Øvingsforelesning 8: Traversering av grafer

TDT4120 Algoritmer og datastrukturer

Fredrik Strupe

22. oktober 2019

NTNU

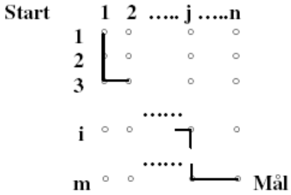
Plan

1. Gjennomgang av øving 6
2. Denne ukens pensum
    2.1 Grafrepresentasjon
    2.2 Bredde-først-søk
    2.3 Dybde-først-søk
    2.4 Topologisk sortering
3. Tips til øving 8, teori
4. Tips til øving 8, praksis
5. Bonus

Gjennomgang av øving 6

Teorioppgave 6

I denne oppgaven skal vi ta for oss et rektangulært rutenett gitt som følger:



Vi skal nå finne hvor mange veier det er fra Start $[1, 1]$ til Mål $[m, n]$ under visse restriksjoner. En lovlig vei fra Start til Mål defineres ved at et skritt fra $[i, j]$ på veien skal gå enten ned ($[i + 1, j]$) eller til høyre ($[i, j + 1]$). To veier er forskjellige dersom de ikke er identiske, skritt for skritt. Funksjonen $f(i, j)$ skal gi antall veier fra $[1, 1]$ til $[i, j]$. Dette fører til at $f(1, 2) = 1$ og $f(3, 2) = 3$.

Hva blir $f(1, 3)$?

Teorioppgave 6

	1	2	3	4	5
1					
2					
3					
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1				
2					
3					
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1			
2					
3					
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1		
2					
3					
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	
2					
3					
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2					
3					
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1				
3					
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2			
3					
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3		
3					
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	
3					
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3					
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1				
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3			
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6		
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4					
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1				
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4			
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4	10		
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4	10	20	
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4	10	20	35
5					

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4	10	20	35
5	1				

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4	10	20	35
5	1	5			

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4	10	20	35
5	1	5	15		

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4	10	20	35
5	1	5	15	35	

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4	10	20	35
5	1	5	15	35	70

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4	10	20	35
5	1	5	15	35	70

$$f(1,3) = 1$$

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4	10	20	35
5	1	5	15	35	70

$$f(1,3) = 1$$
$$f(5,4) = 35$$

Teorioppgave 6

	1	2	3	4	5
1	1	1	1	1	1
2	1	2	3	4	5
3	1	3	6	10	15
4	1	4	10	20	35
5	1	5	15	35	70

$f(1,3) = 1$
$f(5,4) = 35$

$f(m,n) =$
$f(m-1,n) +$
$f(m,n-1)$

Teorioppgave 9

Anta en stav med lengde N. Et kutt av staven med lengde i kan selges for p_i, hvor $i \in \{1, 2, \ldots, N\}$.

Finn hvordan staven skal kuttes opp slik at du maksimerer inntekten R ved å selge staven.

Hva blir inntekten R når

$N = 4$ og

$p_1 = 4,\ p_2 = 5,\ p_3 = 12,\ p_4 = 15$

Teorioppgave 9

Brute-force:

Teorioppgave 9

Brute-force:

$$1, 1, 1, 1 \rightarrow R = 16 \tag{1}$$
$$1, 3 \rightarrow R = 16 \tag{2}$$
$$2, 2 \rightarrow R = 10 \tag{3}$$
$$4 \rightarrow R = 15 \tag{4}$$

Teorioppgave 10

Hvis den unike løsningen for stavkutting når $n = 5$ består av å kutte staven i to deler med lengde 1 og 4, hvilke(t) av disse alternativene kan da ikke være optimal når $n = 6$? Ikke ta utgangspunkt i prisene over.

- [ ] 6 staver av lengde 1
- [ ] 2 staver av lengde 2 og 4
- [ ] 2 staver av lengde 3
- [ ] 3 staver av lengde 2

Teorioppgave 10

Hvis den unike løsningen for stavkutting når $n = 5$ består av å kutte staven i to deler med lengde 1 og 4, hvilke(t) av disse alternativene kan da ikke være optimal når $n = 6$? Ikke ta utgangspunkt i prisene over.

- [x] 6 staver av lengde 1
- [ ] 2 staver av lengde 2 og 4
- [ ] 2 staver av lengde 3
- [x] 3 staver av lengde 2

Praksisoppgave 3

Lag en matrise der hver celle inneholder den letteste totalvekten til cellen.

$$\begin{bmatrix} 3 & 6 & 8 & 6 & 3 \\ 7 & 6 & 5 & 7 & 3 \\ 4 & 10 & 4 & 1 & 6 \\ 10 & 4 & 3 & 1 & 2 \\ 6 & 1 & 7 & 3 & 9 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 6 & 8 & 6 & 3 \\ 10 & 9 & 11 & 10 & 6 \\ 13 & 19 & 13 & 7 & 12 \\ 23 & 17 & 10 & 8 & 9 \\ 23 & 11 & 15 & 11 & 17 \end{bmatrix}$$

Praksisoppgave 3

```
function cumulative(weights)
    pathweights = deepcopy(weights)
    rows, cols = size(weights)
    for i = 2:rows
        for j = 1:cols
            t = fill(Inf, 3)
            t[1] = pathweights[i-1, j]
            if j > 1
                t[2] = pathweights[i-1, j-1]
            end
            if j < cols
                t[3] = pathweights[i-1, j+1]
            end
            pathweights[i, j] = weights[i, j] + minimum(t)
        end
    end
    return pathweights
end
```

Denne ukens pensum

Pensum: Kap. 22.1-22.4

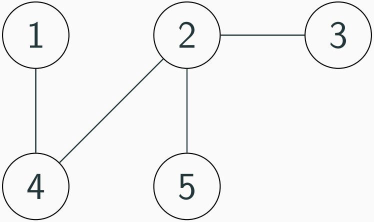
Denne ukens pensum

Grafrepresentasjon

Grafrepresentasjon

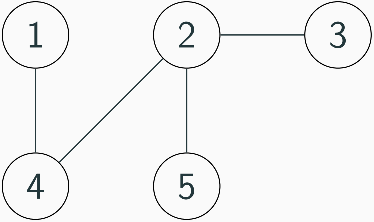
Grafrepresentasjon

Gitt følgende graf:



Grafrepresentasjon

Gitt følgende graf:



Hvordan kan vi representere grafen på en mer maskinvennlig måte?

Grafrepresentasjon

Hovedsakelig to måter:

Grafrepresentasjon

Hovedsakelig to måter:

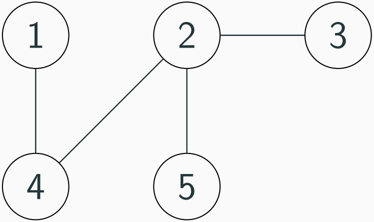
1. Som nabolister

Grafrepresentasjon

Hovedsakelig to måter:

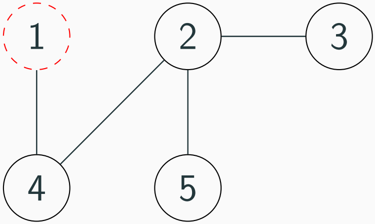
1. Som nabolister
2. Som nabomatrise

Nabolister



$$1 \rightarrow [\ ]$$
$$2 \rightarrow [\ ]$$
$$3 \rightarrow [\ ]$$
$$4 \rightarrow [\ ]$$
$$5 \rightarrow [\ ]$$

Nabolister



$1 \rightarrow [\ ]$

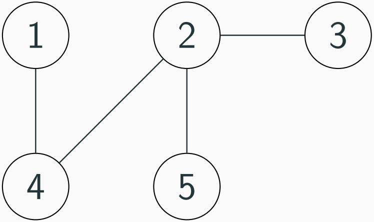
$2 \rightarrow [\ ]$

$3 \rightarrow [\ ]$

$4 \rightarrow [\ ]$

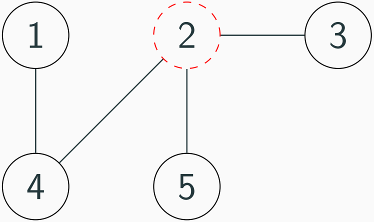
$5 \rightarrow [\ ]$

Nabolister



$$
\begin{aligned}
1 &\rightarrow \begin{bmatrix} 4 \end{bmatrix} \\
2 &\rightarrow \begin{bmatrix} \; \end{bmatrix} \\
3 &\rightarrow \begin{bmatrix} \; \end{bmatrix} \\
4 &\rightarrow \begin{bmatrix} \; \end{bmatrix} \\
5 &\rightarrow \begin{bmatrix} \; \end{bmatrix}
\end{aligned}
$$

Nabolister



$$1 \rightarrow \begin{bmatrix} 4 \end{bmatrix}$$

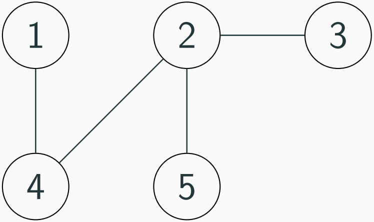
$$2 \rightarrow \begin{bmatrix} \ \end{bmatrix}$$

$$3 \rightarrow \begin{bmatrix} \ \end{bmatrix}$$

$$4 \rightarrow \begin{bmatrix} \ \end{bmatrix}$$

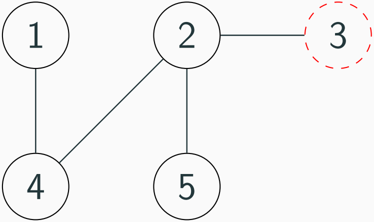
$$5 \rightarrow \begin{bmatrix} \ \end{bmatrix}$$

Nabolister



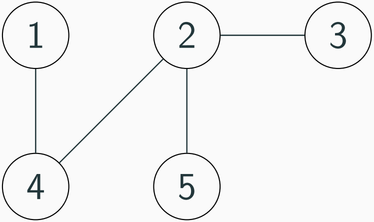
$$
\begin{aligned}
1 &\rightarrow \begin{bmatrix} 4 \end{bmatrix} \\
2 &\rightarrow \begin{bmatrix} 3 & 4 & 5 \end{bmatrix} \\
3 &\rightarrow \begin{bmatrix} \; \end{bmatrix} \\
4 &\rightarrow \begin{bmatrix} \; \end{bmatrix} \\
5 &\rightarrow \begin{bmatrix} \; \end{bmatrix}
\end{aligned}
$$

Nabolister



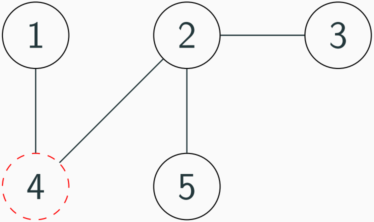
$$
\begin{aligned}
1 &\rightarrow \begin{bmatrix} 4 \end{bmatrix} \\
2 &\rightarrow \begin{bmatrix} 3 & 4 & 5 \end{bmatrix} \\
3 &\rightarrow \begin{bmatrix} \ \end{bmatrix} \\
4 &\rightarrow \begin{bmatrix} \ \end{bmatrix} \\
5 &\rightarrow \begin{bmatrix} \ \end{bmatrix}
\end{aligned}
$$

Nabolister



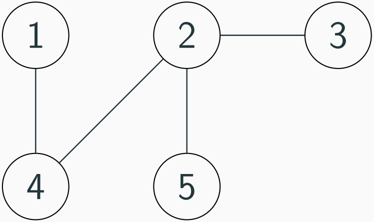
$$
\begin{aligned}
1 &\rightarrow \begin{bmatrix} 4 \end{bmatrix} \\
2 &\rightarrow \begin{bmatrix} 3 & 4 & 5 \end{bmatrix} \\
3 &\rightarrow \begin{bmatrix} 2 \end{bmatrix} \\
4 &\rightarrow \begin{bmatrix} \ \end{bmatrix} \\
5 &\rightarrow \begin{bmatrix} \ \end{bmatrix}
\end{aligned}
$$

Nabolister



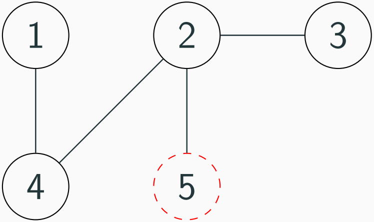
$$1 \rightarrow \begin{bmatrix} 4 \end{bmatrix}$$
$$2 \rightarrow \begin{bmatrix} 3 & 4 & 5 \end{bmatrix}$$
$$3 \rightarrow \begin{bmatrix} 2 \end{bmatrix}$$
$$4 \rightarrow \begin{bmatrix} \; \end{bmatrix}$$
$$5 \rightarrow \begin{bmatrix} \; \end{bmatrix}$$

Nabolister



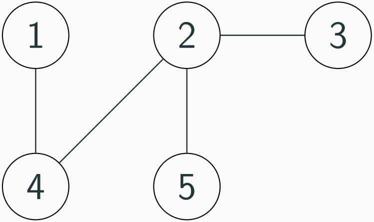
$$1 \rightarrow \begin{bmatrix} 4 \end{bmatrix}$$
$$2 \rightarrow \begin{bmatrix} 3 & 4 & 5 \end{bmatrix}$$
$$3 \rightarrow \begin{bmatrix} 2 \end{bmatrix}$$
$$4 \rightarrow \begin{bmatrix} 1 & 2 \end{bmatrix}$$
$$5 \rightarrow \begin{bmatrix} \; \end{bmatrix}$$

Nabolister



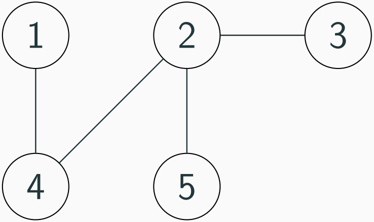
$$1 \to \begin{bmatrix} 4 \end{bmatrix}$$
$$2 \to \begin{bmatrix} 3 & 4 & 5 \end{bmatrix}$$
$$3 \to \begin{bmatrix} 2 \end{bmatrix}$$
$$4 \to \begin{bmatrix} 1 & 2 \end{bmatrix}$$
$$5 \to \begin{bmatrix} \end{bmatrix}$$

Nabolister



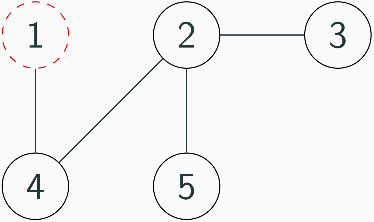
$$1 \to \begin{bmatrix} 4 \end{bmatrix}$$
$$2 \to \begin{bmatrix} 3 & 4 & 5 \end{bmatrix}$$
$$3 \to \begin{bmatrix} 2 \end{bmatrix}$$
$$4 \to \begin{bmatrix} 1 & 2 \end{bmatrix}$$
$$5 \to \begin{bmatrix} 2 \end{bmatrix}$$

Nabomatrise



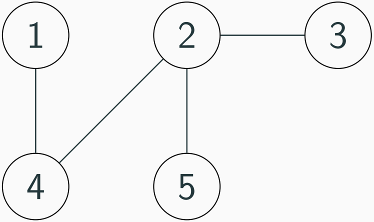
$$\begin{array}{c c} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Nabomatrise



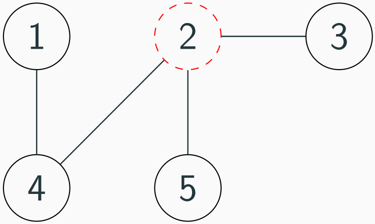
$$
\begin{array}{c c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Nabomatrise



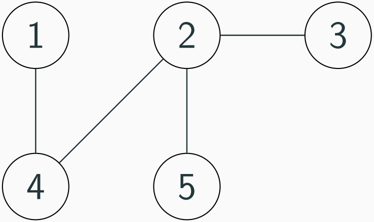
$$
\begin{array}{c c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Nabomatrise



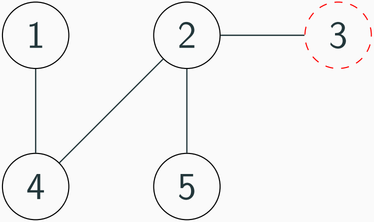
$$
\begin{array}{c c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Nabomatrise



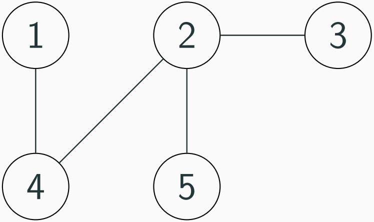
$$
\begin{array}{c c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Nabomatrise



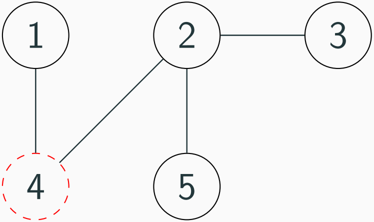
$$\begin{array}{c c} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Nabomatrise



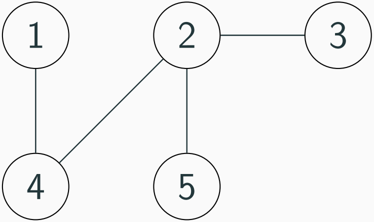
$$\begin{array}{cc} & \begin{array}{ccccc}1 & 2 & 3 & 4 & 5\end{array} \\ \begin{array}{c}1\\2\\3\\4\\5\end{array} & \begin{bmatrix}0 & 0 & 0 & 1 & 0\\0 & 0 & 1 & 1 & 1\\0 & 1 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0\end{bmatrix}\end{array}$$

Nabomatrise



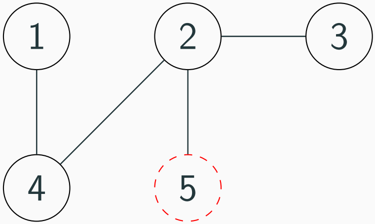
$$
\begin{array}{cc}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Nabomatrise



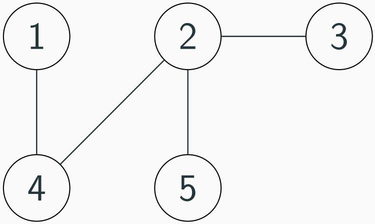
$$\begin{array}{c c} & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Nabomatrise



$$
\begin{array}{c c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Nabomatrise



$$
\begin{array}{c c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Denne ukens pensum

Bredde-først-søk

Bredde-først-søk

Bredde-først-søk (BFS) er en algortime for å traversere grafer, som vil si å besøke noder i en viss rekkefølge.

Bredde-først-søk

Bredde-først-søk (BFS) er en algortime for å traversere grafer, som vil si å besøke noder i en viss rekkefølge.

Den "søker i bredden" ved å plassere noder i en kø, som den deretter velger fra.

Bredde-først-søk

BFS fungerer omtrent slik:

Bredde-først-søk

BFS fungerer omtrent slik:

0. Velg en startnode og legg denne inn i en kø.

Bredde-først-søk

BFS fungerer omtrent slik:

0. Velg en startnode og legg denne inn i en kø.
1. Hent ut noden som er fremst i køen.

Bredde-først-søk

BFS fungerer omtrent slik:

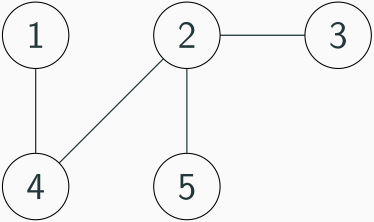
0. Velg en startnode og legg denne inn i en kø.
1. Hent ut noden som er fremst i køen.
2. Legg til nodens naboer (som ikke allerede er besøkt eller i køen) bakerst i køen, og farg noden svart.

Bredde-først-søk

BFS fungerer omtrent slik:

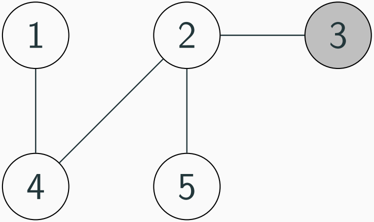
0. Velg en startnode og legg denne inn i en kø.
1. Hent ut noden som er fremst i køen.
2. Legg til nodens naboer (som ikke allerede er besøkt eller i køen) bakerst i køen, og farg noden svart.
3. Gjenta (gå til 1) til køen er tom.

Bredde-først-søk



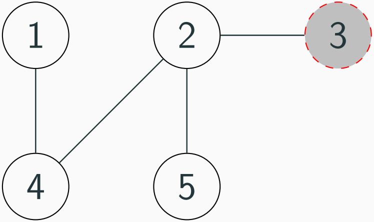
Nodekø:
Besøkt:
Farget svart:

Bredde-først-søk



Nodekø: 3
Besøkt:
Farget svart:

Bredde-først-søk

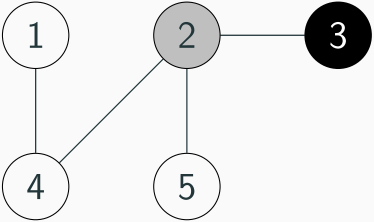


Nodekø:

Besøkt: 3

Farget svart:

Bredde-først-søk

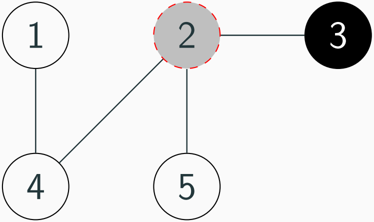


Nodekø: 2

Besøkt: 3

Farget svart: 3

Bredde-først-søk

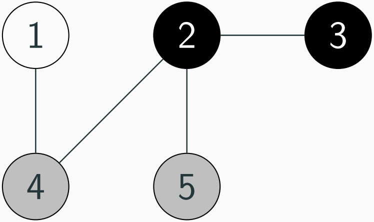


Nodekø:

Besøkt: 3, 2

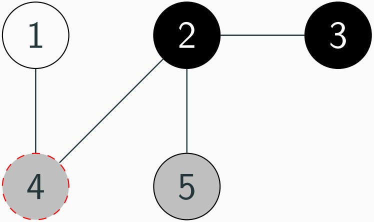
Farget svart: 3

Bredde-først-søk



Nodekø: 4, 5
Besøkt: 3, 2
Farget svart: 3, 2

Bredde-først-søk

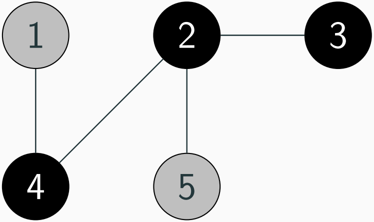


Nodekø: 5

Besøkt: 3, 2, 4

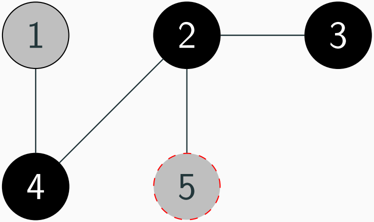
Farget svart: 3, 2

Bredde-først-søk



Nodekø: 5, 1
Besøkt: 3, 2, 4
Farget svart: 3, 2, 4

Bredde-først-søk

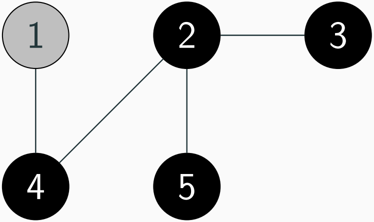


Nodekø: 1

Besøkt: 3, 2, 4, 5

Farget svart: 3, 2, 4

Bredde-først-søk

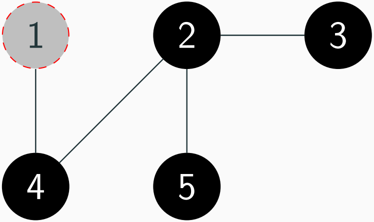


Nodekø: 1

Besøkt: 3, 2, 4, 5

Farget svart: 3, 2, 4, 5

Bredde-først-søk

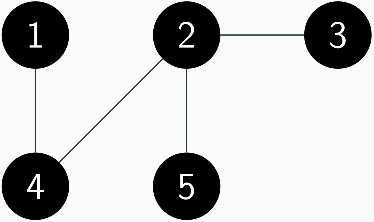


Nodekø:

Besøkt: 3, 2, 4, 5, 1

Farget svart: 3, 2, 4, 5

Bredde-først-søk

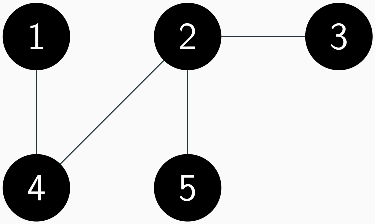


Nodekø:

Besøkt: 3, 2, 4, 5, 1

Farget svart: 3, 2, 4, 5, 1

Bredde-først-søk



Nodekø:

Besøkt: $3, 2, 4, 5, 1$

Farget svart: $3, 2, 4, 5, 1$

Merk at besøkt $=$ farget svart

Denne ukens pensum

Dybde-først-søk

Dybde-først-søk

Dybde-først-søk (DFS) er også en algoritme for graftraversering, og er relativt lik BFS.

Dybde-først-søk

Dybde-først-søk (DFS) er også en algoritme for graftraversering, og er relativt lik BFS.

I motsetning til BFS, kan DFS implementeres både iterativt og rekursivt.

Dybde-først-søk

Den iterative implementasjonen er nesten helt lik BFS.

Dybde-først-søk

Den iterative implementasjonen er nesten helt lik BFS.

Hovedforskjellen er at DFS "søker i dybden" ved å alltid velge den *bakerste* noden i køen, istedenfor den fremste.

Dybde-først-søk

Den iterative implementasjonen er nesten helt lik BFS.

Hovedforskjellen er at DFS "søker i dybden" ved å alltid velge den *bakerste* noden i køen, istedenfor den fremste.

Den bruker altså en stakk istedenfor en (LIFO-)kø.

Dybde-først-søk

Den iterative implementasjonen er nesten helt lik BFS.

Hovedforskjellen er at DFS "søker i dybden" ved å alltid velge den *bakerste* noden i køen, istedenfor den fremste.

Den bruker altså en stakk istedenfor en (LIFO-)kø.

For den rekursive versjonen vil dette si å rekursere så langt som mulig for hver node.

Dybde-først-søk

OBS: I pensumboka er kun den *rekursive* implementasjonen beskrevet.

Dybde-først-søk

OBS: I pensumboka er kun den *rekursive* implementasjonen beskrevet.

Vi kommer derfor til å fokusere på denne her.

Dybde-først-søk

Rekursiv DFS fungerer omtrent slik:

Dybde-først-søk

Rekursiv DFS fungerer omtrent slik:

0. For hver hvite node:

Dybde-først-søk

Rekursiv DFS fungerer omtrent slik:

0. For hver hvite node:
1. Farg noden grå.

Dybde-først-søk

Rekursiv DFS fungerer omtrent slik:

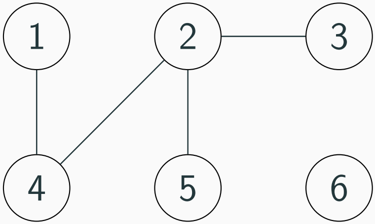
0. For hver hvite node:
1. Farg noden grå.
2. Rekurser for hver hvite nabo (gå til 1 for hver av dem)

Dybde-først-søk

Rekursiv DFS fungerer omtrent slik:

0. For hver hvite node:
1. Farg noden grå.
2. Rekurser for hver hvite nabo (gå til 1 for hver av dem)
3. Farg noden svart.

Dybde-først-søk

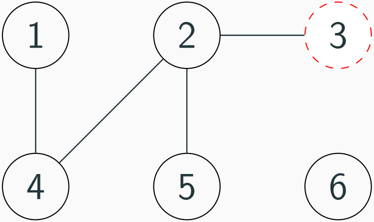


Rekursjonsdybde: 0

Besøkt:

Farget svart:

Dybde-først-søk

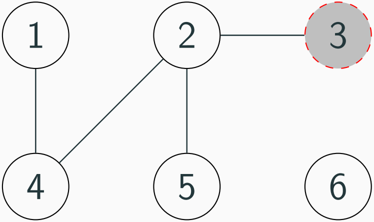


Rekursjonsdybde: 1

Besøkt: 3

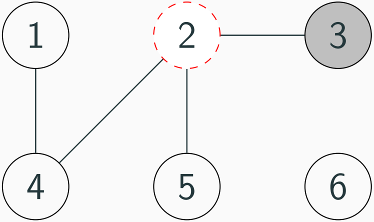
Farget svart:

Dybde-først-søk



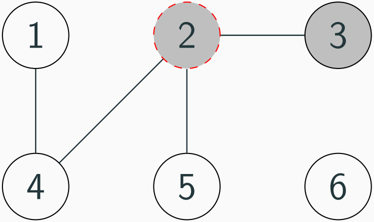
Rekursjonsdybde: 1
Besøkt: 3
Farget svart:

Dybde-først-søk



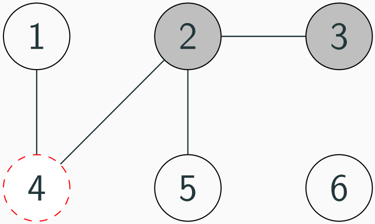
Rekursjonsdybde: 2
Besøkt: 3, 2
Farget svart:

Dybde-først-søk



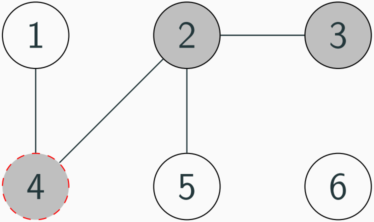
Rekursjonsdybde: 2
Besøkt: 3, 2
Farget svart:

Dybde-først-søk



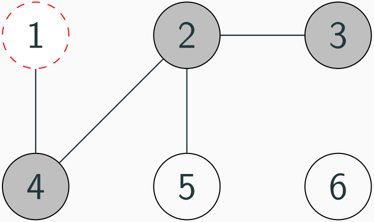
Rekursjonsdybde: 3
Besøkt: 3, 2, 4
Farget svart:

Dybde-først-søk



Rekursjonsdybde: 3
Besøkt: 3, 2, 4
Farget svart:

Dybde-først-søk

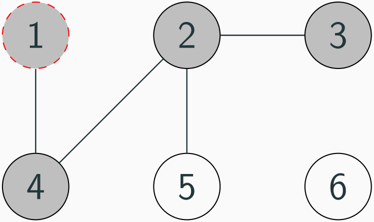


Rekursjonsdybde: 4

Besøkt: 3, 2, 4, 1

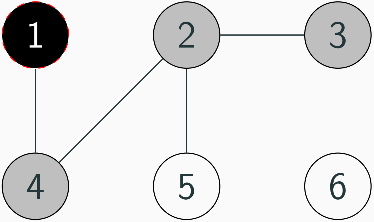
Farget svart:

Dybde-først-søk



Rekursjonsdybde: 4
Besøkt: 3, 2, 4, 1
Farget svart:

Dybde-først-søk

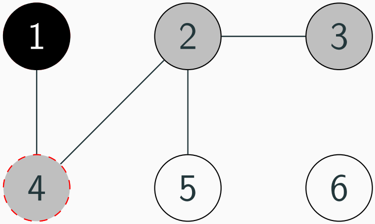


Rekursjonsdybde: 4

Besøkt: 3, 2, 4, 1

Farget svart: 1

Dybde-først-søk

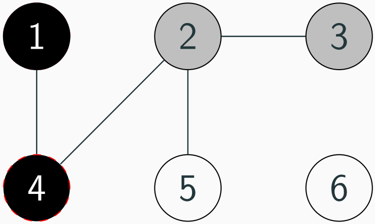


Rekursjonsdybde: 3

Besøkt: 3, 2, 4, 1

Farget svart: 1

Dybde-først-søk

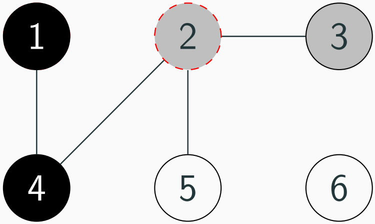


Rekursjonsdybde: 3

Besøkt: 3, 2, 4, 1

Farget svart: 1, 4

Dybde-først-søk

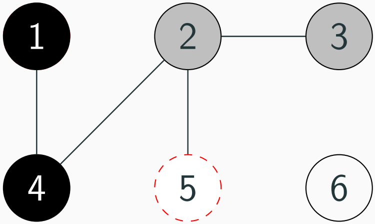


Rekursjonsdybde: 2

Besøkt: 3, 2, 4, 1

Farget svart: 1, 4

Dybde-først-søk

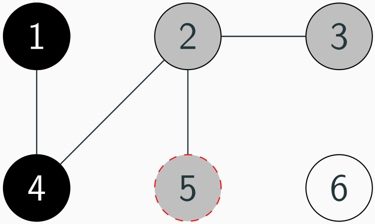


Rekursjonsdybde: 3

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4

Dybde-først-søk

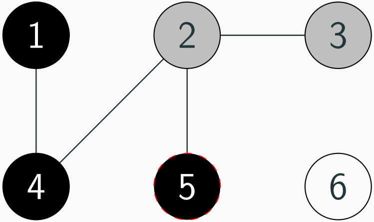


Rekursjonsdybde: 3

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4

Dybde-først-søk

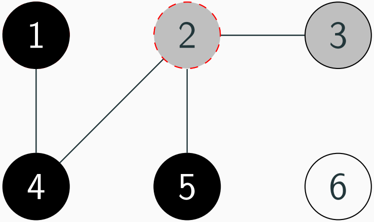


Rekursjonsdybde: 3

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4, 5

Dybde-først-søk

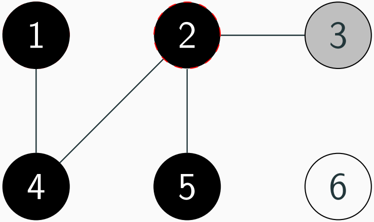


Rekursjonsdybde: 2

Besøkt: 3, 2, 4, 1, 5

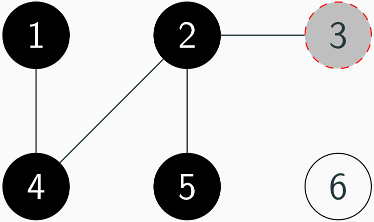
Farget svart: 1, 4, 5

Dybde-først-søk



Rekursjonsdybde: 2
Besøkt: 3, 2, 4, 1, 5
Farget svart: 1, 4, 5, 2

Dybde-først-søk

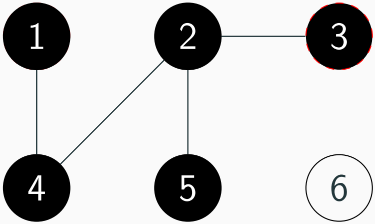


Rekursjonsdybde: 1

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4, 5, 2

Dybde-først-søk

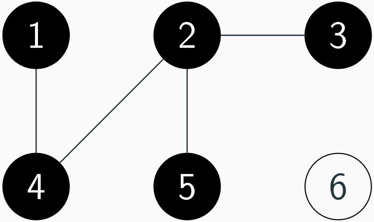


Rekursjonsdybde: 1

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4, 5, 2, 3

Dybde-først-søk

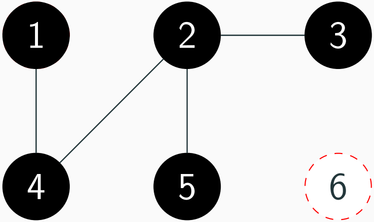


Rekursjonsdybde: 0

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4, 5, 2, 3

Dybde-først-søk

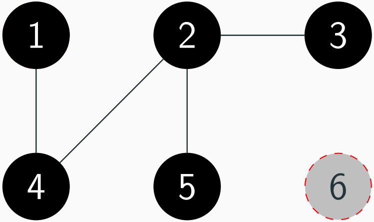


Rekursjonsdybde: 1

Besøkt: 3, 2, 4, 1, 5, 6

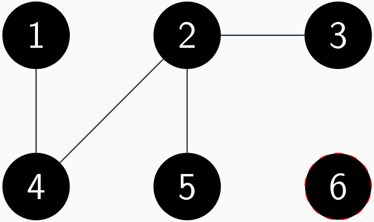
Farget svart: 1, 4, 5, 2, 3

Dybde-først-søk



Rekursjonsdybde: 1
Besøkt: 3, 2, 4, 1, 5, 6
Farget svart: 1, 4, 5, 2, 3

Dybde-først-søk

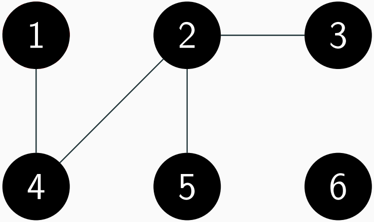


Rekursjonsdybde: 1

Besøkt: 3, 2, 4, 1, 5, 6

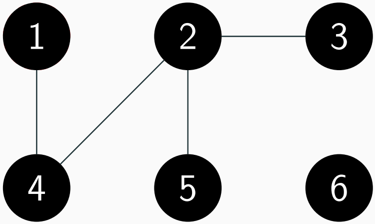
Farget svart: 1, 4, 5, 2, 3, 6

Dybde-først-søk



Rekursjonsdybde: 0
Besøkt: 3, 2, 4, 1, 5, 6
Farget svart: 1, 4, 5, 2, 3, 6

Dybde-først-søk



Rekursjonsdybde: 0
Besøkt: $3, 2, 4, 1, 5, 6$
Farget svart: $1, 4, 5, 2, 3, 6$

Merk at besøkt $\neq$ farget svart

Denne ukens pensum

Topologisk sortering

Topologisk sortering

Topologisk sortering er en måte å ordne/sortere noder i en graf på.

Topologisk sortering

Topologisk sortering er en måte å ordne/sortere noder i en graf på.

Det brukes ofte til å håndtere avhengigheter, slik at ting kjøres i riktig rekkefølge.

Topologisk sortering

Topologisk sortering er en måte å ordne/sortere noder i en graf på.

Det brukes ofte til å håndtere avhengigheter, slik at ting kjøres i riktig rekkefølge.

Topologisk sortering i praksis: `git log --graph`.

Topologisk sortering

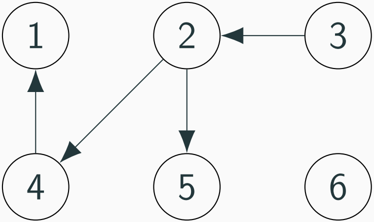
Grafen må være rettet og asyklisk (en *DAG*).

Topologisk sortering

Grafen må være rettet og asyklisk (en *DAG*).

Vi får en topologisk sortering ved å kjøre DFS på grafen og sortere etter når noden sist ble farget svart (siste kommer først).

Topologisk sortering

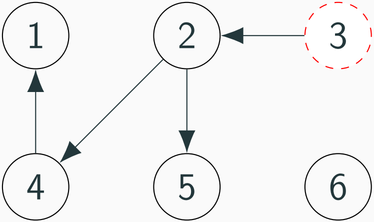


Rekursjonsdybde: 0

Besøkt:

Farget svart:

Topologisk sortering

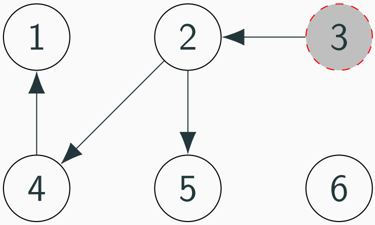


Rekursjonsdybde: 1

Besøkt: 3

Farget svart:

Topologisk sortering

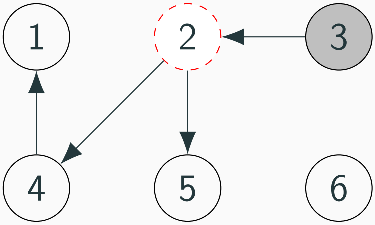


Rekursjonsdybde: 1

Besøkt: 3

Farget svart:

Topologisk sortering

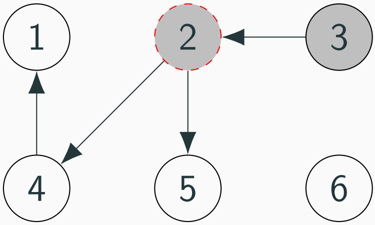


Rekursjonsdybde: 2

Besøkt: 3, 2

Farget svart:

Topologisk sortering

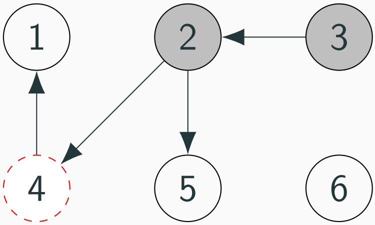


Rekursjonsdybde: 2

Besøkt: 3, 2

Farget svart:

Topologisk sortering

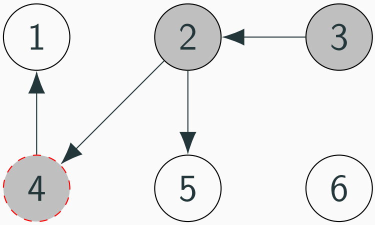


Rekursjonsdybde: 3

Besøkt: 3, 2, 4

Farget svart:

Topologisk sortering

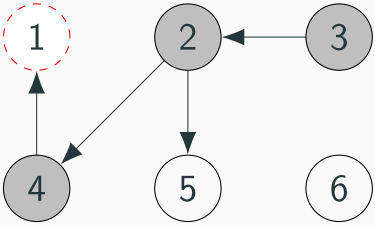


Rekursjonsdybde: 3

Besøkt: 3, 2, 4

Farget svart:

Topologisk sortering

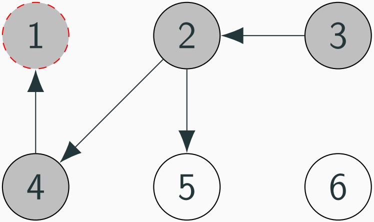


Rekursjonsdybde: 4

Besøkt: 3, 2, 4, 1

Farget svart:

Topologisk sortering

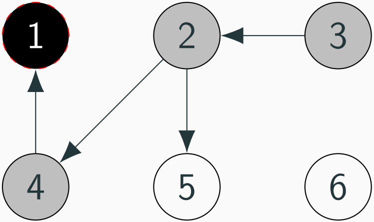


Rekursjonsdybde: 4

Besøkt: 3, 2, 4, 1

Farget svart:

Topologisk sortering

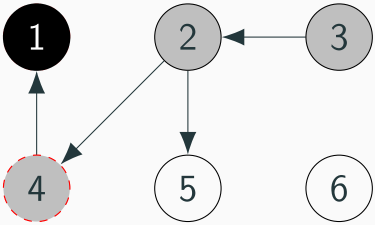


Rekursjonsdybde: 4

Besøkt: 3, 2, 4, 1

Farget svart: 1

Topologisk sortering

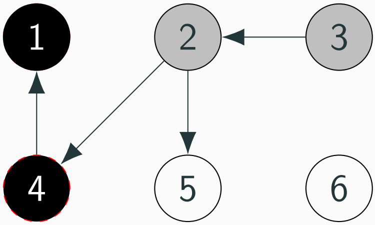


Rekursjonsdybde: 3

Besøkt: 3, 2, 4, 1

Farget svart: 1

Topologisk sortering

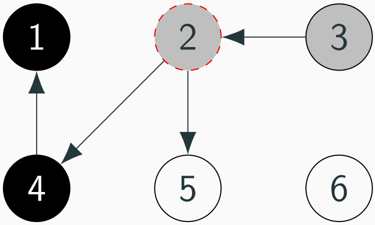


Rekursjonsdybde: 3

Besøkt: 3, 2, 4, 1

Farget svart: 1, 4

Topologisk sortering

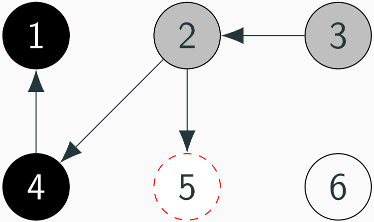


Rekursjonsdybde: 2

Besøkt: 3, 2, 4, 1

Farget svart: 1, 4

Topologisk sortering

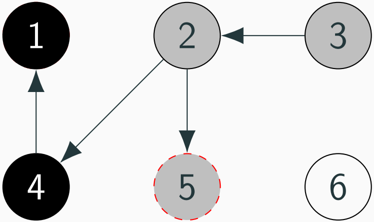


Rekursjonsdybde: 3

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4

Topologisk sortering

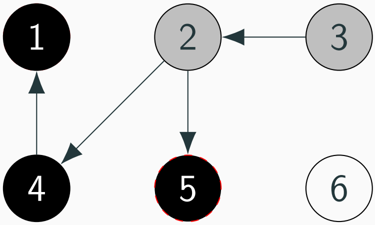


Rekursjonsdybde: 3

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4

Topologisk sortering

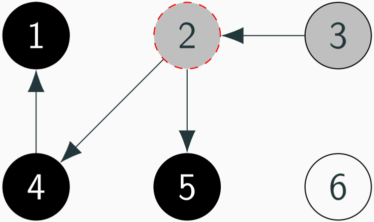


Rekursjonsdybde: 3

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4, 5

Topologisk sortering

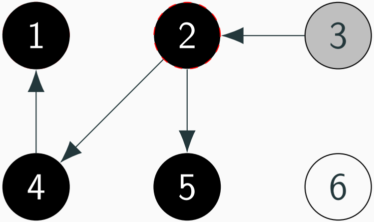


Rekursjonsdybde: 2

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4, 5

Topologisk sortering

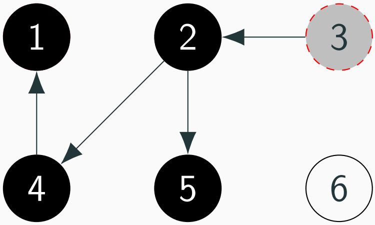


Rekursjonsdybde: 2

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4, 5, 2

Topologisk sortering

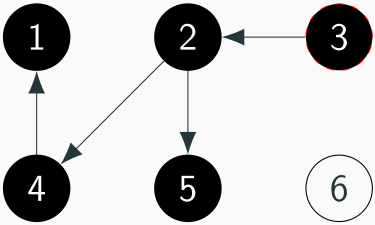


Rekursjonsdybde: 1

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4, 5, 2

Topologisk sortering

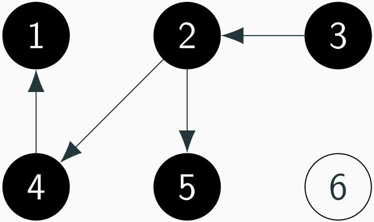


Rekursjonsdybde: 1

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4, 5, 2, 3

Topologisk sortering

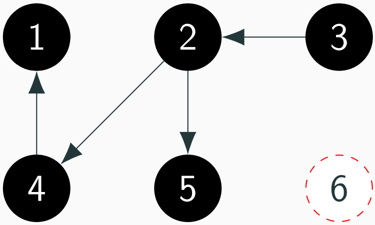


Rekursjonsdybde: 0

Besøkt: 3, 2, 4, 1, 5

Farget svart: 1, 4, 5, 2, 3

Topologisk sortering

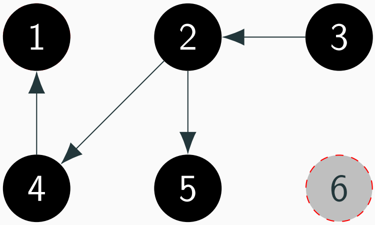


Rekursjonsdybde: 1

Besøkt: 3, 2, 4, 1, 5, 6

Farget svart: 1, 4, 5, 2, 3

Topologisk sortering

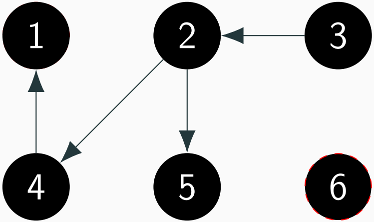


Rekursjonsdybde: 1

Besøkt: 3, 2, 4, 1, 5, 6

Farget svart: 1, 4, 5, 2, 3

Topologisk sortering

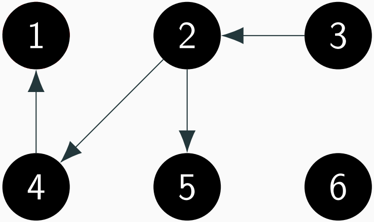


Rekursjonsdybde: 1

Besøkt: 3, 2, 4, 1, 5, 6

Farget svart: 1, 4, 5, 2, 3, 6

Topologisk sortering

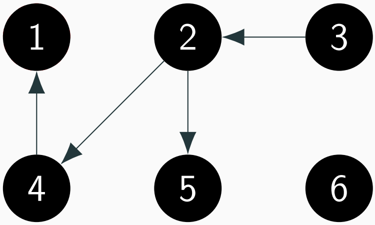


Rekursjonsdybde: 0

Besøkt: 3, 2, 4, 1, 5, 6

Farget svart: 1, 4, 5, 2, 3, 6

Topologisk sortering



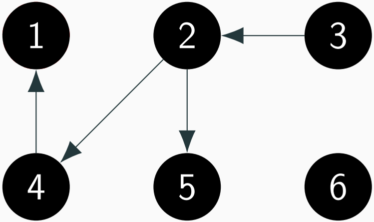
Rekursjonsdybde: 0

Besøkt: $3, 2, 4, 1, 5, 6$

Farget svart: $1, 4, 5, 2, 3, 6$

Reverse(farget svart) $=$ $6, 3, 2, 5, 4, 1 =$ vår topologiske sortering!

Topologisk sortering



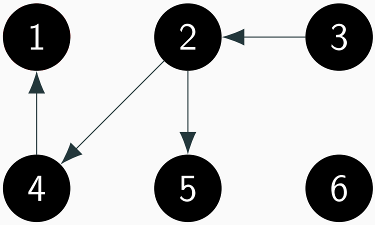
Rekursjonsdybde: 0

Besøkt: $3, 2, 4, 1, 5, 6$

Farget svart: $1, 4, 5, 2, 3, 6$

Reverse(farget svart) $=$ $6, 3, 2, 5, 4, 1 =$ vår topologiske sortering!

Topologisk sortering

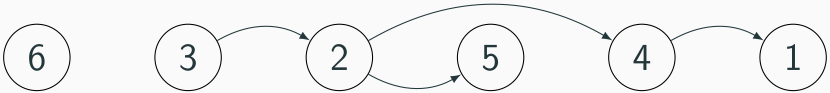


Rekursjonsdybde: 0

Besøkt: $3, 2, 4, 1, 5, 6$

Farget svart: $1, 4, 5, 2, 3, 6$

Reverse(farget svart) $=$ $6, 3, 2, 5, 4, 1 =$ vår topologiske sortering!



Topologisk sortering

Men ...

Topologisk sortering

Men ...

... vi trenger ikke kjøre DFS hvis vi bare skal sjekke om en topologisk sortering er korrekt!

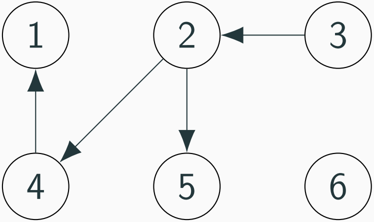
Topologisk sortering

Men …

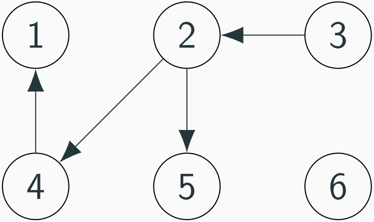
… vi trenger ikke kjøre DFS hvis vi bare skal sjekke om en topologisk sortering er korrekt!

For en gyldig topologisk sortering vil alle nodene ha kanter som går mot høyre.

Topologisk sortering

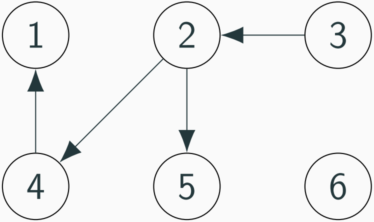


Topologisk sortering



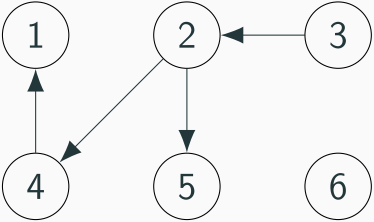
Er $1, 2, 3, 4, 5, 6$ en gyldig topologisk sortering?

Topologisk sortering

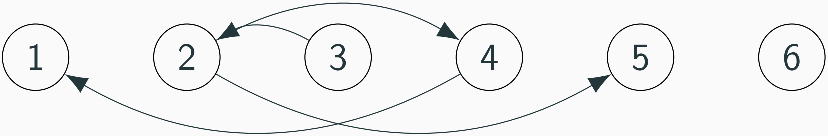


Er $1, 2, 3, 4, 5, 6$ en gyldig topologisk sortering?

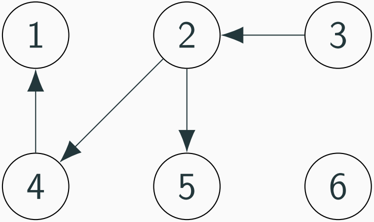
Topologisk sortering



Er $1, 2, 3, 4, 5, 6$ en gyldig topologisk sortering?

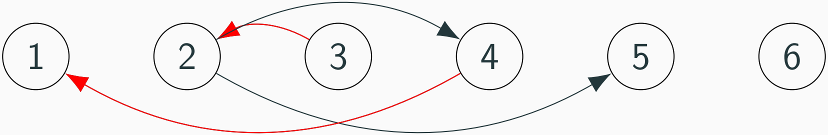


Topologisk sortering

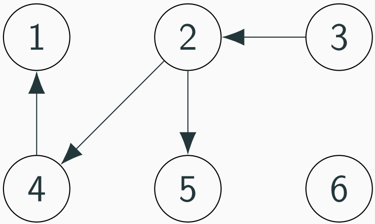


Er 1, 2, 3, 4, 5, 6 en gyldig topologisk sortering?

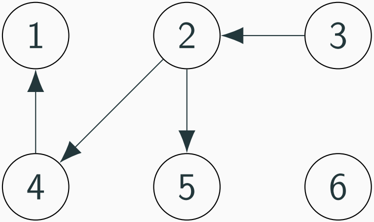
Nei!



Topologisk sortering

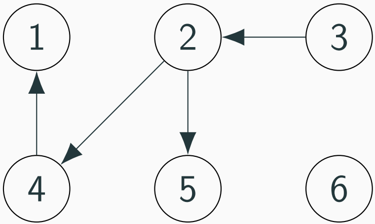


Topologisk sortering



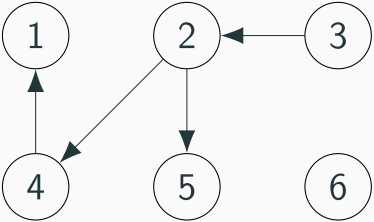
Er $3, 2, 4, 5, 1, 6$ en gyldig topologisk sortering?

Topologisk sortering

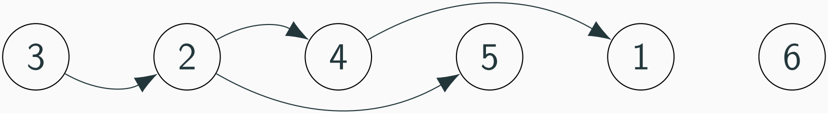


Er $3, 2, 4, 5, 1, 6$ en gyldig topologisk sortering?

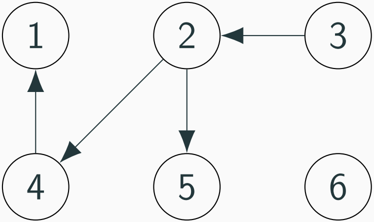
Topologisk sortering



Er $3, 2, 4, 5, 1, 6$ en gyldig topologisk sortering?

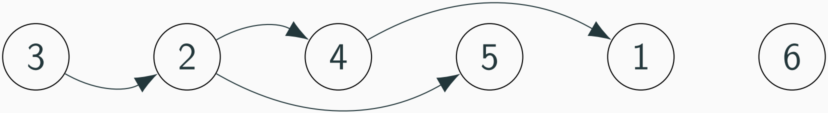


Topologisk sortering

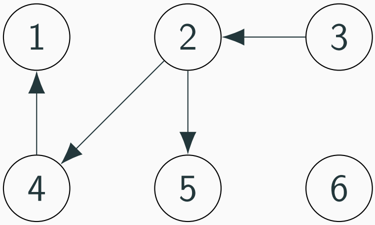


Er $3, 2, 4, 5, 1, 6$ en gyldig topologisk sortering?

Ja!

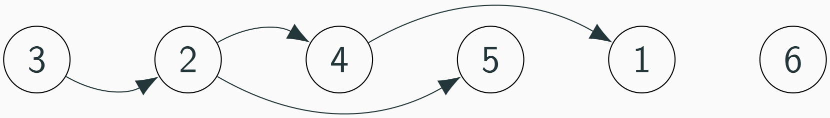


Topologisk sortering



Er $3, 2, 4, 5, 1, 6$ en gyldig topologisk sortering?

Ja! Merk at en topologisk sortering ikke trenger å være unik.



Tips til øving 8, teori

Tips 1

I denne forelesningen gikk vi gjennom BFS og DFS på urettede grafer.

Tips 1

I denne forelesningen gikk vi gjennom BFS og DFS på urettede grafer.

De fungerer helt likt med rettede grafer!

Tips 2

Kompleksitene som bes om i teoriøvingen finnes i pensumboka og/eller på nettet ...

Tips 2

Kompleksitene som bes om i teoriøvingen finnes i pensumboka og/eller på nettet ...

... men prøv å forstå hvorfor de er som de er.

Tips 3

BFS- og DFS-oppgavene spør om hvilken rekkefølge nodene blir *farget svart*.

Tips 3

BFS- og DFS-oppgavene spør om hvilken rekkefølge nodene blir *farget svart*.

Ikke i hvilken rekkefølge de først ble besøkt!

Tips 4

Kantklassifisering i DFS kan være litt tricky ...

Tips 4

Kantklassifisering i DFS kan være litt tricky ...

... men bruk eliminasjonsmetoden.

Tips til øving 8, praksis

Øving 8

Overordnet oppgave: Finn korteste vei mellom to punkter i en labyrint.

Øving 8

Overordnet oppgave: Finn korteste vei mellom to punkter i en labyrint.

Deloppgaver:

Øving 8

Overordnet oppgave: Finn korteste vei mellom to punkter i en labyrint.

Deloppgaver:

1. Gjør om labyrintrepresentasjonen til en grafrepresentasjon.

Øving 8

Overordnet oppgave: Finn korteste vei mellom to punkter i en labyrint.

Deloppgaver:

1. Gjør om labyrintrepresentasjonen til en grafrepresentasjon.
2. Kjør BFS på grafen.

Øving 8

Overordnet oppgave: Finn korteste vei mellom to punkter i en labyrint.

Deloppgaver:

1. Gjør om labyrintrepresentasjonen til en grafrepresentasjon.
2. Kjør BFS på grafen.
3. Regn ut den korteste veien ut ifra BFS-resultatet.

Tips til øving 8, teori

Minst 2 av 3 deloppgaver må være bestått for å bestå øvingen.

Tips til øving 8, teori

Minst 2 av 3 deloppgaver må være bestått for å bestå øvingen.

Deloppgavene kan gjøres uavhengig av hverandre.

Tips til øving 8, teori

Minst 2 av 3 deloppgaver må være bestått for å bestå øvingen.

Deloppgavene kan gjøres uavhengig av hverandre.

Hvis du sitter fast på en oppgave anbefales det å gå til neste oppgave i mellomtiden.

Oppgave 1

Oppgaven er å implementere `mazetonodelist(maze)`.

Oppgave 1

Oppgaven er å implementere `mazetonodelist(maze)`.

Det står allerede en del info i oppgaven, men her kommer enda et eksempel.

Oppgave 1

$maze =$

$$\begin{matrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{matrix}$$

Oppgave 1

$$maze = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 1 & 0 \\ 2 & 1 & 1 & 0 \\ 3 & 0 & 1 & 1 \end{array}$$

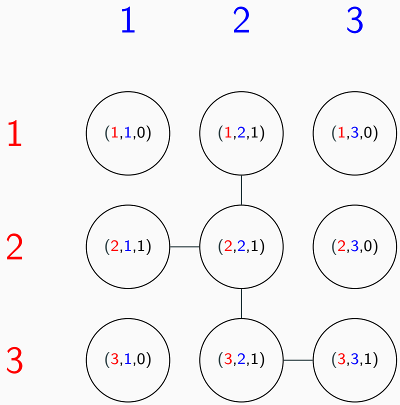
Oppgave 1

nodearray =

	1	2	3
1	(1,1,0)	(1,2,1)	(1,3,0)
2	(2,1,1)	(2,2,1)	(2,3,0)
3	(3,1,0)	(3,2,1)	(3,3,1)

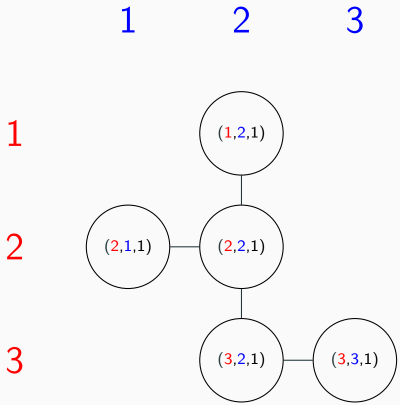
Oppgave 1

$nodearray =$



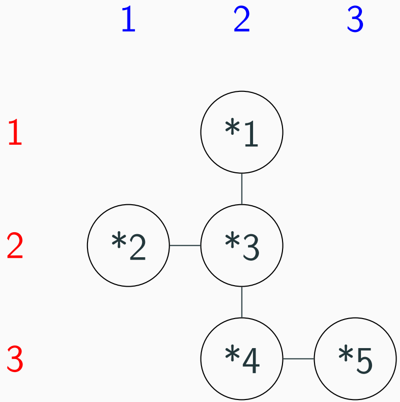
Oppgave 1

nodearray =



Oppgave 1

$nodearray =$



$$mazetonodelist(maze) \rightarrow [*1, *2, *3, *4, *5]$$

Oppgave 2

Standard implementasjon av BFS.

Oppgave 2

Standard implementasjon av BFS.

Se i pensumboka for pseudokode.

Oppgave 3

Oppgaven er å implementere `makepathto(goalnode)`.

Oppgave 3

Oppgaven er å implementere `makepathto(goalnode)`.

Skal returnere en liste med koordinater som utgjør stien fra start til mål.

Oppgave 3

Oppgaven er å implementere `makepathto(goalnode)`.

Skal returnere en liste med koordinater som utgjør stien fra start til mål.

Bruk predecessor til å finne koordinatene i stien, og lagre dem i en liste underveis.

Bonus

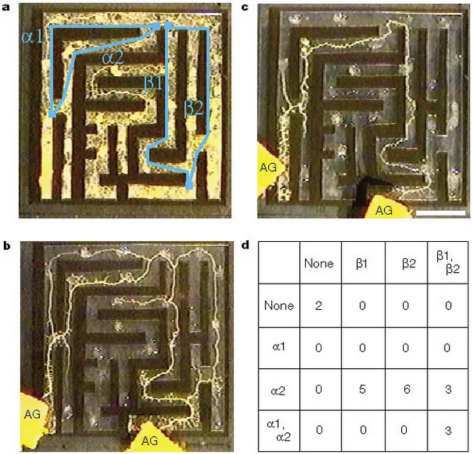
Bonus

Denne mystiske organismen har 720 kjønn og lærer uten hjerne

https://www.nrk.no/urix/viser-frem-mystisk-organisme-med-720-kjonn-1.14748973

Bonus

Figure 1: Maze-solving by *Physarum polycephalum*.



	None	β1	β2	β1, β2
None	2	0	0	0
α1	0	0	0	0
α2	0	5	6	3
α1, α2	0	0	0	3

https://www.nature.com/articles/35035159

Takk for i dag, og lykke til med øvingen!