Øvingsforelesning 9

TDT4120 - Algoritmer og datastrukturer

Øving 8

Nabolister

Oppgave 2: Hvilken graf representerer følgende naboliste?

$1 \to \langle 3, 4 \rangle$

$2 \to \langle 6 \rangle$

$3 \to \langle 1, 4 \rangle$

$4 \to \langle 1, 5 \rangle$

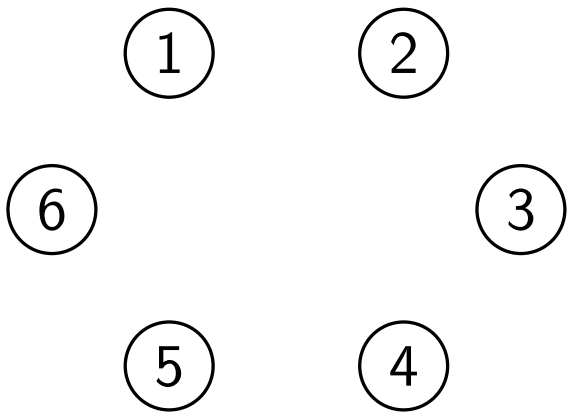
$5 \to \langle 2, 3, 6 \rangle$

$6 \to \langle 3, 5 \rangle$

Nabolister

Oppgave 2: Hvilken graf representerer følgende naboliste?

$1 \to \langle 3, 4 \rangle$
$2 \to \langle 6 \rangle$
$3 \to \langle 1, 4 \rangle$
$4 \to \langle 1, 5 \rangle$
$5 \to \langle 2, 3, 6 \rangle$
$6 \to \langle 3, 5 \rangle$



Nabolister

Oppgave 2: Hvilken graf representerer følgende naboliste?

$1 \to \langle 3, 4 \rangle$

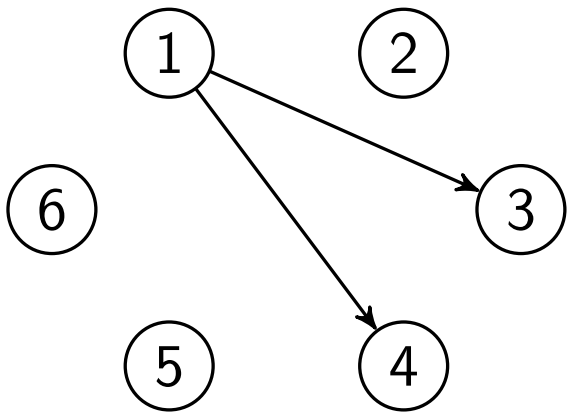
$2 \to \langle 6 \rangle$

$3 \to \langle 1, 4 \rangle$

$4 \to \langle 1, 5 \rangle$

$5 \to \langle 2, 3, 6 \rangle$

$6 \to \langle 3, 5 \rangle$



Nabolister

Oppgave 2: Hvilken graf representerer følgende naboliste?

$1 \to \langle 3, 4 \rangle$

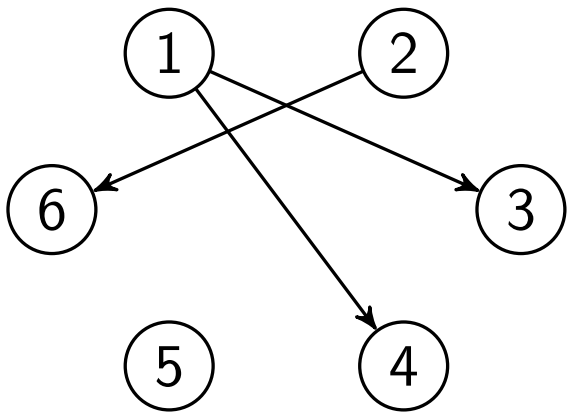
$2 \to \langle 6 \rangle$

$3 \to \langle 1, 4 \rangle$

$4 \to \langle 1, 5 \rangle$

$5 \to \langle 2, 3, 6 \rangle$

$6 \to \langle 3, 5 \rangle$



Nabolister

Oppgave 2: Hvilken graf representerer følgende naboliste?

$1 \to \langle 3, 4 \rangle$

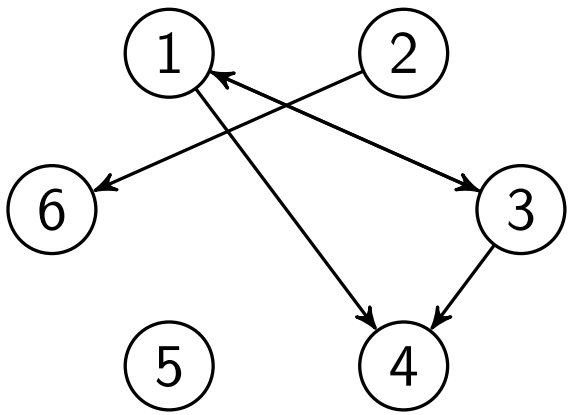
$2 \to \langle 6 \rangle$

$3 \to \langle 1, 4 \rangle$

$4 \to \langle 1, 5 \rangle$

$5 \to \langle 2, 3, 6 \rangle$

$6 \to \langle 3, 5 \rangle$



Nabolister

Oppgave 2: Hvilken graf representerer følgende naboliste?

$1 \to \langle 3, 4 \rangle$

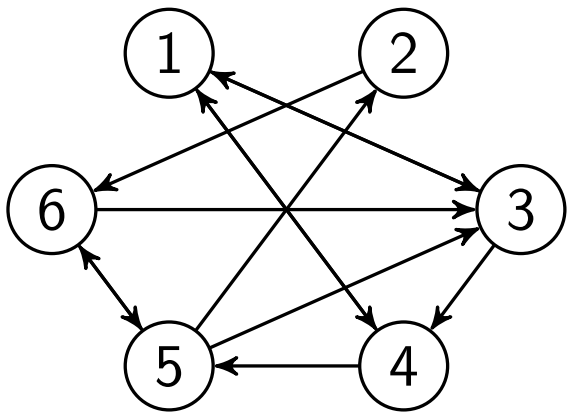
$2 \to \langle 6 \rangle$

$3 \to \langle 1, 4 \rangle$

$4 \to \langle 1, 5 \rangle$

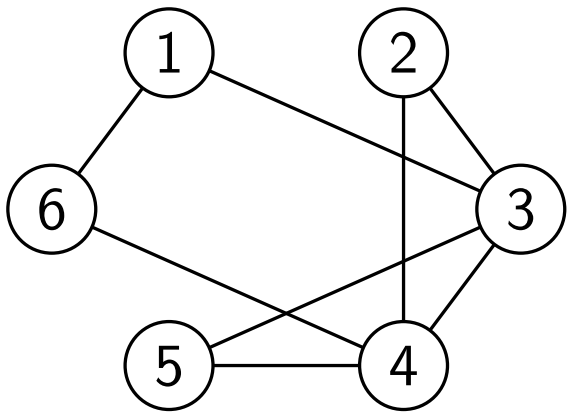
$5 \to \langle 2, 3, 6 \rangle$

$6 \to \langle 3, 5 \rangle$



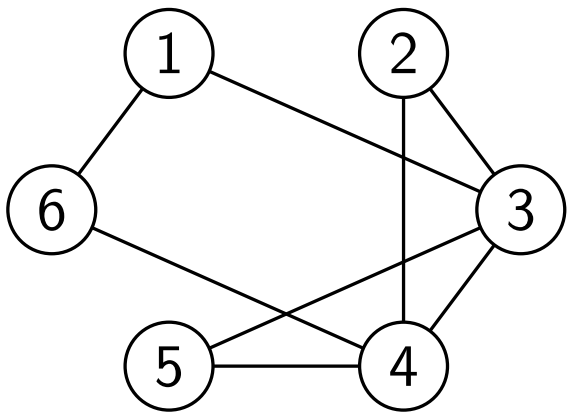
Nabomatriser

Oppgave 3: Hva er nabomatrisen til følgende graf?



Nabomatriser

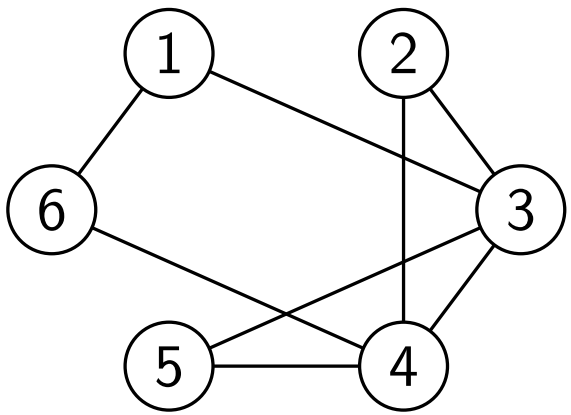
Oppgave 3: Hva er nabomatrisen til følgende graf?



$$
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\left[\begin{array}{cccccc}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]
\end{array}
$$

Nabomatriser

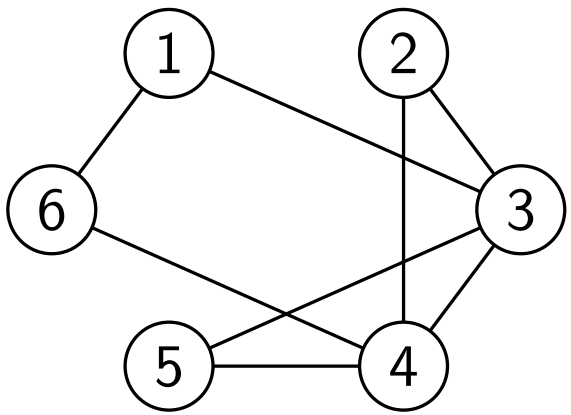
Oppgave 3: Hva er nabomatrisen til følgende graf?



$$
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\left[\begin{array}{cccccc}
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]
\end{array}
$$

Nabomatriser

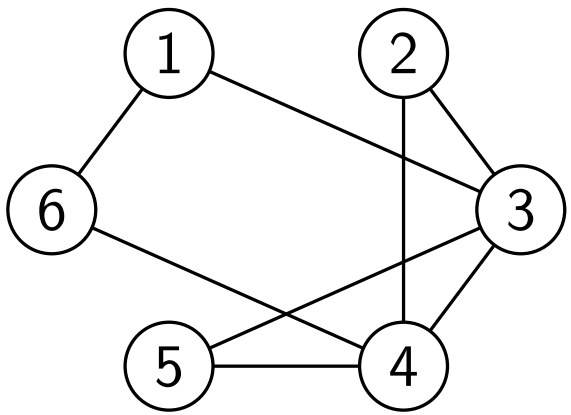
Oppgave 3: Hva er nabomatrisen til følgende graf?



$$
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\left[\begin{array}{cccccc}
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]
\end{array}
$$

Nabomatriser

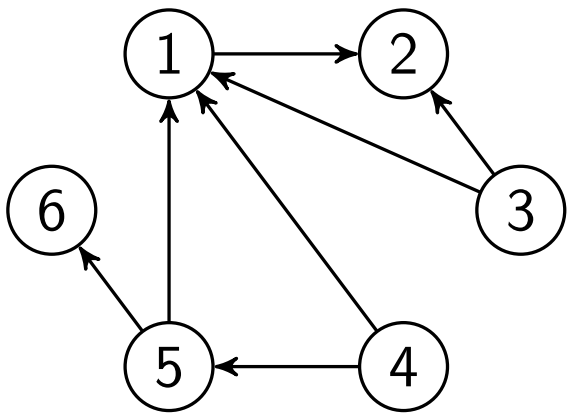
Oppgave 3: Hva er nabomatrisen til følgende graf?



$$
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\left[\begin{array}{cccccc}
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0
\end{array}\right]
\end{array}
$$

Topologisk sortering

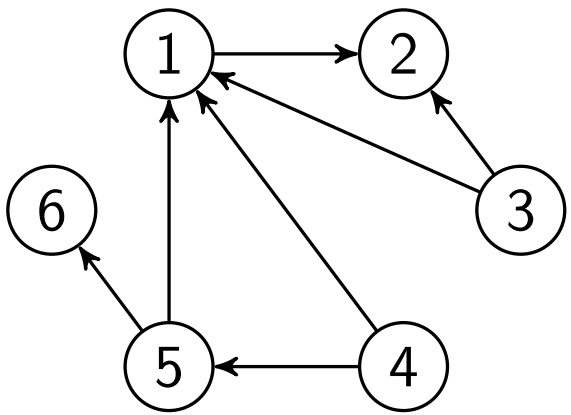
Oppgave 4: Hvilke alternativer er gyldige topologiske sorteringer av grafen?



$\langle 3, 4, 5, 6, 1, 2 \rangle$
$\langle 4, 5, 6, 1, 3, 2 \rangle$
$\langle 4, 5, 1, 6, 3, 2 \rangle$
$\langle 3, 4, 5, 1, 2, 6 \rangle$
$\langle 4, 3, 5, 1, 6, 2 \rangle$
$\langle 4, 5, 6, 3, 2, 1 \rangle$
$\langle 3, 1, 2, 4, 5, 6 \rangle$

Topologisk sortering

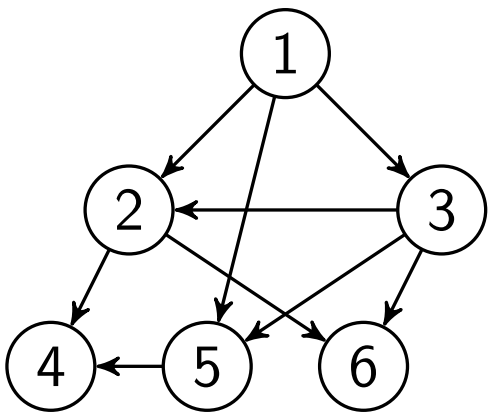
Oppgave 4: Hvilke alternativer er gyldige topologiske sorteringer av grafen?



$\langle 3, 4, 5, 6, 1, 2 \rangle$
$\langle 4, 5, 6, 1, 3, 2 \rangle$
$\langle 4, 5, 1, 6, 3, 2 \rangle$
$\langle 3, 4, 5, 1, 2, 6 \rangle$
$\langle 4, 3, 5, 1, 6, 2 \rangle$
$\langle 4, 5, 6, 3, 2, 1 \rangle$
$\langle 3, 1, 2, 4, 5, 6 \rangle$

Bredde-først-søk

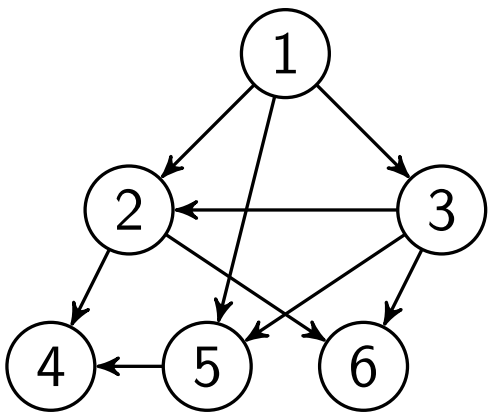
Oppgave 5: I hvilke rekkefølger kan nodene i denne grafen besøkes hvis man kjører BFS fra node 1 og nabolistene har tilfeldig rekkefølge?



$\langle 1, 2, 5, 3, 4, 6 \rangle$
$\langle 1, 2, 5, 3, 6, 4 \rangle$
$\langle 1, 2, 3, 4, 5, 6 \rangle$
$\langle 1, 5, 2, 3, 6, 4 \rangle$
$\langle 1, 3, 2, 5, 6, 4 \rangle$
$\langle 1, 3, 5, 2, 4, 6 \rangle$
$\langle 1, 5, 3, 2, 4, 6 \rangle$

Bredde-først-søk

Oppgave 5: I hvilke rekkefølger kan nodene i denne grafen besøkes hvis man kjører BFS fra node 1 og nabolistene har tilfeldig rekkefølge?



$\langle 1, 2, 5, 3, 4, 6 \rangle$
$\langle 1, 2, 5, 3, 6, 4 \rangle$
$\langle 1, 2, 3, 4, 5, 6 \rangle$
$\langle 1, 5, 2, 3, 6, 4 \rangle$
$\langle 1, 3, 2, 5, 6, 4 \rangle$
$\langle 1, 3, 5, 2, 4, 6 \rangle$
$\langle 1, 5, 3, 2, 4, 6 \rangle$

Kompatible donorer - Programmering

Oppgave 6: Gitt et sett med donorer og mottakere skal vi lage nabolister. Et donor-mottaker-par har en kant mellom seg hvis de har minst k like attributter på samme plass.

Kompatible donorer - Programmering

Oppgave 6: Gitt et sett med donorer og mottakere skal vi lage nabolister. Et donor-mottaker-par har en kant mellom seg hvis de har minst k like attributter på samme plass.

```
COMPATIBILITY-GRAPH(donors, recipients, k)
 1  let donor-edges be an array of donors.length empty arrays
 2  for i = 1 to donors.length
 3        for j = 1 to recipients.length
 4              attrs = 0
 5              ℓ = 1
 6              while ℓ ⩽ donors[i].length and k < attrs
 7                    if donors[i][ℓ] == recipients[i][ℓ]
 8                          attrs = attrs + 1
 9                    ℓ = ℓ + 1
10              if k == attrs
11                    add j to donor-edges[i]
12  return donor-edges
```

Prioritetskøer i BFS og DFS

Oppgave 7: Hvilke typer prioritetskøer bruker vi i BFS og DFS til å holde styr på rekkefølgen vi skal besøke noder vi har sett men ennå ikke besøkt?

Prioritetskøer i BFS og DFS

Oppgave 7: Hvilke typer prioritetskøer bruker vi i BFS og DFS til å holde styr på rekkefølgen vi skal besøke noder vi har sett men ennå ikke besøkt?

Oppgave 8: Hva skjer hvis vi benytter en stakk som prioritetskø i BFS?

Pakkesystem - Programmering

Oppgave 9: Skriv kode som installerer en pakke. For å installere pakken må alle avhengigheter være installert først.

Pakkesystem - Programmering

Oppgave 9: Skriv kode som installerer en pakke. For å installere pakken må alle avhengigheter være installert først.

```
RESOLVE-AND-INSTALL(package)
1  for p ∈ package.dependencies
2        if not p.is-installed
3              RESOLVE-AND-INSTALL(p)
4  INSTALL(package)
```

Grafrepresentasjon

Oppgave 10: Hva er minnekompleksiteten til en graf representert med henholdsvis nabolister og nabomatrise?

Oppgave 11: Hva er tidskompleksiteten for å sjekke om en kant (u, v) eksisterer i en graf med disse representasjonene?

Grafrepresentasjon

Oppgave 10: Hva er minnekompleksiteten til en graf representert med henholdsvis nabolister og nabomatrise?

Oppgave 11: Hva er tidskompleksiteten for å sjekke om en kant (u, v) eksisterer i en graf med disse representasjonene?

Nabolister: En liste for hver node med naboene til noden (kanter).
Nabomatrise: En $n \times n$ matrise med en verdi per nodepar.

Grafrepresentasjon

Oppgave 10: Hva er minnekompleksiteten til en graf representert med henholdsvis nabolister og nabomatrise?

Oppgave 11: Hva er tidskompleksiteten for å sjekke om en kant (u, v) eksisterer i en graf med disse representasjonene?

Nabolister: En liste for hver node med naboene til noden (kanter).
Nabomatrise: En $n \times n$ matrise med en verdi per nodepar.

Nabolister: $\Theta(\mathrm{V} + \mathrm{E})$ minne
Nabomatrise: $\Theta(\mathrm{V}^2)$ minne

Grafrepresentasjon

Oppgave 10: Hva er minnekompleksiteten til en graf representert med henholdsvis nabolister og nabomatrise?

Oppgave 11: Hva er tidskompleksiteten for å sjekke om en kant (u, v) eksisterer i en graf med disse representasjonene?

Nabolister: En liste for hver node med naboene til noden (kanter).
Nabomatrise: En $n \times n$ matrise med en verdi per nodepar.

Nabolister: $\Theta(\mathrm{V} + \mathrm{E})$ minne – $\mathrm{O}(\mathrm{E})$ tid
Nabomatrise: $\Theta(\mathrm{V}^2)$ minne – $\Theta(1)$ tid

3×3-spill

Oppgave 12: Du har et 3×3-spill hvor en brikke mangler. Du kan skyve andre brikker inn i den tomme plassen og ønsker å gjøre minst mulig slike flyttinger for å få plassert alle brikkene på riktig plass. Hvilken algoritme ville du brukt for å finne ut av dette?

2	3	6
5	8	
1	4	7

1	2	3
4	5	6
7	8	

3×3-spill

Oppgave 12: Du har et 3×3-spill hvor en brikke mangler. Du kan skyve andre brikker inn i den tomme plassen og ønsker å gjøre minst mulig slike flyttinger for å få plassert alle brikkene på riktig plass. Hvilken algoritme ville du brukt for å finne ut av dette?

2	3	6
5	8	
1	4	7

1	2	3
4	5	6
7	8	

Mulig algoritme:

1. Representer som en graf ved å la nodene være alle mulig tilstander og kantene være tilstander man kan gå mellom ved å skyve en brikke.
2. Bruk BFS fra starttilstanden til å finne stien med lavest antall kanter til sluttilstanden.

Emnerekkefølge

Oppgave 13: Ønsker å ta et sett med emner i riktig rekkefølge, slik at du oppfyller forkunnskapskravene i hvert emne før du tar det. Hvilken algoritme ville du brukt for å finne en slik rekkefølge?

Emnerekkefølge

Oppgave 13: Ønsker å ta et sett med emner i riktig rekkefølge, slik at du oppfyller forkunnskapskravene i hvert emne før du tar det. Hvilken algoritme ville du brukt for å finne en slik rekkefølge?

Mulig algoritme:

1. Lag en graf bestående av emnene med kanter fra et emne til et annet hvis det er et forkunnskapskrav.
2. Bruk Topological-Sort.

Internasjonalt konglomerat

Oppgave 14: Du har et tre bestående av alle de ansatte i et konglomerat. Du ønsker å lage en datastruktur hvor man i konstant tid kan sjekke om en ansatt er underordnet en annen ansatt. Datastrukturen kan bruke $\mathrm{O}(n)$ minne. Hvilken algoritme ville du brukt for å finne ut av dette?

Internasjonalt konglomerat

Oppgave 14: Du har et tre bestående av alle de ansatte i et konglomerat. Du ønsker å lage en datastruktur hvor man i konstant tid kan sjekke om en ansatt er underordnet en annen ansatt. Datastrukturen kan bruke $\mathrm{O}(n)$ minne. Hvilken algoritme ville du brukt for å finne ut av dette?

Kan bruke DFS.
Underordnet u og overordnet v tilsier at $v.d < u.d$ og $v.f > u.f$.

BFS og DFS

Oppgave 15: Hva er sant om BFS?

- Kan brukes til å finne korteste vei fra en node til en annen dersom alle kantvektene er like og ikke-negative.
- Man er garantert at alle nodene blir besøkt i traverseringen
- Fungerer ikke om grafen har sykler.

BFS og DFS

Oppgave 15: Hva er sant om BFS?

- Kan brukes til å finne korteste vei fra en node til en annen dersom alle kantvektene er like og ikke-negative.
- ~~Man er garantert at alle nodene blir besøkt i traverseringen~~
- Fungerer ikke om grafen har sykler.

BFS og DFS

Oppgave 15: Hva er sant om BFS?

- Kan brukes til å finne korteste vei fra en node til en annen dersom alle kantvektene er like og ikke-negative.
- ~~Man er garantert at alle nodene blir besøkt i traverseringen~~
- ~~Fungerer ikke om grafen har sykler.~~

BFS og DFS

Oppgave 15: Hva er sant om BFS?

Oppgave 16: Hva er sant om DFS?

- Man er garantert at alle nodene blir besøkt i traversingen.
- Kan brukes til kantklassifisering.
- Kan brukes til å finne kortest vei fra en node til en annen dersom alle kantvektene er like og ikke-negative.
- Fungerer ikke om grafen har sykler.

BFS og DFS

Oppgave 15: Hva er sant om BFS?

Oppgave 16: Hva er sant om DFS?

- Man er garantert at alle nodene blir besøkt i traversingen.
- Kan brukes til kantklassifisering.
- ~~Kan brukes til å finne kortest vei fra en node til en annen dersom alle kantvektene er like og ikke-negative.~~
- Fungerer ikke om grafen har sykler.

BFS og DFS

Oppgave 15: Hva er sant om BFS?

Oppgave 16: Hva er sant om DFS?

- Man er garantert at alle nodene blir besøkt i traversingen.
- Kan brukes til kantklassifisering.
- ~~Kan brukes til å finne kortest vei fra en node til en annen dersom alle kantvektene er like og ikke-negative.~~
- ~~Fungerer ikke om grafen har sykler.~~

Kantklassifisering

Oppgave 17: Hvilke typer kanter kan du ende opp med ved kantklassifisering av et tre?

Oppgave 18: Hvilke typer kanter kan du ende opp med ved kantklassifisering av en asyklisk graf?

Kantklassifisering

Oppgave 17: Hvilke typer kanter kan du ende opp med ved kantklassifisering av et tre?

Oppgave 18: Hvilke typer kanter kan du ende opp med ved kantklassifisering av en asyklisk graf?

Tre-kant: Kant som ble brukt første gangen en node ble oppdaget.
Bakoverkant: Kant som går til en forgjenger.
Foroverkant: Ikke-tre-kant som går til en etterkommer.
Krysskant: Alle andre kanter.

Kantklassifisering

Oppgave 17: Hvilke typer kanter kan du ende opp med ved kantklassifisering av et tre?

Oppgave 18: Hvilke typer kanter kan du ende opp med ved kantklassifisering av en asyklisk graf?

Tre-kant: Kant som ble brukt første gangen en node ble oppdaget.
Bakoverkant: Kant som går til en forgjenger.
Foroverkant: Ikke-tre-kant som går til en etterkommer.
Krysskant: Alle andre kanter.

Tre: kun tre-kanter.

Kantklassifisering

Oppgave 17: Hvilke typer kanter kan du ende opp med ved kantklassifisering av et tre?

Oppgave 18: Hvilke typer kanter kan du ende opp med ved kantklassifisering av en asyklisk graf?

Tre-kant: Kant som ble brukt første gangen en node ble oppdaget.
Bakoverkant: Kant som går til en forgjenger.
Foroverkant: Ikke-tre-kant som går til en etterkommer.
Krysskant: Alle andre kanter.

Tre: kun tre-kanter.
DAG: tre-kanter, foroverkanter og krysskanter.

Veibygging - Programmering

Oppgave 19: Gitt et rutenett over hvor det er mulig å bygge vei ønsker vi å finne korteste mulige vei som kan bygges mellom to steder i rutenettet, hvis mulig. Skriv kode som gjøre dette.

Veibygging - Programmering

Oppgave 19: Gitt et rutenett over hvor det er mulig å bygge vei ønsker vi å finne korteste mulige vei som kan bygges mellom to steder i rutenettet, hvis mulig. Skriv kode som gjøre dette.

Mulig algoritme:

1. Lag en graf hvor alle nodene er ruter det er mulig å bygge i, og kantene er ruter som ligger inntil hverandre.
2. Bruk et bredde-først-søk fra start til slutt.

Topologisk sortering med tabell

Oppgave 20: Vil kjøretiden endre seg hvis vi bruker en dynamisk tabell i stedet for en lenket liste i TOPOLOGICAL-SORT? Kan vi i så fall unngå endringen i tidskompleksitet?

TOPOLOGICAL-SORT(G)
1 call DFS(G) to computer finishing times $v.f$ for each vertex v
2 as each vertex is finished, insert it onto the front of a linked list
3 **return** the linked list of vertices

Topologisk sortering med tabell

Oppgave 20: Vil kjøretiden endre seg hvis vi bruker en dynamisk tabell i stedet for en lenket liste i TOPOLOGICAL-SORT? Kan vi i så fall unngå endringen i tidskompleksitet?

TOPOLOGICAL-SORT(G)
1 call DFS(G) to computer finishing times $v.f$ for each vertex v
2 as each vertex is finished, insert it onto the front of a linked list
3 **return** the linked list of vertices

Innsettelse på starten av en dynamisk tabell tar $\Theta(n)$ tid.

Topologisk sortering med tabell

Oppgave 20: Vil kjøretiden endre seg hvis vi bruker en dynamisk tabell i stedet for en lenket liste i TOPOLOGICAL-SORT? Kan vi i så fall unngå endringen i tidskompleksitet?

TOPOLOGICAL-SORT(G)
1 call DFS(G) to computer finishing times $v.f$ for each vertex v
2 as each vertex is finished, insert it onto the front of a linked list
3 **return** the linked list of vertices

Innsettelse på starten av en dynamisk tabell tar $\Theta(n)$ tid.
Kjøretid på $\Theta(V^2 + E)$

Traversering - minprioritetskø

Oppgave 21: Vil $u.d$ være den korteste avstanden fra s til u etter å ha kjørt $\textsc{Traverse}(s)$? Hva må vi eventuelt anta for at dette skal stemme?

```
TRAVERSE(G, s)
 1  let Q be an empty min-priority queue of vertices with u.d as priority
 2  for each vertex u ∈ G.V
 3        u.d = ∞
 4  s.d = 0
 5  INSERT(Q, s)
 6  while Q ≠ ∅
 7        u = EXTRACT-MIN(Q)
 8        for v ∈ G.adj[u]
 9              if v.d = ∞
10                    INSERT(Q, v)
11              v.d = min(v.d, u.d+weight of edge between u and v)
12              DECREASE-KEY(Q, v, v.d)
```