Oplæg og øvelser, herunder frugt og vand

Gerth Stølting Brodal

Datalogisk Institut

Aarhus Universitet

MasterClass Matematik, Mærsk Mc-Kinney Møller Videncenter, Sorø, 29-31. oktober 2009

Algoritmer: Matricer og Grafer

Gerth Stølting Brodal

Datalogisk Institut

Aarhus Universitet

MasterClass Matematik, Mærsk Mc-Kinney Møller Videncenter, Sorø, 29-31. oktober 2009

Gerth Stølting Brodal

1985-1988 Aabenraa Gymnasium

1989-1994 **Datalogi-Matematik**, Aarhus Universitet (*cand. scient*)

1993-1997 PhD i **Datalogi**, Aarhus Universitet

1997-1998 Post. doc., Max-Planck-Institut für Informatik, Saarbrücken, Tyskland

1999- Lektor, Datalogisk Institut, Aarhus Universitet

Indhold

- Algoritmik
- Matricer
  - Multiplikation: Naive algoritme, Strassen's algoritme, rekursionsligninger, optimal rækkefølge for multiplikation af matricer
- Grafer
  - Korteste veje: Korteste veje: matrix potensopløftning (tropisk algebra), Seidel's algoritme, Dijkstra's algoritme
  - Planare grafer: Euler's formel, Kuratowski's Sætning, planaritets test
- Eksempel fra DM i programmering 2009

Formiddagens arbejdsform: Forelæsning med små øvelser

Algoritmik

Algoritmik

- Metoder til at løse problemer

- Krav til en algoritme
  - Korrekt (gør det den skal)
  - **Effektiv** (f.eks. m.h.t. tid for at løse et problem)

Assymptotisk Tid

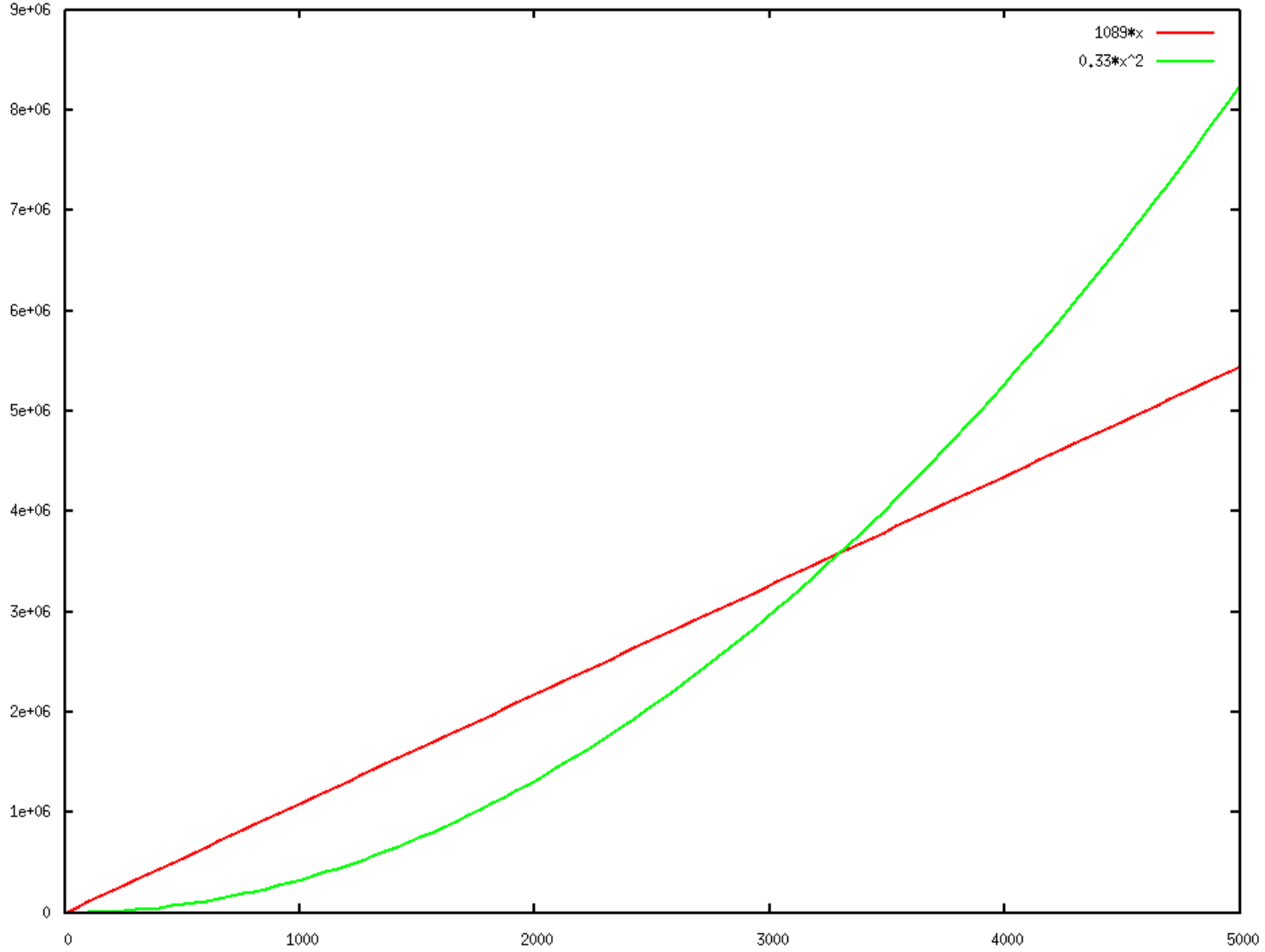
- Lad n være størrelsen af et problem

- Algoritme **A**: Løser problemet i tid n
- Algoritme **B**: Løser problemet i tid n^2

Opgave

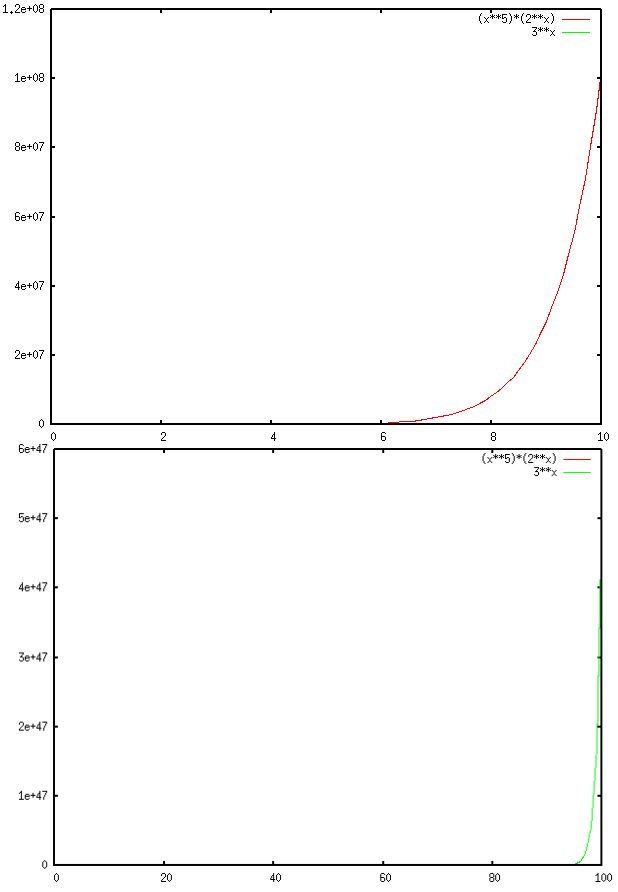
- Hvilken algoritme er hurtigst ? for $n \to \infty$

$1089 \cdot n$ vs $0.33 \cdot n^2$

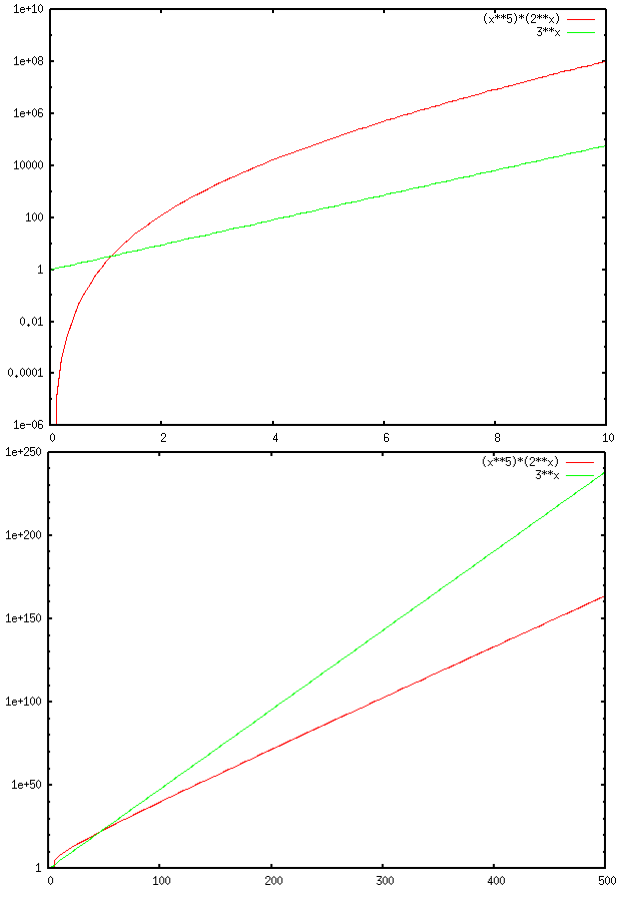


$n^5 \cdot 2^n$ vs 3^n

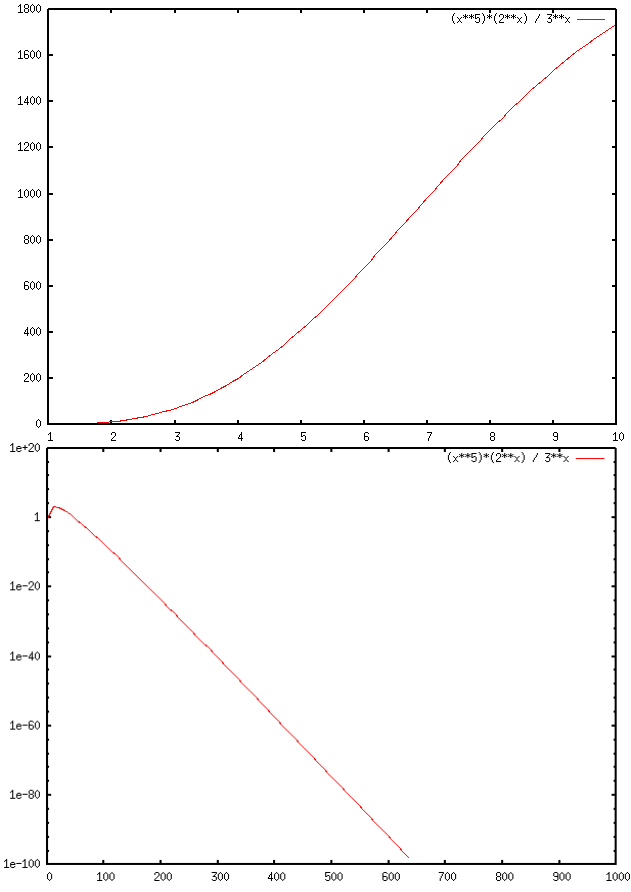
Plot af de to funktioner
– ikke særlig informativ



Plot af de to funktioner
med logaritmisk y-akse
– første plot misvisende



Plot af brøken mellem
de to funktioner
– første plot misvisende



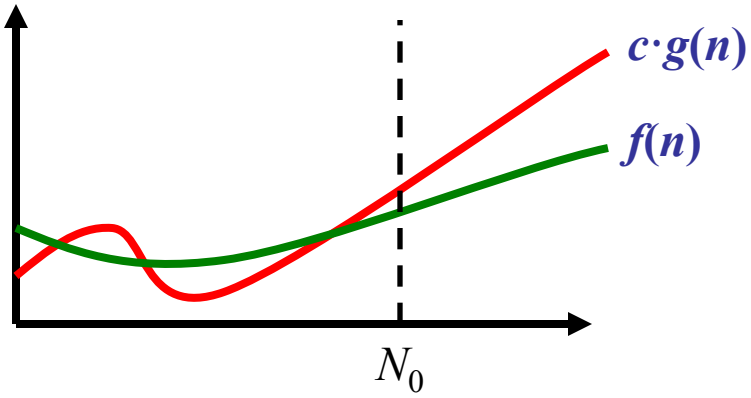
Assymptotisk Tid

A	B	Opgave Mindste funktion
n^3	n^2	
$3 \cdot n^2 + 7 \cdot n$	n^2	
2^n	3^n	
4^n	$n^2 \cdot 2^n$	
n^3	$\log_2 n \cdot n^{0.1}$	
$(\log_2 n)^3$	$n^{0.1}$	
$n^2 \cdot \log_2 n + 7 \cdot n^{2.5}$	$n^{2.5}$	

Assymptotisk Tid (formel)

Definition: $f(n) = O(g(n))$

hvis $f(n)$ og $g(n)$ er ikke negative funktioner og der findes $c \leq 0$ og N_0 så for alle $n \geq N_0$: $f(n) \leq c \cdot g(n)$



Eksempel

$$n^5 \cdot 2^n = O(3^n)$$

$$6 \cdot 7 = 42$$

Multiplikation af lange heltal

- I og J hver heltal med n cifre
- Naive implementation kræver $O(n^2)$ operationer
- Lad $I = I_h \cdot 10^{n/2} + I_l$ og $J = J_h \cdot 10^{n/2} + J_l$
- $I \cdot J = I_h \cdot J_h \cdot 10^n + ((I_h - I_l) \cdot (J_l - J_h) + I_l \cdot J_l + I_h \cdot J_h) \cdot 10^{n/2} + I_l \cdot J_l$

$T(n) \leq 3 \cdot T(n/2) + c \cdot n$	for $n \geq 2$
$T(n) \leq c$	for $n = 1$

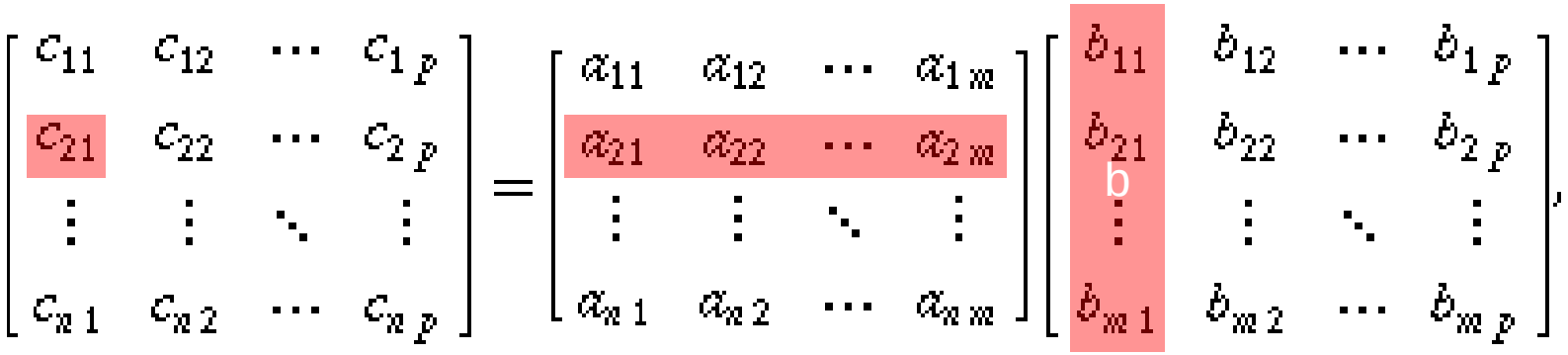
- $T(n) = O(n^{\log_2 3})$

Matricer

Matrix

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{np} \end{bmatrix}$$

Matrix Multiplikation



$$c_{ij} = \sum_{k=1..m} a_{ik} \cdot b_{kj}$$

Naive algoritme: tid $O(npm)$

Matrix Multiplikation

$$(7)(3) = \boxed{\text{Opgave}\quad ?}$$

Matrix Multiplikation

$$\begin{pmatrix} 3 & 2 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 7 \\ 4 & 3 \end{pmatrix} =$$

Opgave

?

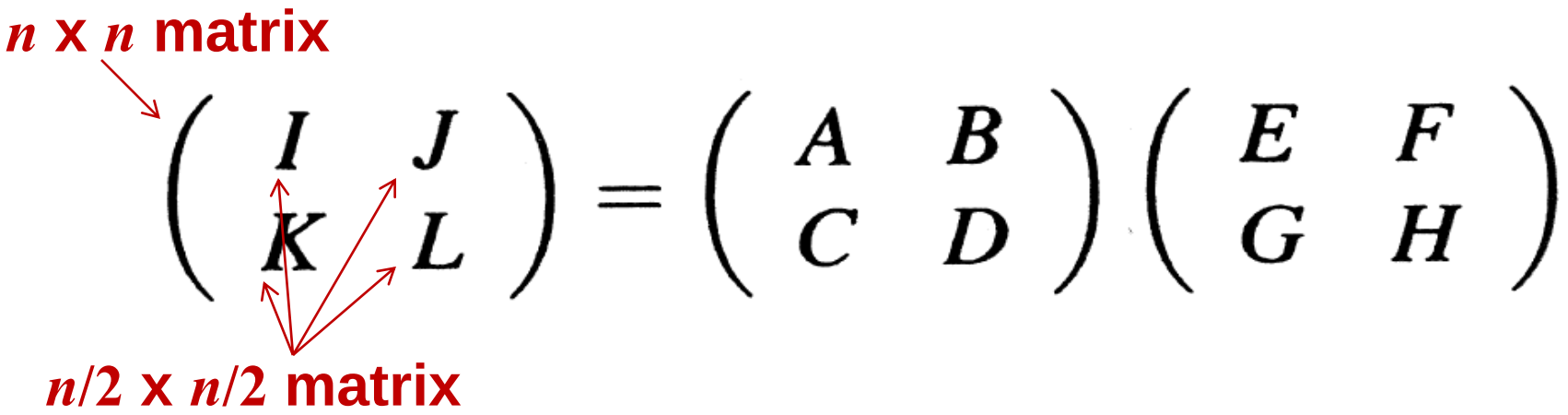
Matrix Multiplikation

$$\begin{pmatrix} 3 & -1 & 1 & 3 \\ 4 & 4 & 0 & 2 \\ 3 & 1 & 2 & 8 \\ 7 & 4 & 4 & 5 \end{pmatrix} \begin{pmatrix} 3 & -1 & 1 & 4 \\ 4 & 2 & 0 & 1 \\ 1 & 0 & -3 & 7 \\ 6 & 4 & 4 & 2 \end{pmatrix} =$$

Opgave

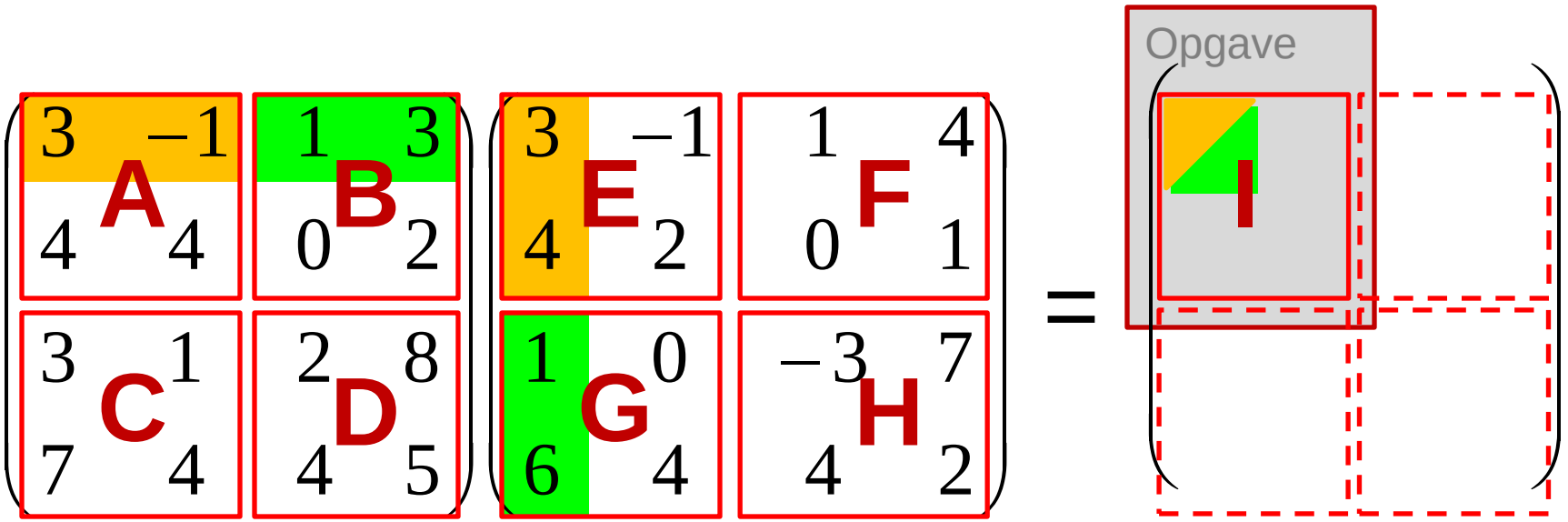
?

(Kvadratisk) Matrix Multiplikation



$$
\begin{aligned}
I &= AE + BG \\
J &= AF + BH \\
K &= CE + DG \\
L &= CF + DH
\end{aligned}
$$

Matrix Multiplikation



$$I = AE + BG$$

(Kvadratisk) Matrix Multiplikation

$$\begin{pmatrix} I & J \\ K & L \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} E & F \\ G & H \end{pmatrix}$$

$$\begin{aligned} I &= AE + BG \\ J &= AF + BH \\ K &= CE + DG \\ L &= CF + DH \end{aligned}$$

- $A,B,...,K,L$ er $n/2$ x $n/2$-matricer
- I,J,K,L kan beregnes med 8 rekursive multiplication på $n/2$ x $n/2$-matricer + 4 matrix additioner
- $T(n) \leq 8 \cdot T(n/2) + c \cdot n^2$ for $n \geq 2$

  $T(n) \leq c$ for $n = 1$
- $T(n) = O(n^3)$

Strassen's Matrix Multiplikation

1969

$$\begin{pmatrix} I & J \\ K & L \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} E & F \\ G & H \end{pmatrix}$$

$$\begin{aligned}
I &= S_5 + S_6 + S_4 - S_2 \\
&= (A+D)(E+H) + (B-D)(G+H) + D(G-E) - (A+B)H \\
&= AE + DE + AH + DH + BG - DG + BH - DH + DG - DE - AH - BH \\
&= AE + BG.
\end{aligned}$$

$$\begin{aligned}
J &= S_1 + S_2 \\
&= A(F-H) + (A+B)H \\
&= AF - AH + AH + BH \\
&= AF + BH.
\end{aligned}$$

$$\begin{aligned}
K &= S_3 + S_4 \\
&= (C+D)E + D(G-E) \\
&= CE + DE + DG - DE \\
&= CE + DG.
\end{aligned}$$

$$\begin{aligned}
L &= S_1 - S_7 - S_3 + S_5 \\
&= A(F-H) - (A-C)(E+F) - (C+D)E + (A+D)(E+H) \\
&= AF - AH - AE + CE - AF + CF - CE - DE + AE + DE + AH + DH \\
&= CF + DH.
\end{aligned}$$

$$\begin{aligned}
S_1 &= A(F-H) \\
S_2 &= (A+B)H \\
S_3 &= (C+D)E \\
S_4 &= D(G-E) \\
S_5 &= (A+D)(E+H) \\
S_6 &= (B-D)(G+H) \\
S_7 &= (A-C)(E+F)
\end{aligned}$$

7 rekursive multiplikationer

Strassen's Matrix Multiplikation

- Bruger 18 matrix additioner (tid $O(n^2)$) og 7 rekursive matrix multiplikationer

$$T(n) \leq 7 \cdot T(n/2) + c \cdot n^2 \quad \text{for } n \geq 2$$

$$T(n) \leq c \quad \text{for } n = 1$$

- $T(n) = O(n^{2.81})$ hvor $2.81 = \log_2 7$

Matrix Multiplikation

Naive algoritme $O(n^3)$

Strassen's algoritme $O(n^{2.81})$

Coppersmith-Winograd $O(n^{2.376})$

(kun teoretisk interesse)

Matrix-kæde Multiplikation

Matrix-kæde Multiplikation

$(A \cdot B) \cdot C$ eller $A \cdot (B \cdot C)$?

Matrix multiplikation er associativ (kan sætte paranteser som man vel) men ikke kommutative (kan ikke bytte rundt ¨på rækkefølgen af matricerne)

Matrix-kæde Multiplikation

Opgave

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & -1 & 0 & 3 \\ 4 & 0 & -2 & 1 & 0 \end{pmatrix} \begin{pmatrix} 4 \\ 1 \\ -2 \\ 0 \\ 1 \end{pmatrix} = ?$$

Matrix-kæde Multiplikation

Problem: Find den bedste rækkefølge (paranteser) for at gange n matricer sammen

$$A_1 \cdot A_2 \cdots A_n$$

hvor A_i er en p_{i-1} x p_i matricer

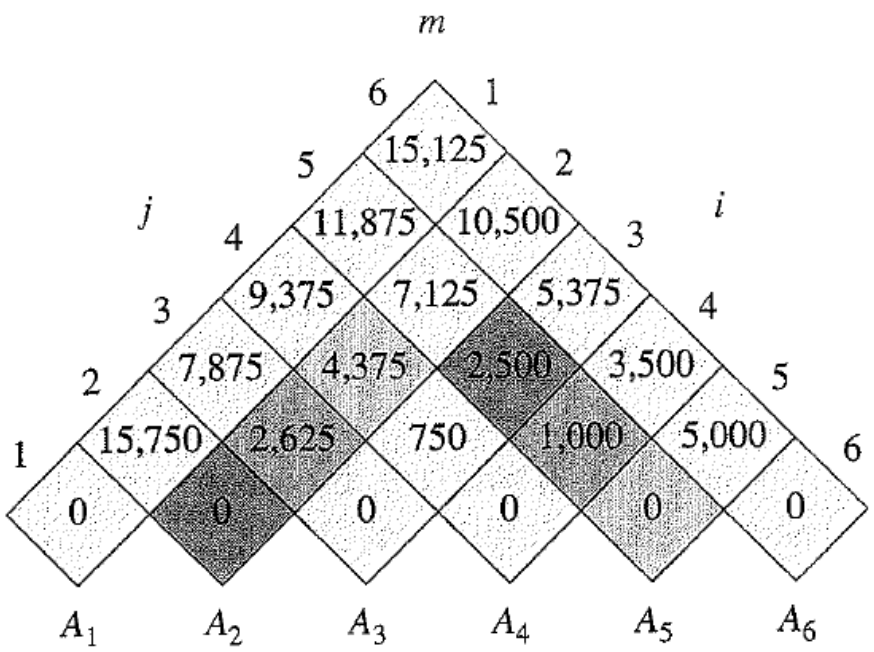
NB: Der er $C_n \approx \frac{4^n}{n^3\sqrt{\pi}}$ mulige måder for paranteserne (det n'te Catalan tal)

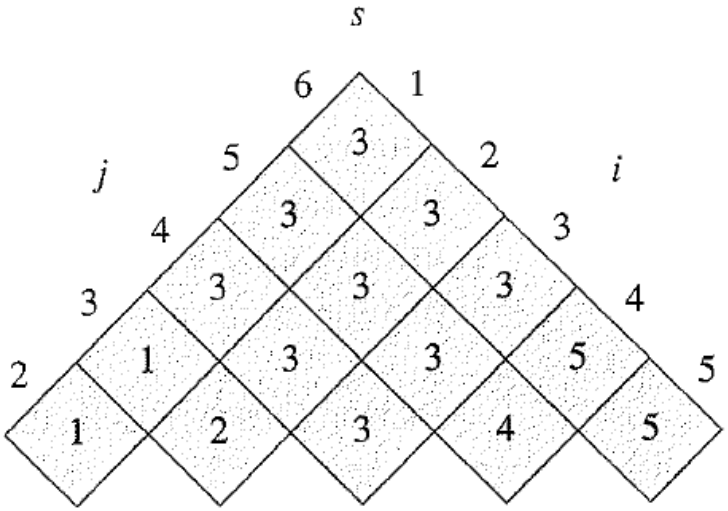
Matrix-kæde Multiplikation

$m[i,j]$ = minimale antal (primitive) multiplikationer for at beregne $A_i \cdot \ldots \cdot A_j$

$$m[i, j] = \begin{cases} 0 & \text{if } i = j \\ \min\limits_{i \le k < j} \{m[i, k] + m[k+1, j] + p_{i-1} p_k p_j\} & \text{if } i < j \end{cases}$$

Matrix-kæde Multiplikation





matrix	dimension
A_1	30×35
A_2	35×15
A_3	15×5
A_4	5×10
A_5	10×20
A_6	20×25

$$m[i, j] = \begin{cases} 0 & \text{if } i = j \\ \min\limits_{i \le k < j} \{m[i, k] + m[k+1, j] + p_{i-1} p_k p_j\} & \text{if } i < j \end{cases}$$

Matrix-kæde Multiplikation

```
MATRIX-CHAIN-ORDER(p)
 1  n ← length[p] − 1
 2  for i ← 1 to n
 3        do m[i, i] ← 0
 4  for l ← 2 to n                ▷ l is the chain length.
 5        do for i ← 1 to n − l + 1
 6              do j ← i + l − 1
 7                 m[i, j] ← ∞
 8                 for k ← i to j − 1
 9                    do q ← m[i, k] + m[k + 1, j] + p_{i−1} p_k p_j
10                       if q < m[i, j]
11                          then m[i, j] ← q
12                               s[i, j] ← k
13  return m and s
```

Tid $O(n^3)$

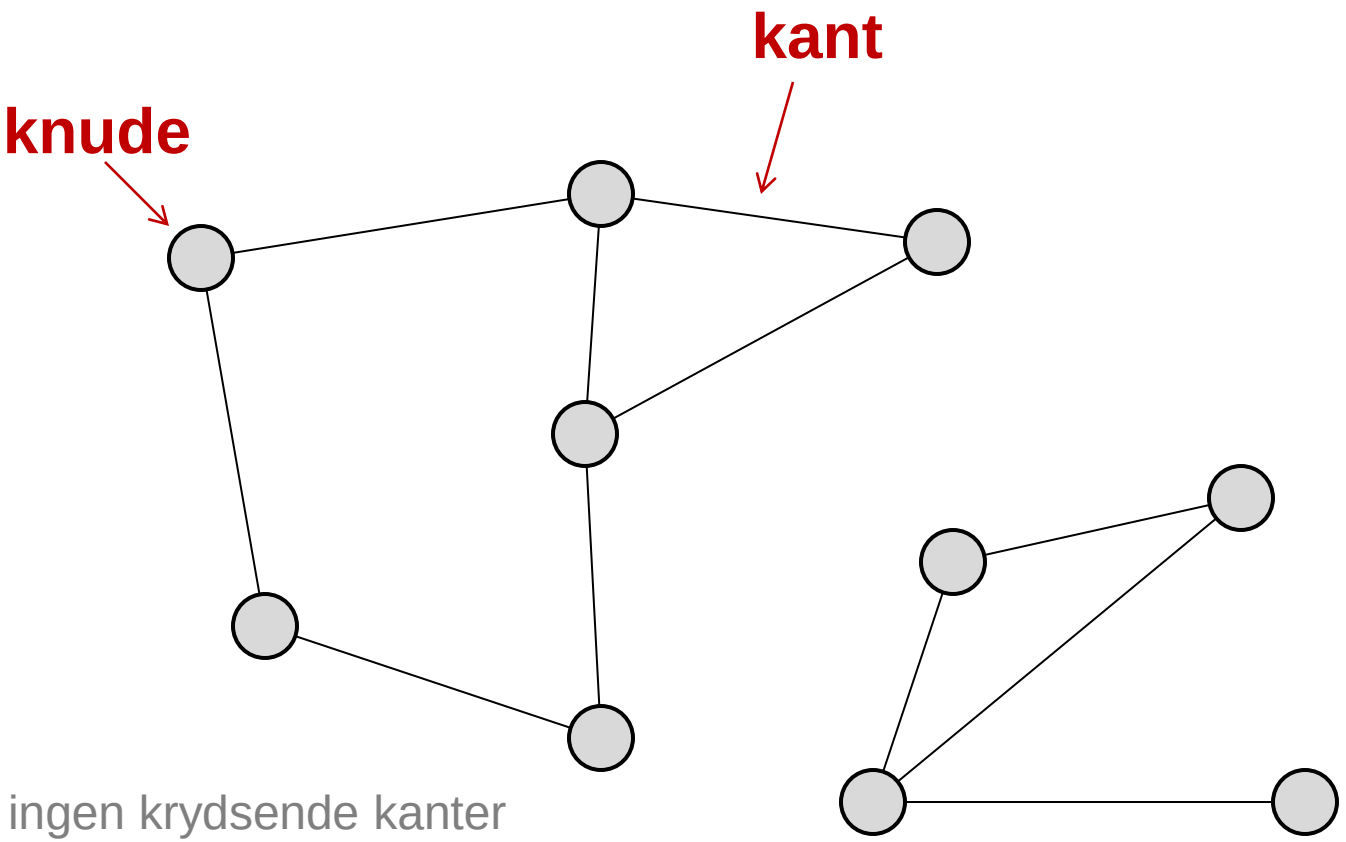
Matrix-kæde Multiplikation

```
PRINT-OPTIMAL-PARENS(s, i, j)
1  if i = j
2      then print "A"_i
3      else print "("
4           PRINT-OPTIMAL-PARENS(s, i, s[i, j])
5           PRINT-OPTIMAL-PARENS(s, s[i, j] + 1, j)
6           print ")"
```

Tid $O(n)$

Grafer

Grafer



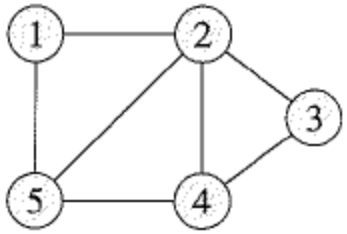
planar = ingen krydsende kanter

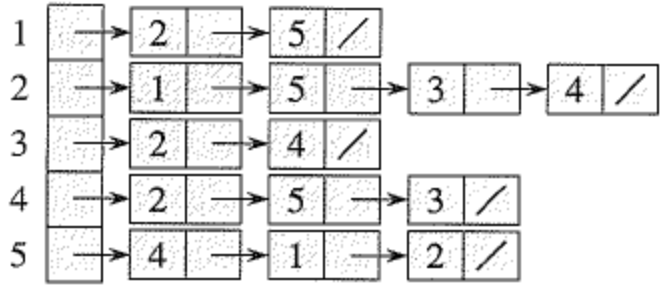
Opgave

- Hvor mange knuder v og kanter e er der?

(vertices and edges)

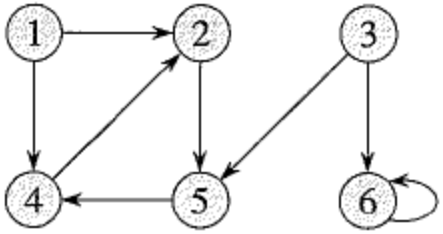
Graf repræsentationer: Incidenslister og incidensmatricer

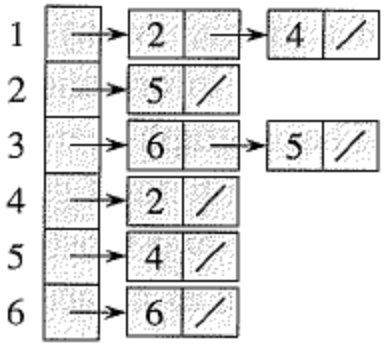




	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Uorienterede grafer





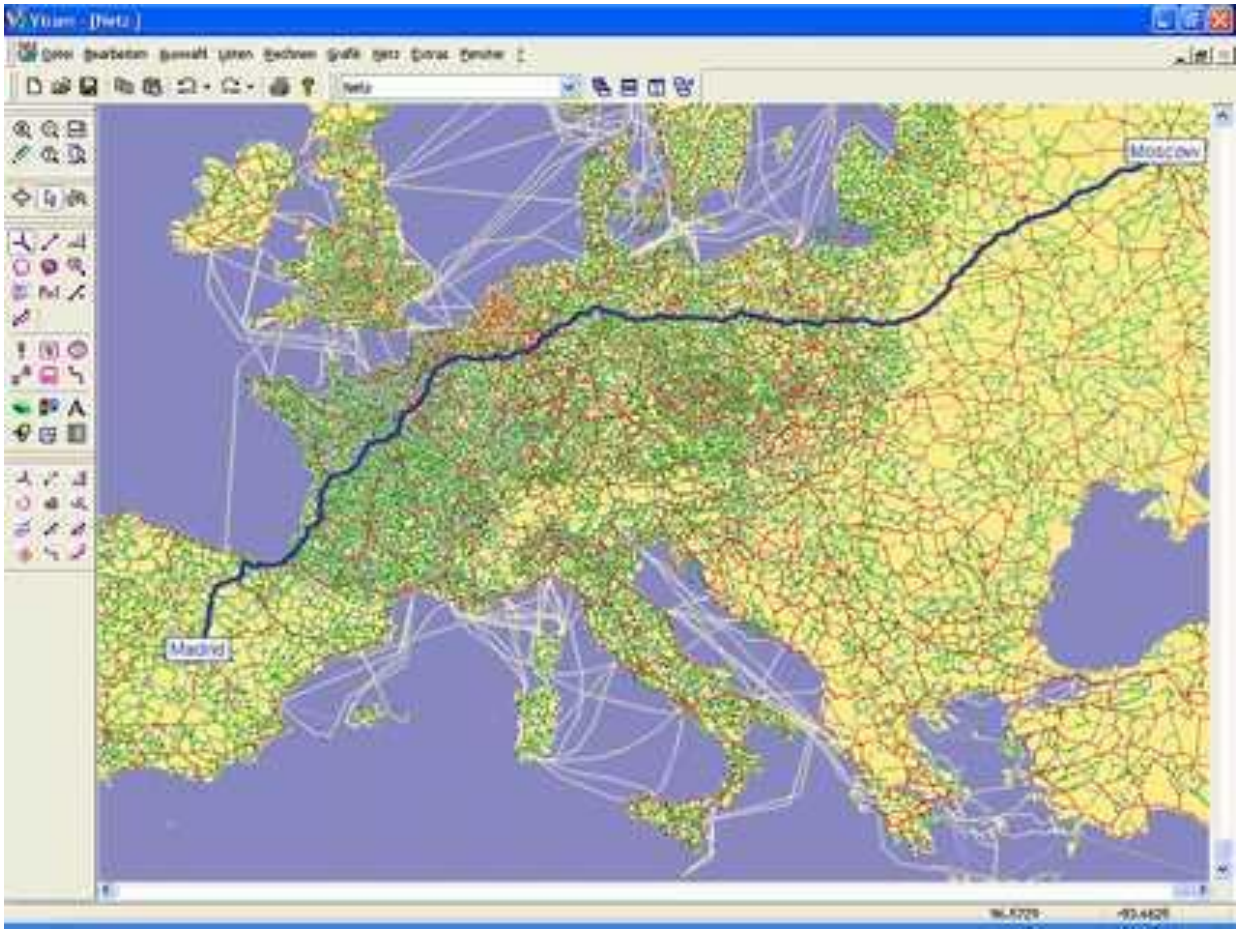
	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Orienterede grafer

Kort over Vest-Europa

- 18.029.721 knuder
- 42.199.587 orienterede kanter

traffic mobility logistics.



Korteste Veje mellem alle Par af Knude

Multiplikation – Tropiske Algebra

$$L' = L \cdot W$$

$$l'_{i,j} = \min_k \left(l_{i,k} + w_{k,j} \right)$$

- Normalt (+,*)-matrix multiplikation
- Tropisk algebra: (min,+)-matrix multiplikation

Multiplikation – Tropiske Algebra

$$L' = L \cdot W \qquad l'_{i,j} = \min_k \left(L_{i,k} + W_{k,j} \right)$$

EXTEND-SHORTEST-PATHS(L, W)

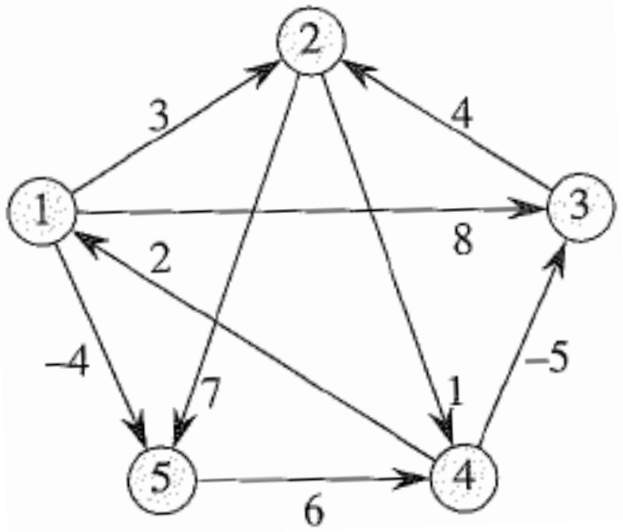
```
1  n ← rows[L]
2  let L' = (l'_ij) be an n × n matrix
3  for i ← 1 to n
4       do for j ← 1 to n
5            do l'_ij ← ∞
6               for k ← 1 to n
7                    do l'_ij ← min(l'_ij, l_ik + w_kj)
8  return L'
```

Korteste Veje mellem alle Par af Knude

```
SLOW-ALL-PAIRS-SHORTEST-PATHS(W)
1  n ← rows[W]
2  L^(1) ← W
3  for m ← 2 to n − 1
4       do L^(m) ← EXTEND-SHORTEST-PATHS(L^(m−1), W)
5  return L^(n−1)
```

diagonalen = 0

Tid $O(v^4)$



$$L^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad L^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad L^{(4)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Korteste Veje mellem alle Par af Knude

```
FASTER-ALL-PAIRS-SHORTEST-PATHS(W)
1  n ← rows[W]
2  L^(1) ← W
3  m ← 1
4  while m < n − 1
5       do L^(2m) ← EXTEND-SHORTEST-PATHS(L^(m), L^(m))
6          m ← 2m
7  return L^(m)
```

Tid $O(v^3 \cdot \log v)$

Floyd-Warshall

FLOYD-WARSHALL(W)

1 $n \leftarrow rows[W]$
2 $D^{(0)} \leftarrow W$
3 **for** $k \leftarrow 1$ **to** n
4 **do for** $i \leftarrow 1$ **to** n
5 **do for** $j \leftarrow 1$ **to** n
6 **do** $d_{ij}^{(k)} \leftarrow \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$
7 **return** $D^{(n)}$

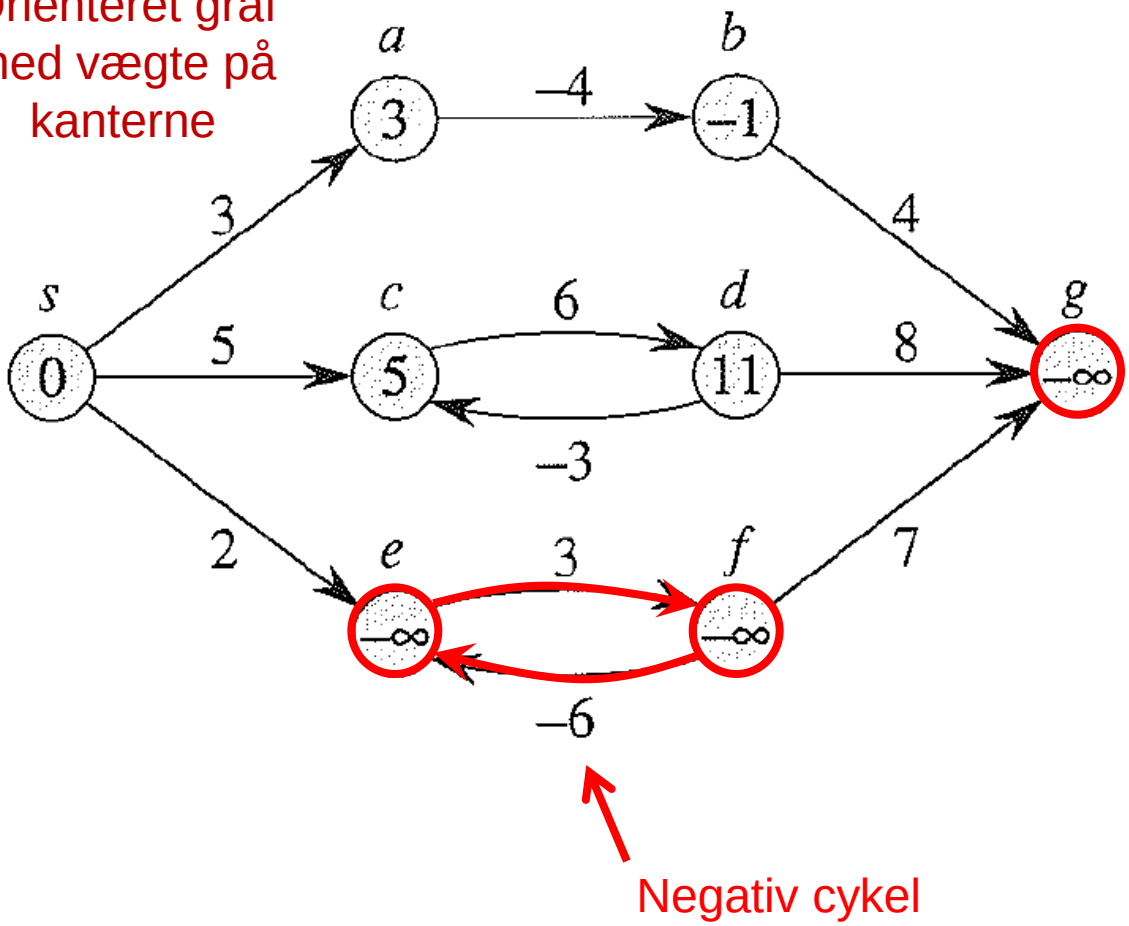
$d_{i,j}^{(k)}$ = afstand fra i til j kun igennem knuderne $\{1,..,k\}$

Tid $O(v^3)$

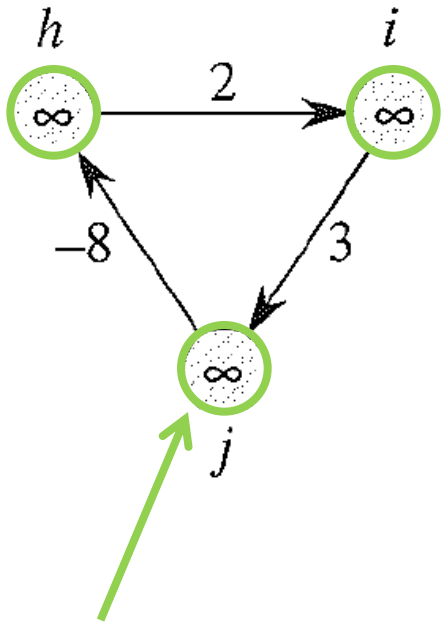
Korteste Veje fra s til alle Par af Knude

Eksempel: Korteste veje fra *s*

Orienteret graf med vægte på kanterne

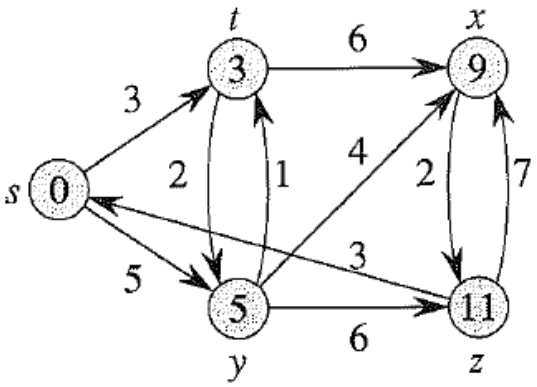


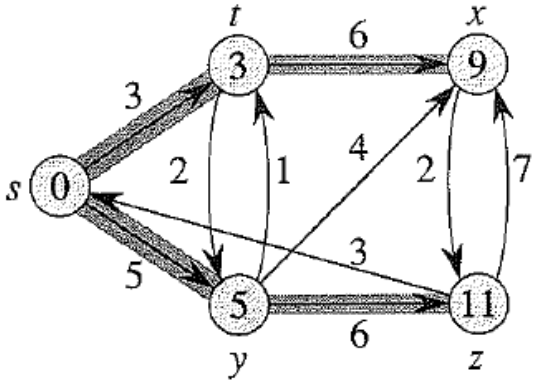
Negativ cykel

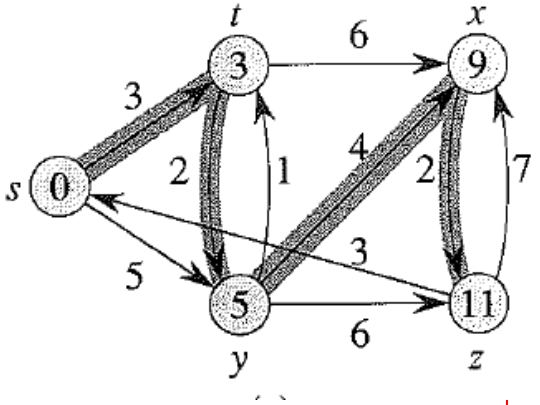


Uforbundet til *s*

Eksempel: Korteste veje træer





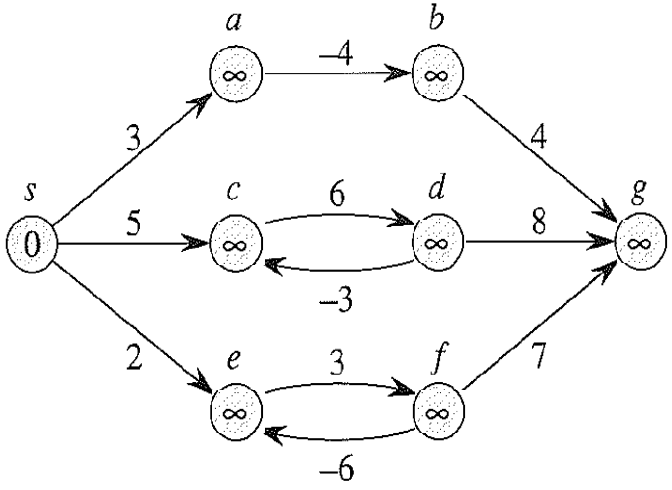


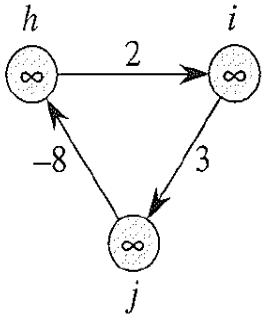
2 forskellige korteste veje træer der
repræsenterer stier fra s med samme længde

Korteste Veje Estimater : Initialisering

INITIALIZE-SINGLE-SOURCE(G, s)

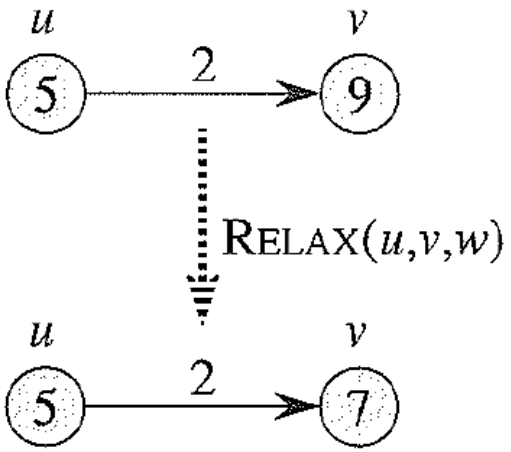
1 **for** each vertex $v \in V[G]$
2 **do** $d[v] \leftarrow \infty$
3 $\pi[v] \leftarrow$ NIL
4 $d[s] \leftarrow 0$



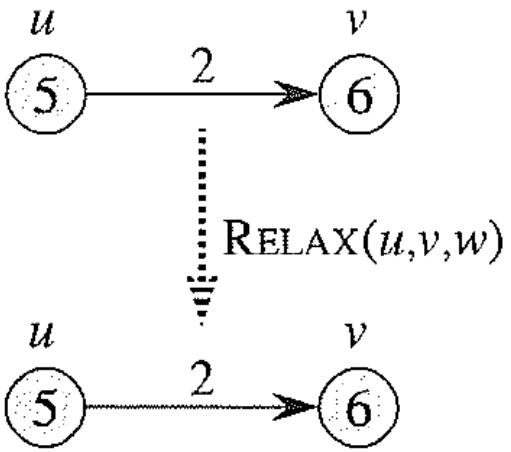


Korteste Veje Estimater : Relax

```
RELAX(u, v, w)
1   if d[v] > d[u] + w(u, v)
2       then d[v] ← d[u] + w(u, v)
3            π[v] ← u
```



Kortere afstand til *v* fundet



Forbedrer ikke afstanden til *v*

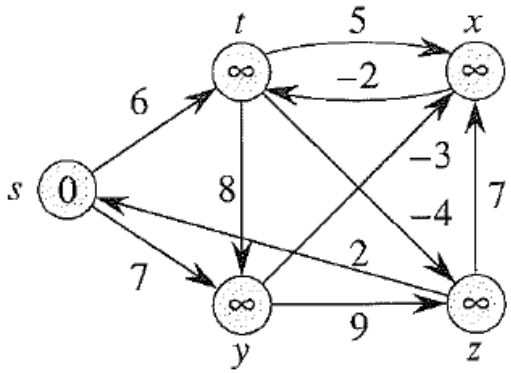
Bellman-Ford:
Korteste Veje i Grafer med Negative Vægte

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i ← 1 to |V[G]| − 1
3       do for each edge (u, v) ∈ E[G]
4            do RELAX(u, v, w)
5  for each edge (u, v) ∈ E[G]
6       do if d[v] > d[u] + w(u, v)
7            then return FALSE
8  return TRUE
```

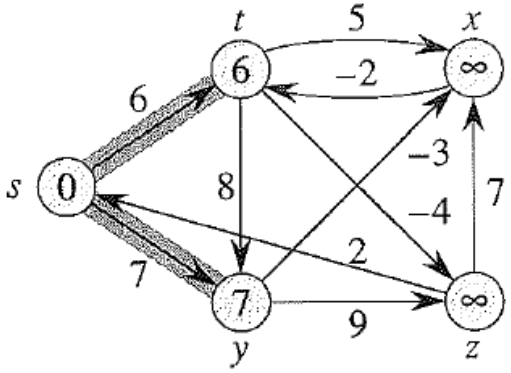
Check for negativ cykel

Tid $O(ve)$

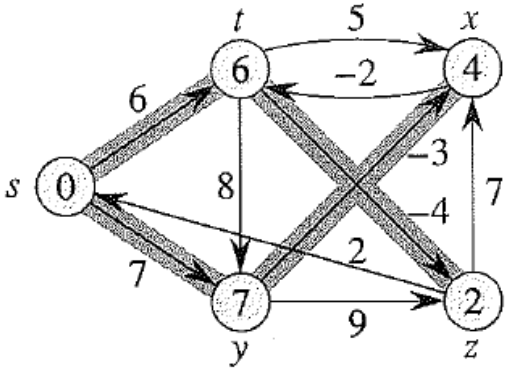
Bellman-Ford: Eksempel



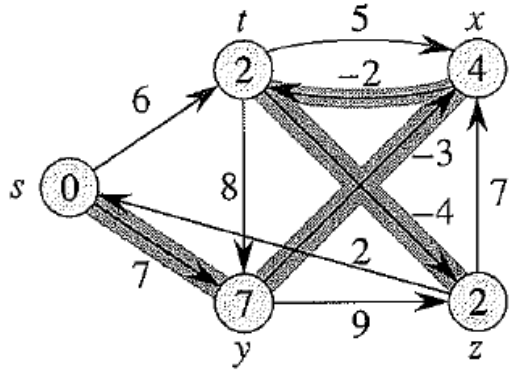
(a)



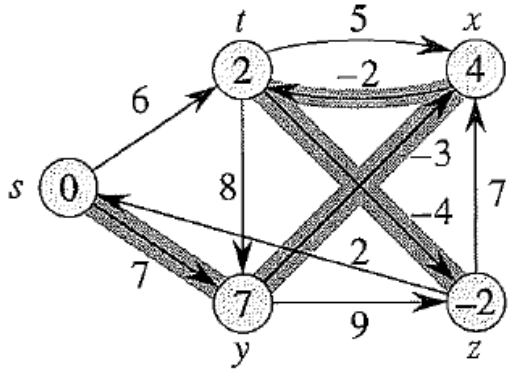
(b)



(c)



(d)



(e)

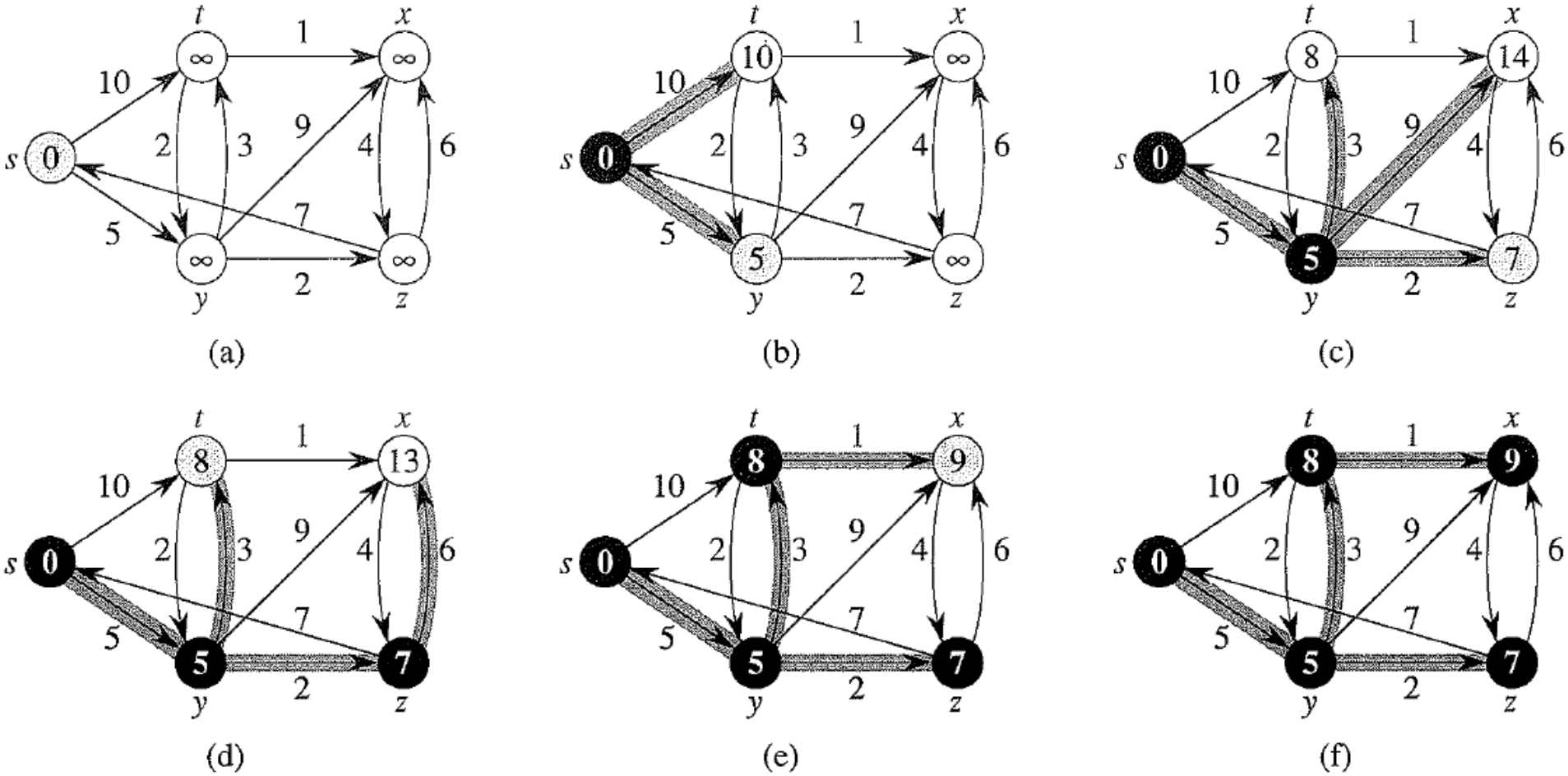
Dijkstra:
Korteste Veje i Grafer uden Negative Vægte

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S ← ∅
3  Q ← V[G]
4  while Q ≠ ∅
5       do u ← EXTRACT-MIN(Q)
6          S ← S ∪ {u}
7          for each vertex v ∈ Adj[u]
8               do RELAX(u, v, w)
```

Q = prioritets kø (besøger knuderne efter stigende afstand fra s)

Tid $O((v+e)\cdot \log v)$
eller $O(v^2+e)$

Dijkstra : Eksempel



(a) (b) (c) (d) (e) (f)

Korteste Veje

<table>
<tr><th colspan="2"></th><th>SSSP
En-til-alle
korteste veje</th><th>APSP
Alle-til-alle
korteste veje</th></tr>
<tr><td colspan="2">Acykliske grafer
(positive og negative vægte)</td><td>$O(v+e)$</td><td>$O(v \cdot (v+e))$</td></tr>
<tr><td rowspan="2">Generelle grafer</td><td>Kun positive vægte</td><td>Dijkstra
$O((v+e) \cdot \log v)$</td><td rowspan="2">Floyd-Warshall
$O(v^3)$</td></tr>
<tr><td>Positive og negative vægte</td><td>Bellman-Ford
$O(e \cdot v)$</td></tr>
</table>

On the All-Pairs-Shortest-Path Problem

RAIMUND SEIDEL*

Computer Science Division
University of California, Berkeley
Berkeley CA 94720

Abstract

The following algorithm solves the distance version of the all-pairs-shortest-path problem for undirected, unweighted n-vertex graphs in time $O(M(n)\log n)$, where $M(n)$ denotes the time necessary to multiply two $n \times n$ matrices of small integers (which is currently known to be $o(n^{2.376})$):

Input: $n \times n$ 0-1 matrix A, the adjacency matrix of undirected, connected graph G

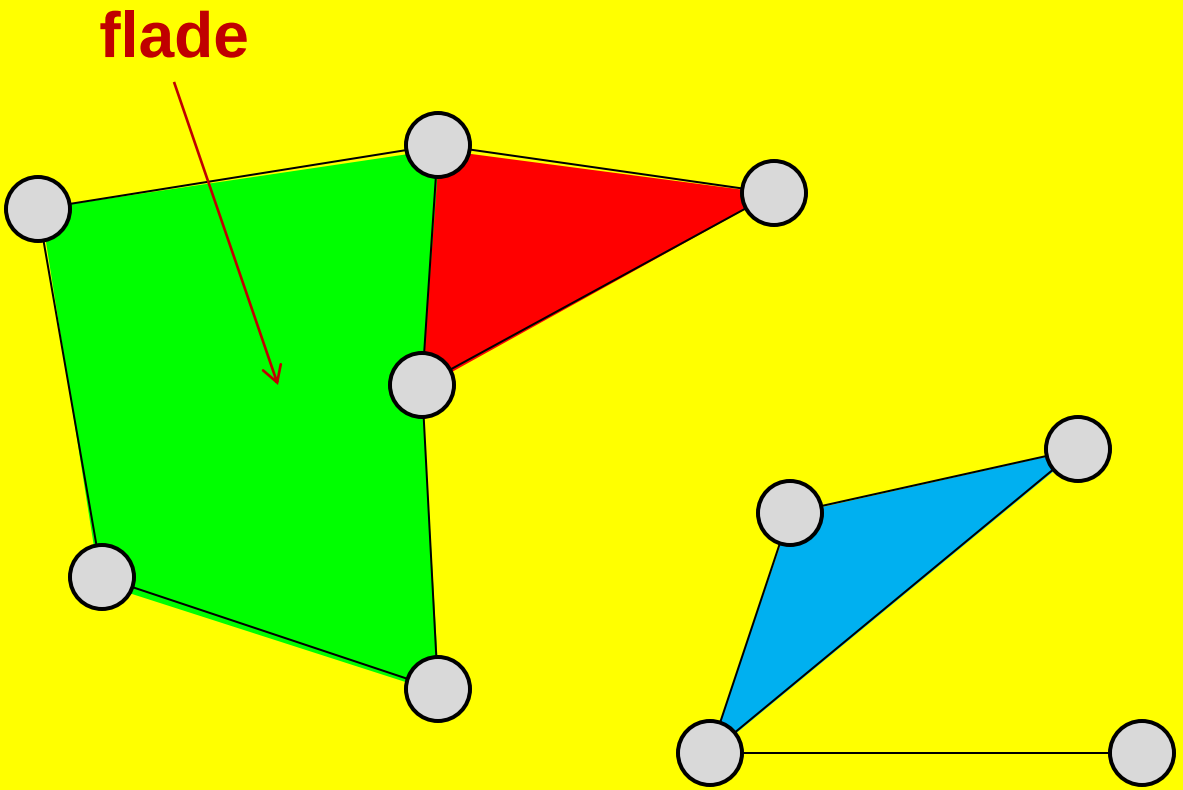
Output: $n \times n$ integer matrix D, with d_{ij} the length of a shortest path joining vertices i and j in G

function APD(A : $n \times n$ 0-1 matrix) : $n \times n$ integer matrix
let $Z = A \cdot A$
let B be an $n \times n$ 0-1 matrix, where $b_{ij} = 1$ iff $i \neq j$ **and** ($a_{ij} = 1$ **or** $z_{ij} > 0$)
if $b_{ij} = 1$ for all $i \neq j$ **then return** $n \times n$ matrix $D = 2B - A$
let $T = \text{APD}(B)$
let $X = T \cdot A$
return $n \times n$ matrix D, where $d_{ij} = \begin{cases} 2t_{ij} & \text{if } x_{ij} \geq t_{ij} \cdot degree(j) \\ 2t_{ij} - 1 & \text{if } x_{ij} < t_{ij} \cdot degree(j) \end{cases}$

STOC 1992

Planare Grafer

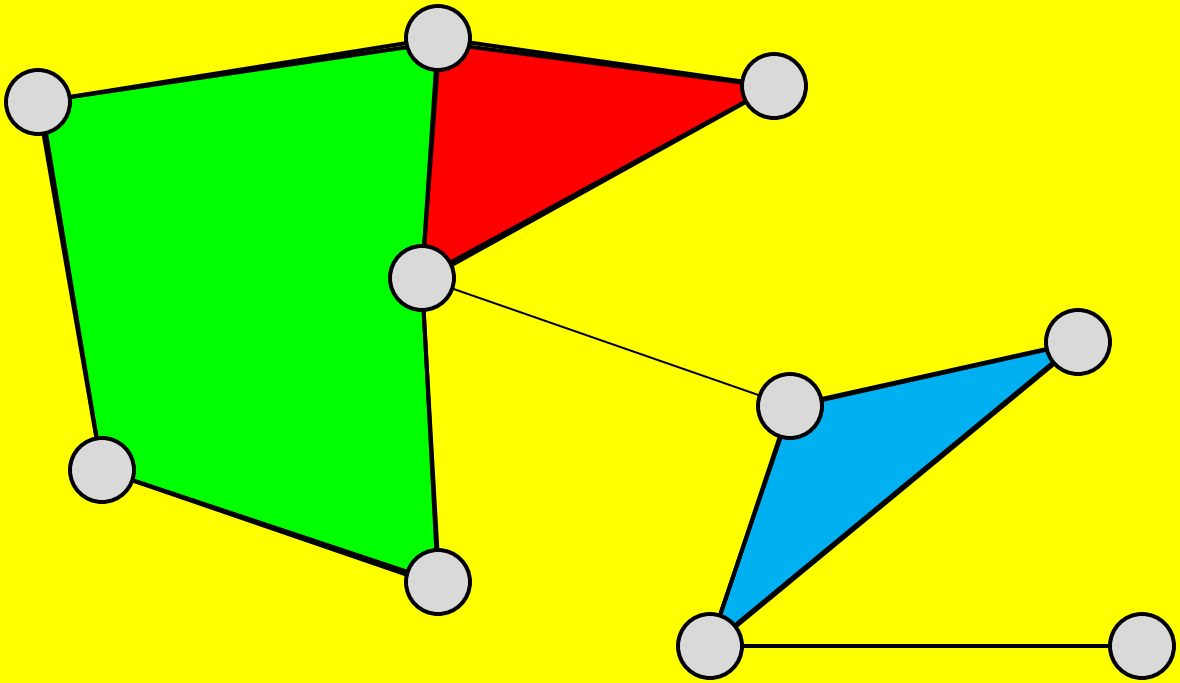
Planare Grafer



Opgave

- Hvor mange flader f er der?

Euler's Formel

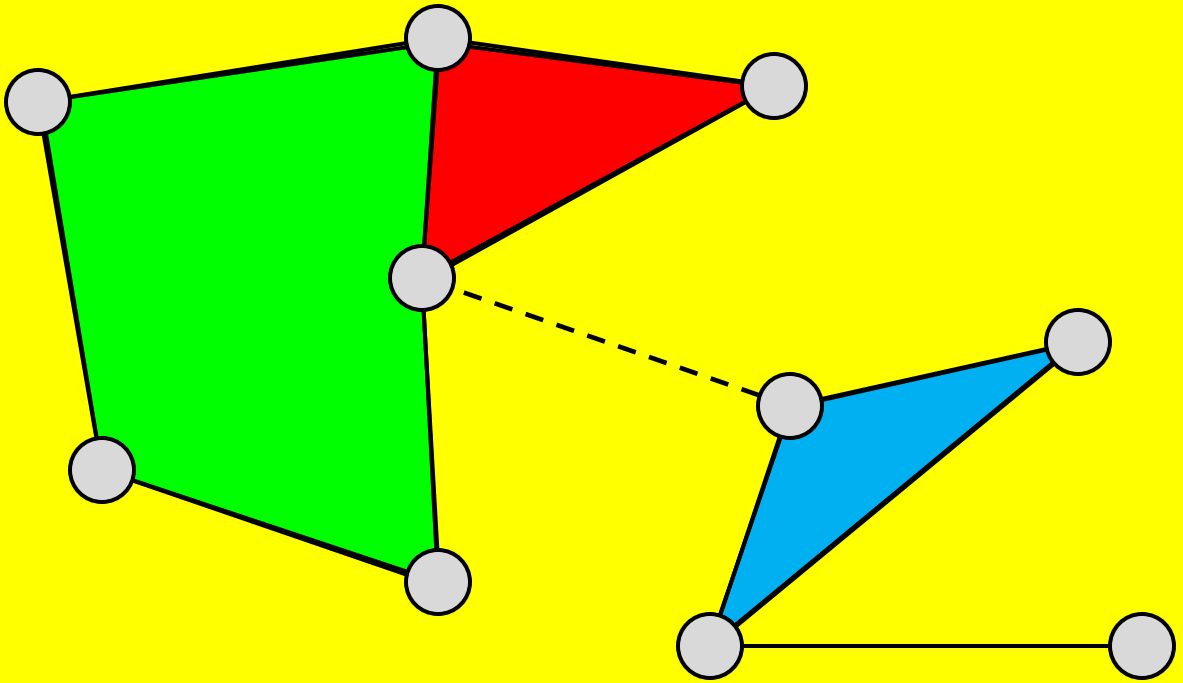


Opgave

- Vis $v-e+f=2$ for en **sammenhængene** planar graf

(10-12+4 = 2)

Euler's Formel



Opgave

- Vis $e \leq 3v - 6$ for $v \geq 3$ for en planar graf

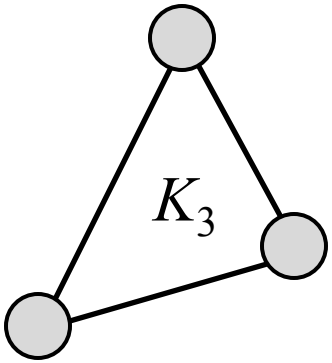
Er følgende grafer planare?

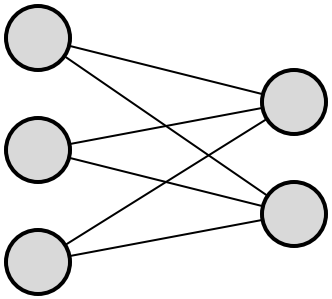
Opgave

n	K_n ?
1	
2	
3	
4	
5	
6	
7	

Opgave

$K_{n,m}$?	1	2	3	4	5
1					
2					
3					
4					
5					



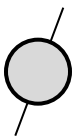


$K_{3,2}$

Kuratowski's Sætning

Kuratowski 1896–1980

En graf er planar hvis og kun hvis den ikke indeholder en delgraf der er en K_5 eller $K_{3,3}$

En delgraf er en graf der opnås ved at slette knuder og kanter og erstatte  med 

Historie – Planaritets Test

- Baseret på Kuratowski's Sætning findes flere algoritmer til at teste om en graf er planar

  (givet en liste af e par (i,j) som angiver kanterne)

1961	Auslander and Parter	test, $O(v^3)$
1964	Demoucron, Malgrange and Pertuiset	test, $O(v^2)$
1974	Hopcroft and Tarjan	test, $O(v)$
1985	Chiba, Nishizeki, Abe and Ozawa	indlejring, $O(v)$

Eksempel fra DM i Programmering 2009

Speedy Escape

The Newton brothers are planning to rob a bank in the city of Alviso and want to figure out a way to escape the city's only police car. They know that their car is faster than the police car so if they could just reach one of the highways exiting the city they will be able to speed away from the police.

The police car has a maximum speed of 160 km/h. Luckily, the brothers know where the police car will start (it's parked at the police station). To be on the safe side they assume that the police car will start moving as soon as they leave the bank and start their car (this is when the alarm goes off).

The brothers want to find a fixed route that ensures that they are able to leave the city no matter what route the police car take and at what speed it drives. However, since the brothers are not very confident drivers they don't want to drive faster than necessary. Luckily they have recently invested in a new hi-tech in-car police escape system that *you* have constructed. This system will tell them what the minimal top speed needed to escape is (and probably other useful things like what route to take).

Let's turn the clock back a bit to the time when you were constructing the escape system and focused on finding the minimal required speed. Can you get it right?

You may treat all roads as infinitesimally narrow and both cars as point objects. If the brothers ever end up at the same point (on any road or intersection) at the same time as the police car they will be caught and by Murphy's law if there is any possibility of this happening it will happen. The two cars start simultaneously and can accelerate/decelerate instantaneously at any time to any speed below or equal to its maximum speed. They can also change roads at intersections or direction anywhere on a road instantaneously no matter what speed they are traveling at.


THE
END