektede grafer

BFS ville ikke gitt oss korteste sti!

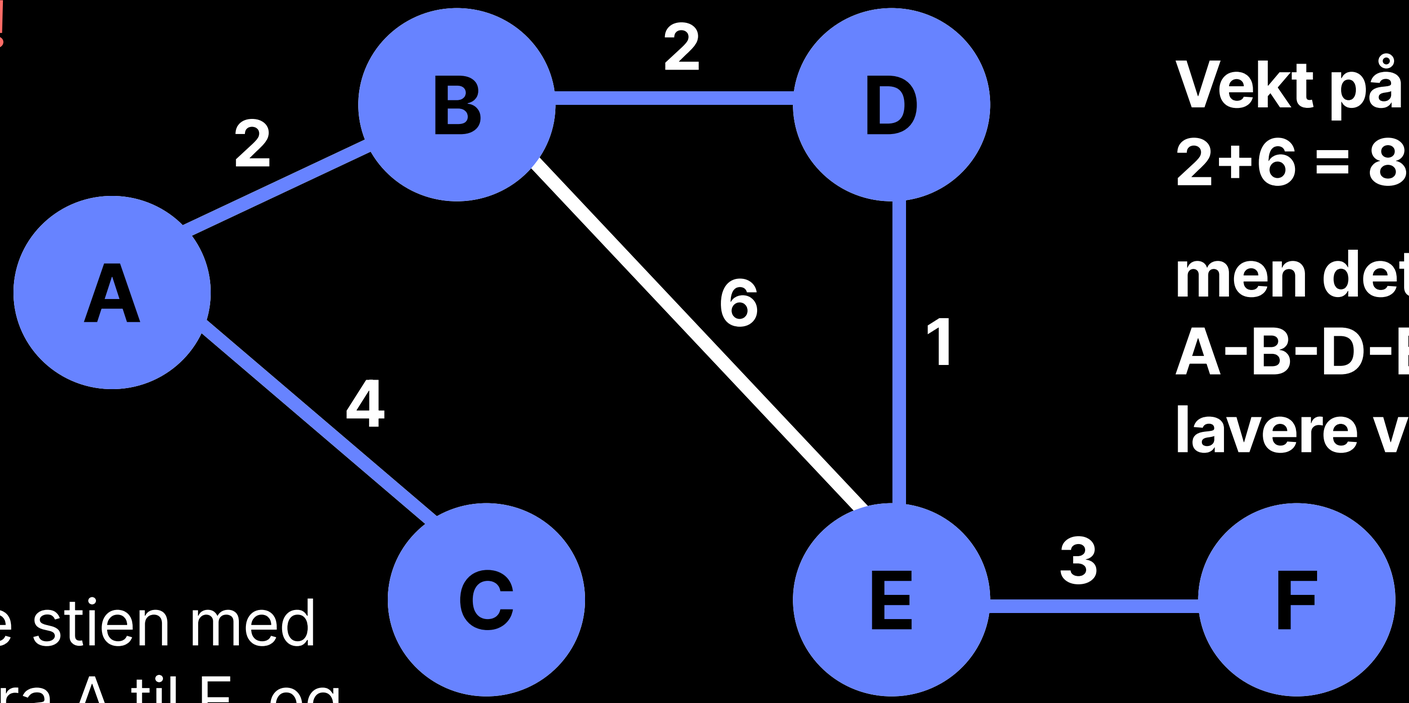


Vekt på stien A-B-E er 2+6 = 8

Dette er ikke stien med lavest vekt fra A til E, og dermed ikke korteste sti

Korteste sti i vektede grafer

BFS ville ikke gitt oss korteste sti!



Vekt på stien A-B-E er 2+6 = 8

men det finnes en sti A-B-D-E som har lavere vekt: 2+2+1 = 5

Dette er ikke stien med lavest vekt fra A til E, og dermed ikke korteste sti

Korteste sti i vektede grafer

Dijkstra

Når vi vet eller kan anta at det **ikke** finnes kanter med negativ vekt

Bellman-Ford

Når det finnes kanter med negativ vekt

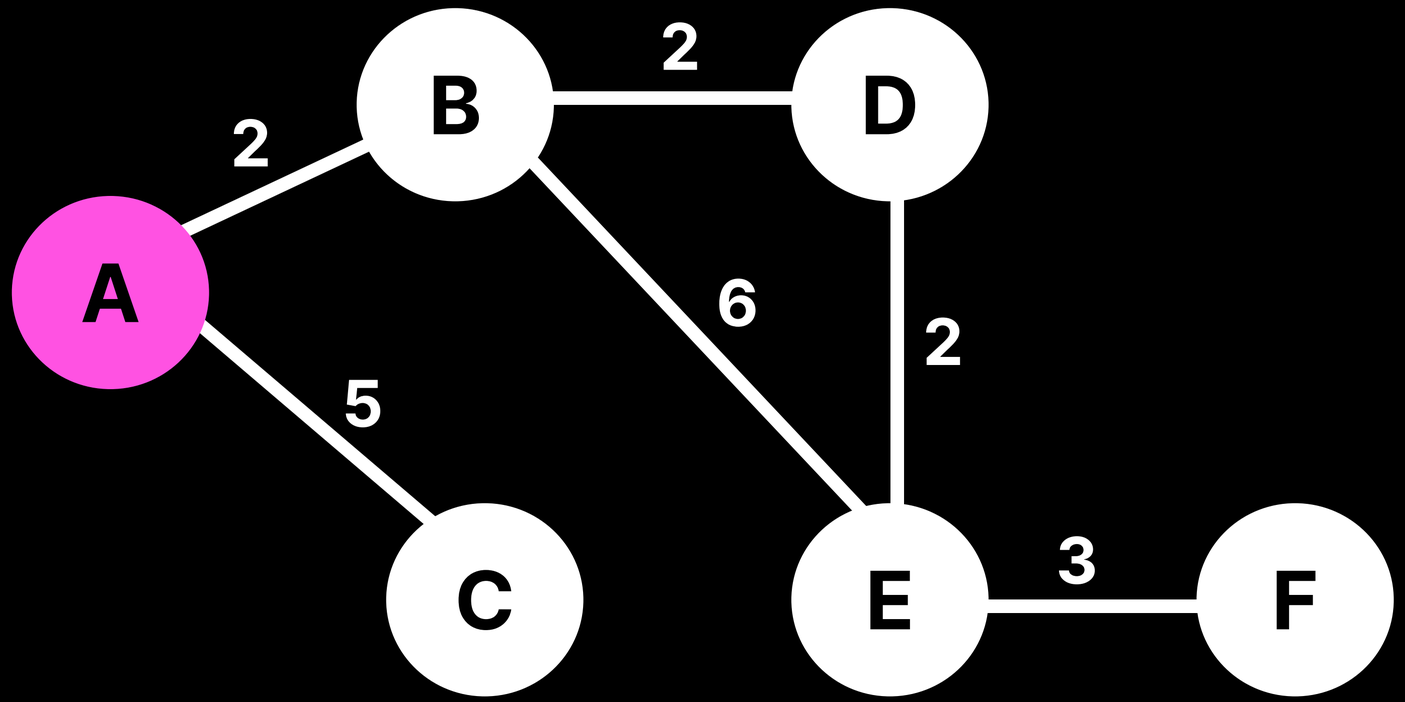
Begge algoritmene fungerer både for rettede og urettede grafer

Dijkstra

Når vi vet eller kan anta at det **ikke finnes kanter med negativ vekt**

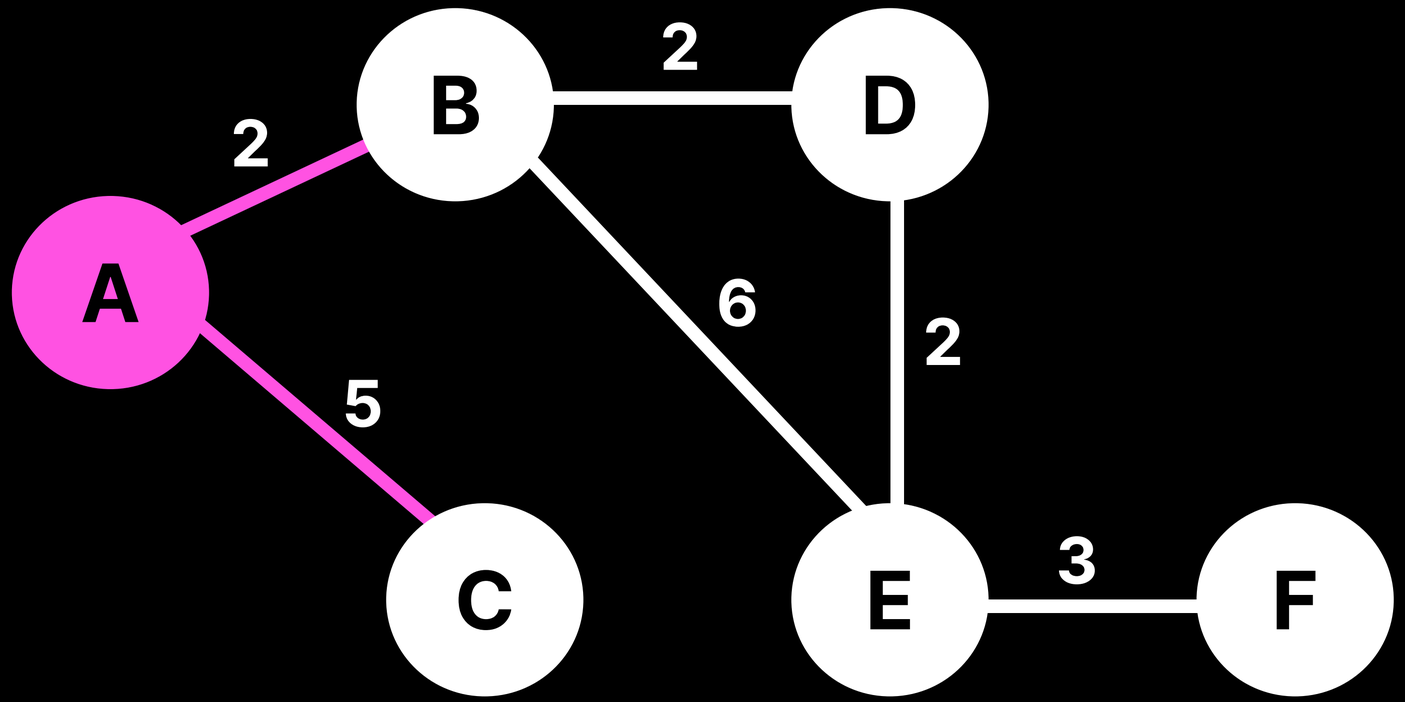
Fungerer både for rettede og urettede grafer

Dijkstra



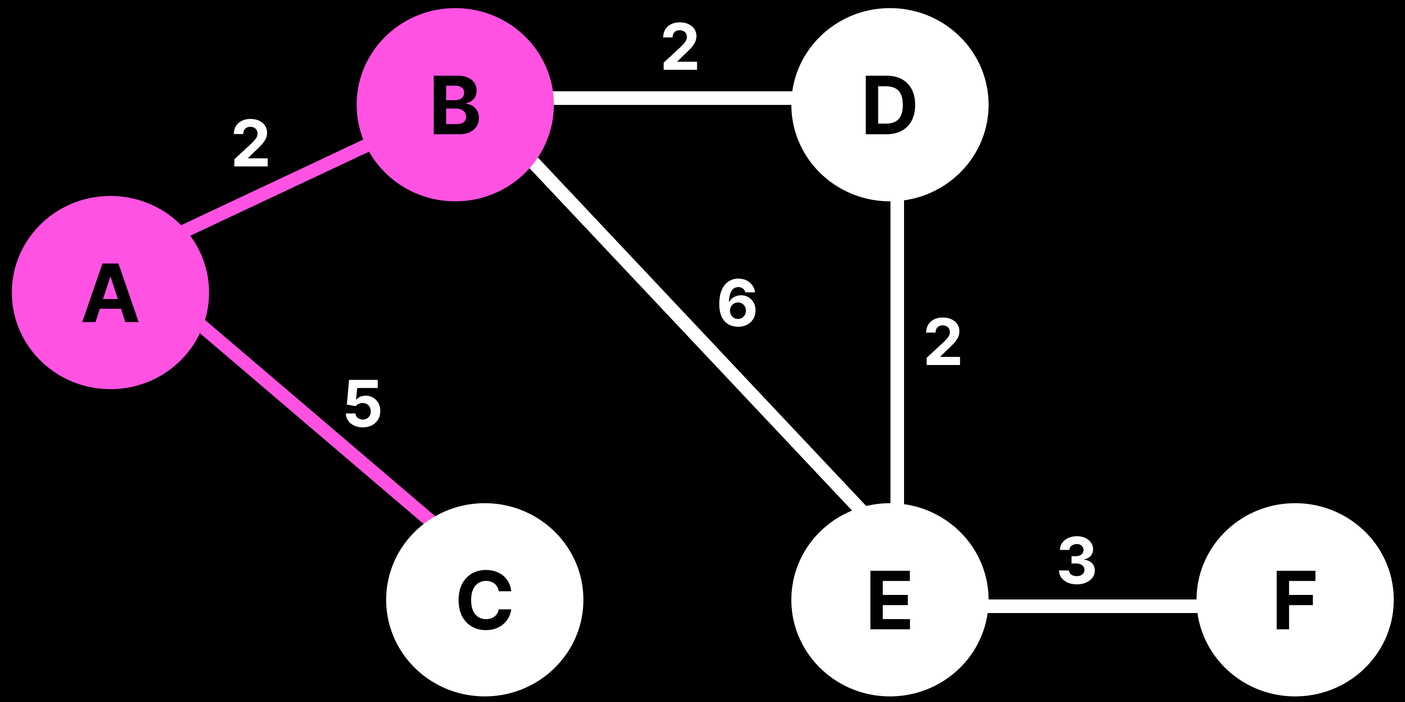
queue	distance
A	0
B	∞
C	∞
D	∞
E	∞
F	∞

Dijkstra



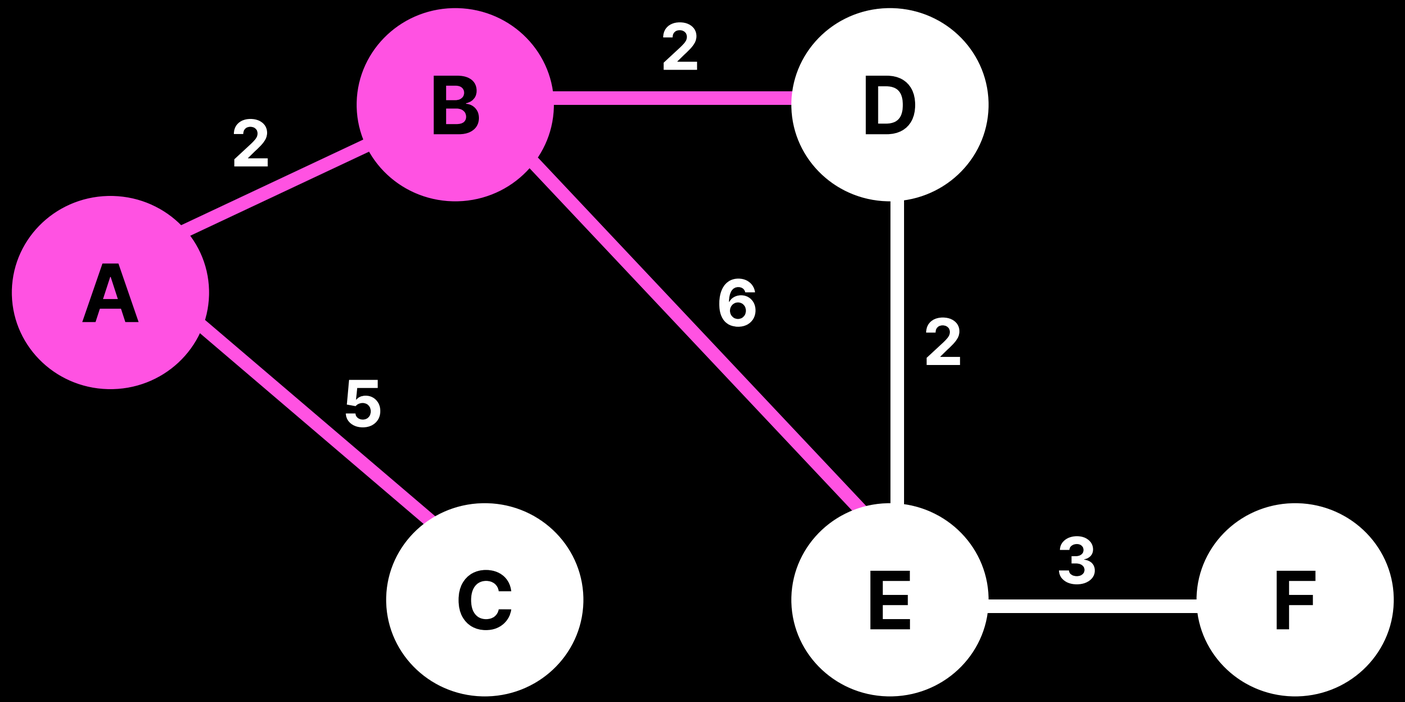
queue	distance
A	0
B	2
C	5
D	∞
E	∞
F	∞

Dijkstra



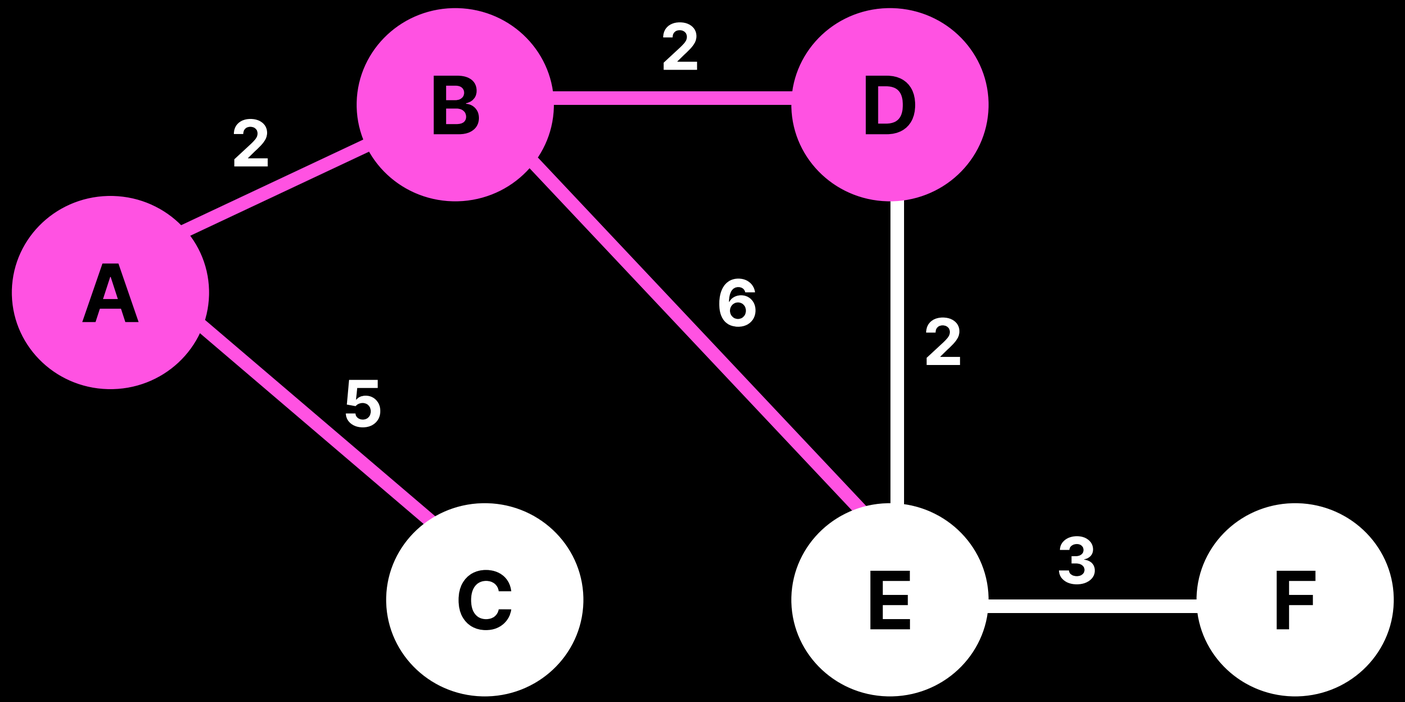
queue	distance
A	0
B	2 den minste
C	5
D	∞
E	∞
F	∞

Dijkstra



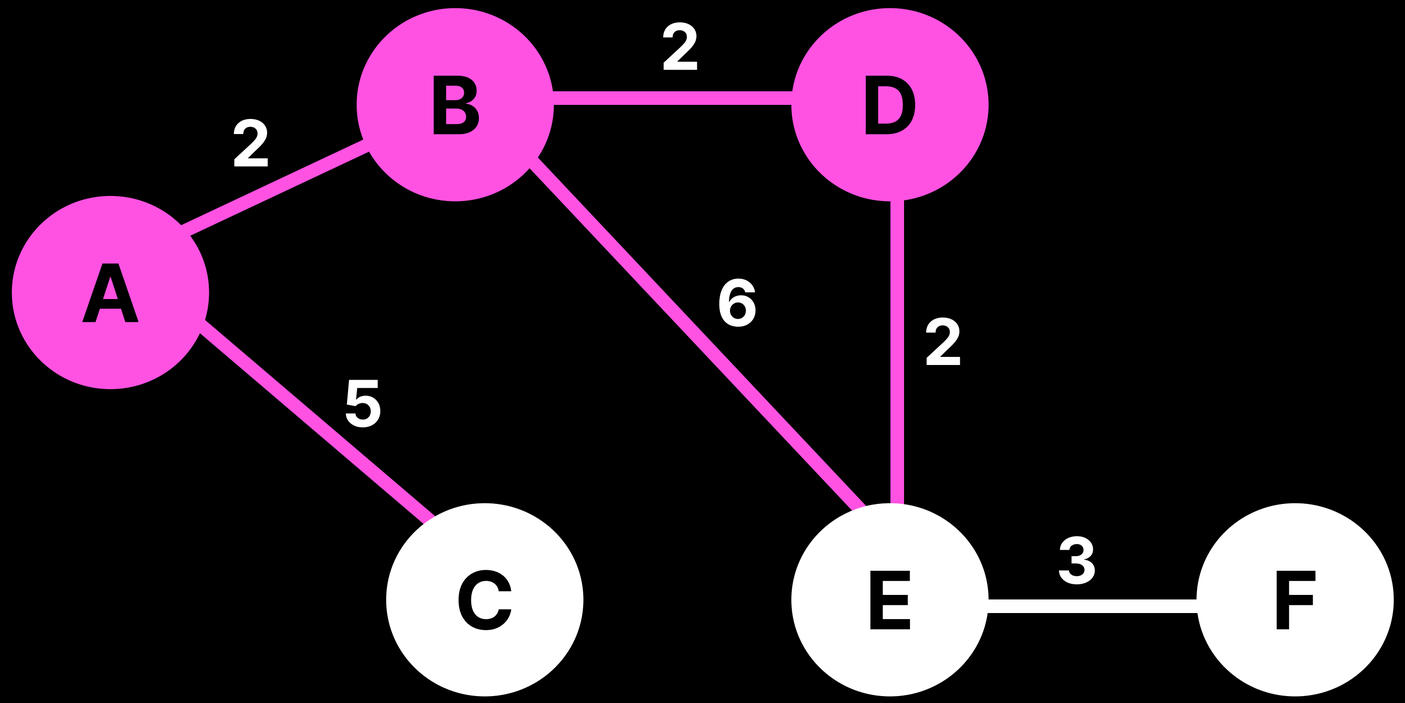
queue	distance
A	0
B	2
C	5
D	2+2
E	2+6
F	∞

Dijkstra



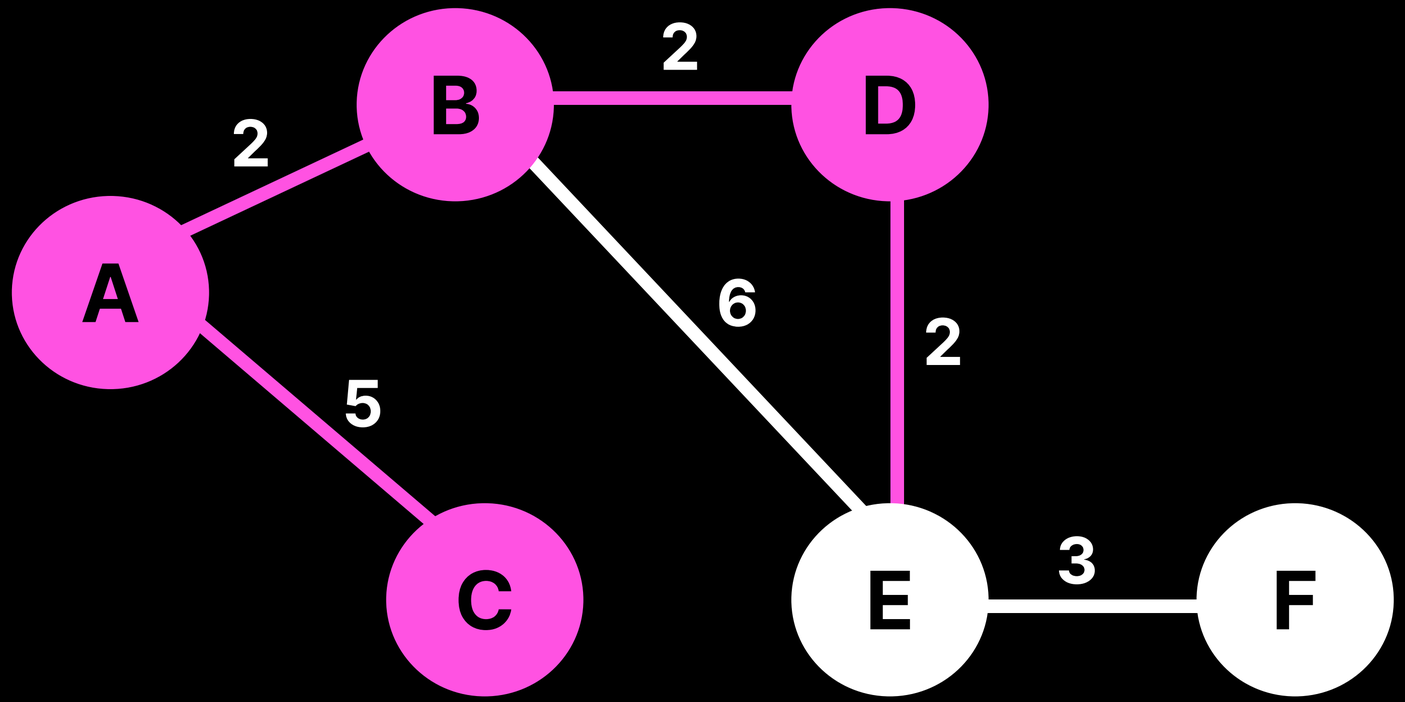
queue	distance
A	0
B	2
C	5
D	4 den minste
E	8
F	∞

Dijkstra



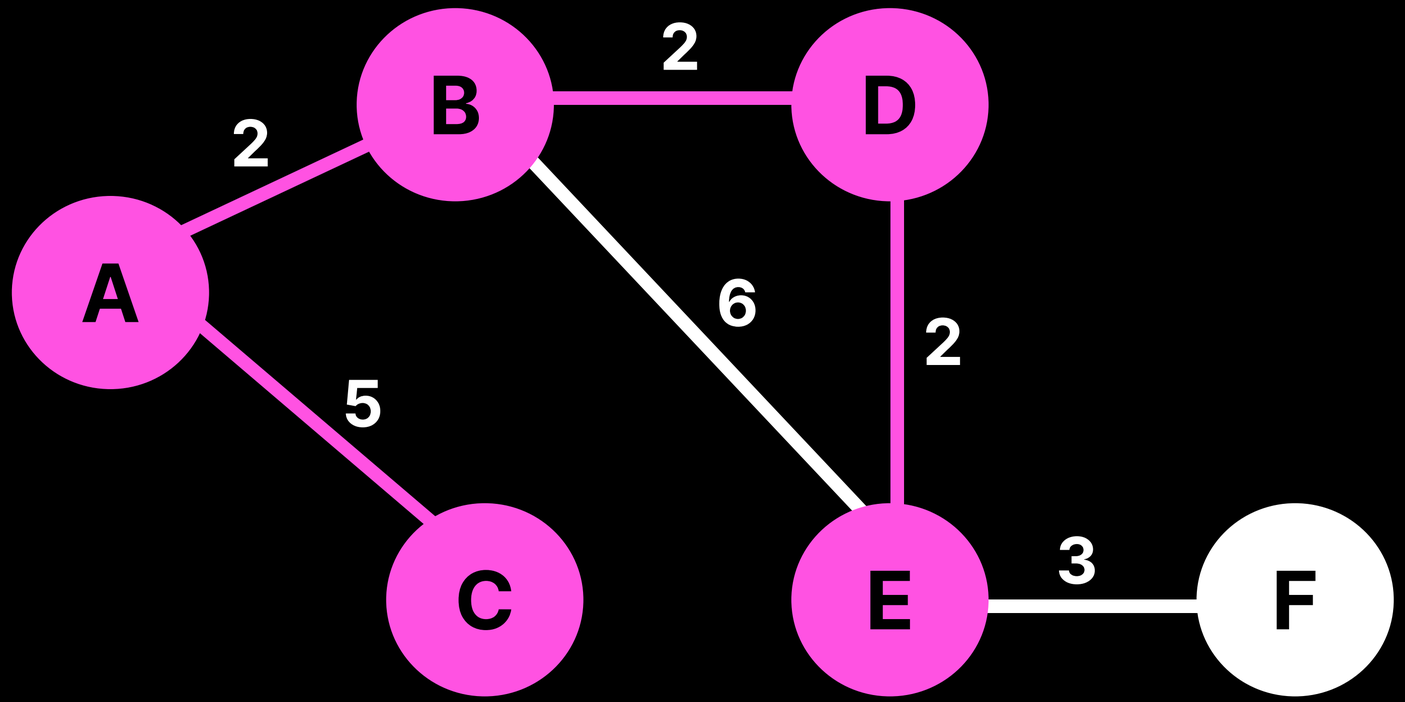
queue	distance
A	0
B	2
C	5
D	4
E	8>6 denne oppdateres!
F	∞

Dijkstra



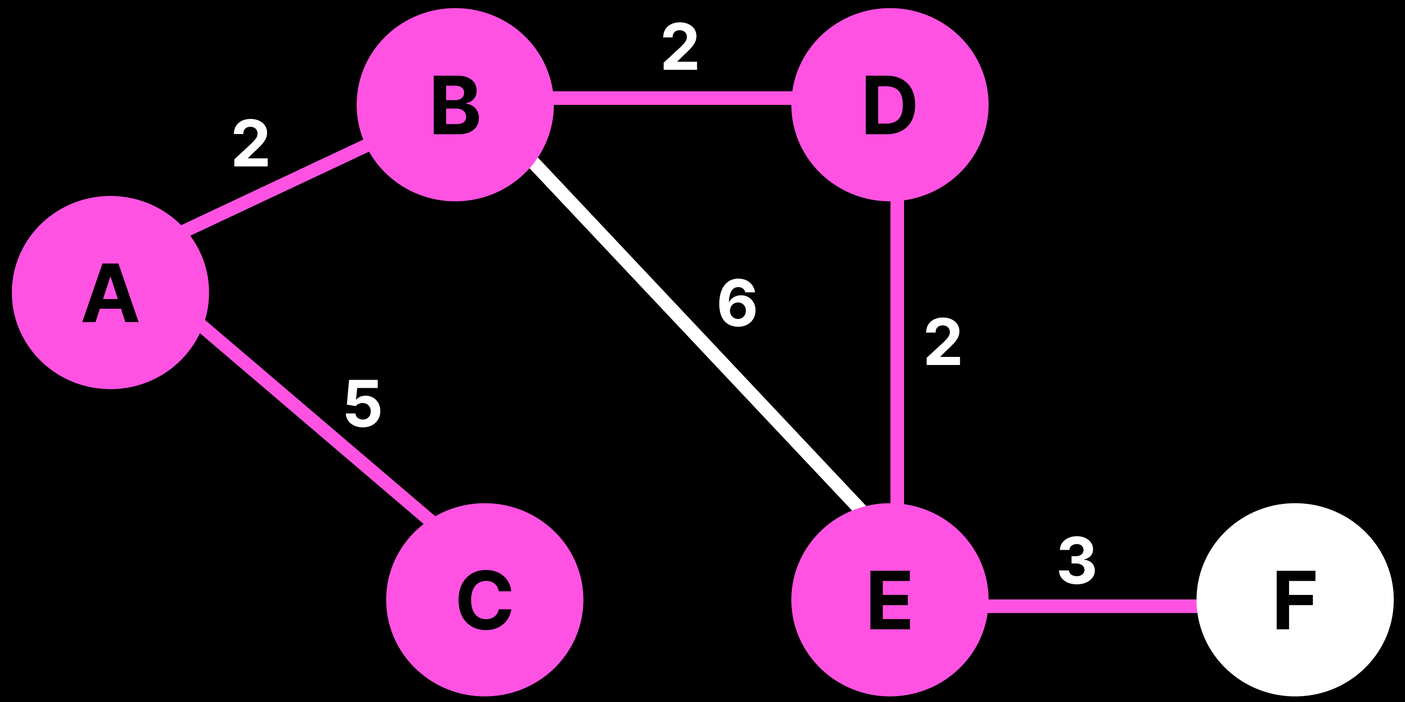
queue	distance
A	0
B	2
C	5 den minste
D	4
E	6
F	∞

Dijkstra



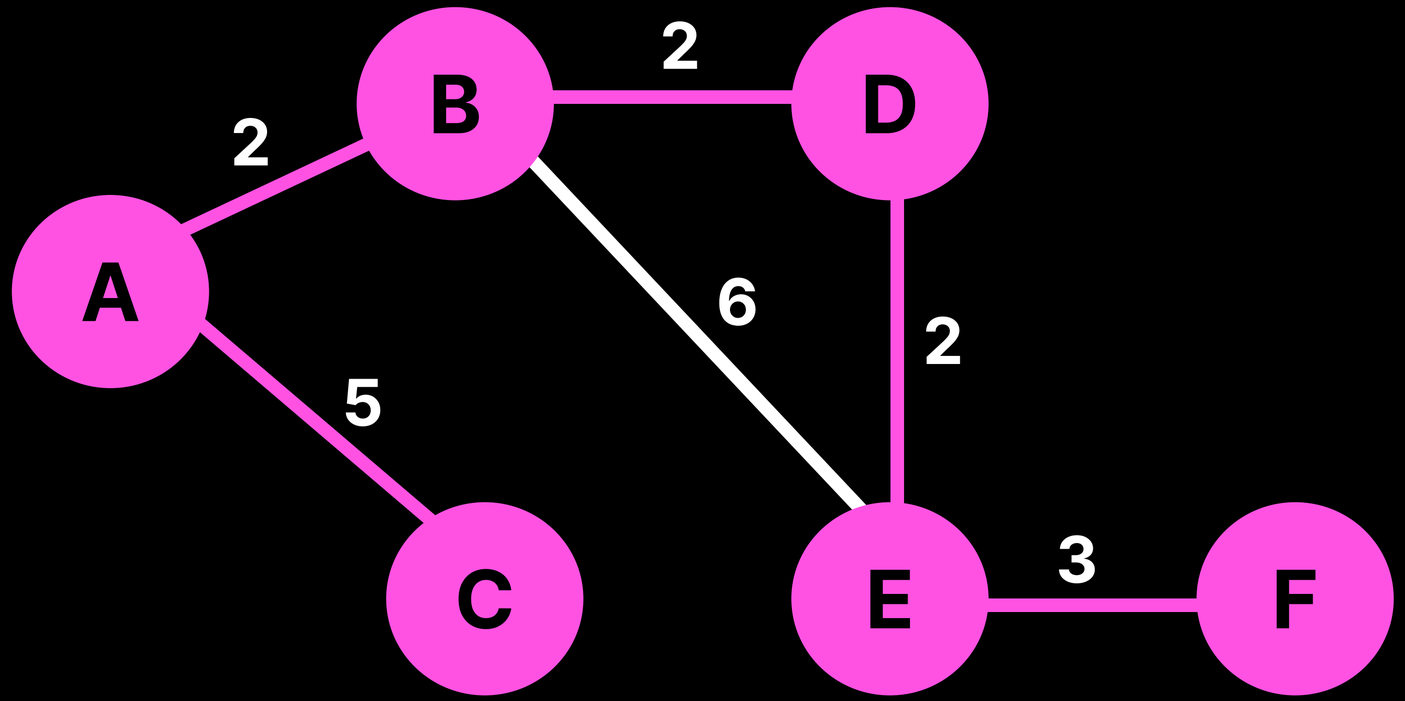
queue	distance
A	0
B	2
C	5
D	4
E	6 den minste
F	∞

Dijkstra



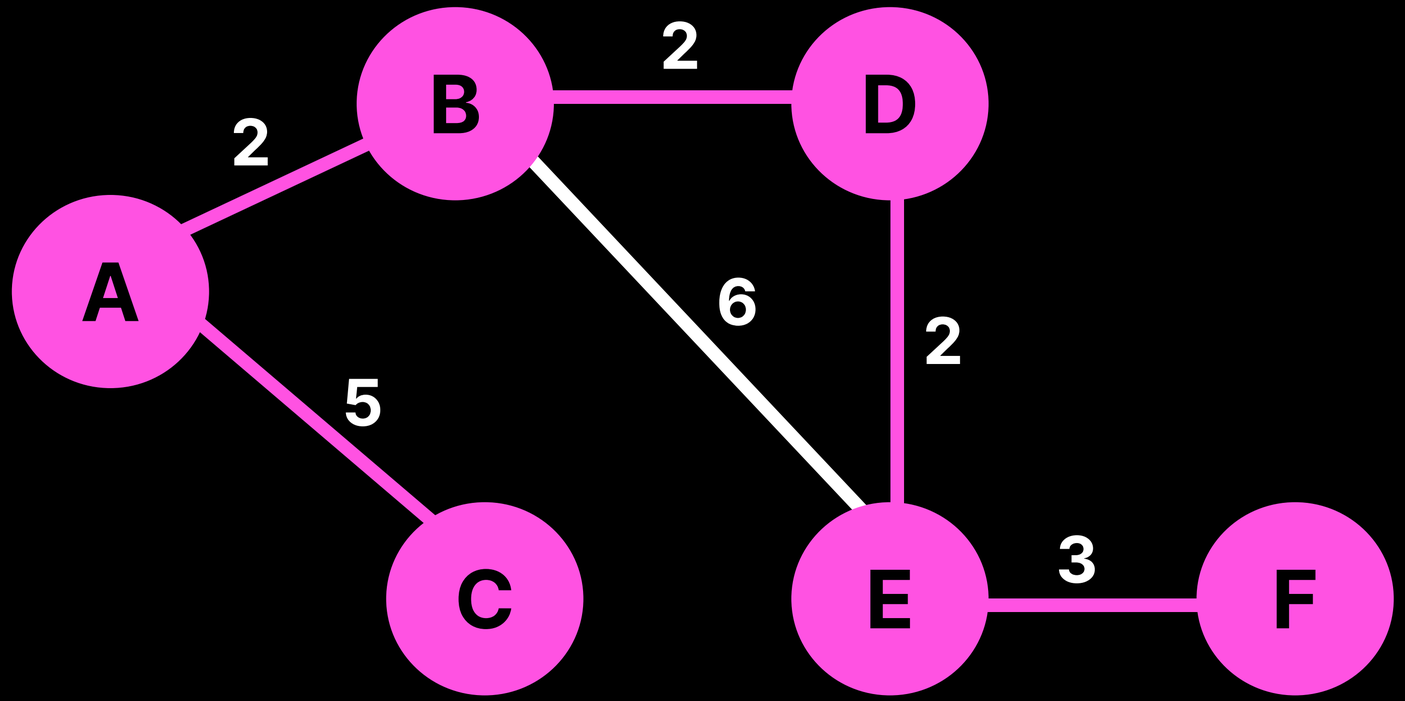
queue	distance
A	0
B	2
C	5
D	4
E	6
F	2+2+2+3

Dijkstra



queue	distance
A	0
B	2
C	5
D	4
E	6
F	9

Dijkstra



queue	distance
A	0
B	2
C	5
D	4
E	6
F	9

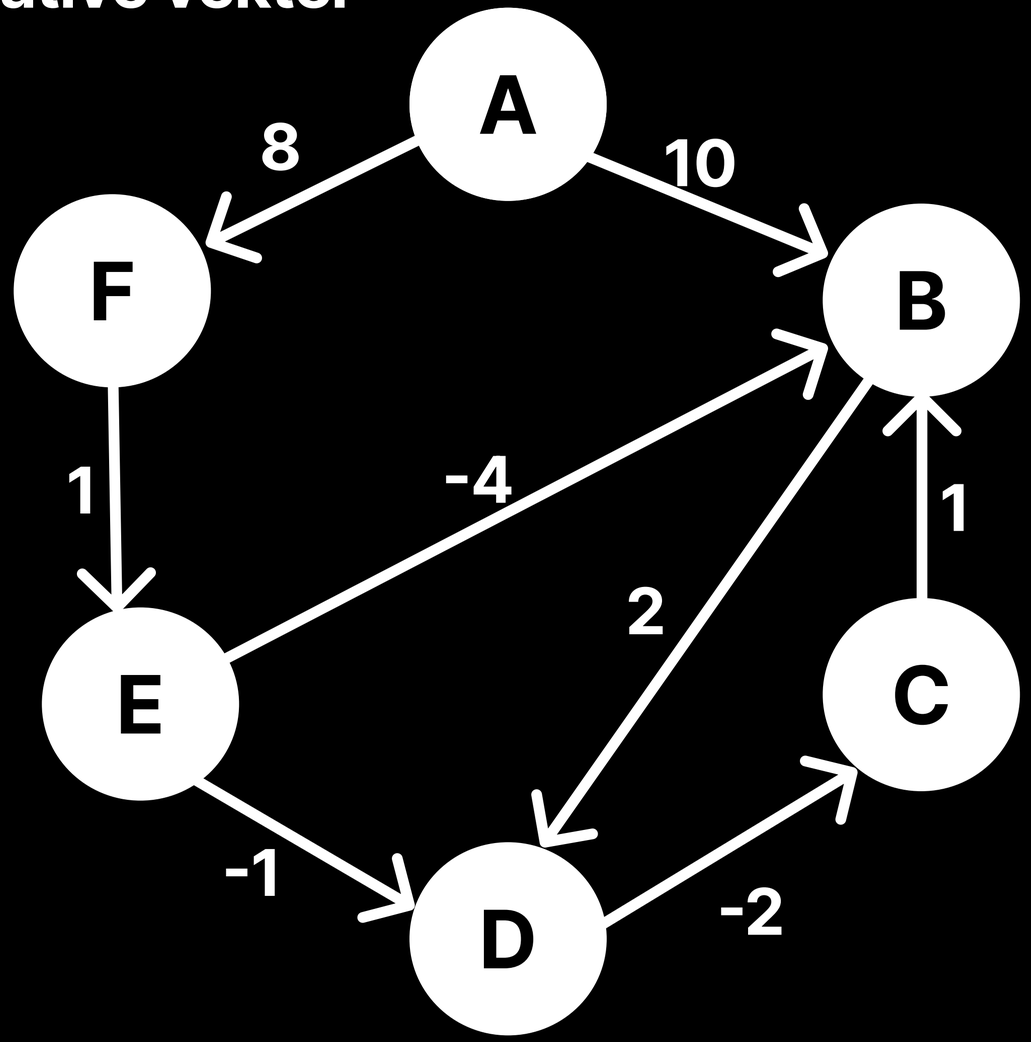
Bellman-Ford

Når det **finnes kanter med negativ vekt**

Fungerer både for rettede og urettede grafer

Bellman-Ford

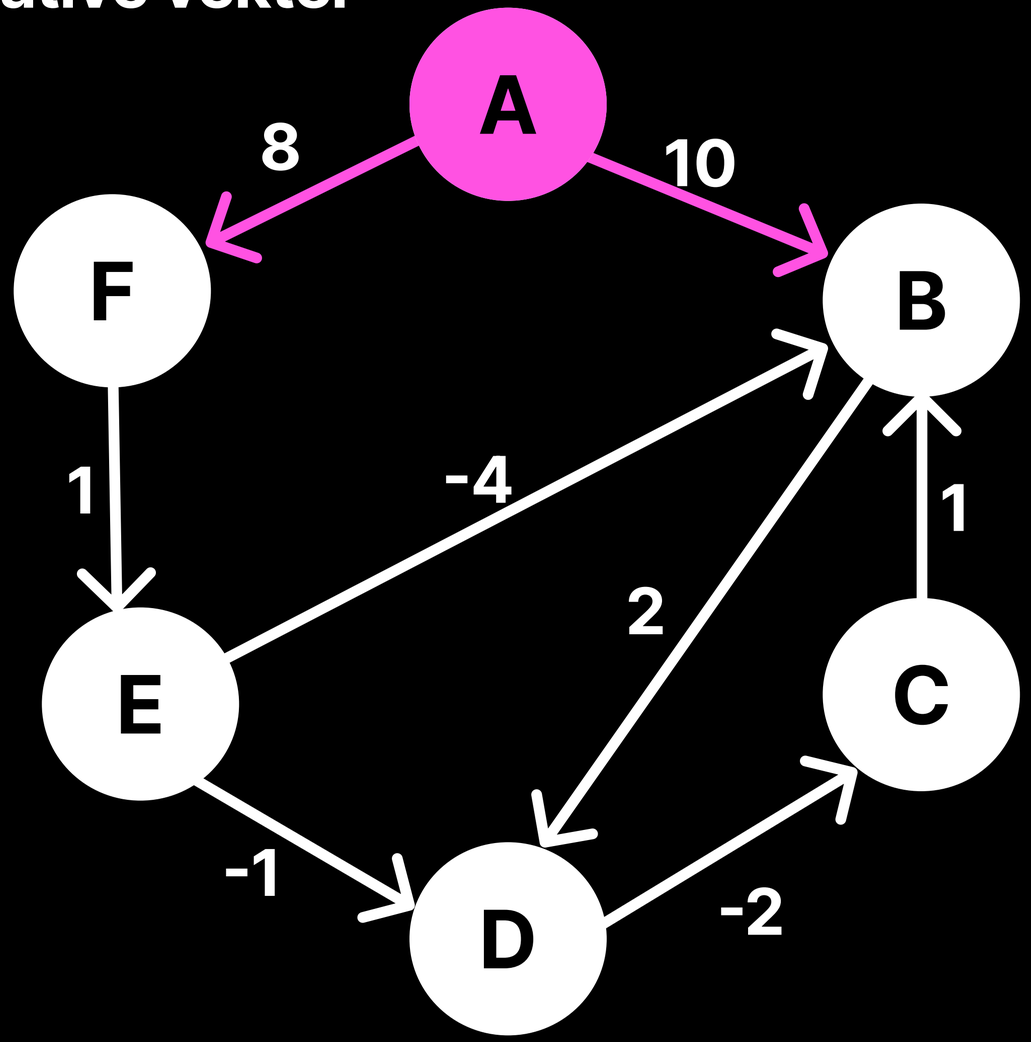
Negative vekter



queue	distance
A	0
B	∞
C	∞
D	∞
E	∞
F	∞

Bellman-Ford

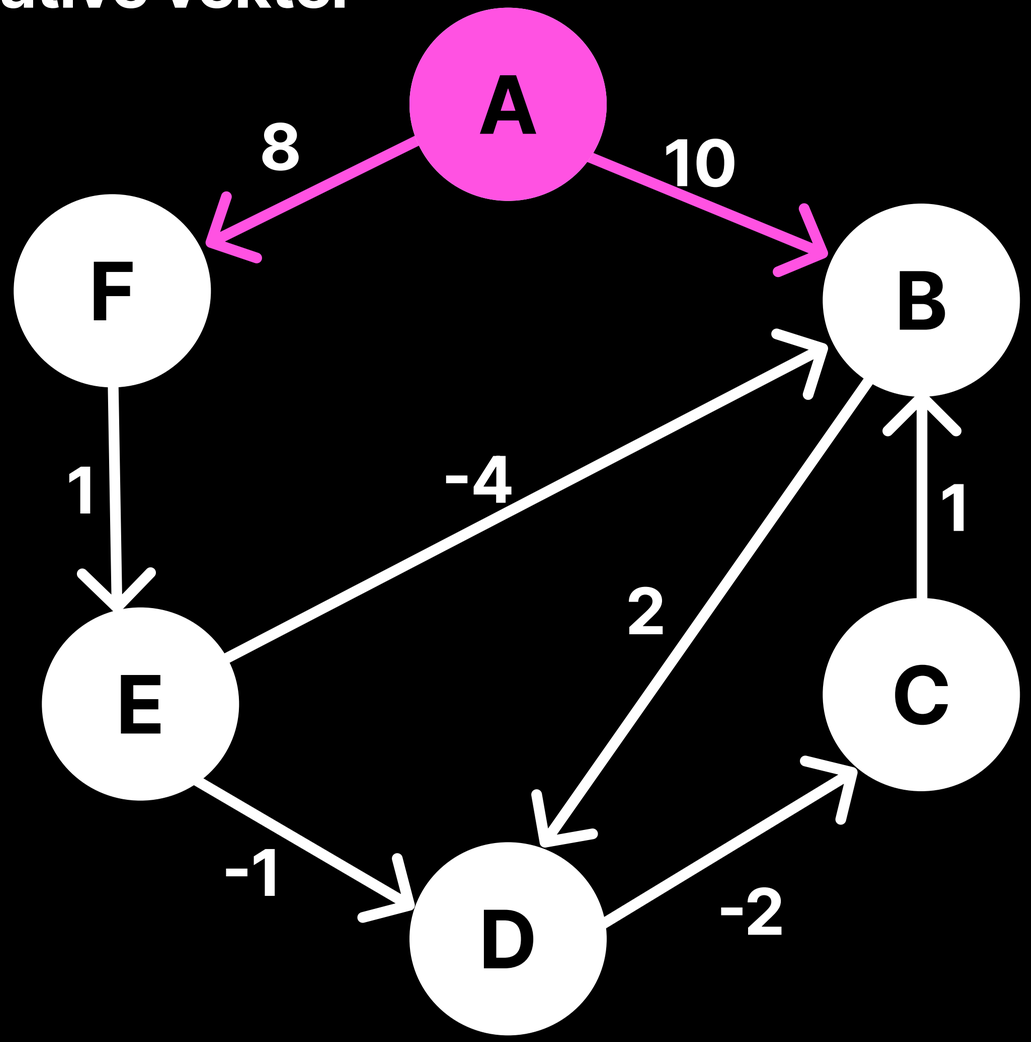
Negative vekter



queue	distance
A	0
B	∞
C	∞
D	∞
E	∞
F	∞

Bellman-Ford

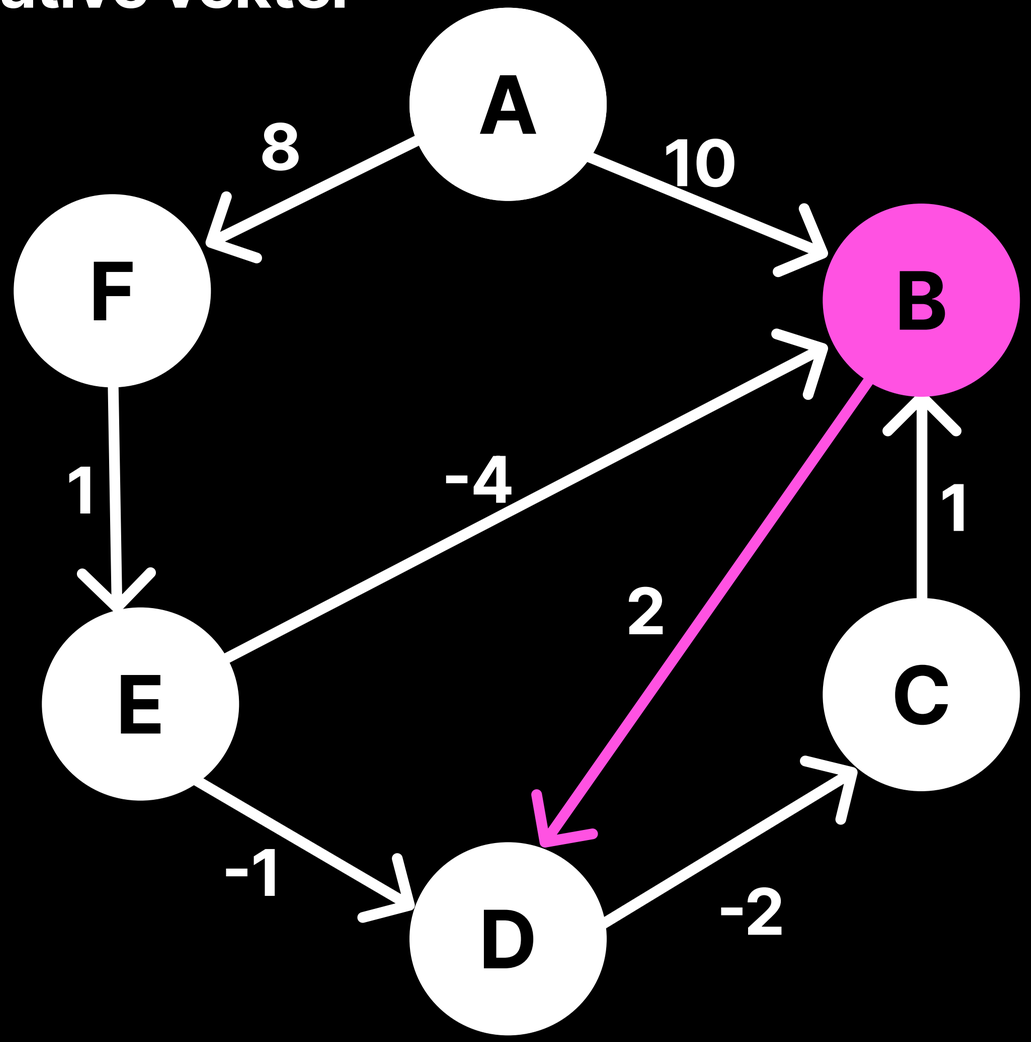
Negative vekter



queue	distance
A	0
B	10
C	∞
D	∞
E	∞
F	8

Bellman-Ford

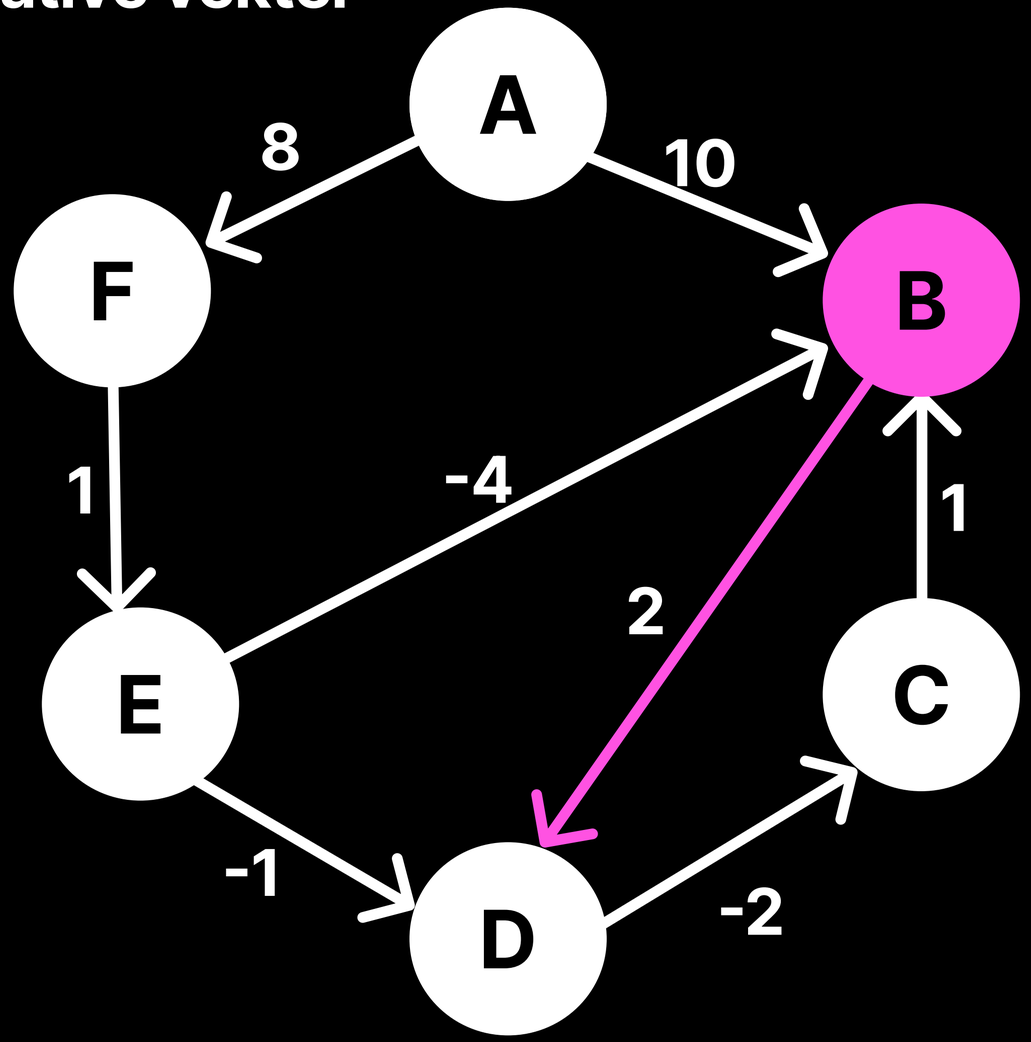
Negative vekter



queue	distance
A	0
B	10
C	∞
D	∞
E	∞
F	8

Bellman-Ford

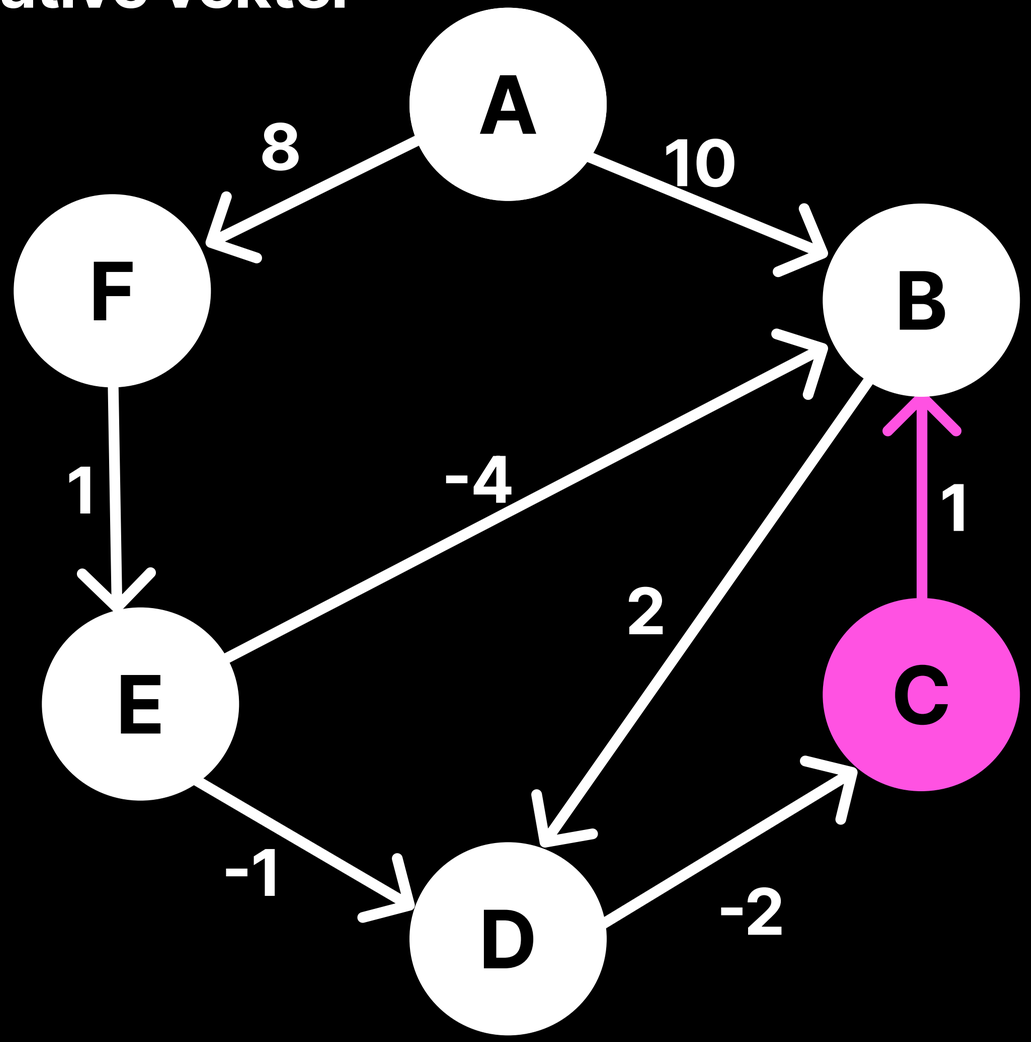
Negative vekter



queue	distance
A	0
B	10
C	∞
D	12
E	∞
F	8

Bellman-Ford

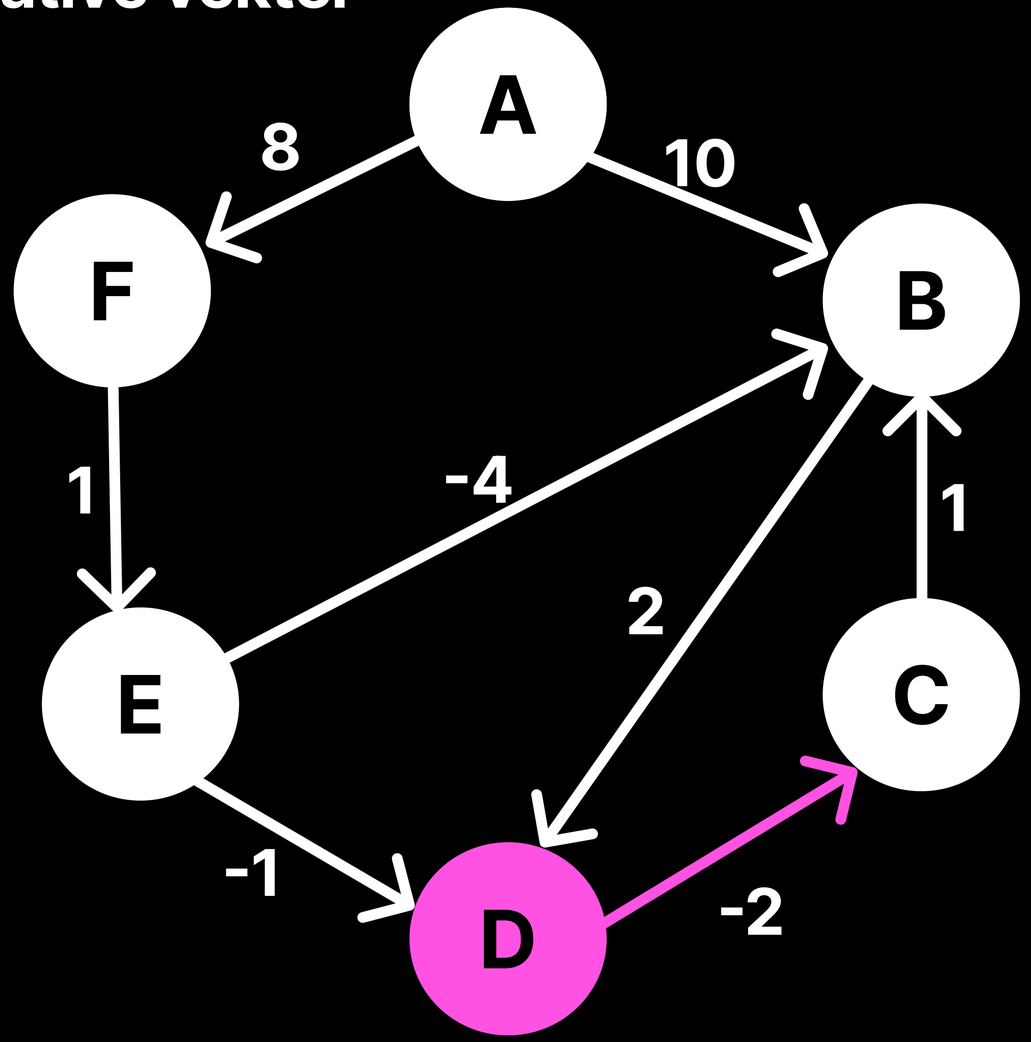
Negative vekter



queue	distance
A	0
B	10
C	∞
D	12
E	∞
F	8

Bellman-Ford

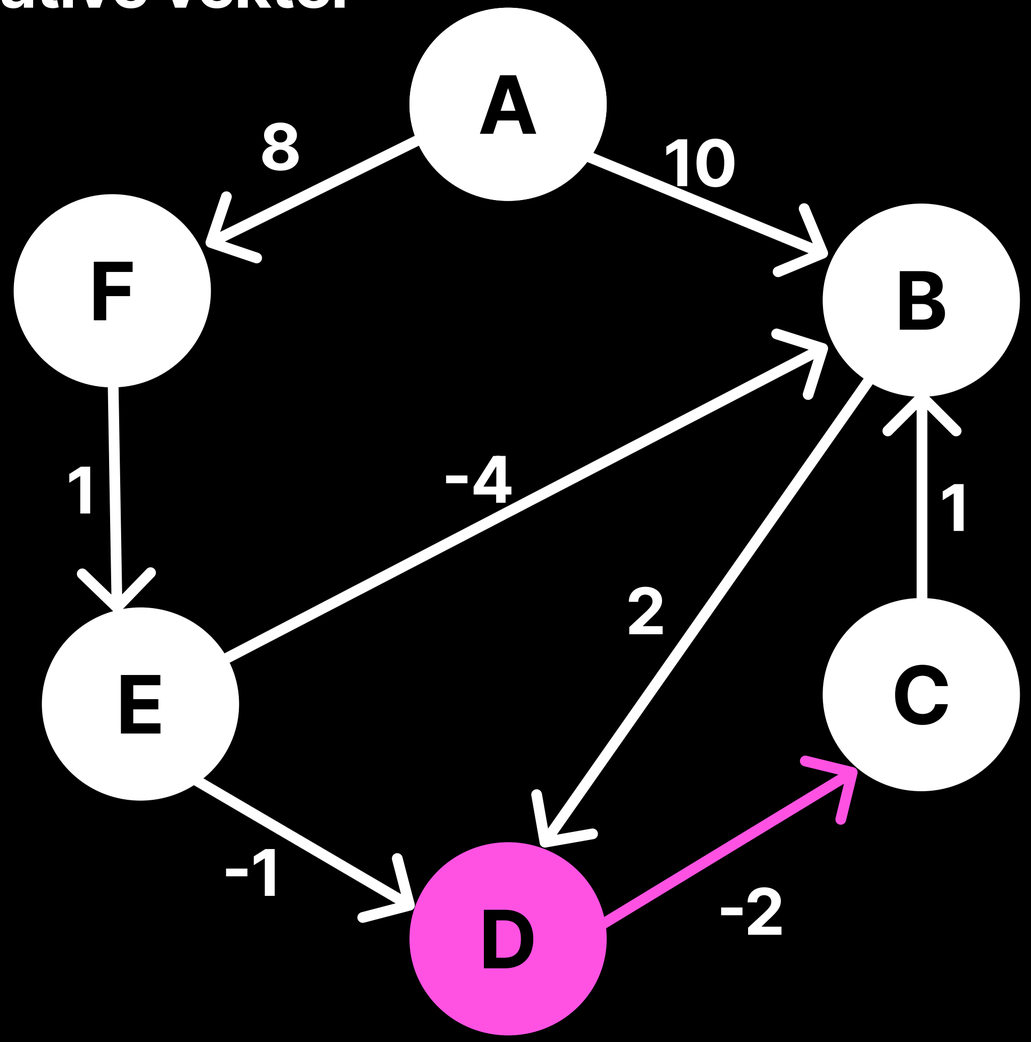
Negative vekter



queue	distance
A	0
B	10
C	∞
D	12
E	∞
F	8

Bellman-Ford

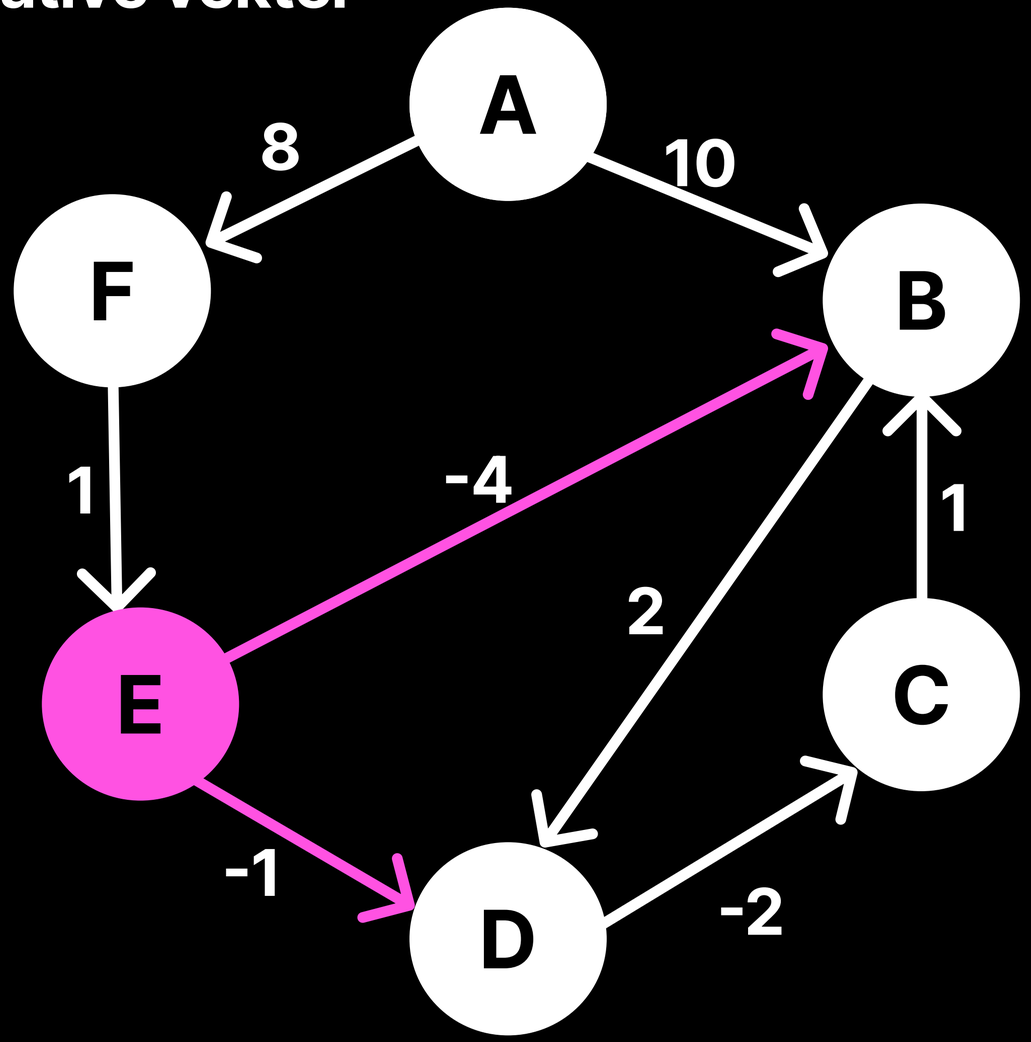
Negative vekter



queue	distance
A	0
B	10
C	10
D	12
E	∞
F	8

Bellman-Ford

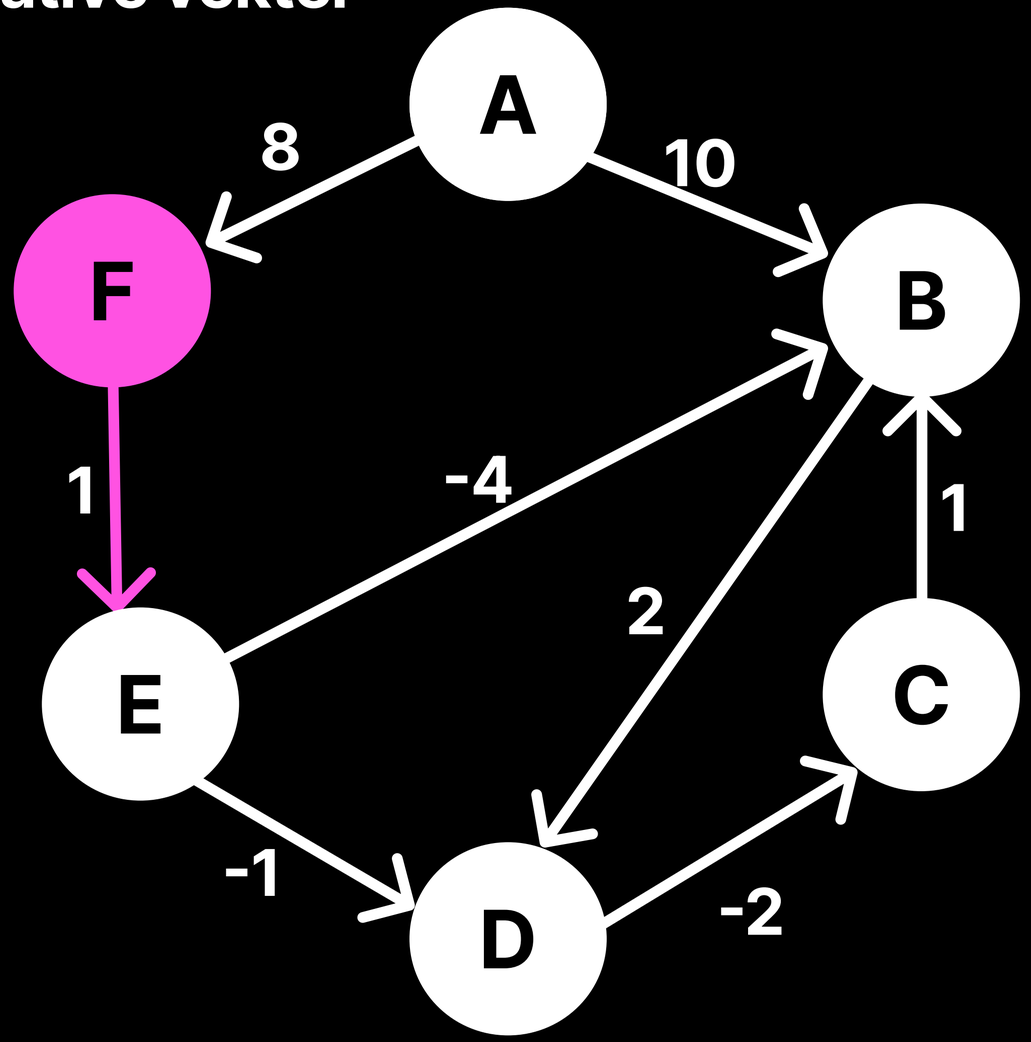
Negative vekter



queue	distance
A	0
B	10
C	10
D	12
E	∞
F	8

Bellman-Ford

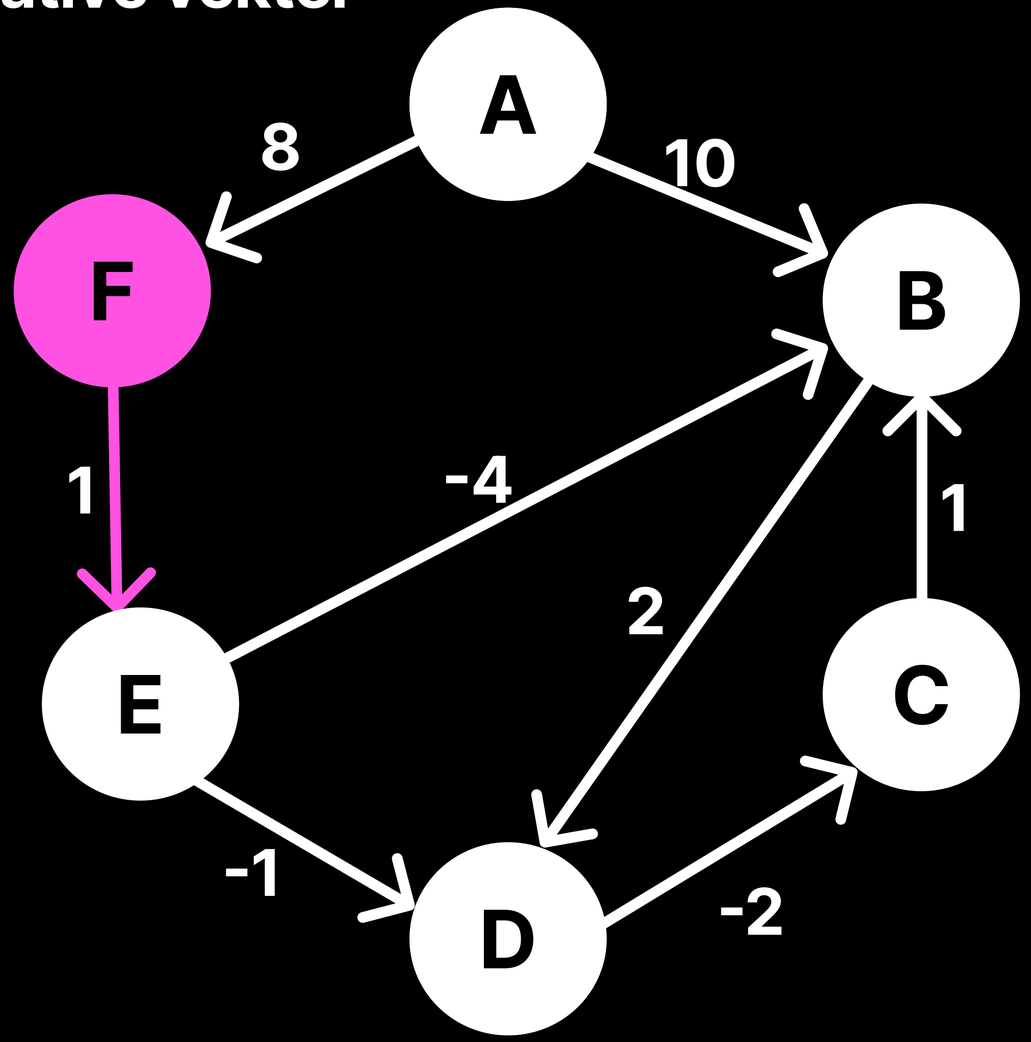
Negative vekter



queue	distance
A	0
B	10
C	10
D	12
E	∞
F	8

Bellman-Ford

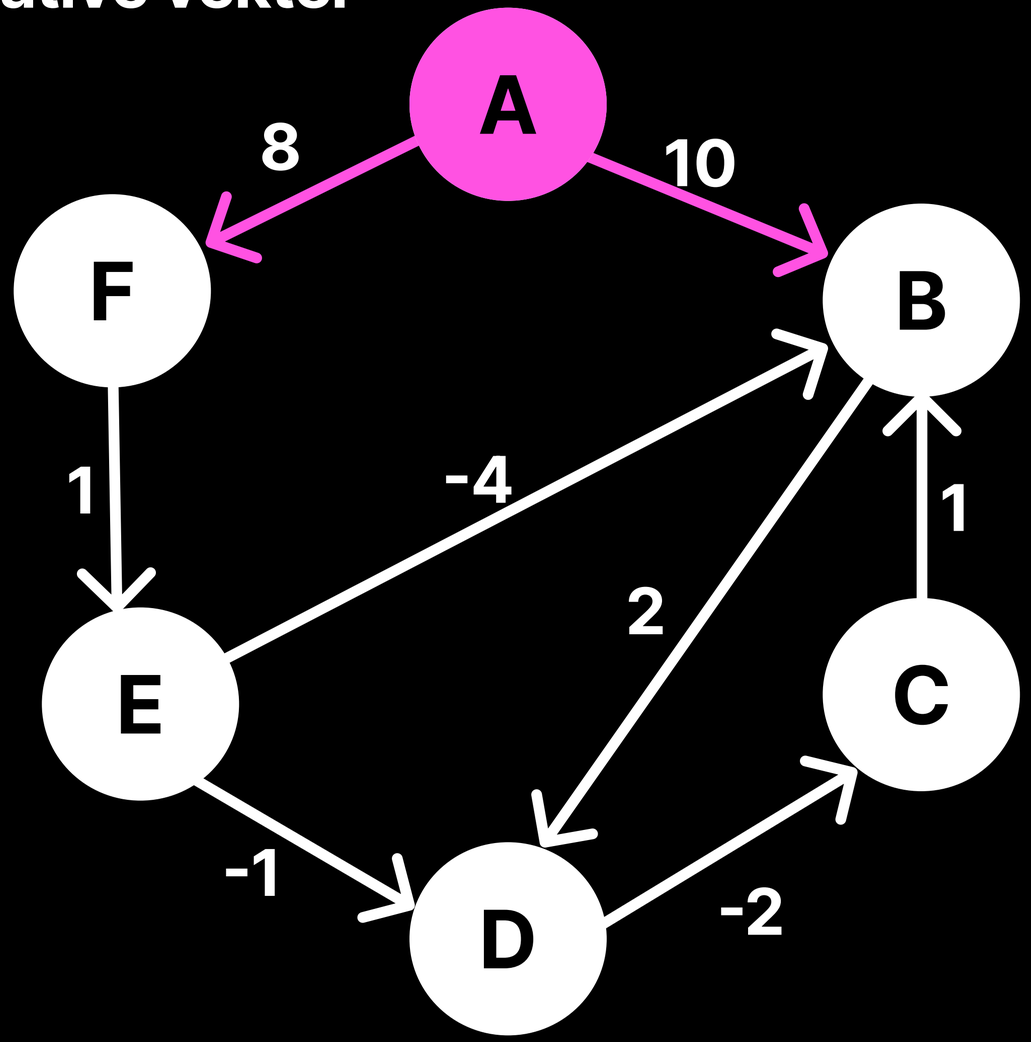
Negative vekter



queue	distance
A	0
B	10
C	10
D	12
E	9
F	8

Bellman-Ford

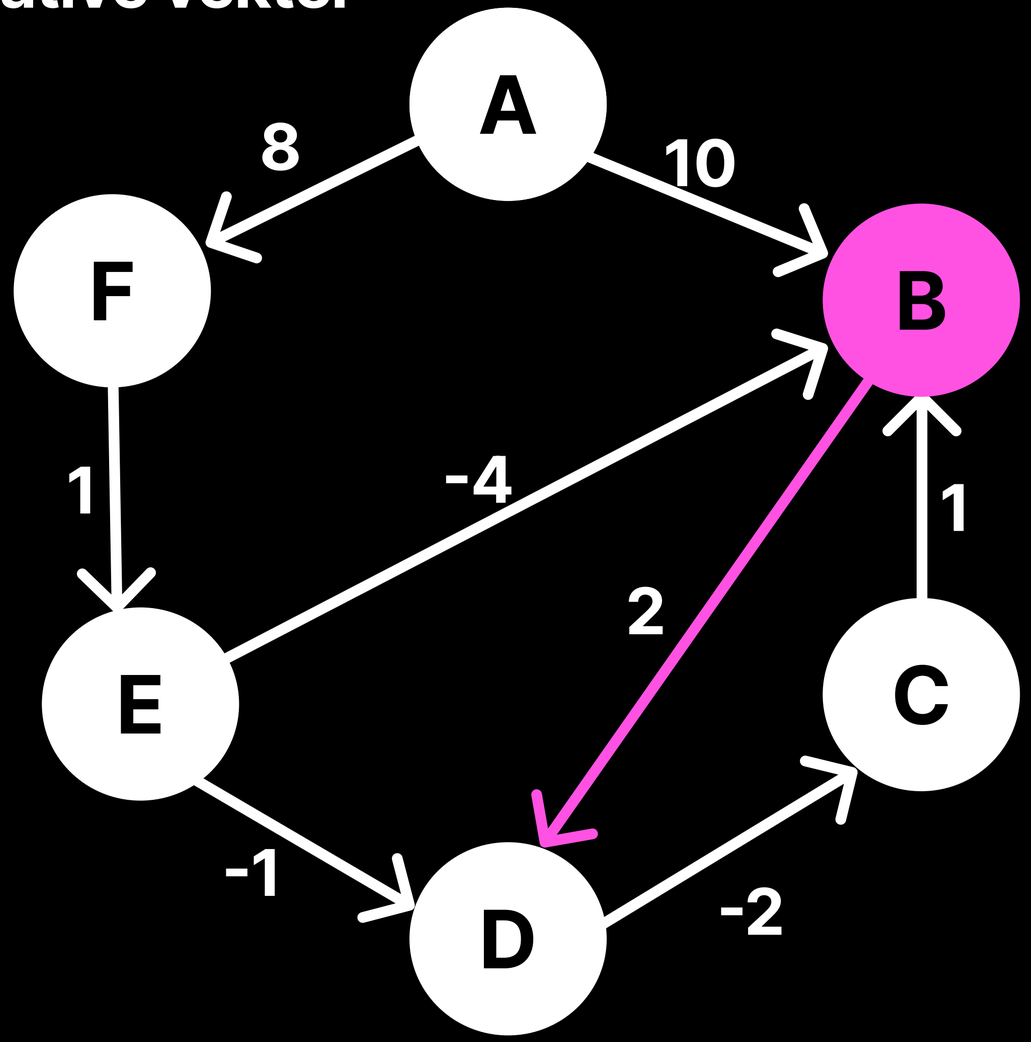
Negative vekter



queue	distance
A	0
B	10=10
C	10
D	12
E	9
F	8=10

Bellman-Ford

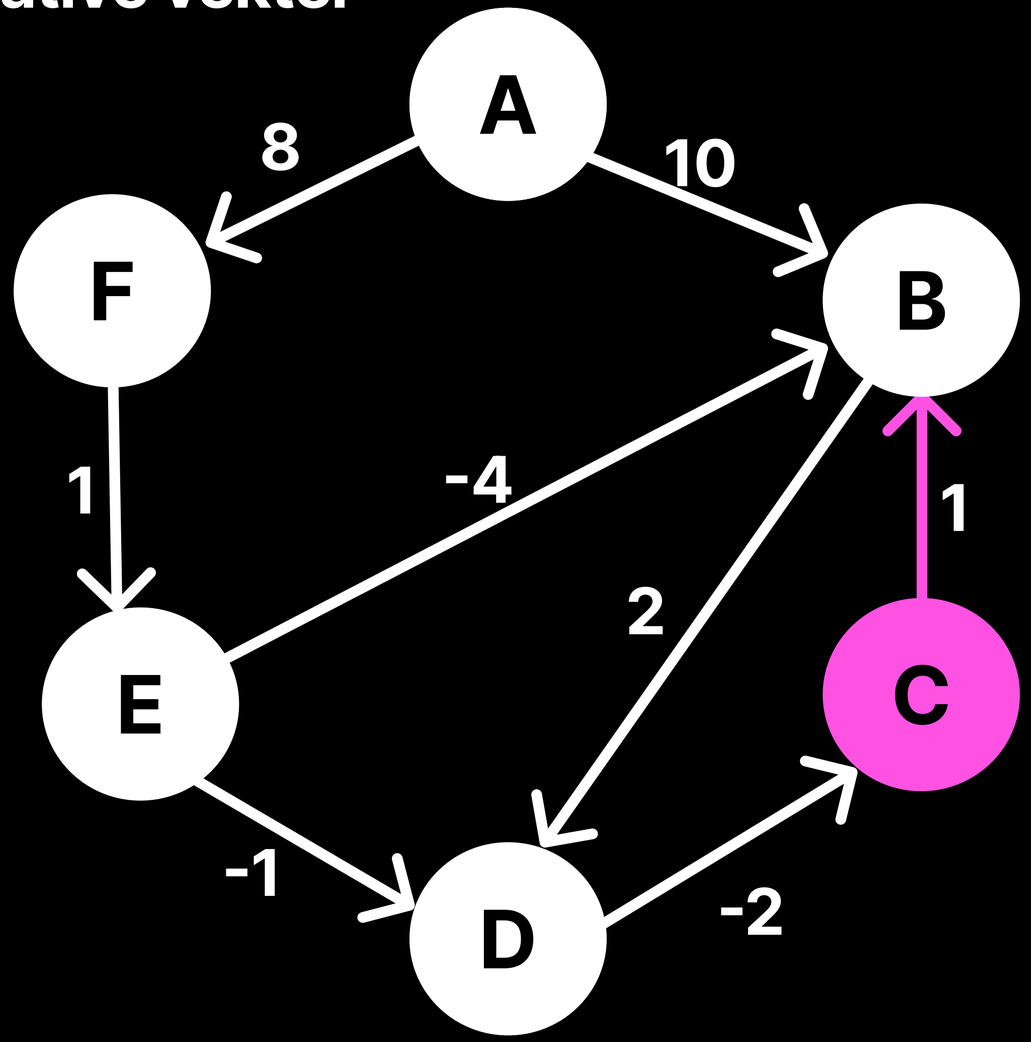
Negative vekter



queue	distance
A	0
B	10
C	10
D	12=12
E	9
F	8

Bellman-Ford

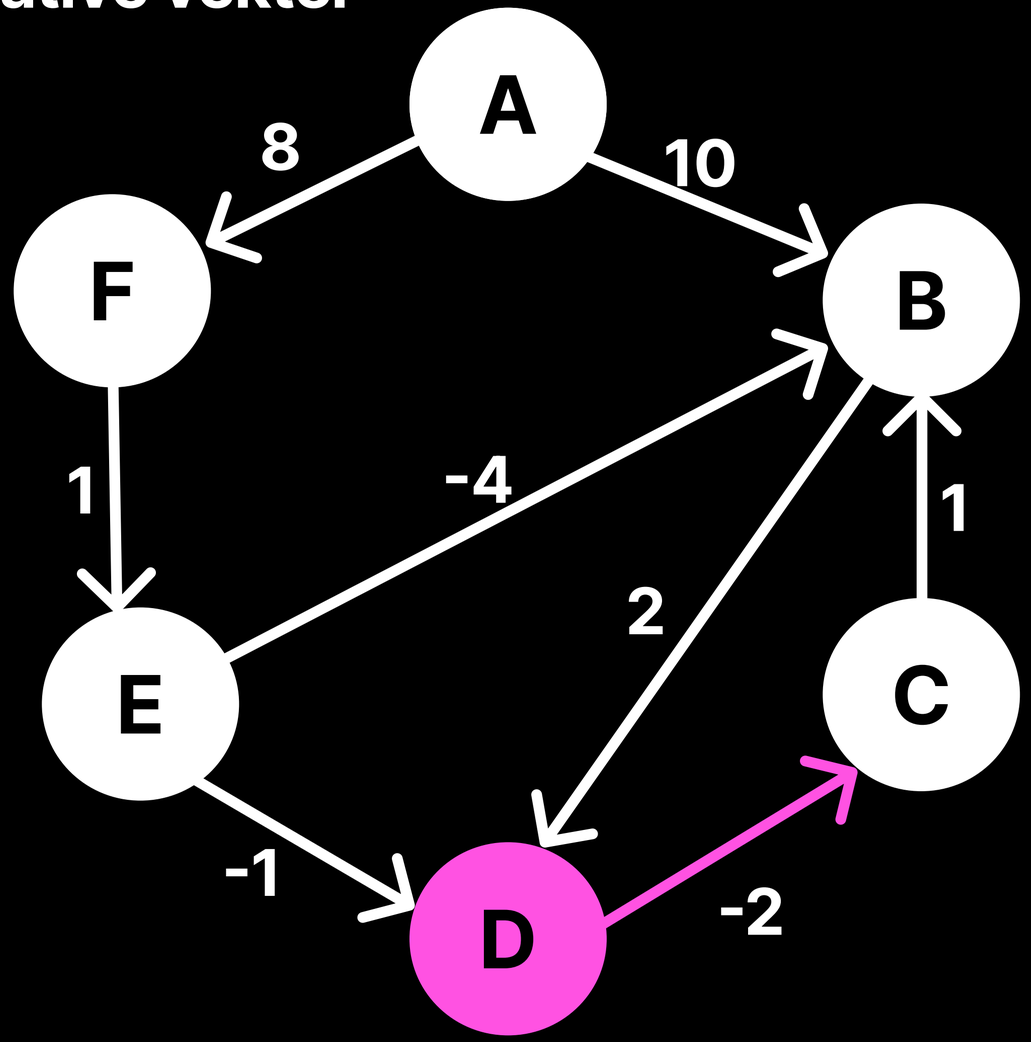
Negative vekter



queue	distance
A	0
B	10<11
C	13
D	12
E	9
F	8

Bellman-Ford

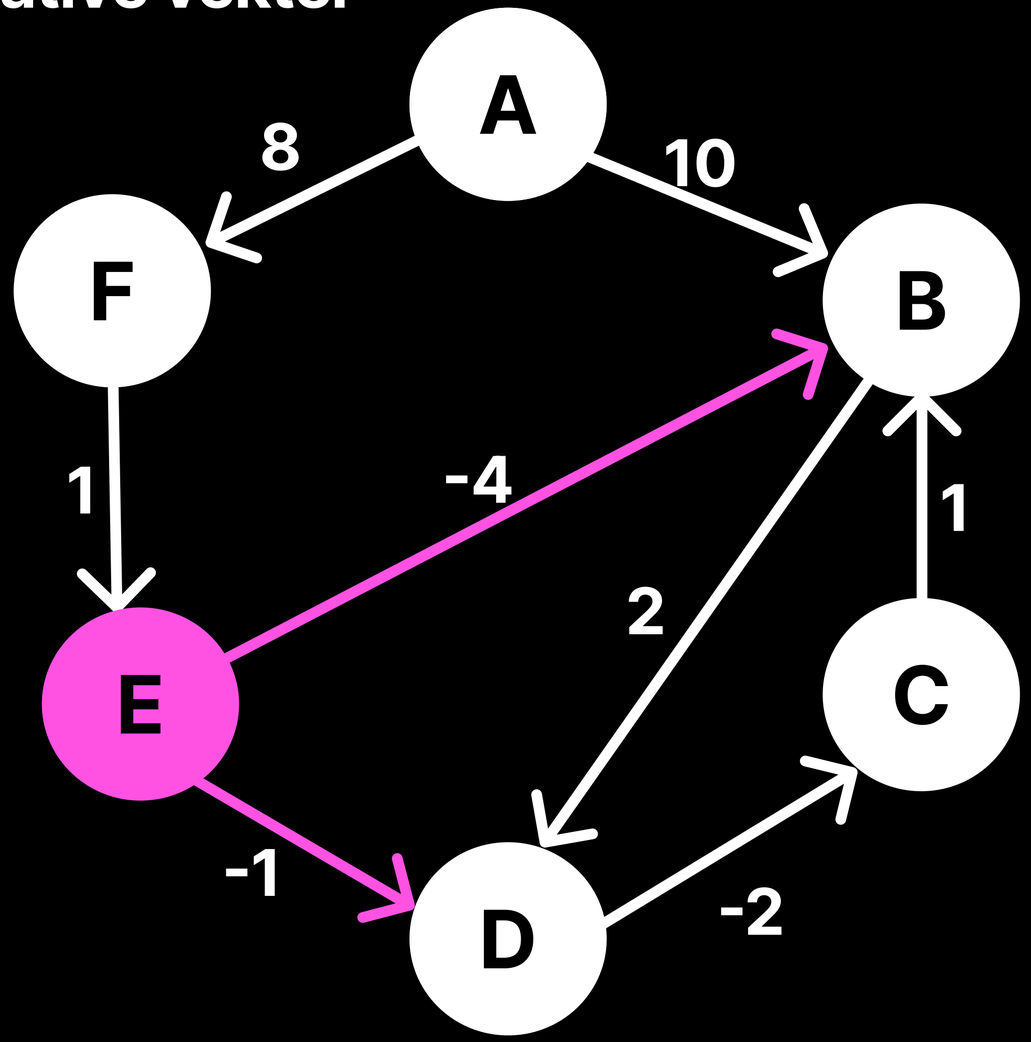
Negative vekter



queue	distance
A	0
B	10
C	10=10
D	12
E	9
F	8

Bellman-Ford

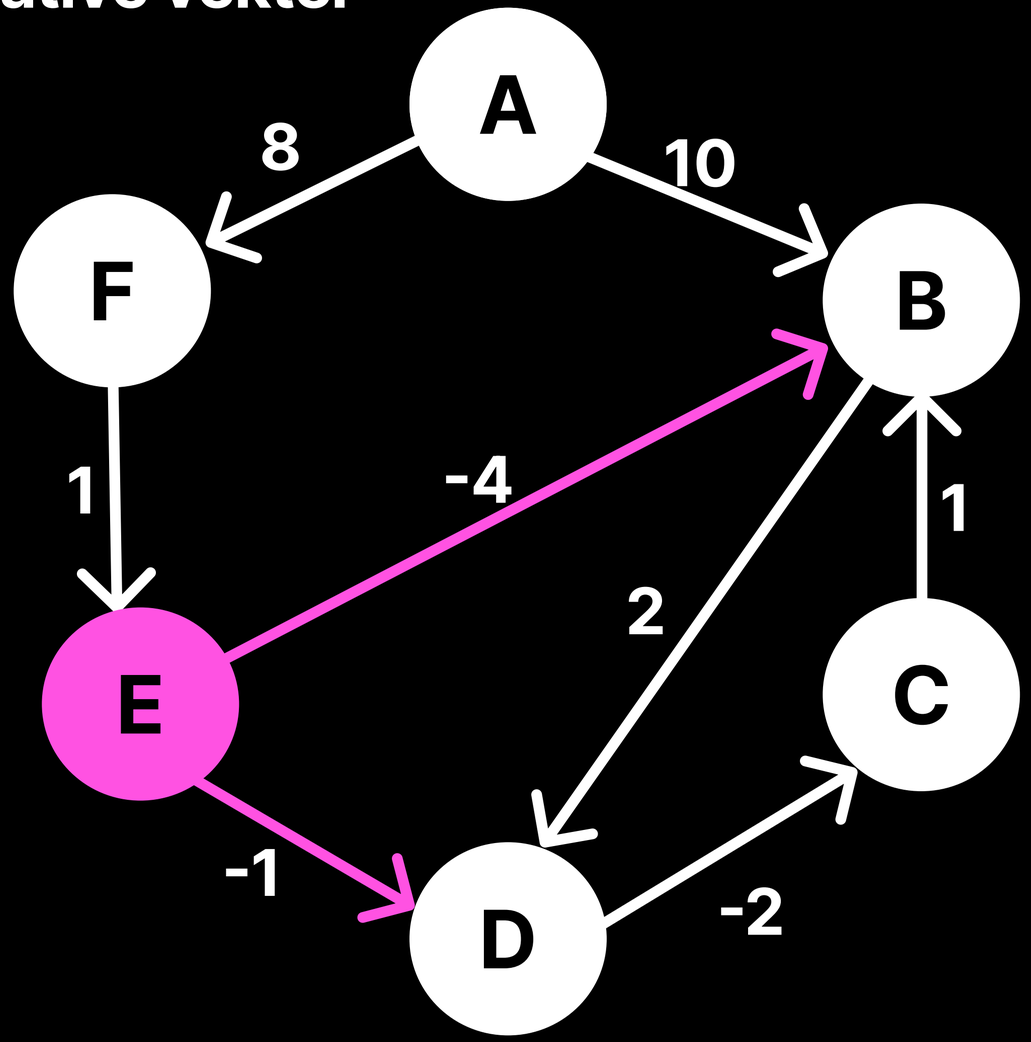
Negative vekter



queue	distance
A	0
B	10>5
C	10
D	12>8
E	9
F	8

Bellman-Ford

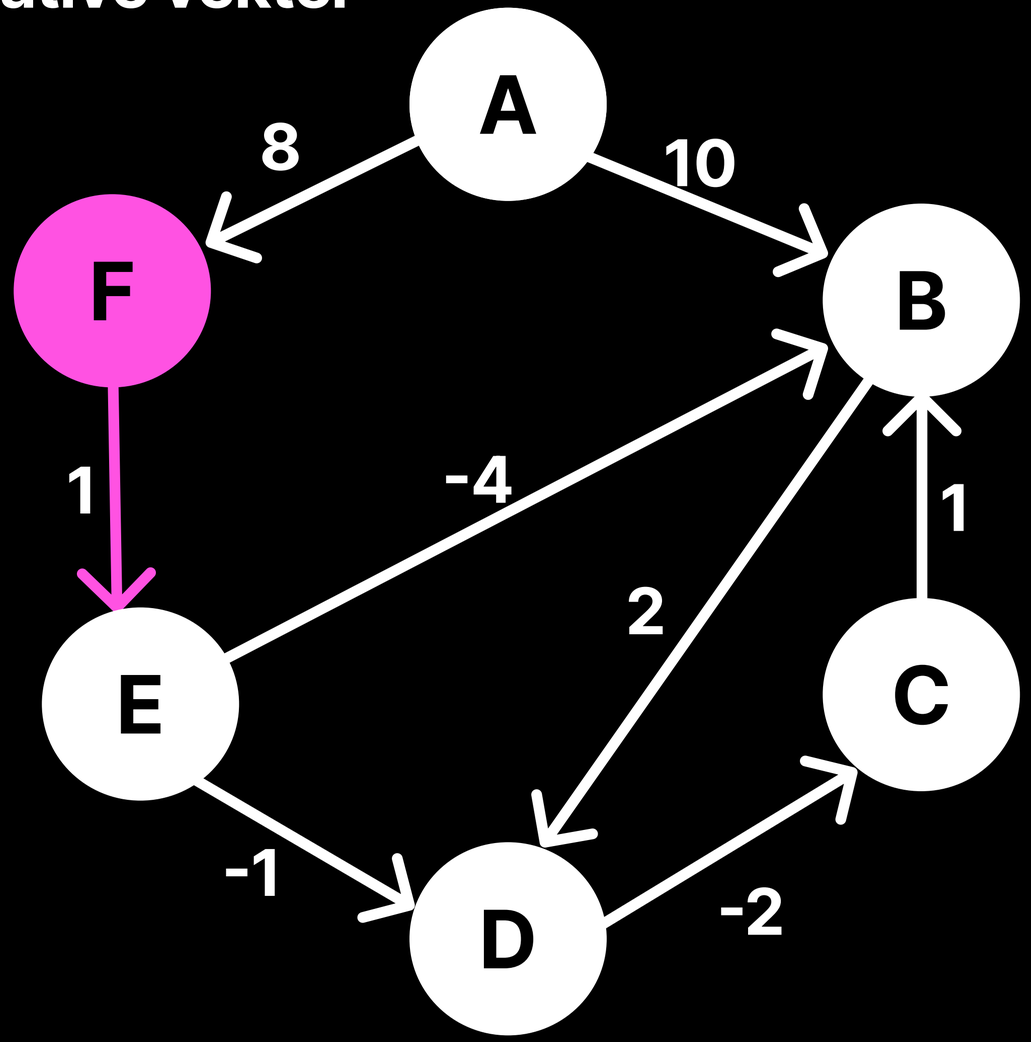
Negative vekter



queue	distance
A	0
B	5
C	10
D	8
E	9
F	8

Bellman-Ford

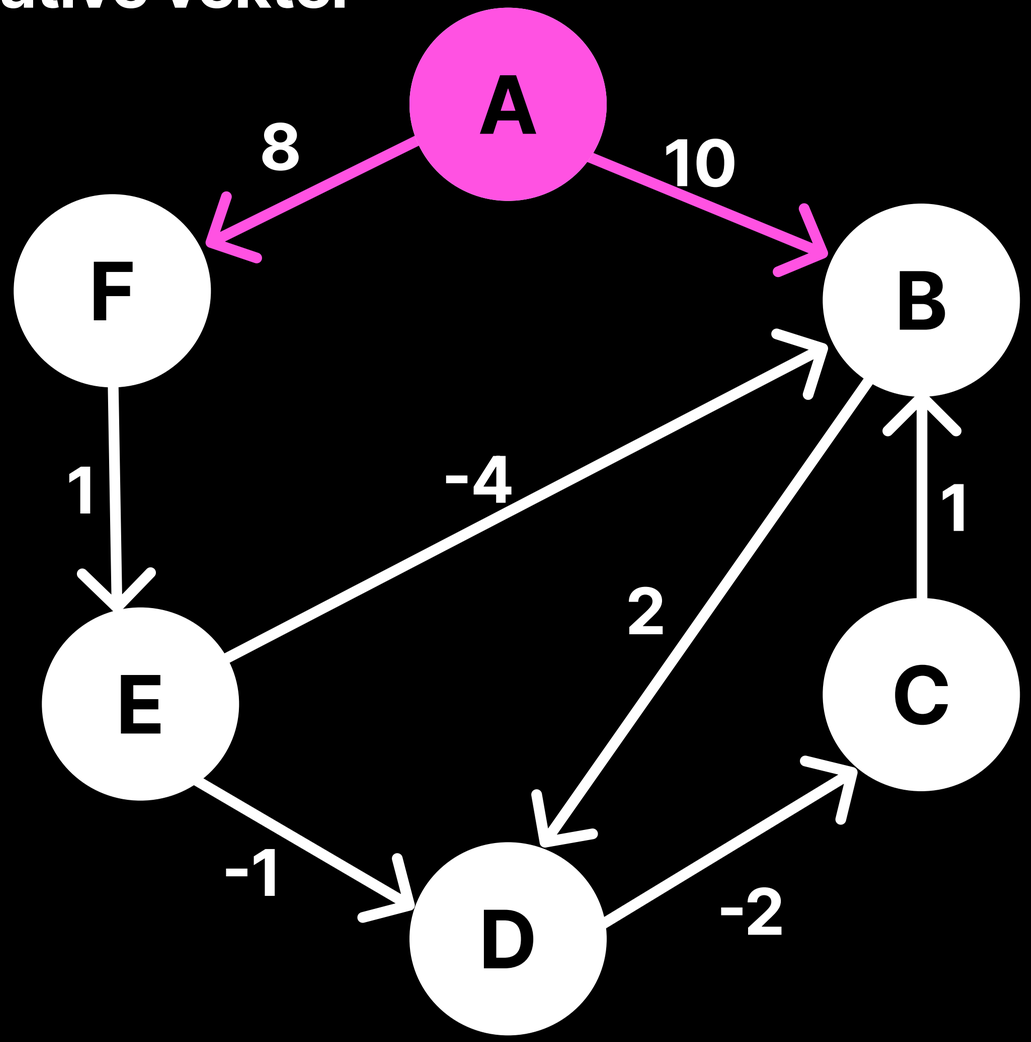
Negative vekter



queue	distance
A	0
B	5
C	10
D	8
E	9=9
F	8

Bellman-Ford

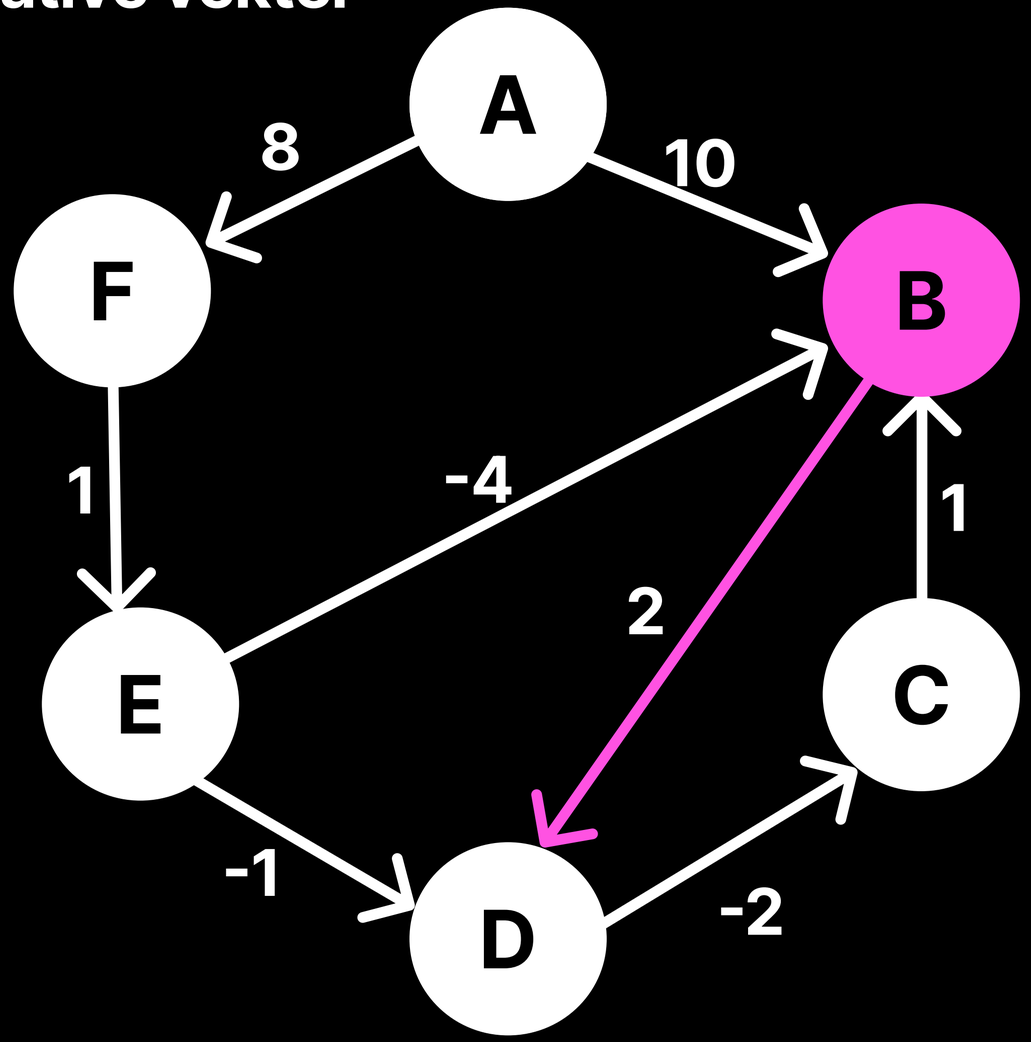
Negative vekter



queue	distance
A	0
B	5<10
C	10
D	8
E	9
F	8=8

Bellman-Ford

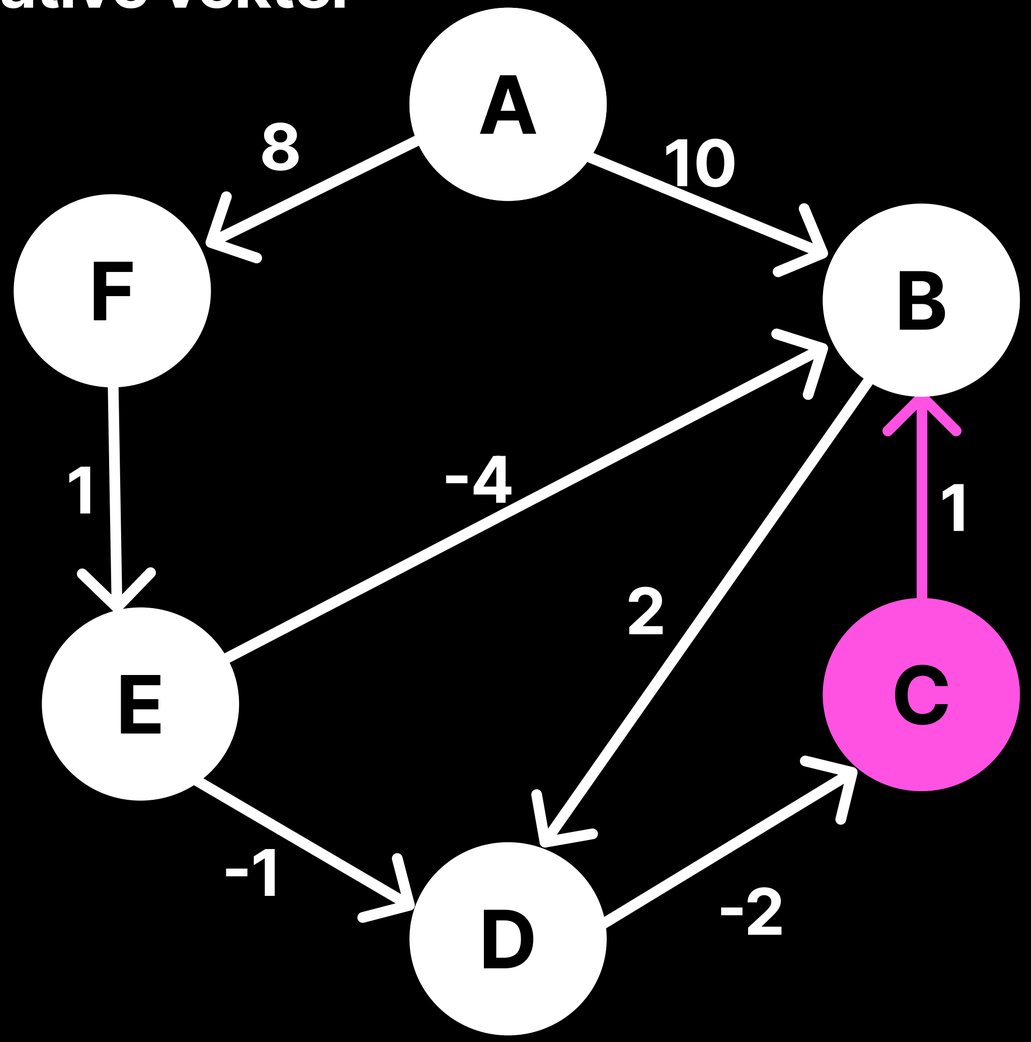
Negative vekter



queue	distance
A	0
B	5
C	10
D	8<12
E	9
F	8

Bellman-Ford

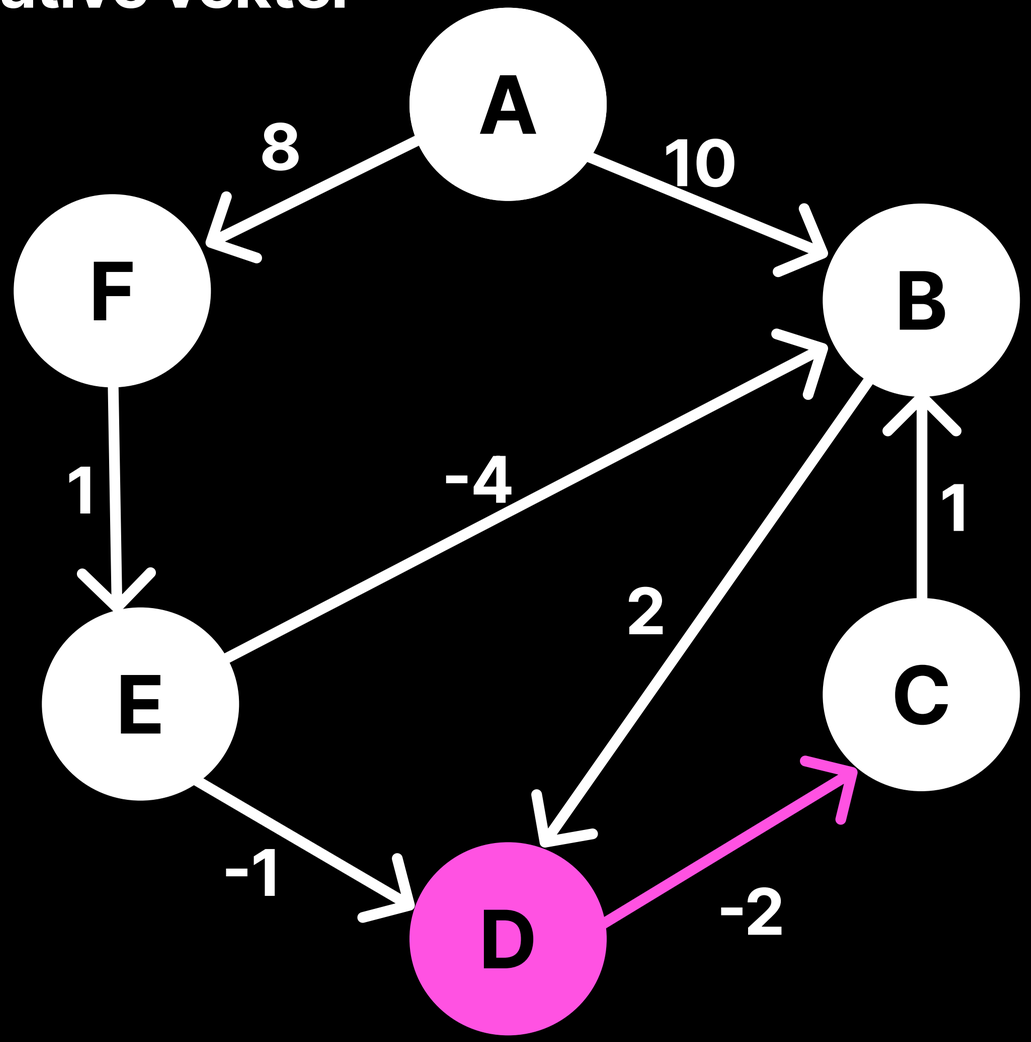
Negative vekter



queue	distance
A	0
B	5<11
C	10
D	8
E	9
F	8

Bellman-Ford

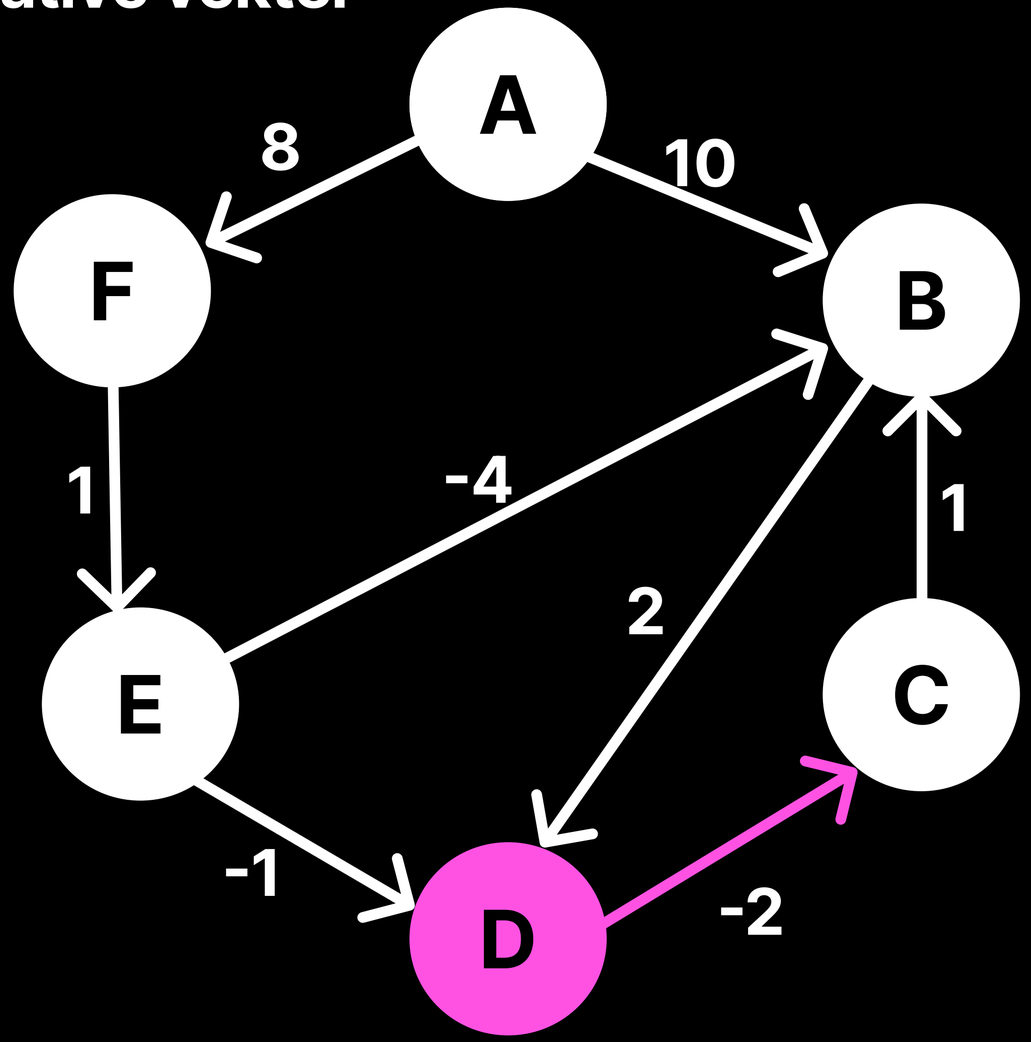
Negative vekter



queue	distance
A	0
B	5
C	10>6
D	8
E	9
F	8

Bellman-Ford

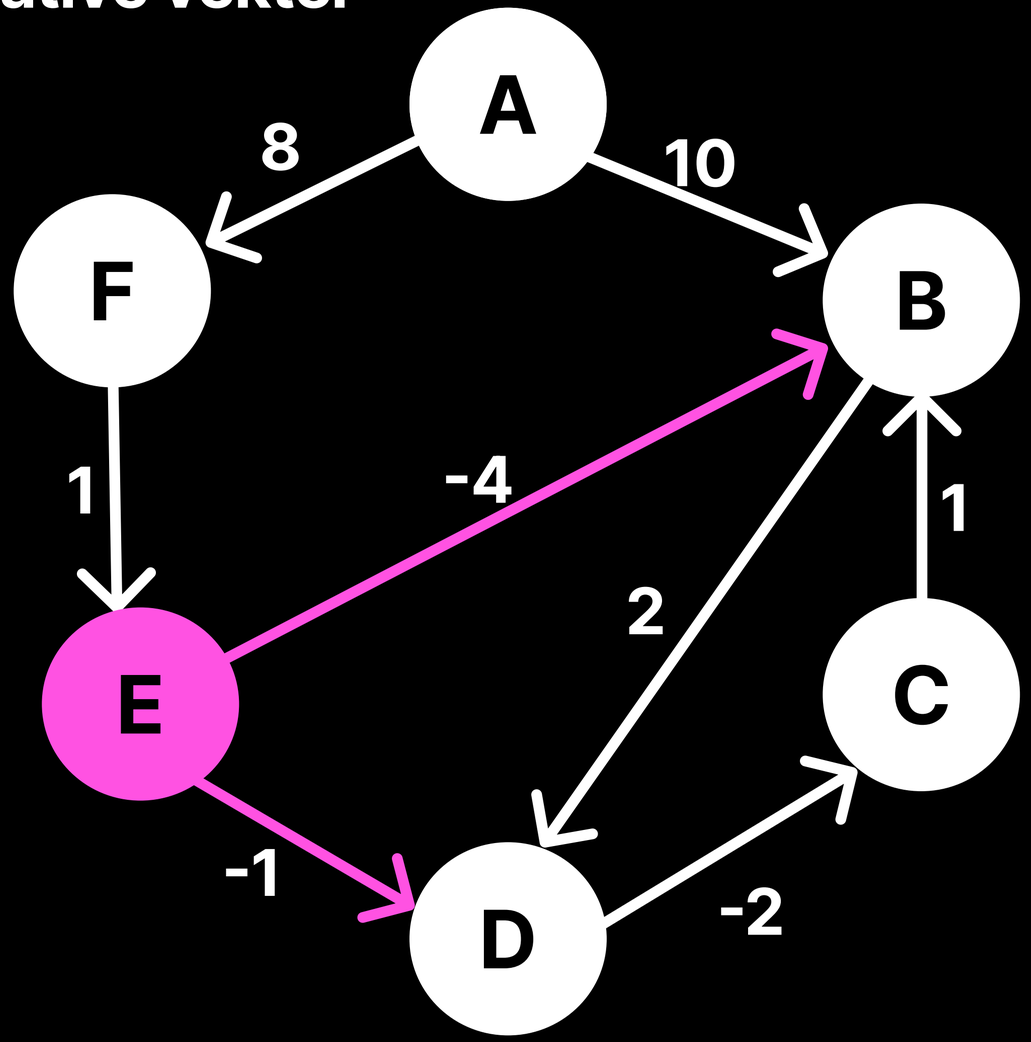
Negative vekter



queue	distance
A	0
B	5
C	6
D	8
E	9
F	8

Bellman-Ford

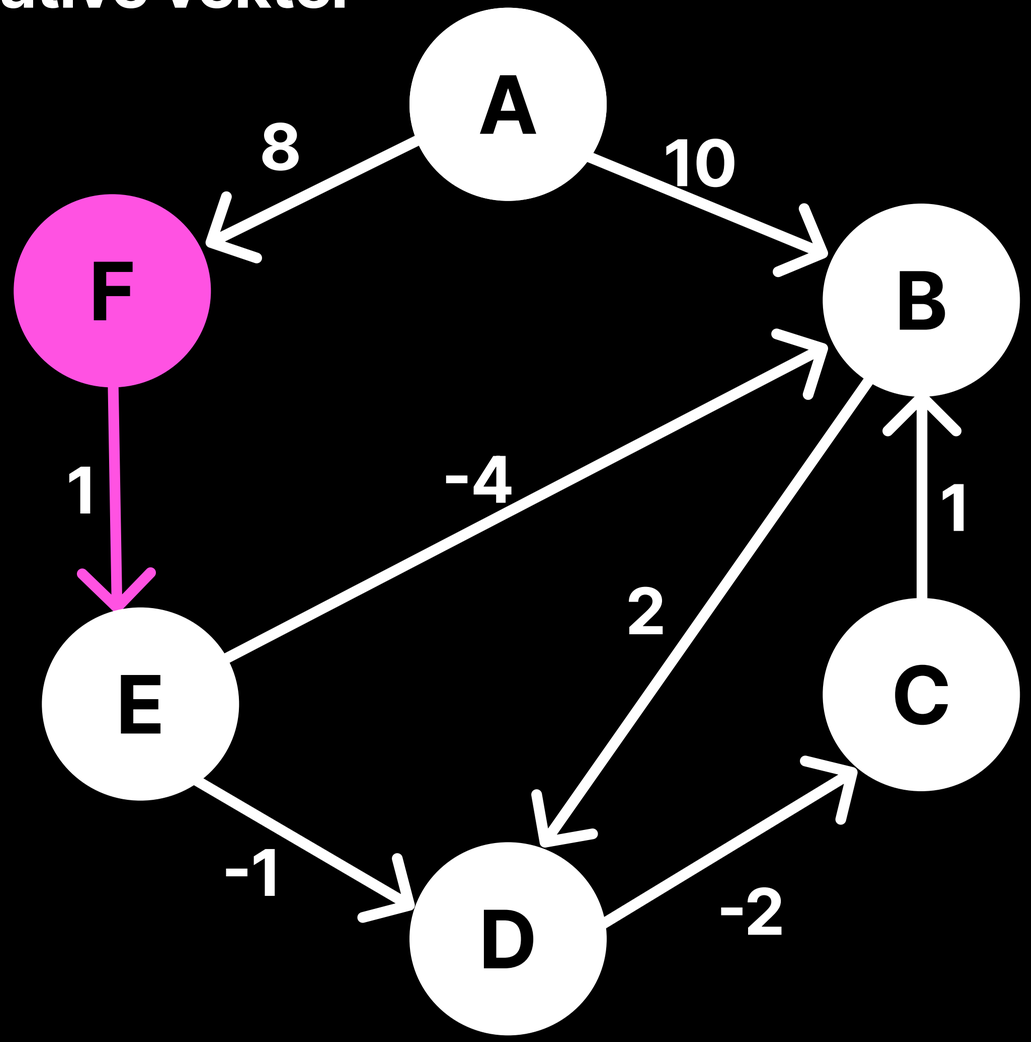
Negative vekter



queue	distance
A	0
B	5=5
C	6
D	8=8
E	9
F	8

Bellman-Ford

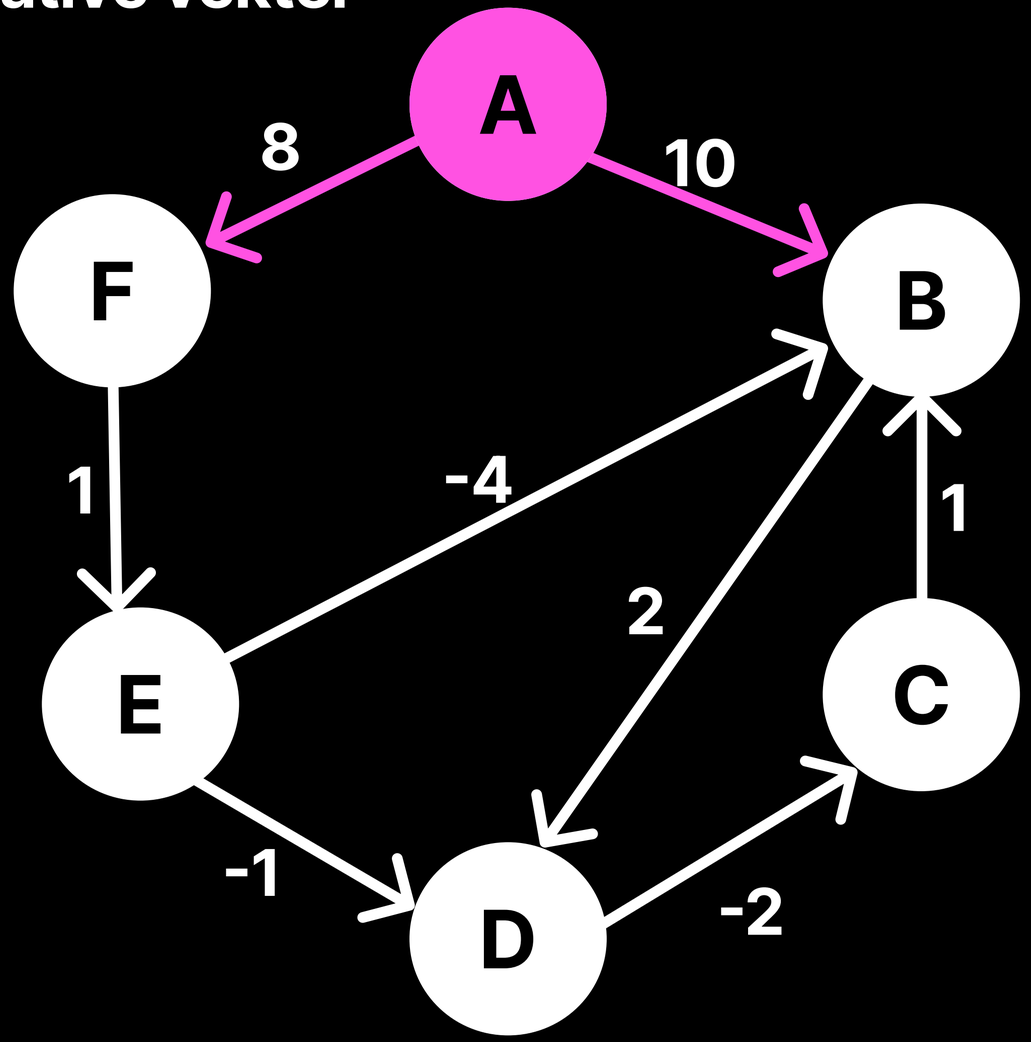
Negative vekter



queue	distance
A	0
B	5
C	6
D	8
E	9=9
F	8

Bellman-Ford

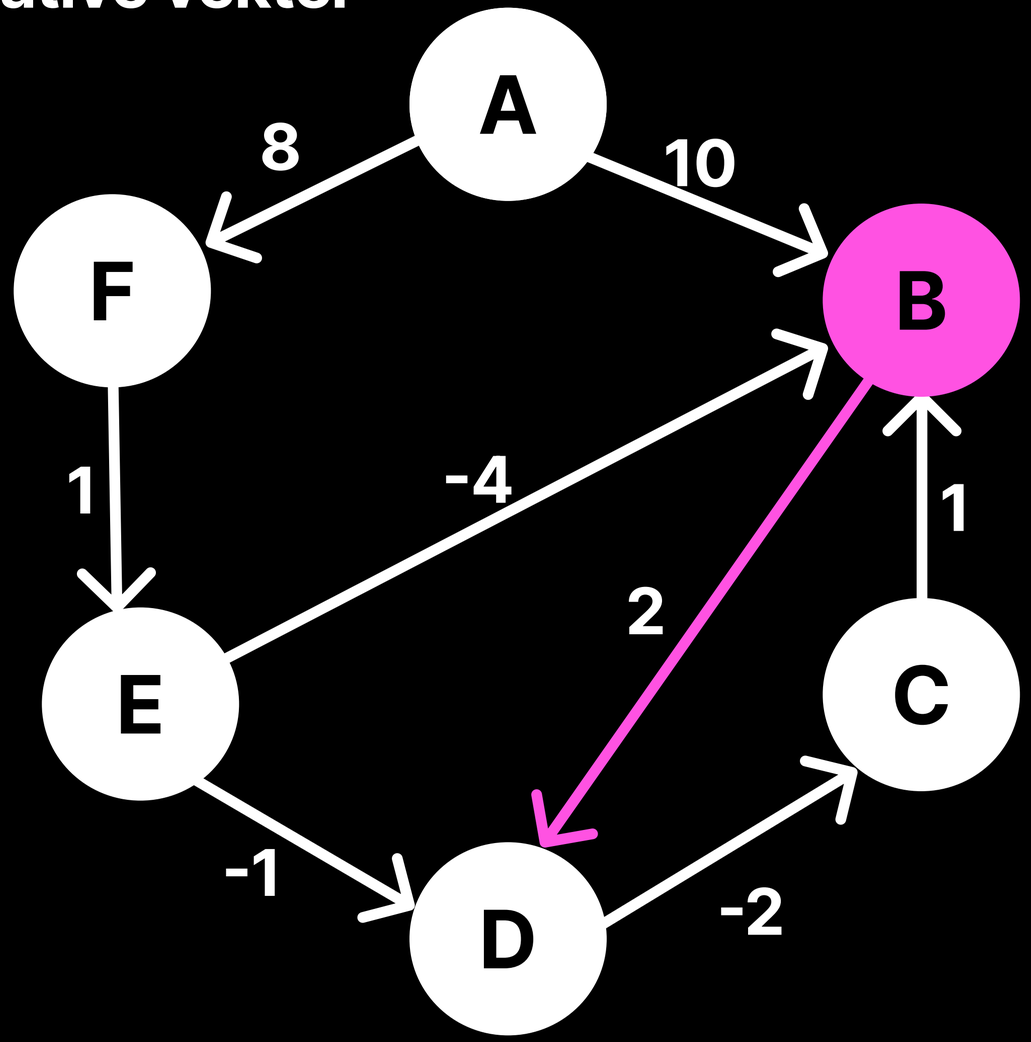
Negative vekter



queue	distance
A	0
B	5<10
C	6
D	8
E	9
F	8=8

Bellman-Ford

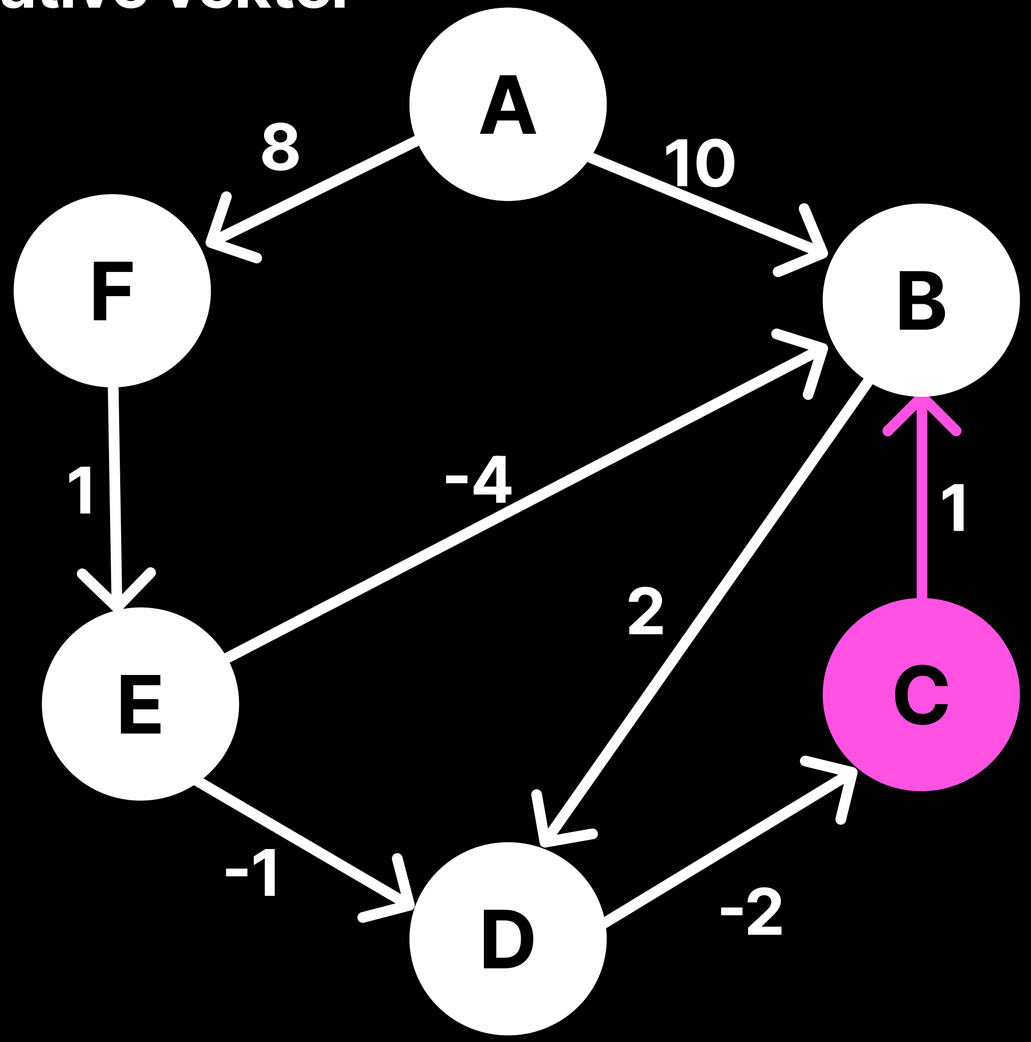
Negative vekter



queue	distance
A	0
B	5
C	6
D	8<12
E	9
F	8

Bellman-Ford

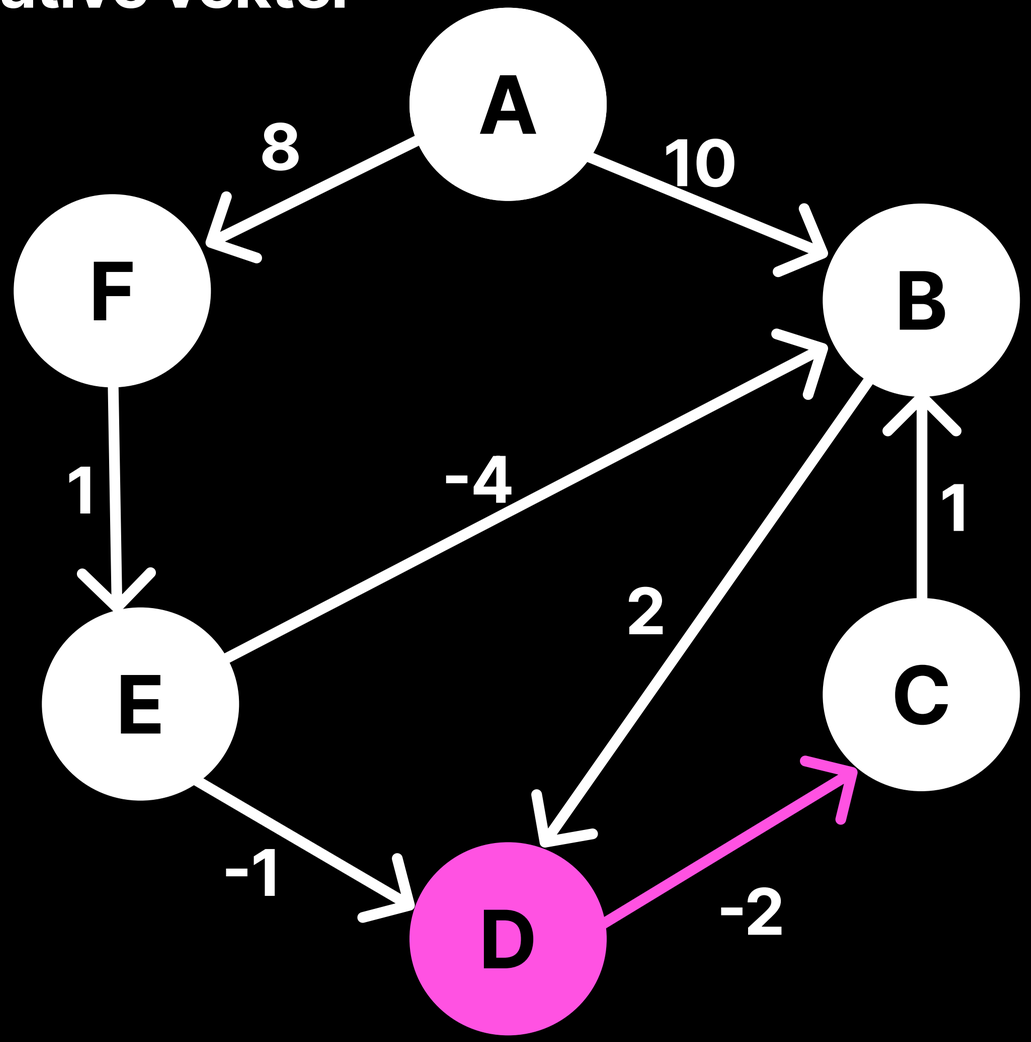
Negative vekter



queue	distance
A	0
B	5<7
C	6
D	8
E	9
F	8

Bellman-Ford

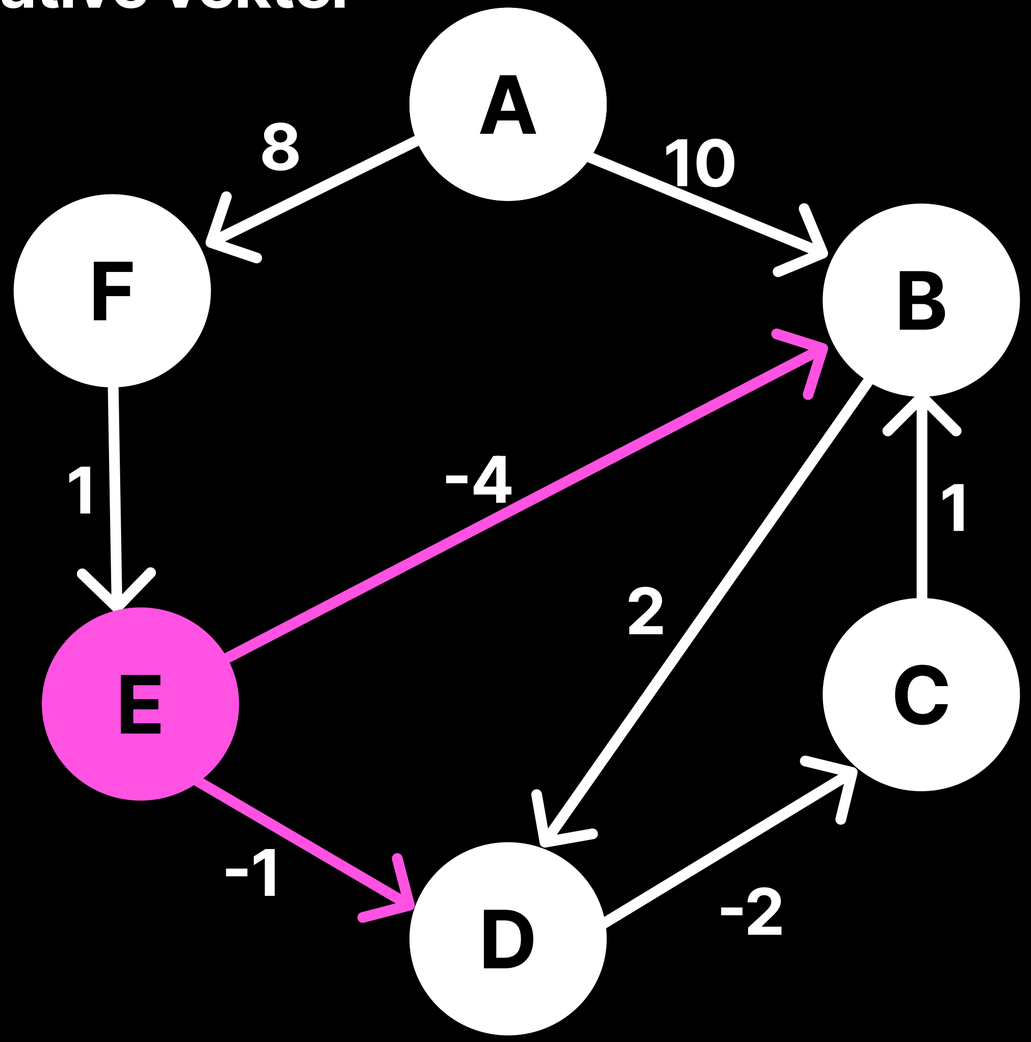
Negative vekter



queue	distance
A	0
B	5
C	6=6
D	8
E	9
F	8

Bellman-Ford

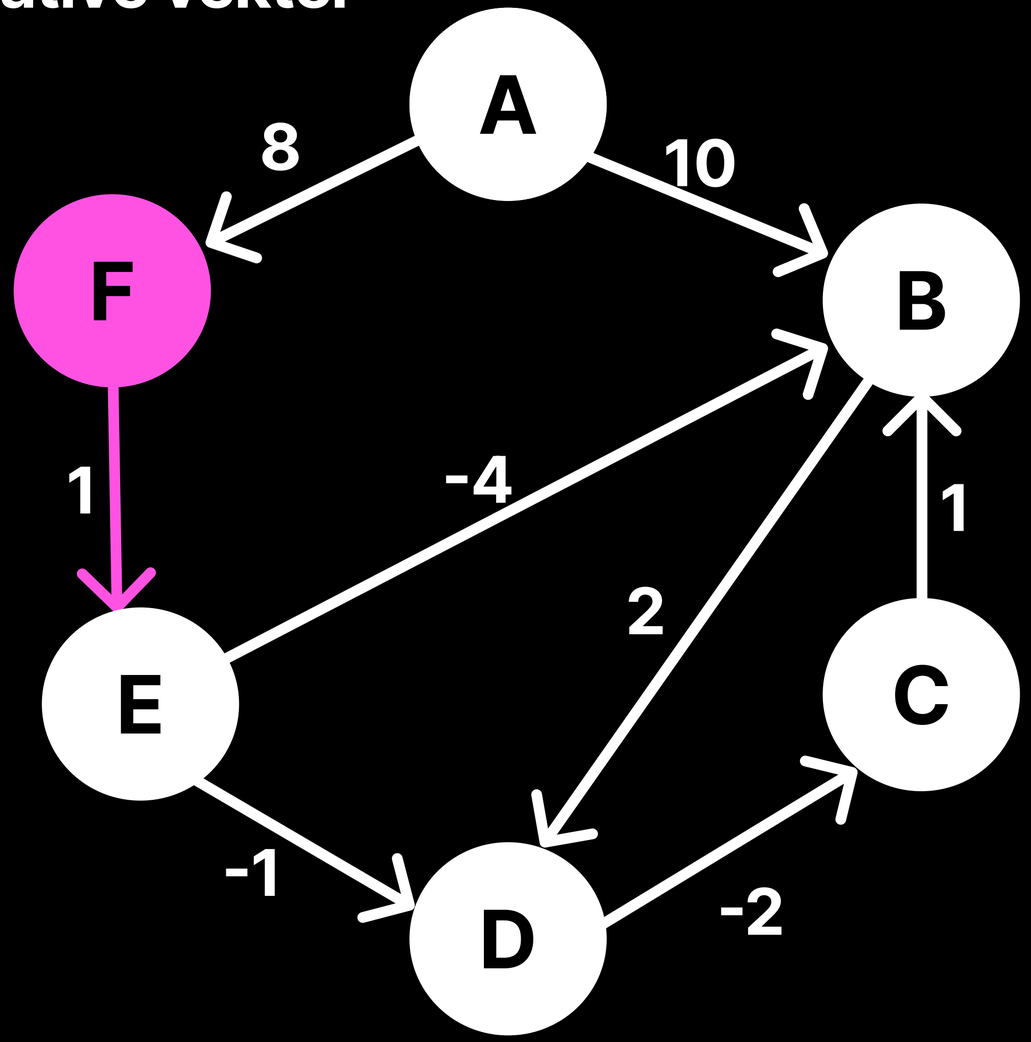
Negative vekter



queue	distance
A	0
B	5=5
C	6
D	8=8
E	9
F	8

Bellman-Ford

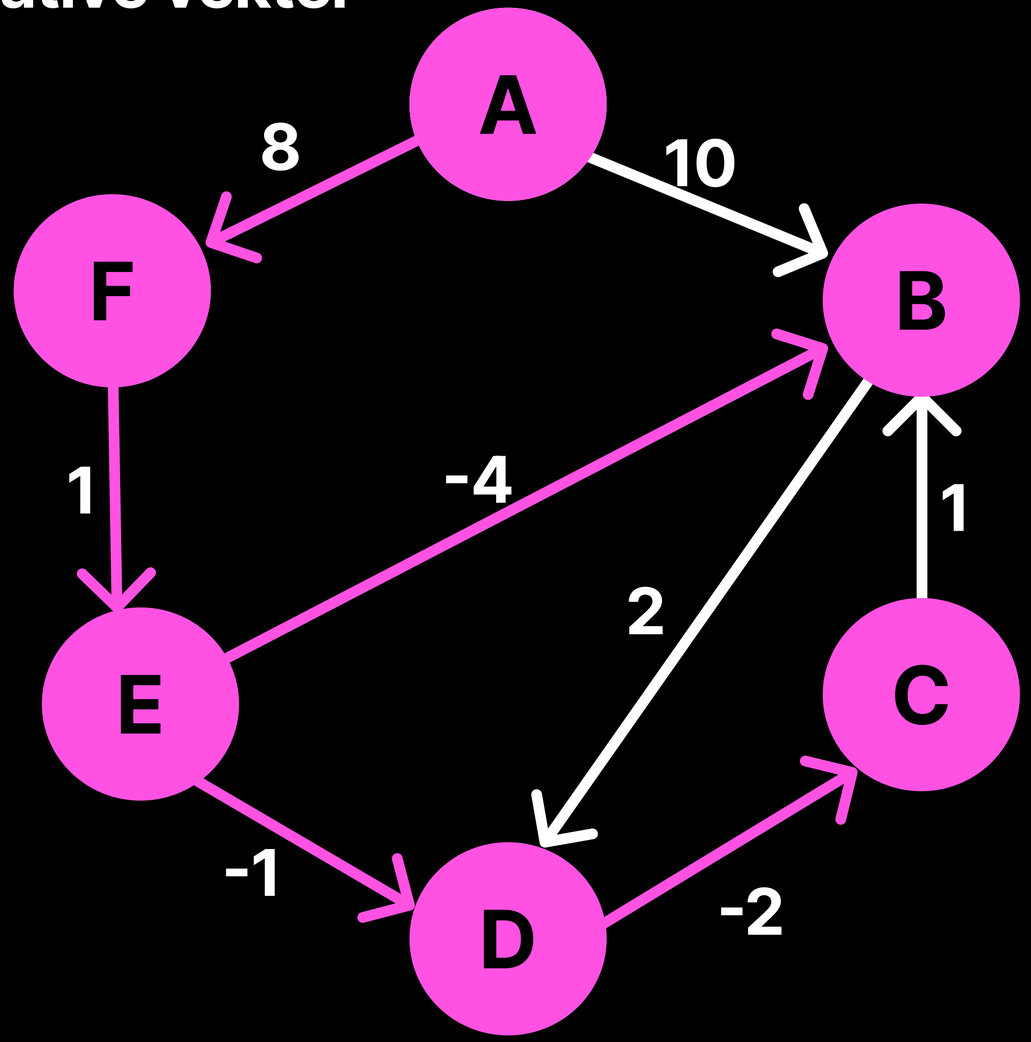
Negative vekter



queue	distance
A	0
B	5
C	6
D	8
E	9=9
F	8

Bellman-Ford

Negative vekter



queue	distance
A	0
B	5
C	6
D	8
E	9
F	8

Popquiz - Dijkstra vs. Bellman-Ford

	Dijkstra	Bellman-Ford
Funker ikke med negative vekter		
Funker med negative vekter		
Funker på rettede grafer		
Funker på urettede grafer		

Popquiz - Dijkstra vs. Bellman-Ford

	Dijkstra	Bellman-Ford
Funker ikke med negative vekter		
Funker med negative vekter		
Funker på rettede grafer		
Funker på urettede grafer		

Popquiz - Dijkstra vs. Bellman-Ford

	Dijkstra	Bellman-Ford
Funker ikke med negative vekter		
Funker med negative vekter		
Funker på rettede grafer		
Funker på urettede grafer		

Popquiz - Dijkstra vs. Bellman-Ford

	Dijkstra	Bellman-Ford
Funker ikke med negative vekter	✓	
Funker med negative vekter		✓
Funker på rettede grafer	✓	✓
Funker på urettede grafer	✓	✓

Spenntrær

Trær: En sammenhengende, urettet og asyklisk graf er et **tre**.

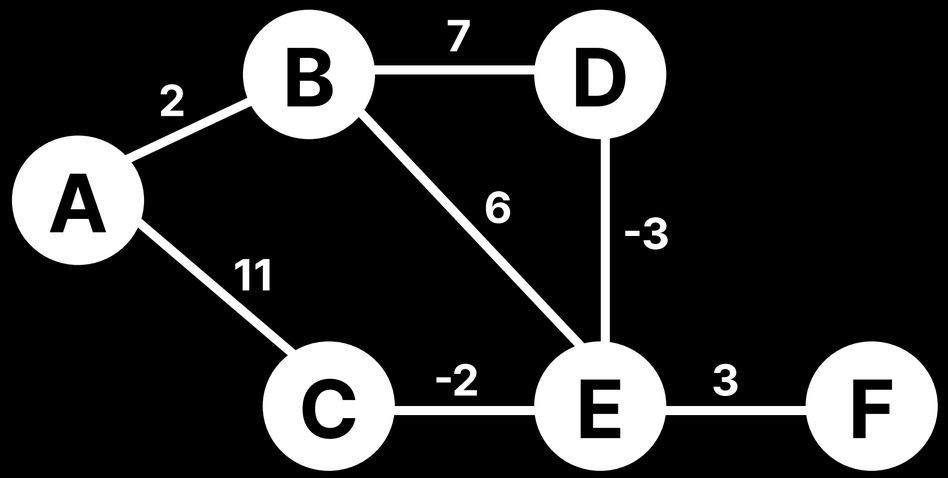
- nøyaktig $|V| - 1$ kanter

Spenntrær

Trær: En sammenhengende, urettet og asyklisk graf er et **tre**.

- nøyaktig |V| − 1 kanter

Hvis dette er grafen

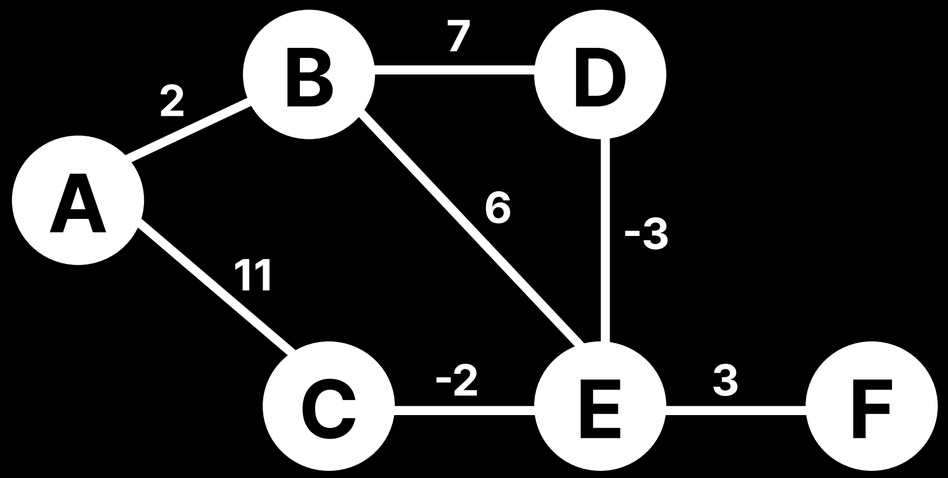


Spenntrær

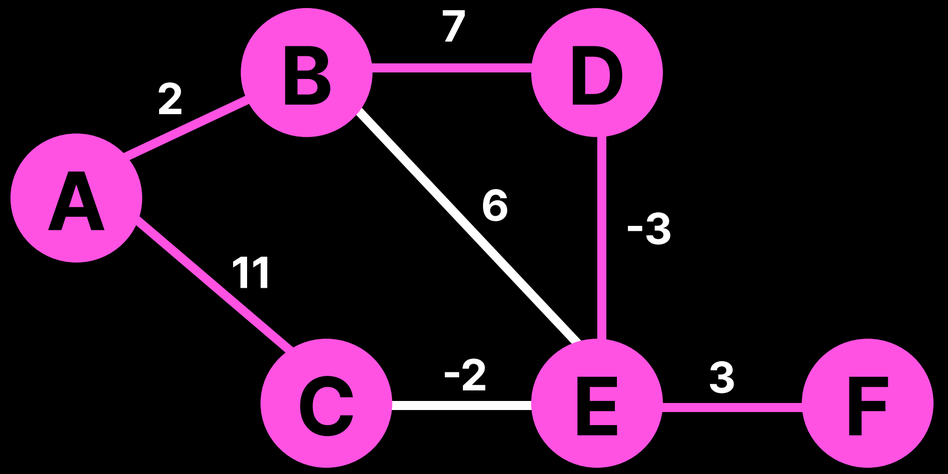
Trær: En sammenhengende, urettet og asyklisk graf er et **tre**.

- nøyaktig |V| − 1 kanter

Hvis dette er grafen

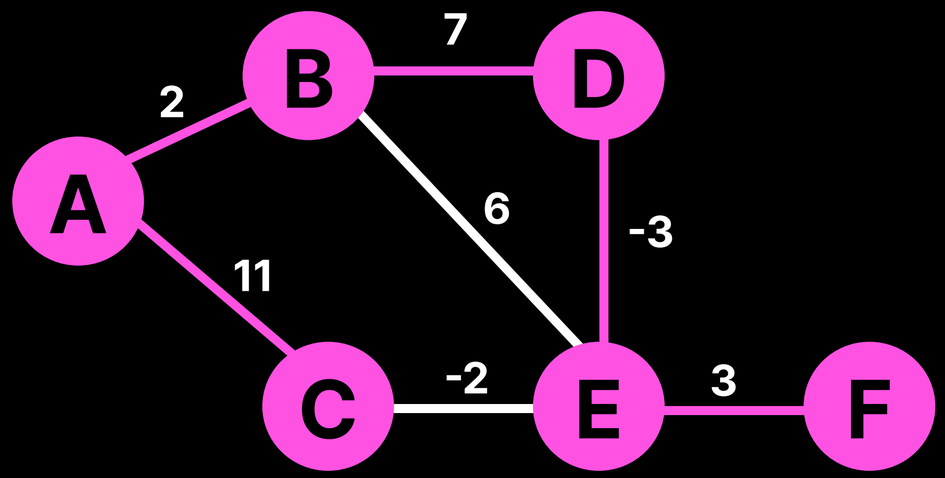


Så er dette et spenntre i grafen



Spenntrær

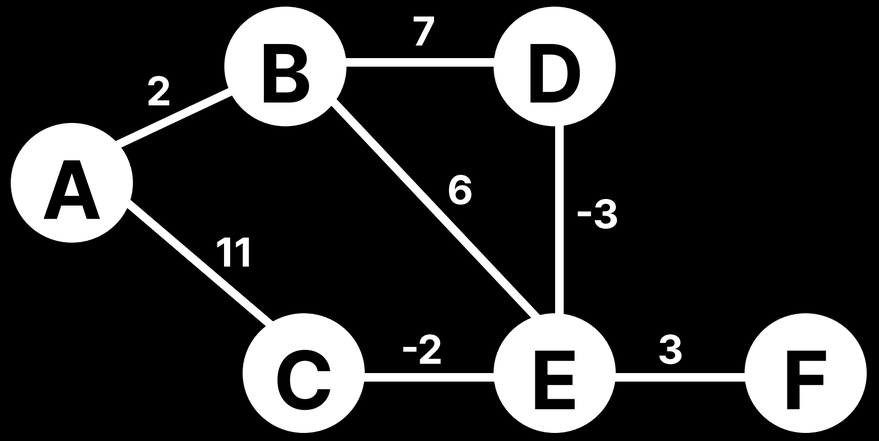
MEN det er IKKE et minimalt spenntre!



Minimale spenntrær

I et minimalt spenntre leter vi etter kantene med minst vekt, fram til vi har knyttet sammen alle nodene i grafen (uten sykler! Det skal være et tre)

VIKTIG: vi leter ikke lenger etter korteste sti!

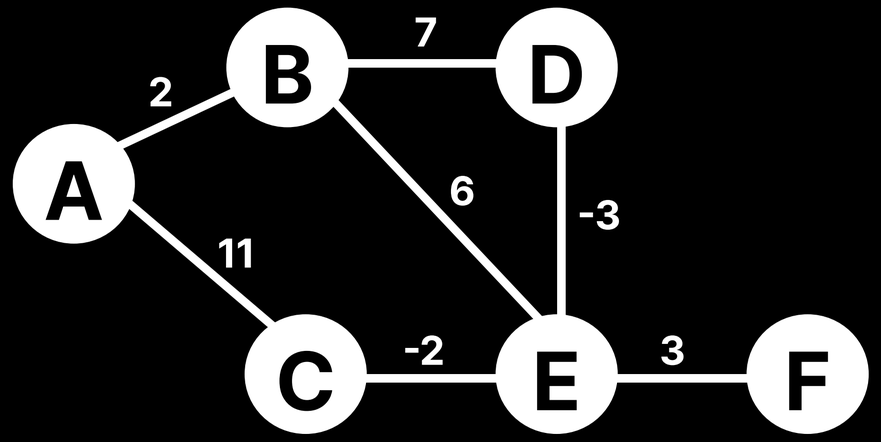


Minimale spenntrær

I et minimalt spenntre leter vi etter kantene med minst vekt, fram til vi har knyttet sammen alle nodene i grafen (uten sykler! Det skal være et tre)

VIKTIG: vi leter ikke lenger etter korteste sti!

Hva er den laveste kanten her?

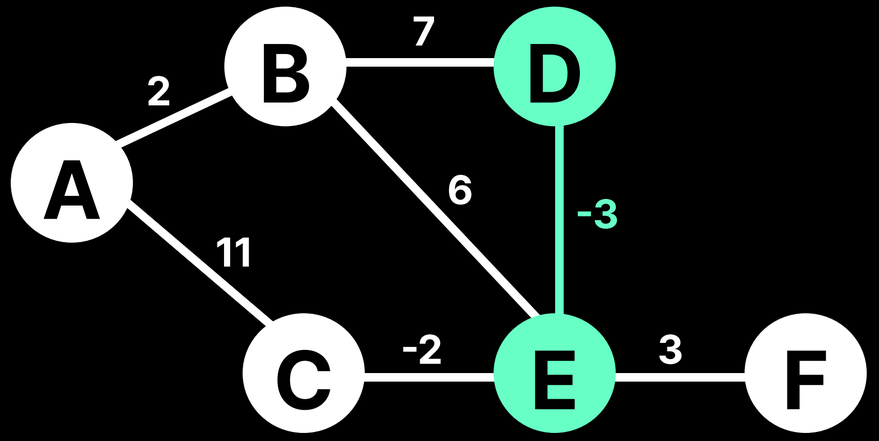


Minimale spenntrær

I et minimalt spenntre leter vi etter kantene med minst vekt, fram til vi har knyttet sammen alle nodene i grafen (uten sykler! Det skal være et tre)

VIKTIG: vi leter ikke lenger etter korteste sti!

Hva er den laveste kanten nå?

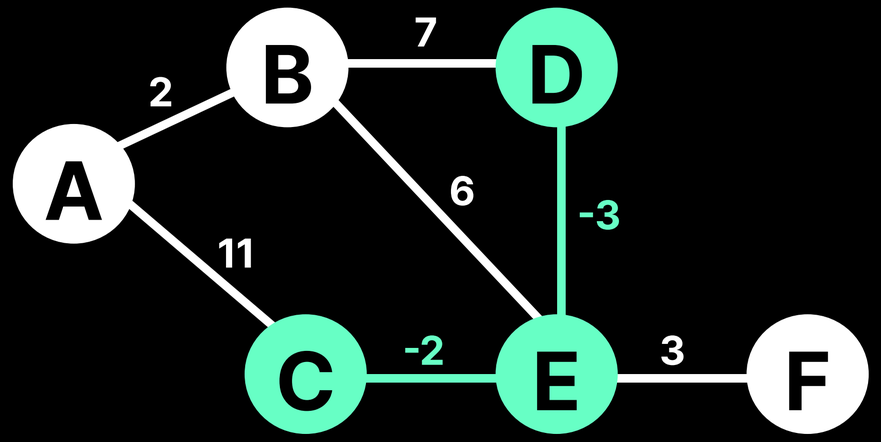


Minimale spenntrær

I et minimalt spenntre leter vi etter kantene med minst vekt, fram til vi har knyttet sammen alle nodene i grafen (uten sykler! Det skal være et tre)

VIKTIG: vi leter ikke lenger etter korteste sti!

Hva er den laveste kanten nå?

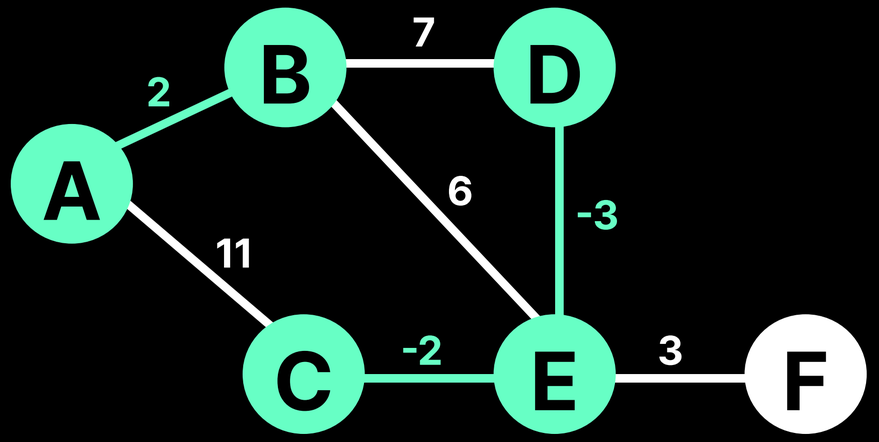


Minimale spenntrær

I et minimalt spenntre leter vi etter kantene med minst vekt, fram til vi har knyttet sammen alle nodene i grafen (uten sykler! Det skal være et tre)

VIKTIG: vi leter ikke lenger etter korteste sti!

Hva er den laveste kanten nå?

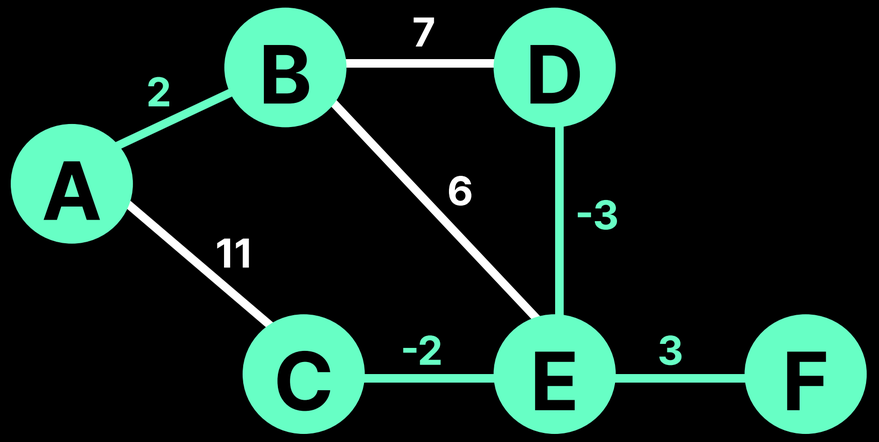


Minimale spenntrær

I et minimalt spenntre leter vi etter kantene med minst vekt, fram til vi har knyttet sammen alle nodene i grafen (uten sykler! Det skal være et tre)

VIKTIG: vi leter ikke lenger etter korteste sti!

Hva er den laveste kanten nå?

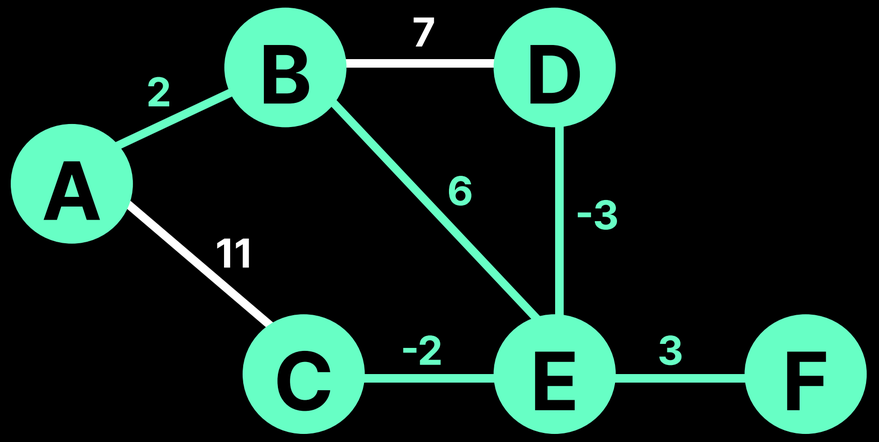


Minimale spenntrær

I et minimalt spenntre leter vi etter kantene med minst vekt, fram til vi har knyttet sammen alle nodene i grafen (uten sykler! Det skal være et tre)

VIKTIG: vi leter ikke lenger etter korteste sti!

Hva er den laveste kanten nå?

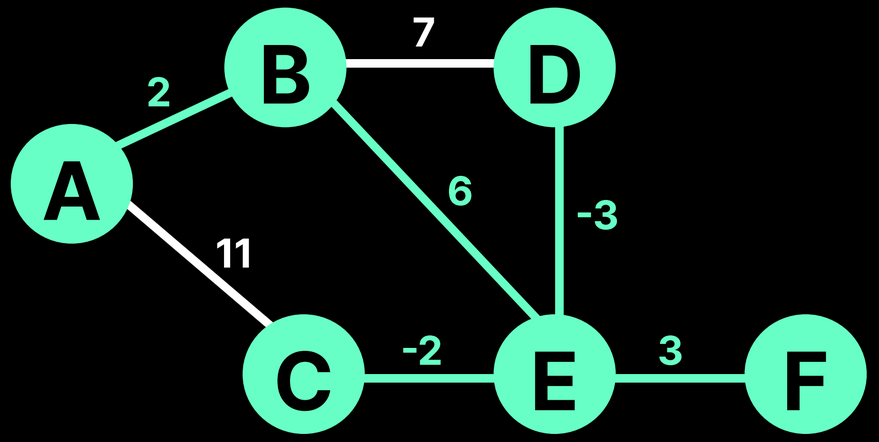


Minimale spenntrær

I et minimalt spenntre leter vi etter kantene med minst vekt, fram til vi har knyttet sammen alle nodene i grafen (uten sykler! Det skal være et tre)

VIKTIG: vi leter ikke lenger etter korteste sti!

Nå har vi et minimalt spenntre!



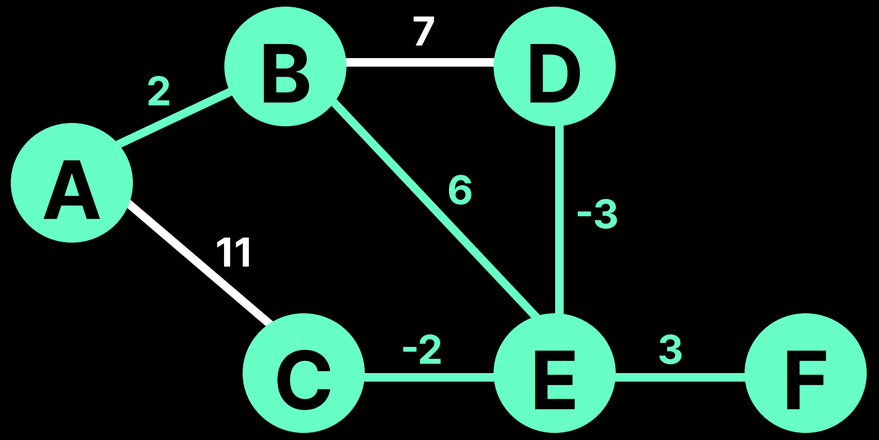
Minimale spenntrær

I et minimalt spenntre leter vi etter kantene med minst vekt, fram til vi har knyttet sammen alle nodene i grafen (uten sykler! Det skal være et tre)

VIKTIG: vi leter ikke lenger etter korteste sti!

Nå har vi et minimalt spenntre!

- **Vi har valgt kantene med minst vekt**



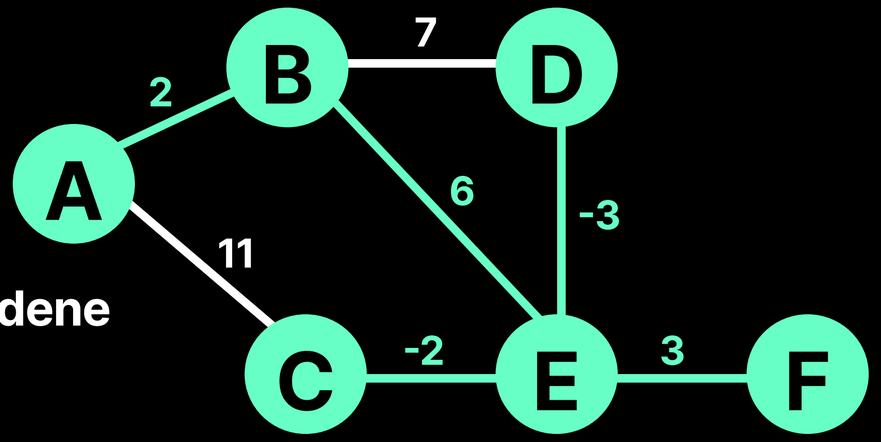
Minimale spenntrær

I et minimalt spenntre leter vi etter kantene med minst vekt, fram til vi har knyttet sammen alle nodene i grafen (uten sykler! Det skal være et tre)

VIKTIG: vi leter ikke lenger etter korteste sti!

Nå har vi et minimalt spenntre!

- **Vi har valgt kantene med minst vekt**
- **Fram til vi fikk knyttet sammen alle nodene**



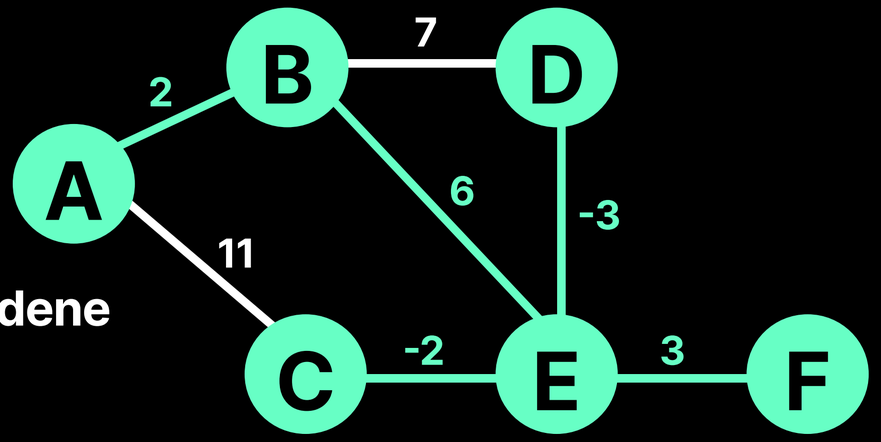
Minimale spenntrær

I et minimalt spenntre leter vi etter kantene med minst vekt, fram til vi har knyttet sammen alle nodene i grafen (uten sykler! Det skal være et tre)

VIKTIG: vi leter ikke lenger etter korteste sti!

Nå har vi et minimalt spenntre!

- **Vi har valgt kantene med minst vekt**
- **Fram til vi fikk knyttet sammen alle nodene**
- **UTEN sykler!**



Minimale spenntrær

Algoritmer for å finne minimale spenntrær

Prims

- Best for **tette grafer**
- Funker **bare** på **sammenhengende grafer**

Kruskals

- Best for **tynne grafer**
- Funker også på **ikke-sammenhengende grafer**. Resultatet blir da en spennskog

Borůvkas

- Effektiv for både **tette og tynne grafer**
- Funker også på **ikke-sammenhengende grafer.**
- Effektiv for parallellprogrammering

Popquiz - Prims vs. Kruskals vs. Borůvkas

	Prims	Kruskals	Borůvkas
Brukes på tette grafer			
Brukes på tynne grafer			
Funker på sammenhengende grafer			
Funker på ikke-sammenhengende grafer			
Effektiv til parallellprogrammering			

Popquiz - Prims vs. Kruskals vs. Borůvkas

	Prims	Kruskals	Borůvkas
Brukes på tette grafer			
Brukes på tynne grafer			
Funker på sammenhengende grafer			
Funker på ikke-sammenhengende grafer			
Effektiv til parallellprogrammering			

Popquiz - Prims vs. Kruskals vs. Borůvkas

	Prims	Kruskals	Borůvkas
Brukes på tette grafer	✓		✓
Brukes på tynne grafer		✓	✓
Funker på sammenhengende grafer			
Funker på ikke-sammenhengende grafer			
Effektiv til parallellprogrammering			

Popquiz - Prims vs. Kruskals vs. Borůvkas

	Prims	Kruskals	Borůvkas
Brukes på tette grafer	✓		✓
Brukes på tynne grafer		✓	✓
Funker på sammenhengende grafer	✓	✓	✓
Funker på ikke-sammenhengende grafer			
Effektiv til parallellprogrammering			

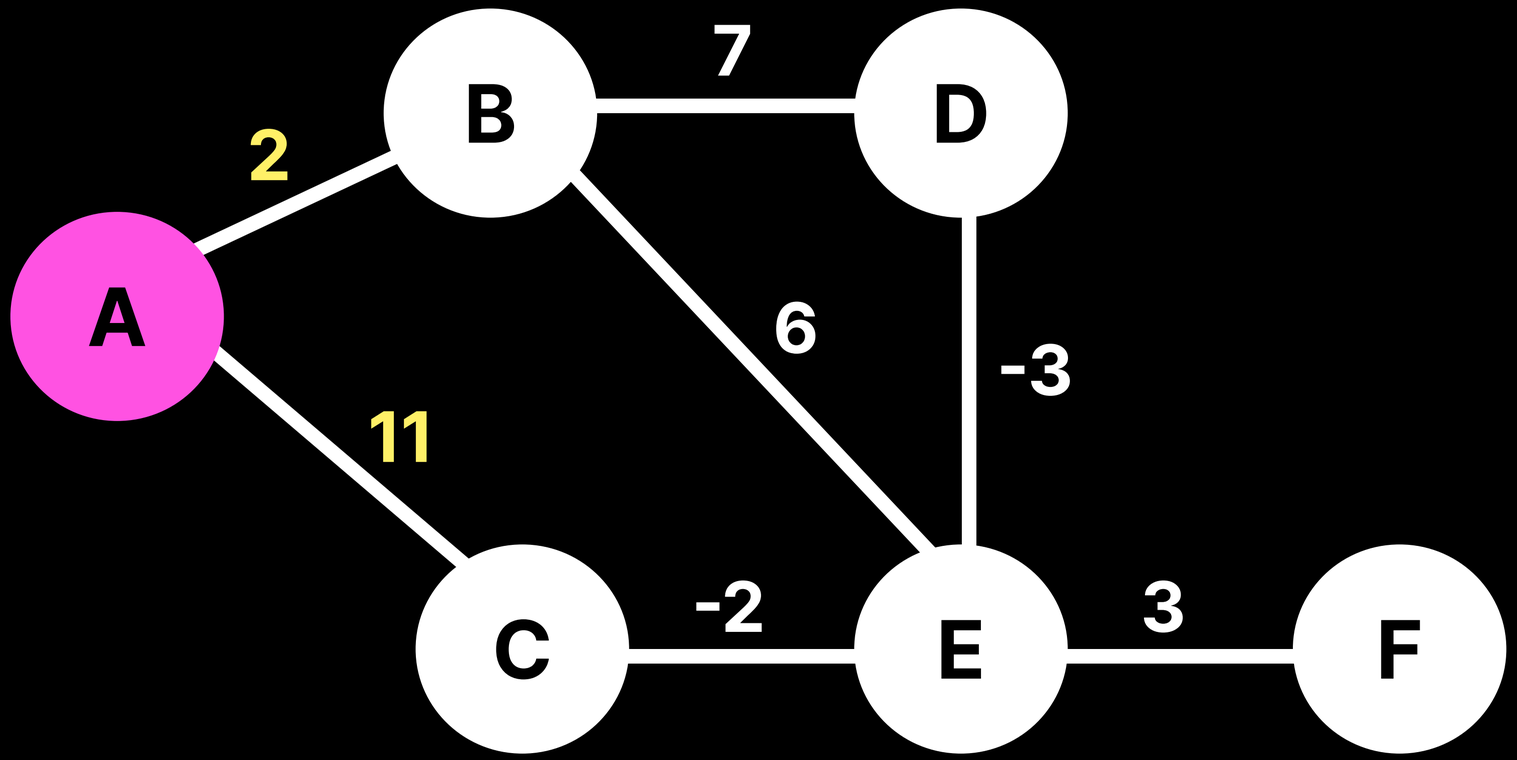
Popquiz - Prims vs. Kruskals vs. Borůvkas

	Prims	Kruskals	Borůvkas
Brukes på tette grafer	✓		✓
Brukes på tynne grafer		✓	✓
Funker på sammenhengende grafer	✓	✓	✓
Funker på ikke-sammenhengende grafer		✓	✓
Effektiv til parallellprogrammering			

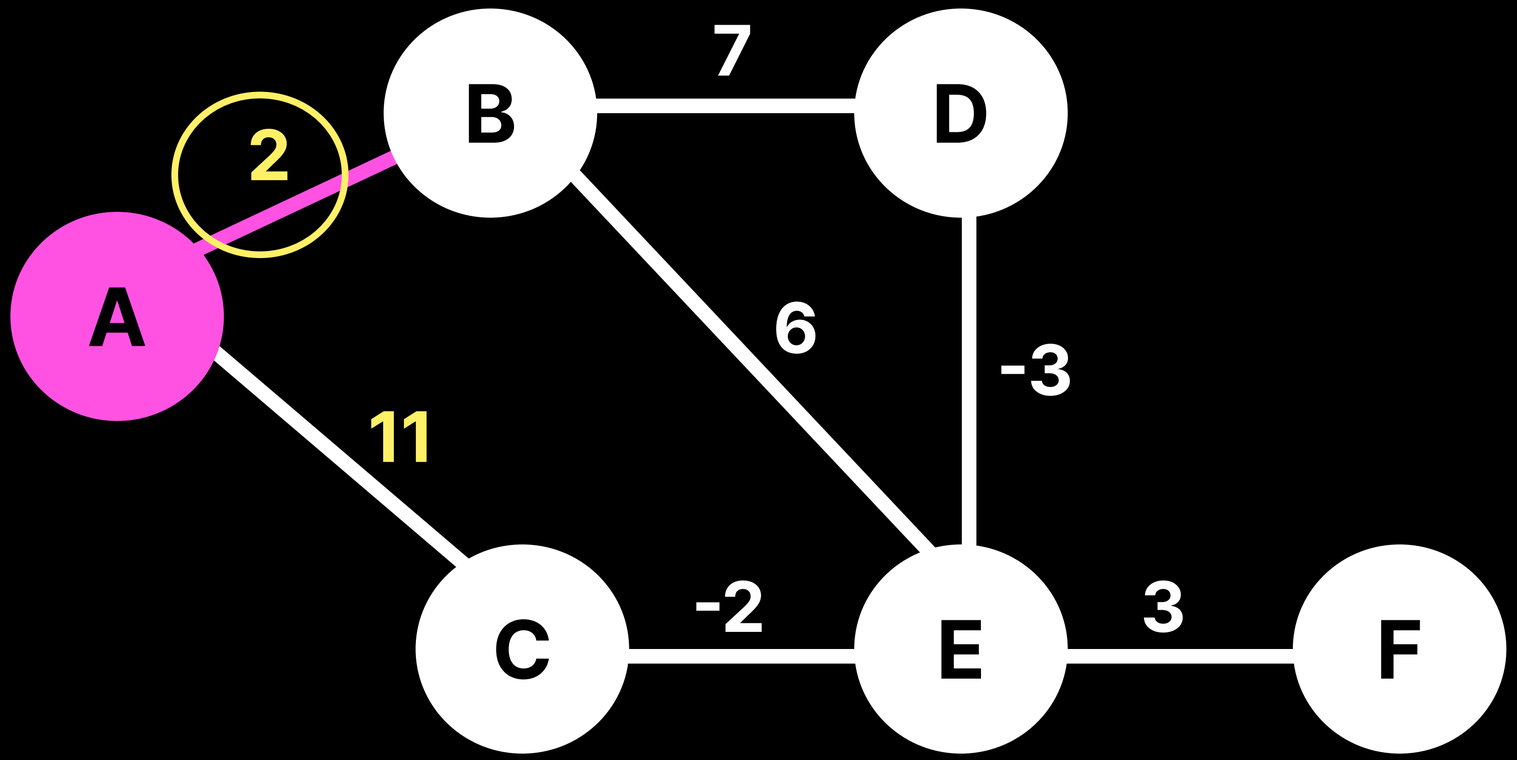
Popquiz - Prims vs. Kruskals vs. Borůvkas

	Prims	Kruskals	Borůvkas
Brukes på tette grafer	✓		✓
Brukes på tynne grafer		✓	✓
Funker på sammenhengende grafer	✓	✓	✓
Funker på ikke-sammenhengende grafer		✓	✓
Effektiv til parallellprogrammering			✓

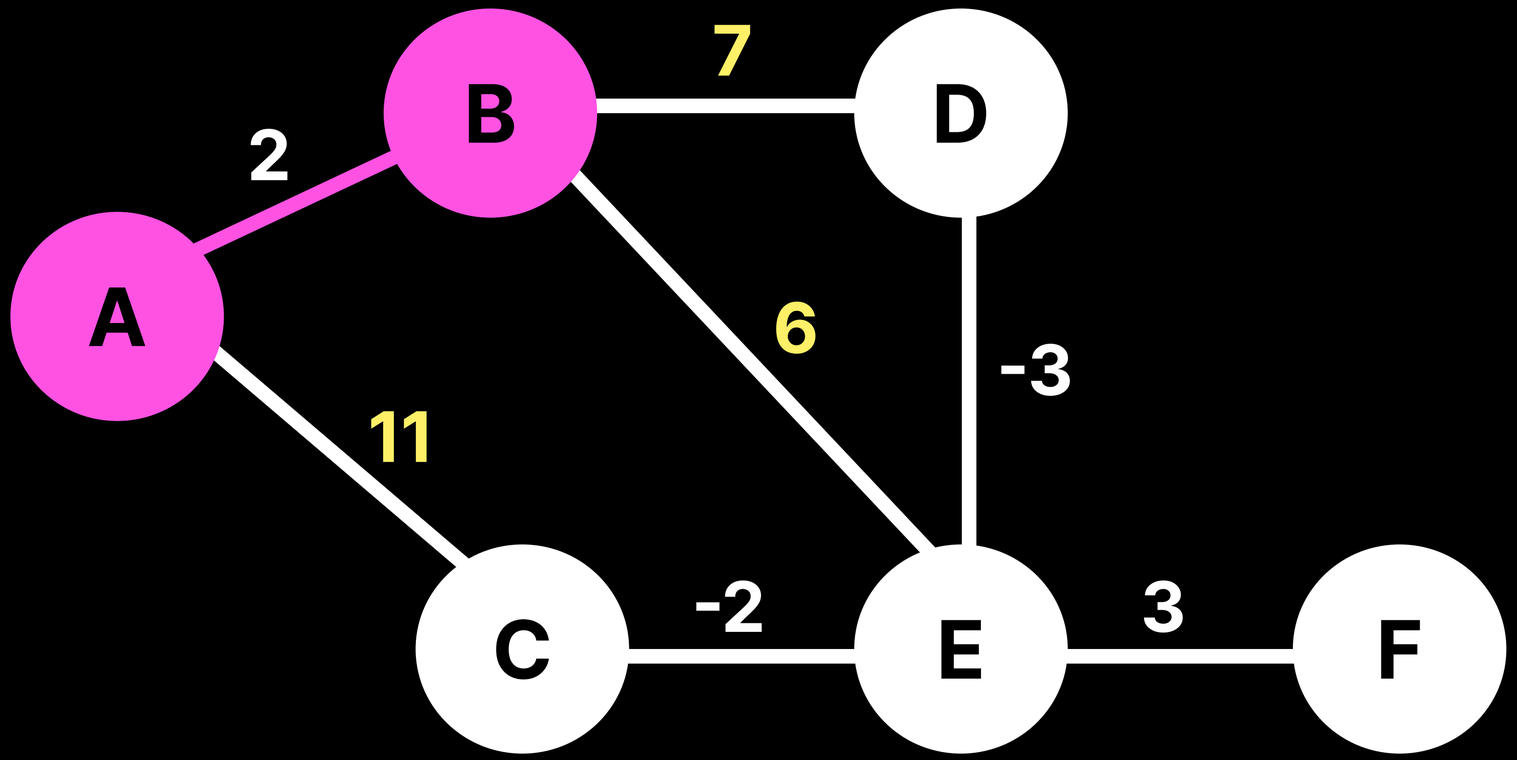
Prims



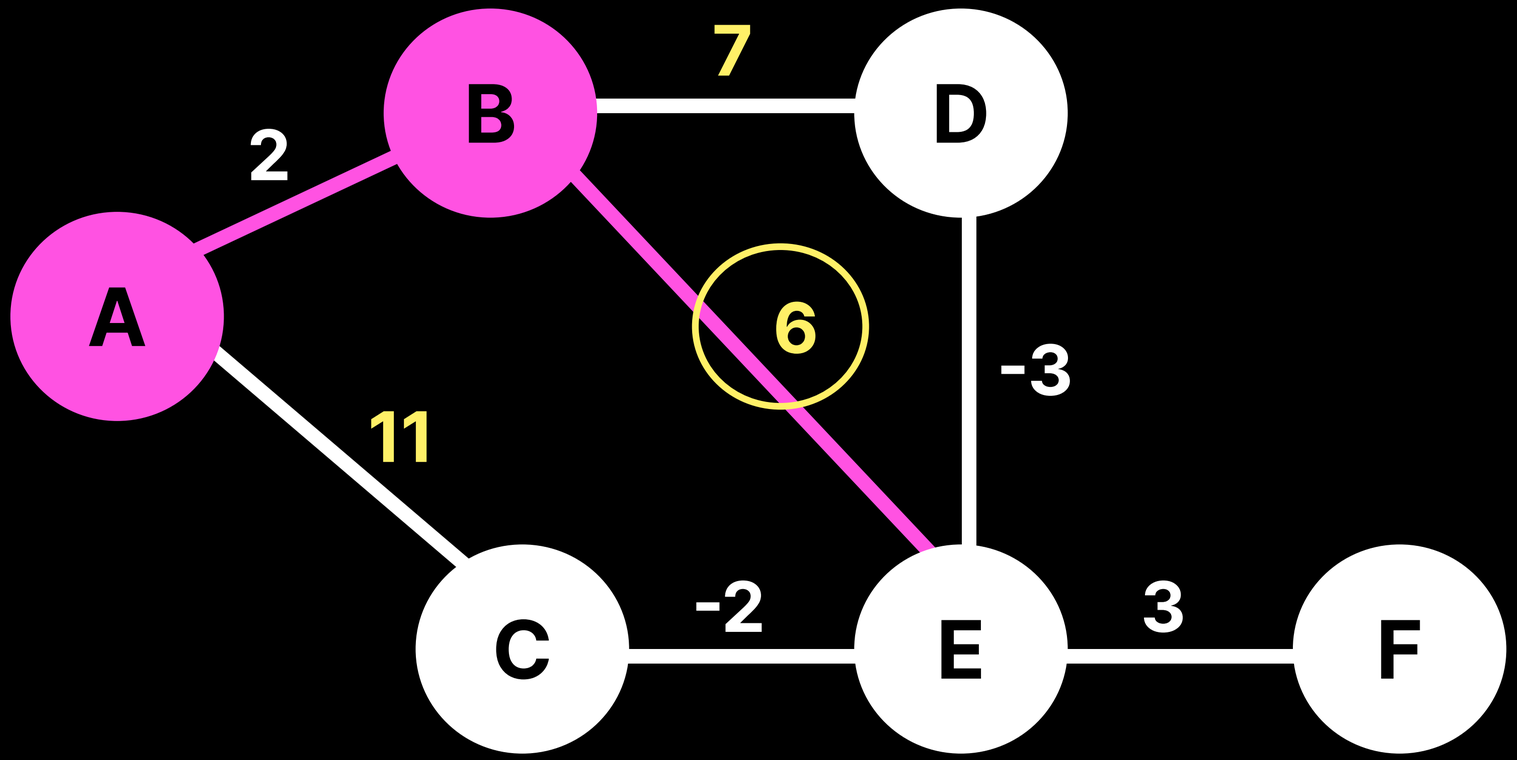
Prims



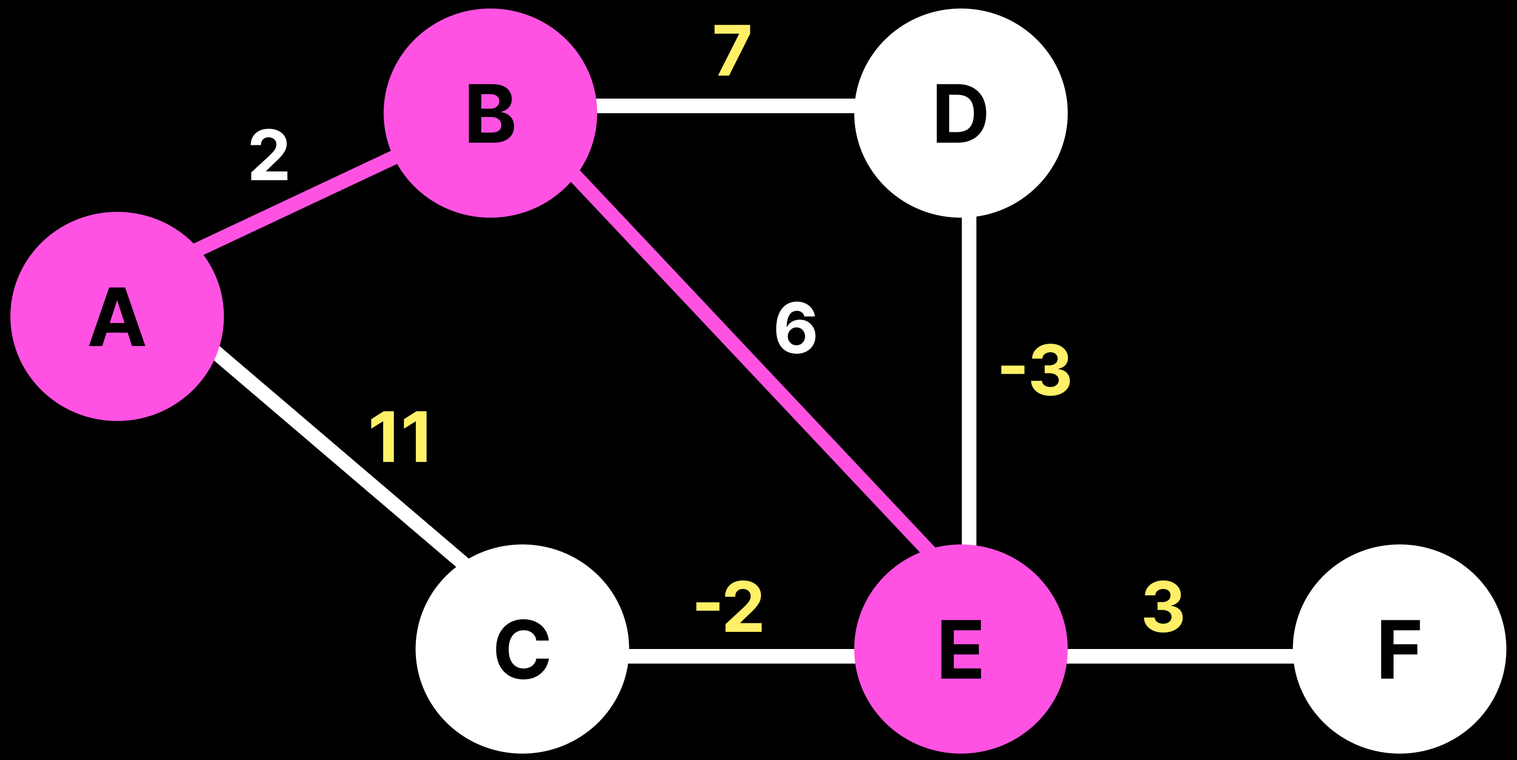
Prims



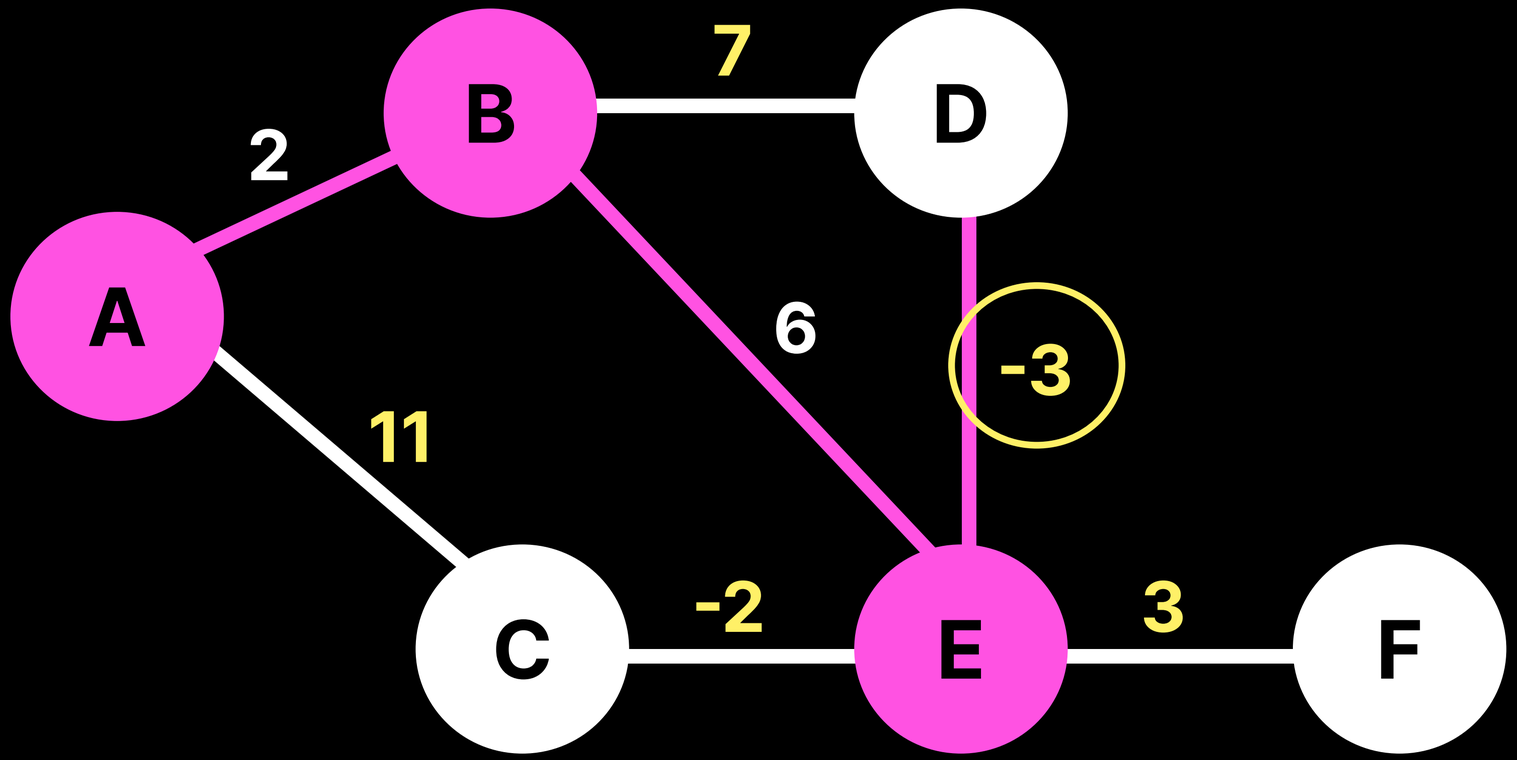
Prims



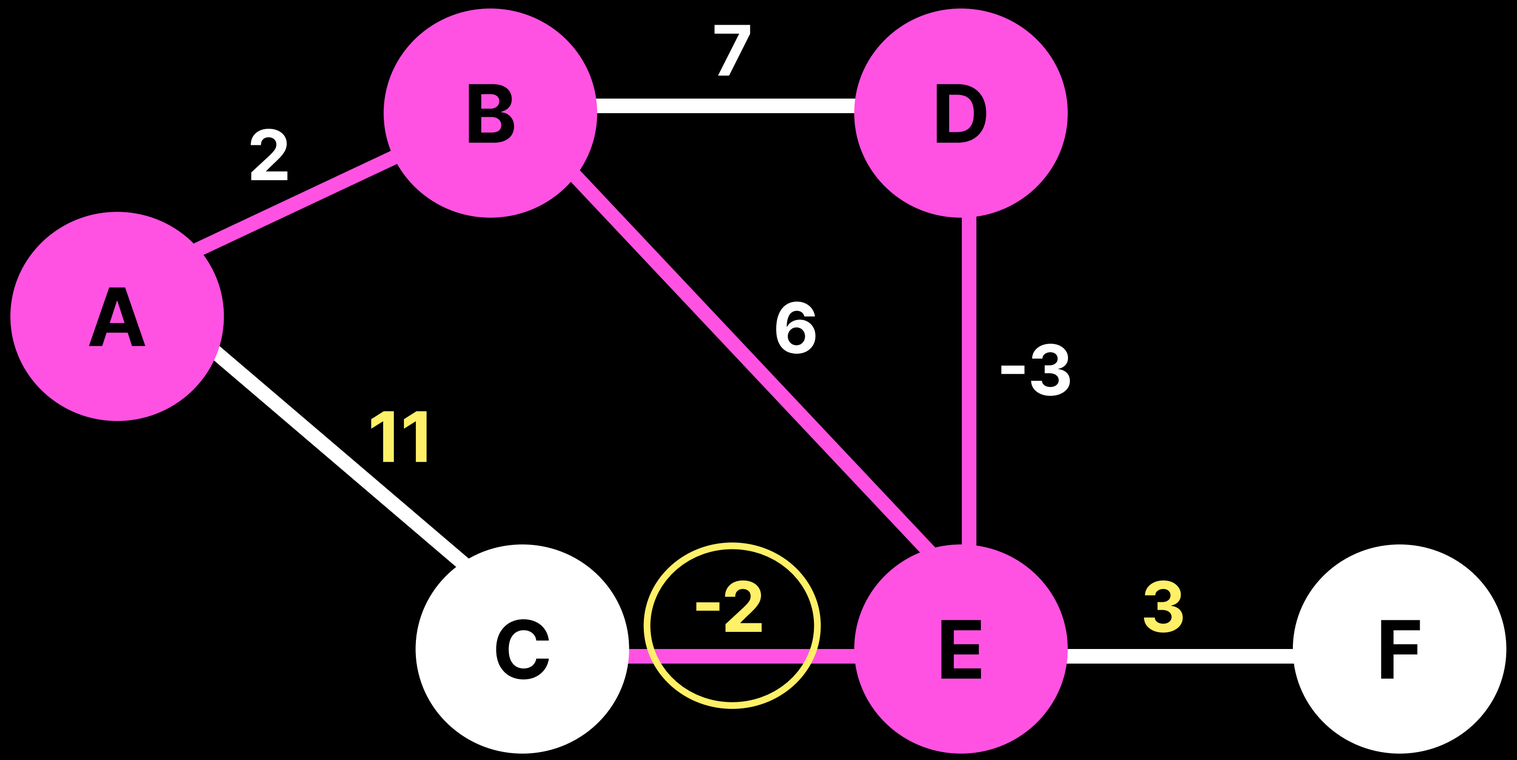
Prims



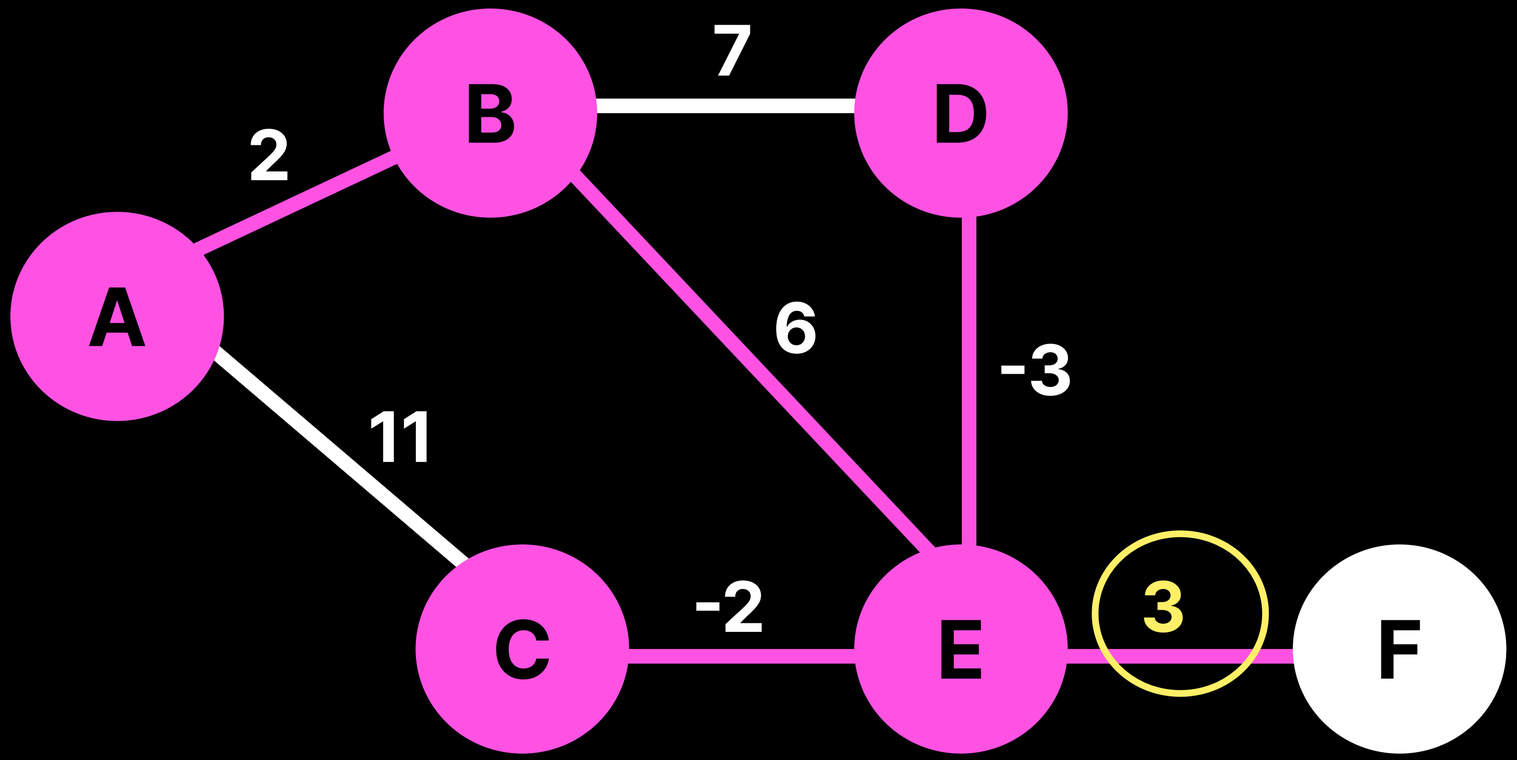
Prims



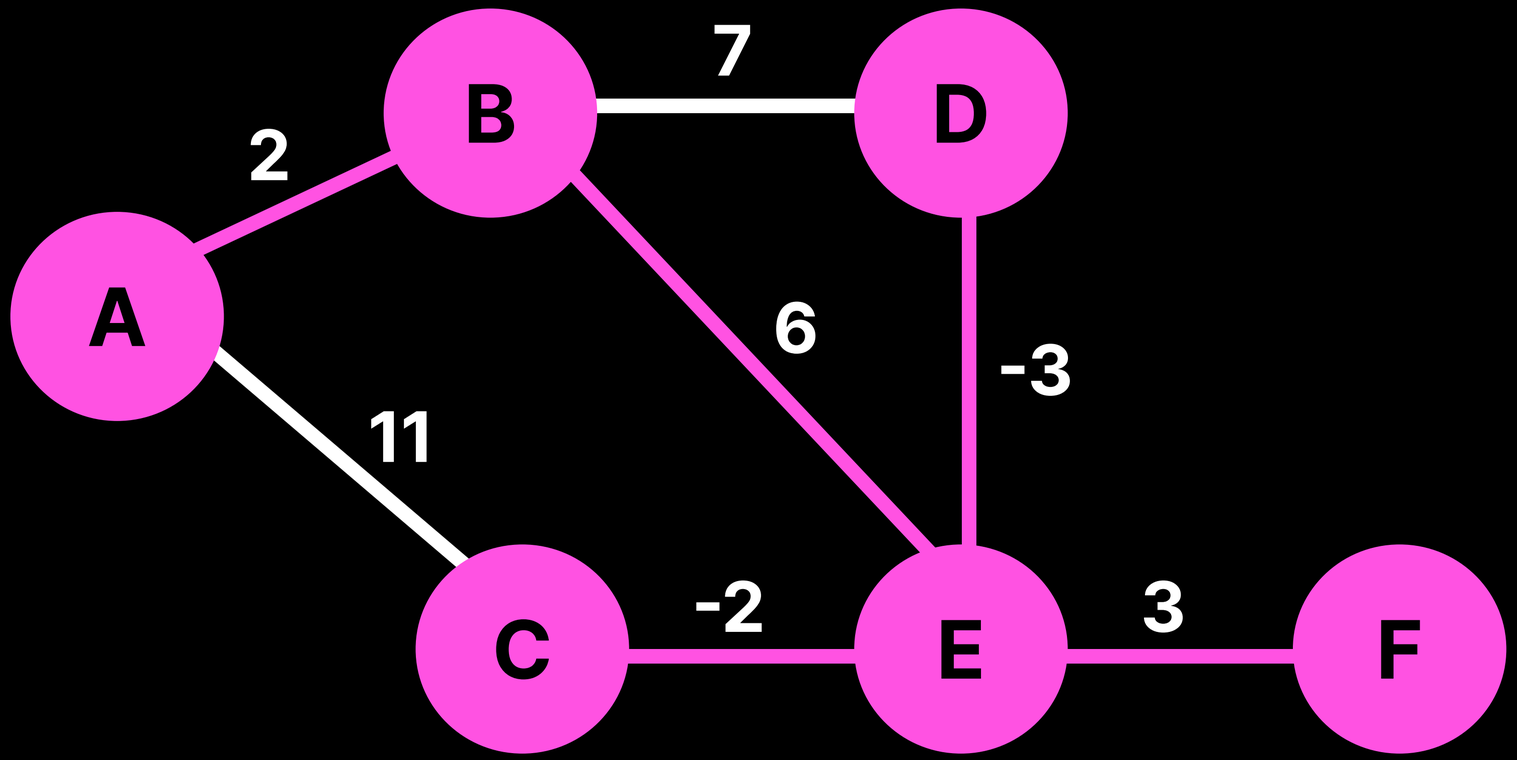
Prims



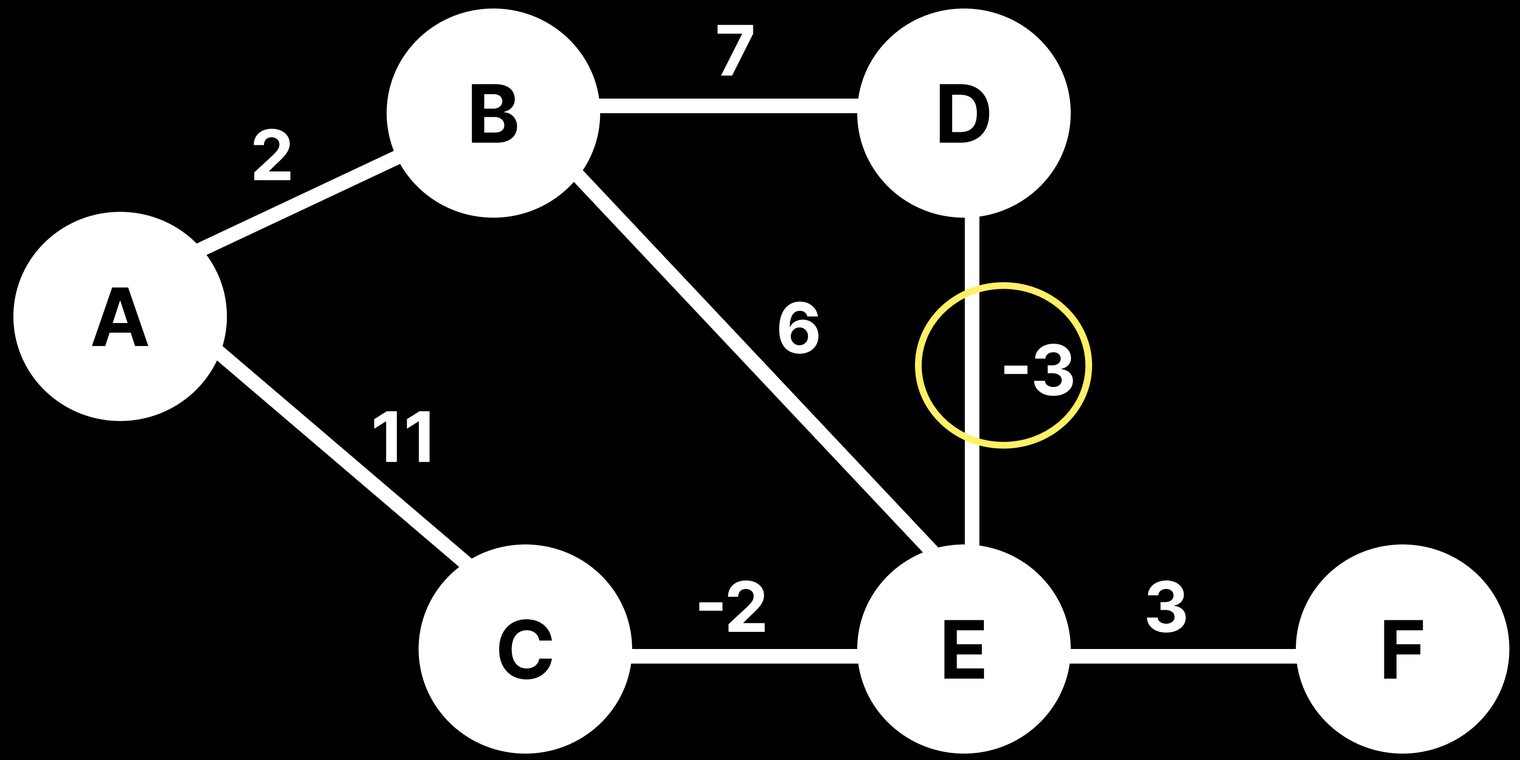
Prims



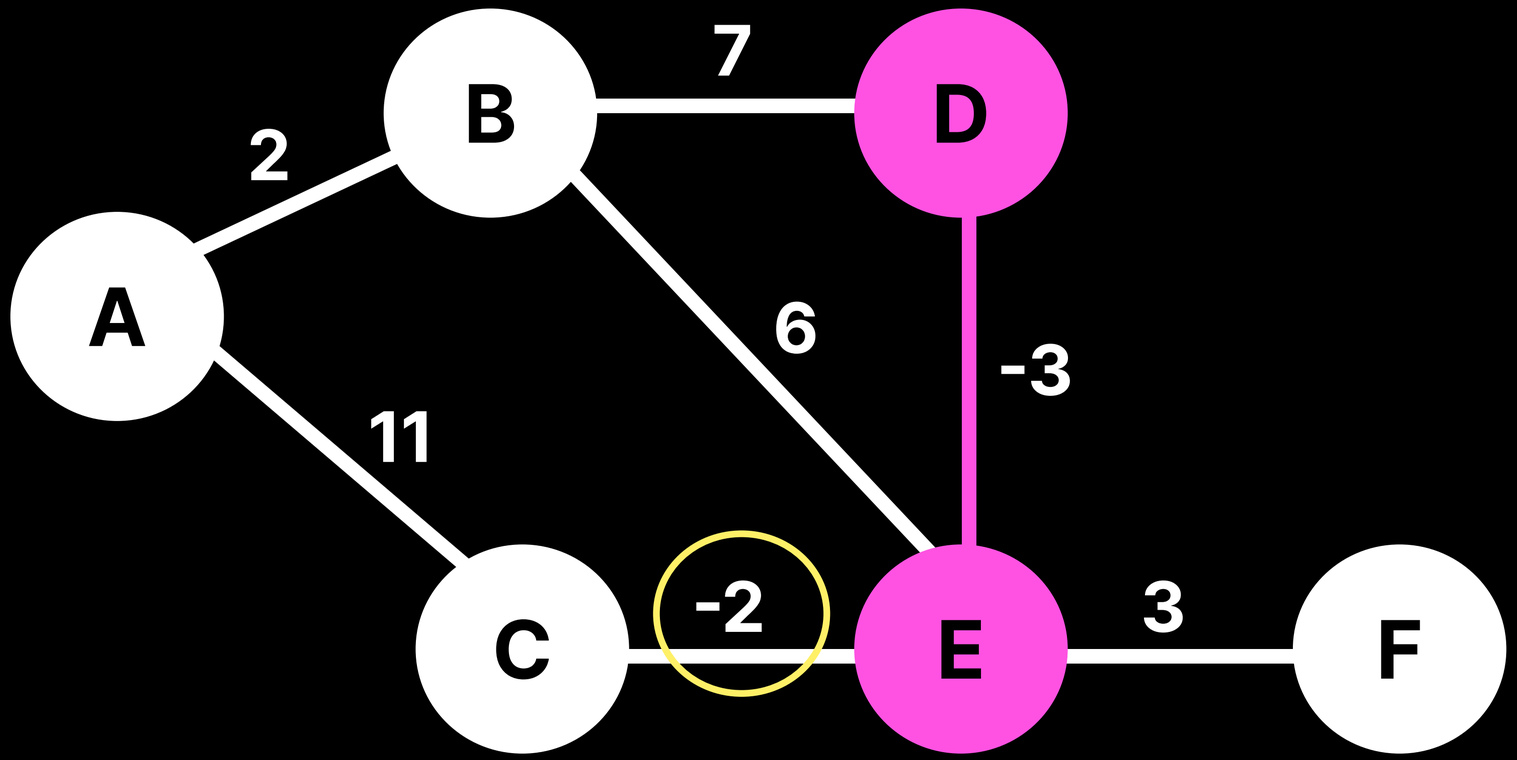
Prims



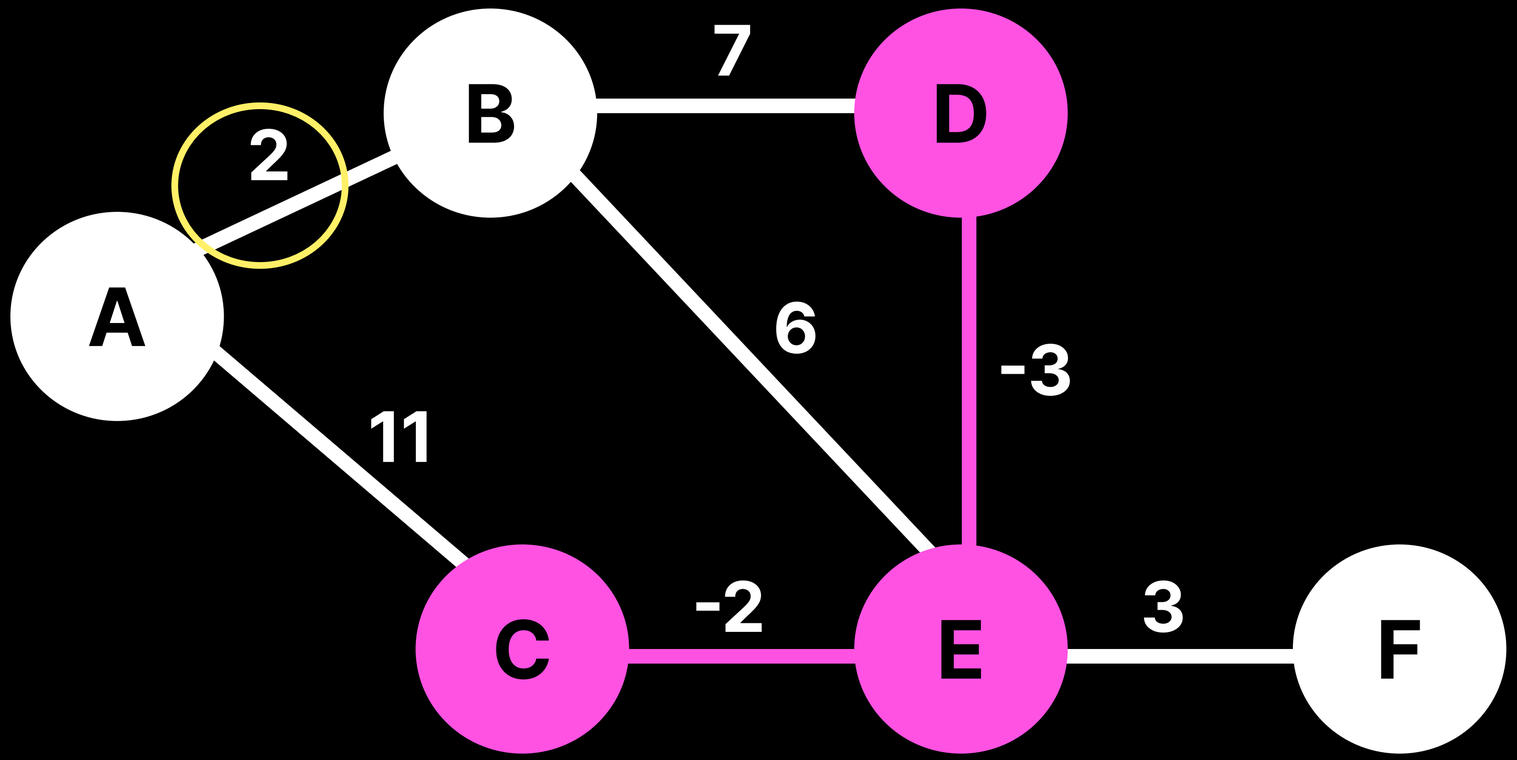
Kruskals



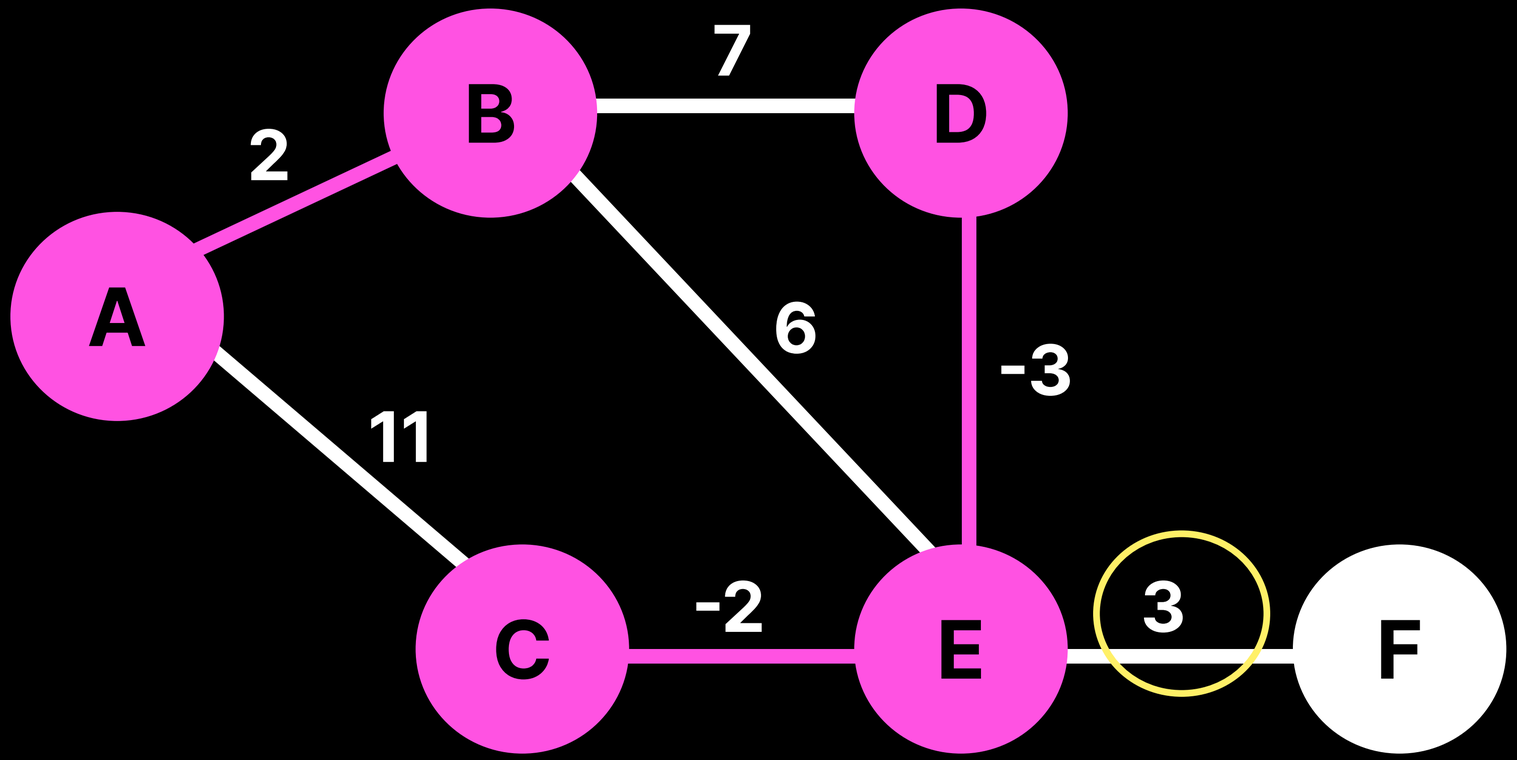
Kruskals



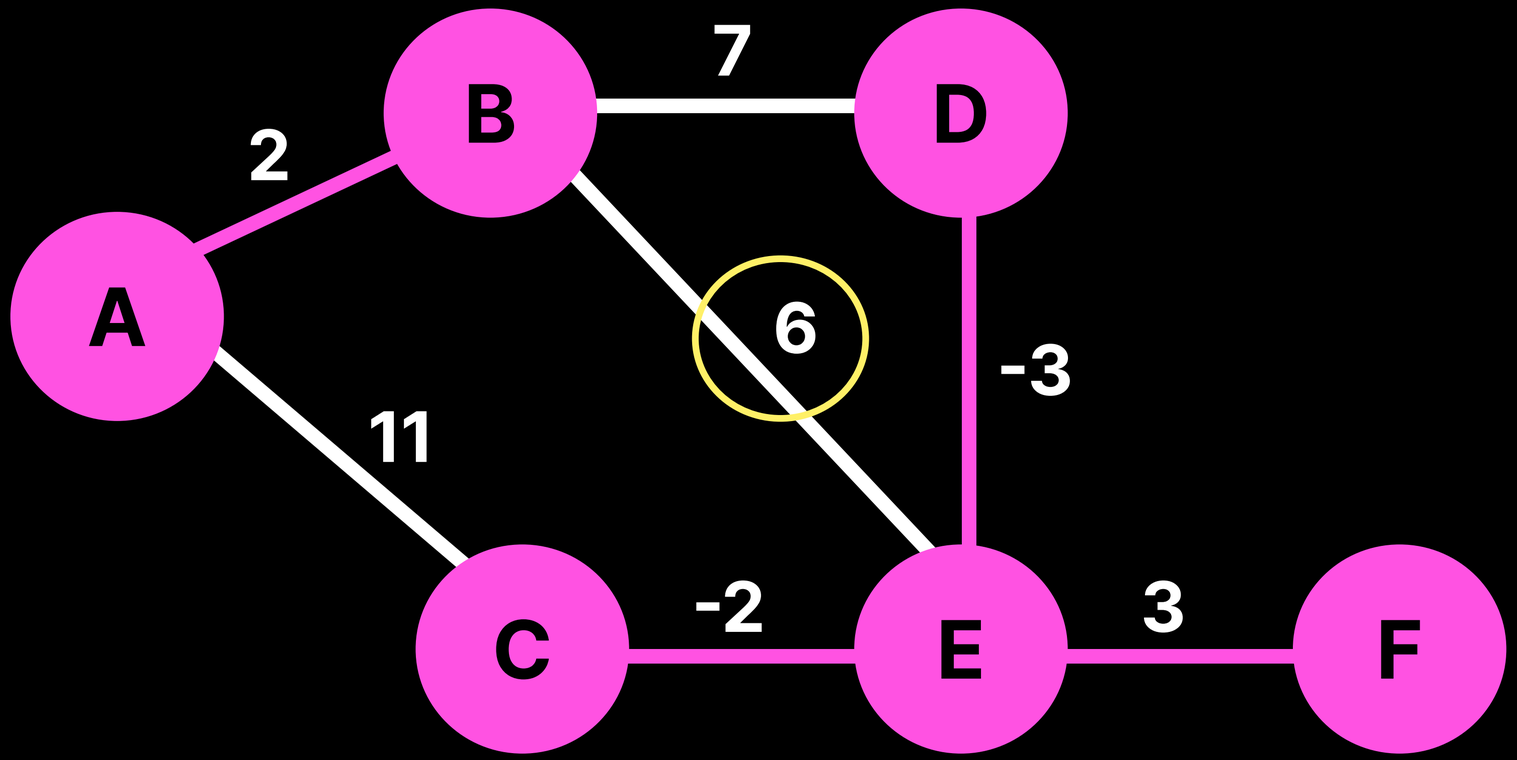
Kruskals



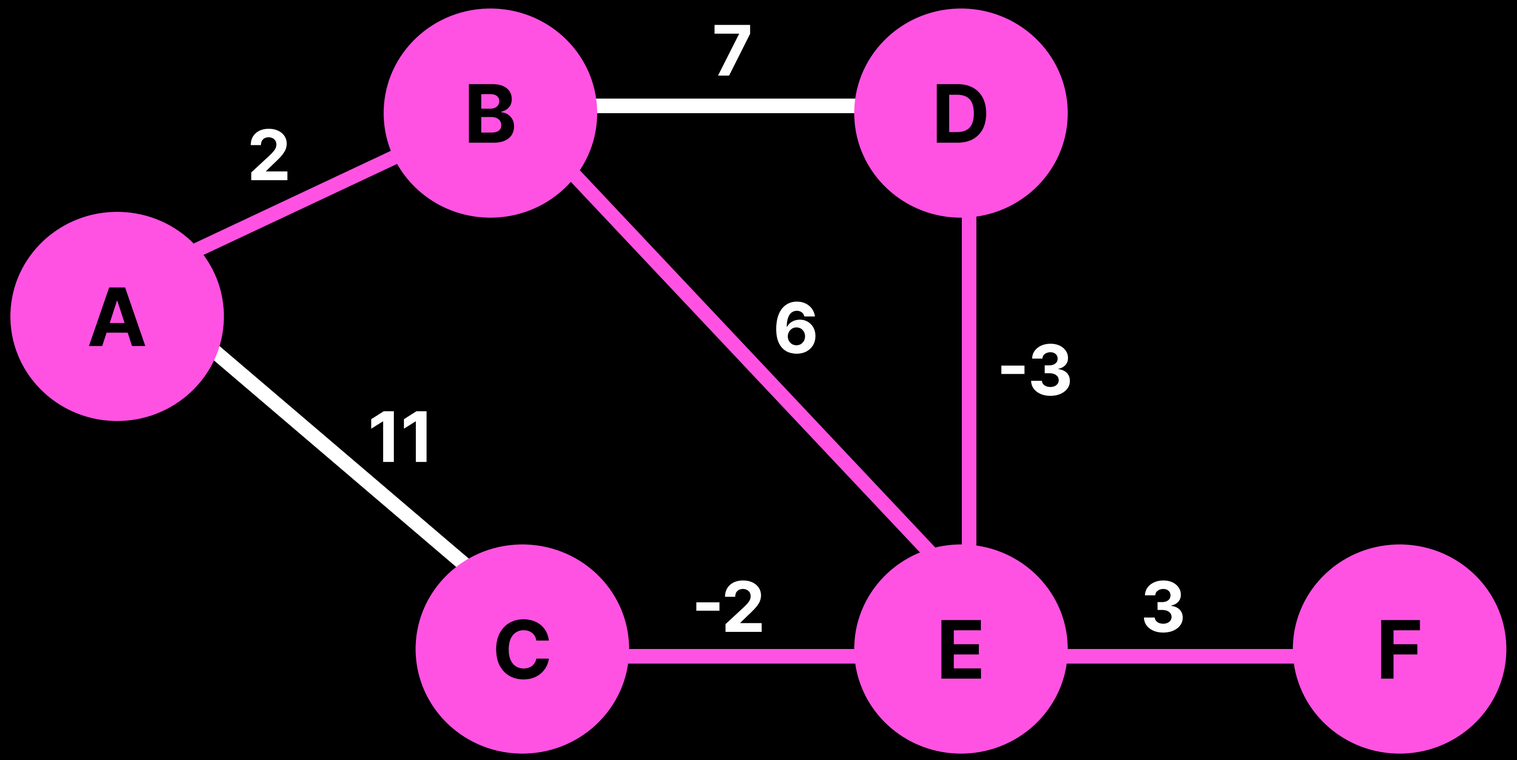
Kruskals



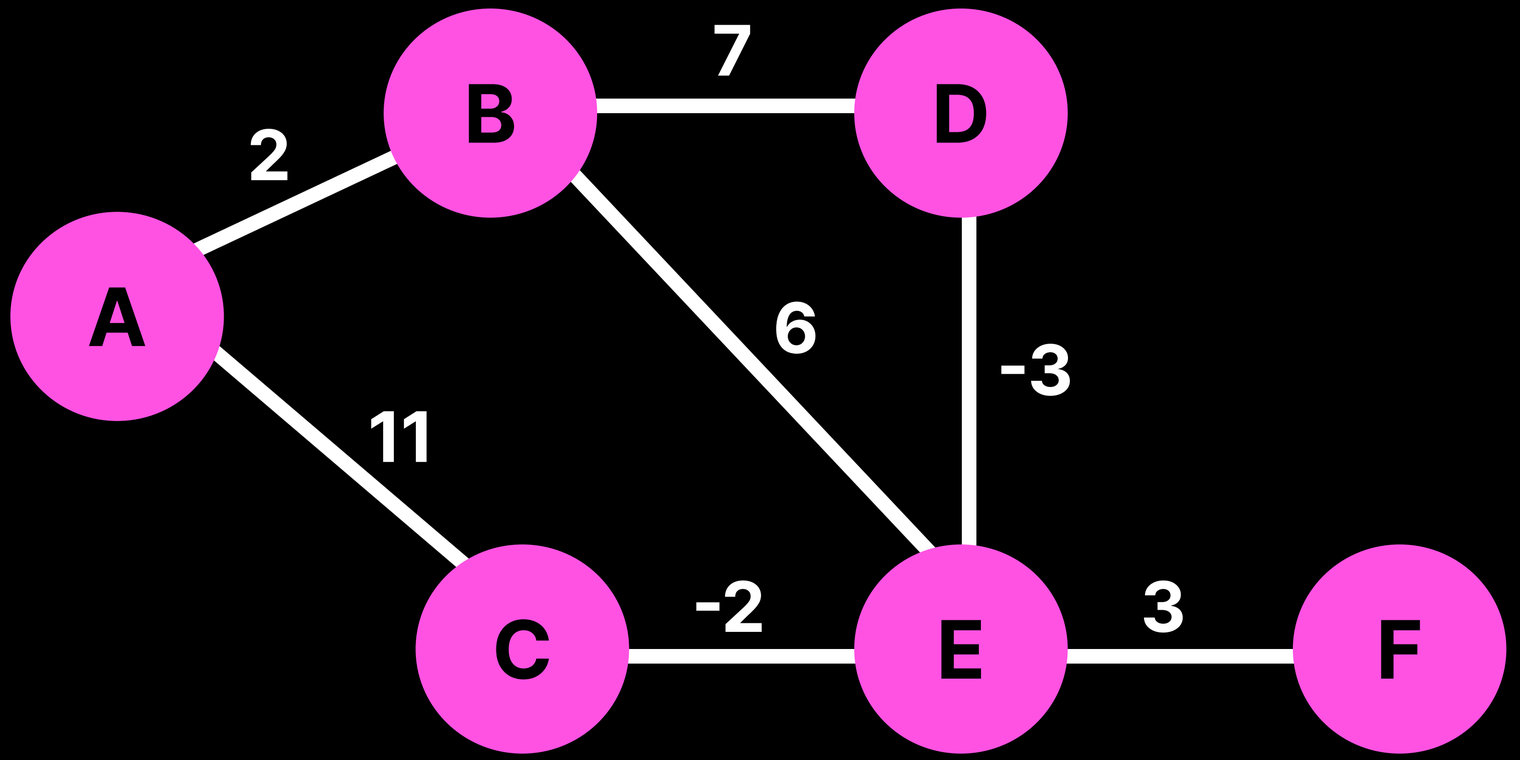
Kruskals



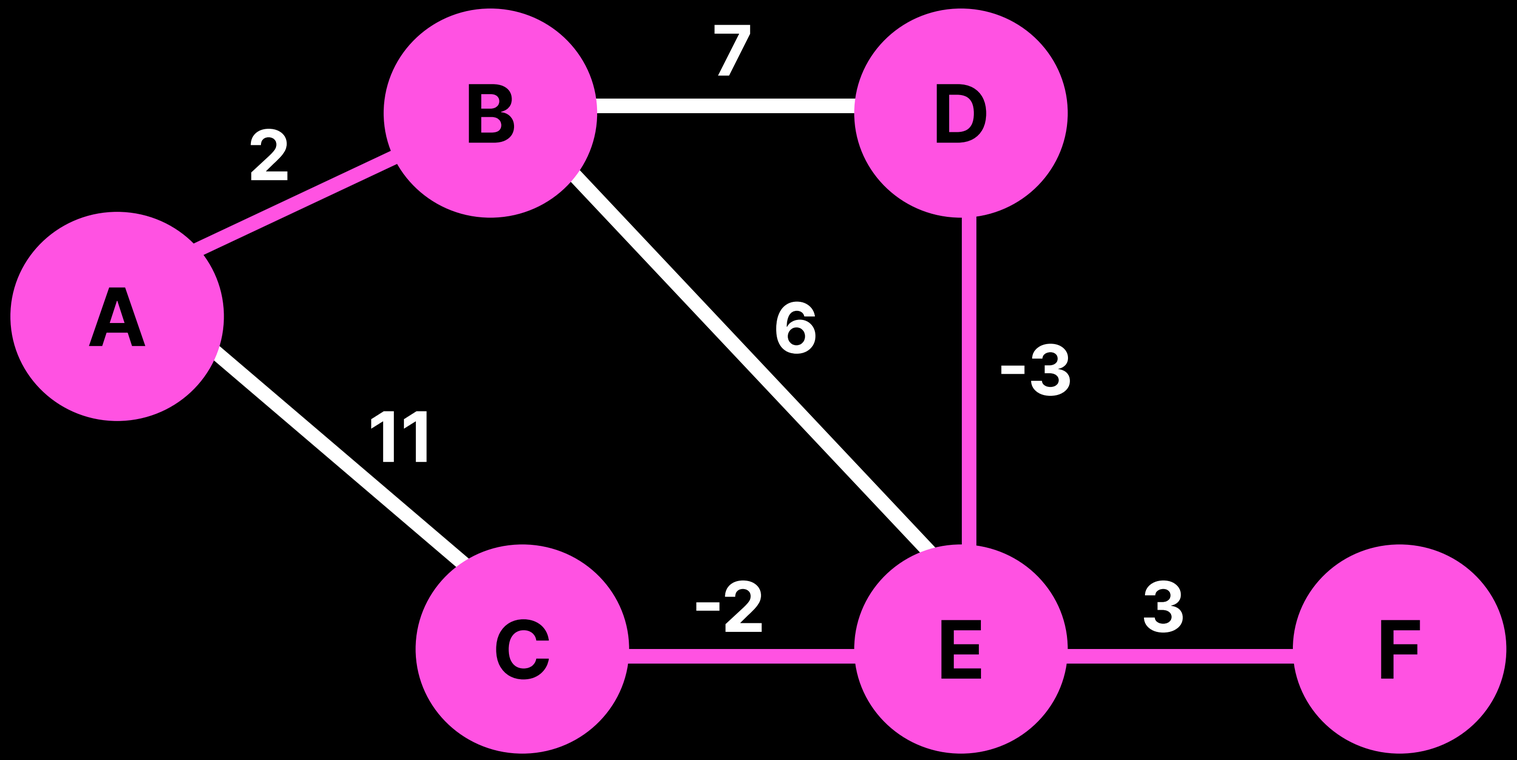
Kruskals



Borůvkas



Borůvkas



Borůvkas

