Forelesning 11

Korteste vei fra alle til alle

1. **Johnsons algoritme**

2. **Transitiv lukning**

3. **Floyd-Warshall**

1:3

Johnsons algoritme

Fra 1977.

Efficient Algorithms for Shortest Paths in Sparse Networks

DONALD B. JOHNSON

The Pennsylvania State University, University Park, Pennsylvania

ABSTRACT Algorithms for finding shortest paths are presented which are faster than algorithms previously known on networks which are relatively sparse in arcs Known results which the results of this paper extend are surveyed briefly and analyzed A new implementation for priority queues is employed, and a class of "arc set partition" algorithms is introduced For the single source problem on networks with nonnegative arcs a running

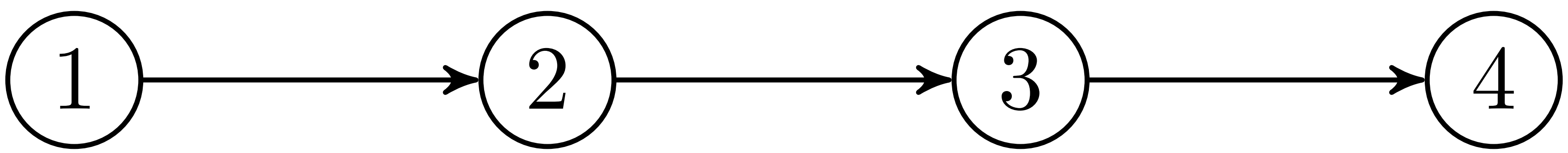
Input: En vektet, rettet graf $G = (V, E)$ uten negative sykler, der $V = \{1, \dots, n\}$, og vektene er gitt av matrisen $W = (w_{ij})$.

Output: En $n \times n$-matrise $D = (d_{ij})$ med avstander, dvs., $d_{ij} = \delta(i, j)$.

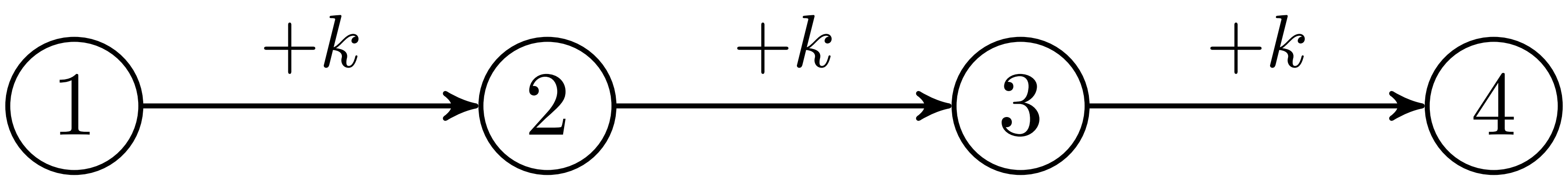
Vi returnerer også en forgjengermatrise $\Pi = (\pi_{ij})$

For spinkle grafer: Dijkstra fra hver node!

Men ... hva om vi har negative kanter?

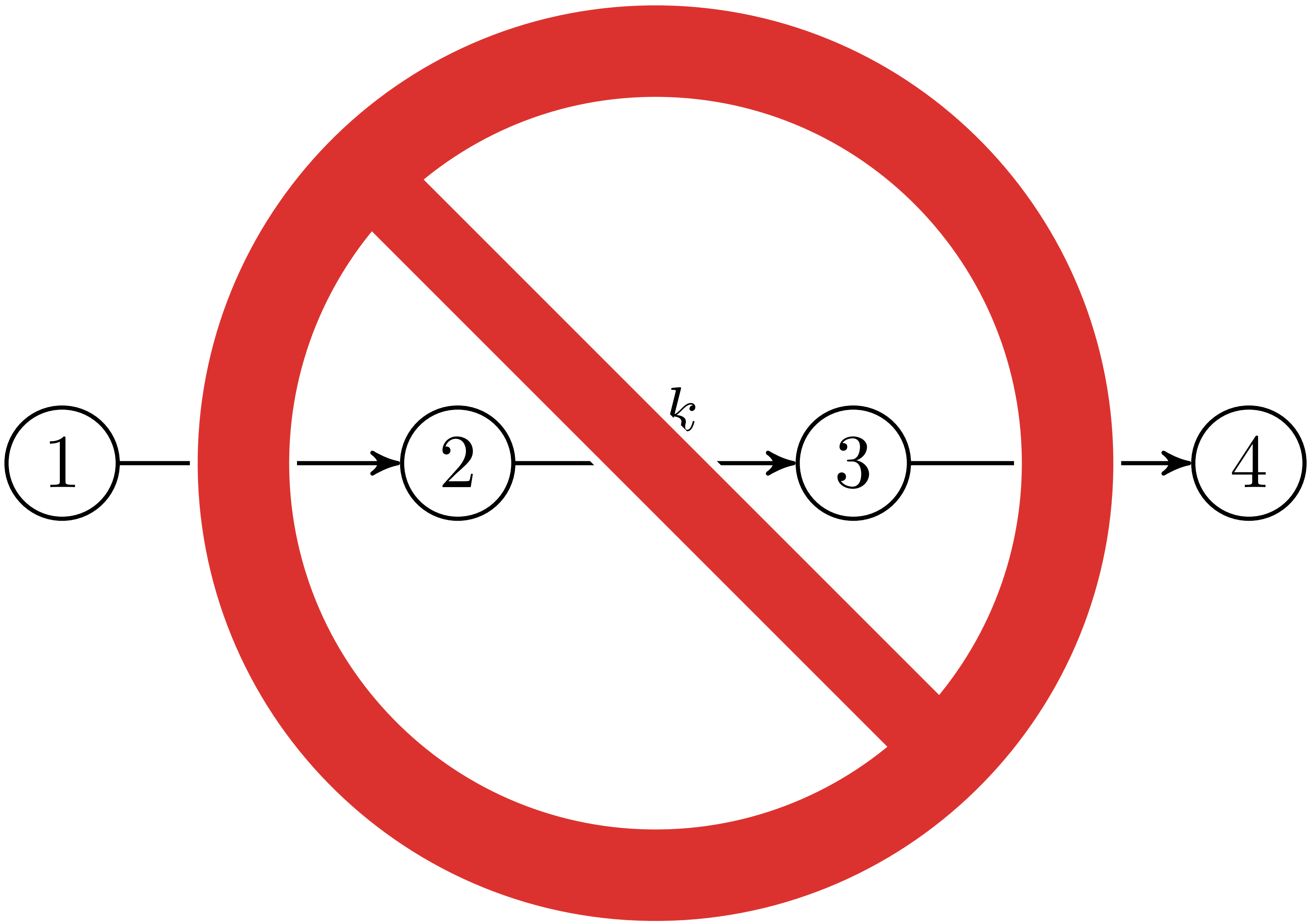


Vi vil øke vekter, men beholde rangering av stier

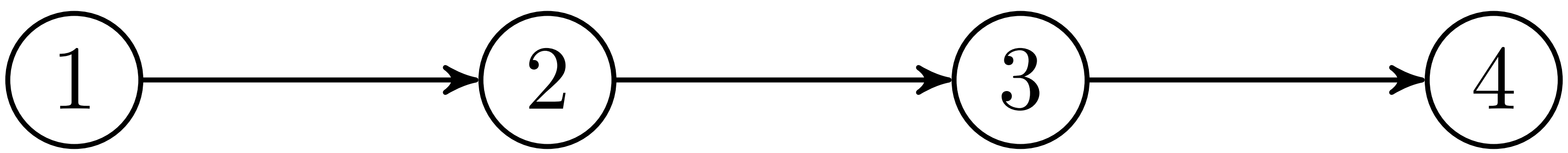


Fast økning: Stier med mange kanter taper på det

mlh@ntnu.no

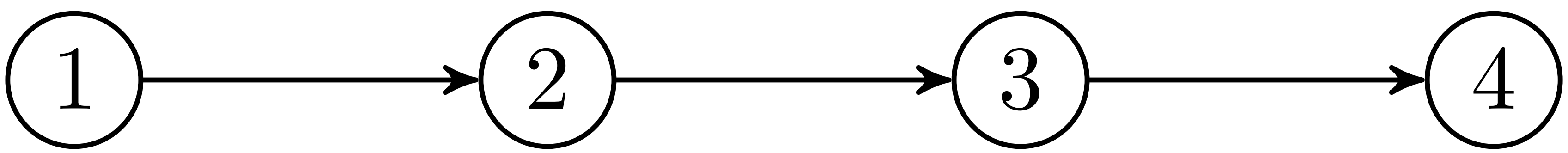


Fast økning: Stier med mange kanter taper på det

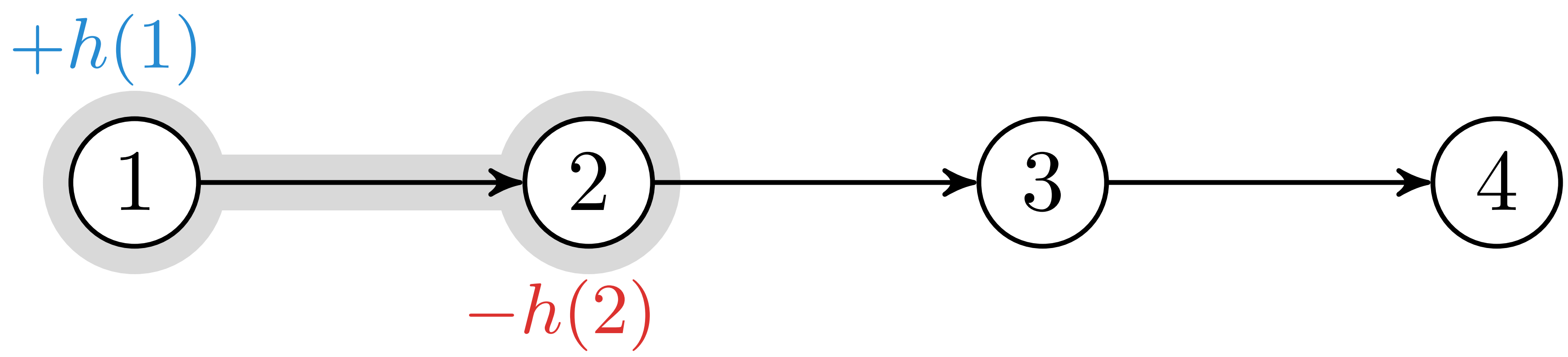


Vi kan tillate oss en teleskopsum . . .

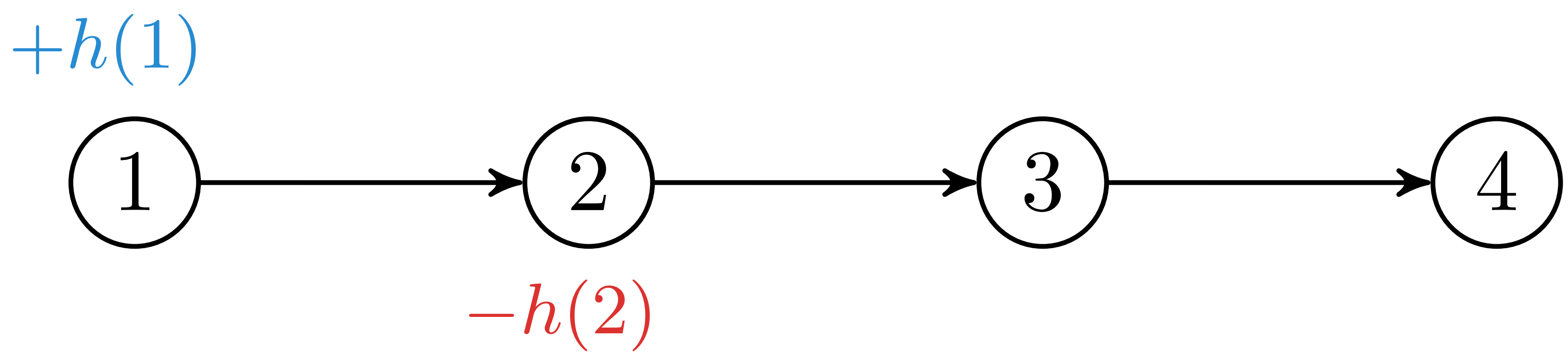
Hver node tilordnes en verdi



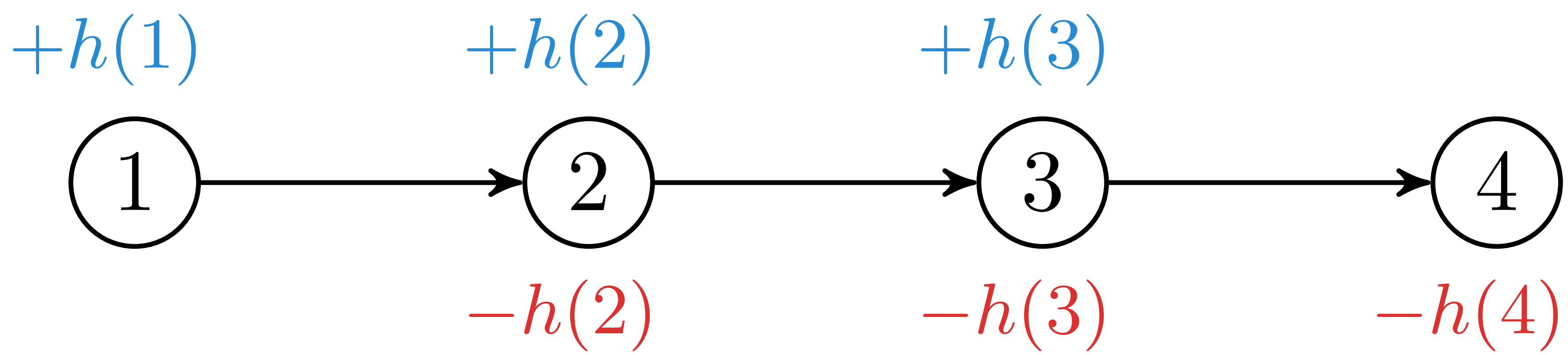
Vekten $w(u, v)$ økes med differansen $h(u) - h(v)$



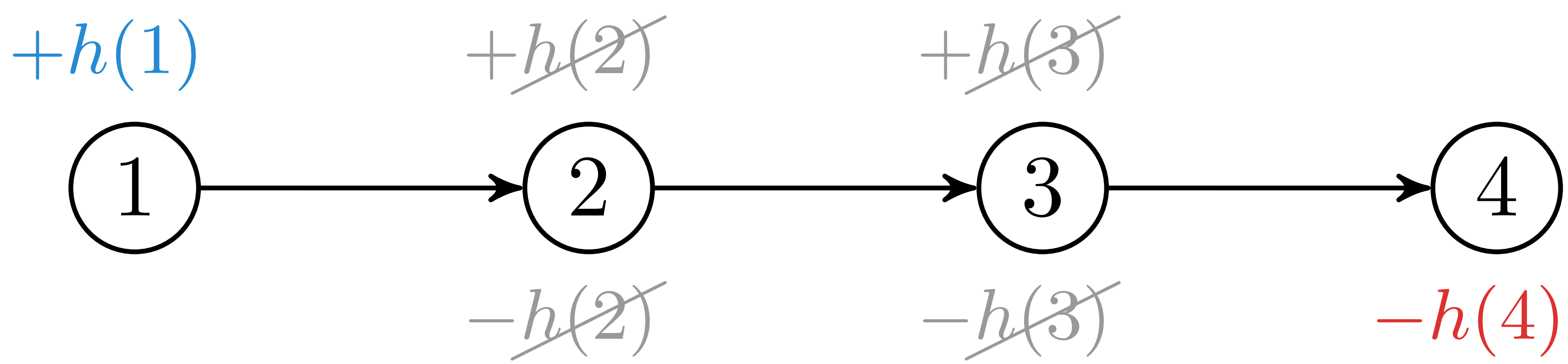
Vekten $w(u, v)$ økes med differansen $h(u) - h(v)$



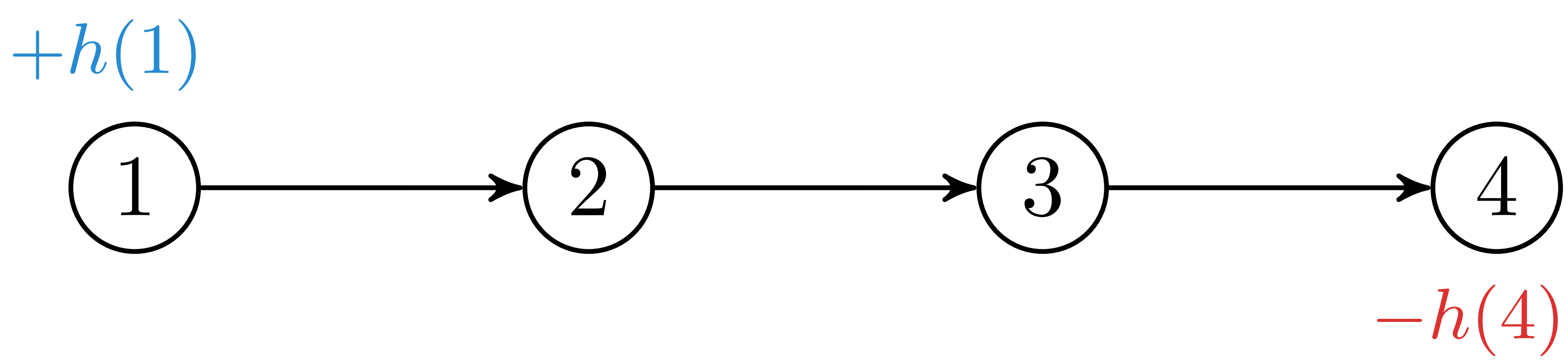
Positive og negative ledd opphever hverandre …



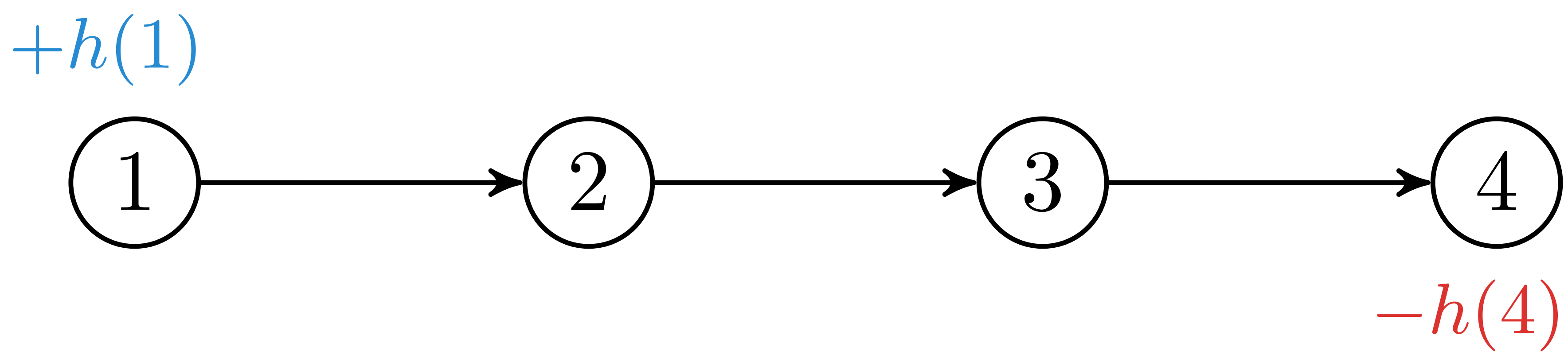
Positive og negative ledd opphever hverandre …



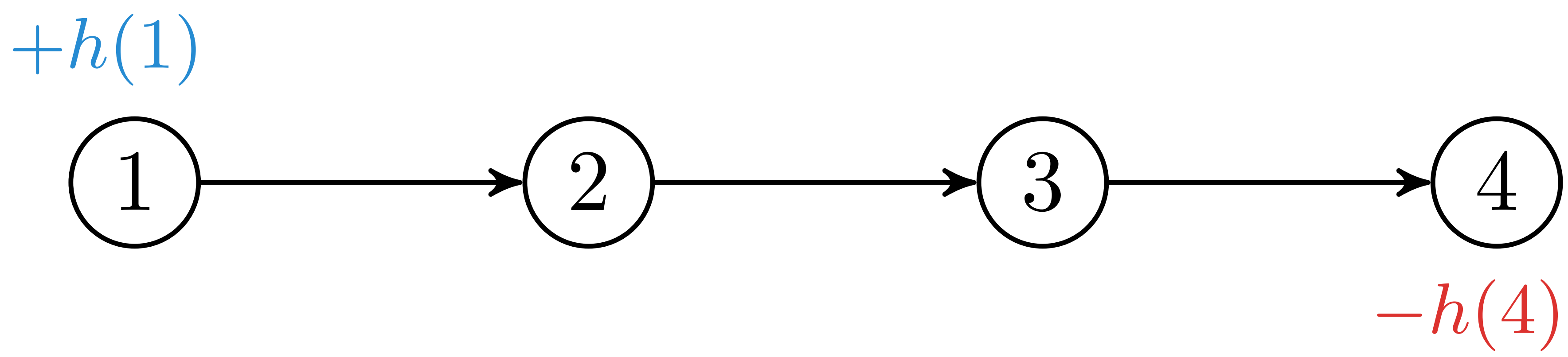
Positive og negative ledd opphever hverandre . . .



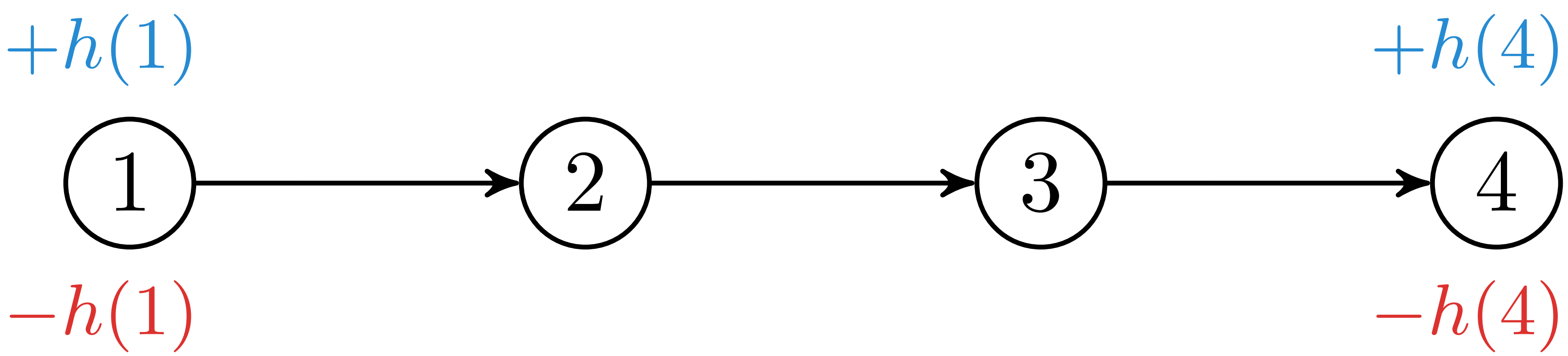
Unntatt første og siste ledd



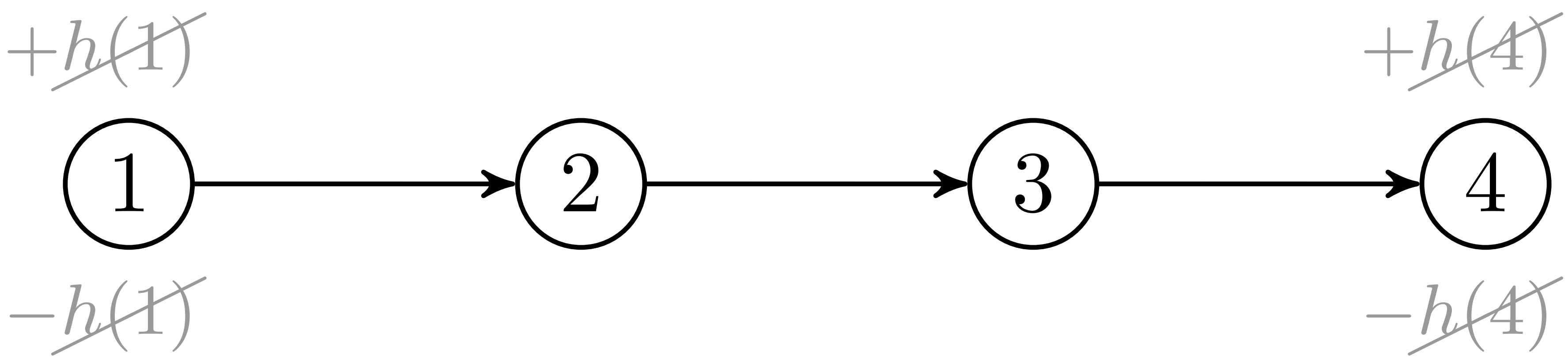
Men disse deles av alle stier mellom disse nodene!



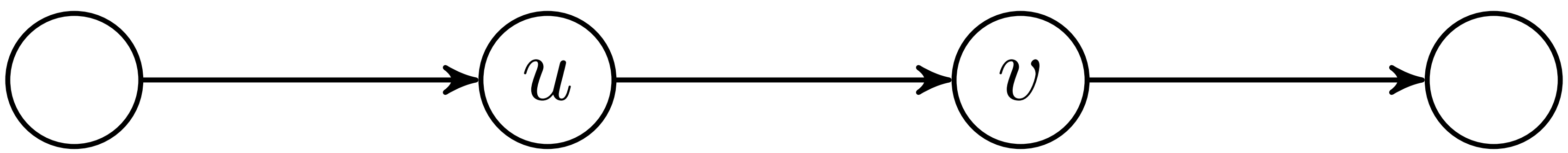
Rekonstruer lengden ved å trekke fra differansen deres



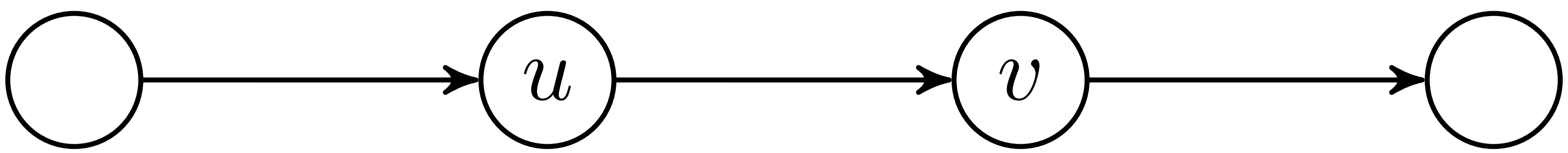
Rekonstruer lengden ved å trekke fra differansen deres



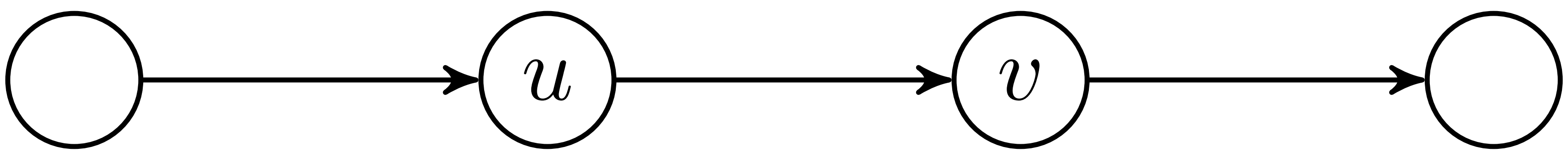
Rekonstruer lengden ved å trekke fra differansen deres



Hvordan sikrer vi ikke-negative vekter?

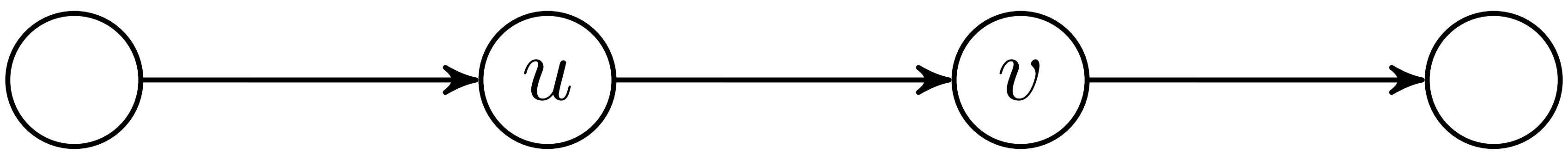


Vi må ha $w(u, v) + h(u) - h(v) \geqslant 0$, dvs. $w(u, v) + h(u) \geqslant h(v)$



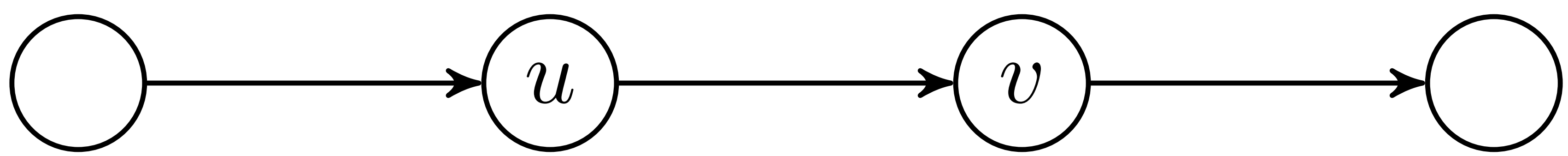
$$w(u, v) + h(u) \geqslant h(v)$$

Vi må ha $w(u, v) + h(u) - h(v) \geqslant 0$, dvs. $w(u, v) + h(u) \geqslant h(v)$



$$w(u,v) + h(u) \geqslant h(v)$$

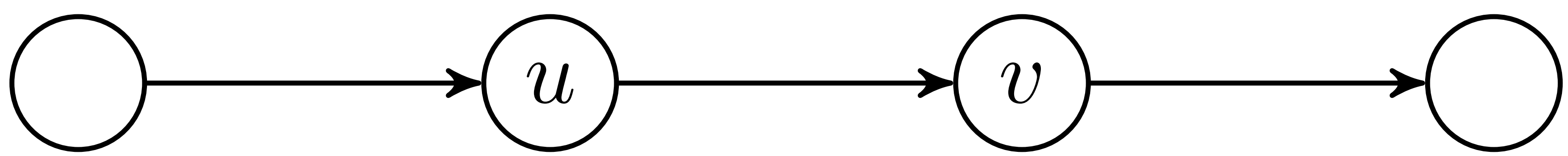
Fra én-til-alle: $\delta(s,v) \leqslant \delta(s,u) + w(u,v)$. Kan la $h(v) = \delta(s,v)$!



$$w(u, v) + h(u) \geqslant h(v)$$

$$w(u, v) + \delta(s, u) \geqslant \delta(s, v)$$

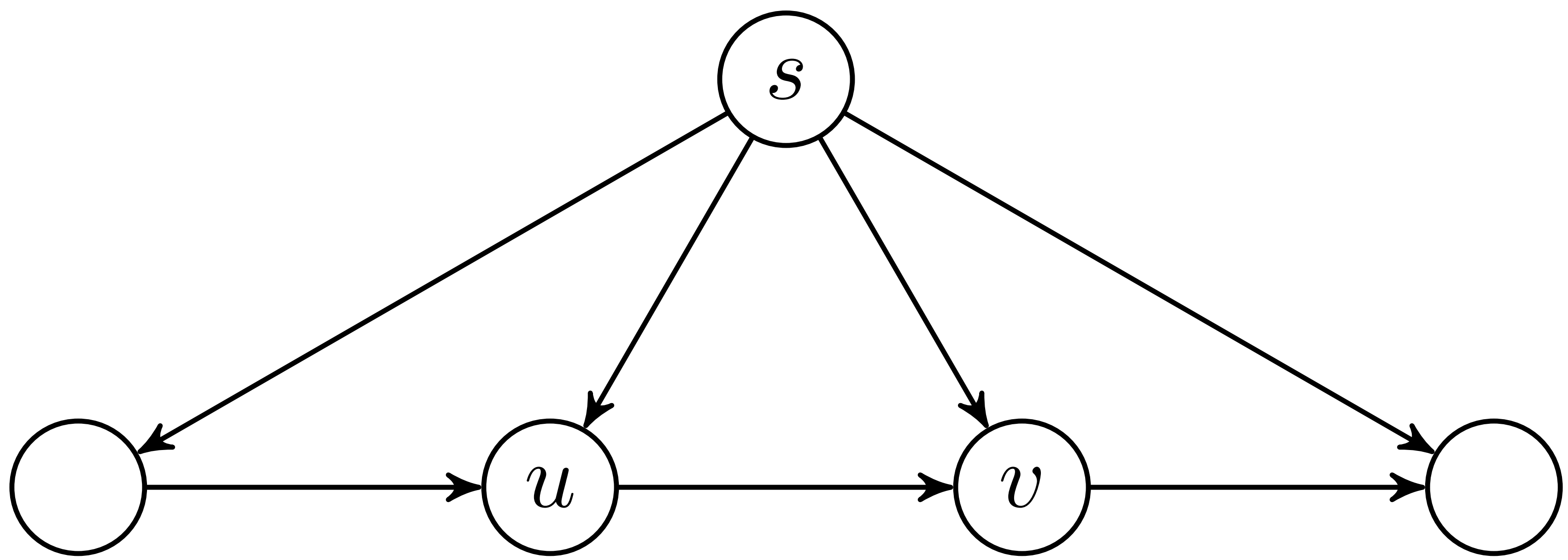
Men hva er s? Vi må sikre at vi når alle…



$$w(u, v) + h(u) \geqslant h(v)$$

$$w(u, v) + \delta(s, u) \geqslant \delta(s, v)$$

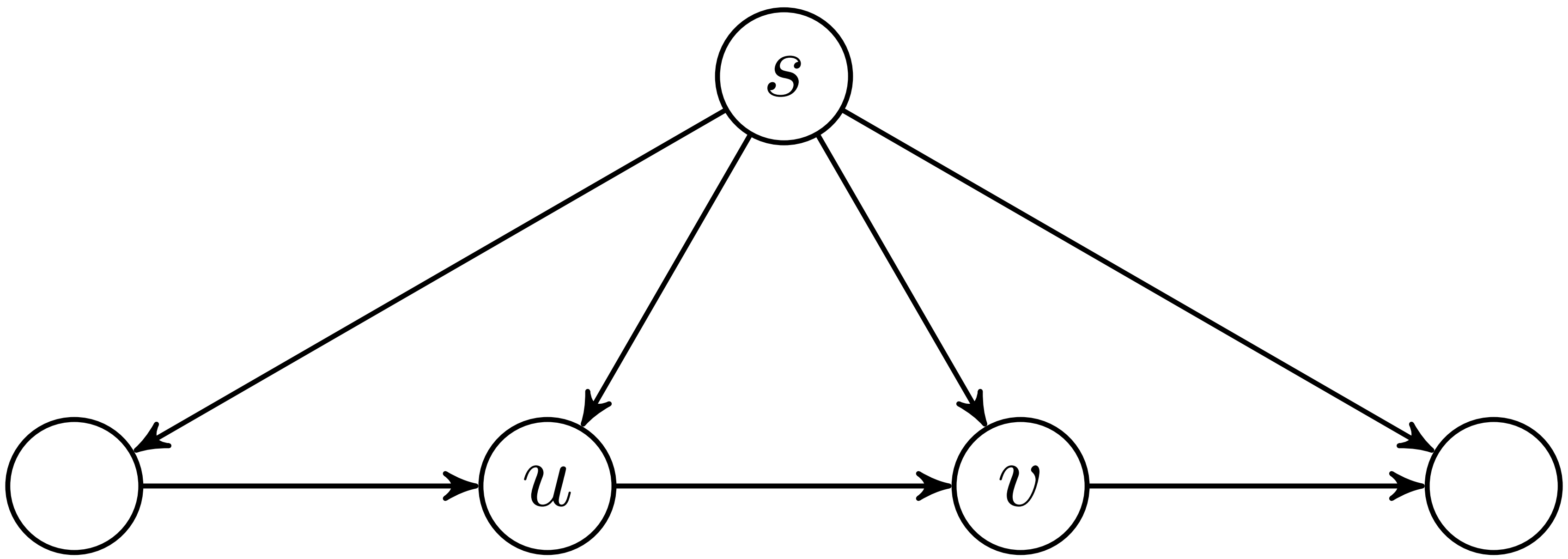
Vi kan legge til en ny node!



$$w(u,v) + h(u) \geqslant h(v)$$

$$w(u,v) + \delta(s,u) \geqslant \delta(s,v)$$

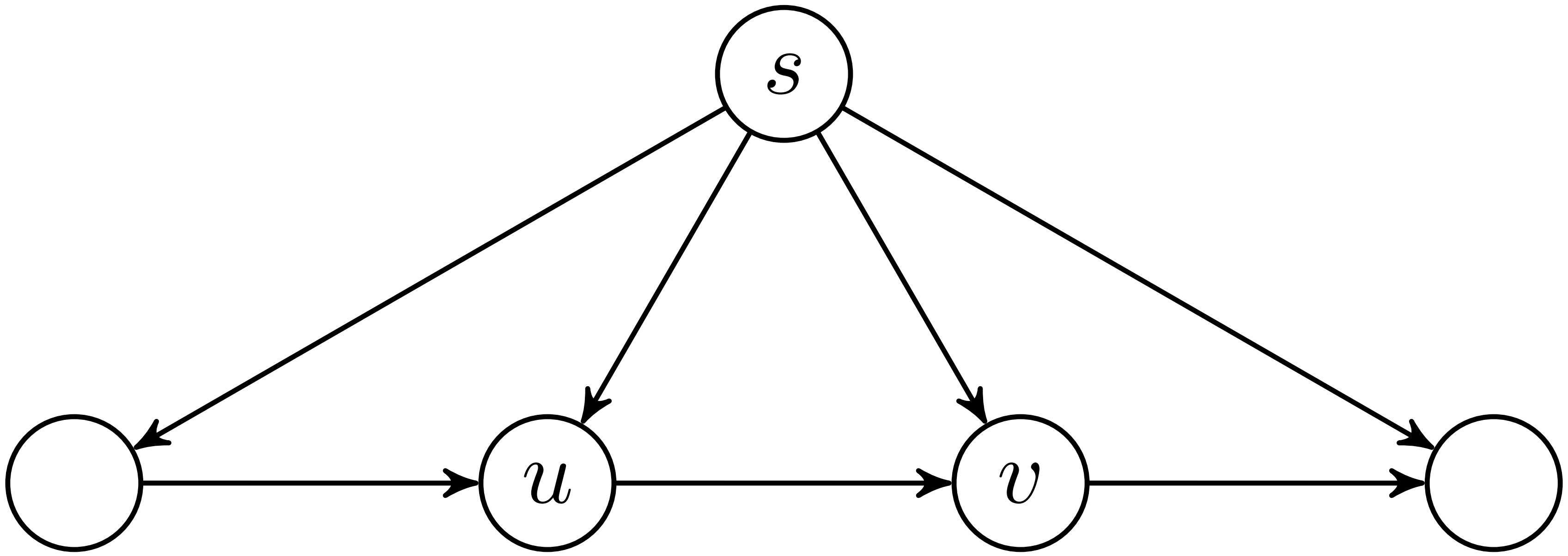
Vi kan legge til en ny node!



$$w(u, v) + h(u) \geqslant h(v)$$

$$w(u, v) + \delta(s, u) \geqslant \delta(s, v)$$

Kantene fra s kan f.eks. få vekt 0



$$w(u, v) + h(u) \geqslant h(v)$$

$$w(u, v) + \delta(s, u) \geqslant \delta(s, v)$$

(Merk: Vi verken innfører eller fjerner negative sykler)

Forenkling: Antar at vi ikke har negative sykler, heller enn å sjekke for det – dvs., jeg ignorerer returverdien fra Bellman-Ford. Vil normalt avbryte om den er false (som de gjør i boka).

$\textsc{Johnson}(G, w)$

G graf
w vekting

Alle til alle. Tillater negative vekter

JOHNSON(G, w)
1 construct G' with start node s

G	graf
w	vekting

Skal brukes til forarbeide med BELLMAN-FORD

JOHNSON(G, w)
1 construct G′ with start node s
2 BELLMAN-FORD(G′, w, s)

G graf
w vekting

Forarbeide: Gir oss info vi trenger til h

JOHNSON(G, w)
1 construct G' with start node s
2 BELLMAN-FORD(G', w, s)
3 **for** each vertex $v \in G.V$

G	graf
w	vekting
v	node

(Cormen et al. bruker her G', men s er overflødig)

JOHNSON(G, w)
construct G′ with start node s
BELLMAN-FORD(G′, w, s)
for each vertex $v \in$ G.V
    $h(v) = v.d$

G graf
w vekting

v node
h $\delta(s, v)$

Her er $v.d = \delta(s, v)$, beregnet av BELLMAN-FORD

```
JOHNSON(G, w)
  construct G′ with start node s
  BELLMAN-FORD(G′, w, s)
  for each vertex v ∈ G.V
      h(v) = v.d
  for each edge (u, v) ∈ G.E
```

G	graf
w	vekting
u	node
v	node
h	$\delta(s, v)$

(Cormen et al. bruker her G′, men s er overflødig)

JOHNSON(G, w)

construct G' with start node s
BELLMAN-FORD(G', w, s)
for each vertex $v \in \mathrm{G.V}$
  $h(v) = v.d$
for each edge $(u, v) \in \mathrm{G.E}$
  $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$

G	graf
w	vekting
u	node
v	node
h	$\delta(s, v)$
$\hat{w}$	ny vekting

$$h(v) \leqslant h(u) + w(u, v) \implies \hat{w}(u, v) \geqslant 0$$

JOHNSON(G, w)

```
construct G′ with start node s
BELLMAN-FORD(G′, w, s)
for each vertex v ∈ G.V
    h(v) = v.d
for each edge (u, v) ∈ G.E
    ŵ(u, v) = w(u, v) + h(u) − h(v)
let D = (d_uv) be a new n × n matrix
```

G	graf
w	vekting
u	node
v	node
h	$\delta(s, v)$
$\hat{w}$	ny vekting
d_{uv}	$\delta(u, v)$

Resultat: Alle til alle med DIJKSTRA

mlh@ntnu.no

JOHNSON(G, w)

construct G′ with start node s
BELLMAN-FORD(G′, w, s)
for each vertex $v \in$ G.V
    $h(v) = v.d$
for each edge $(u, v) \in$ G.E
    $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$
let D $= (d_{uv})$ be a new $n \times n$ matrix
for each vertex $u \in$ G.V

G	graf
w	vekting
u	node
v	node
h	$\delta(s, v)$
$\hat{w}$	ny vekting
d_{uv}	$\delta(u, v)$

For hver u, finn $\delta(u, \cdot\,)$ for $\hat{w}$ med DIJKSTRA

Johnson(G, w)

construct G′ with start node s
Bellman-Ford(G′, w, s)
for each vertex $v \in$ G.V
    $h(v) = v.d$
for each edge $(u, v) \in$ G.E
    $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$
let D $= (d_{uv})$ be a new $n \times n$ matrix
for each vertex $u \in$ G.V
    Dijkstra(G, $\hat{w}$, u)

G	graf
w	vekting
u	node
v	node
h	$\delta(s, v)$
$\hat{w}$	ny vekting
d_{uv}	$\delta(u, v)$

Lovlig, fordi $\hat{w}$ er ikke-negativ

JOHNSON(G, w)

```
construct G′ with start node s
BELLMAN-FORD(G′, w, s)
for each vertex v ∈ G.V
    h(v) = v.d
for each edge (u, v) ∈ G.E
    ŵ(u, v) = w(u, v) + h(u) − h(v)
let D = (d_uv) be a new n × n matrix
for each vertex u ∈ G.V
    DIJKSTRA(G, ŵ, u)
    for each vertex v ∈ G.V
```

G	graf
w	vekting
u	node
v	node
h	$\delta(s, v)$
$\hat{w}$	ny vekting
d_{uv}	$\delta(u, v)$

For hver v, finn $d_{uv} = \delta(u, v)$ for w fra $\delta(u, v)$ for $\hat{w}$

JOHNSON(G, w)

construct G' with start node s
BELLMAN-FORD(G', w, s)
for each vertex $v \in G.V$
    $h(v) = v.d$
for each edge $(u, v) \in G.E$
    $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$
let $D = (d_{uv})$ be a new $n \times n$ matrix
for each vertex $u \in G.V$
    DIJKSTRA($G, \hat{w}, u$)
    for each vertex $v \in G.V$
        $d_{uv} = v.d + h(v) - h(u)$

G	graf
w	vekting
u	node
v	node
h	$\delta(s, v)$
$\hat{w}$	ny vekting
d_{uv}	$\delta(u, v)$

Teleskopsum med $-h(x) + h(x)$ for x mellom u og v

Johnson(G, w)

construct G' with start node s
Bellman-Ford(G', w, s)
for each vertex $v \in \mathrm{G.V}$
    $h(v) = v.d$
for each edge $(u, v) \in \mathrm{G.E}$
    $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$
let $\mathrm{D} = (d_{uv})$ be a new $n \times n$ matrix
for each vertex $u \in \mathrm{G.V}$
    Dijkstra$(\mathrm{G}, \hat{w}, u)$
    for each vertex $v \in \mathrm{G.V}$
        $d_{uv} = v.d + h(v) - h(u)$
return D

G	graf
w	vekting
u	node
v	node
h	$\delta(s, v)$
$\hat{w}$	ny vekting
d_{uv}	$\delta(u, v)$

Rett fordi $w(p) \leqslant w(q) \iff \hat{w}(p) \leqslant \hat{w}(q)$ for stier $u \overset{p,q}{\leadsto} v$

Johnson(G, w)

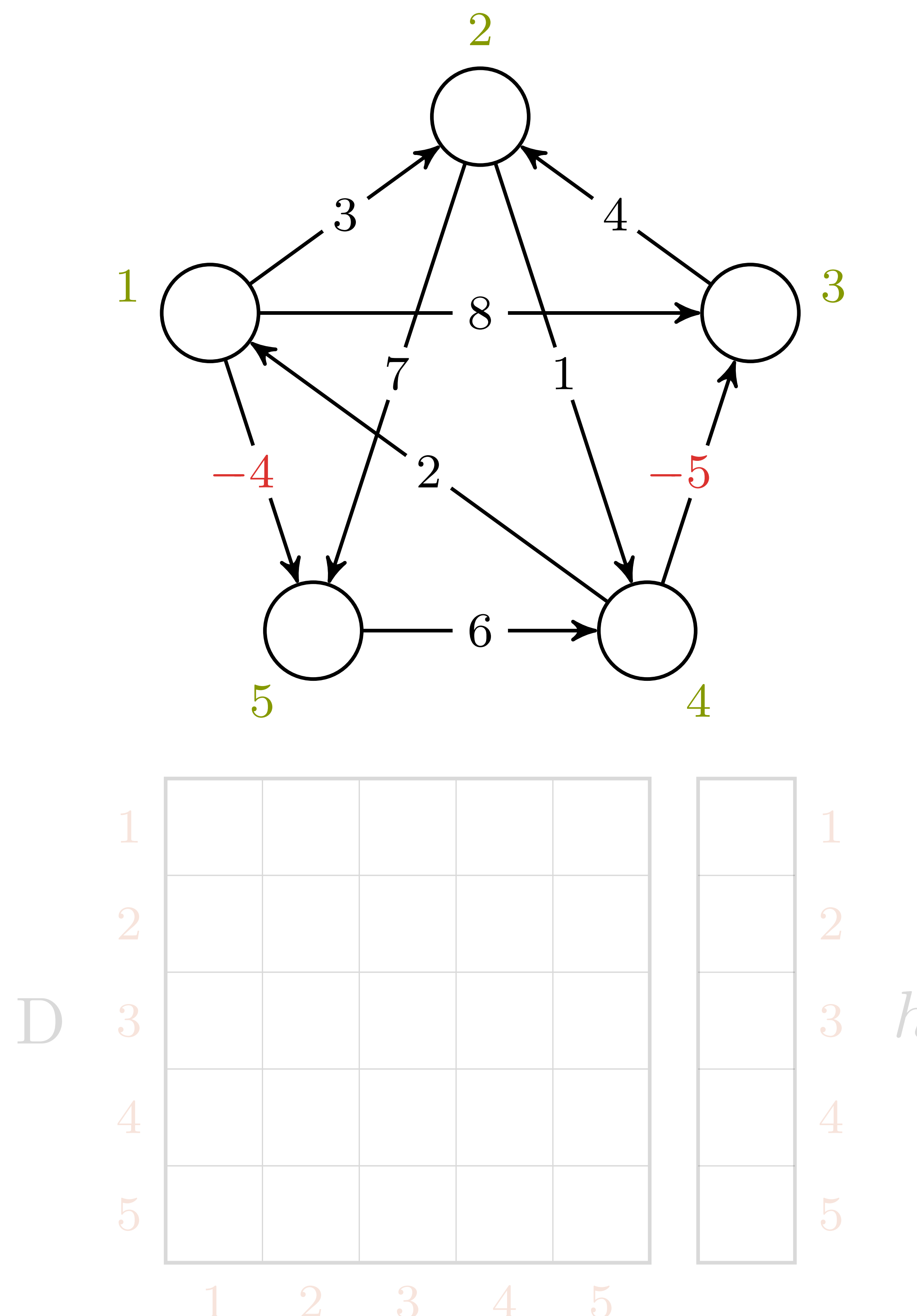
construct G' with start node s
Bellman-Ford(G', w, s)
for each vertex $v \in G.V$
    $h(v) = v.d$
for each edge $(u, v) \in G.E$
    $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$
let $D = (d_{uv})$ be a new $n \times n$ matrix
for each vertex $u \in G.V$
    Dijkstra($G, \hat{w}, u$)
    for each vertex $v \in G.V$
        $d_{uv} = v.d + h(v) - h(u)$
return D

G	graf
w	vekting
u	node
v	node
h	$\delta(s, v)$
$\hat{w}$	ny vekting
d_{uv}	$\delta(u, v)$

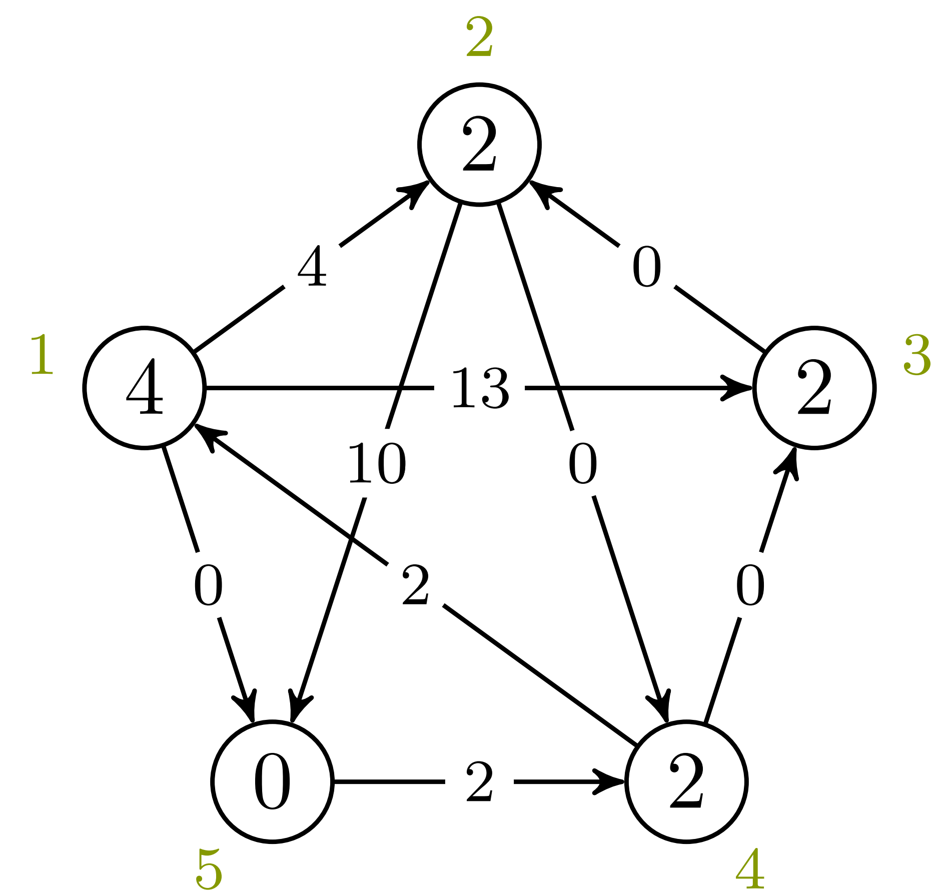
mlh@ntnu.no

JOHNSON(G, w)

construct G' with start node s
BELLMAN-FORD(G', w, s)
for each vertex $v \in G.V$
    $h(v) = v.d$
for each edge $(u, v) \in G.E$
    $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$
let $D = (d_{uv})$ be a new $n \times n$ matrix
for each vertex $u \in G.V$
    DIJKSTRA$(G, \hat{w}, u)$
    for each vertex $v \in G.V$
        $d_{uv} = v.d + h(v) - h(u)$
return D



```
JOHNSON(G, w)
construct G′ with start node s
BELLMAN-FORD(G′, w, s)
for each vertex v ∈ G.V
    h(v) = v.d
for each edge (u, v) ∈ G.E
    ŵ(u, v) = w(u, v) + h(u) − h(v)
let D = (d_uv) be a new n × n matrix
for each vertex u ∈ G.V
    DIJKSTRA(G, ŵ, u)
    for each vertex v ∈ G.V
        d_uv = v.d + h(v) − h(u)
return D
```



D

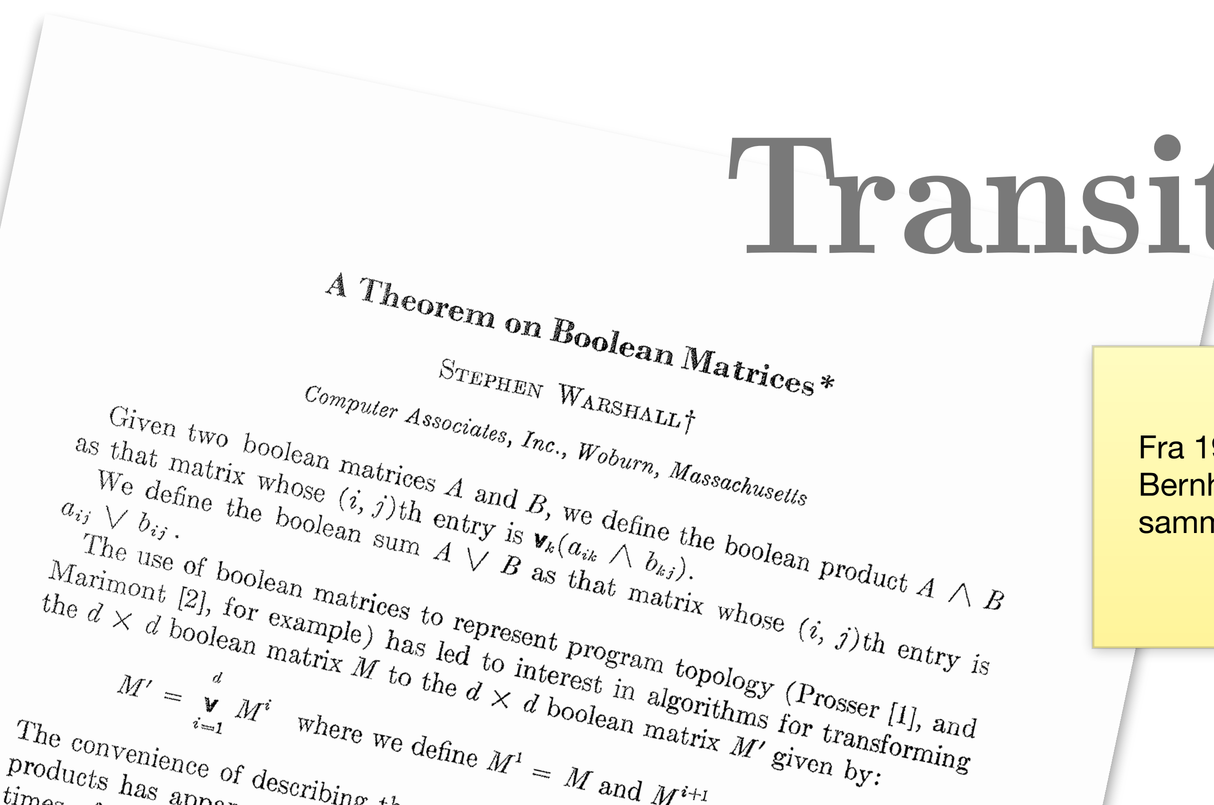
	1	2	3	4	5
1	0	1	−3	2	−4
2	3	0	−4	1	−1
3	7	4	0	5	3
4	2	−1	−5	0	−2
5	8	5	1	6	0

h

	h
1	0
2	−1
3	−5
4	0
5	−4

2:3

Transitiv lukning


A Theorem on Boolean Matrices*

Stephen Warshall†

Computer Associates, Inc., Woburn, Massachusetts

Given two boolean matrices A and B, we define the boolean product $A \wedge B$ as that matrix whose (i, j)th entry is $\vee_k(a_{ik} \wedge b_{kj})$.

We define the boolean sum $A \vee B$ as that matrix whose (i, j)th entry is $a_{ij} \vee b_{ij}$.

The use of boolean matrices to represent program topology (Prosser [1], and Marimont [2], for example) has led to interest in algorithms for transforming the $d \times d$ boolean matrix M to the $d \times d$ boolean matrix M' given by:

$M' = \bigvee_{i=1}^{d} M^i$ where we define $M^1 = M$ and M^{i+1}

The convenience of describing th

products has appe

Fra 1960 (publisert 1962).
Bernhard Roy publiserte
samme resultat separat i 1959.

Input: En rettet graf $G = (V, E)$.

Output: En rettet graf $G^* = (V, E^*)$ der $(i, j) \in E^*$ hvis og bare hvis det finnes en sti fra i til j i G.

Vi kommer til å returnere en nabomatrise T for G^*.

Traversér fra hver node?

- Kjøretid: $V \times \Theta(E+V) = \Theta(VE+V^2)$
- Bra når vi har få kanter, f.eks. $E = o(V^2)$
- Mye overhead; høye konstantledd

Målsetting:

- Vi fokuserer på tilfellet $E = \Theta(V^2)$
- Vi vil ha et lavere konstantledd

Observasjon:

- Korteste stier har felles segmenter
- Overlappende delproblemer …

Dekomponering: Hva blir «koordinatene» til delproblemene?

$$i$$

Det finnes en vei fra i . . .

$$i \quad j$$

Det finnes en vei fra i til j . . .

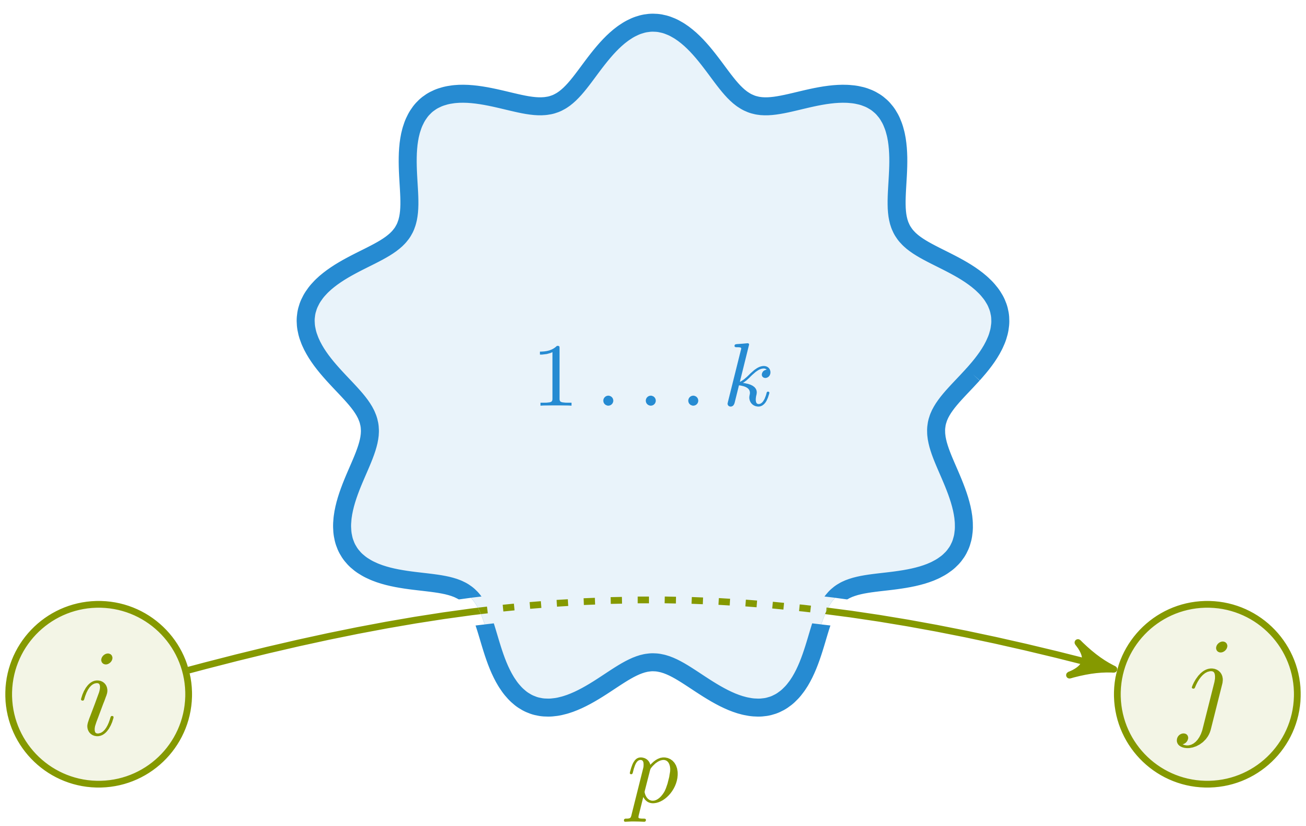
$$i\,j\,k$$

$$ijk$$

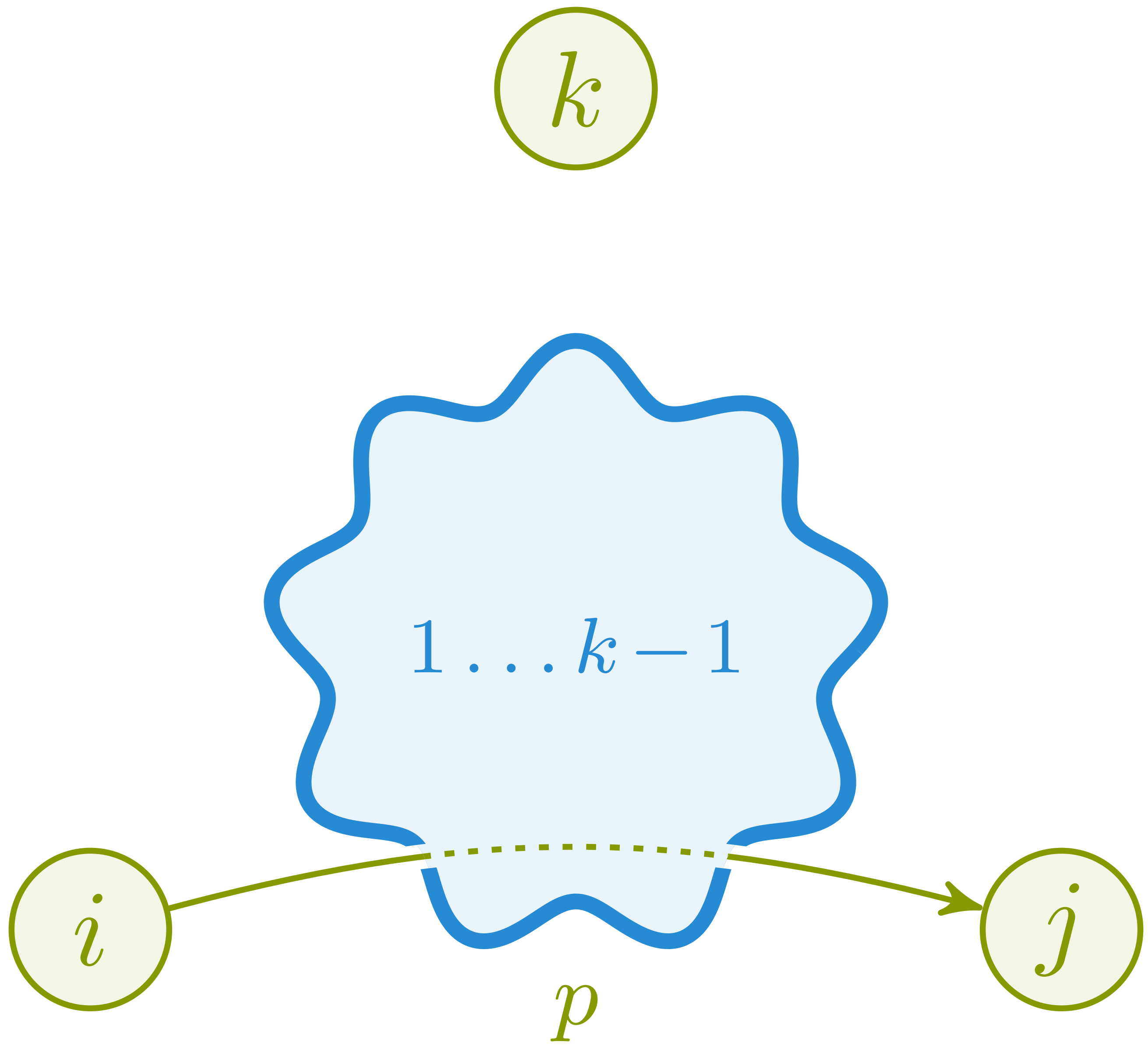
Det finnes en vei fra i til j via noder fra $\{1 \dots k\}$

$$t_{ij}^{(k)}$$

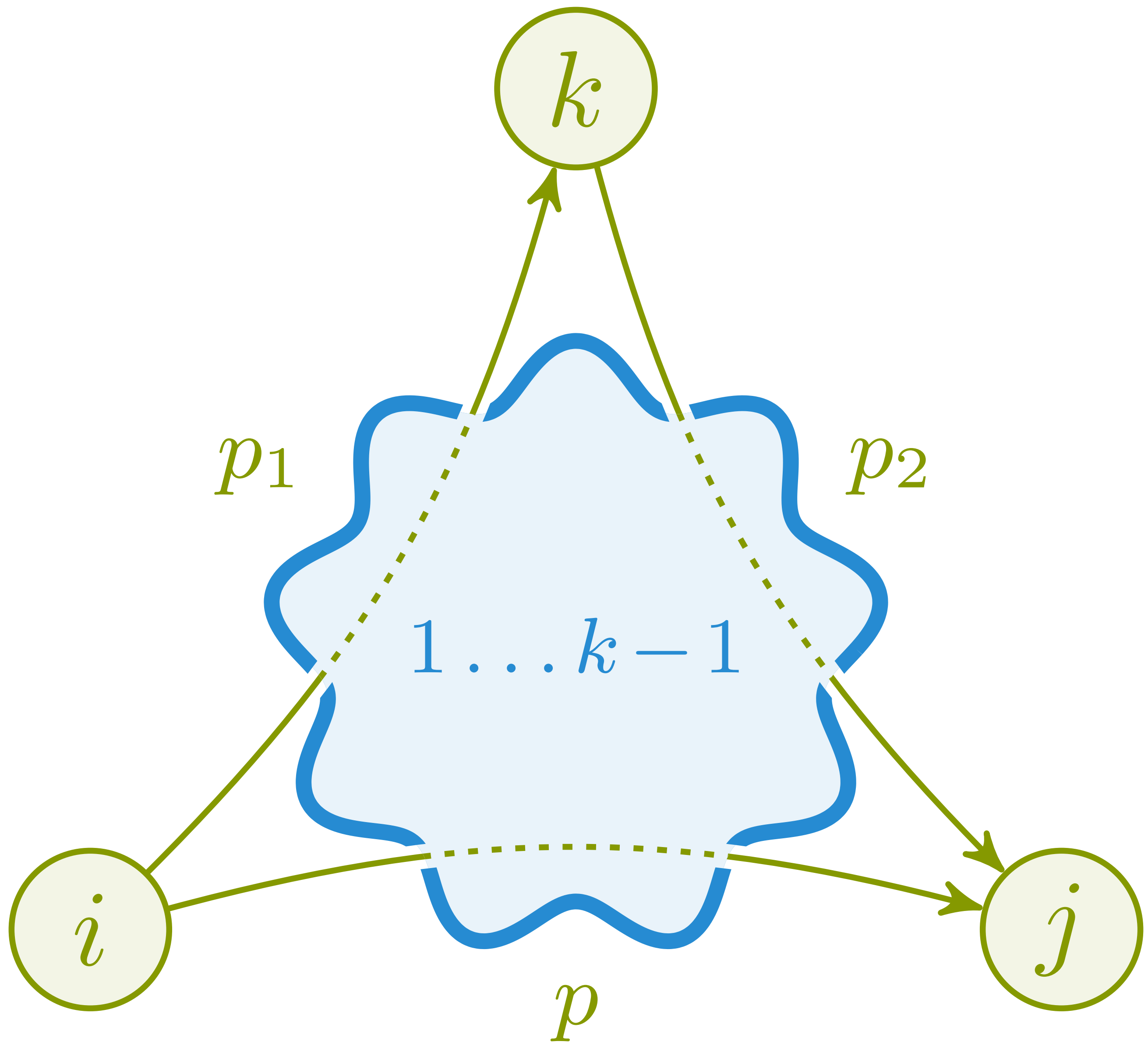
Det finnes en vei fra i til j via noder fra $\{1 \dots k\}$



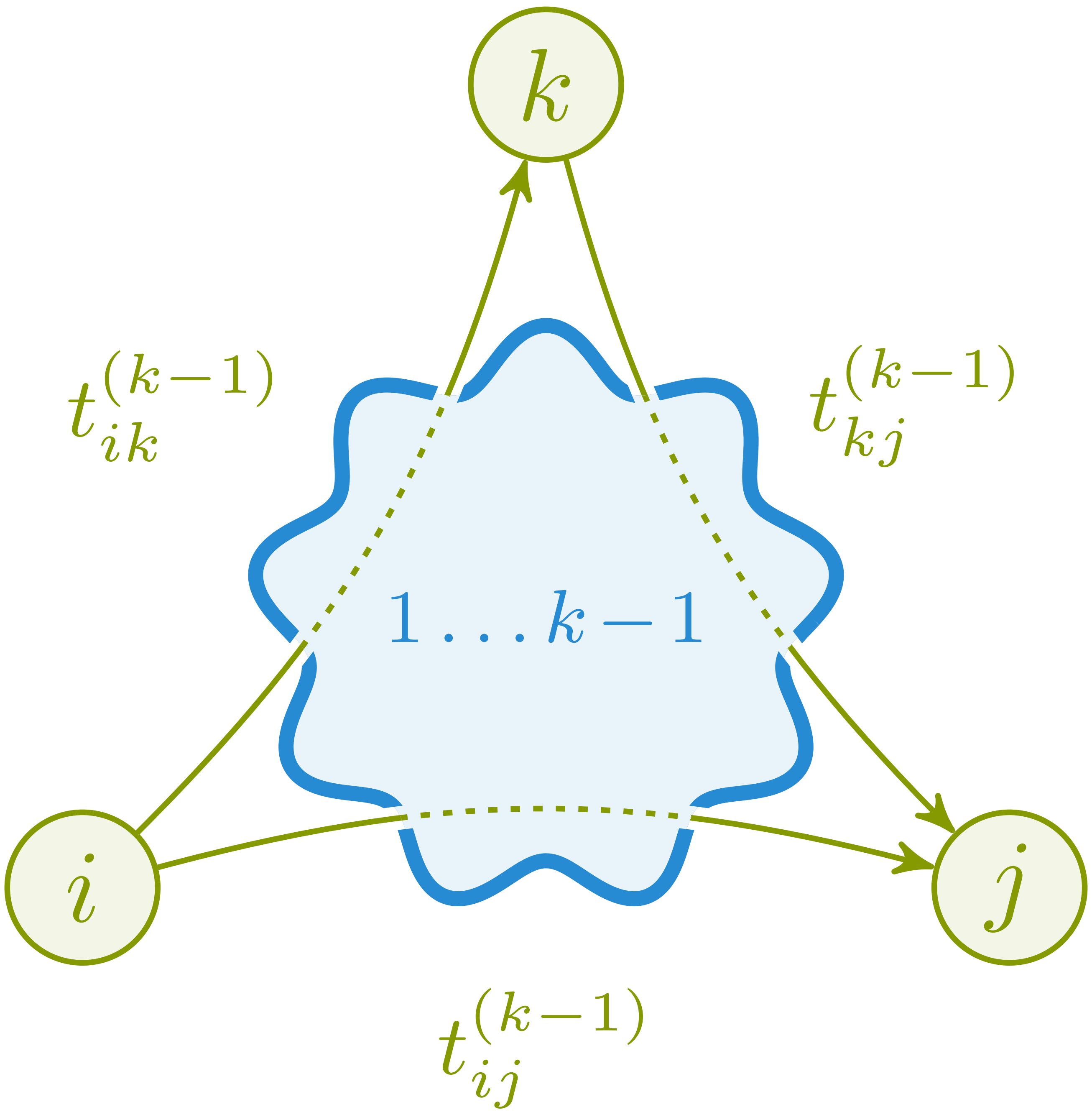
$t_{ij}^{(k)}$ = det går en vei fra i til j via noder fra $\{1 \ldots k\}$

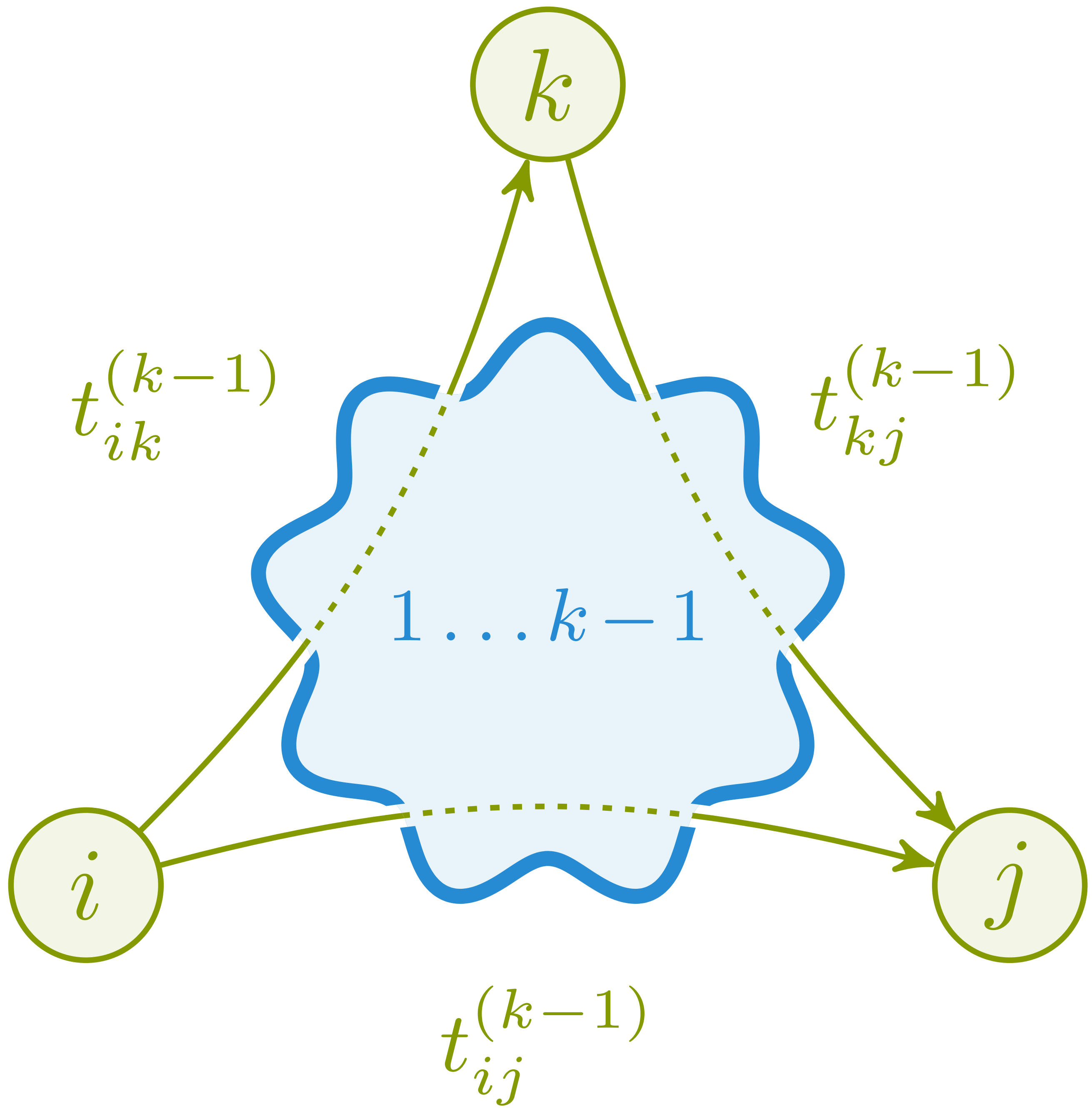


Som for ryggsekkproblemet: Skal k være med eller ikke?



De mulige stiene p, p_1 og p_2 går kun via noder fra $\{1 \ldots k-1\}$


k
$t_{ik}^{(k-1)}$
$t_{kj}^{(k-1)}$
$1 \ldots k-1$
i
j
$t_{ij}^{(k-1)}$



$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}\right)$$

$$t_{ij}^{(0)} = \begin{cases} 0 & \text{if } i \neq j \text{ and } (i, j) \notin \mathrm{E}\,, \\ 1 & \text{if } i = j \text{ or } (i, j) \in \mathrm{E}\,. \end{cases}$$

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}\right)$$

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}\right)$$

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}\right)$$

Problemet er at iterasjonene 1…k «blandes», så vi kan risikere at noen av del-stiene allerede går innom k – så kanskje vi går innom k mer enn én gang? I så fall har vi en sykel … men om det finnes en sti *med* en sykel, så finnes det også en sti *uten* en sykel!

Det er trygt å bruke én tabell. Hvorfor?

TRANSITIVE-CLOSURE(G)

G graf

For hver node i og j: Finnes det en sti $i \rightsquigarrow j$?

TRANSITIVE-CLOSURE(G)
1 $n = |G.V|$

G	graf
n	ant. noder

TRANSITIVE-CLOSURE(G)
1 $n = |G.V|$
2 let $T^{(0)} = (t_{ij}^{(0)})$ be a new $n \times n$ matrix

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1\cdots k}{\rightsquigarrow} j$?

Finnes en sti $i \rightsquigarrow j$ som ikke går via andre noder?

TRANSITIVE-CLOSURE(G)
1 $n = |\mathrm{G.V}|$
2 let $\mathrm{T}^{(0)} = (t_{ij}^{(0)})$ be a new $n \times n$ matrix
3 **for** $i = 1$ **to** n

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1 \cdots k}{\leadsto} j$?

For hver mulig startnode…

TRANSITIVE-CLOSURE(G)

$n = |\text{G.V}|$

let $\text{T}^{(0)} = (t_{ij}^{(0)})$ be a new $n \times n$ matrix

for $i = 1$ **to** n

  for $j = 1$ **to** n

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1\cdots k}{\leadsto} j$?

For hver mulig sluttnode . . .

TRANSITIVE-CLOSURE(G)

$n = |\mathrm{G.V}|$
let $\mathrm{T}^{(0)} = (t_{ij}^{(0)})$ be a new $n \times n$ matrix
for $i = 1$ **to** n
    for $j = 1$ **to** n
        if $i == j$ or $(i, j) \in \mathrm{G.E}$

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1\cdots k}{\rightsquigarrow} j\,?$

Samme node eller sammenkoblet med kant?

TRANSITIVE-CLOSURE(G)

$n = |G.V|$
let $T^{(0)} = (t_{ij}^{(0)})$ be a new $n \times n$ matrix
for $i = 1$ **to** n
    for $j = 1$ **to** n
        if $i == j$ or $(i, j) \in G.E$
            $t_{ij}^{(0)} = 1$

G graf
n ant. noder
$t_{ij}^{(k)}$ $i \overset{1\cdots k}{\leadsto} j$?

Da finnes det en sti $i \leadsto j$ som ikke går via andre noder

```
Transitive-Closure(G)
n = |G.V|
let T^(0) = (t_ij^(0)) be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        if i == j or (i, j) ∈ G.E
            t_ij^(0) = 1
        else t_ij^(0) = 0
```

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1\cdots k}{\rightsquigarrow} j$?

Ellers finnes ingen slik sti

TRANSITIVE-CLOSURE(G)

$n = |\text{G.V}|$
let $\text{T}^{(0)} = (t_{ij}^{(0)})$ be a new $n \times n$ matrix
for $i = 1$ **to** n
    for $j = 1$ **to** n
        if $i == j$ or $(i, j) \in \text{G.E}$
            $t_{ij}^{(0)} = 1$
        else $t_{ij}^{(0)} = 0$
$\ldots$

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1\cdots k}{\rightsquigarrow} j\,?$

Grunntilfelle på plass

TRANSITIVE-CLOSURE(G)

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1\cdots k}{\rightsquigarrow} j\,?$

Transitive-Closure(G)

7 …

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1\cdots k}{\rightsquigarrow} j\,?$

Transitive-Closure(G)

7 . . .

8 **for** $k = 1$ **to** n

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1\cdots k}{\leadsto} j$?

Vi får nå lov til å gå innom node k

Transitive-Closure(G)

```
7  ...
8  for k = 1 to n
9      let T^(k) = (t_ij^(k)) be a new n × n matrix
```

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1\cdots k}{\rightsquigarrow} j$?

Finnes en sti $i \rightsquigarrow j$ som kun får gå innom $\{1, \ldots, k\}$?

```
Transitive-Closure(G)
...
for k = 1 to n
    let T^(k) = (t_ij^(k)) be a new n × n matrix
    for i = 1 to n
```

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1\cdots k}{\rightsquigarrow} j\,?$

For hver mulig startnode . . .

```
TRANSITIVE-CLOSURE(G)
...
for k = 1 to n
        let T^(k) = (t_ij^(k)) be a new n × n matrix
        for i = 1 to n
                for j = 1 to n
```

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1\cdots k}{\rightsquigarrow} j\,?$

For hver mulig sluttnode…

Transitive-Closure(G)

...
for $k = 1$ **to** n
  let $T^{(k)} = (t_{ij}^{(k)})$ be a new $n \times n$ matrix
  for $i = 1$ **to** n
    for $j = 1$ **to** n
      $t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1\cdots k}{\rightsquigarrow} j$?

Finnes $i \rightsquigarrow j$ eller $i \rightsquigarrow k \rightsquigarrow j$, om vi kun får gå innom $\{1, \ldots, k-1\}$?

Transitive-Closure(G)

…
for $k = 1$ **to** n
    let $T^{(k)} = (t_{ij}^{(k)})$ be a new $n \times n$ matrix
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            $t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$
return $T^{(n)}$

G	graf
n	ant. noder
$t_{ij}^{(k)}$	$i \overset{1\cdots k}{\rightsquigarrow} j$?

Finnes det en sti $i \rightsquigarrow j$ som får gå innom $\{1, \ldots, n\}$, dvs. alle?

Transitive-Closure′(G)

Forenklet utgave: Bruker bare én tabell, T

TRANSITIVE-CLOSURE′(G)
1 $n = |G.V|$

TRANSITIVE-CLOSURE′(G)
1 $n = |G.V|$
2 initialize T

Samme node (I er identitetsmatrise) eller nabo (A er nabomatrise)

TRANSITIVE-CLOSURE′(G)
1 $n = |\text{G.V}|$
2 initialize T
3 **for** $k = 1$ **to** n

Vi får nå lov til å gå innom node k

TRANSITIVE-CLOSURE′(G)

1 $n = |G.V|$
2 initialize T
3 **for** $k = 1$ **to** n
4 **for** $i = 1$ **to** n

For hver mulig startnode . . .

Transitive-Closure′(G)

$n = |G.V|$
  initialize T
  for $k = 1$ **to** n
    for $i = 1$ **to** n
      for $j = 1$ **to** n

For hver mulig sluttnode . . .

TRANSITIVE-CLOSURE′(G)

$n = |G.V|$
initialize T
for $k = 1$ **to** n
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$

Enten allerede en sti $i \rightsquigarrow j$ eller en ny sti $i \rightsquigarrow k \rightsquigarrow j$?

TRANSITIVE-CLOSURE′(G)

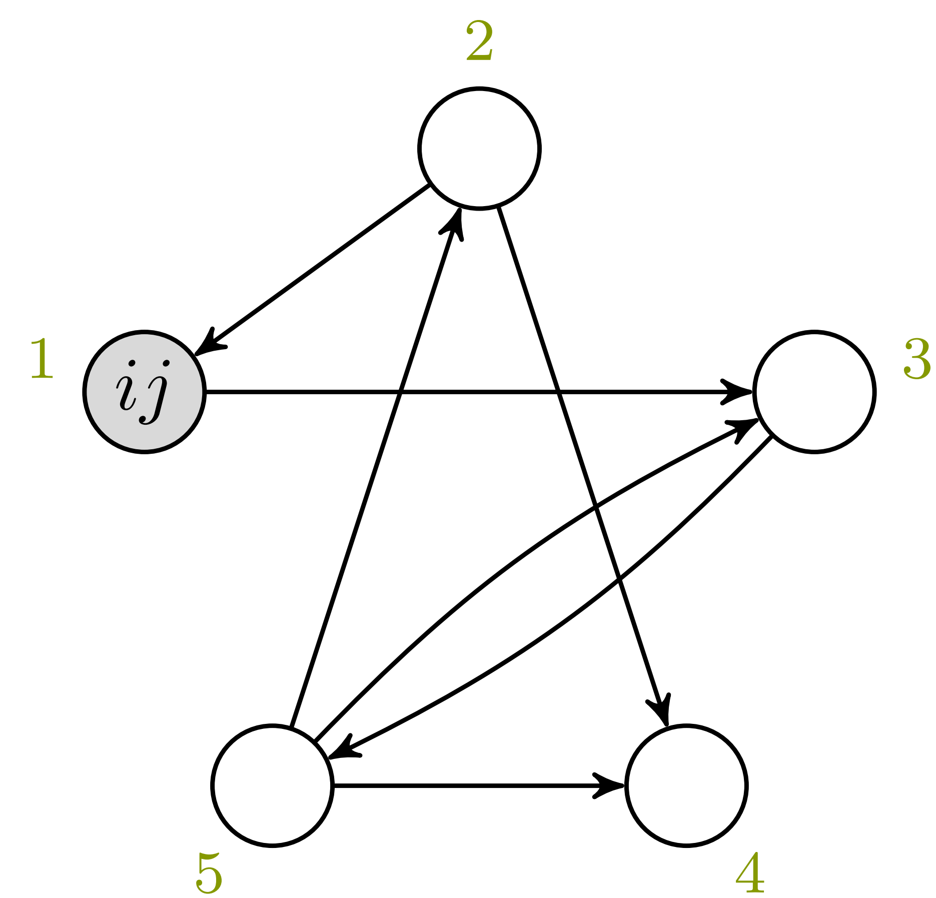
$n = |G.V|$
initialize T
for $k = 1$ **to** n
  for $i = 1$ **to** n
    for $j = 1$ **to** n
      $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$
return T

Finnes det en sti $i \rightsquigarrow j$?

TRANSITIVE-CLOSURE′(G)

```
n = |G.V|
initialize T
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            t_ij = t_ij ∨ (t_ik ∧ t_kj)
return T
```

$k, i, j \;=\; 1, 1, 1$

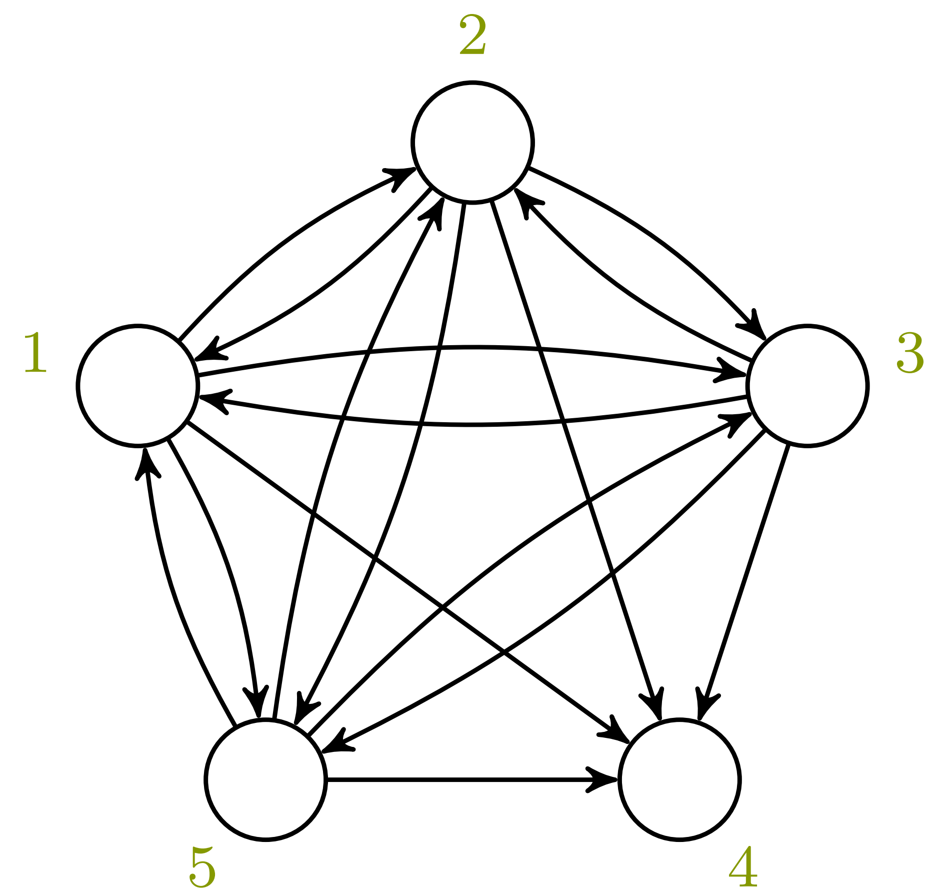


	1	2	3	4	5
1	1		1		
2	1	1		1	
3			1		1
4				1	
5		1	1	1	1

Transitive-Closure′(G)

```
n = |G.V|
initialize T
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            t_ij = t_ij ∨ (t_ik ∧ t_kj)
return T
```

$k, i, j \;=\; \text{-}, \text{-}, \text{-}$



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

Korteste vei › Transitiv lukning › Kjøretid

init › $\Theta(n^2)$

for $i = 1$ **to** n

  for $j = 1$ **to** n

    sett $t_{ij}^{(0)}$ › $O(1)$

for $k = 1$ **to** n

  evt. ny matrise › $\Theta(n^2)$

  for $i = 1$ **to** n

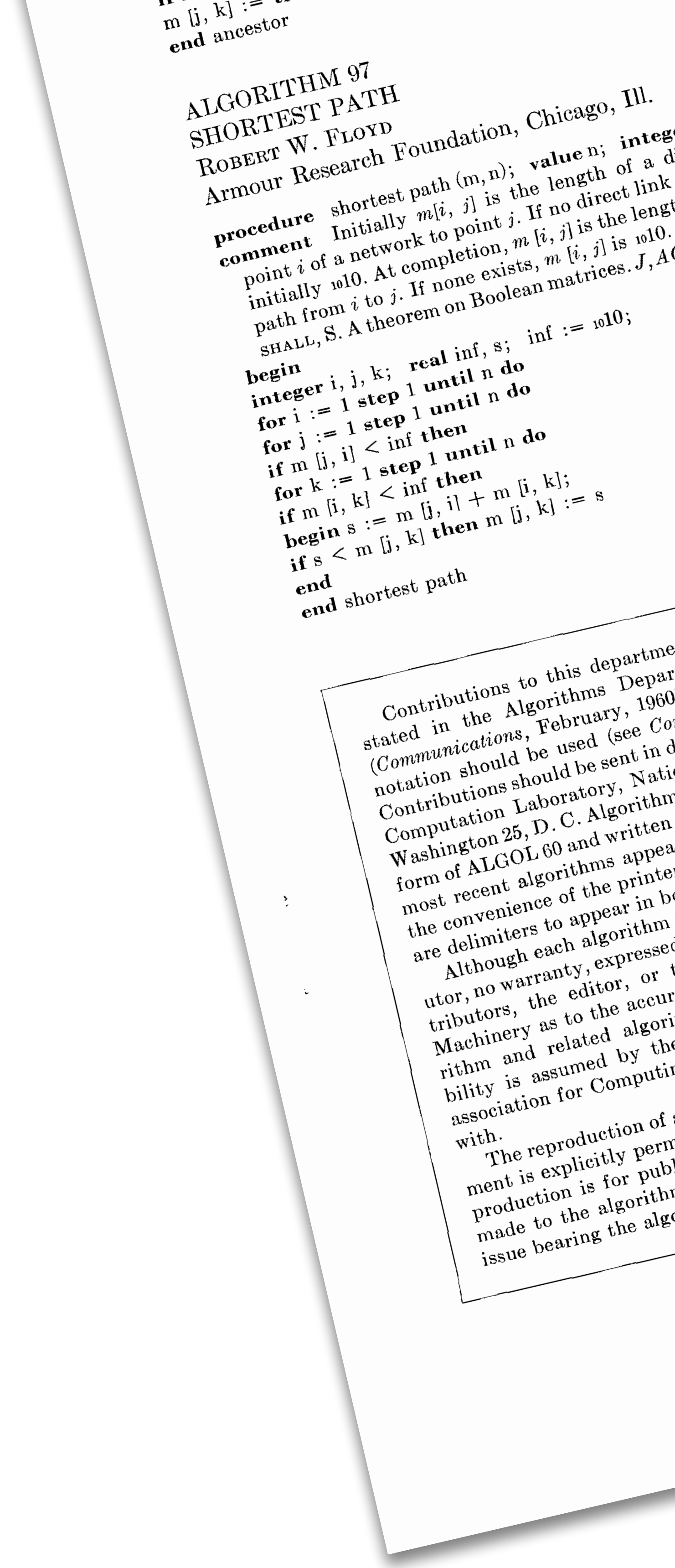
    for $j = 1$ **to** n

      sett $t_{ij}^{(k)}$ › $O(1)$

return › $O(1)$

Totalt: $\Theta(n^3)$

Fra 1962.



m [j, k] := ...
end ancestor

ALGORITHM 97
SHORTEST PATH
Robert W. Floyd
Armour Research Foundation, Chicago, Ill.

```
procedure shortest path (m,n); value n; integ
comment Initially m[i, j] is the length of a d
  point i of a network to point j. If no direct link
  initially ₁₀10. At completion, m [i, j] is the lengt
  path from i to j. If none exists, m [i, j] is ₁₀10.
  SHALL, S. A theorem on Boolean matrices. J, AC
begin
integer i, j, k; real inf, s; inf := ₁₀10;
for i := 1 step 1 until n do
for j := 1 step 1 until n do
if m [j, i] < inf then
for k := 1 step 1 until n do
if m [i, k] < inf then
begin s := m [j, i] + m [i, k];
if s < m [j, k] then m [j, k] := s
end
end shortest path
```

Contributions to this departme
stated in the Algorithms Depar
(*Communications*, February, 1960
notation should be used (see Co
Contributions should be sent in d
Computation Laboratory, Natio
Washington 25, D. C. Algorithm
form of ALGOL 60 and written
most recent algorithms appea
the convenience of the printe
are delimiters to appear in bo
Although each algorithm
utor, no warranty, expressed
tributors, the editor, or t
Machinery as to the accur
rithm and related algori
bility is assumed by the
association for Computin
with.
The reproduction of a
ment is explicitly perm
production is for publ
made to the algorithm
issue bearing the algo

3:3

Floyd-Warshall

Fra hver node til alle andre?

- DIJKSTRA med tabell: $O(V^3 + VE)$
- ... med binærhaug: $O(VE \lg V)$
- ... med Fib.-haug: $O(V^2 \lg V + VE)$
- BELLMAN-FORD: $\Theta(V^2 E)$

Målsetting:

- Vi vil tillate negative kanter
- Vi vil ha lavere asymptotisk kjøretid ...
- ... og vil ha lavere konstantledd

Nok en gang: Hva blir «koordinatene» til delproblemene?

$$i\,j\,k$$

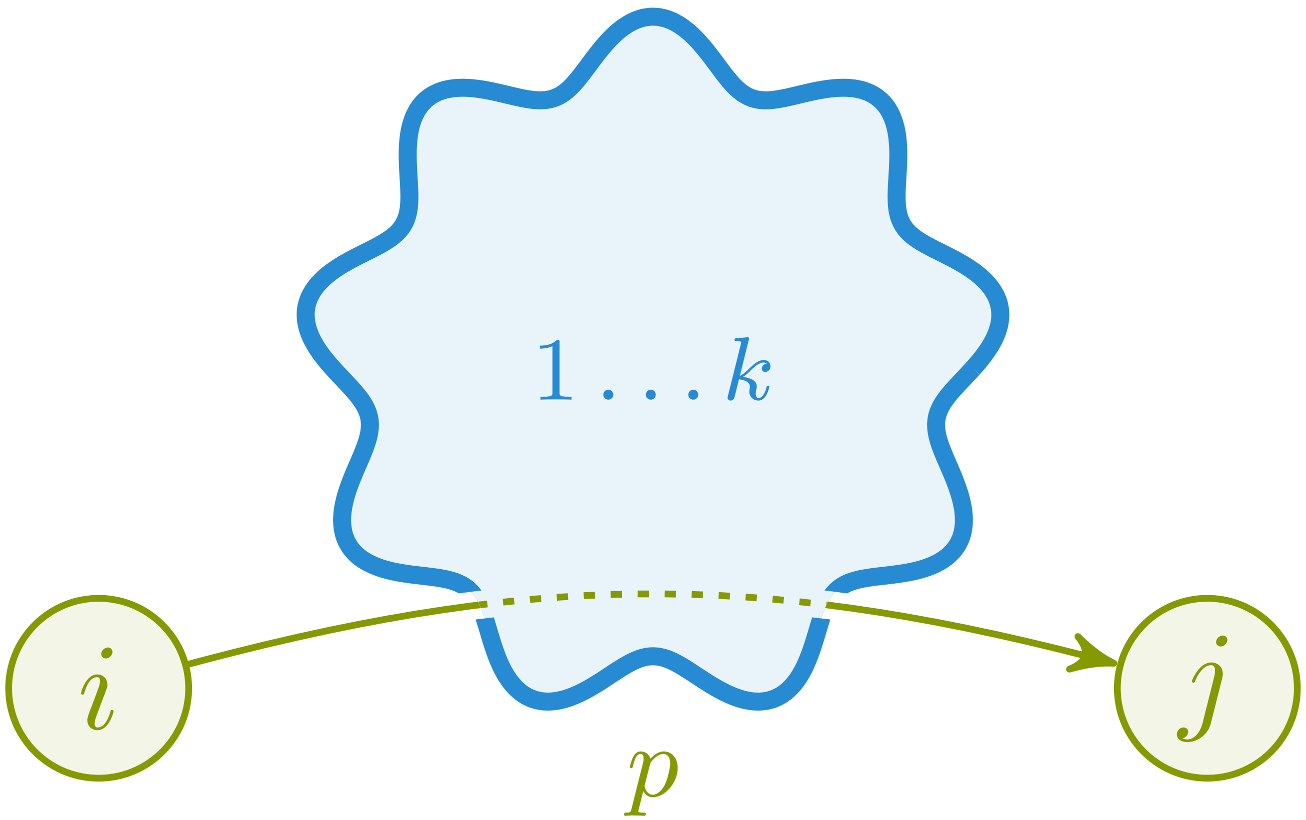
Korteste vei fra i til j via noder fra $\{1 \dots k\}$

$$d_{ij}^{(k)}$$

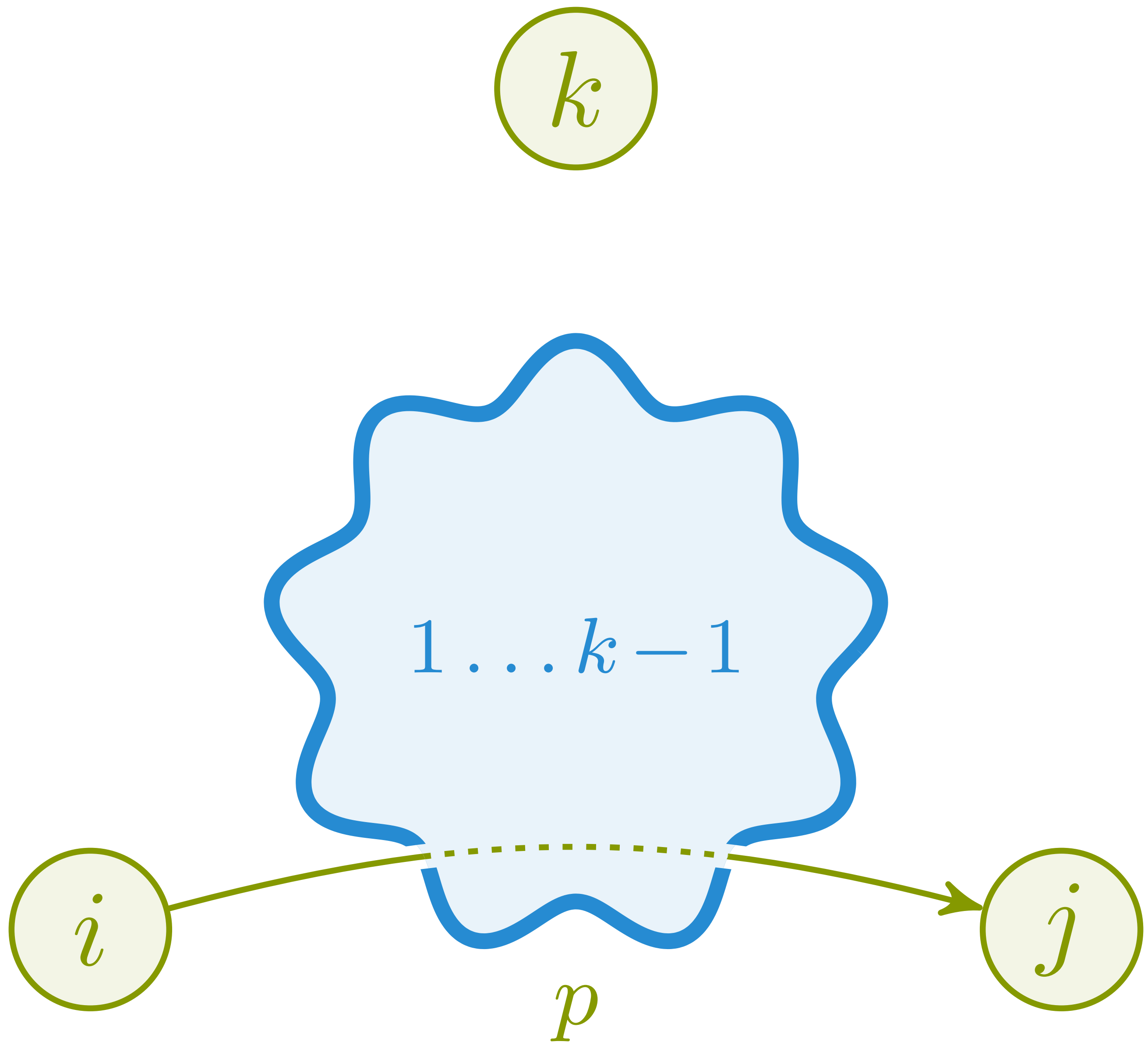
Korteste vei fra i til j via noder fra $\{1 \ldots k\}$

$$\pi_{ij}^{(k)}$$

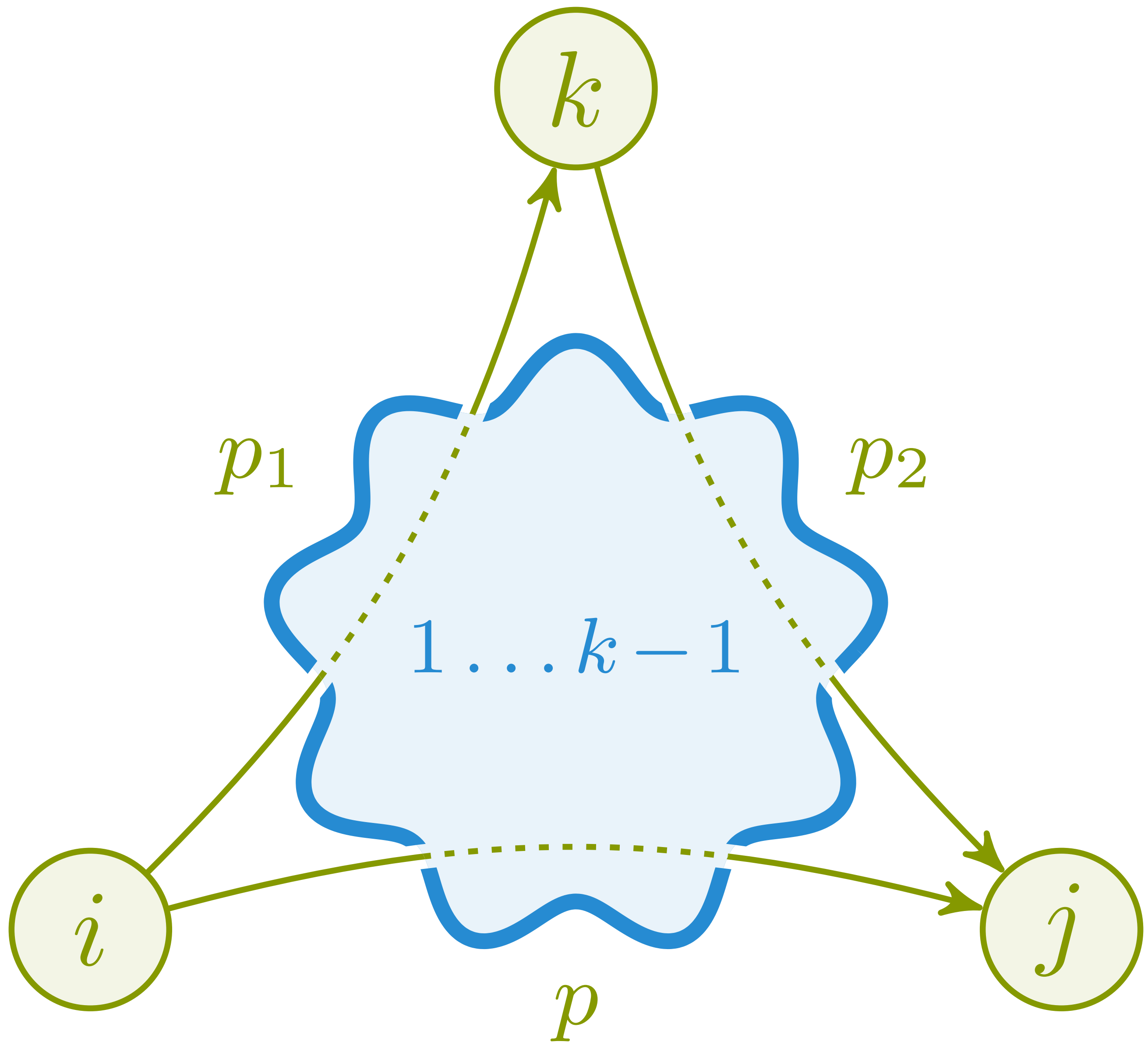
Forgjengeren til j om vi starter i i og går via noder fra $\{1 \ldots k\}$



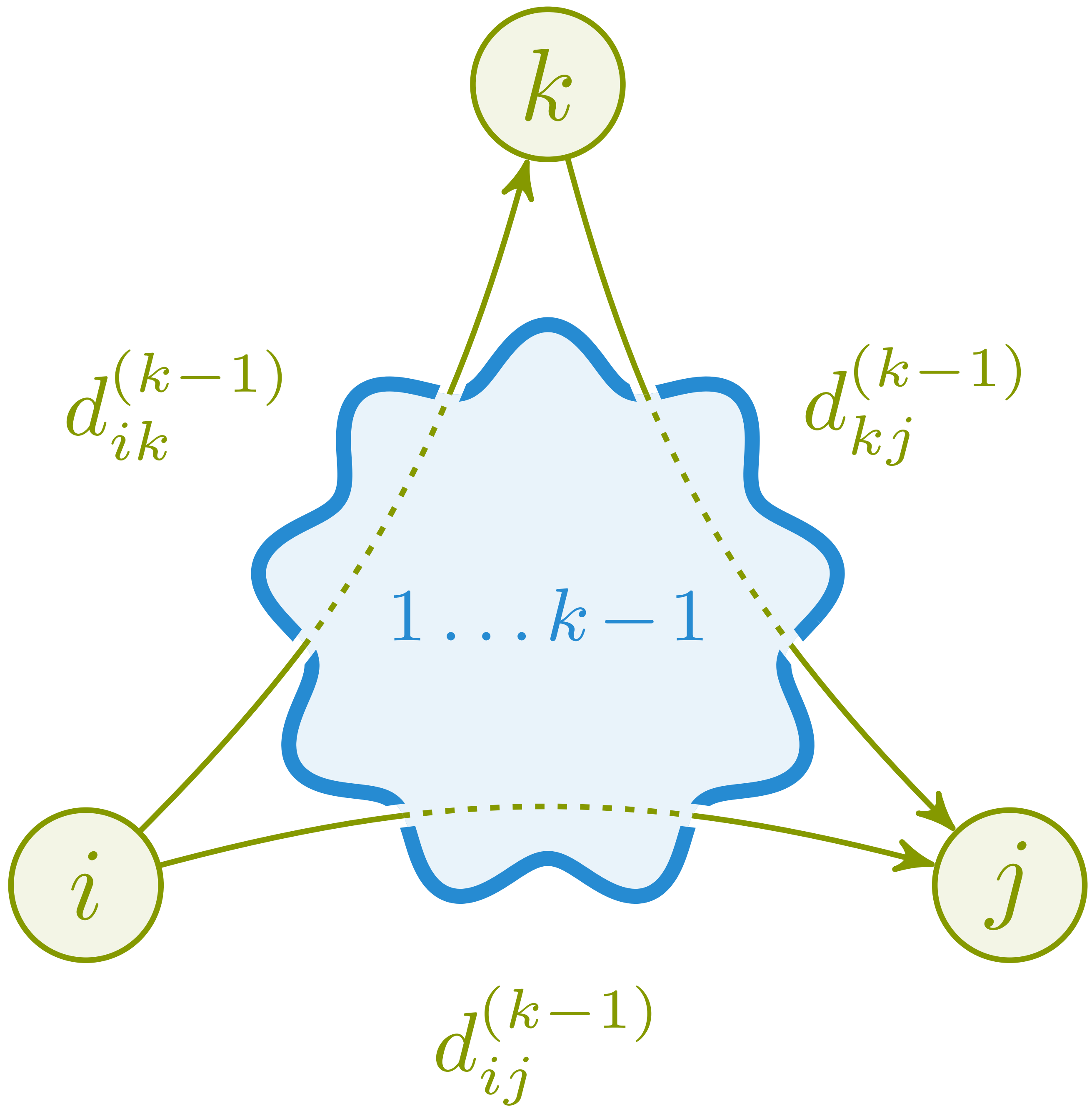
$d_{ij}^{(k)}$ = korteste vei fra i til j via noder fra $\{1 \ldots k\}$



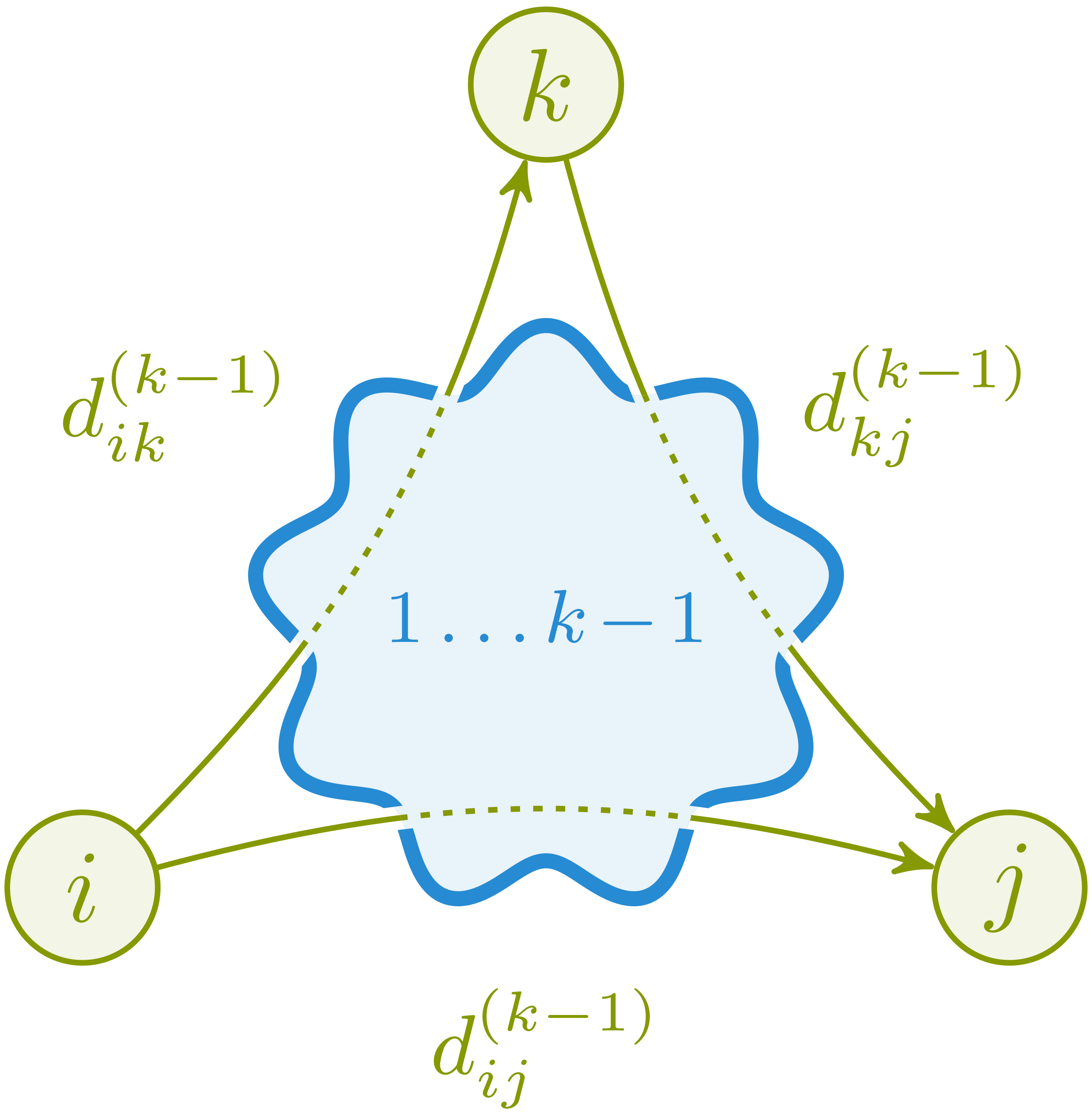
Som for ryggsekkproblemet: Skal k være med eller ikke?



Stiene p, p_1 og p_2 går kun via noder fra $\{1 \ldots k-1\}$



$d_{ij}^{(k)}$ kan enten være $d_{ij}^{(k-1)}$ eller $d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$



$$d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)},\, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$$

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0\,, \\ \min\left(d_{ij}^{(k-1)},\, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1\,. \end{cases}$$

$$\pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{if } i = j \text{ or } w_{ij} = \infty\,, \\ i & \text{if } i \neq j \text{ and } w_{ij} < \infty\,. \end{cases}$$

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\,, \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\,. \end{cases}$$

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0\,, \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1\,. \end{cases}$$

Som før: Vi kan ha gått innom k i én av delstiene, siden vi blander iterasjoner – men vi antar at det ikke er noen negative sykler, og en positiv sykel vil aldri lønne seg (og vil dermed ikke bli med).

Vi kan bruke én tabell igjen (se oppgave 25.2-4)

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \le d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\,, \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\,. \end{cases}$$

Fordi denne bare baserer seg på valget vi gjør for d.

Også her holder det med én tabell

FLOYD-WARSHALL(W)

For hver node i og j: Hva er korteste vei $i \leadsto j$?

```
FLOYD-WARSHALL(W)
  n = W.rows
```

Floyd-Warshall(W)

$n = \mathrm{W}.rows$

$\mathrm{D}^{(0)} = \mathrm{W}$

Korteste vei $i \rightsquigarrow j$ som ikke går via andre $= w(i, j)$

Floyd-Warshall(W)

$n = \mathrm{W}.rows$

$\mathrm{D}^{(0)} = \mathrm{W}$

for $k = 1$ **to** n

Vi får nå lov til å gå innom node k

Floyd-Warshall(W)

```
n = W.rows
D^(0) = W
for k = 1 to n
        let D^(k) = (d_ij^(k)) be a new n × n matrix
```

Hva er korteste vei $i \rightsquigarrow j$ som kun får gå innom $\{1, \ldots, k\}$?

Floyd-Warshall(W)

$n = \mathrm{W}.rows$

$\mathrm{D}^{(0)} = \mathrm{W}$

for $k = 1$ **to** n

  let $\mathrm{D}^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix

  for $i = 1$ **to** n

For hver mulig startnode . . .

FLOYD-WARSHALL(W)
$n = \mathrm{W}.rows$
$\mathrm{D}^{(0)} = \mathrm{W}$
for $k = 1$ **to** n
    let $\mathrm{D}^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix
    for $i = 1$ **to** n
        for $j = 1$ **to** n

For hver mulig sluttnode . . .

Floyd-Warshall(W)

$n = \mathrm{W}.\mathit{rows}$

$\mathrm{D}^{(0)} = \mathrm{W}$

for $k = 1$ **to** n

  let $\mathrm{D}^{(k)} = \left(d_{ij}^{(k)}\right)$ be a new $n \times n$ matrix

  for $i = 1$ **to** n

    for $j = 1$ **to** n

      $d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$

Er det raskere å gå innom k (og ellers fortsatt kun $\{1, \ldots, k-1\}$)?

FLOYD-WARSHALL(W)

$n = \text{W}.rows$
$\text{D}^{(0)} = \text{W}$
for $k = 1$ **to** n
    let $\text{D}^{(k)} = (d_{ij}^{(k)})$ be a new $n \times n$ matrix
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            $d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$
return $\text{D}^{(n)}$

Korteste vei $i \rightsquigarrow j$ som får gå innom $\{1, \ldots, n\}$, dvs. alle

mlh@ntnu.no

Floyd-Warshall′(W)

W vekter

Forenklet: Bruker bare D heller enn $D^{(k)}$ (se 25.2-4, uten Π)

Floyd-Warshall′(W)
1 $n = \mathrm{W}.rows$

W	vekter
n	ant. noder

Floyd-Warshall′(W)
1 $n = \mathrm{W}.rows$
2 initialize D and Π

W	vekter
n	ant. noder
D	avstander
Π	forgjengere

Ikke via noen: $d_{ij} = w(i,j)$; $\pi_{ij} = i$ når $i \neq j$ og $w(i,j) < \infty$

Floyd-Warshall′(W)
1 $n = \mathrm{W}.rows$
2 initialize D and Π
3 **for** $k = 1$ **to** n

W	vekter
n	ant. noder
D	avstander
Π	forgjengere
k	mellomlanding

Vi får nå lov til å gå innom node k

```
FLOYD-WARSHALL′(W)
n = W.rows
initialize D and Π
for k = 1 to n
        for i = 1 to n
```

W	vekter
n	ant. noder
D	avstander
Π	forgjengere
k	mellomlanding
i	fra-node

For hver mulig startnode . . .

FLOYD-WARSHALL′(W)

$n = \mathrm{W}.rows$
initialize D and Π
for $k = 1$ **to** n
    for $i = 1$ **to** n
        for $j = 1$ **to** n

W	vekter
n	ant. noder
D	avstander
Π	forgjengere
k	mellomlanding
i	fra-node
j	til-node

For hver mulig sluttnode . . .

FLOYD-WARSHALL′(W)

$n = \mathrm{W}.rows$
initialize D and Π
for $k = 1$ **to** n
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            if $d_{ij} > d_{ik} + d_{kj}$

W	vekter
n	ant. noder
D	avstander
Π	forgjengere
k	mellomlanding
i	fra-node
j	til-node
d_{ij}	celle i D

Er det raskere å gå innom k (og ellers fortsatt kun $\{1, \ldots, k-1\}$)?

FLOYD-WARSHALL$'$(W)

$n = W.rows$
initialize D and Π
for $k = 1$ **to** n
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            if $d_{ij} > d_{ik} + d_{kj}$
                $d_{ij} = d_{ik} + d_{kj}$

W	vekter
n	ant. noder
D	avstander
Π	forgjengere
k	mellomlanding
i	fra-node
j	til-node
d_{ij}	celle i D

Oppdater avstanden

FLOYD-WARSHALL$'$(W)

$n = \mathrm{W}.rows$
initialize D and Π
for $k = 1$ **to** n
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            if $d_{ij} > d_{ik} + d_{kj}$
                $d_{ij} = d_{ik} + d_{kj}$
                $\pi_{ij} = \pi_{kj}$

W	vekter
n	ant. noder
D	avstander
Π	forgjengere
k	mellomlanding
i	fra-node
j	til-node
d_{ij}	celle i D
π_{ij}	celle i Π

Husk valget: Forgjengeren til i om vi kommer fra j

```
FLOYD-WARSHALL′(W)
n = W.rows
initialize D and Π
for k = 1 to n
        for i = 1 to n
                for j = 1 to n
                        if d_ij > d_ik + d_kj
                                d_ij = d_ik + d_kj
                                π_ij = π_kj
return D, Π
```

W	vekter
n	ant. noder
D	avstander
Π	forgjengere
k	mellomlanding
i	fra-node
j	til-node
d_{ij}	celle i D
π_{ij}	celle i Π

mlh@ntnu.no

```
FLOYD-WARSHALL'(W)
n = W.rows
initialize D and Π
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            if d_ij > d_ik + d_kj
                d_ij = d_ik + d_kj
                π_ij = π_kj
return D, Π
```

$k, i, j \;=\; 1, 1, 1$

D

	1	2	3	4	5
1	0	3	8	∞	–4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	–5	0	∞
5	∞	∞	∞	6	0

Π

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

FLOYD-WARSHALL′(W)

$n = W.rows$
initialize D and Π
for $k = 1$ **to** n
    for $i = 1$ **to** n
        for $j = 1$ **to** n
            if $d_{ij} > d_{ik} + d_{kj}$
                $d_{ij} = d_{ik} + d_{kj}$
                $\pi_{ij} = \pi_{kj}$
return D, Π

$k, i, j \;=\;$ -, -, -

D

	1	2	3	4	5
1	0	1	–3	2	–4
2	3	0	–4	1	–1
3	7	4	0	5	3
4	2	–1	–5	0	–2
5	8	5	1	6	0

Π

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

PRINT-ALL-PAIRS-SHORTEST-PATH(Π, i, j)

Hvilke noder er i stien fra i til j?

$\textsc{Print-All-Pairs-Shortest-Path}(\Pi, i, j)$

if $i == j$

Samme node? Kommer ingenstedsfra . . .

$\textsc{Print-All-Pairs-Shortest-Path}(\Pi, i, j)$

if $i == j$
    print i

…så bare skriv ut noden

Print-All-Pairs-Shortest-Path(Π, i, j)

if $i == j$

  print i

elseif $\pi_{ij} ==$ NIL

Hvis vi ellers ikke kom fra noe sted . . .

Print-All-Pairs-Shortest-Path(Π, i, j)

if $i == j$
    print i
elseif $\pi_{ij} ==$ NIL
    print "no path from" i "to" j "exists"

…så finnes ingen sti!

Print-All-Pairs-Shortest-Path(Π, i, j)

if $i == j$
    print i
elseif $\pi_{ij} ==$ NIL
    print "no path from" i "to" j "exists"
else Print-All-Pairs-Shortest-Path(Π, i, π_{ij})

Ellers kom vi fra π_{ij}. Skriv veien dit først ...

```
Print-All-Pairs-Shortest-Path(Π, i, j)
if i == j
    print i
elseif π_ij == NIL
    print "no path from" i "to" j "exists"
else Print-All-Pairs-Shortest-Path(Π, i, π_ij)
    print j
```

…og deretter sluttnoden

mlh@ntnu.no

PRINT-PATH(Π, i, j)

if $i == j$
    print i
elseif $\pi_{ij} ==$ NIL
    print "no such path"
else PRINT-PATH(Π, i, π_{ij})
    print j

$i, j \;=\; 1, 2$

Π

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

```
PRINT-PATH(Π, i, j)
if i == j
      print i
elseif π_ij == NIL
      print "no such path"
else PRINT-PATH(Π, i, π_ij)
      print j
```

Π

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

$i, j \;=\; 1, 2$

1 5 4 3 2

Korteste vei › Floyd-Warshall › Kjøretid

init › $\Theta(n^2)$

for $k = 1$ **to** n

  for $i = 1$ **to** n

    for $j = 1$ **to** n

      min › $O(1)$

return › $O(1)$

Totalt: $\Theta(n^3)$

1. **Johnsons algoritme**
2. **Transitiv lukning**
3. **Floyd-Warshall**