Kræsjkurs MNF130

Steinar Simonnes og Lukas Schramm

Institutt for informatikk
Universitetet i Bergen

12 Mai 2022

Agenda

Last meg ned

Figure: https://tinyurl.com/mnf130v22

Divisjon og Modulær aritmetikk

Delelighet $a|b$ (a deler b)

- a kan dele b uten rest
- $a|b$ er det samme som $\frac{b}{a} = c$ eller $b = a \cdot c$ med c som heltall
  Eksempel: $3|12$ eller $\frac{12}{3} = 4$ eller $12 = 3 \cdot 4$

Modulo (Klokkearitmetikk)

- $a \, mod \, b$ gir ut resten av heltall divisjon av $\frac{a}{b}$ ($a\%b$ i programmeringsspråk)
- $a \, mod \, b \; = \; r$ kalles *remainder*
  Eksempel: $17 \, mod \, 5 \; = 2$ fordi $17 = 3 \cdot 5 + 2$

Algoritme for divisjon /modulo

- $d = q \cdot a + r$ med
- $q = \left\lfloor \frac{d}{a} \right\rfloor$ og $r = d \, mod \, a$
- Eksempel: $q = \left\lfloor \frac{17}{5} \right\rfloor = \lfloor 3,4 \rfloor = 3$
- $17 = 3 \cdot 5 + r \iff 17 = 15 + r \iff r = 2$

```
def rest(d, a):                  # d = 17, a = 5
   q = floor(d/a)                # q = floor(17/5 = 3
   r = d-q*a                     # r = 17-3*5 = 2
   print(f"{d}={q}*{a}+{r}")       # 17 = 3*5+2
   # r=d\%a kan også brukes
```

Modulo regneregler

Kongruens $\equiv$

- $a \equiv b \,(mod\, m)$: a og b kongruent i forhold til mod m
- $a \equiv b \,(mod\, m)$ betyr $a\, mod\, m = b\, mod\, m$
- Eksempel: $8 \equiv 3 \,(mod\, 5)$ betyr $8\, mod\, 5 = 3 = 3\, mod\, 5$

- Addisjon: $(a + b)\, mod\, m = (a\, mod\ m + b\, mod\, m)\, mod\, m$
- $(8 + 21)\, mod\, 6 = (8\, mod\ 6 + 21\, mod\, 6)\, mod\, 6$
- Multiplikasjon: $(a \cdot b)\, mod\, m = (a\, mod\ m \cdot b\, mod\, m)\, mod\, m$
- $(8 \cdot 21)\, mod\, 6 = (8\, mod\ 6 \cdot 21\, mod\, 6)\, mod\, 6$

Eksempel

- $x \equiv 3\,(mod\,5)$
- $y \equiv 4\,(mod\,5)$
- Finn løsningen: $(57 \cdot x^3)\,mod\,5$

$$x \equiv 3\,(mod\,5) \rightarrow x = 3$$
$$(57 \cdot x^3)\,mod\,5 = (57\,mod\,5) \cdot (x\,mod\,5) \cdot (x\,mod\,5) \cdot (x\,mod\,5)\,mod\,5$$
$$= (57\,mod\,5) \cdot (3\,mod\,5) \cdot (3\,mod\,5) \cdot (3\,mod\,5)\,mod\,5$$
$$= (2 \cdot 3 \cdot 3 \cdot 3)\,mod\,5 = 54\,mod\,5 = 4$$

Eksempel

- $x \equiv 3\,(mod\,5)$
- $y \equiv 4\,(mod\,5)$
- Finn løsningen: $(3 \cdot x + 2 \cdot y^2)\,mod\,5$

$$x \equiv 3\,(mod\,5) \rightarrow x = 3$$
$$y \equiv 4\,(mod\,5) \rightarrow y = 4$$
$$(3 \cdot x)\,mod\,5 = (3\,mod\,5) \cdot (x\,mod\,5)\,mod\,5 = (3\,mod\,5) \cdot (3\,mod\,5)\,mod\,5 = 9\,mod\,5 = 4$$
$$(2 \cdot y^2)\,mod\,5 = (2\,mod\,5) \cdot (y\,mod\,5) \cdot (y\,mod\,5)\,mod\,5 =$$
$$(2\,mod\,5) \cdot (4\,mod\,5) \cdot (4\,mod\,5)\,mod\,5 = (2 \cdot 4 \cdot 4)\,mod\,5 = 32\,mod\,5 = 2$$
$$(3 \cdot x + 2 \cdot y^2)\,mod\,5 = ((3 \cdot x)\,mod\,5 + (2 \cdot y^2))\,mod\,5 = (4 + 2)\,mod\,5 = 6\,mod\,5 = 1$$

Tallsystem

En representasjon av tall med forskjellige tegner med en base

Navn	Tall	5	11	34
Desimal (b=10)	0-9	5	11	34
Binær (b=2)	0-1	101	1011	100010
Octal (b=8)	0-7	5	13	42
Hexadesimal (b=16)	0-9,a-f	5	D	22
base=13	0-9,a-c	5	B	28

Table: Eksempler på forskjellige tallsystemer

Desimal til base b

```
def dec_to_base(n, b):
    output = ""
    while n != 0:
        next_digit = n%b
        n = n//b
        output = str(next_digit) + output
        print(f"n: {n}, bin: {next_digit}")
    return output

print(dec_to_base(22, 2))
```

n	nextDigit	output
22		0
11	0	0
5	1	10
2	1	110
1	0	0110
0	1	10110

Table: Eksempel for $dec\_to\_base(22, 2)$

Base b til desimal

```python
def base_to_dec(n, b):
    sum = 0; idx = 0
    while n != 0:
        to_add = (b**idx * (n%b))
        sum += to_add
        n = n // 10
        idx += 1
        print(f"Add: {to_add}, n: {n}, sum: {sum}")
    return sum

print(base_to_dec(10110, 2))
```

toAdd	n	sum
	10110	0
0	1011	0
2	101	2
4	10	6
0	1	6
16	0	22

Table: Eksempel for $base\_to\_dec(10110, 2)$

Primtall

Et tall som bare kan deles av seg selv og 1
Eksempler: 2,3,5,7,11,13,...

Greatest common divisor (største felles faktor)

$gcd(a, b)$:=det største tallet som deler både a og b
Eksempel: $gcd(4, 6) = 2$
Co-prime: a og b er co-prime dersom $gcd(a, b) = 1$

Least common multiple

$lcm(a, b) :=$ det minste tallet som kan deles av både a og b
Eksempel: $lcm(4, 6) = 12$
hvis $gcd(a, b) = 1 \rightarrow lcm(a, b) = a \cdot b$

Euklids algoritme

```
def gcd(a, b):
    while b > 0:
        q = a//b        # quotient
        r = a-q*b       # resten
        a = b
        b = r
    return a
```

a	b	q	r
28	12	2	4
12	4	3	0
4	0		

Table: Eksempel for $gcd(28, 12)$

Extended Euklids algoritme

Regner ut to parameter s og t slik at $gcd(a, b)$ kan skrives som linærkombinasjon
$gcd(a, b) = s \cdot a + t \cdot b$
$gcd(12, 28) = 4 = -2 \cdot 12 + 1 \cdot 28$

Kan brukes for å finne multiplikativ inverse
Multiplikativ inverse finnes dersom $g(a, b) = 1$

Finne multiplicative inverse for a med $mod\, m$

- Funker bare dersom $gcd(a, m) = 1$
- Regn ut linærkombinasjon $gcd(a, b) = s \cdot a + t \cdot b$ med gcd
- $a \cdot x \equiv 1(mod\, m)$ er multiplicative inverse

Extended Euklids algoritme

```
def gcdExtended(a, b):
    if a == 0:                  # basis
            return b, 0, 1

    # rekursjon
    gcd, x1, y1 = gcdExtended(b%a, a)
    x = y1 - (b//a) * x1
    y = x1

    return gcd, x, y
```

Call		Rekursjon				
a	b	gcd	x1	x2	x	y
12	28	4	1	0	-2	1
4	12	4	0	1	1	0
0	4	4	0	1		

Table: Eksempel for $gcd(12, 18)$

Eksempel Multiplicate Inverse

- Hva er multiplicative inverse av $5 \, mod \, 13$?
- $gcd(a, m) = gcd(5, 13) = 1 \rightarrow$ har multiplicative inverse
- Linærkombinasjon fra gcd: $-5 \cdot 5 + 13 \cdot 2 = 1$
- $a = -5$
- Hvilket tall mellom 0 og 12 har samme kongruensklasse $mod \, 13$?
- $13 - 5 = 8$ er multiplicative inverse til 5 for $mod \, 13$

Symmetrisk og asymmetrisk kryptografi

Symmetrisk kryptografi

- Det finnes bare *én* nøkkel, som begge personer bruker
- Brukes for både kryptering og dekryptering
- Eksempel: Caesar – $f(c) = (c + key)\%mod26$

Asymmetrisk kryptografi

- Hver person har *to* nøkler: Privat og offentlig
- Kryptering med offentlig nøkkel av den andre personen
- Dekryptering med privat nøkkel
- Eksempel: RSA

RSA

- Asymmetrisk kryptering med to nøkler for hver deltaker
- Kryptering
  - Offentlig nøkkel for kryptering
  - Privat nøkkel for dekryptering
- Digitale sertifikater/ signaturer
  - Privat nøkkel for signering
  - Offentlig nøkkel for verifisering

Instruksjon	Eksempel
Velg to primtall p, q	$p = 7, q = 13$
Regn ut $n = p \cdot q$	$7 \cdot 13 = 91$
Regn ut $\phi(n) = (p-1) \cdot (q-1)$	$\phi(n) = 6 \cdot 12 = 72$
Velg e med $2 < e < \phi(n)$ og $gcd(e, \phi(n)) = 1$	23, $gcd(72, 23) = 1$
Finn $d = e^{-1} (mod\, \phi(n))$ med EEA	$d = 23^{-1} (mod\, 72)$
Lineærkombinasjon	$-25 \cdot 23 + 8 \cdot 72 = 1$
	$a = -25,\ 72 - 25 = 47,\ d = 47$
Kryptering av blokk M	$M = 42$
$C = M^e (mod\, n)$	$42^{23} (mod\, 91) = 35$
Dekryptering av blokk C	$C = 35$
$M = C^d (mod\, n)$	$35^{47} (mod\, 91) = 42$

Table: Hvordan brukes RSA?

Counting

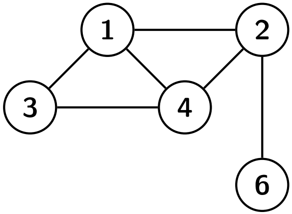
TODO: Fill with content @Lukas Basics of Counting Pigeonhole Principle Permutations and Combinations Binomial conefficients and identities

TODO: Fill with content @Lukas Bayes theorem, probabilitiy theory discrete probability expected value and variance

Graf $G(V, E)$

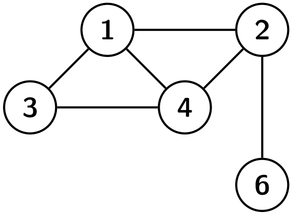
En Graf G er en tuple med en set av noder V og en set av kanter (edges) E

- Noder: $V = 1, 2, 3, 4, 6$



Graf $G(V, E)$

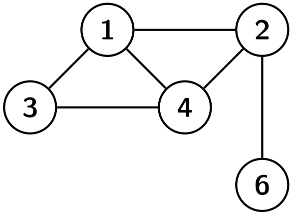
En Graf G er en tuple med en set av noder V og en set av kanter (edges) E



- Noder: $V = 1, 2, 3, 4, 6$
- Kanter: $E = (1, 2), (1, 4), (3, 4), (2, 4), (2, 6), (1, 3)$

Graf $G(V, E)$

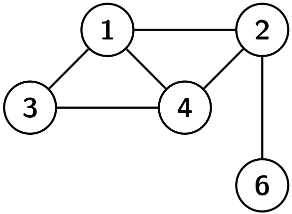
En Graf G er en tuple med en set av noder V og en set av kanter (edges) E



- Noder: $V = 1, 2, 3, 4, 6$
- Kanter: $E = (1,2), (1,4), (3,4), (2,4), (2,6), (1,3)$
- Path: Vei fra A til B
  Eksempel: $Path(1,6) = (1,2,6)$

Graf $G(V, E)$

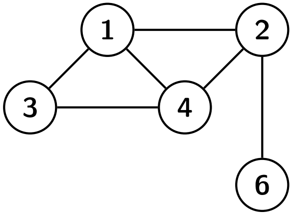
En Graf G er en tuple med en set av noder V og en set av kanter (edges) E



- Noder: $V = 1, 2, 3, 4, 6$
- Kanter: $E = (1,2), (1,4), (3,4), (2,4), (2,6), (1,3)$
- Path: Vei fra A til B
  Eksempel: $Path(1,6) = (1,2,6)$
- Cycle: En path med samme start og slutt
  Eksempel: $(1,3,4,1)$, $(1,2,4,3,1)$

Graf $G(V, E)$

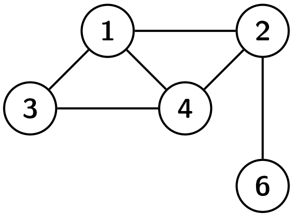
En Graf G er en tuple med en set av noder V og en set av kanter (edges) E



- Noder: $V = 1, 2, 3, 4, 6$
- Kanter: $E = (1, 2), (1, 4), (3, 4), (2, 4), (2, 6), (1, 3)$
- Path: Vei fra A til B
  Eksempel: $Path(1, 6) = (1, 2, 6)$
- Cycle: En path med samme start og slutt
  Eksempel: $(1, 3, 4, 1)$, $(1, 2, 4, 3, 1)$
- Naboer: Set of noder som har en kante til en node
  Eksempel: $N(4) = 1, 2, 3$, $N(6) = 2$

Graf $G(V, E)$

En Graf G er en tuple med en set av noder V og en set av kanter (edges) E



- Noder: $V = 1, 2, 3, 4, 6$
- Kanter: $E = (1, 2), (1, 4), (3, 4), (2, 4), (2, 6), (1, 3)$
- Path: Vei fra A til B
  Eksempel: $Path(1, 6) = (1, 2, 6)$
- Cycle: En path med samme start og slutt
  Eksempel: $(1, 3, 4, 1)$, $(1, 2, 4, 3, 1)$
- Naboer: Set of noder som har en kante til en node
  Eksempel: $N(4) = 1, 2, 3$, $N(6) = 2$
- Degree: Antall naboer av en node
  Eksempel: $deg(4) = 3$, $deg(6) = 1$

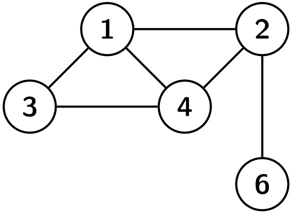


Figure: Urettet graf (undirected)

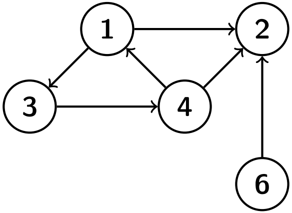


Figure: Rettet graf (directed)

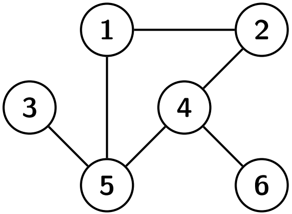
- $deg(4) = 3$

- $deg^{-}(4) = 1$ (ingoing)
- $deg^{+}(4) = 2$ (outgoing)

Bipartite graf $G(V, A, B)$

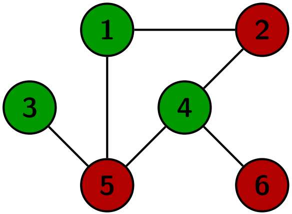
Set av nodene er delt i to sets A, B der alle kanter $v \in V$ går fra en node i A til en node i B

Grafen kan farges i to farger med ingen to nabonoder i samme farge

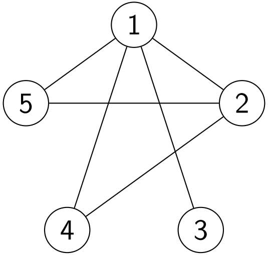


Bipartite graf $G(V, A, B)$

Set av nodene er delt i to sets A, B der alle kanter $v \in V$ går fra en node i A til en node i B
Grafen kan farges i to farger med ingen to nabonoder i samme farge



Representasjon



	1	2	3	4	5
1	0	1	1	1	1
2	1	0	0	1	1
3	1	0	0	0	0
4	1	1	0	0	0
5	1	1	0	0	0

Table: Adjacency matrix (directed)

Node	Neighbours
1	2,3,4,5
2	1,4,5
3	1
4	1,2
5	1,2

Table: Adjacency list (directed)

Tre $G(V, E)$

Et tre er en *connected*, *undirected* graph der ingen cycles eksisterer.

Forest $G(V, E)$

En mengde av trær som ikke er tilknyttet med hverandre.

Rooted tre $G(V, E)$

Et tre med en root node.

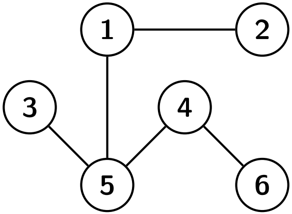


Figure: Et tre

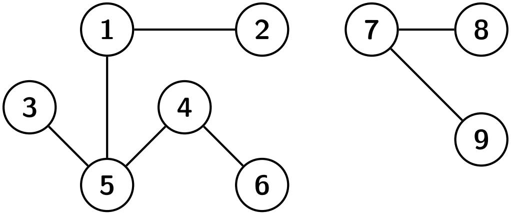


Figure: En skog (forest)

Rooted trees

binary (m-ary) tre $G(V, E)$

Et tre med en root node der alle interne noder har eksakt to (m) barn.

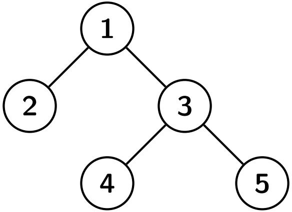


Figure: binary tre

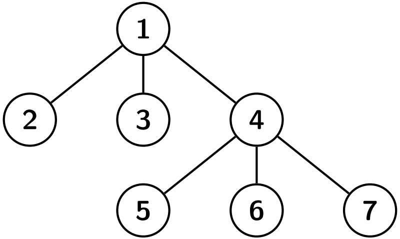


Figure: 3-ary tre

Egenskaper

- Et tre med n noder har $n-1$ kanter

Noder	Interne noder	Leaves
n	$i=(n-1)/m$	$l=((m-1)\cdot n+1)/m$
$n=m\cdot i+1$	i	$l=(m-1)\cdot i+1$
$n=(m\cdot l-1)/(m-1)$	$i=(l-1)/(m-1)$	l

Table: Regne ut antall noder for fulle m-any trær

Eksempel

Et kjedebrev starter med en person som sender et brev til fem andre mennesker. Hver person som får et brev sender den enten videre til fem andre eller stopper å sende ting videre.

Gå ut ifra at 10.000 personer sender brevet videre og ingen får brevet to ganger.

(1) Hvor mange personer fikk et brev?

(2) Hvor mange sendte ikke brevet videre?

Eksempel

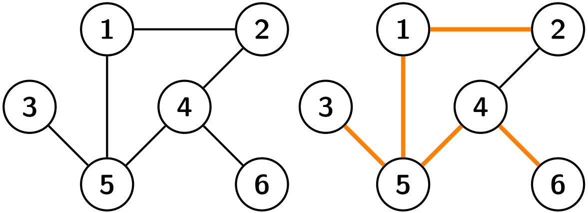
Et kjedebrev starter med en person som sender et brev til fem andre mennesker. Hver person som får et brev sender den enten videre til fem andre eller stopper å sende ting videre.
Gå ut ifra at 10.000 personer sender brevet videre og ingen får brevet to ganger.
(1) Hvor mange personer fikk et brev?
(2) Hvor mange sendte ikke brevet videre?

- Folk som sender videre: Interne noder $i = 10000$
- Folk som ikke sender videre: Leaves $l = (5-1) \cdot 10000 + 1 = 40001$
- Folk som fikk et brev: Noder $n = 5 \cdot 10000 + 1 = 50001$

Spanning Trees

Spanning Trees

Et Spanning Tree for en graf er et tree som besøker alle noder, men ikke lager cycles.



Algoritmer for Spanning Trees

Finne *en* Spanning Tree

- Breadth-first search (BFS)
- Depth-first search (DFS)

Finne *en* Minimum Spanning Tree

- Prims algoritme
- Kruskals algoritme

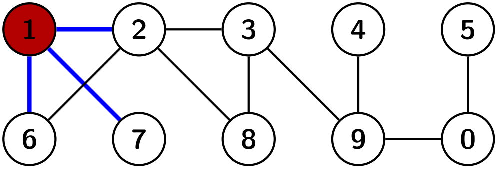
BFS og DFS

- Begge to går gjennom grafen fra en startnode
- I hver runde går man videre til naboene til en node
- Forskjell: BFS bruker kø, DFS stack
- BFS: Rekkefølgen noder blir markert er rekkefølgen man går gjennom grafen
- $\rightarrow$ Bredden blir utforsket før
- DFS: Første noder som blir markiert er siste man ser på
- $\rightarrow$ Algoritmen søker dypt først

Breadth-first search (BFS)

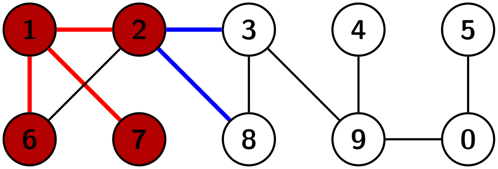
```
def bfs(graph):
    visited = [node] # alle besøkte noder
    queue = [node]   # køen

    while queue:     # så lenge noder er igjen
        m = queue.pop(0) # neste node
        print(f"Visited: {m}")
        for neighbour in graph.neighbours(m): # gå gjennom naboer
            if neighbour not in visited:        # hvis ikke sett før
                visited += neighbour            # add til visited og kø
                queue += neighbour
```

Eksempel BFS



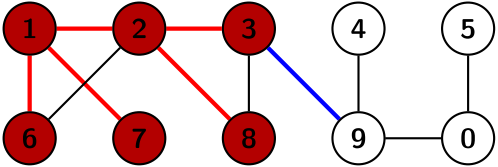
Queue: 2 6 7

Eksempel BFS



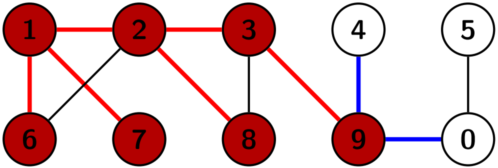
Queue: ~~2~~ ~~6~~ ~~7~~ 3 8

Eksempel BFS



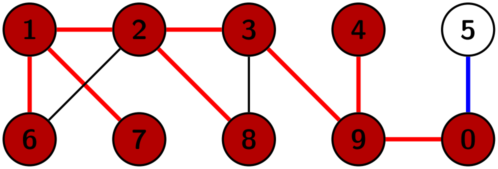
Queue: ~~2~~ ~~6~~ ~~7~~ ~~3~~ ~~8~~ 9

Eksempel BFS



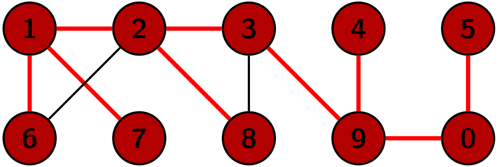
Queue: $\cancel{2}$ $\cancel{6}$ $\cancel{7}$ $\cancel{3}$ $\cancel{8}$ $\cancel{9}$ 4 0

Eksempel BFS



Queue: $\cancel{2}$ $\cancel{6}$ $\cancel{7}$ $\cancel{3}$ $\cancel{8}$ $\cancel{9}$ $\cancel{4}$ $\cancel{0}$ 5

Eksempel BFS

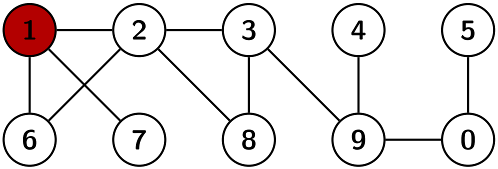


Queue: $\cancel{2}$ $\cancel{6}$ $\cancel{7}$ $\cancel{3}$ $\cancel{8}$ $\cancel{9}$ $\cancel{4}$ $\cancel{0}$ $\cancel{5}$

Depth-first search (DFS)

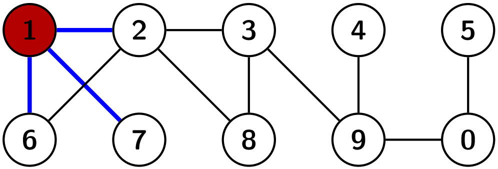
```
visited = []          # alle besøkte noder

def dfs(visited, graph, node):
   if node not in visited:      # hvis noden ikke er besøkt
      print(f"Visited: {node}") # markere som besøkt
      visited += node

      for neighbour in graph.neighbours(node): # gå gjennom naboer
         dfs(visited, graph, neighbour)        # rekursiv call
```

Eksempel DFS



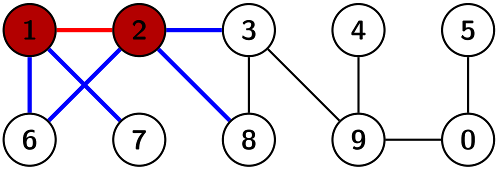
Stack:

Eksempel DFS



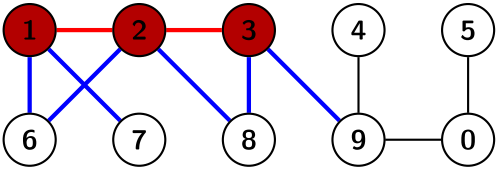
Stack: 7 6 2

Eksempel DFS



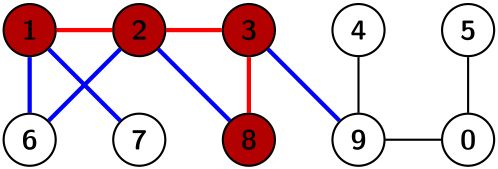
Stack: 7 6 $\cancel{2}$ 6 8 3

Eksempel DFS



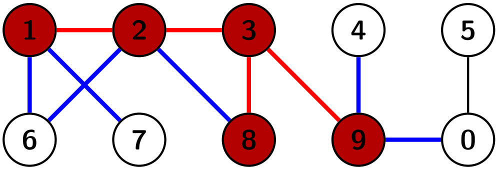
Stack: 7 6 ~~2~~ 6 8 ~~3~~ 9 8

Eksempel DFS



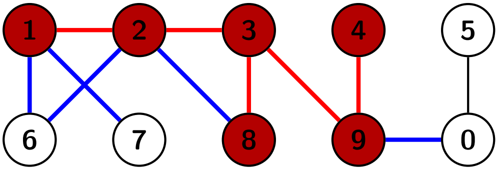
Stack: 7 6 ~~2~~ 6 8 ~~3~~ 9 ~~8~~

Eksempel DFS



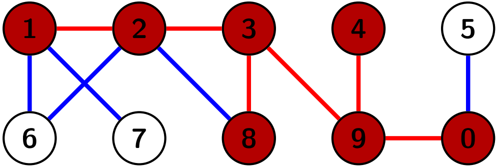
Stack: 7 6 $\cancel{2}$ 6 8 $\cancel{3}$ $\cancel{9}$ $\cancel{8}$ 0 4

Eksempel DFS



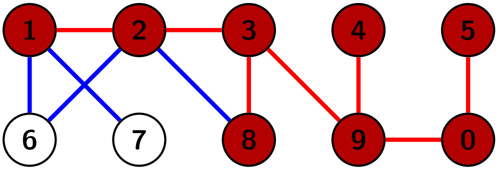
Stack: 7 6 $\cancel{2}$ 6 8 $\cancel{3}$ $\cancel{9}$ $\cancel{8}$ 0 $\cancel{4}$

Eksempel DFS



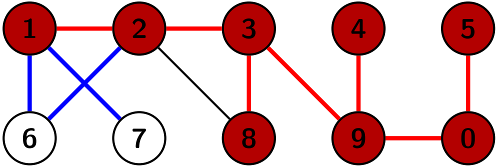
Stack: 7 6 $\cancel{2}$ 6 8 $\cancel{3}$ $\cancel{9}$ $\cancel{8}$ $\cancel{0}$ $\cancel{4}$ 5

Eksempel DFS



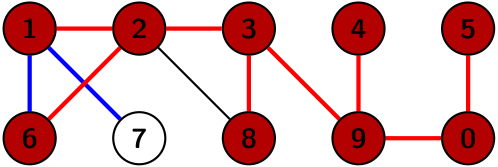
Stack: 7 6 $\cancel{2}$ 6 8 $\cancel{3}$ $\cancel{9}$ $\cancel{8}$ $\cancel{0}$ $\cancel{4}$ $\cancel{5}$

Eksempel DFS



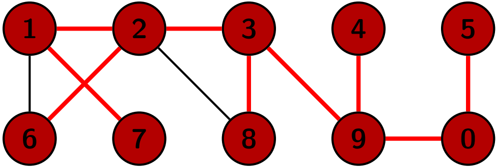
Stack: 7 6 ~~2~~ 6 ~~8~~ ~~3~~ ~~9~~ ~~8~~ ~~0~~ ~~4~~ ~~5~~

Eksempel DFS



Stack: 7 6 ~~2~~ ~~6~~ ~~8~~ ~~3~~ ~~9~~ ~~8~~ ~~0~~ ~~4~~ ~~5~~

Eksempel DFS



Stack: $\cancel{7}$ $\cancel{6}$ $\cancel{2}$ $\cancel{6}$ $\cancel{8}$ $\cancel{3}$ $\cancel{9}$ $\cancel{8}$ $\cancel{0}$ $\cancel{4}$ $\cancel{5}$

Minimum Spanning Trees

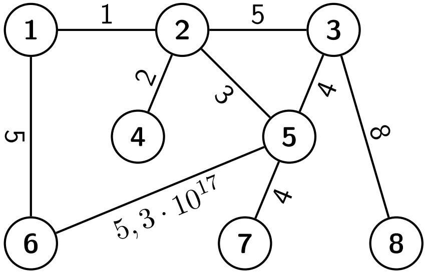
Minimum Spanning Trees

Et Minimum Spanning Tree for en graf er en Spanning Tree der summen av vektene (weights) er minimal.

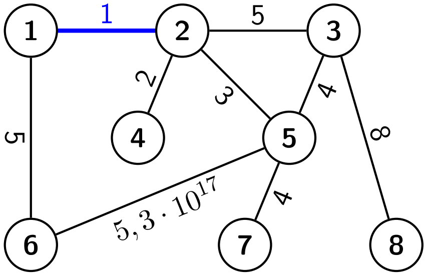
Kruskals algoritme

1. Sorter alle kanter etter sine vekter
2. Velg kanten med minst vekt som ikke ble valgt før. Hvis det nå blir en syklus, ignorer kanten. Hvis ikke, legg kanten til treet.
3. Repeter (2) så lenge til det er (n-1) kanter / alle noder er knyttet sammen.

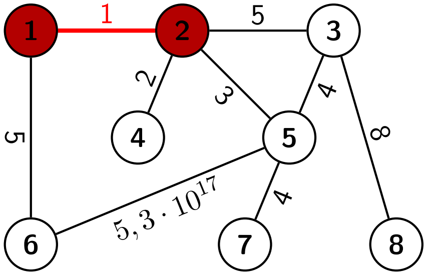
Eksempel Kruskal



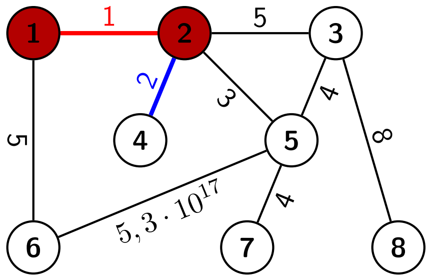
Eksempel Kruskal



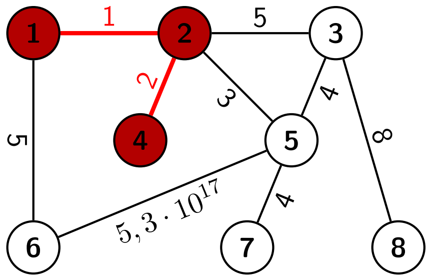
Eksempel Kruskal



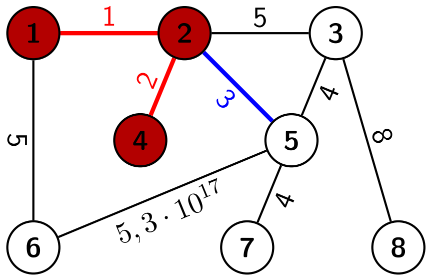
Eksempel Kruskal



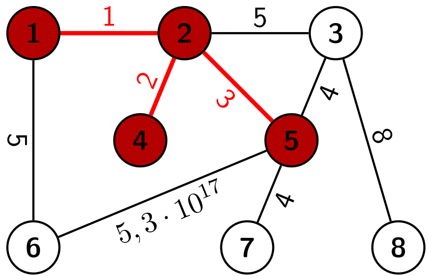
Eksempel Kruskal



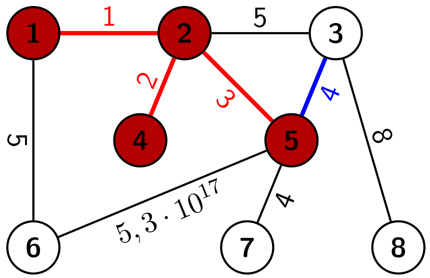
Eksempel Kruskal



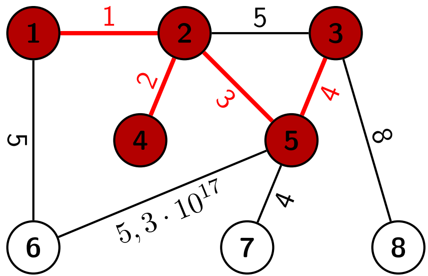
Eksempel Kruskal



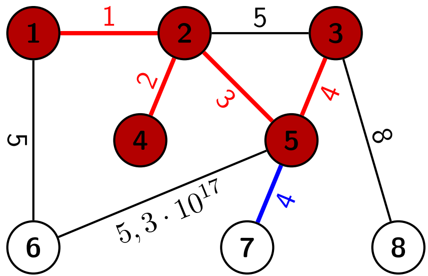
Eksempel Kruskal



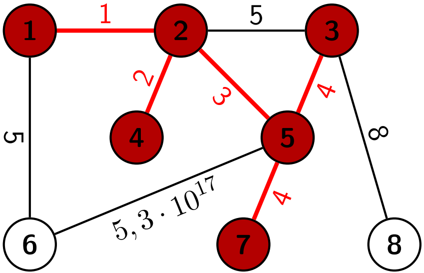
Eksempel Kruskal



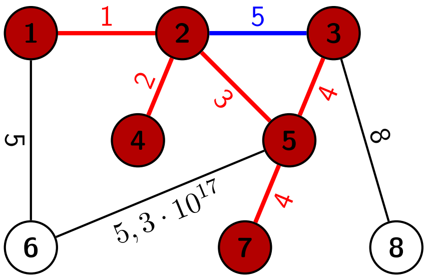
Eksempel Kruskal



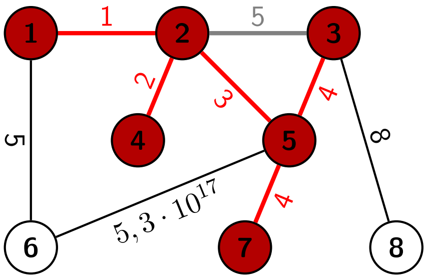
Eksempel Kruskal



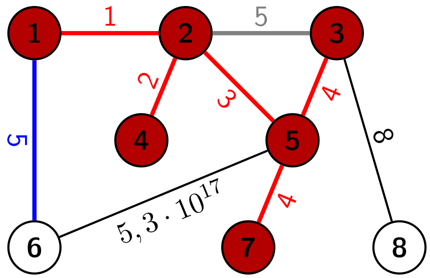
Eksempel Kruskal



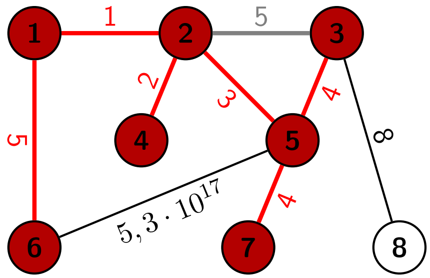
Eksempel Kruskal



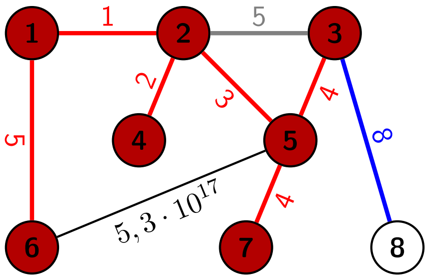
Eksempel Kruskal



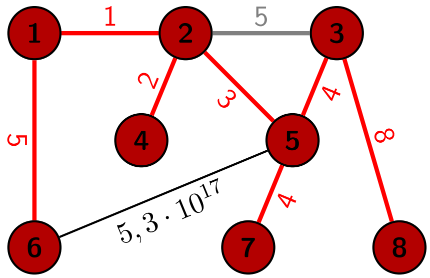
Eksempel Kruskal



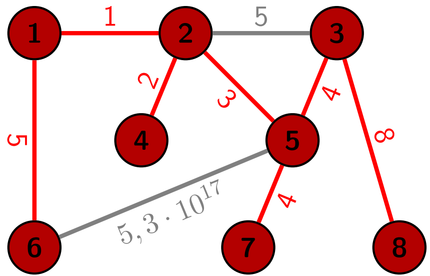
Eksempel Kruskal



Eksempel Kruskal



Eksempel Kruskal



Lykke til på eksamen!

Takk for oss :)