

Forelesning 11

Vi kan finne de korteste veiene fra hver node etter tur, men mange av delproblemene vil overlappe – om vi har mange nok kanter lønner det seg å bruke dynamisk programmering med dekomponeringen «Skal vi innom k eller ikke?»

Pensum

- Kap. 25. All-pairs shortest paths: Innledning, 25.2 og 25.3

Læringsmål

- [K₁] Forstå *forgjengerstrukturen* for *alle-til-alle*-varianten av korteste vei-problemet
Operasjoner: PRINT-ALL-PAIRS-SHORTEST-PATH
- [K₂] Forstå FLOYD-WARSHALL
- [K₃] Forstå TRANSITIVE-CLOSURE
- [K₄] Forstå JOHNSON

Forelesningen filmes



0:3

Kjøretid, Dijkstra

Operasjon	Antall	Kjøretid
-----------	--------	----------

Operasjon	Antall	Kjøretid
Initialisering		

Operasjon	Antall	Kjøretid
Initialisering	1	

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$

Operasjon	Antall	Kjøretid
Initialisering BUILD-HEAP	1	$\Theta(V)$

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$
EXTRACT-MIN		

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$
EXTRACT-MIN	V	

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$
EXTRACT-MIN	V	$O(\lg V)$

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$
EXTRACT-MIN	V	$O(\lg V)$
DECREASE-KEY		

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$
EXTRACT-MIN	V	$O(\lg V)$
DECREASE-KEY*		

*Nødvendig i RELAX

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$
EXTRACT-MIN	V	$O(\lg V)$
DECREASE-KEY*	E	

*Nødvendig i RELAX

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$
EXTRACT-MIN	V	$O(\lg V)$
DECREASE-KEY*	E	$O(\lg V)$

*Nødvendig i RELAX

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$
EXTRACT-MIN	V	$O(\lg V)$
DECREASE-KEY*	E	$O(\lg V)$

Totalt: $O(V \lg V + E \lg V)$

*Nødvendig i RELAX

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$
EXTRACT-MIN	V	$O(\lg V)$
DECREASE-KEY*	E	$O(\lg V)$

Totalt: $O(V \lg V + E \lg V)$

*Nødvendig i RELAX

Med binærheap; bedre enn lineært søk for $E = o(V^2 / \lg V)$

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$
EXTRACT-MIN	V	$O(\lg V)$
DECREASE-KEY*	E	$O(\lg V)$

Totalt: $O(V \lg V + E \lg V)$

*Nødvendig i RELAX

Boka bruker $V \times \text{INSERT}$; fortsatt $O(E \lg V)$

Operasjon	Antall	Kjøretid
Initialisering	1	$\Theta(V)$
BUILD-HEAP	1	$\Theta(V)$
EXTRACT-MIN	V	$O(\lg V)$
DECREASE-KEY*	E	$O(\lg V)$

Totalt: $O(V \lg V + E \lg V)$

*Nødvendig i RELAX

Fibonacci-heaps: DECREASE-KEY er $O(1)$; vi får $O(V \lg V + E)$



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 $W = (w_{ij})$.

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

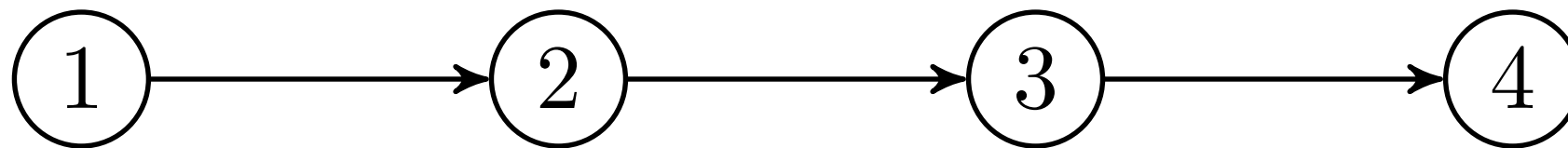
Output: En $n \times n$ -matrise $D = (d_{ij})$ med avstander, dvs., $d_{ij} = \delta(i, j)$.

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

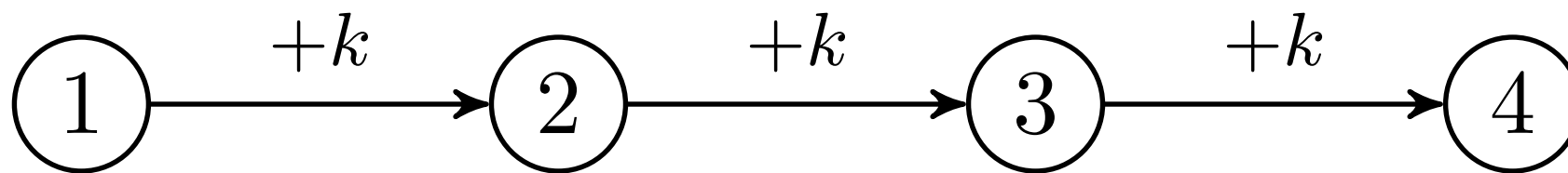
Output: En $n \times n$ -matrise $D = (d_{ij})$ med avstander, dvs., $d_{ij} = \delta(i, j)$.

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

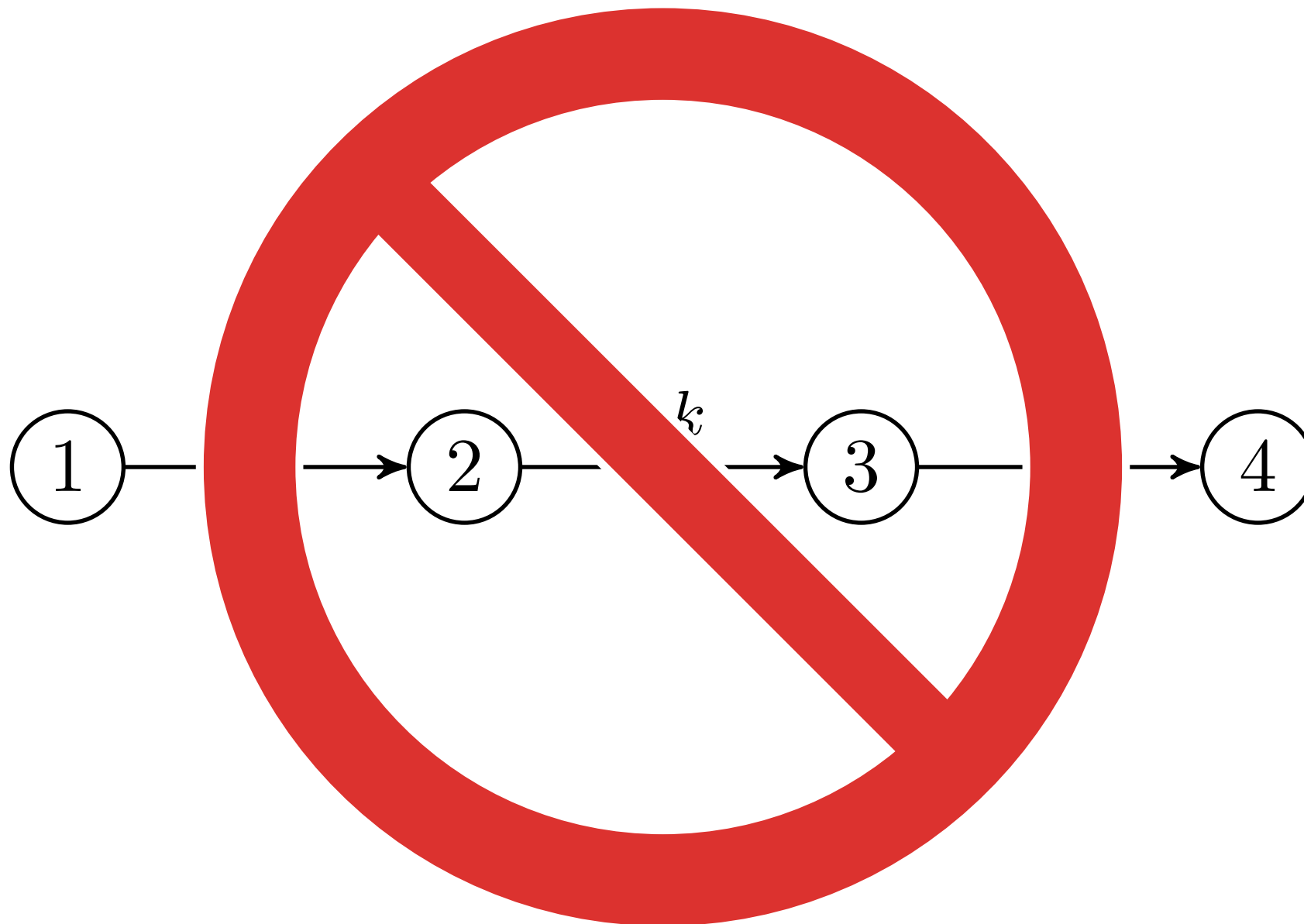
For spinkle grafer: Dijkstra fra hver node!
Men ... hva om vi har negative kanter?



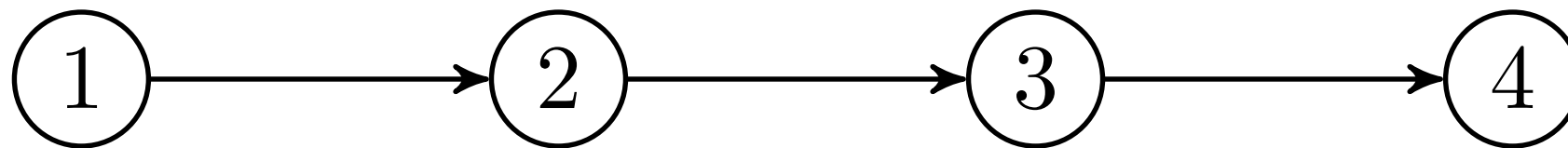
Vi vil øke vektor, men beholde rangering av stier



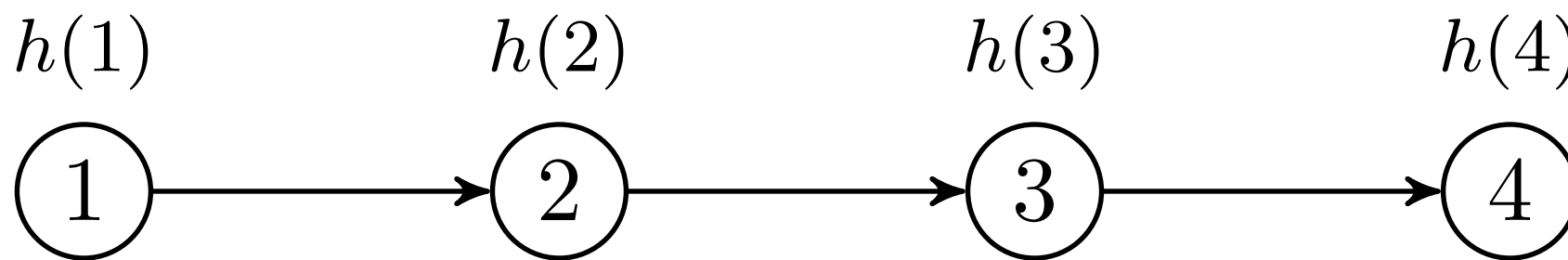
Fast økning: Stier med mange kanter taper på det



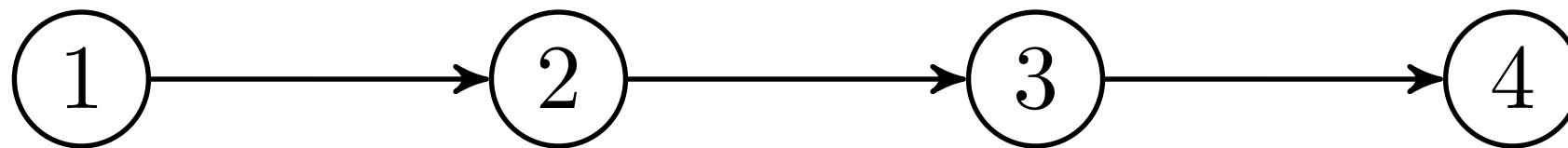
Fast økning: Stier med mange kanter taper på det



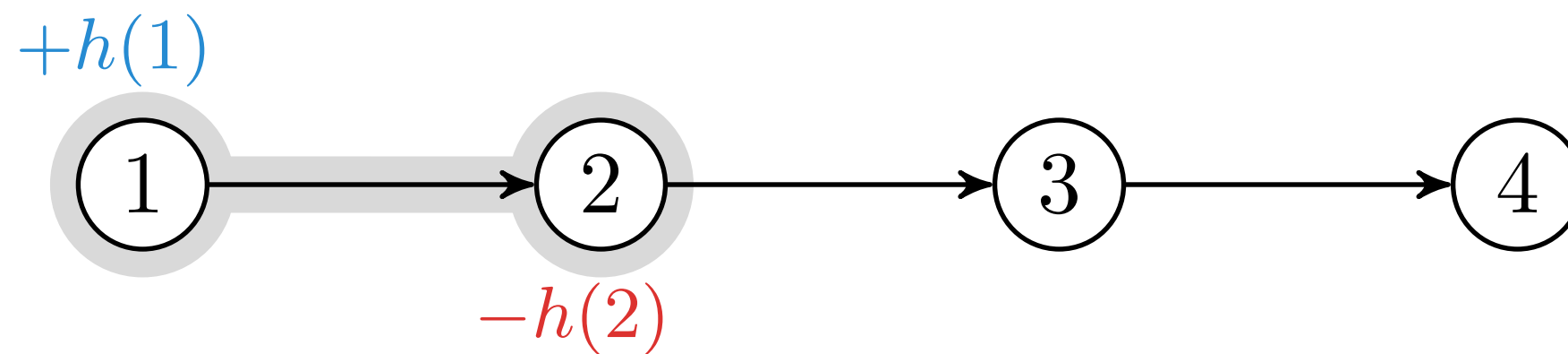
Vi kan tillate oss en teleskopsum...



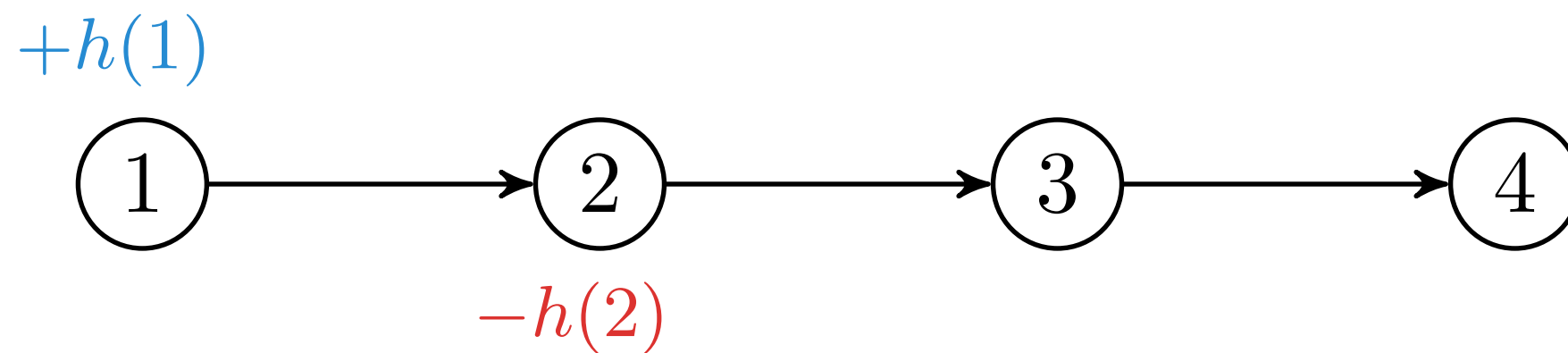
Hver node tilordnes en verdi



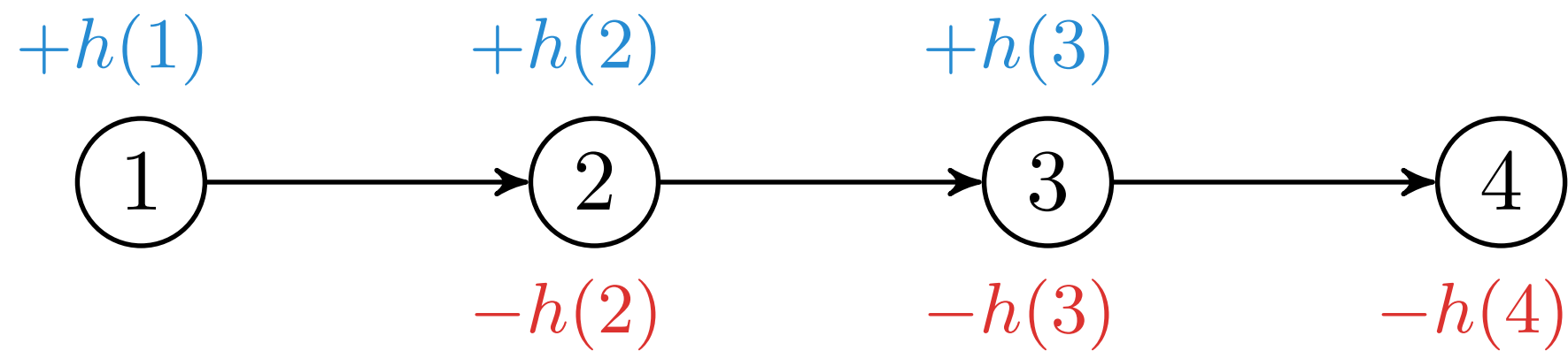
Vekten $w(u, v)$ økes med differansen $h(u) - h(v)$



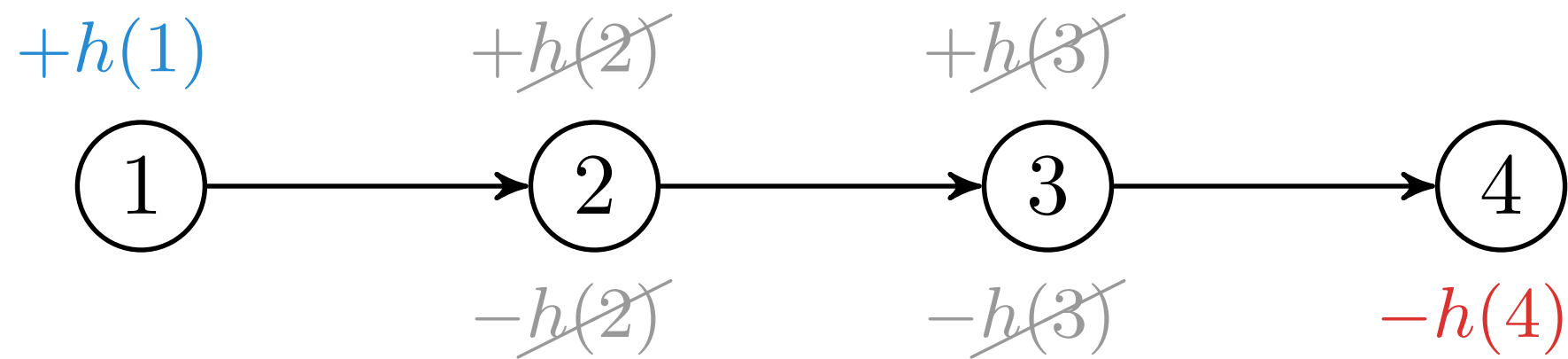
Vekten $w(u, v)$ økes med differansen $h(u) - h(v)$



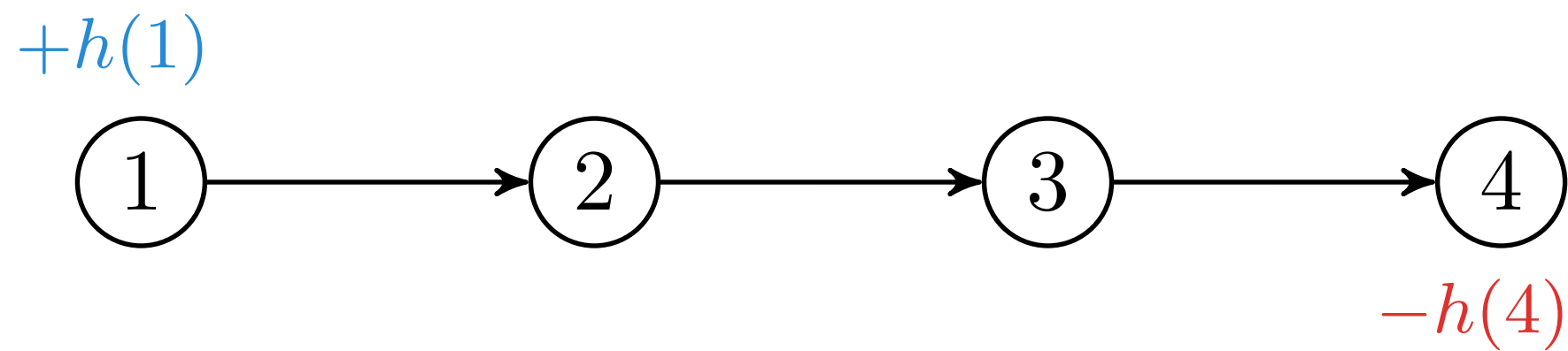
Positive og negative ledd opphever hverandre...



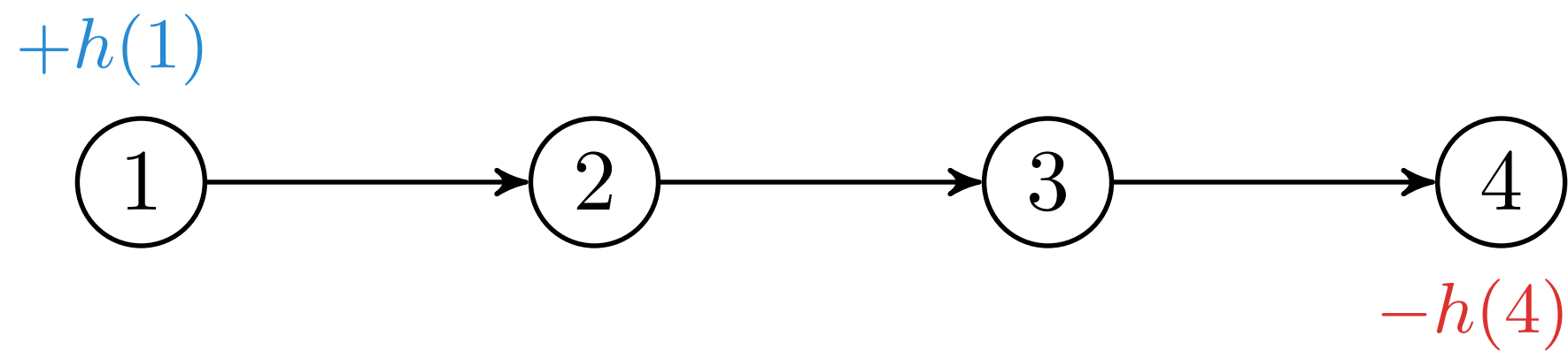
Positive og negative ledd opphever hverandre...



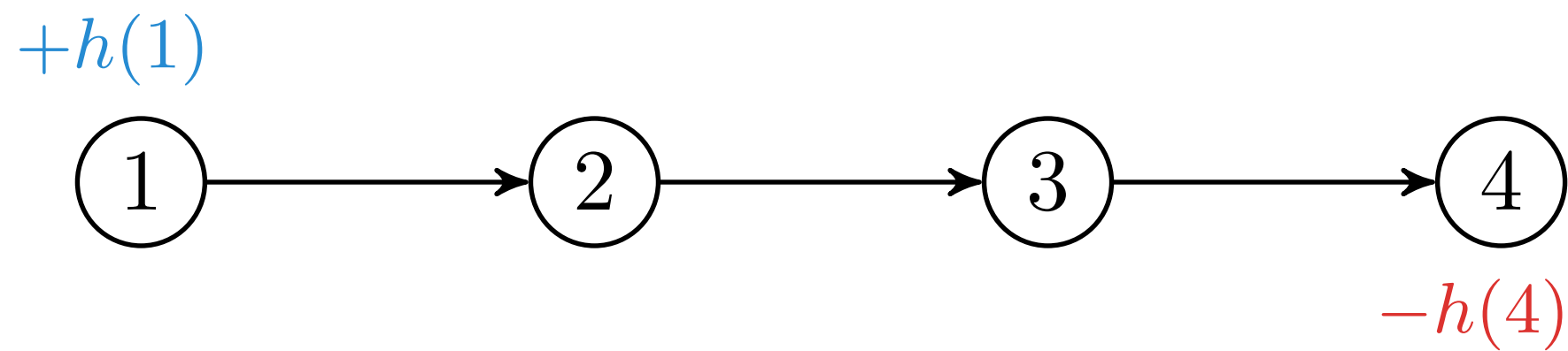
Positive og negative ledd opphever hverandre...



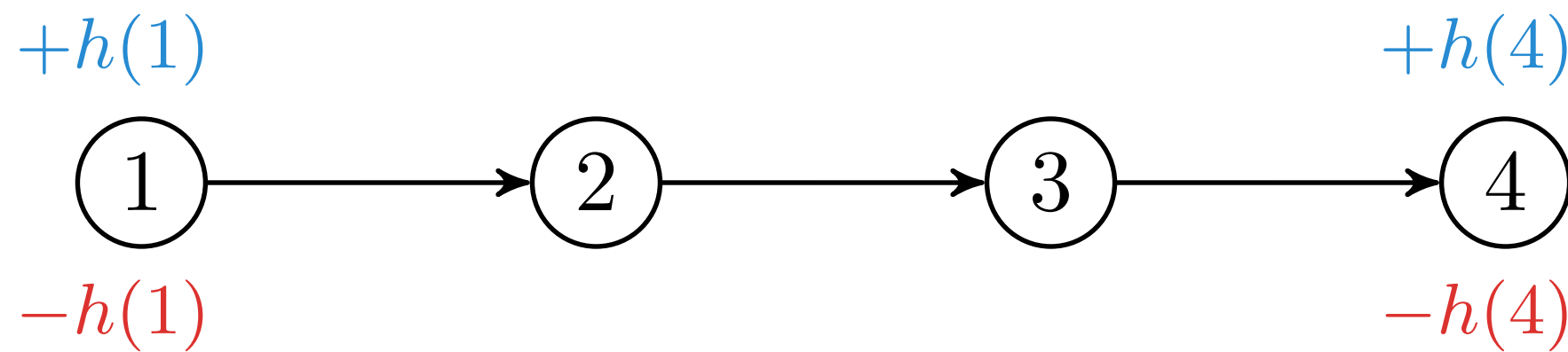
Unntatt første og siste ledd



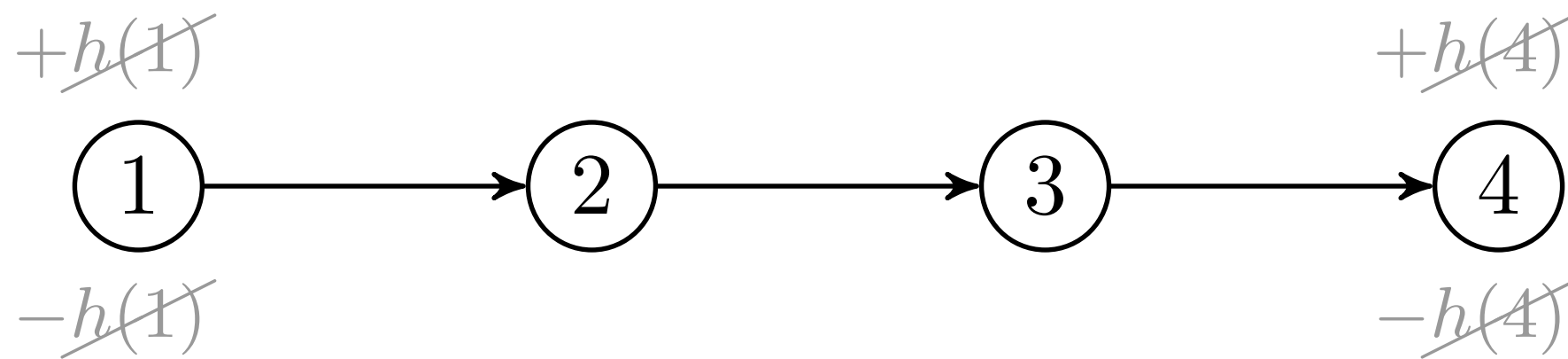
Men disse deles av alle stier mellom disse nodene!



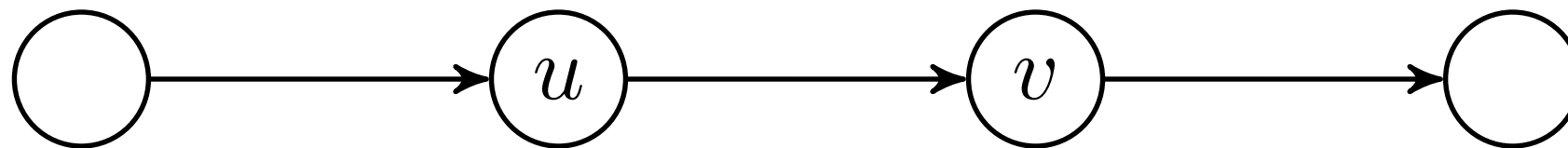
Rekonstruer lengden ved å trekke fra differansen deres



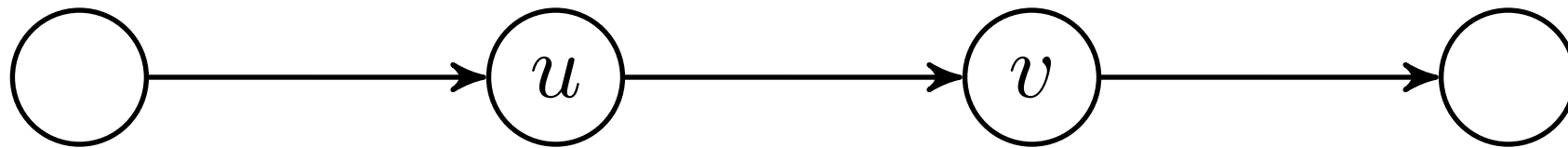
Rekonstruer lengden ved å trekke fra differansen deres



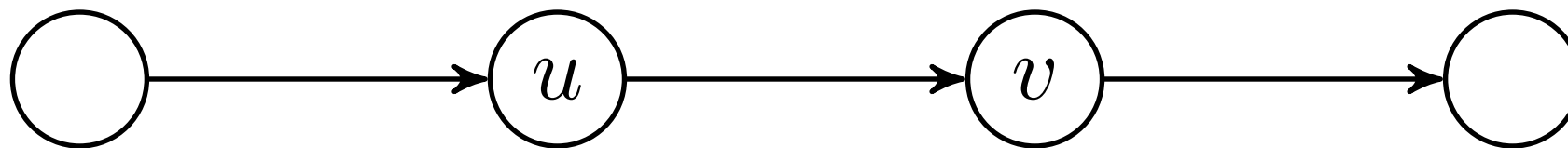
Rekonstruer lengden ved å trekke fra differansen deres



Hvordan sikrer vi ikke-negative vekter?

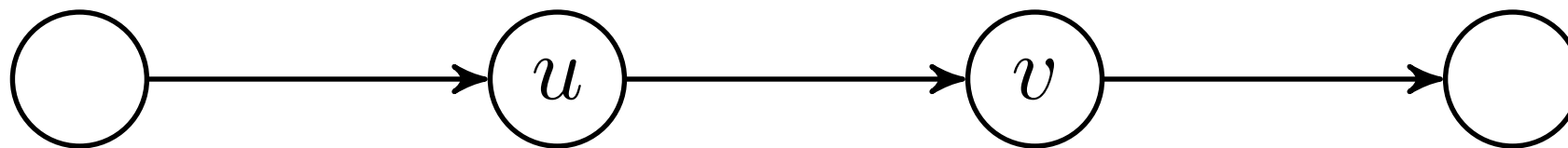


Vi må ha $w(u, v) + h(u) - h(v) \geq 0$, dvs. $w(u, v) + h(u) \geq h(v)$



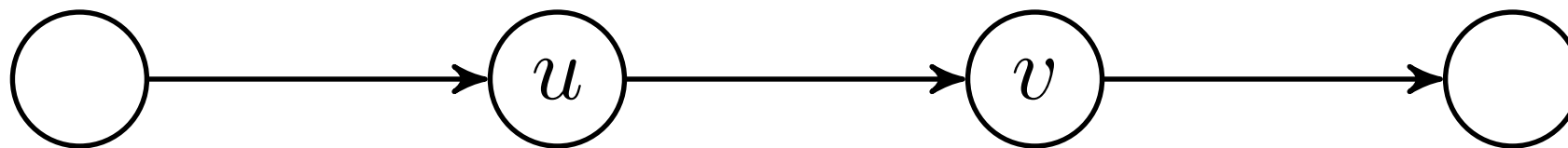
$$w(u, v) + h(u) \geq h(v)$$

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



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

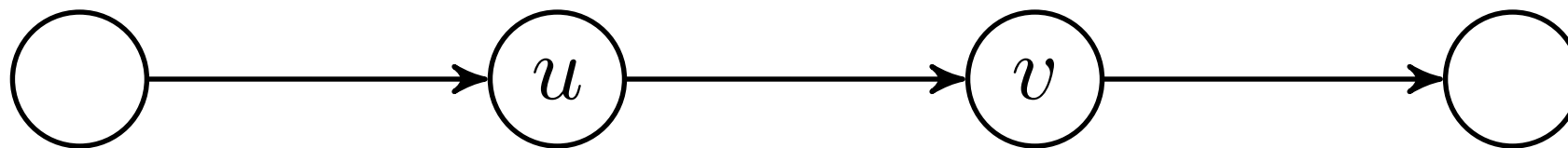
Fra én-til-alle: $\delta(s, v) \leq \delta(s, u) + w(u, v)$. Kan la $h(v) = \delta(s, v)$!



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

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

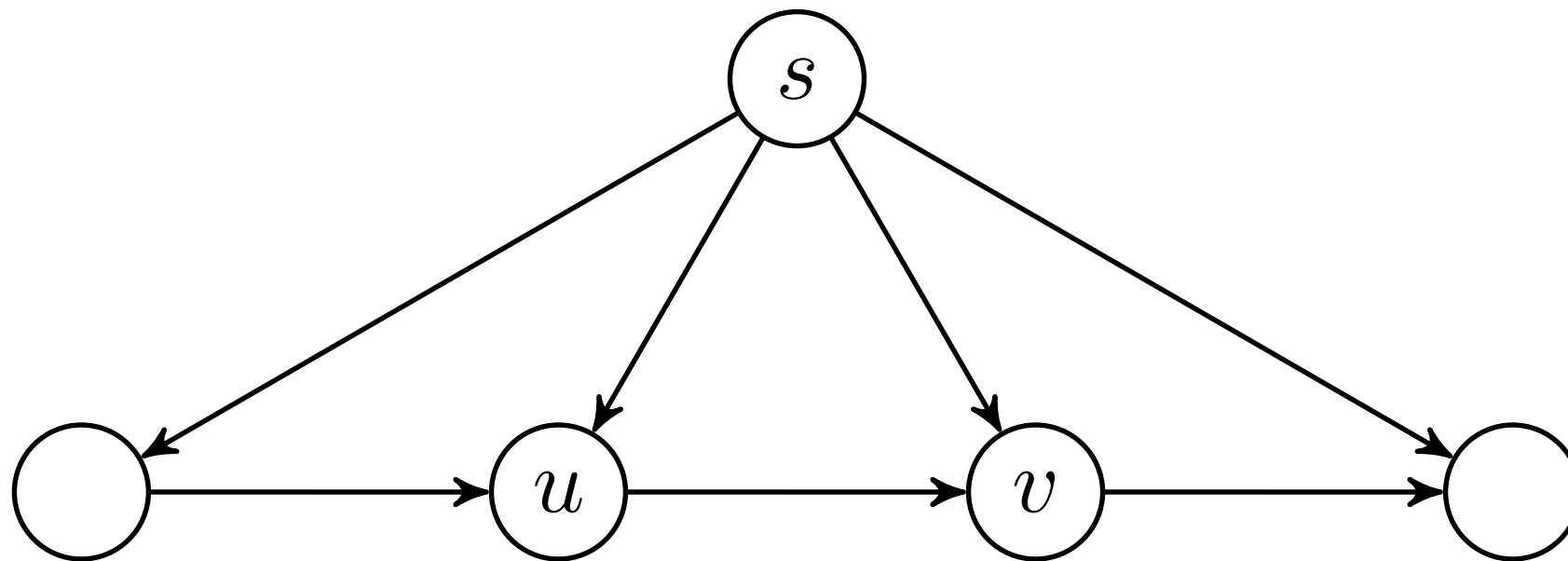
Men hva er s ? Vi må sikre at vi når alle...



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

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

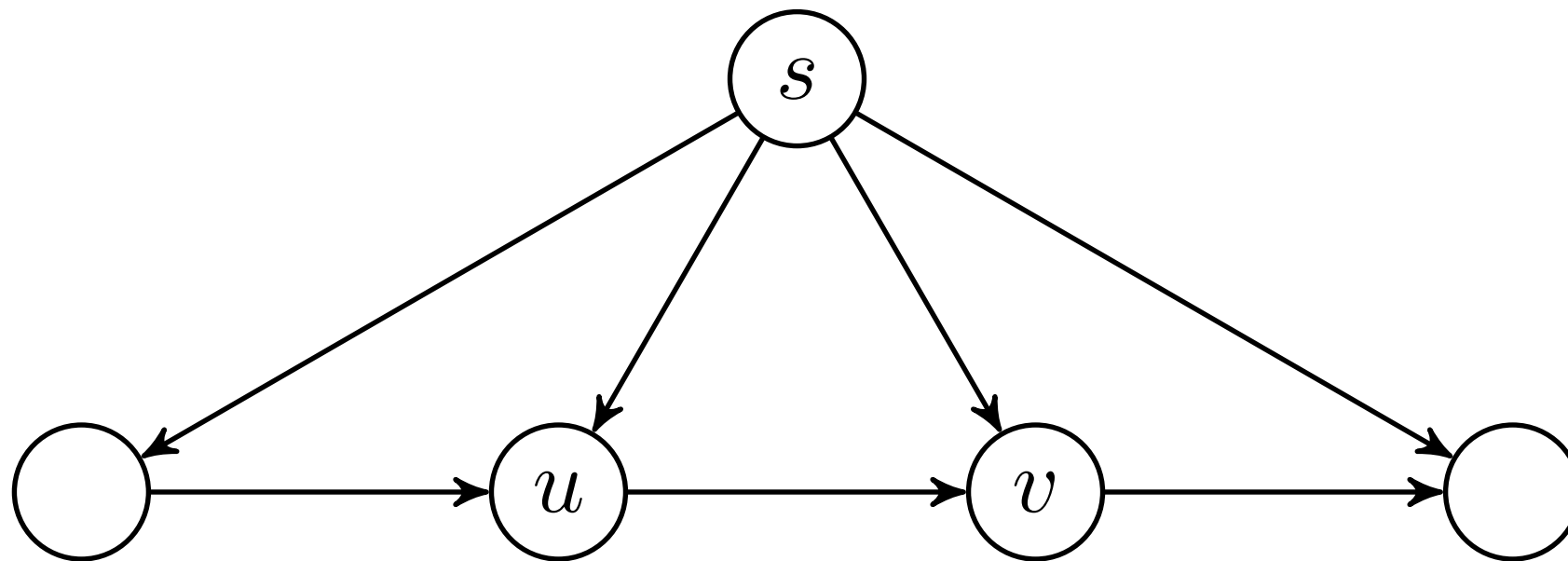
Vi kan legge til en ny node!



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

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

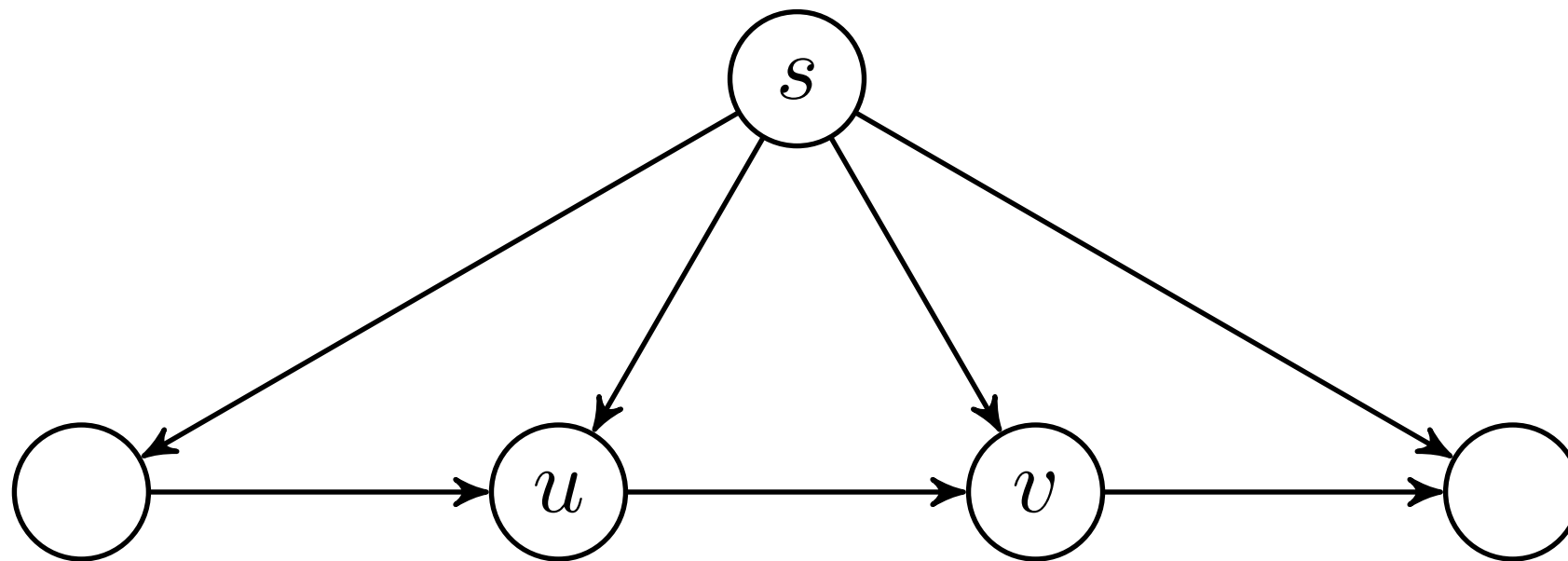
Vi kan legge til en ny node!



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

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

Kantene fra s kan f.eks. få vekt 0



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

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

(Merk: Vi verken innfører eller fjerner negative sykler)

$\text{JOHNSON}(G, w)$

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).

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( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
    
```

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

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

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
    
```

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

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

JOHNSON(G, w)

- 1 construct G' with start node s
- 2 BELLMAN-FORD(G', w, s)
- 3 **for** each vertex $v \in G.V$
- 4 $h(v) = v.d$
- 5 **for** each edge $(u, v) \in G.E$
- 6 $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$

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

$$h(v) \leq h(u) + w(u, v) \implies \hat{w}(u, v) \geq 0$$

JOHNSON(G, w)

```

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

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

Resultat: Alle til alle med DIJKSTRA

JOHNSON(G, w)

```

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

```

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

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

JOHNSON(G, w)

```

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

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

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

JOHNSON(G, w)

```

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

```

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

```

1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
    
```

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

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

JOHNSON(G, w)

```

1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 

```

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

Rett fordi $w(p) \leq w(q) \iff \hat{w}(p) \leq \hat{w}(q)$ for stier $u \xrightarrow{p,q} v$

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 

```

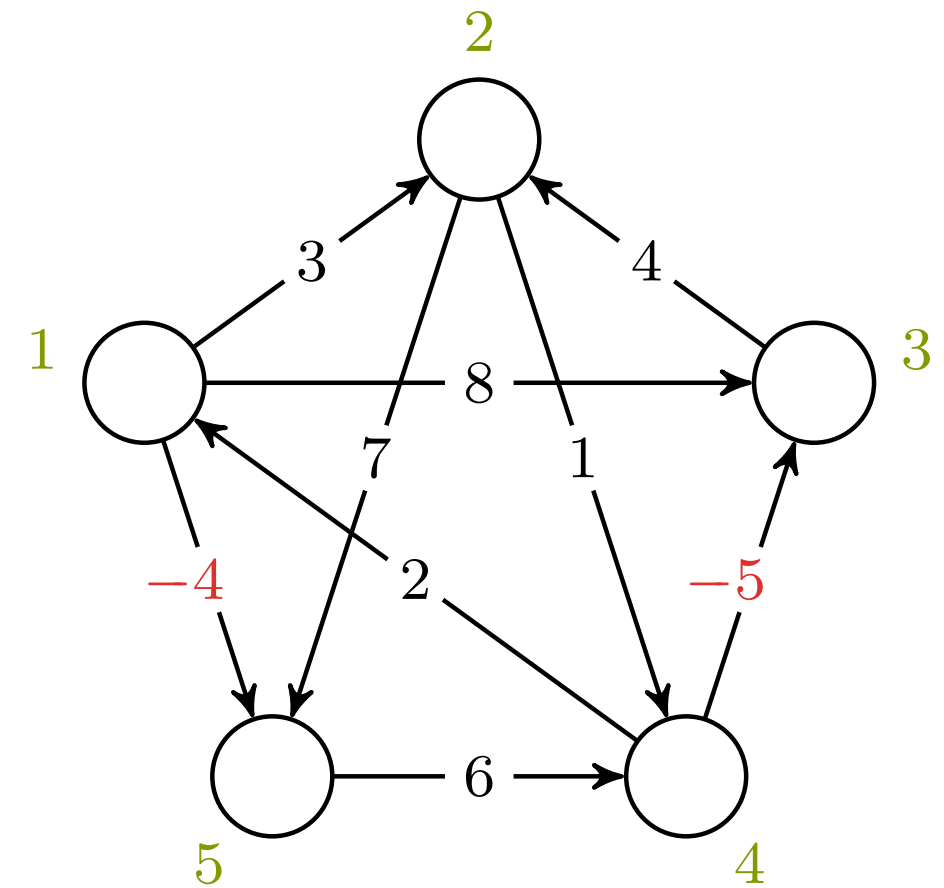
G	graf
w	veking
u	node
v	node
h	$\delta(s, v)$
\hat{w}	ny veking
d_{uv}	$\delta(u, v)$

JOHNSON(G, w)

```

1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 

```



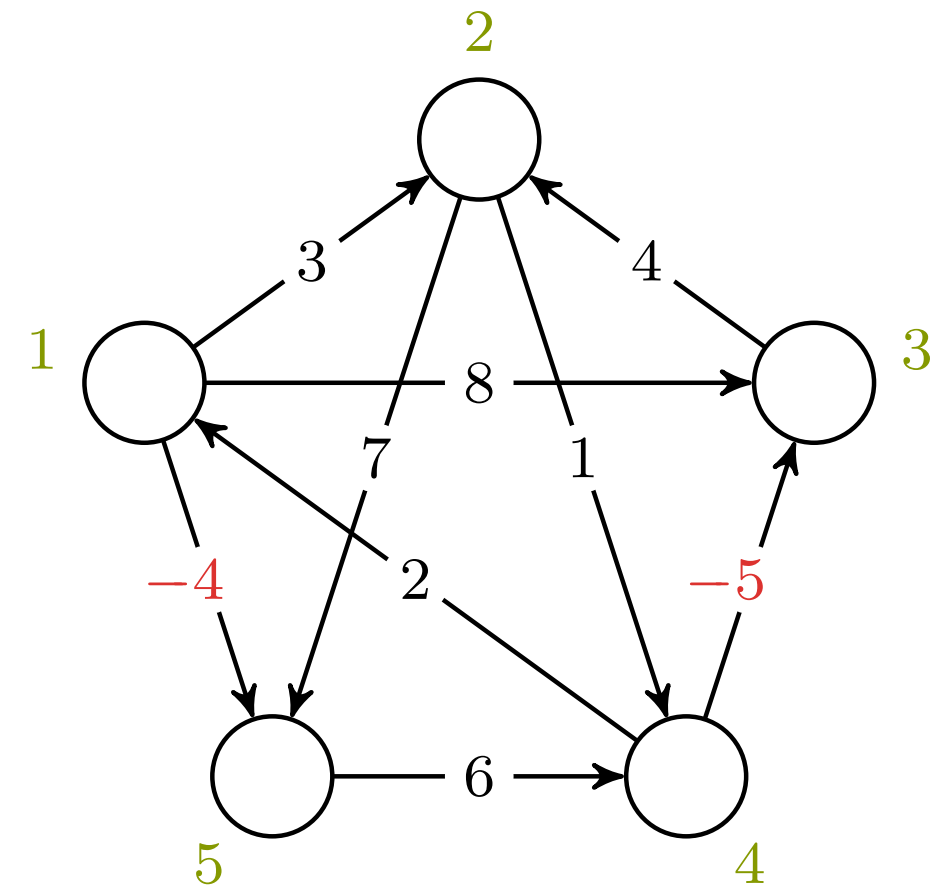
D	1						1
	2						2
	3						3
	4						4
	5						5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```

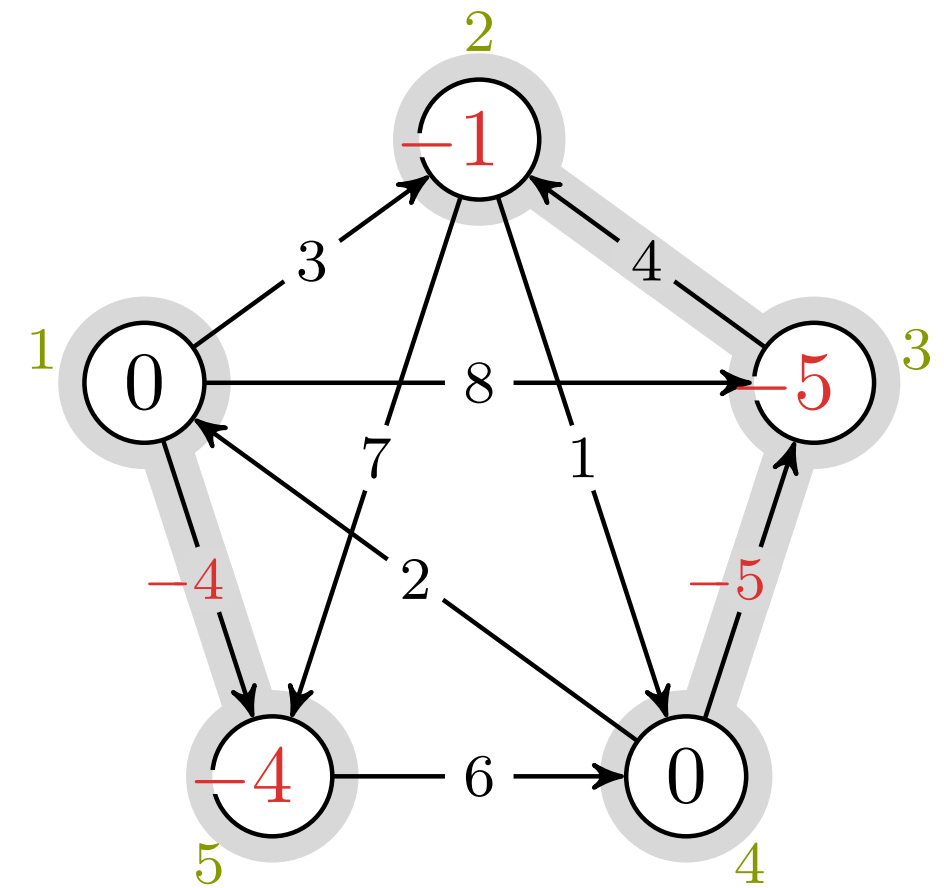
Merk: s er ikke tegnet inn



D	1						1	h
	2						2	
	3						3	
	4						4	
	5						5	
		1	2	3	4	5		

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```



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

D

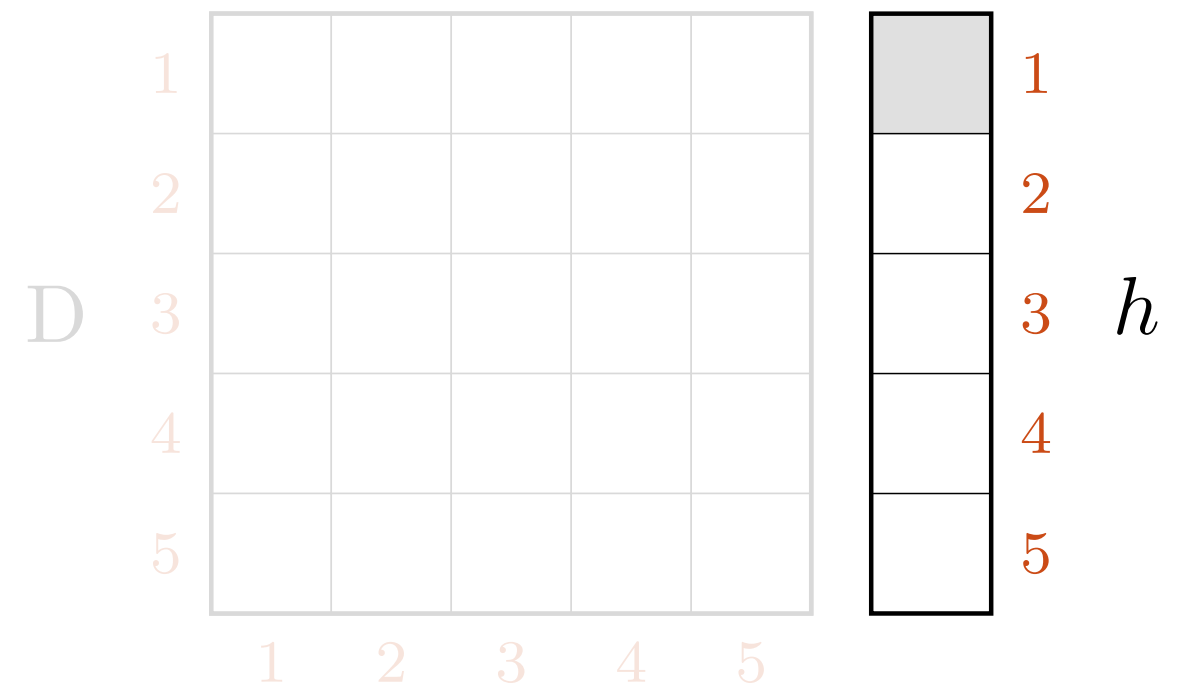
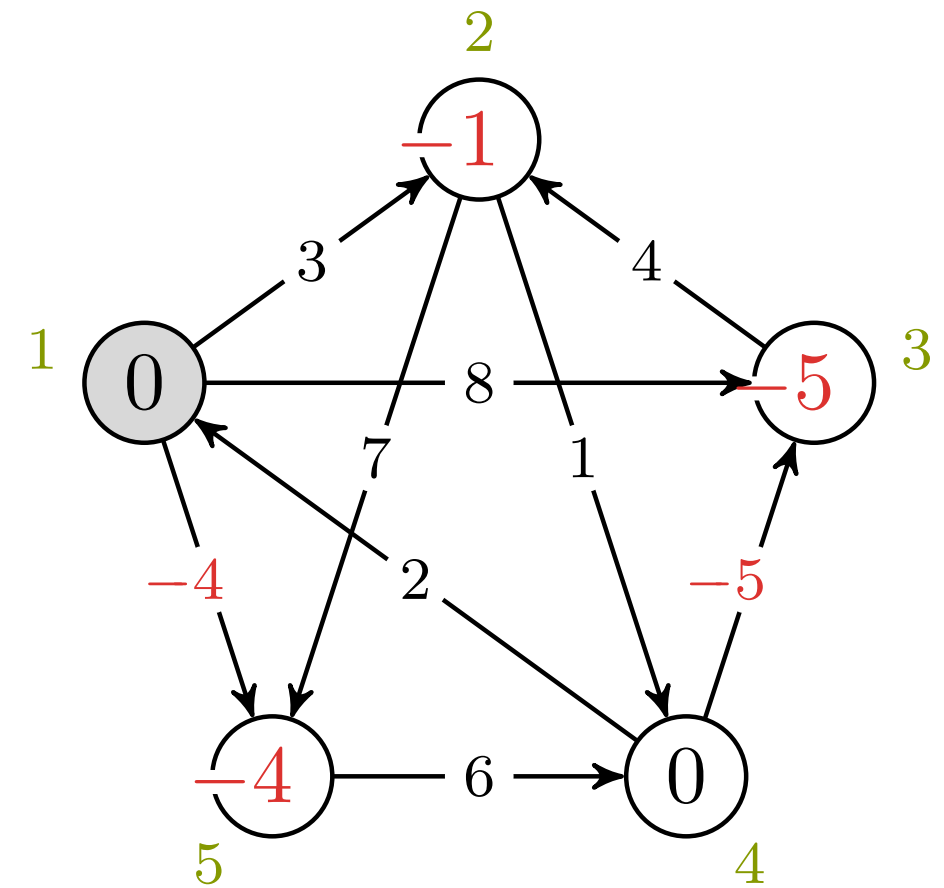
h

korteste vei › johnson

```

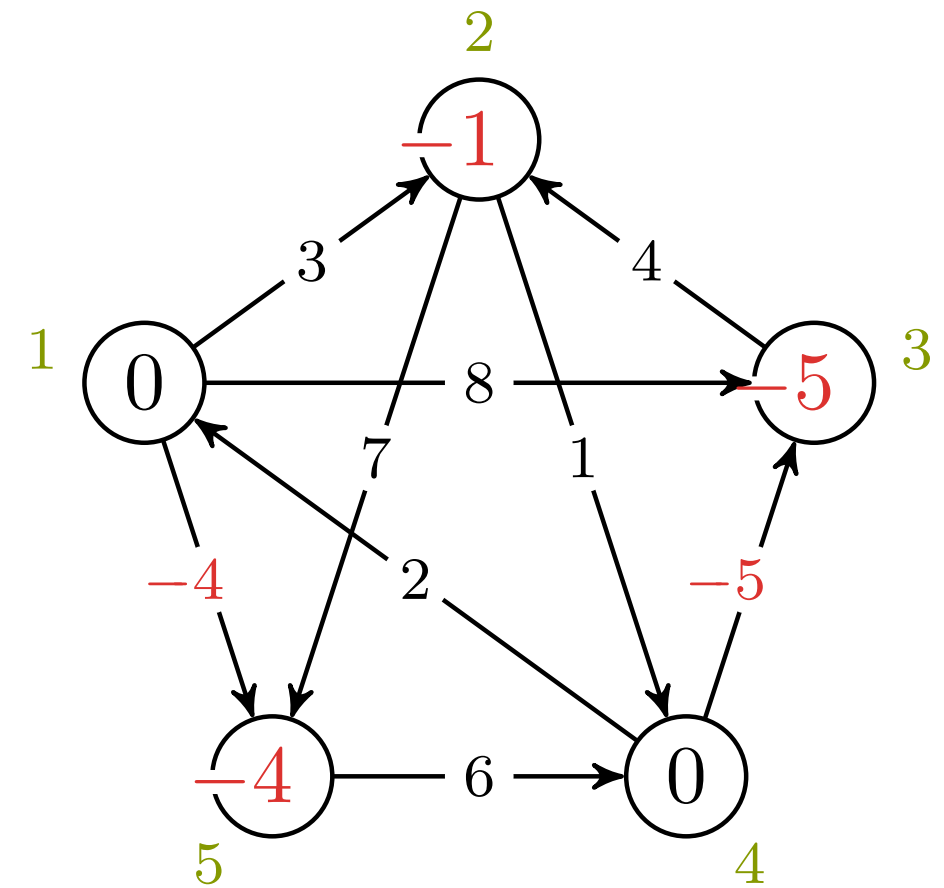
1  JOHNSON( $G, w$ )
2    construct  $G'$  with start node  $s$ 
3    BELLMAN-FORD( $G', w, s$ )
4    for each vertex  $v \in G.V$ 
5       $h(v) = v.d$ 
6    for each edge  $(u, v) \in G.E$ 
7       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
8    let  $D = (d_{uv})$  be a new  $n \times n$  matrix
9    for each vertex  $u \in G.V$ 
10     DIJKSTRA( $G, \hat{w}, u$ )
11     for each vertex  $v \in G.V$ 
12        $d_{uv} = v.d + h(v) - h(u)$ 
13  return  $D$ 

```



```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```



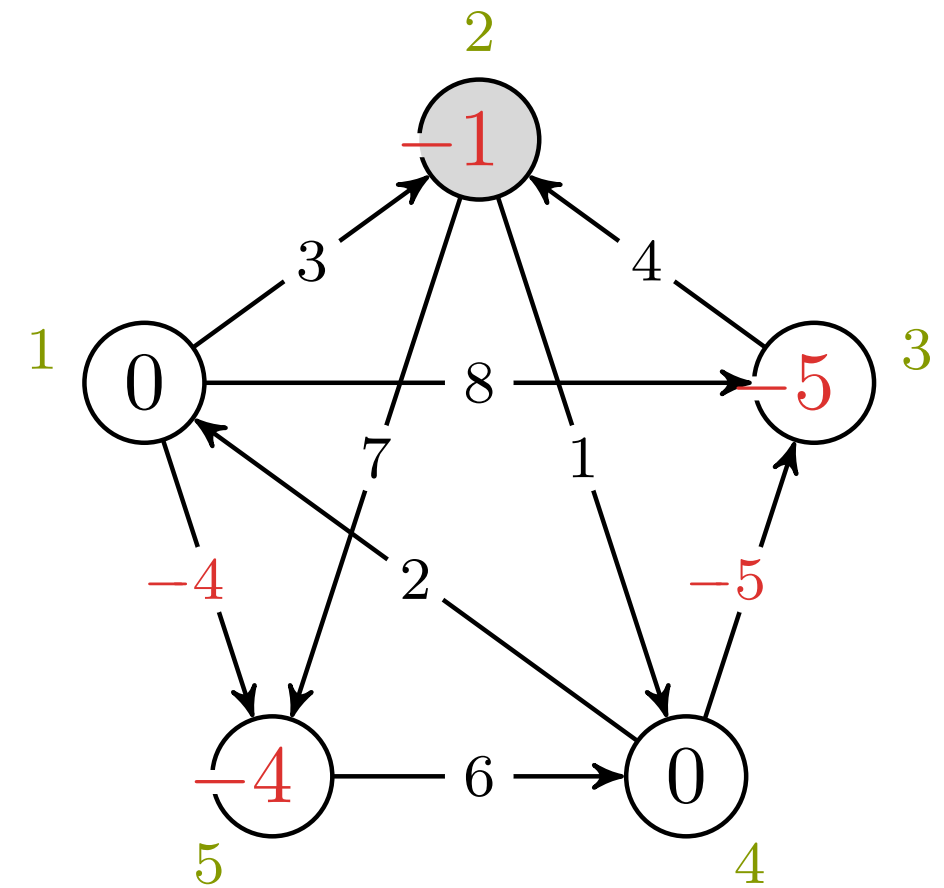
D	1					0	1
	2						2
	3						3
	4						4
	5						5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4   $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6     $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9    DIJKSTRA( $G, \hat{w}, u$ )
10   for each vertex  $v \in G.V$ 
11      $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 

```



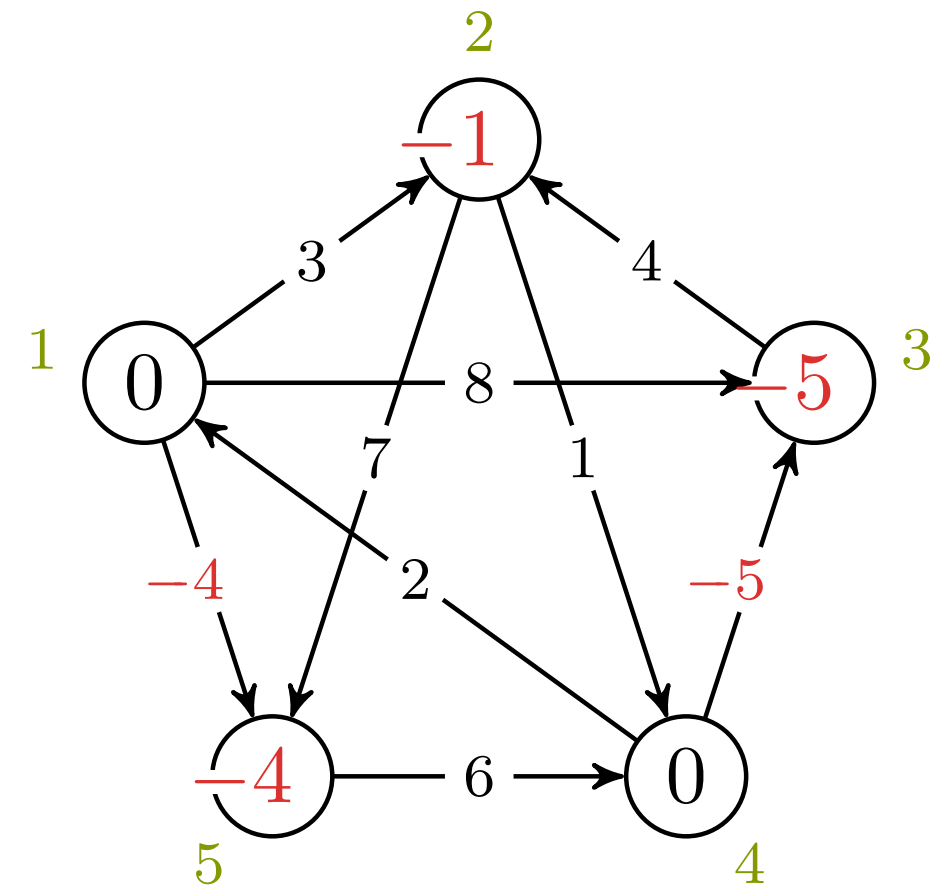
D

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

h


```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```



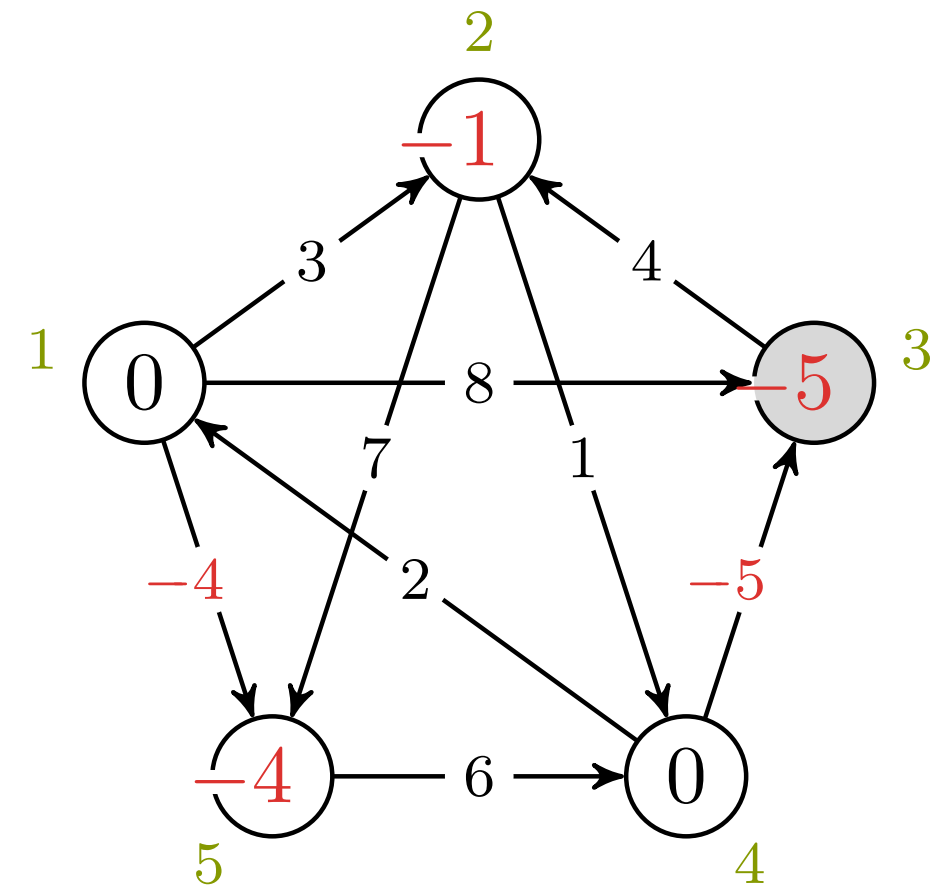
D	1					0	1
	2					-1	2
	3						3
	4						4
	5						5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4   $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6     $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9    DIJKSTRA( $G, \hat{w}, u$ )
10   for each vertex  $v \in G.V$ 
11      $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 

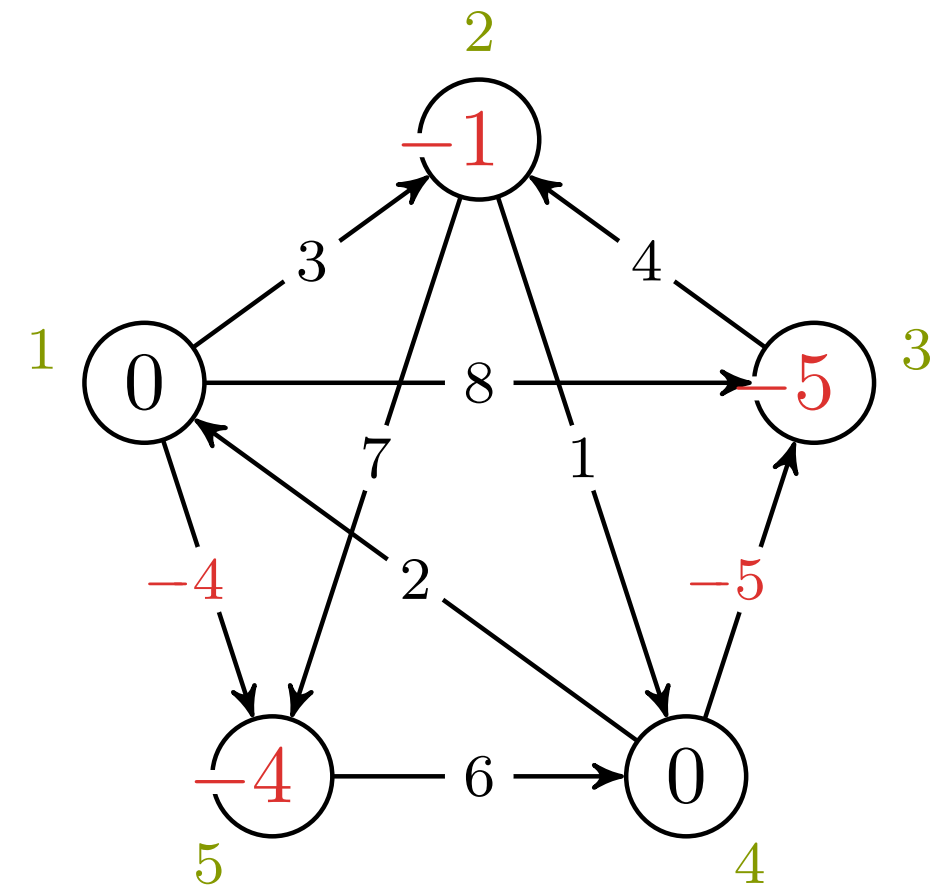
```



D	1						0	1	h
	2						-1	2	
	3							3	
	4							4	
	5							5	
		1	2	3	4	5			

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```

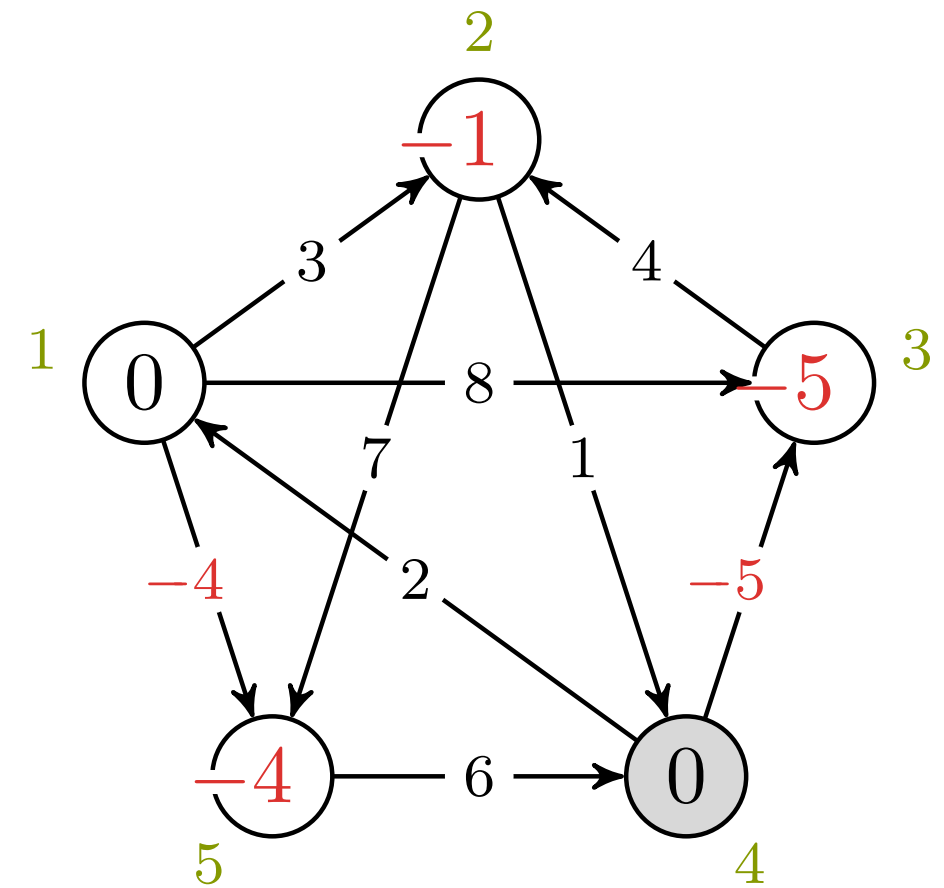


D	1					0	1
	2					-1	2
	3					-5	3
	4						4
	5						5
		1	2	3	4	5	

h

```

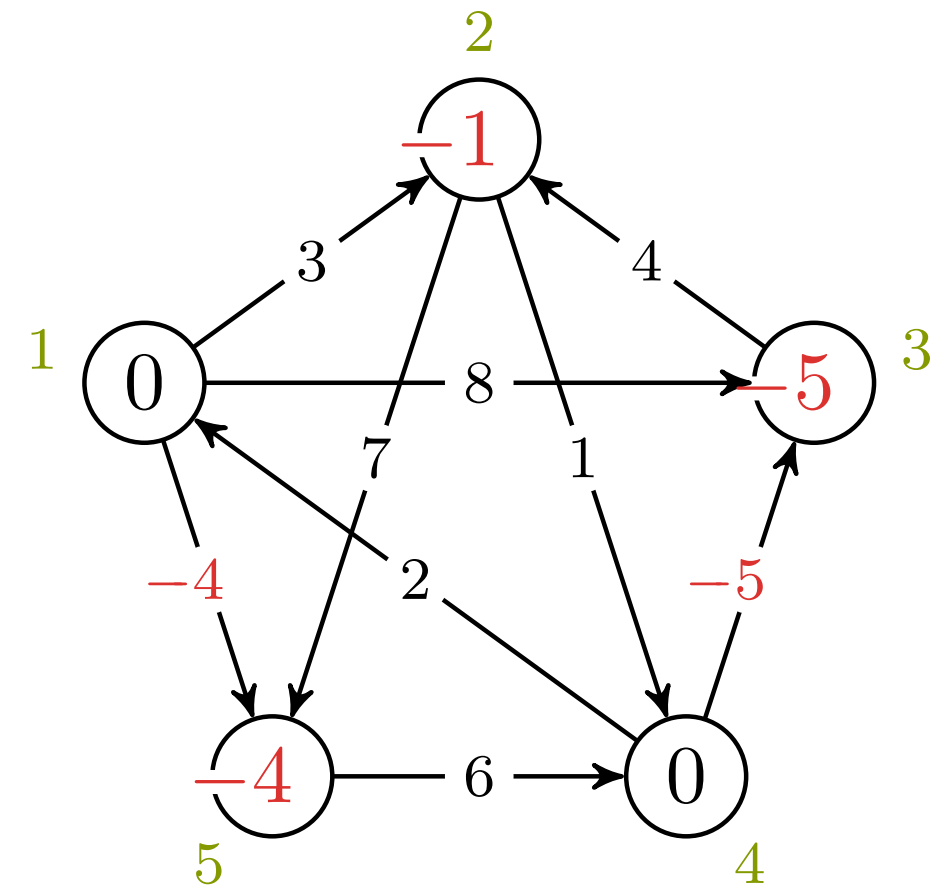
JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```



D	1						0	1	h
	2						-1	2	
	3						-5	3	
	4							4	
	5							5	
		1	2	3	4	5			

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```



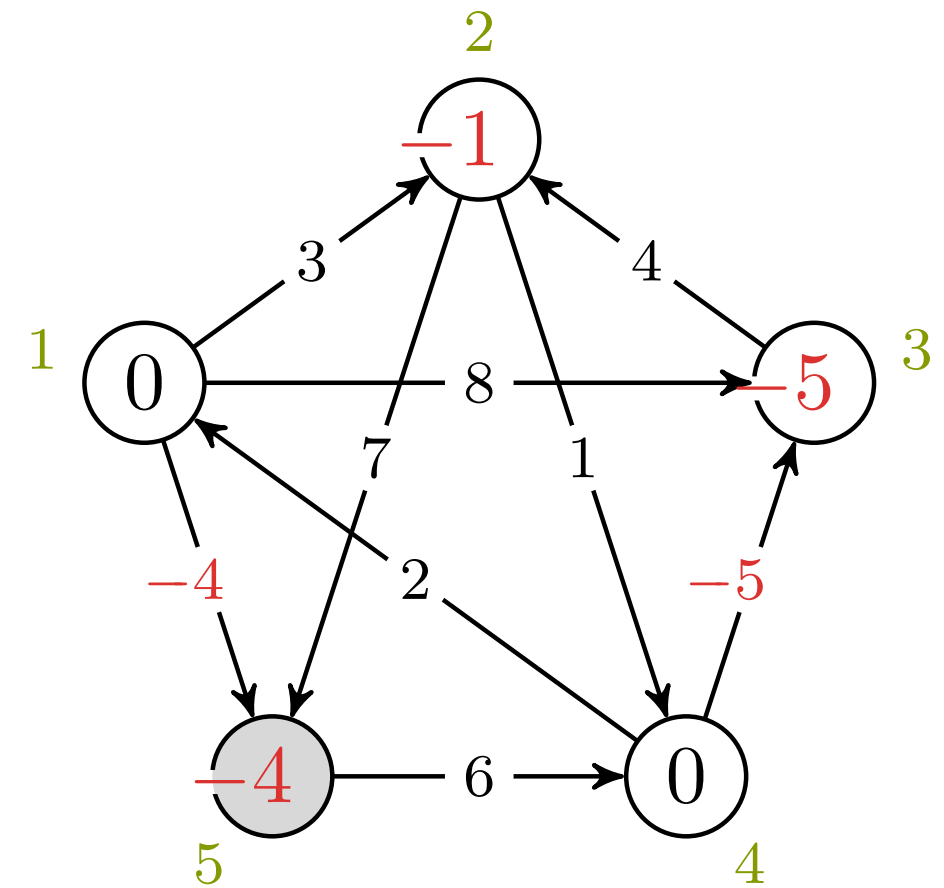
D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5						5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4     $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6     $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9    DIJKSTRA( $G, \hat{w}, u$ )
10   for each vertex  $v \in G.V$ 
11      $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 

```



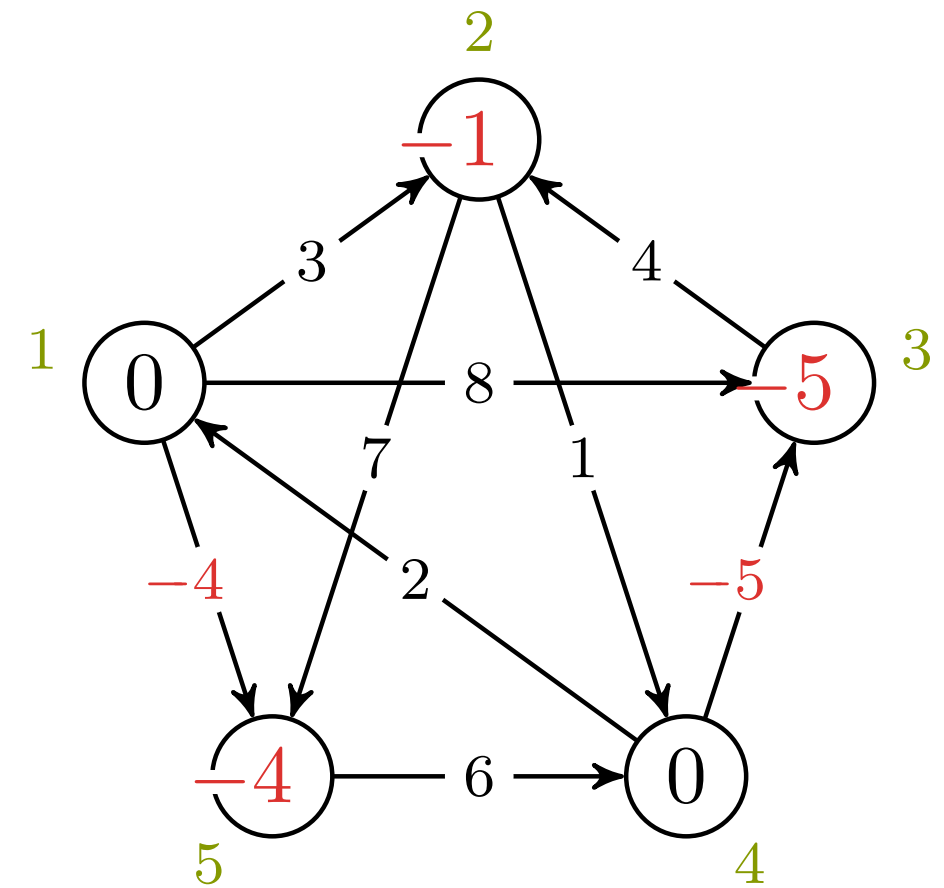
D

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

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```



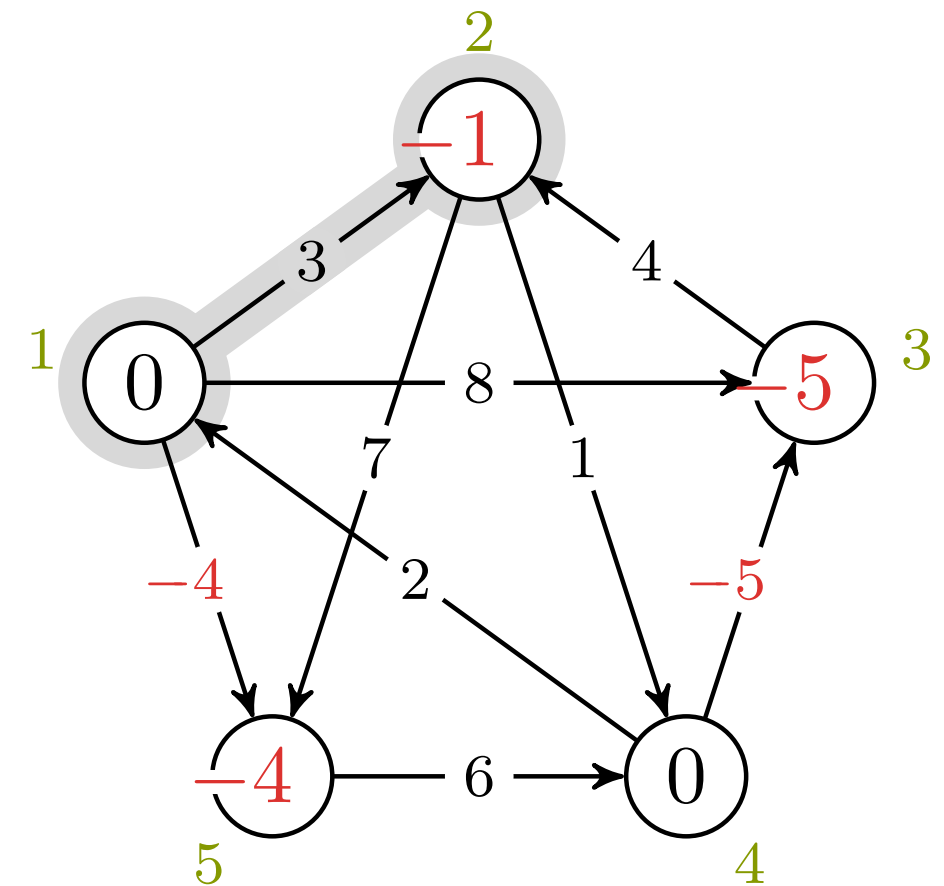
D

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

h

```

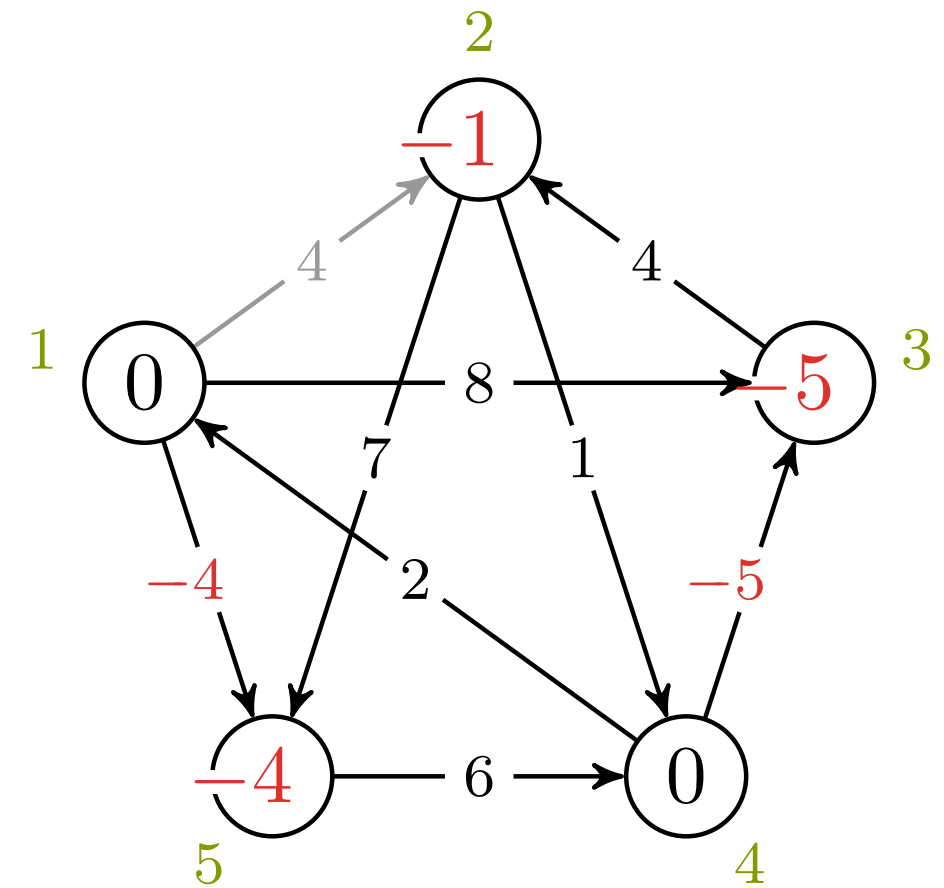
JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```



D	1						0	1	h
	2						-1	2	
	3						-5	3	
	4						0	4	
	5						-4	5	
		1	2	3	4	5			


```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```

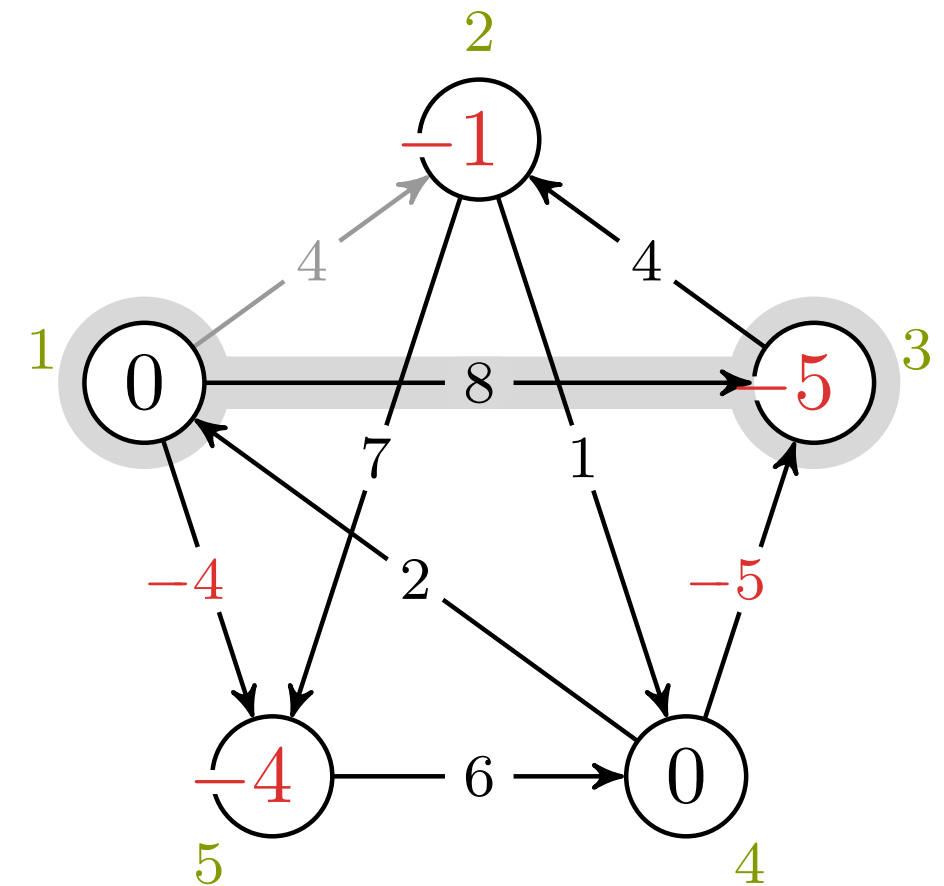


D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```

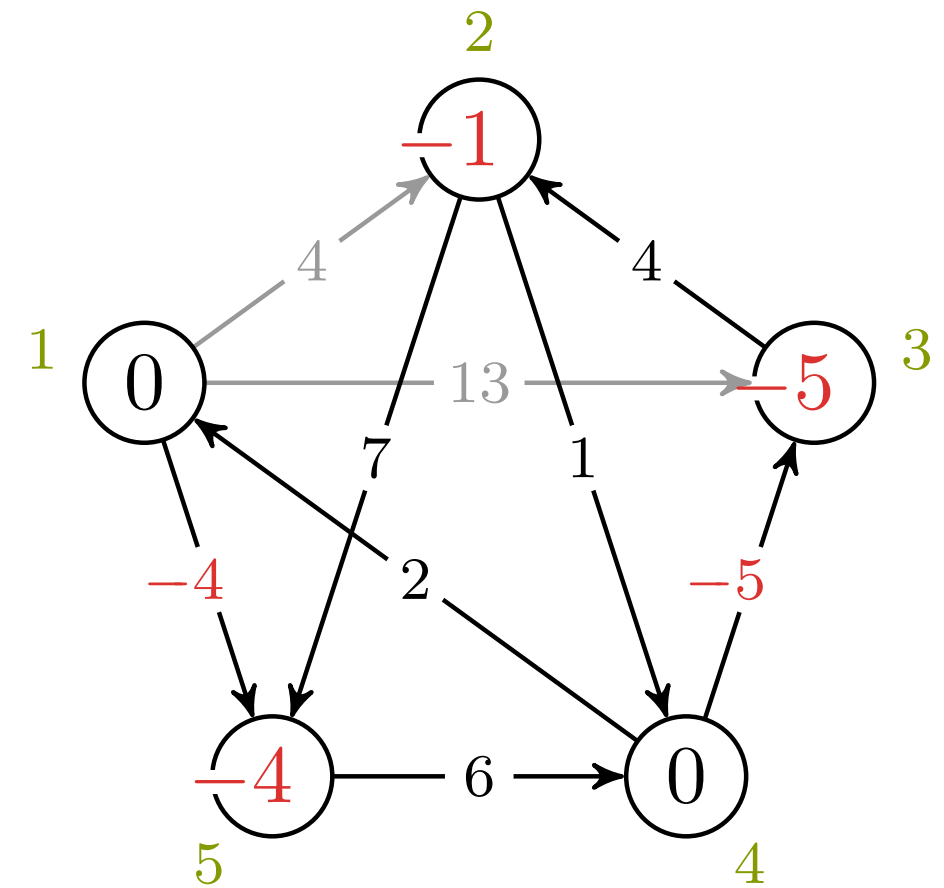


D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```

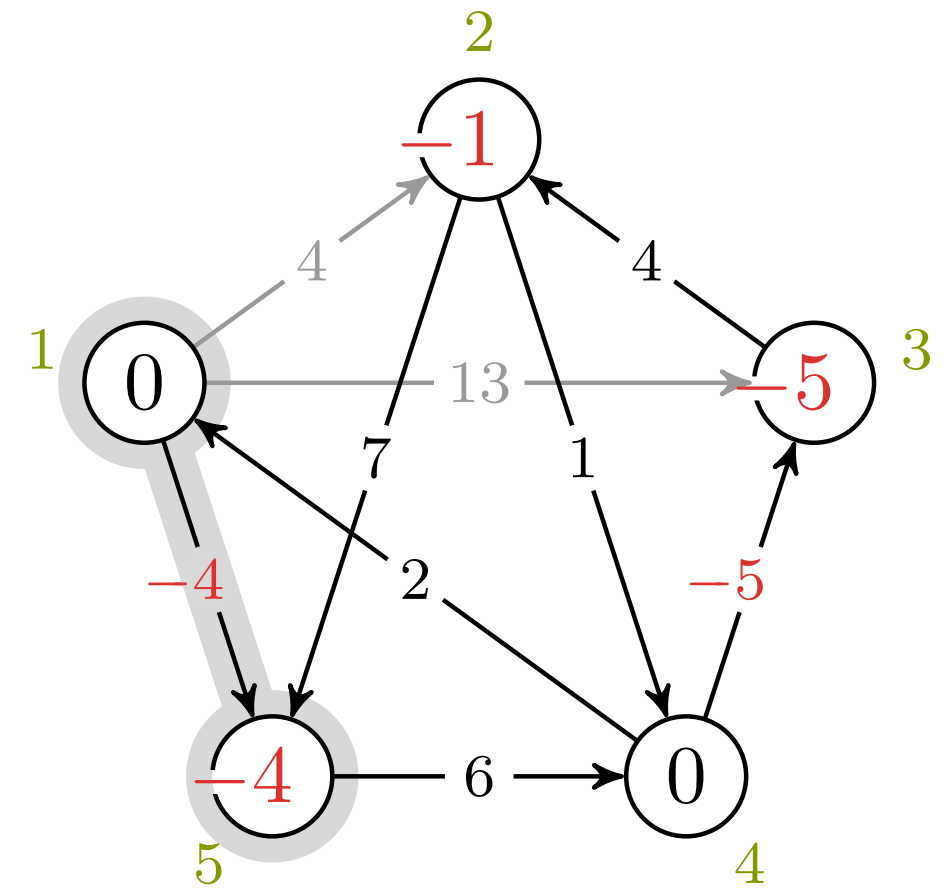


D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

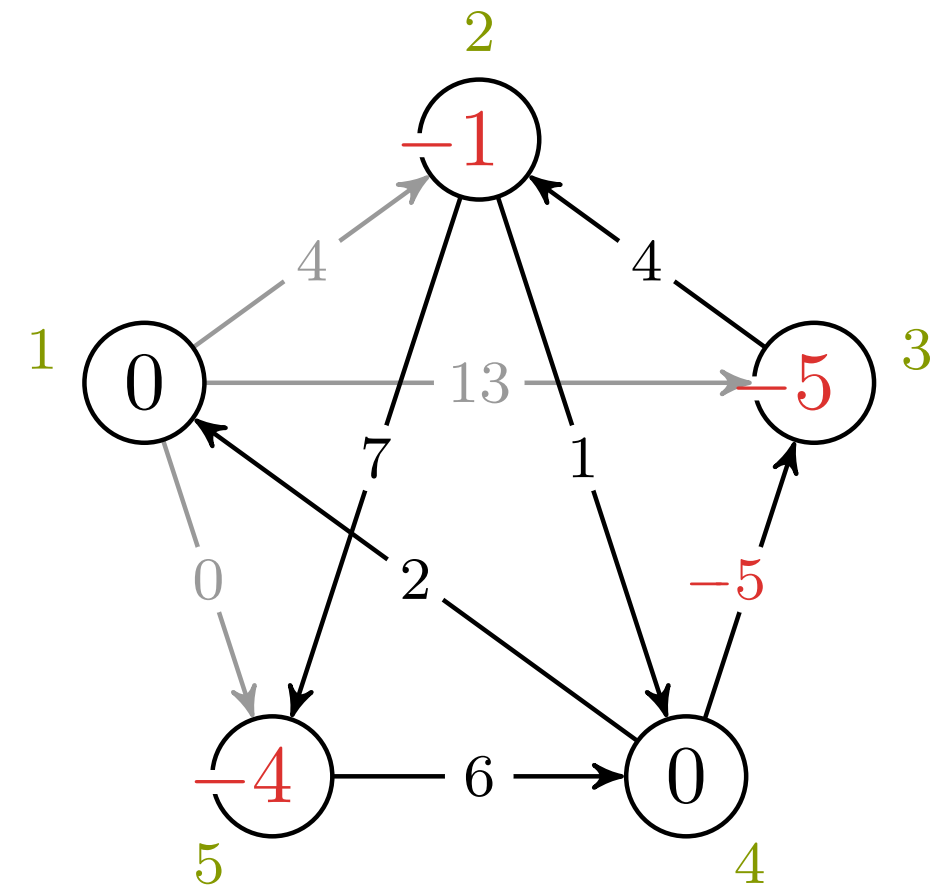


D	1						0	1
	2						-1	2
	3						-5	3
	4						0	4
	5						-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```

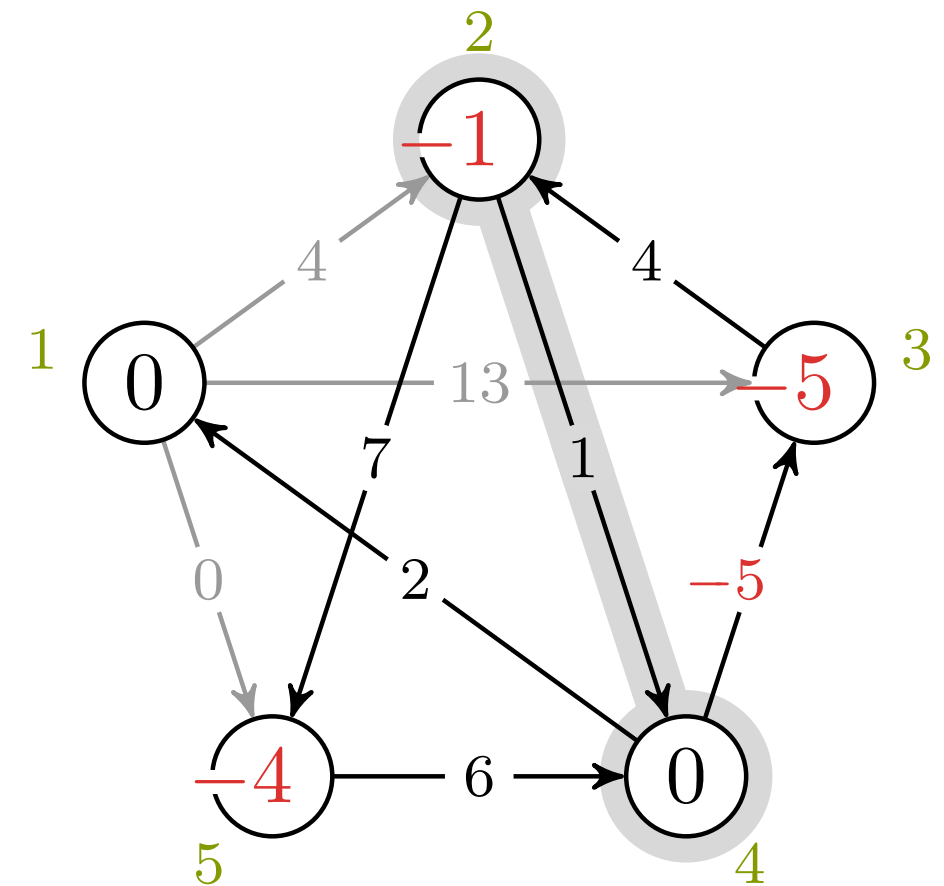


D	1						0	1
	2						-1	2
	3						-5	3
	4						0	4
	5						-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```



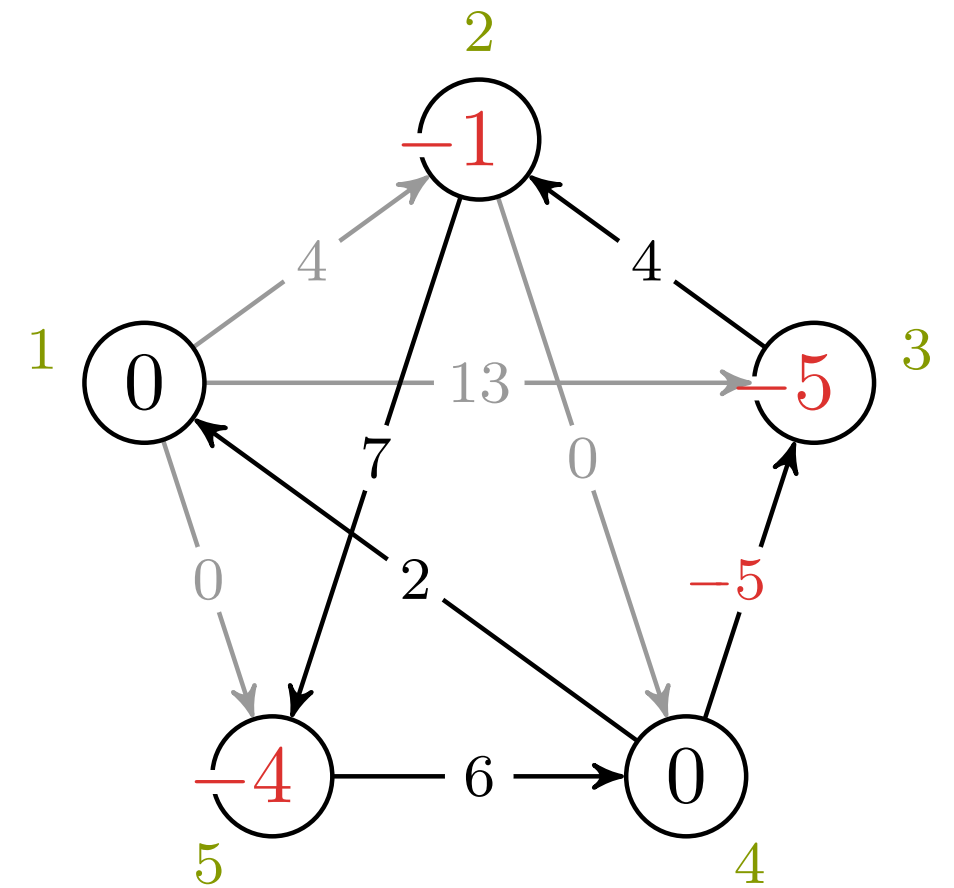
1						0	1
2						-1	2
3						-5	3
4						0	4
5						-4	5
	1	2	3	4	5		

D

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```

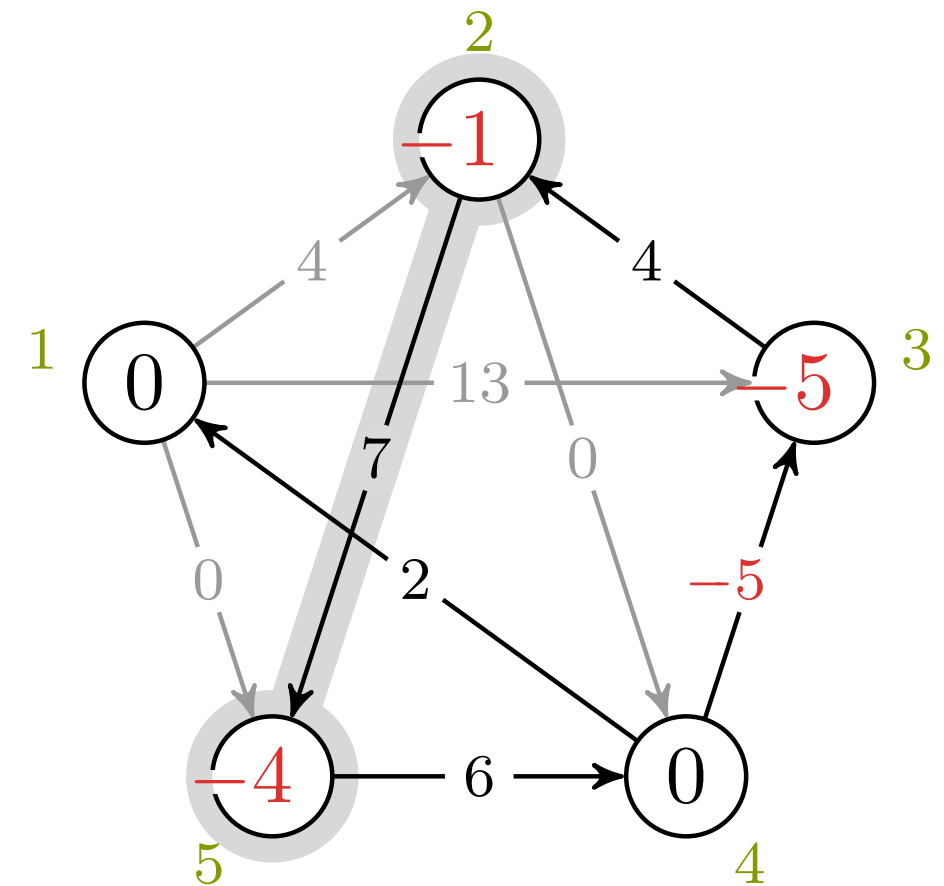


D	1						0	1
	2						-1	2
	3						-5	3
	4						0	4
	5						-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

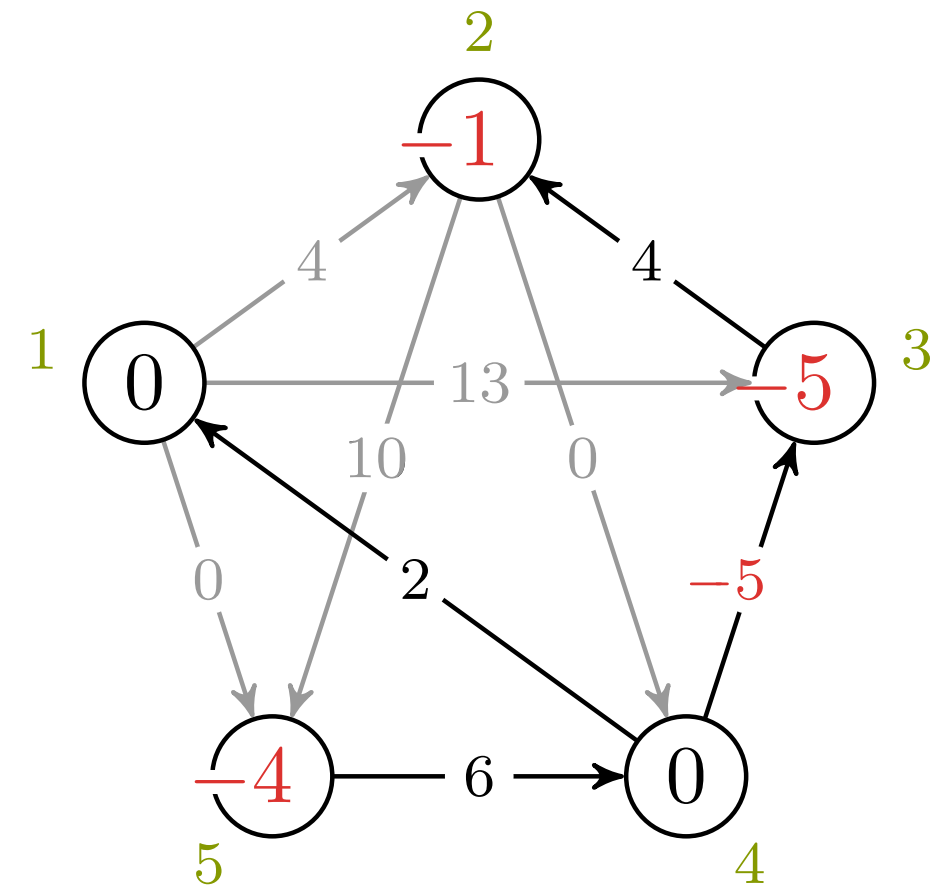


D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	

h


```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

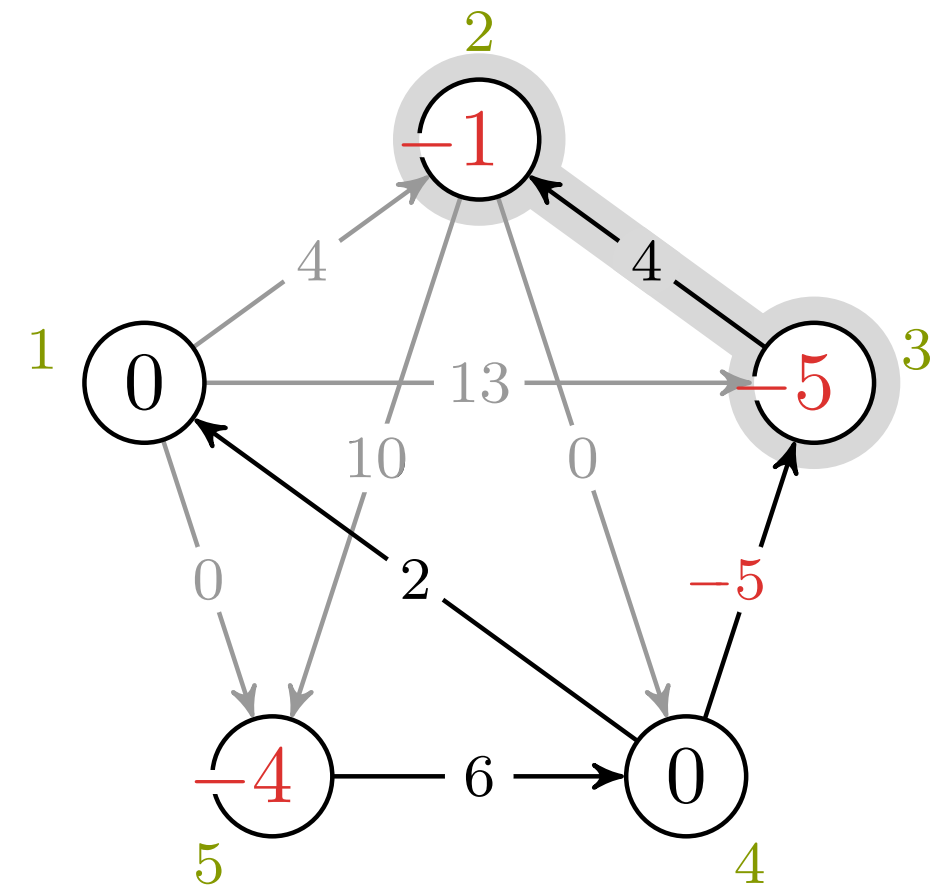


D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

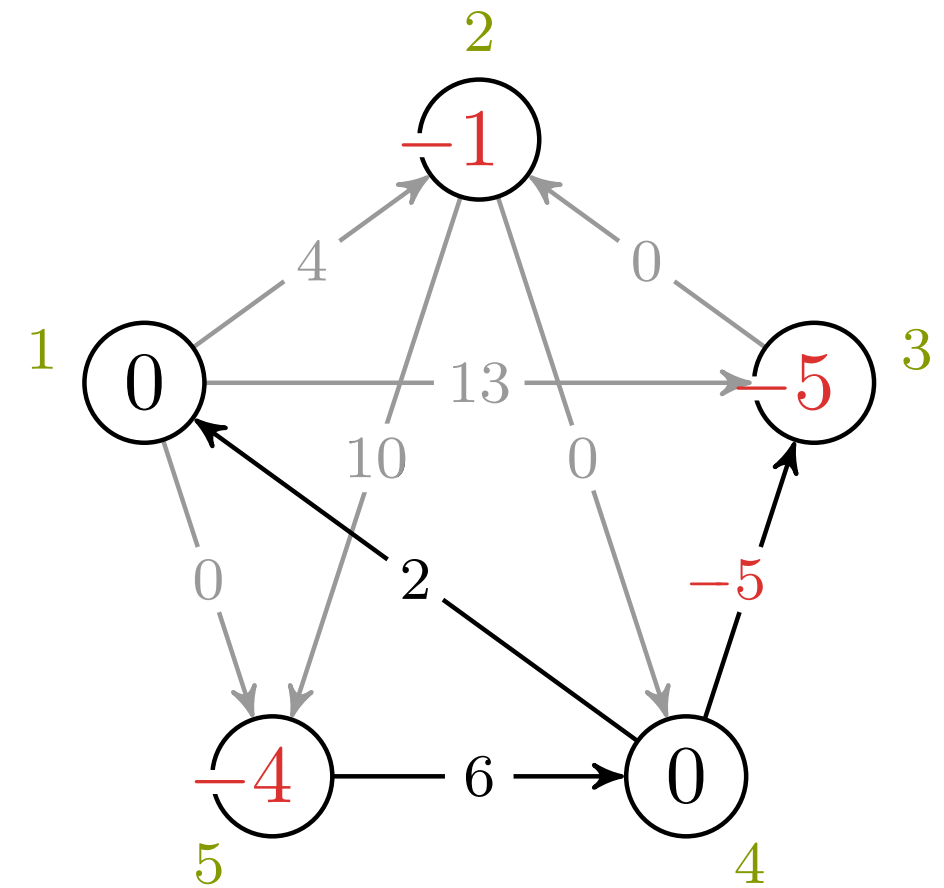


D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```

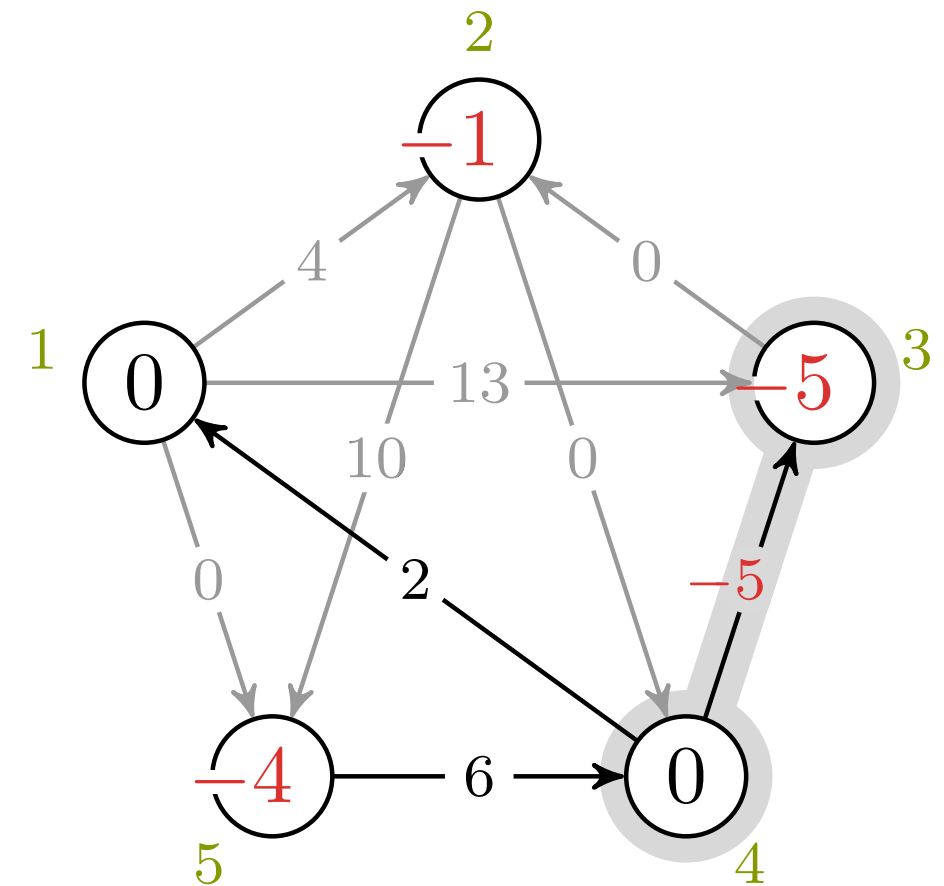


D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

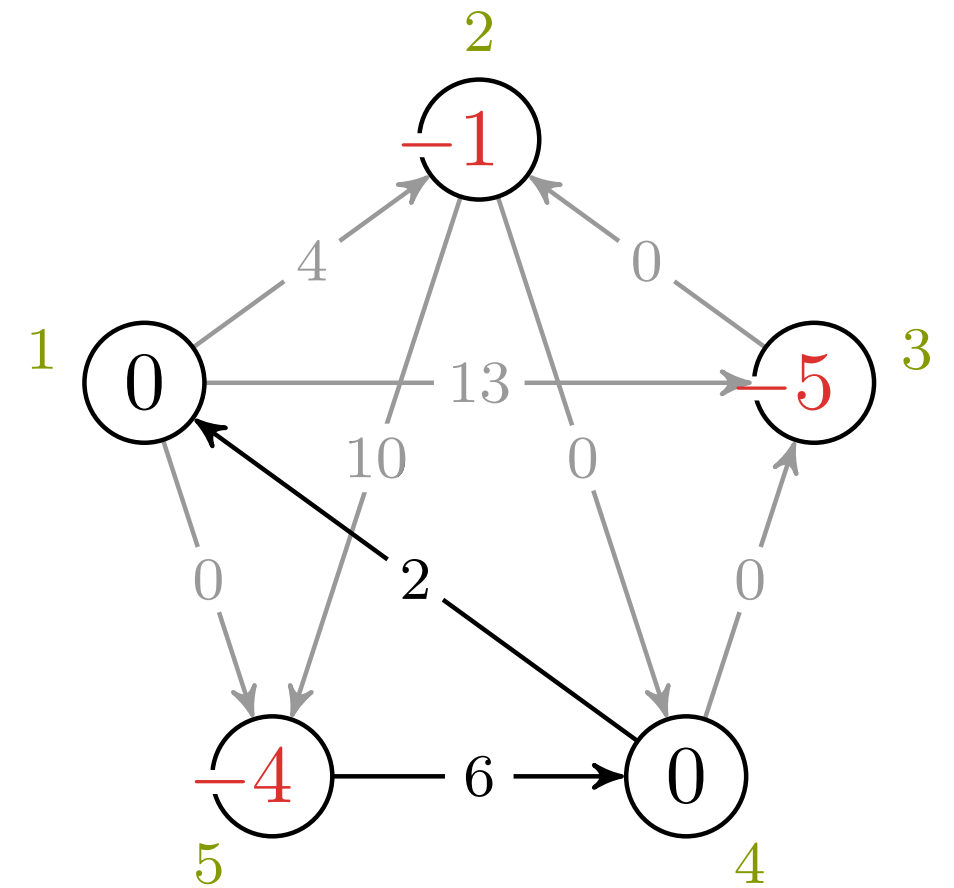


D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```

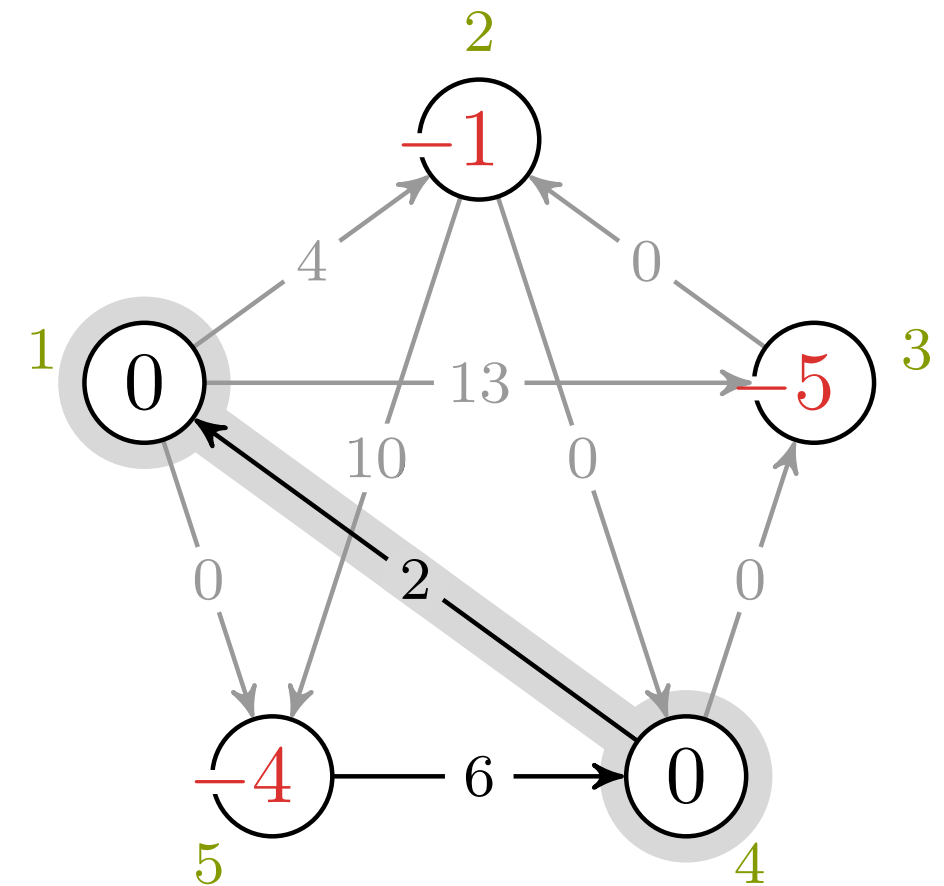


D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```



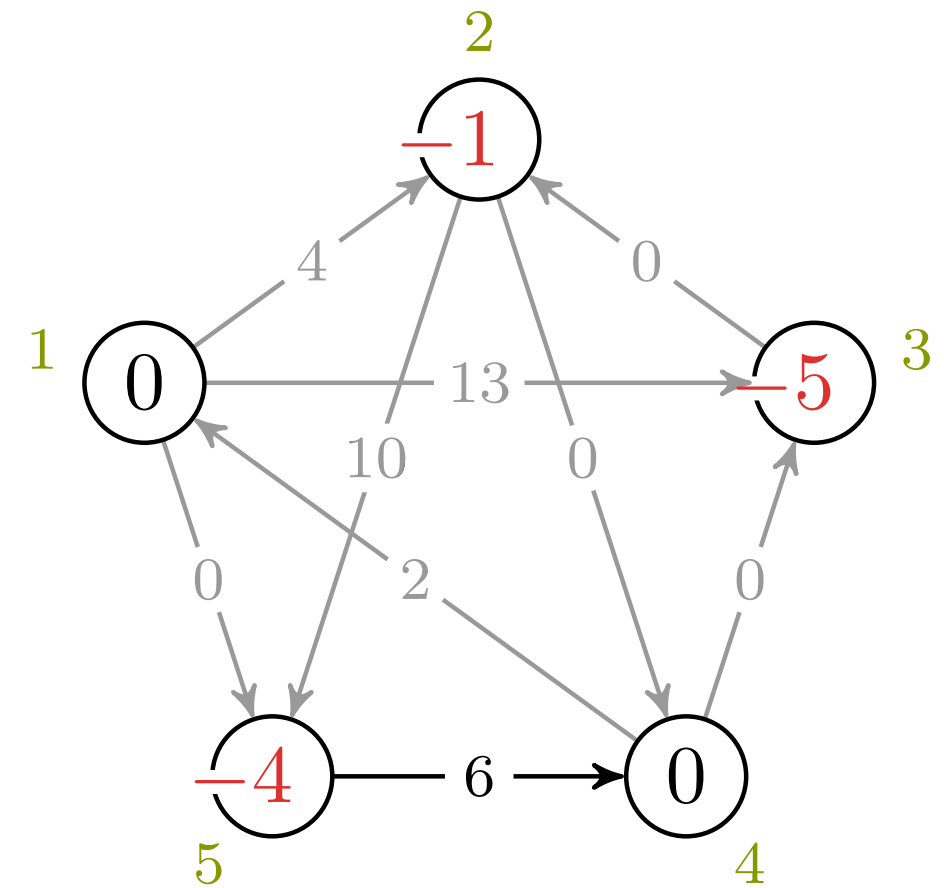
1						0	1
2						-1	2
3						-5	3
4						0	4
5						-4	5
	1	2	3	4	5		

D

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

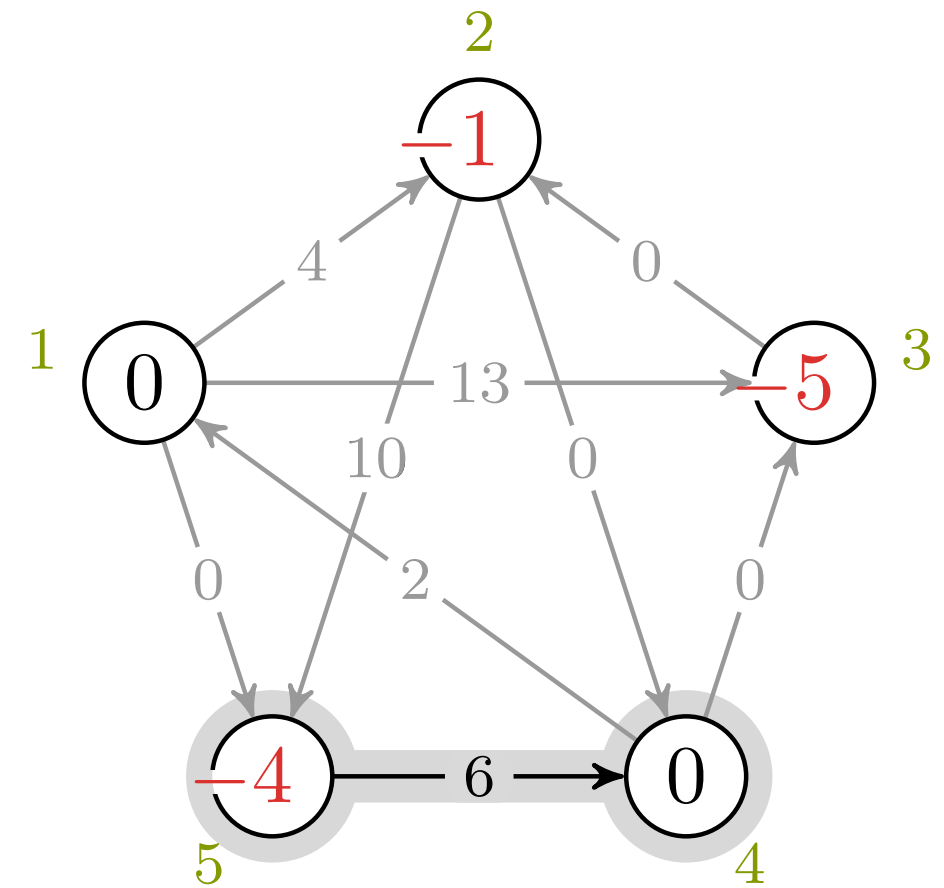


D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```



D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	

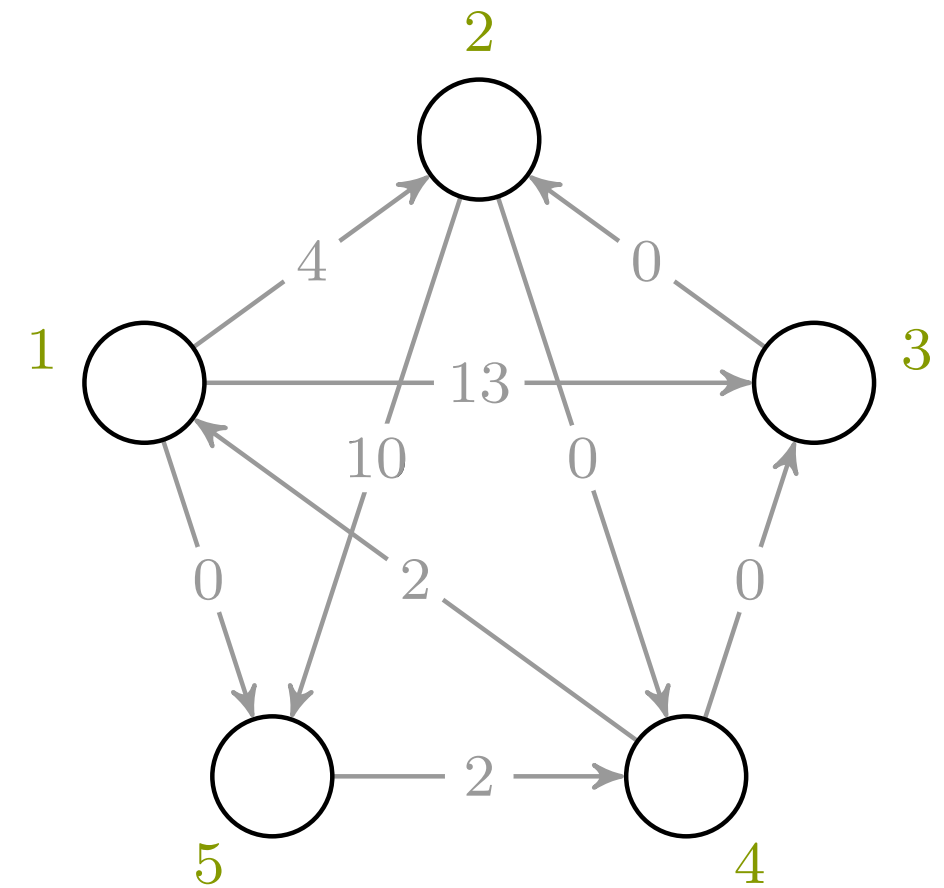
h

JOHNSON(G, w)

```

1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 

```

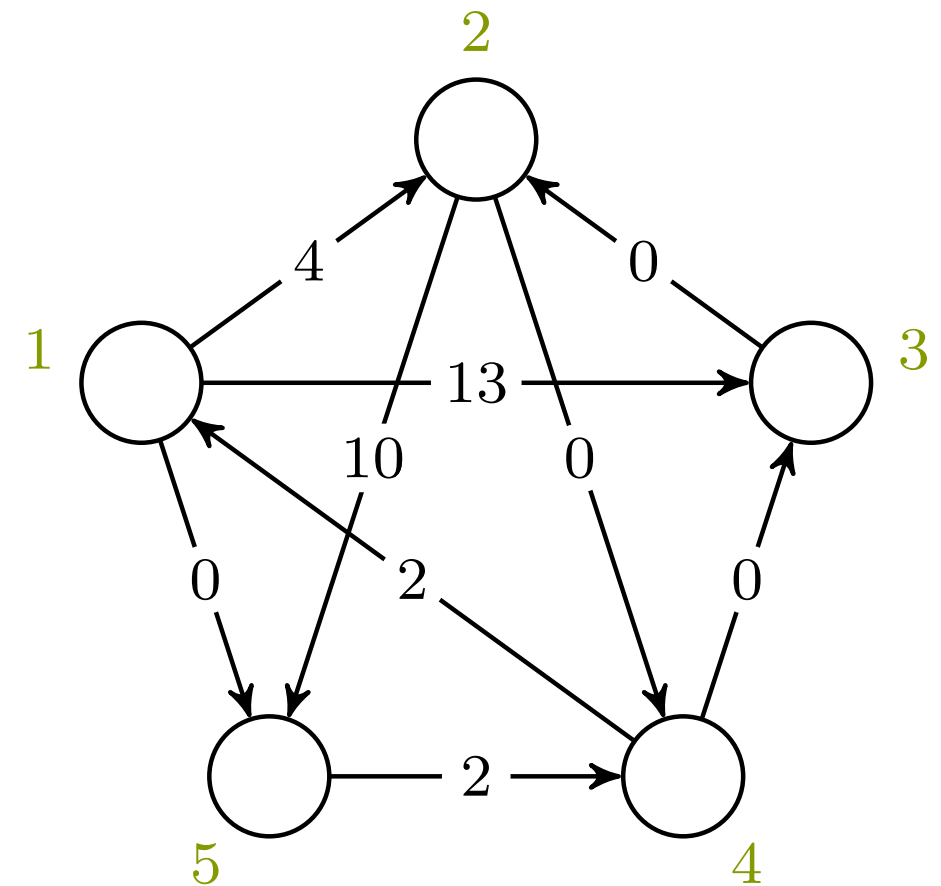


D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

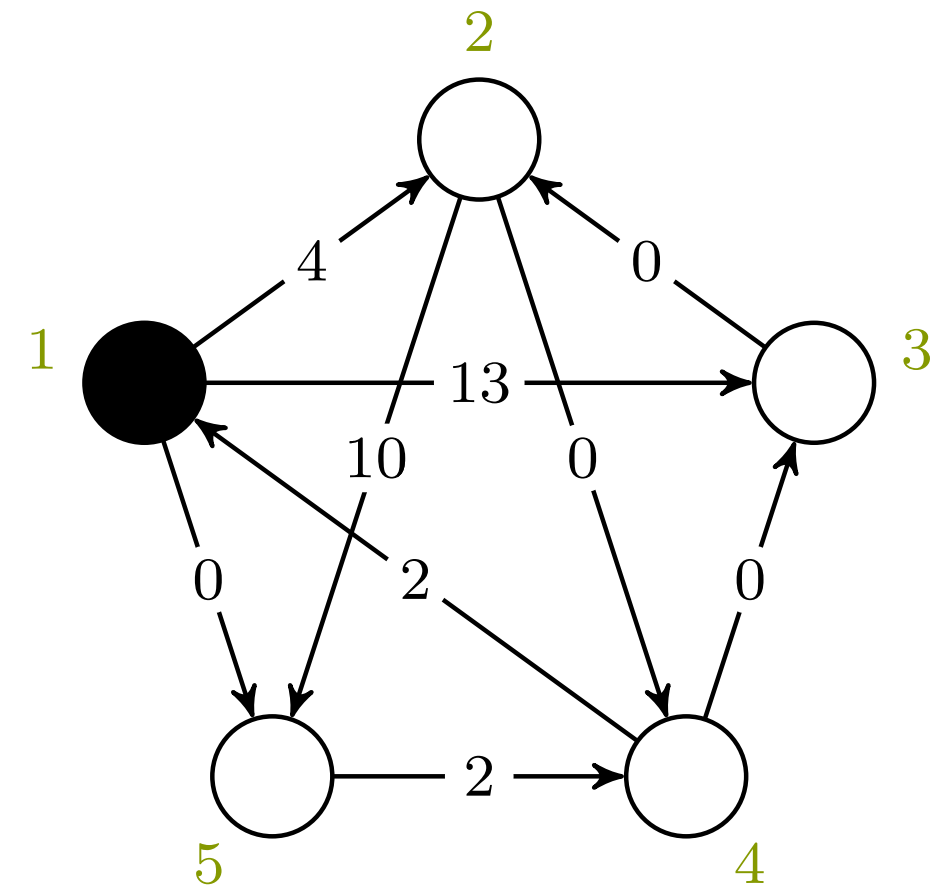


D	1						0	1
	2						-1	2
	3						-5	3
	4						0	4
	5						-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

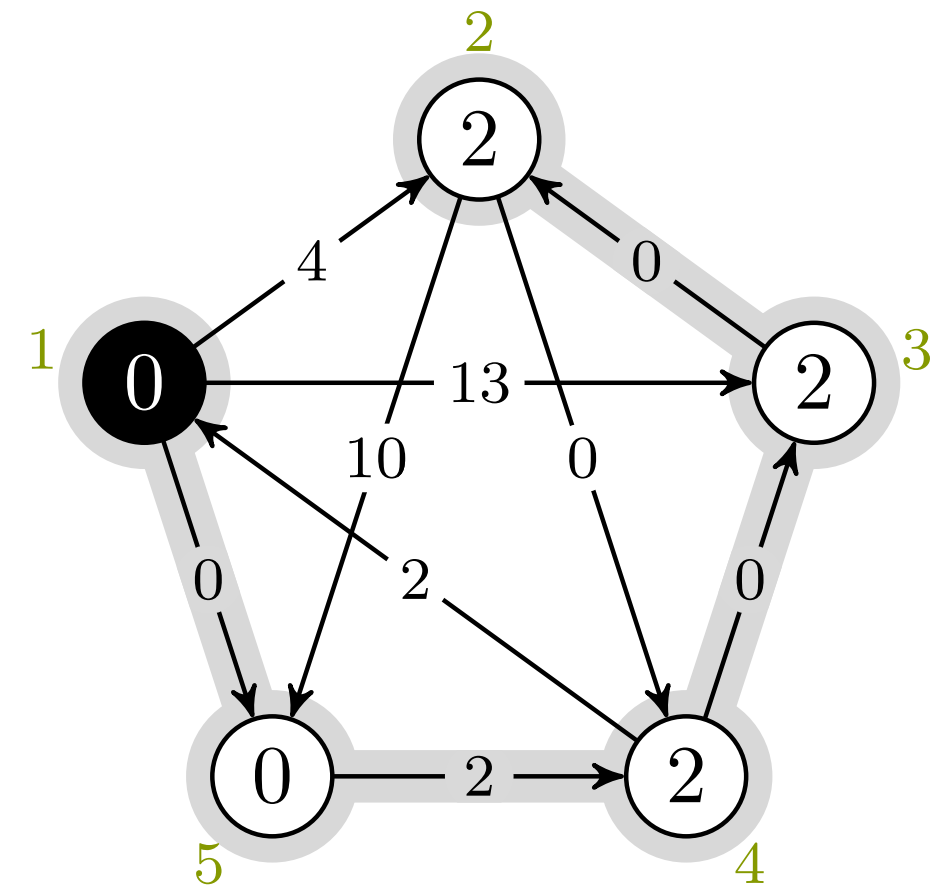


D	1						0	1
	2						-1	2
	3						-5	3
	4						0	4
	5						-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10  for each vertex  $v \in G.V$ 
11       $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

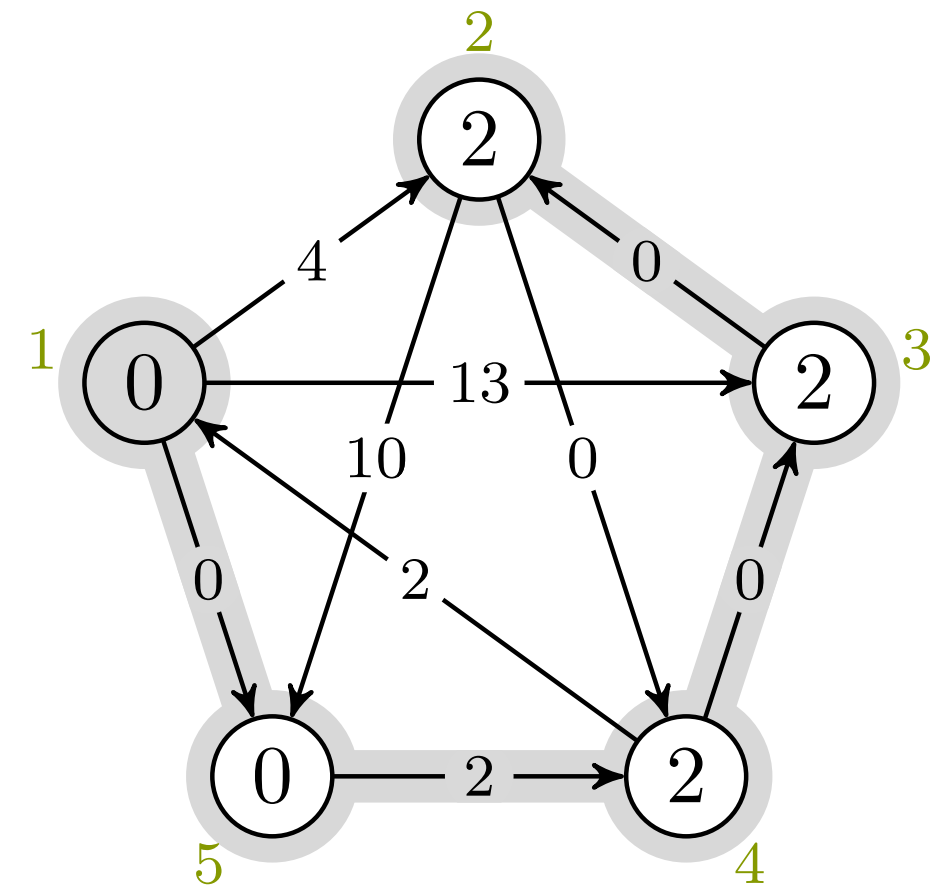


D	1						0	1
	2						-1	2
	3						-5	3
	4						0	4
	5						-4	5
		1	2	3	4	5		

h

```

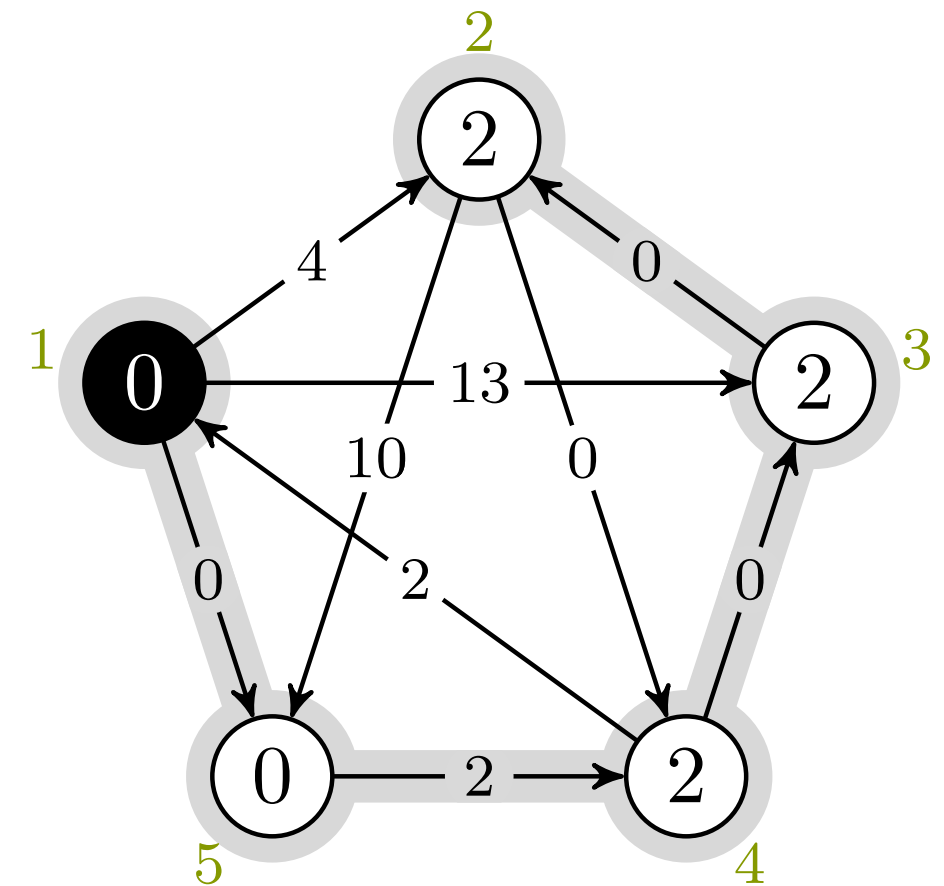
JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```



D	1					0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10  for each vertex  $v \in G.V$ 
11       $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

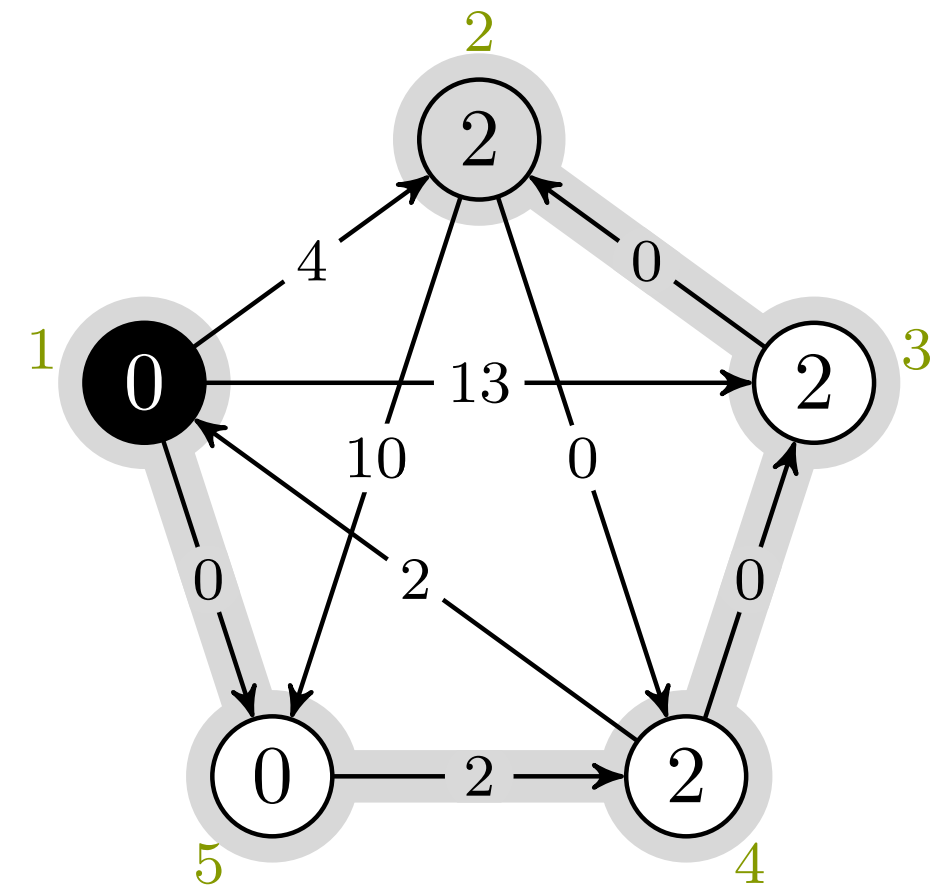


D	1	0					0	1
	2						-1	2
	3						-5	3
	4						0	4
	5						-4	5
		1	2	3	4	5		

h

```

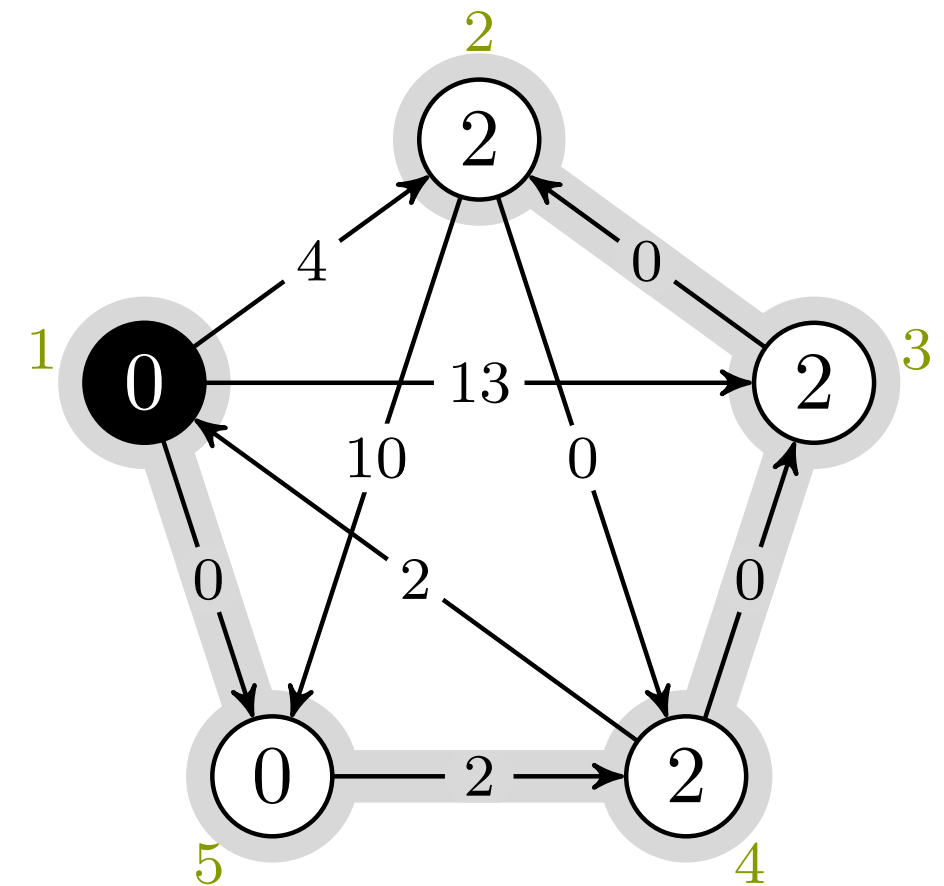
JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```



D	1	0				0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10 for each vertex  $v \in G.V$ 
11      $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```

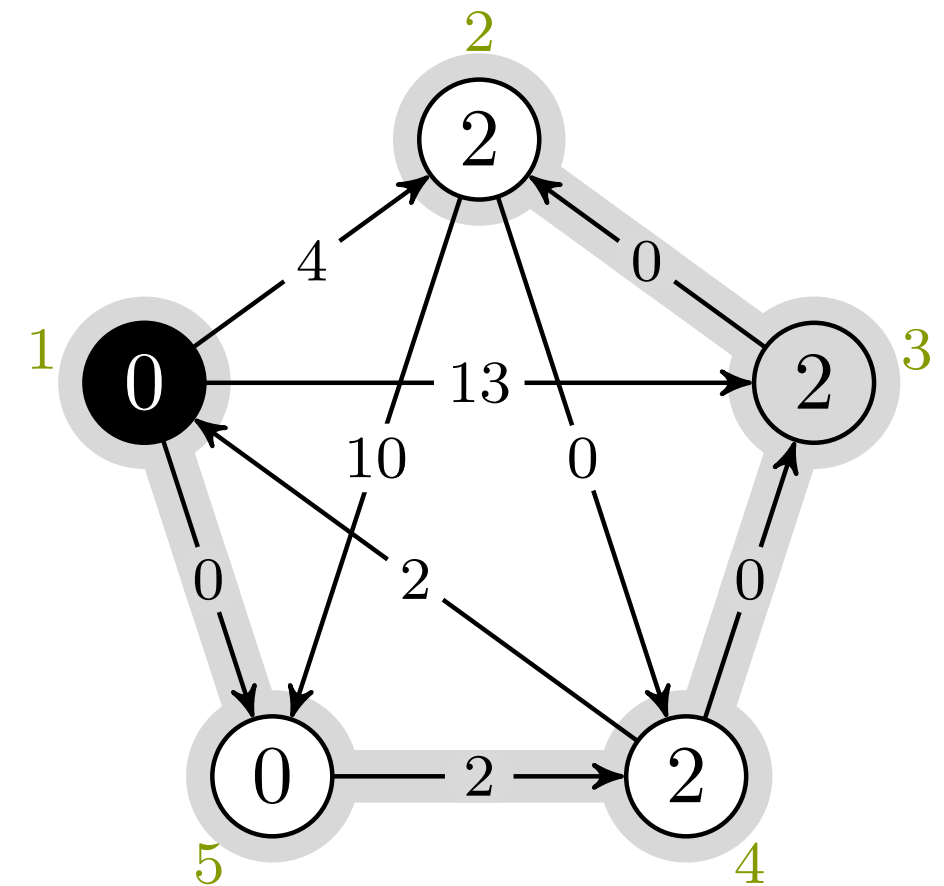


D	1	0	1				0	1
	2						-1	2
	3						-5	3
	4						0	4
	5						-4	5
		1	2	3	4	5		

h


```

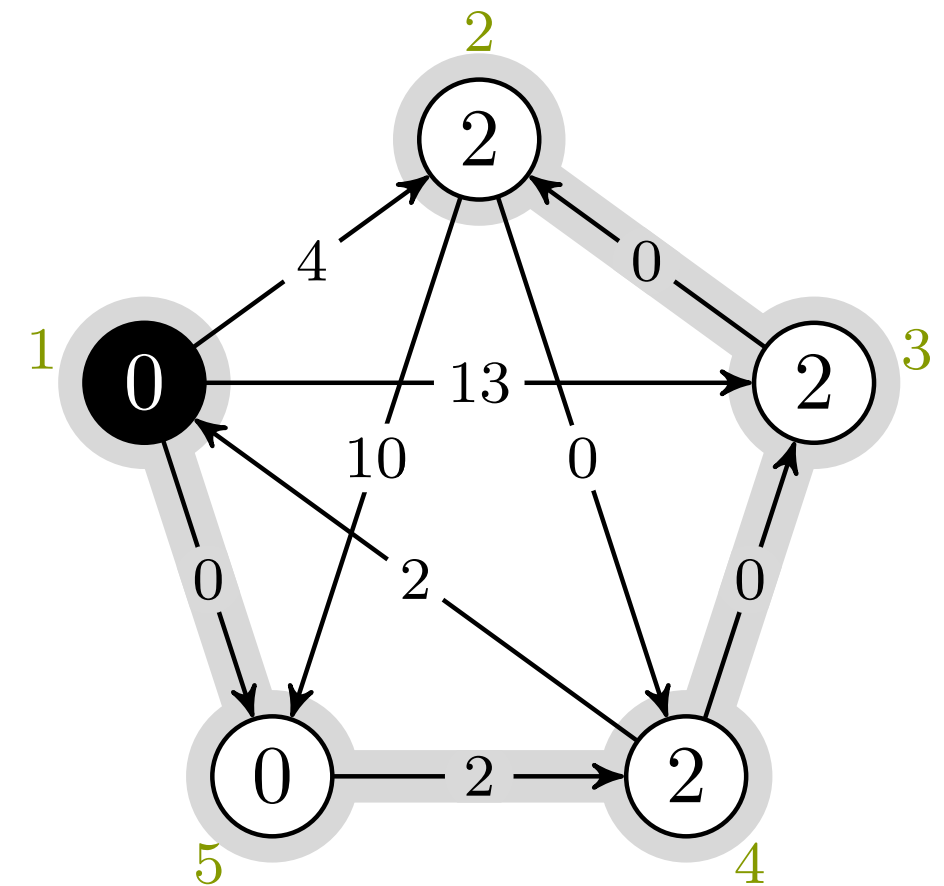
JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```



D	1	0	1			0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10  for each vertex  $v \in G.V$ 
11       $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

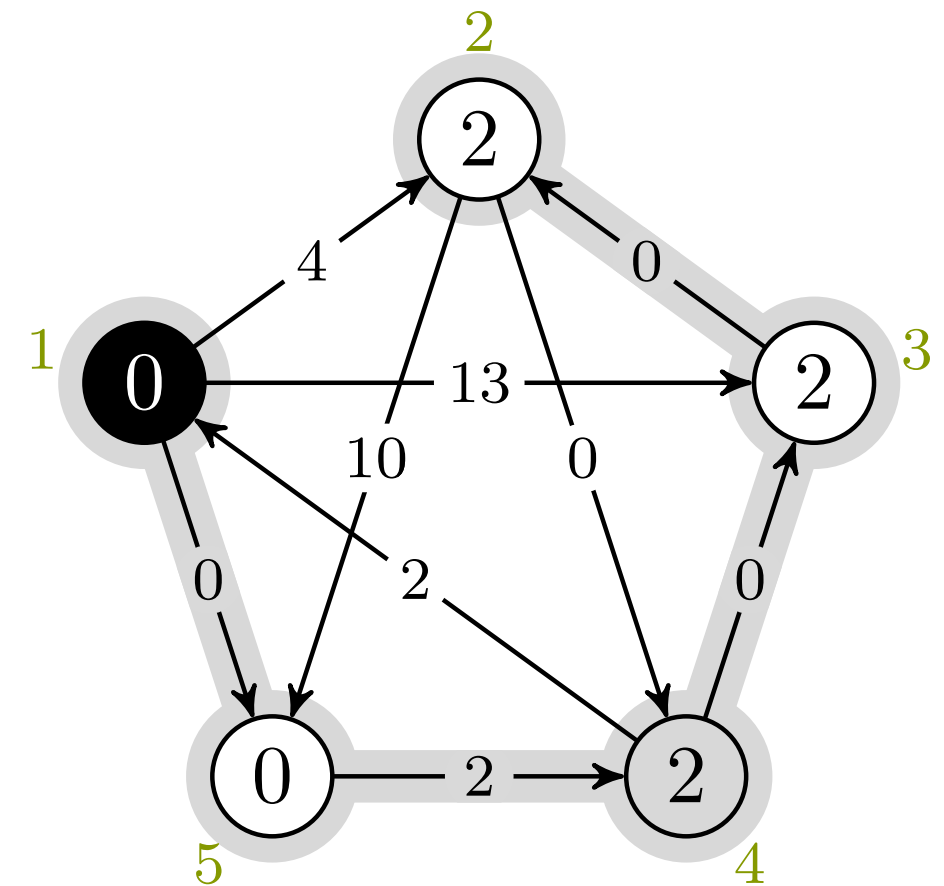


D	1	0	1	-3			0	1
	2						-1	2
	3						-5	3
	4						0	4
	5						-4	5
		1	2	3	4	5		

h

```

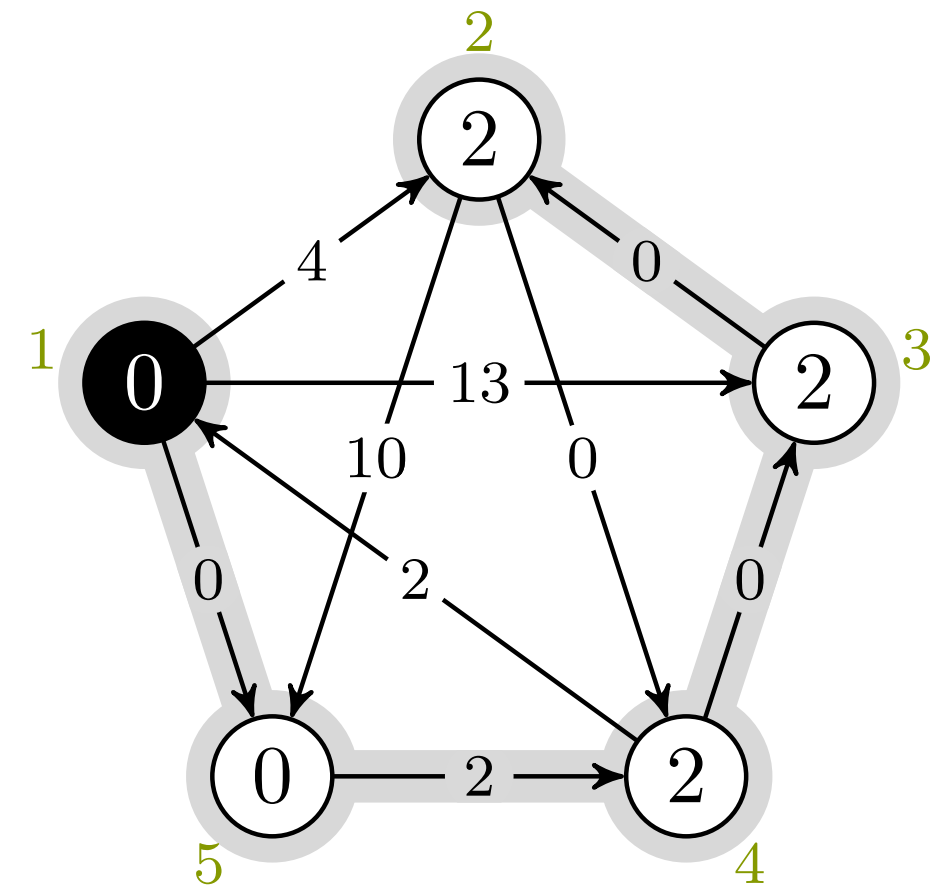
JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```



D	1	0	1	-3		0	1
	2					-1	2
	3					-5	3
	4					0	4
	5					-4	5
		1	2	3	4	5	h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10  for each vertex  $v \in G.V$ 
11       $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

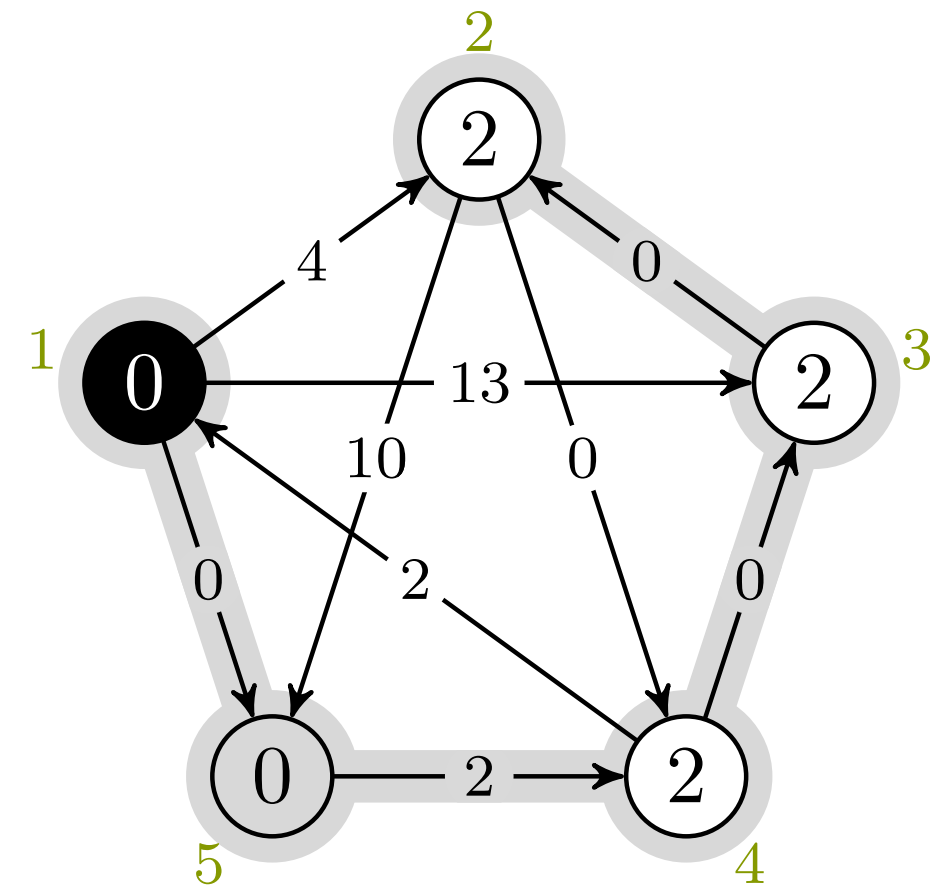


D	1	0	1	-3	2		0	1
	2						-1	2
	3						-5	3
	4						0	4
	5						-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

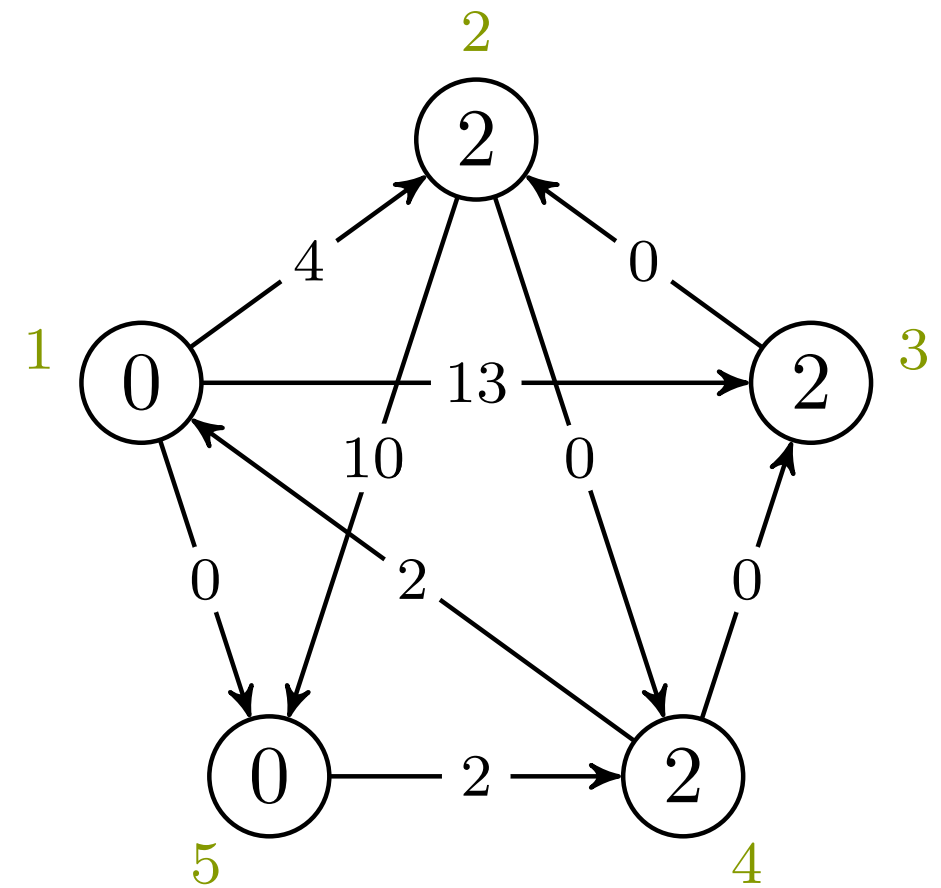


D	1	0	1	-3	2		0	1
	2						-1	2
	3						-5	3
	4						0	4
	5						-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```



D	1	0	1	-3	2	-4	0	1
	2						-1	2
	3						-5	3
	4						0	4
	5						-4	5
		1	2	3	4	5		

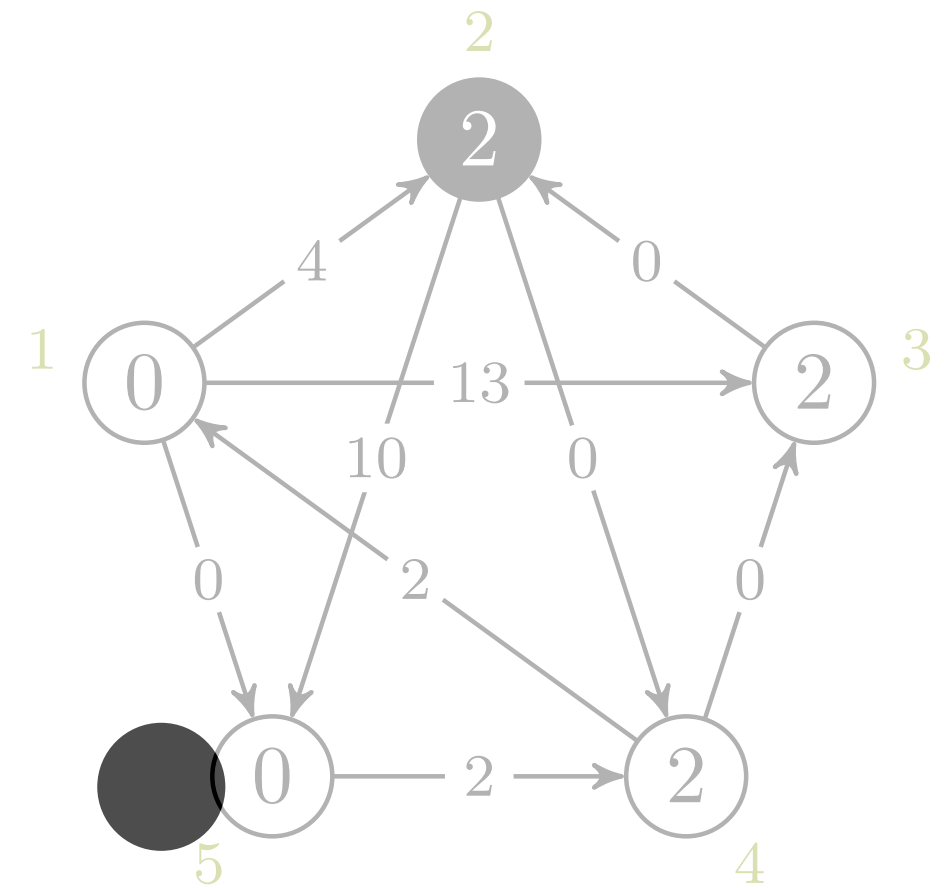
h

JOHNSON(G, w)

```

1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 

```



D

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

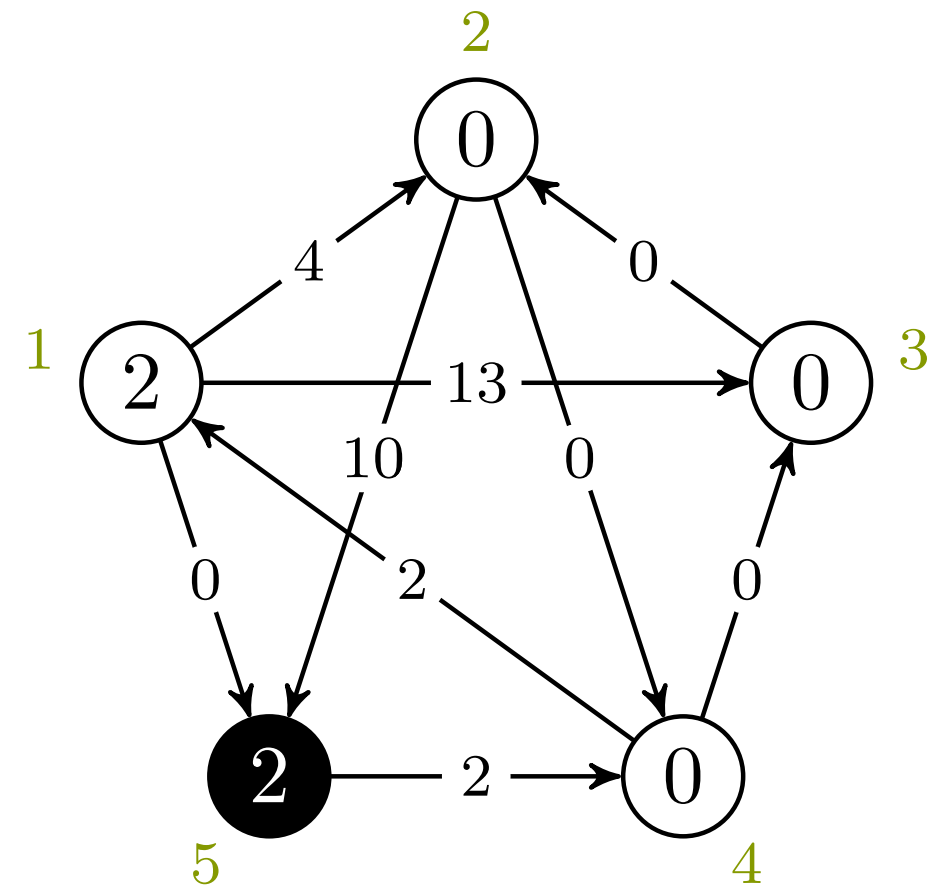
0	1
-1	2
-5	3
0	4
-4	5

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 

```

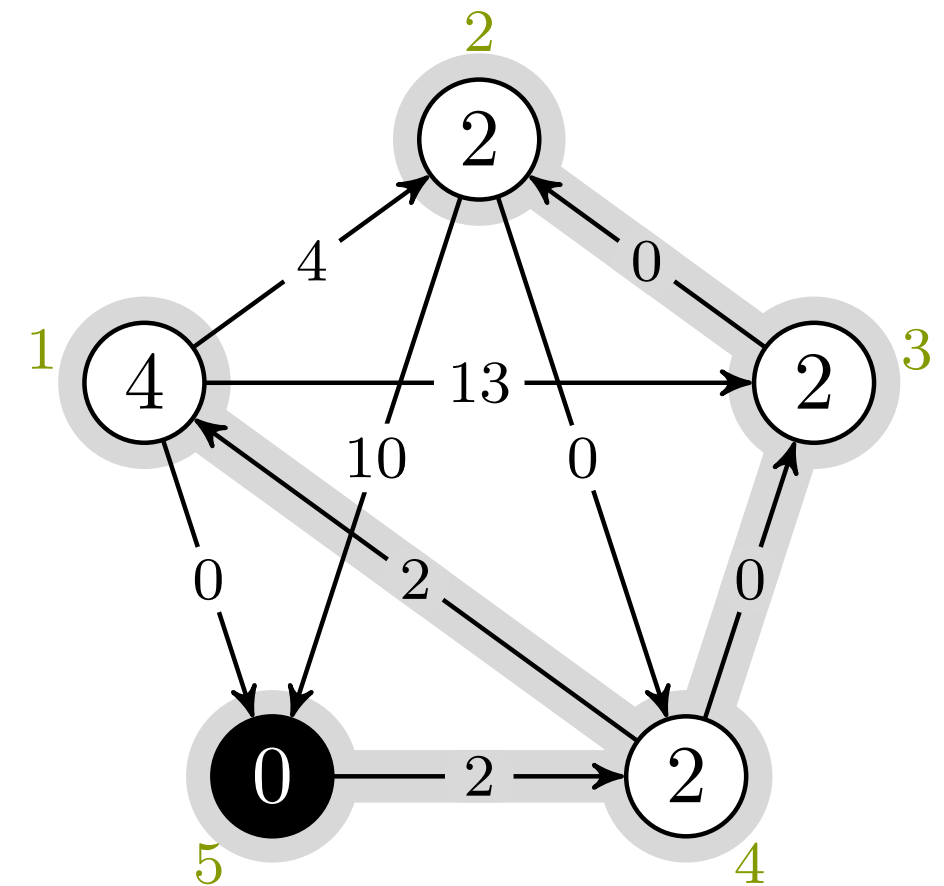


D	1	0	1	-3	2	-4	0	1
	2	3	0	-4	1	-1	-1	2
	3	7	4	0	5	3	-5	3
	4	2	-1	-5	0	-2	0	4
	5						-4	5
		1	2	3	4	5		

h


```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10  for each vertex  $v \in G.V$ 
11       $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

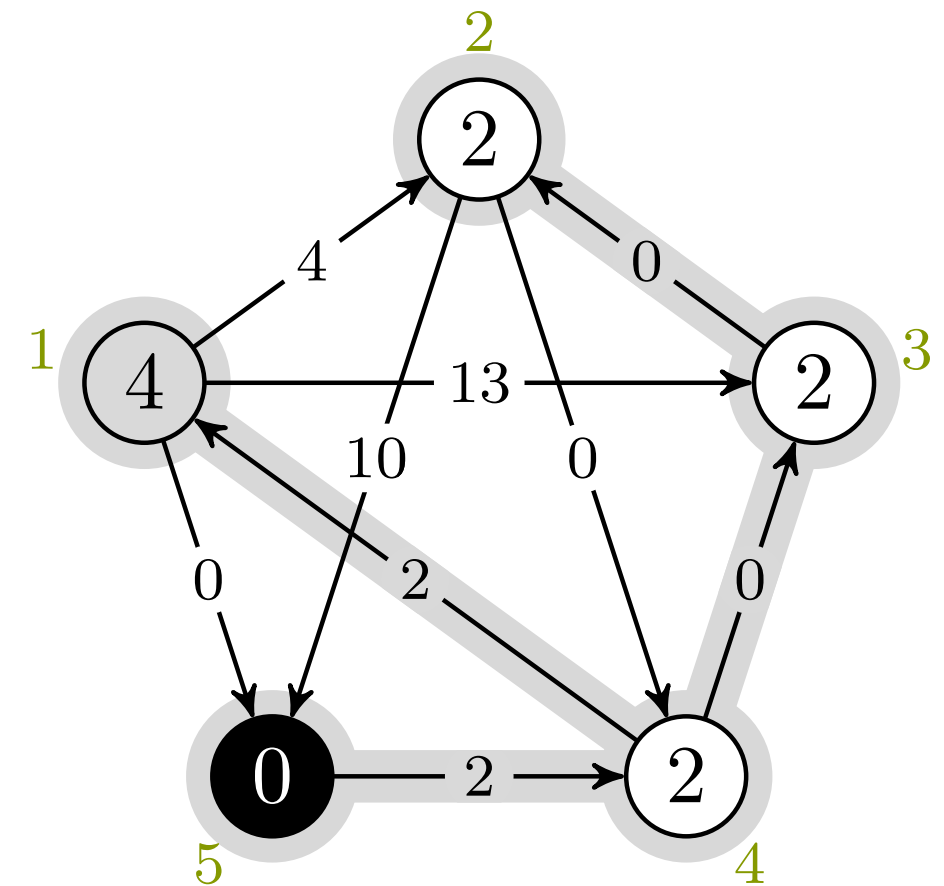


D	1	0	1	-3	2	-4	0	1
	2	3	0	-4	1	-1	-1	2
	3	7	4	0	5	3	-5	3
	4	2	-1	-5	0	-2	0	4
	5						-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

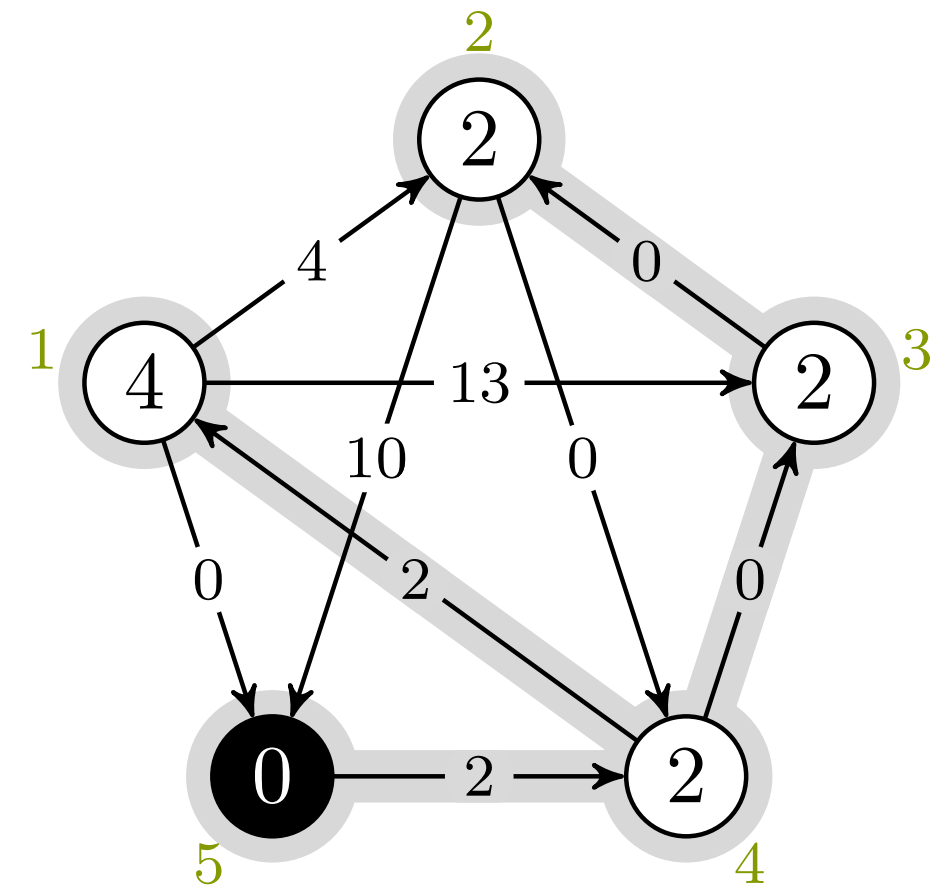


D	1	0	1	-3	2	-4	0	1
	2	3	0	-4	1	-1	-1	2
	3	7	4	0	5	3	-5	3
	4	2	-1	-5	0	-2	0	4
	5						-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10  for each vertex  $v \in G.V$ 
11       $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

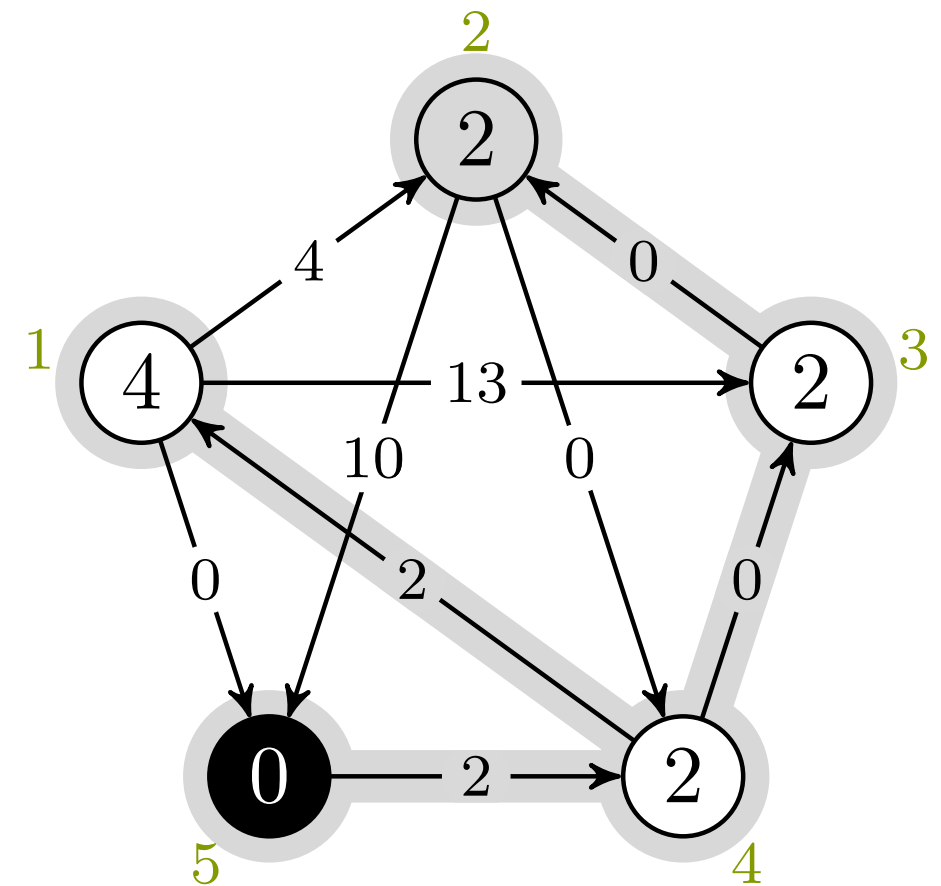


D	1	0	1	-3	2	-4	0	1
	2	3	0	-4	1	-1	-1	2
	3	7	4	0	5	3	-5	3
	4	2	-1	-5	0	-2	0	4
	5	8					-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

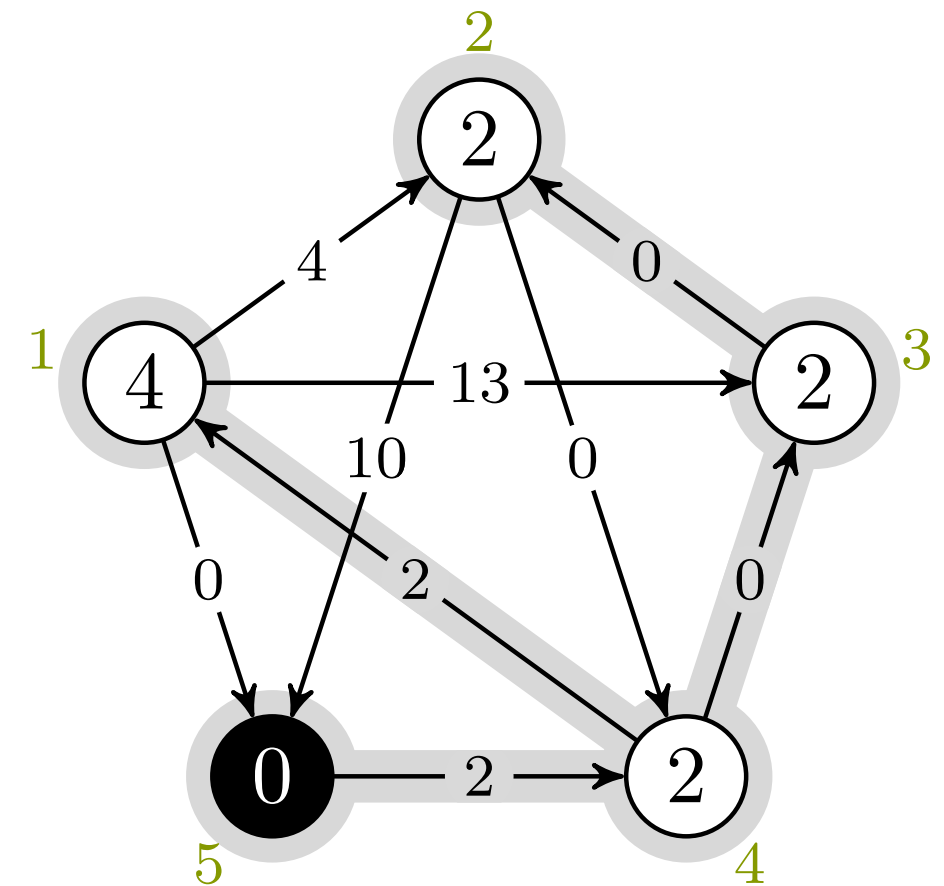


D	1	0	1	-3	2	-4	0	1
	2	3	0	-4	1	-1	-1	2
	3	7	4	0	5	3	-5	3
	4	2	-1	-5	0	-2	0	4
	5	8					-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10  for each vertex  $v \in G.V$ 
11       $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

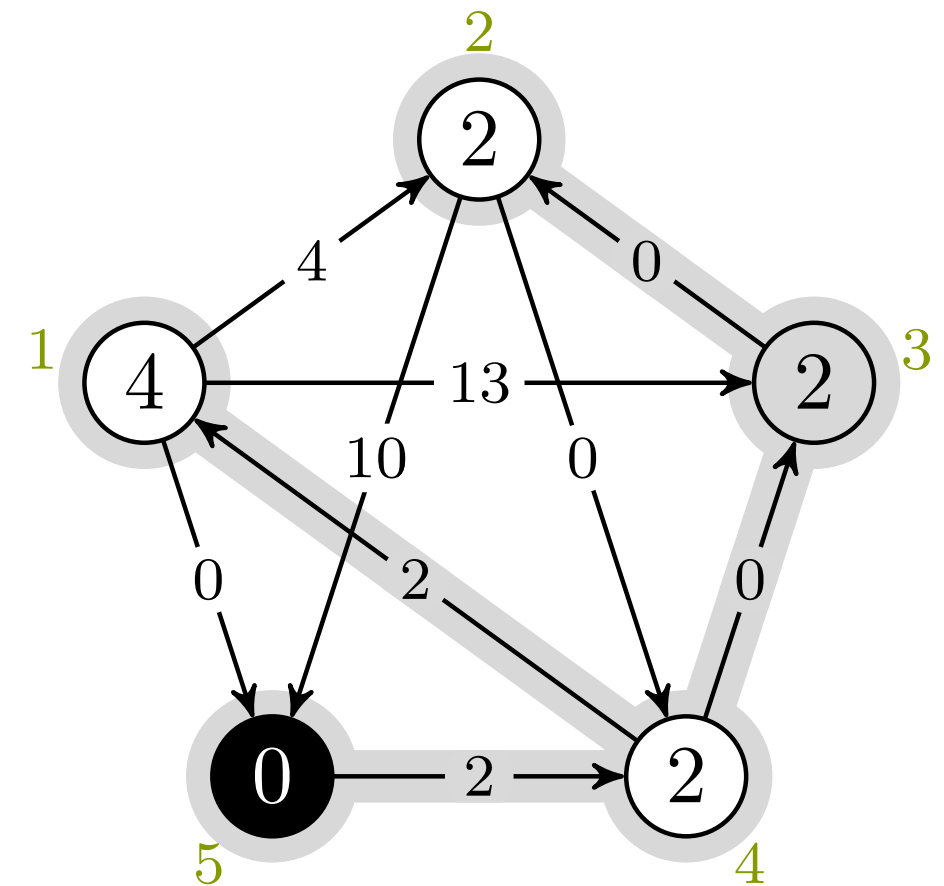


D	1	0	1	-3	2	-4	0	1
	2	3	0	-4	1	-1	-1	2
	3	7	4	0	5	3	-5	3
	4	2	-1	-5	0	-2	0	4
	5	8	5				-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

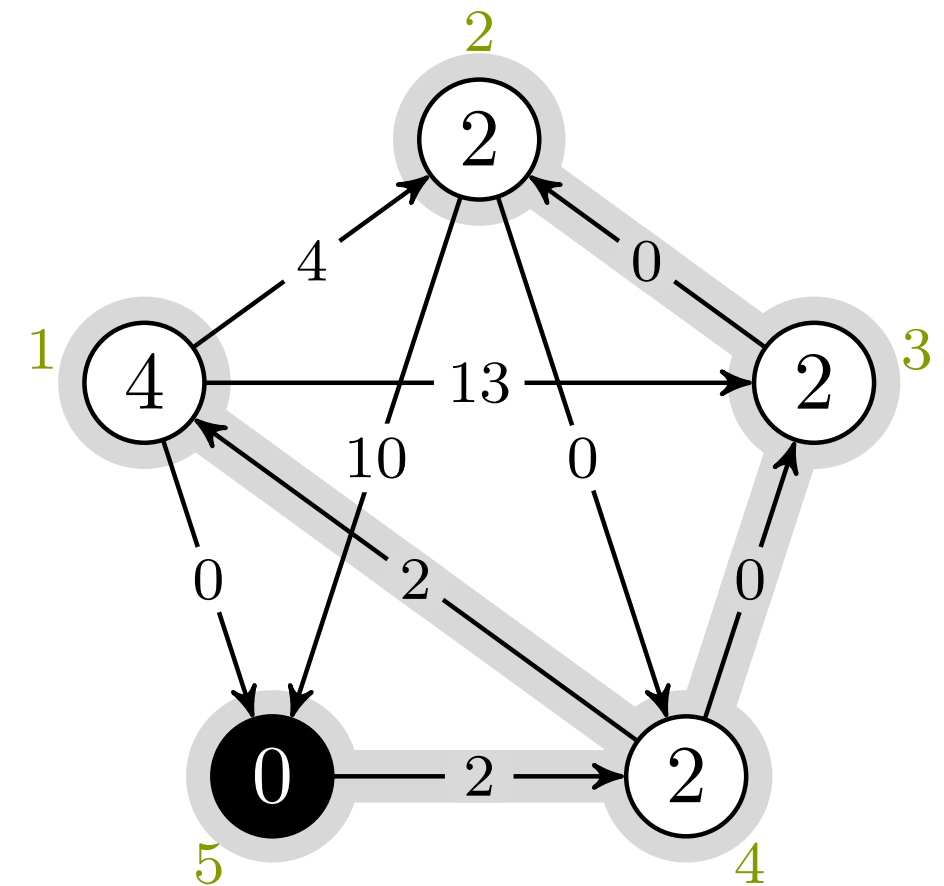


D	1	0	1	-3	2	-4	0	1
	2	3	0	-4	1	-1	-1	2
	3	7	4	0	5	3	-5	3
	4	2	-1	-5	0	-2	0	4
	5	8	5				-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10  for each vertex  $v \in G.V$ 
11       $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```



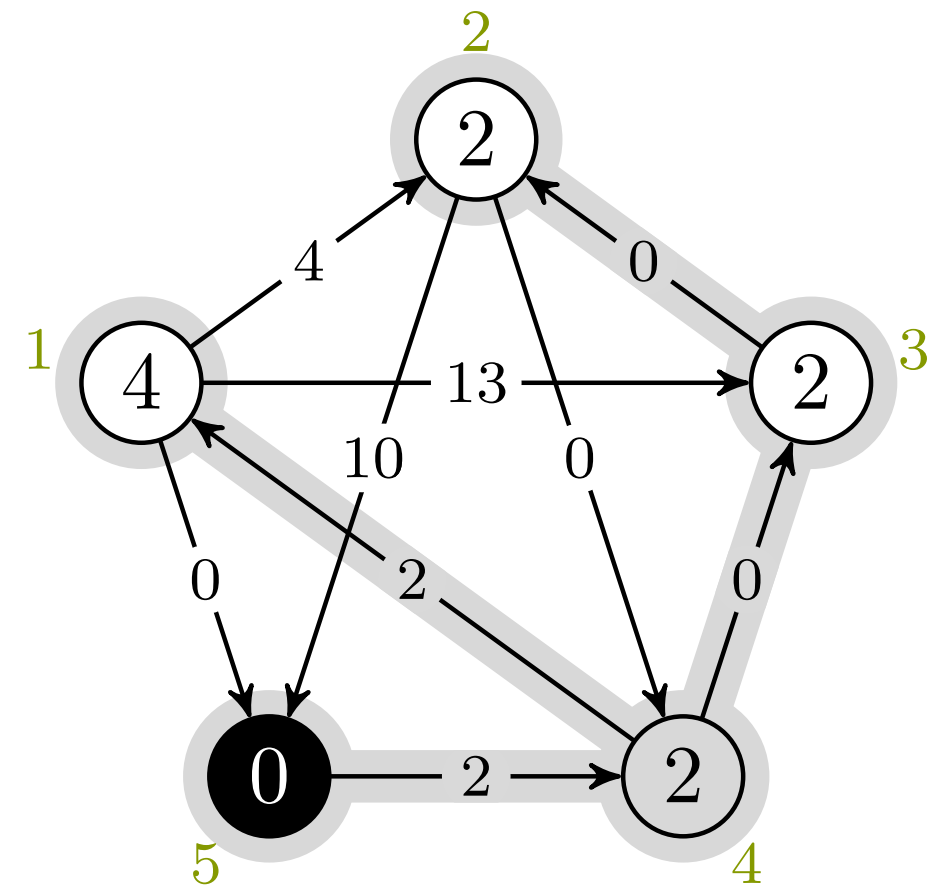
D	1	0	1	-3	2	-4	0	1
	2	3	0	-4	1	-1	-1	2
	3	7	4	0	5	3	-5	3
	4	2	-1	-5	0	-2	0	4
	5	8	5	1			-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 

```

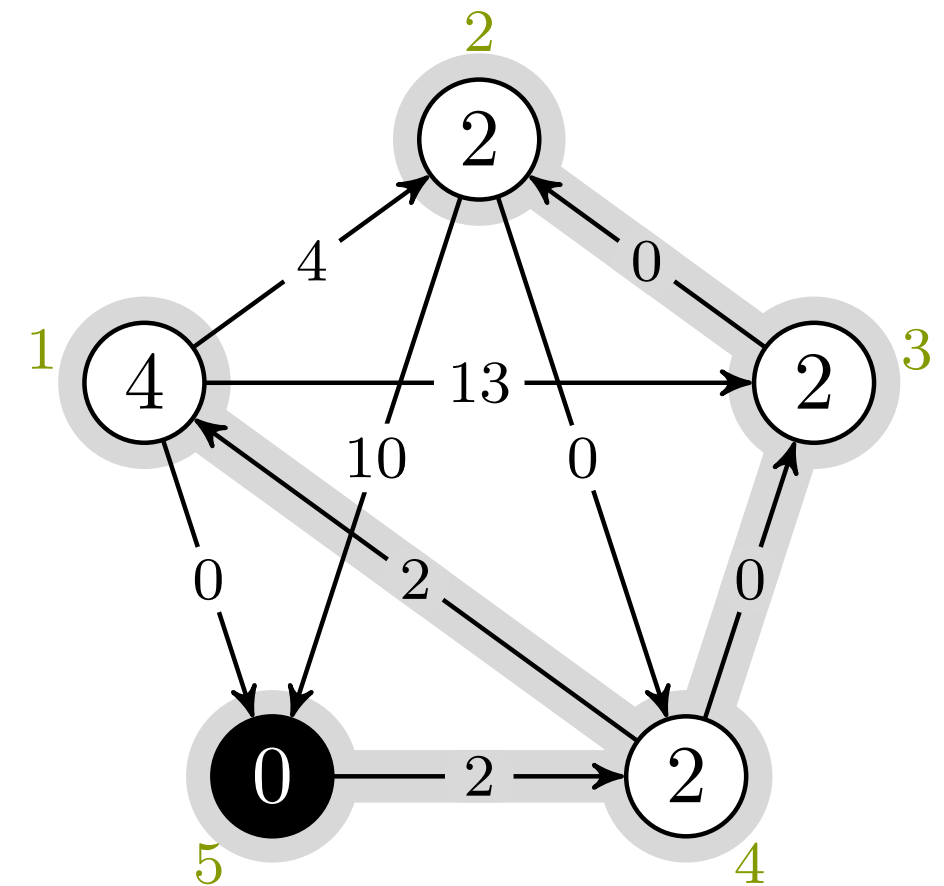


D	1	0	1	-3	2	-4	0	1
	2	3	0	-4	1	-1	-1	2
	3	7	4	0	5	3	-5	3
	4	2	-1	-5	0	-2	0	4
	5	8	5	1			-4	5
		1	2	3	4	5		

h


```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10  for each vertex  $v \in G.V$ 
11       $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```

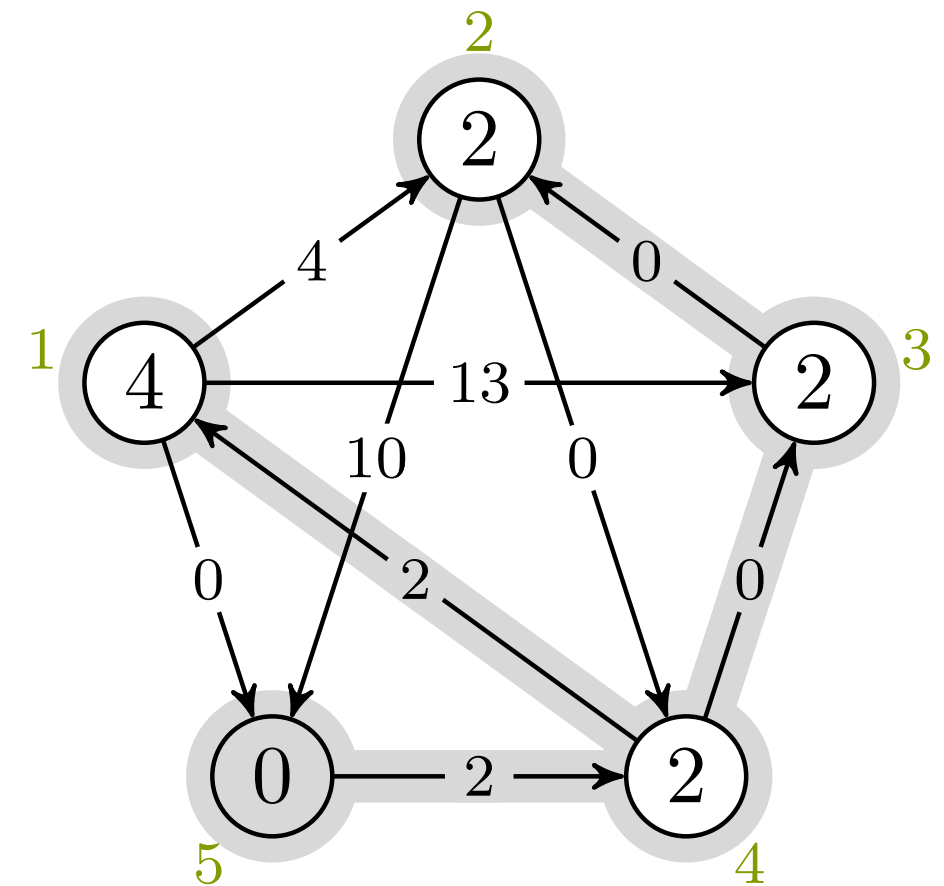


D	1	0	1	-3	2	-4	0	1
	2	3	0	-4	1	-1	-1	2
	3	7	4	0	5	3	-5	3
	4	2	-1	-5	0	-2	0	4
	5	8	5	1	6		-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 
    
```



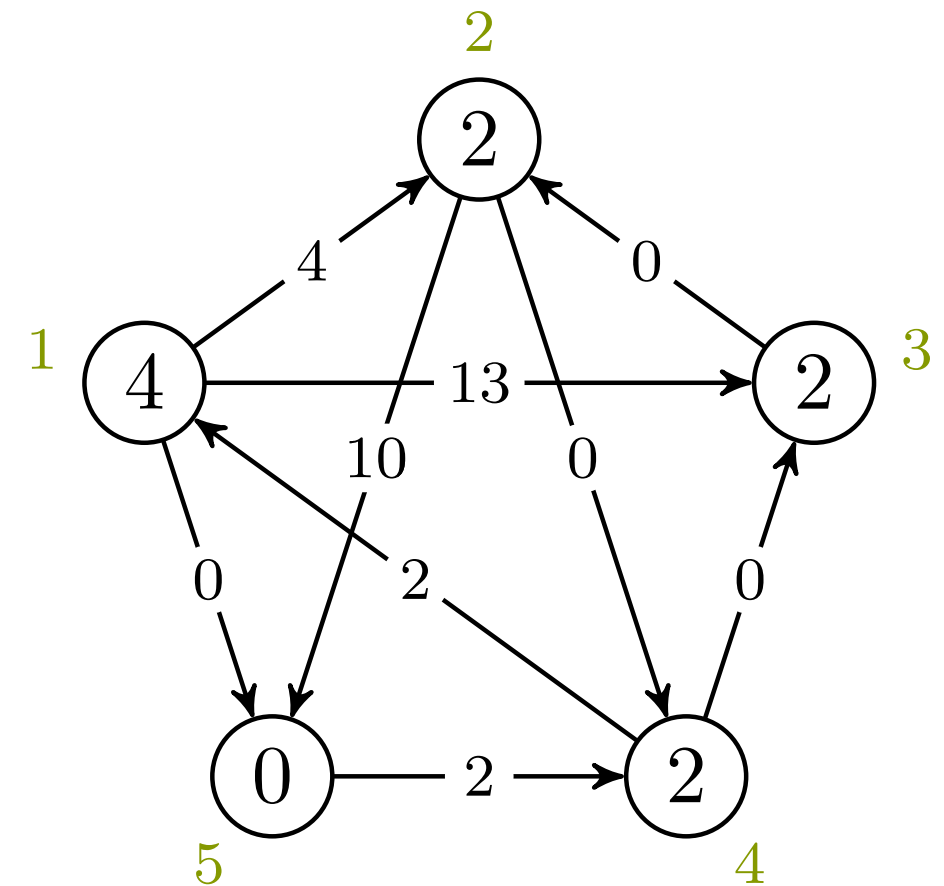
D	1	0	1	-3	2	-4	0	1
	2	3	0	-4	1	-1	-1	2
	3	7	4	0	5	3	-5	3
	4	2	-1	-5	0	-2	0	4
	5	8	5	1	6		-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12  return  $D$ 

```

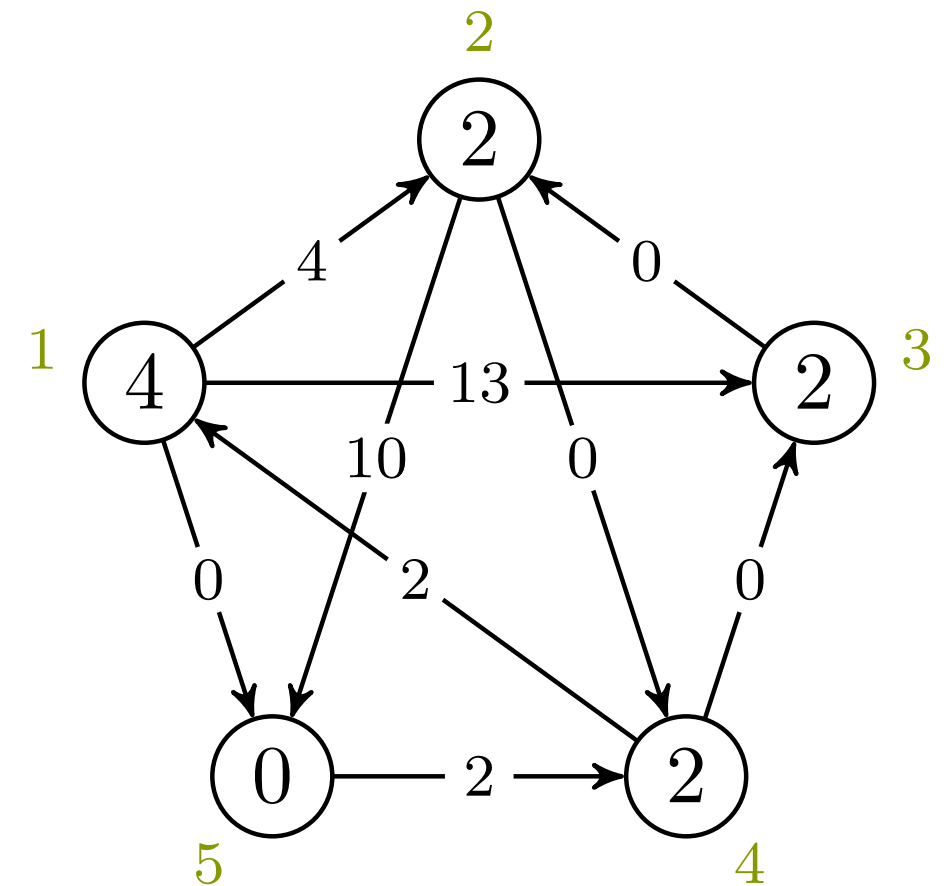


D	1	0	1	-3	2	-4	0	1
	2	3	0	-4	1	-1	-1	2
	3	7	4	0	5	3	-5	3
	4	2	-1	-5	0	-2	0	4
	5	8	5	1	6	0	-4	5
		1	2	3	4	5		

h

```

JOHNSON( $G, w$ )
1  construct  $G'$  with start node  $s$ 
2  BELLMAN-FORD( $G', w, s$ )
3  for each vertex  $v \in G.V$ 
4       $h(v) = v.d$ 
5  for each edge  $(u, v) \in G.E$ 
6       $\hat{w}(u, v) = w(u, v) + h(u) - h(v)$ 
7  let  $D = (d_{uv})$  be a new  $n \times n$  matrix
8  for each vertex  $u \in G.V$ 
9      DIJKSTRA( $G, \hat{w}, u$ )
10     for each vertex  $v \in G.V$ 
11          $d_{uv} = v.d + h(v) - h(u)$ 
12 return  $D$ 
    
```



D	1	0	1	-3	2	-4	0	1
	2	3	0	-4	1	-1	-1	2
	3	7	4	0	5	3	-5	3
	4	2	-1	-5	0	-2	0	4
	5	8	5	1	6	0	-4	5
		1	2	3	4	5		

h

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 $\mathbf{v}_k(a_{ik} \wedge b_{kj})$. 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:

$$M' = \mathbf{v}_{i=1}^d M^i \text{ where we define } M^1 = M \text{ and } M^{i+1}$$

The convenience of describing the products has appeared many times.

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

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

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 .

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?

Traversér fra hver node?

› Kjøretid: $V \times \Theta(E + V) = \Theta(VE + V^2)$

Traversér fra hver node?

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

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

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:

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 $E = \Theta(V^2)$

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

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

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...

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 j

Det finnes en vei fra i til $j \dots$

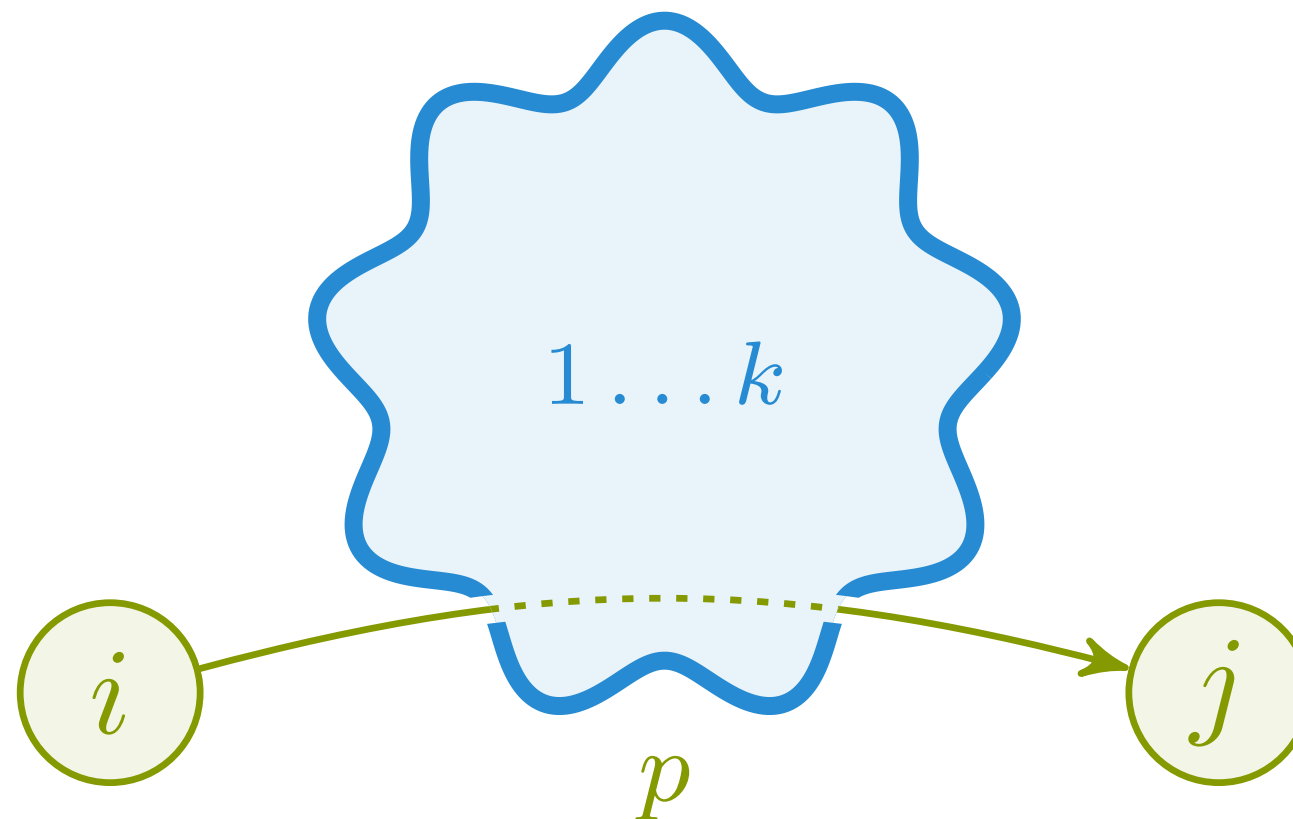
ijk

ijk

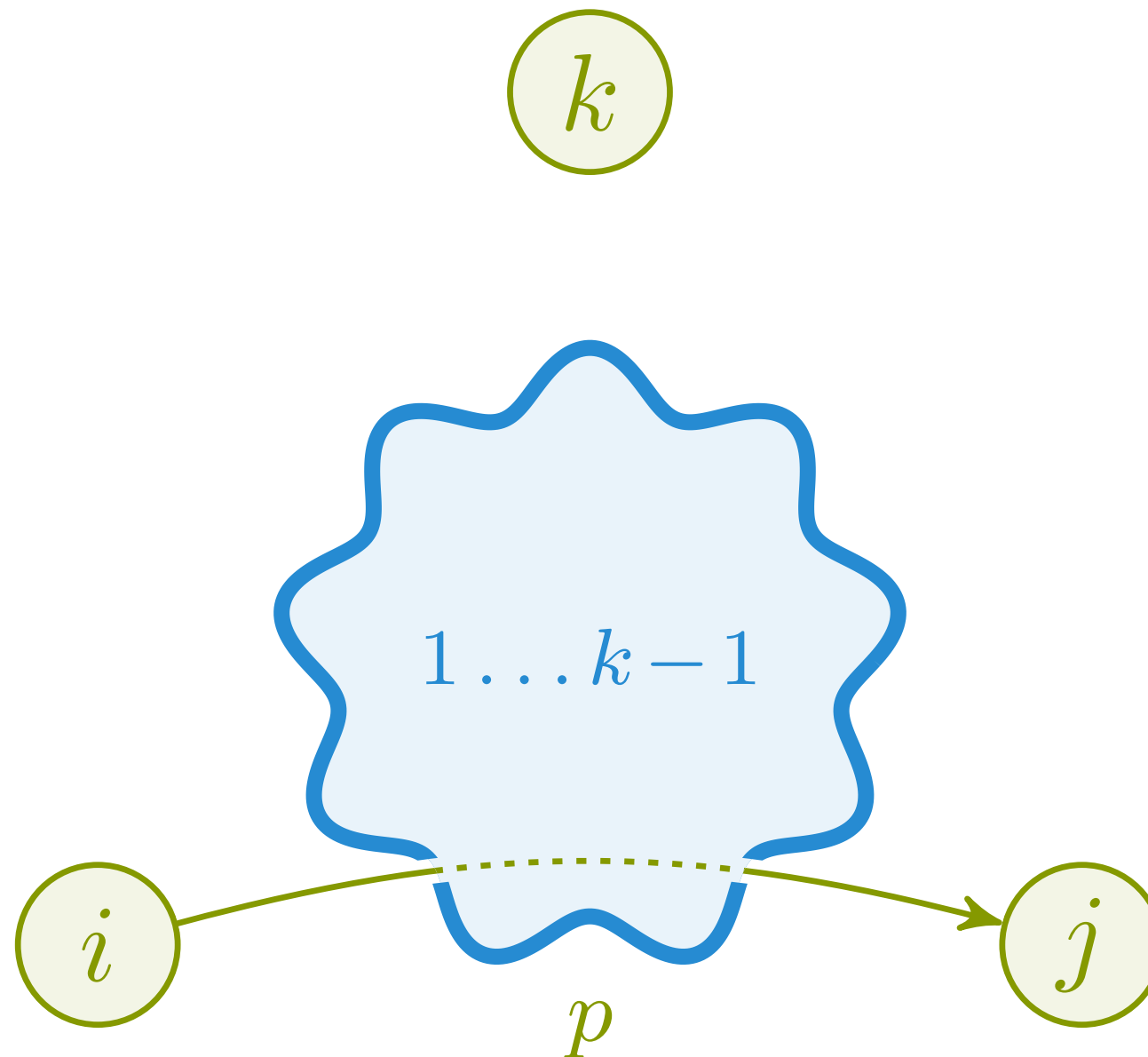
Det finnes en vei fra i til j via noder fra $\{1 \dots k\}$

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

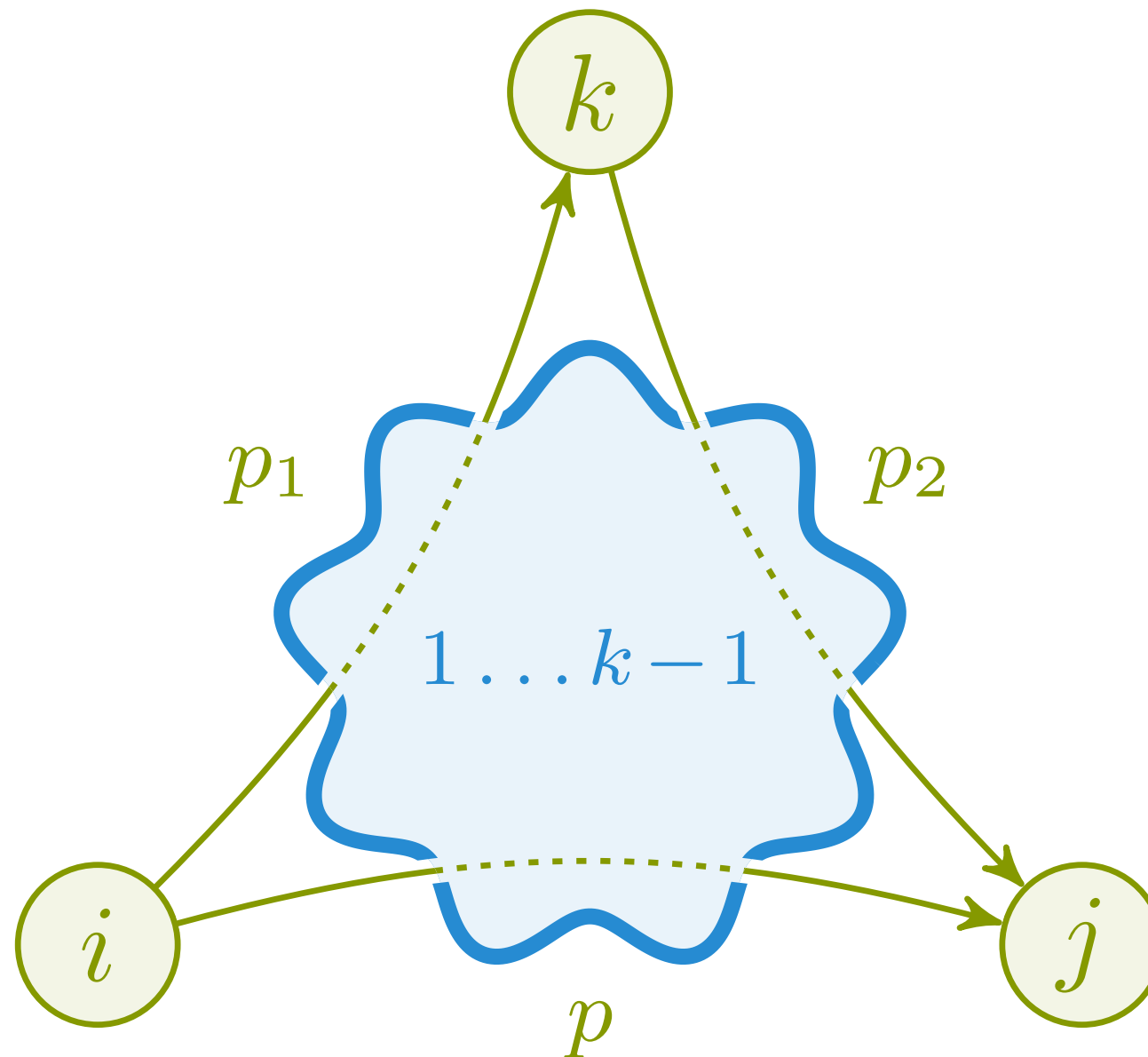
Det finnes en vei fra i til j via noder fra $\{1 \dots k\}$



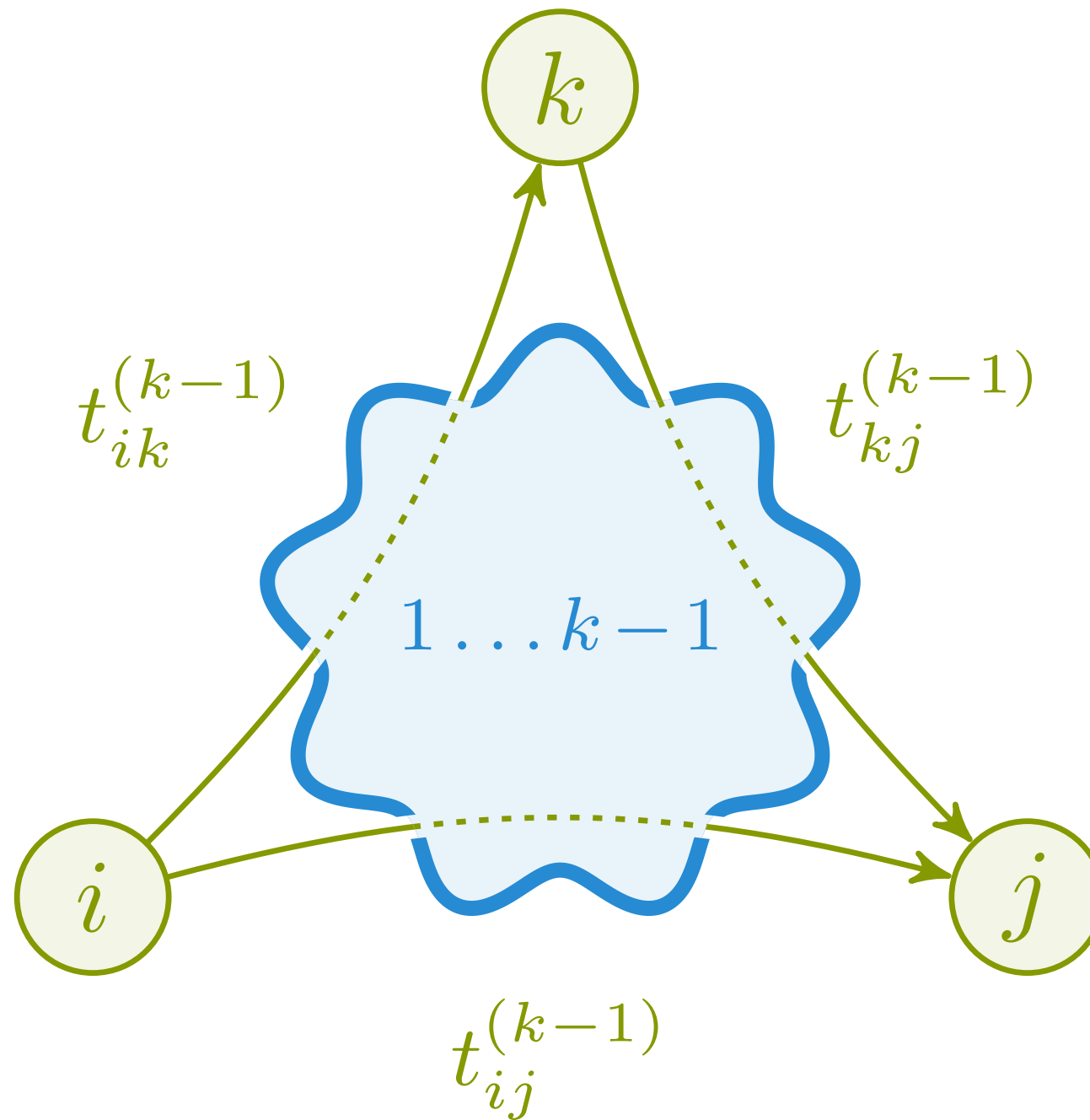
$t_{ij}^{(k)}$ = det går en vei fra i til j via noder fra $\{1 \dots k\}$

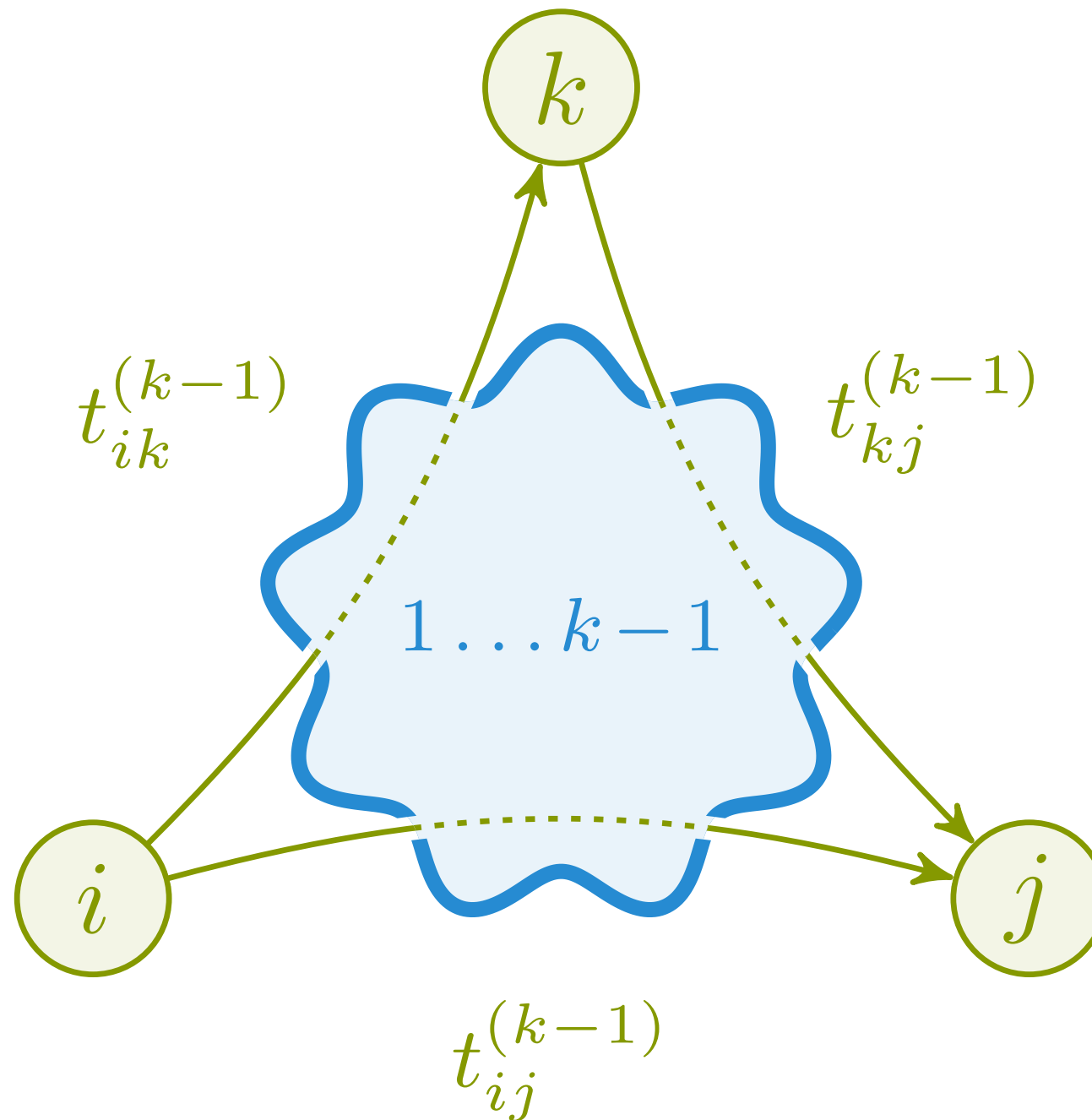


Som for ryggsekkproblemet: Skal k være med eller ikke?



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





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

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

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

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

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

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

$t_{ij}^{(k)}$ $i \overset{1 \dots k}{\rightsquigarrow} j$?

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

TRANSITIVE-CLOSURE(G)

1 $n = |G.V|$

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

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

G graf

n ant. noder

$t_{ij}^{(k)}$ $i \xrightarrow{1 \dots k} j$?

For hver mulig startnode...

TRANSITIVE-CLOSURE(G)

1 $n = |G.V|$

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

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

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

G graf

n ant. noder

$t_{ij}^{(k)}$ $i \xrightarrow{1 \dots k} j$?

For hver mulig sluttnode...

TRANSITIVE-CLOSURE(G)

```

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

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

Samme node eller sammenkoblet med kant?

TRANSITIVE-CLOSURE(G)

```

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

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

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

TRANSITIVE-CLOSURE(G)

```

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

```

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

Ellers finnes ingen slik sti

TRANSITIVE-CLOSURE(G)

```

1   $n = |G.V|$ 
2  let  $T^{(0)} = (t_{ij}^{(0)})$  be a new  $n \times n$  matrix
3  for  $i = 1$  to  $n$ 
4      for  $j = 1$  to  $n$ 
5          if  $i == j$  or  $(i, j) \in G.E$ 
6               $t_{ij}^{(0)} = 1$ 
7          else  $t_{ij}^{(0)} = 0$ 
8  ...

```

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

Grunntilfelle på plass

TRANSITIVE-CLOSURE(G)

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

TRANSITIVE-CLOSURE(G)

7 ...

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

TRANSITIVE-CLOSURE(G)

7 ...

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

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

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 $n \times n$ matrix

G graf

n ant. noder

$t_{ij}^{(k)}$ $i \stackrel{1 \dots k}{\rightsquigarrow} j$?

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

```

TRANSITIVE-CLOSURE(G)
7  ...
8  for  $k = 1$  to  $n$ 
9      let  $T^{(k)} = (t_{ij}^{(k)})$  be a new  $n \times n$  matrix
10     for  $i = 1$  to  $n$ 

```

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

For hver mulig startnode...

```

TRANSITIVE-CLOSURE(G)
7  ...
8  for  $k = 1$  to  $n$ 
9      let  $T^{(k)} = (t_{ij}^{(k)})$  be a new  $n \times n$  matrix
10     for  $i = 1$  to  $n$ 
11         for  $j = 1$  to  $n$ 
    
```

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

For hver mulig sluttnode...

TRANSITIVE-CLOSURE(G)

```

7  ...
8  for  $k = 1$  to  $n$ 
9      let  $T^{(k)} = (t_{ij}^{(k)})$  be a new  $n \times n$  matrix
10     for  $i = 1$  to  $n$ 
11         for  $j = 1$  to  $n$ 
12              $t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$ 

```

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

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


```

TRANSITIVE-CLOSURE(G)
7  ...
8  for  $k = 1$  to  $n$ 
9      let  $T^{(k)} = (t_{ij}^{(k)})$  be a new  $n \times n$  matrix
10     for  $i = 1$  to  $n$ 
11         for  $j = 1$  to  $n$ 
12              $t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$ 
13 return  $T^{(n)}$ 

```

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

Finnes det en sti $i \rightsquigarrow j$ som får gå innom $\{1, \dots, n\}$, 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   $n = |G.V|$   
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)

1 $n = |G.V|$

2 initialize T

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

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

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

For hver mulig sluttnode...

TRANSITIVE-CLOSURE'(G)

1 $n = |G.V|$

2 initialize T

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

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

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

6 $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$

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

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

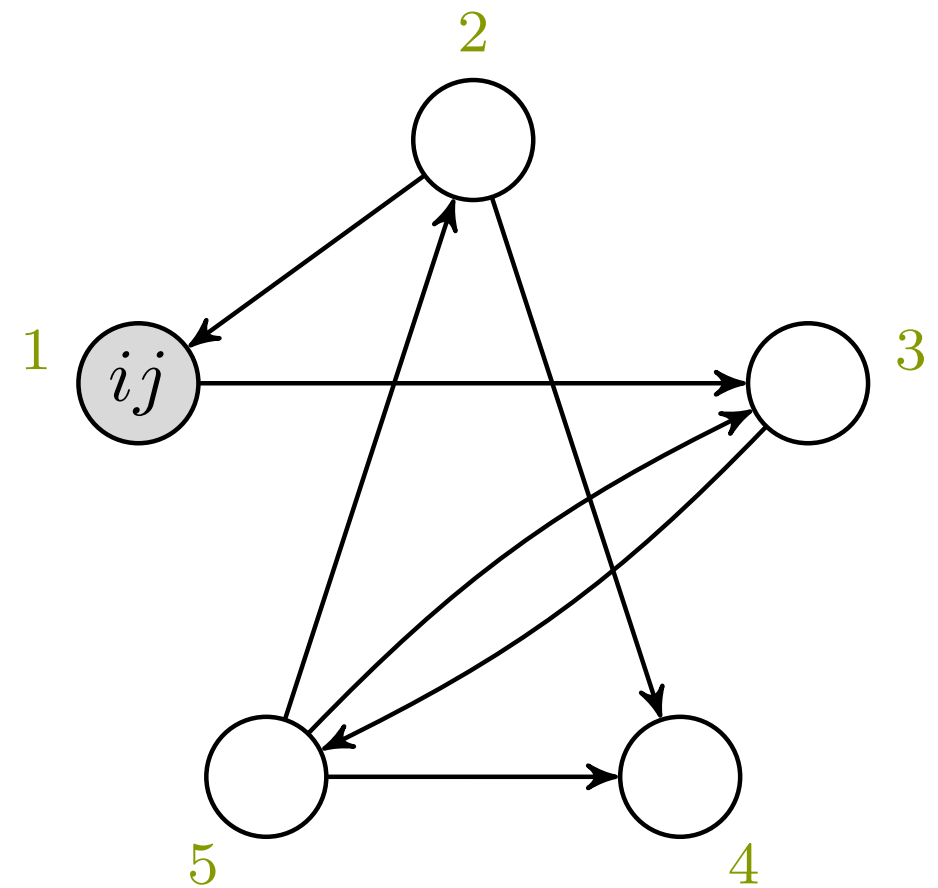
Finnes det en sti $i \rightsquigarrow j$?

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 1, 1, 1$



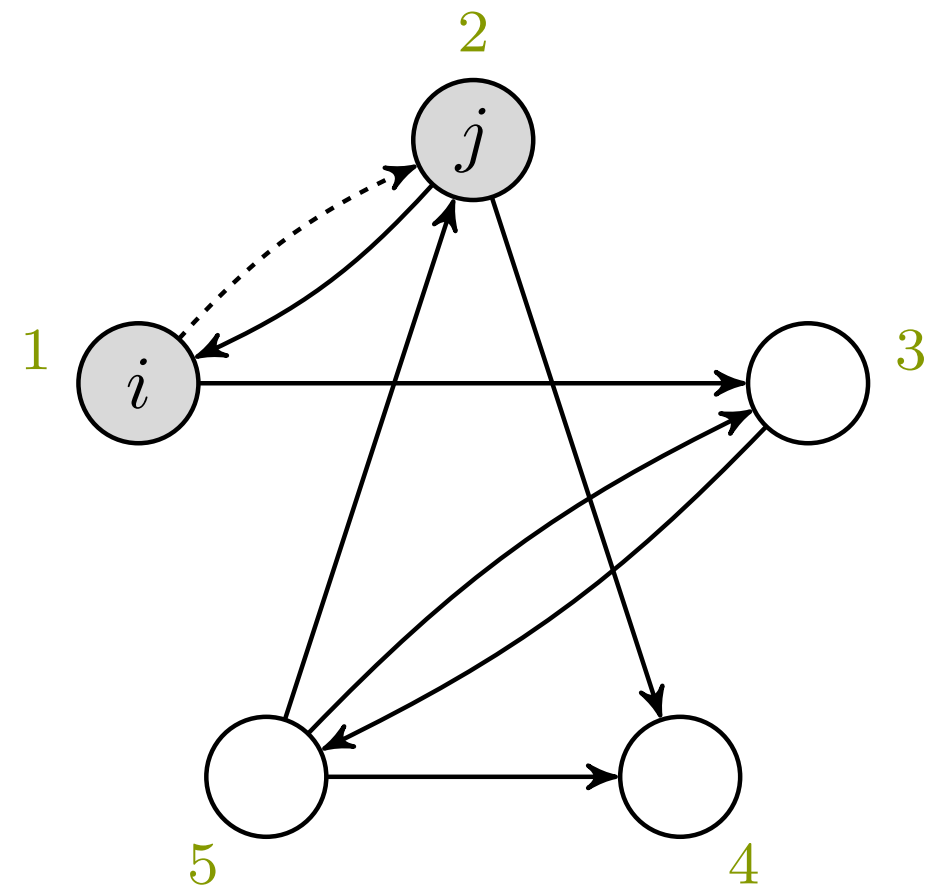
1	1		1		
2	1	1		1	
3			1		1
4				1	
5		1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 1, 1, 2$



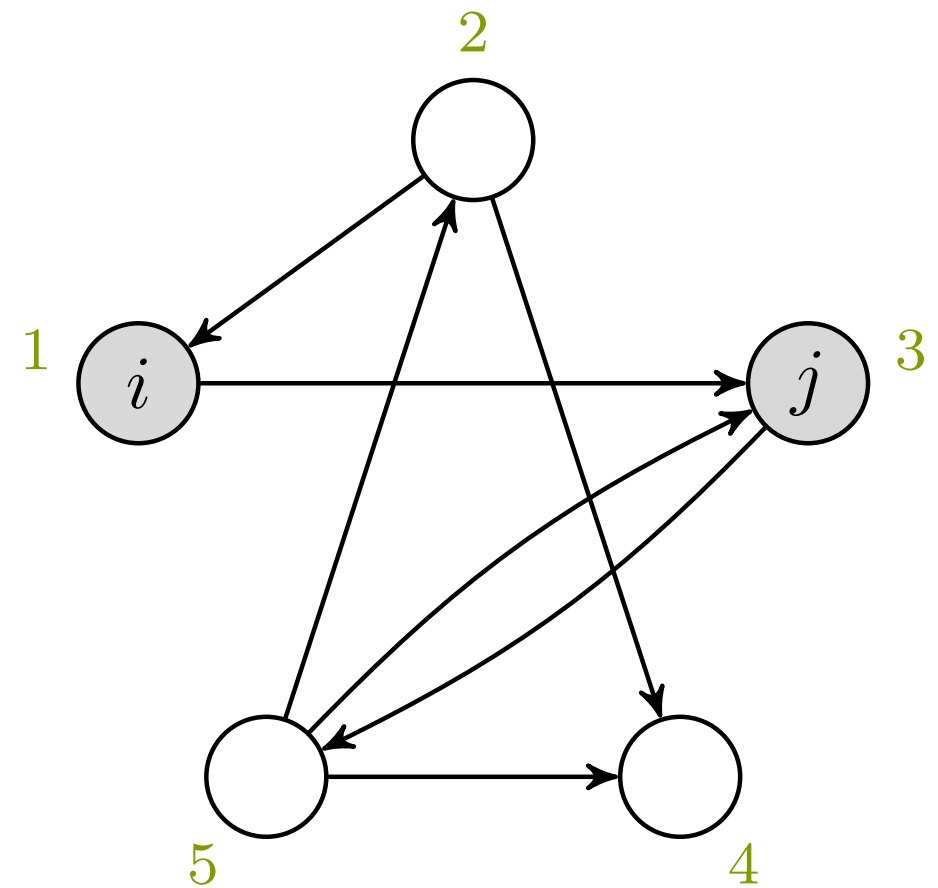
1	1		1		
2	1	1		1	
3			1		1
4				1	
5		1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 1, 1, 3$



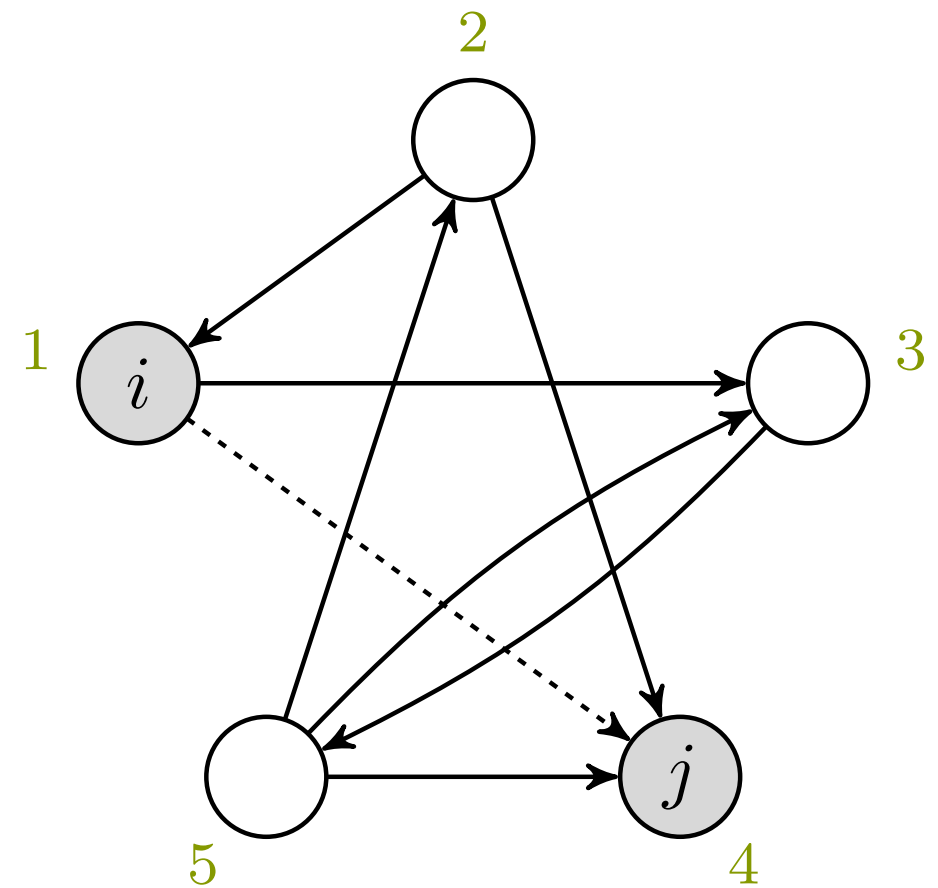
1	1		1		
2	1	1		1	
3			1		1
4				1	
5		1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 1, 1, 4$



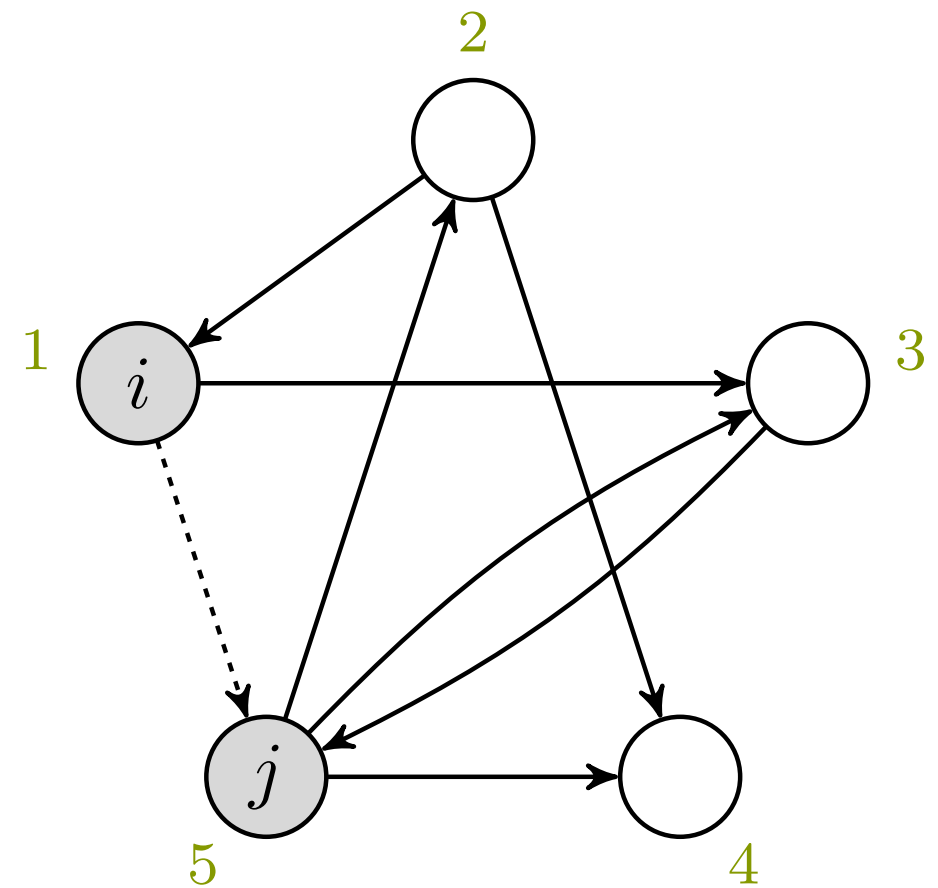
1	1		1		
2	1	1		1	
3			1		1
4				1	
5		1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 1, 1, 5$



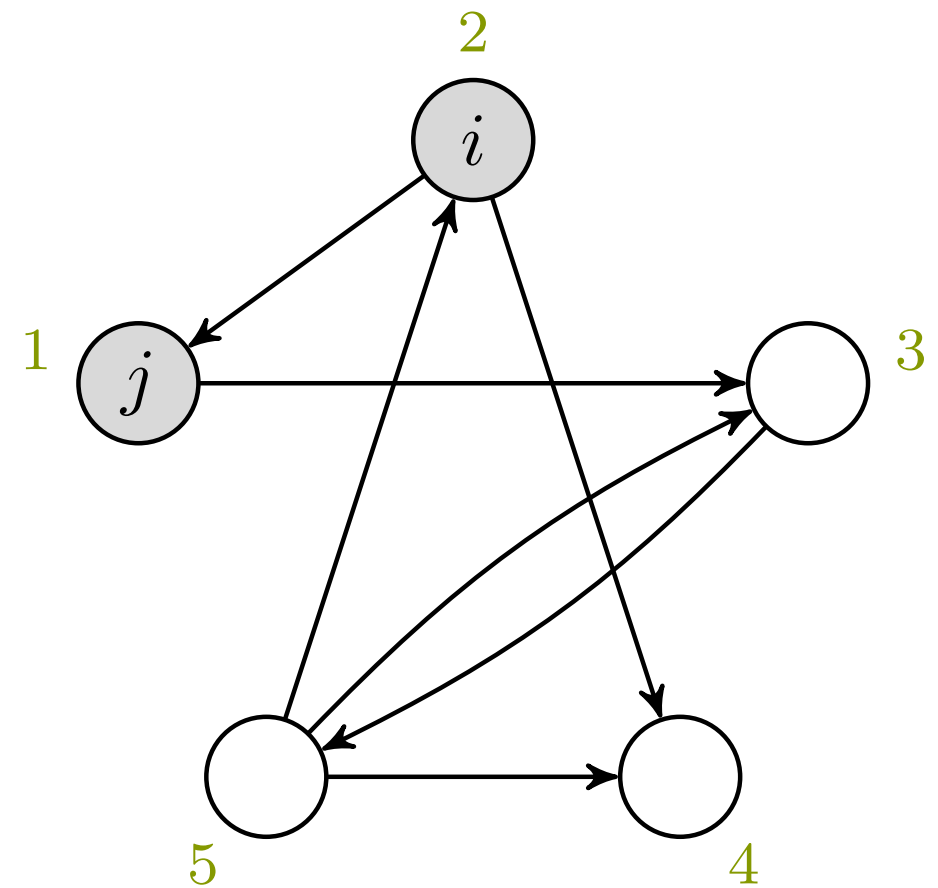
1	1		1		
2	1	1		1	
3			1		1
4				1	
5		1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 1, 2, 1$



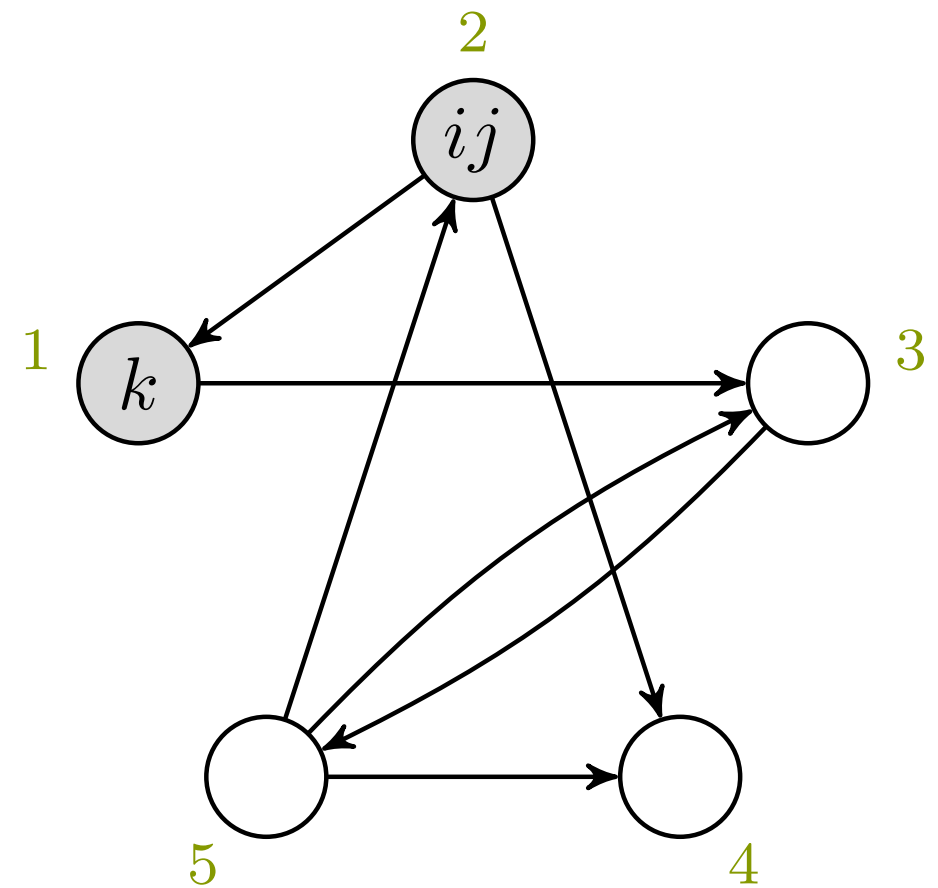
1	1		1		
2	1	1		1	
3			1		1
4				1	
5		1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 1, 2, 2$



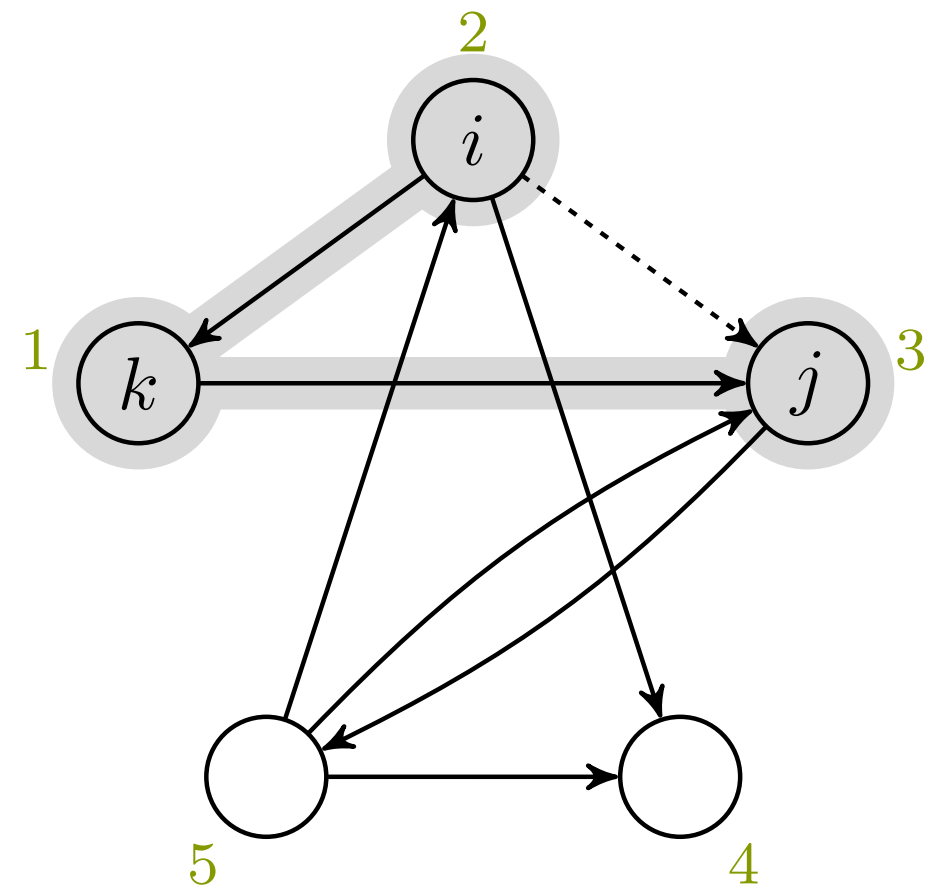
1	1		1		
2	1	1		1	
3			1		1
4				1	
5		1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 1, 2, 3$



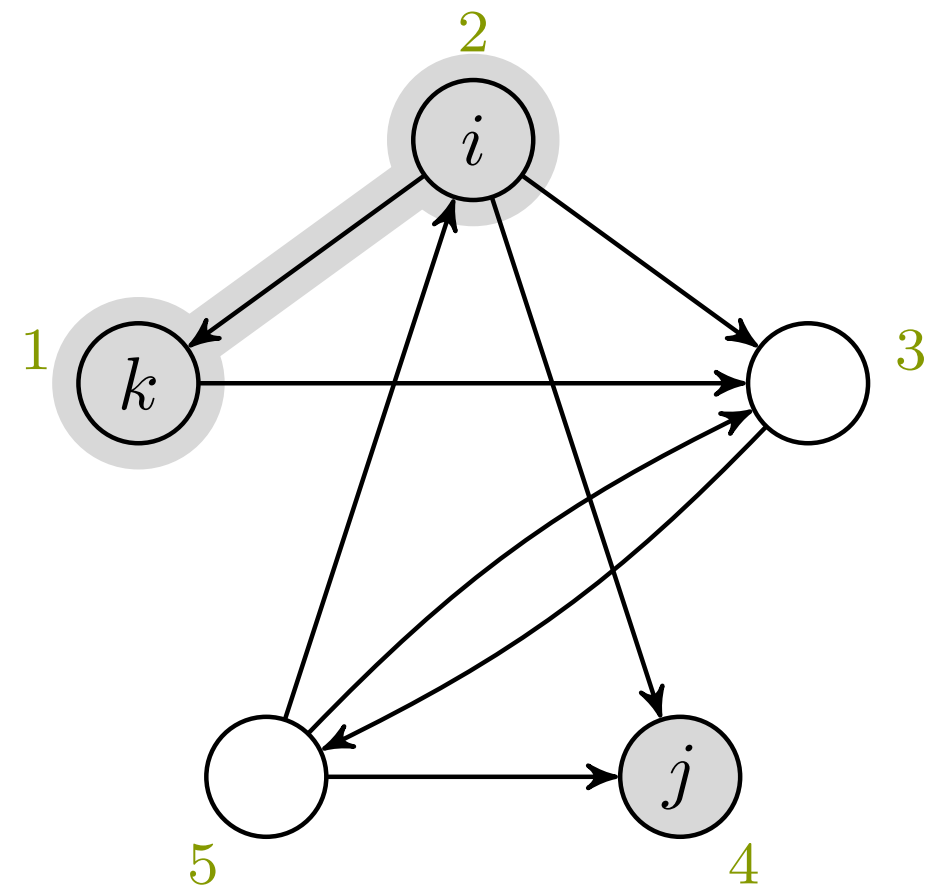
1	1		1		
2	1	1		1	
3			1		1
4				1	
5		1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 1, 2, 4$



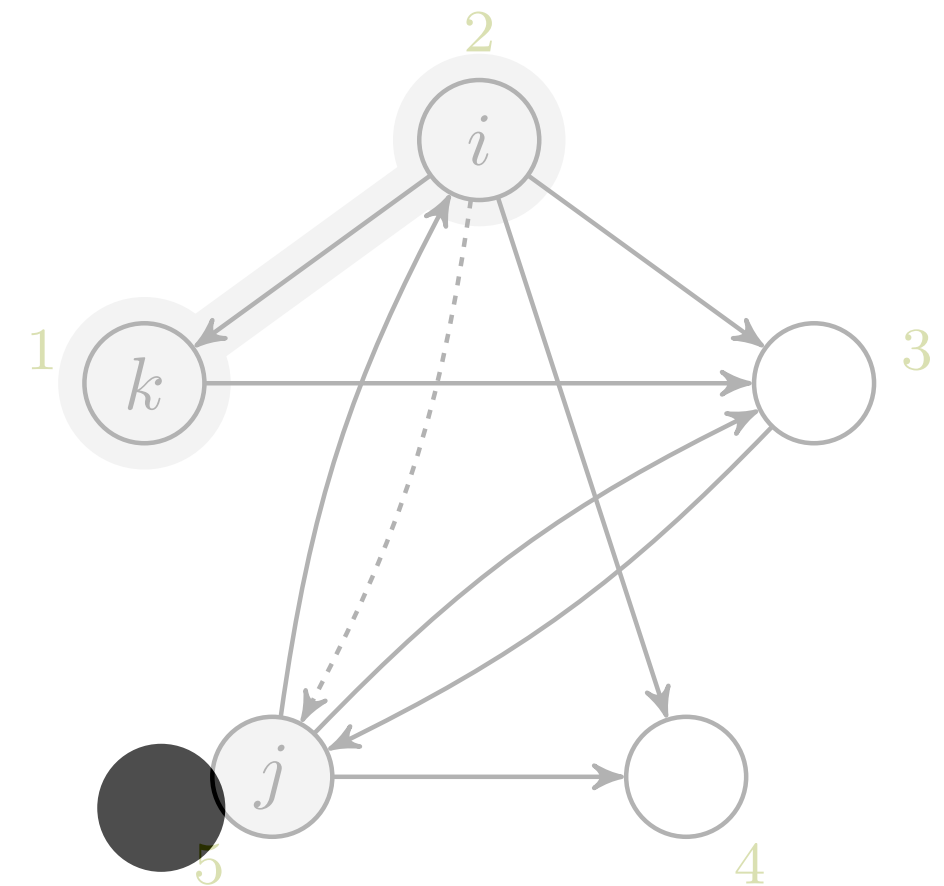
1	1		1		
2	1	1	1		
3			1		1
4				1	
5		1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 1, 2, 5$



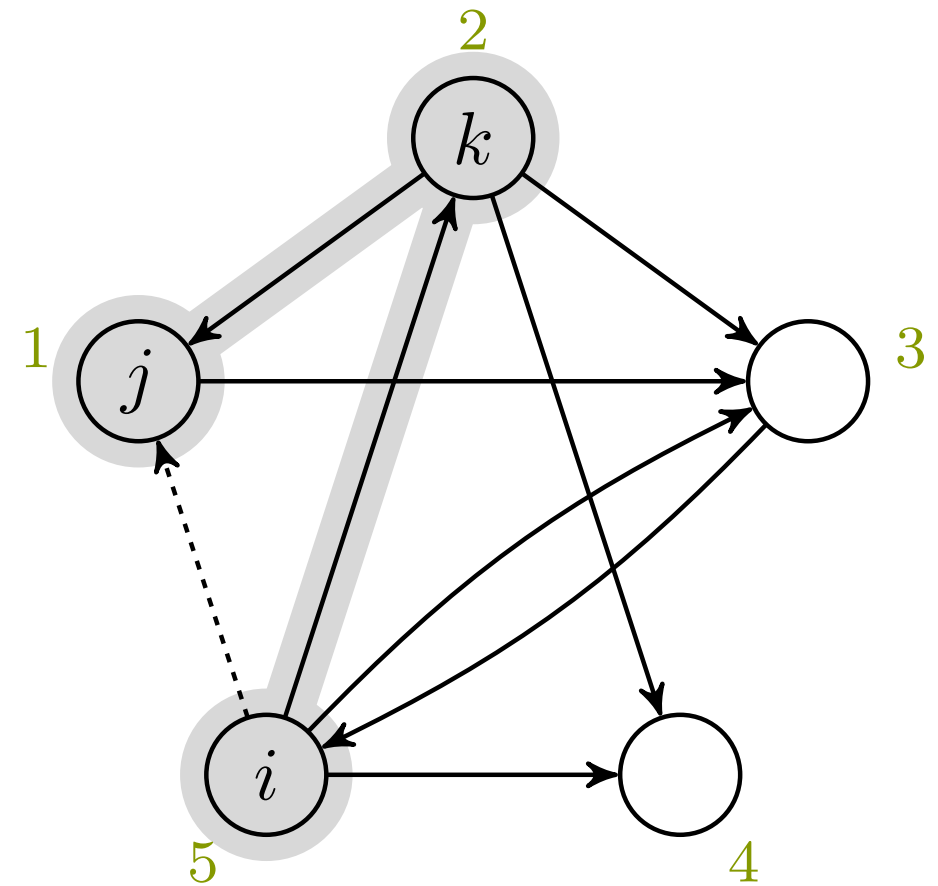
1	1		1		
2	1	1	1	1	
3			1		1
4				1	
5		1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 2, 5, 1$



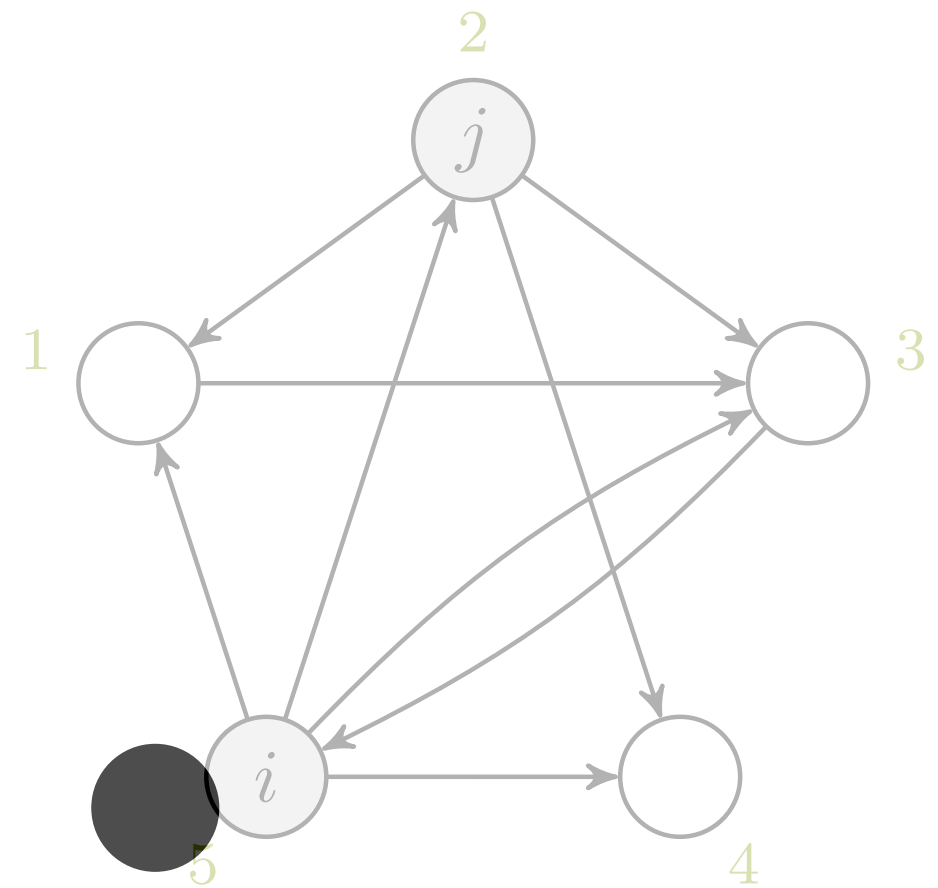
1	1		1		
2	1	1	1	1	
3			1		1
4				1	
5		1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 2, 5, 2$



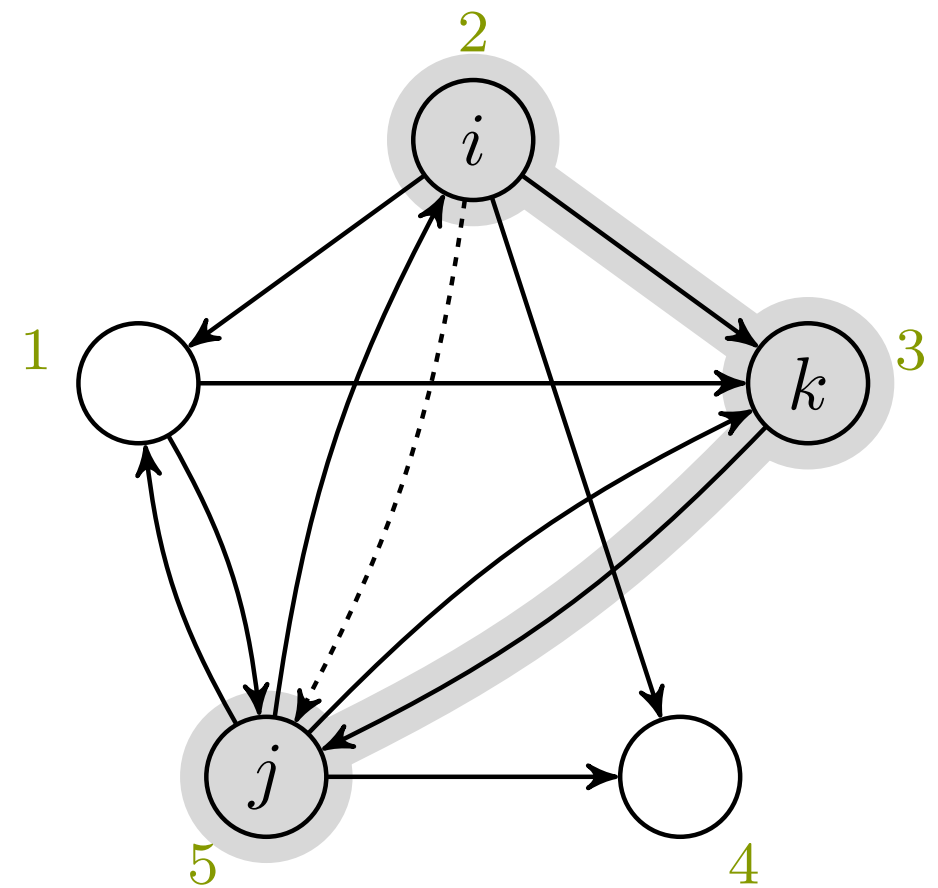
1	1		1		
2	1	1	1	1	
3			1		1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 3, 2, 5$



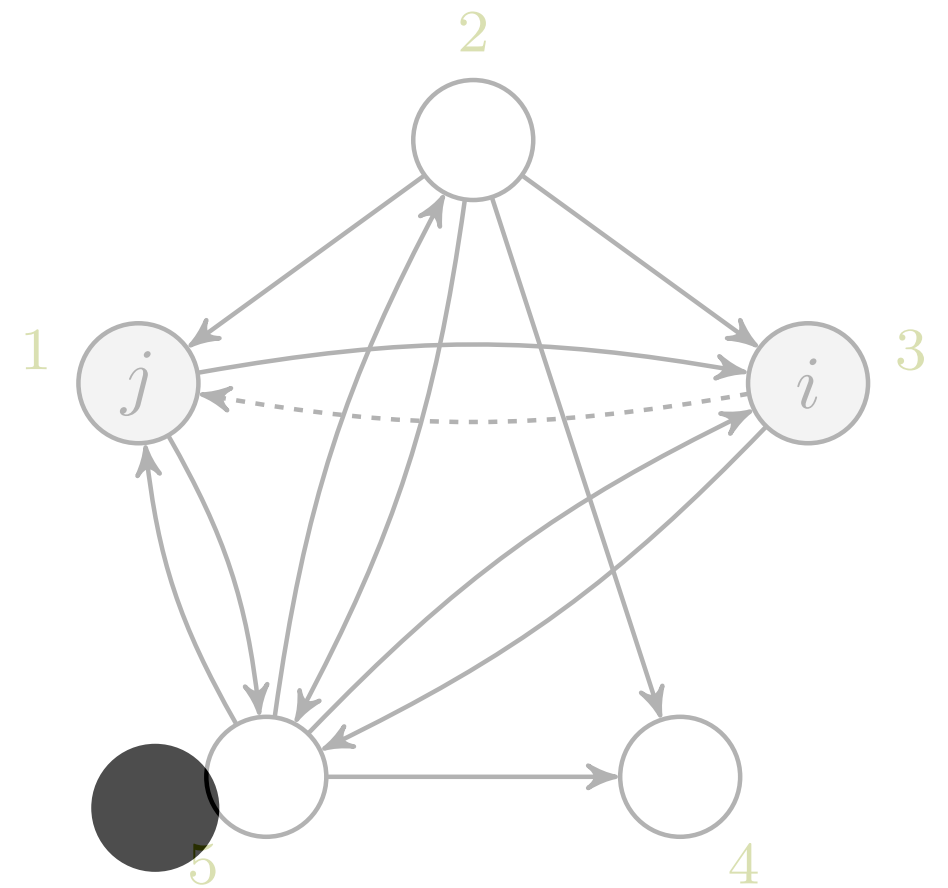
1	1		1		1
2	1	1	1		
3			1		1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 3, 3, 1$



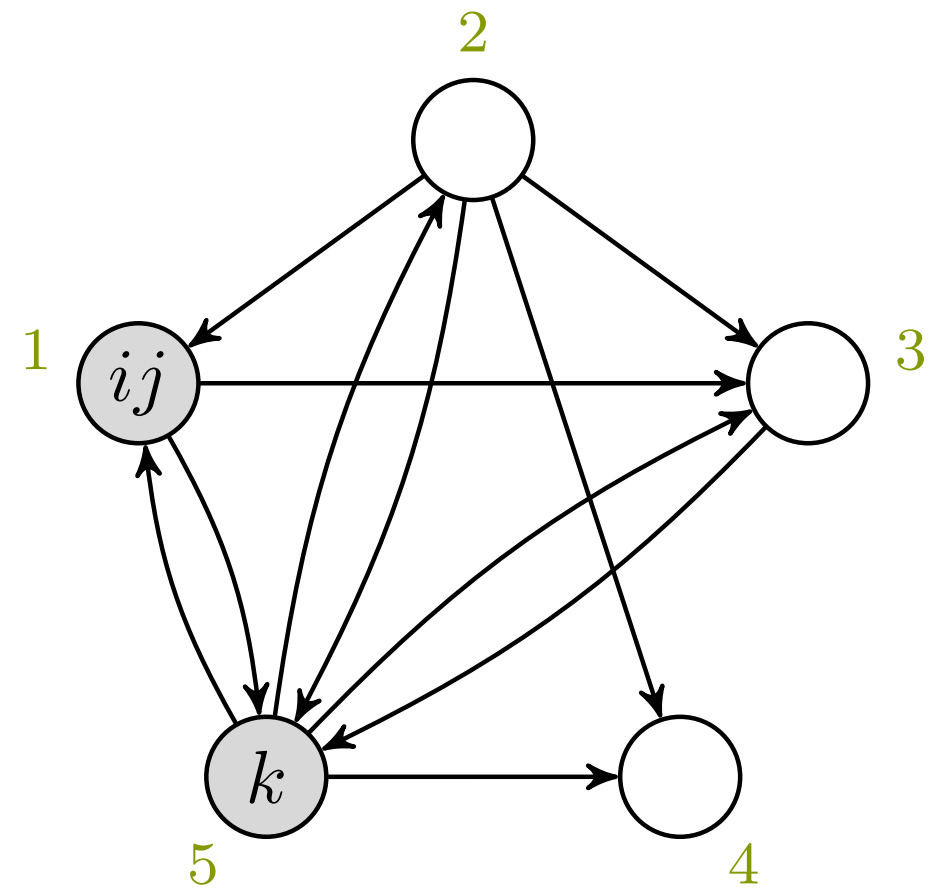
1	1		1		1
2	1	1	1	1	1
3			1		1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 1, 1$



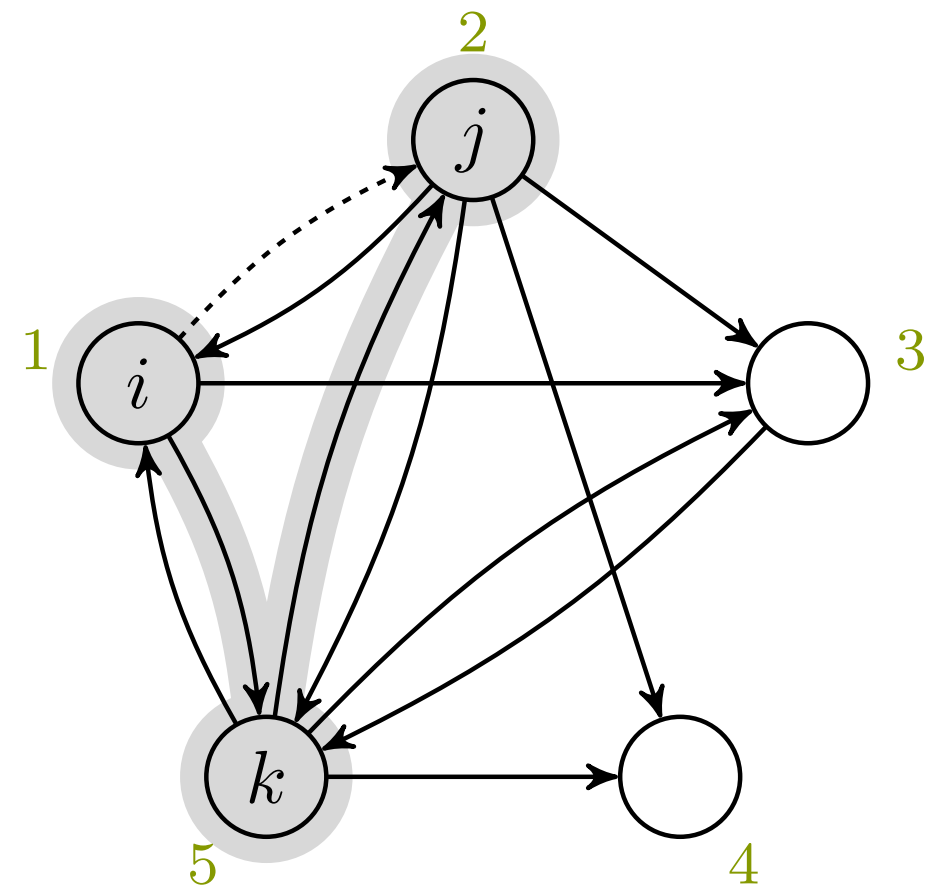
1	1		1		1
2	1	1	1	1	1
3			1		1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 1, 2$



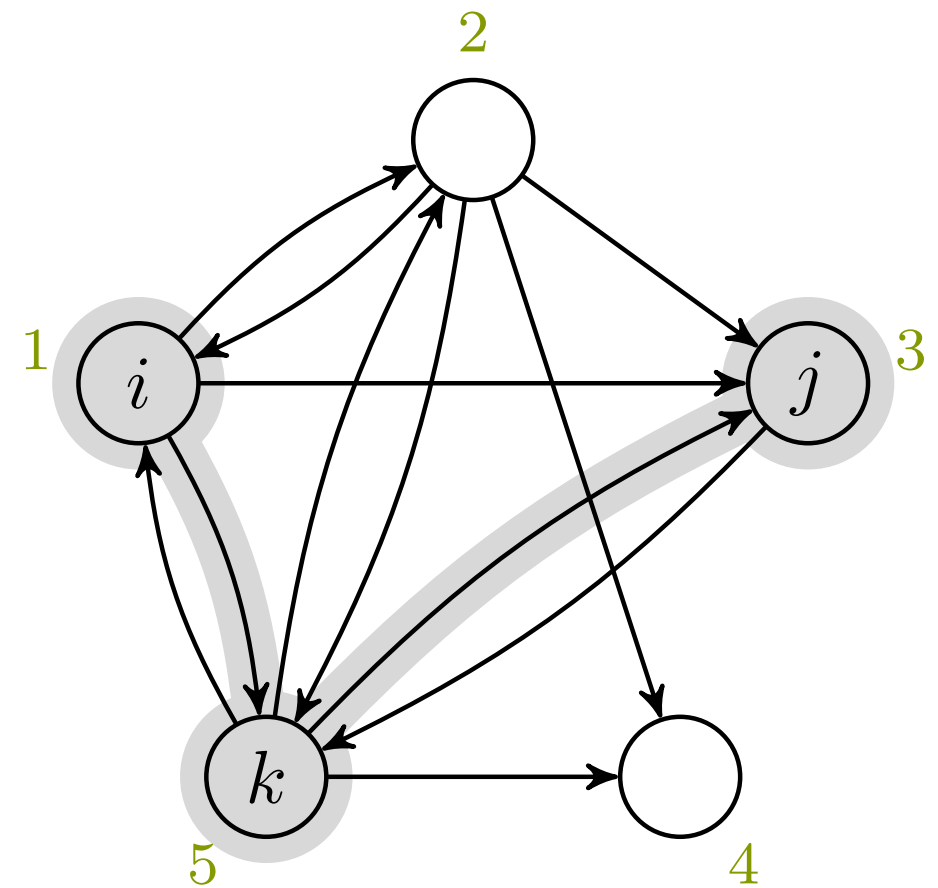
1	1		1		1
2	1	1	1	1	1
3			1		1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 1, 3$



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

korteste vei \succ transitiv lukning

TRANSITIVE-CLOSURE'(G)

$$1 \quad n = |\text{G.V}|$$

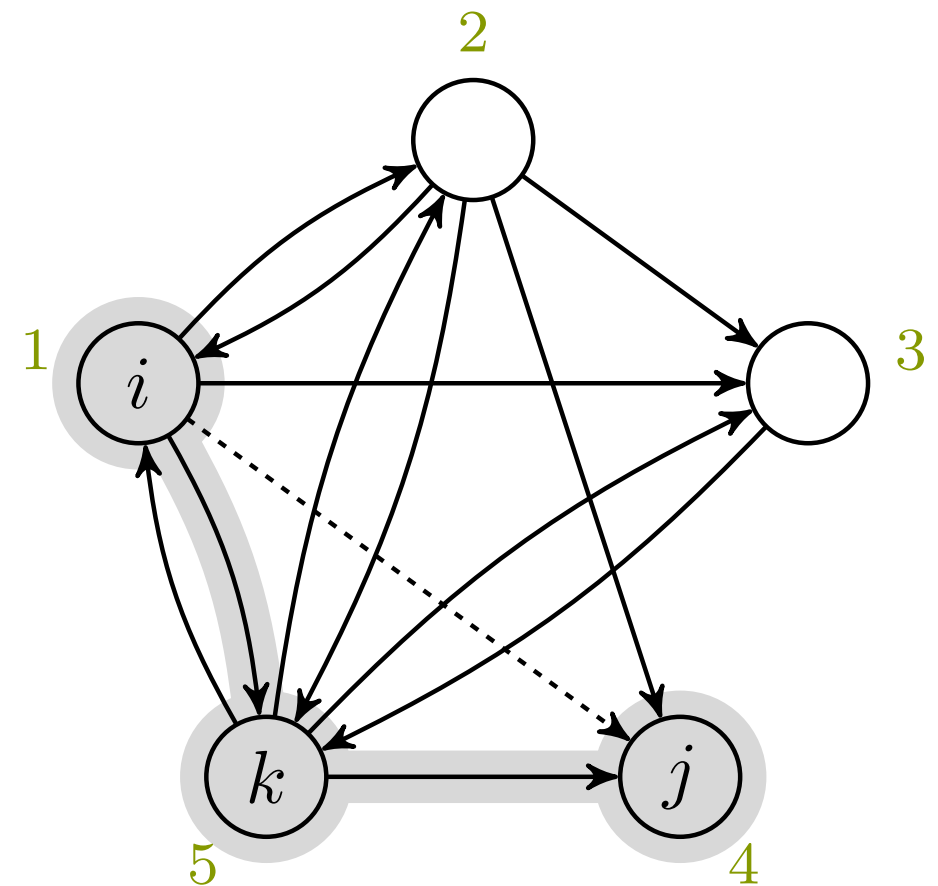
```
2  initialize T
```

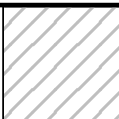

3 **for** $k = 1$ **to** n 4 **for** $i = 1$ **to** n 5 **for** $j = 1$ **to** n
$$6 \quad t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$$

```

7 return T

```



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

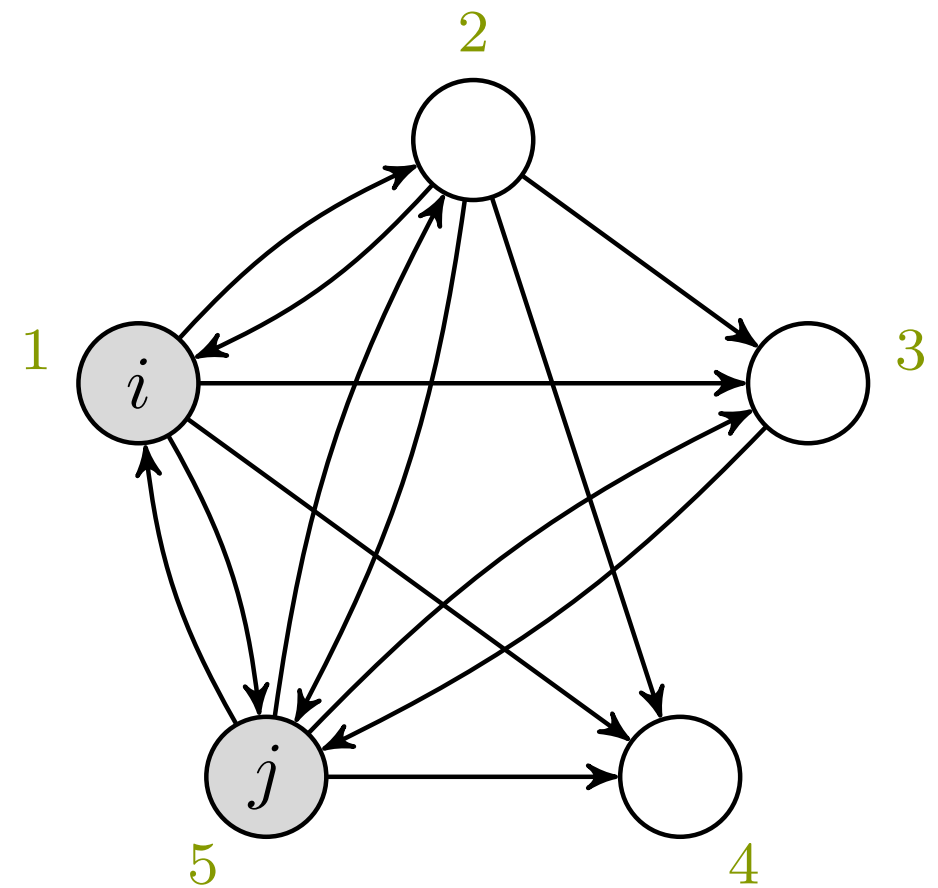
$$k, i, j = 5, 1, 4$$

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 1, 5$



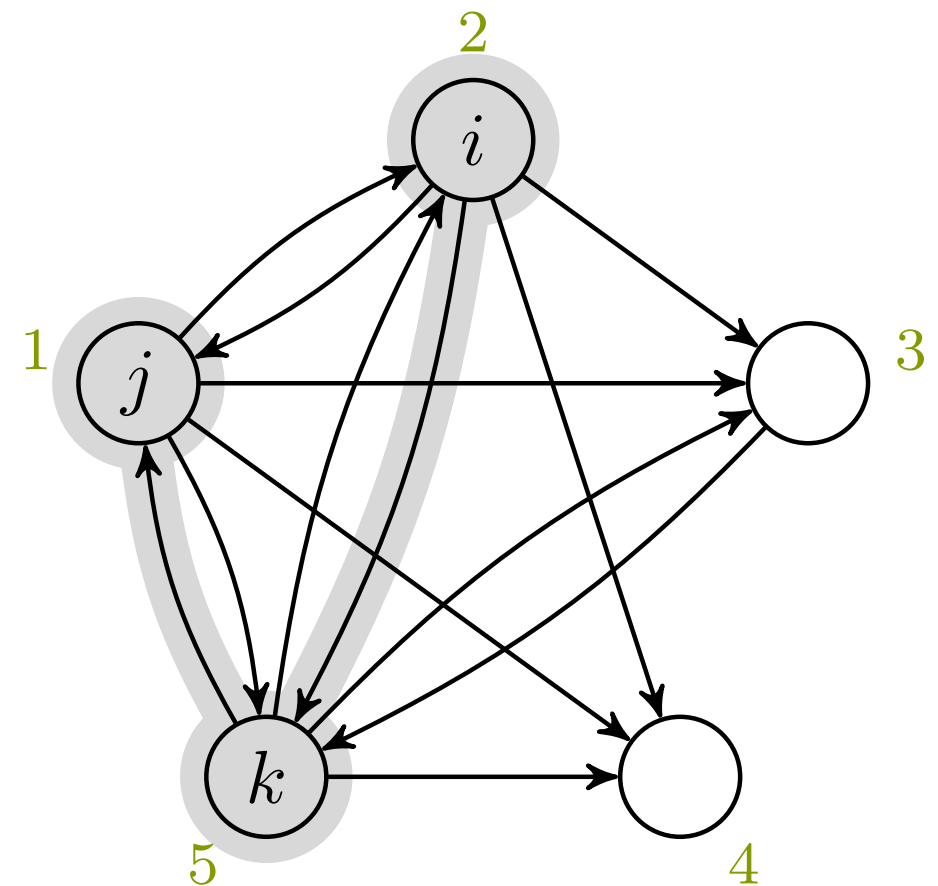
1	1	1	1	1	1
2	1	1	1	1	1
3			1		1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 2, 1$



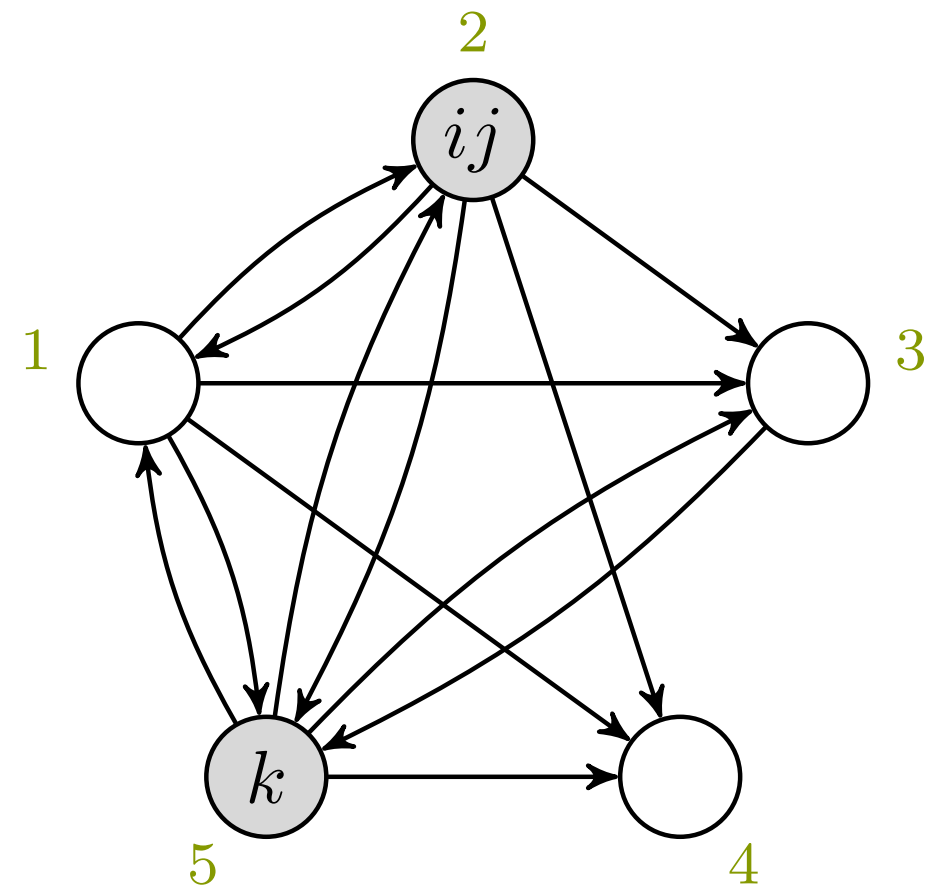
1	1	1	1	1	1
2	1	1	1	1	1
3			1		1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 2, 2$



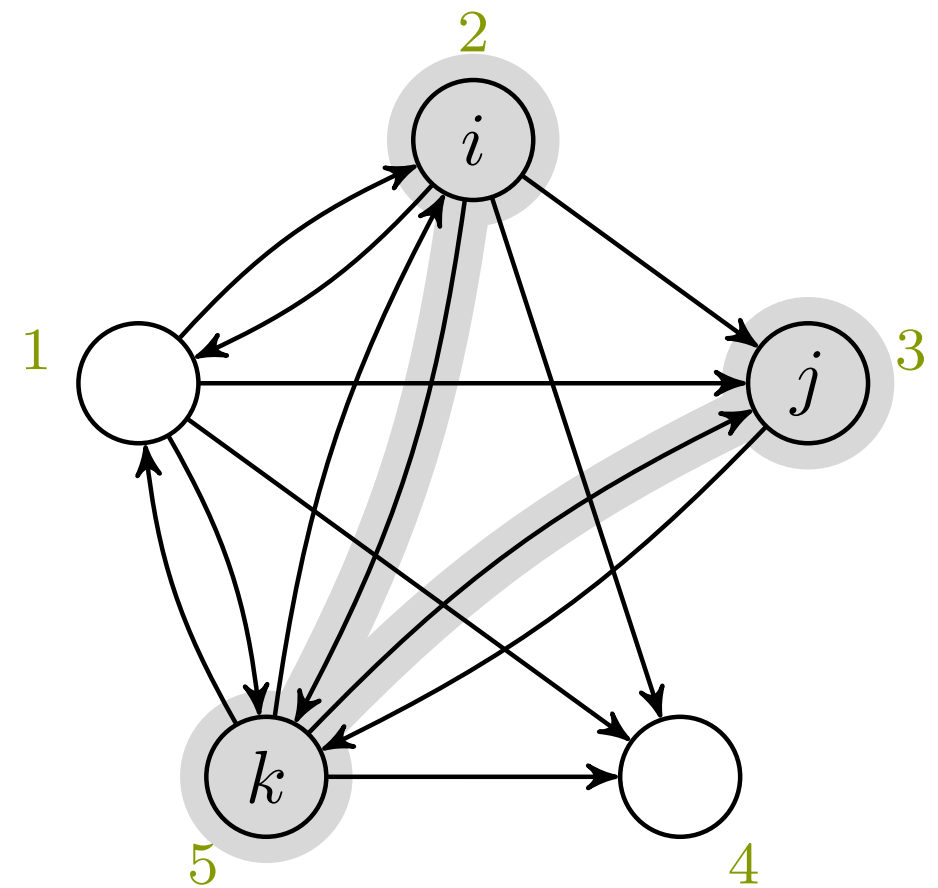
1	1	1	1	1	1
2	1	1	1	1	1
3			1		1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 2, 3$



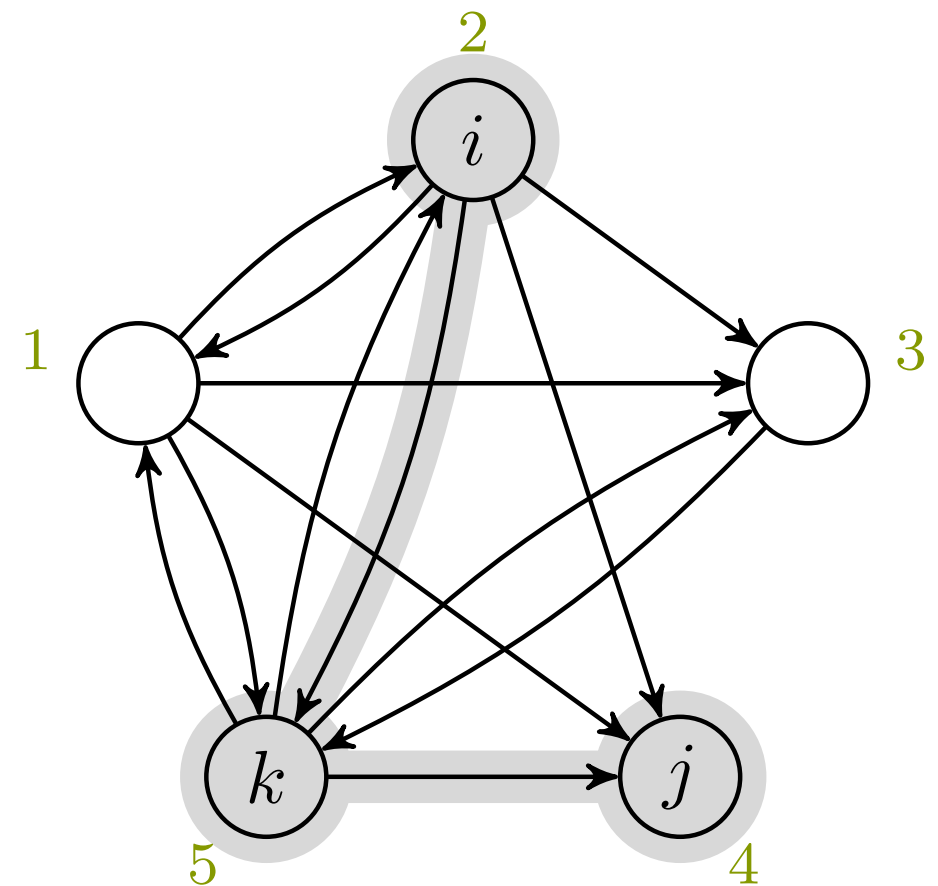
1	1	1	1	1	1
2	1	1	1	1	1
3			1		1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 2, 4$



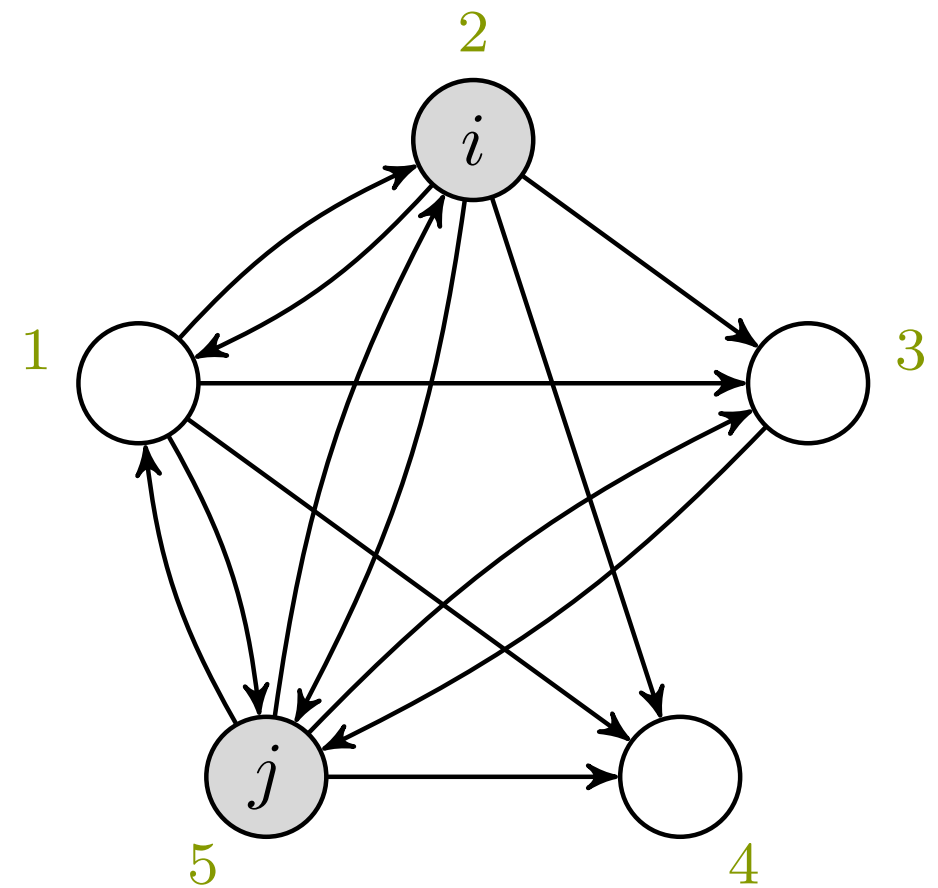
1	1	1	1	1	
2	1	1	1	1	
3			1	1	
4			1		
5	1	1	1	1	
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 2, 5$



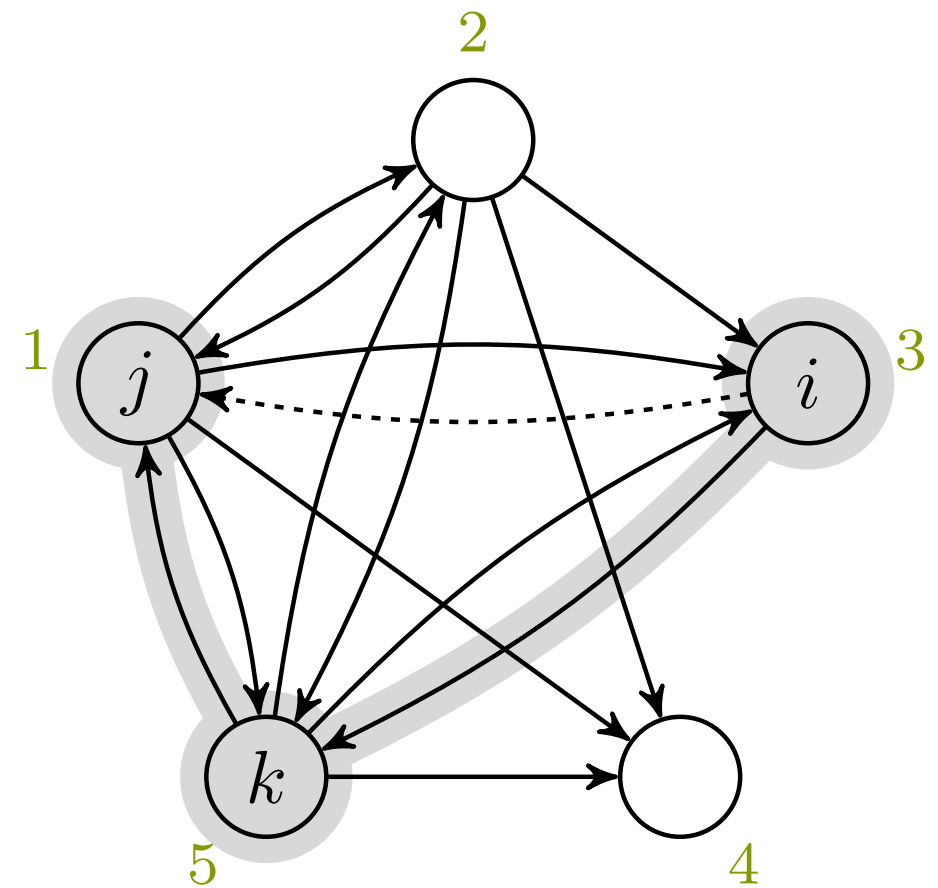
1	1	1	1	1	1
2	1	1	1	1	1
3			1		1
4				1	
5	1	1	1	1	1
	1	2	3	4	5




TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 3, 1$



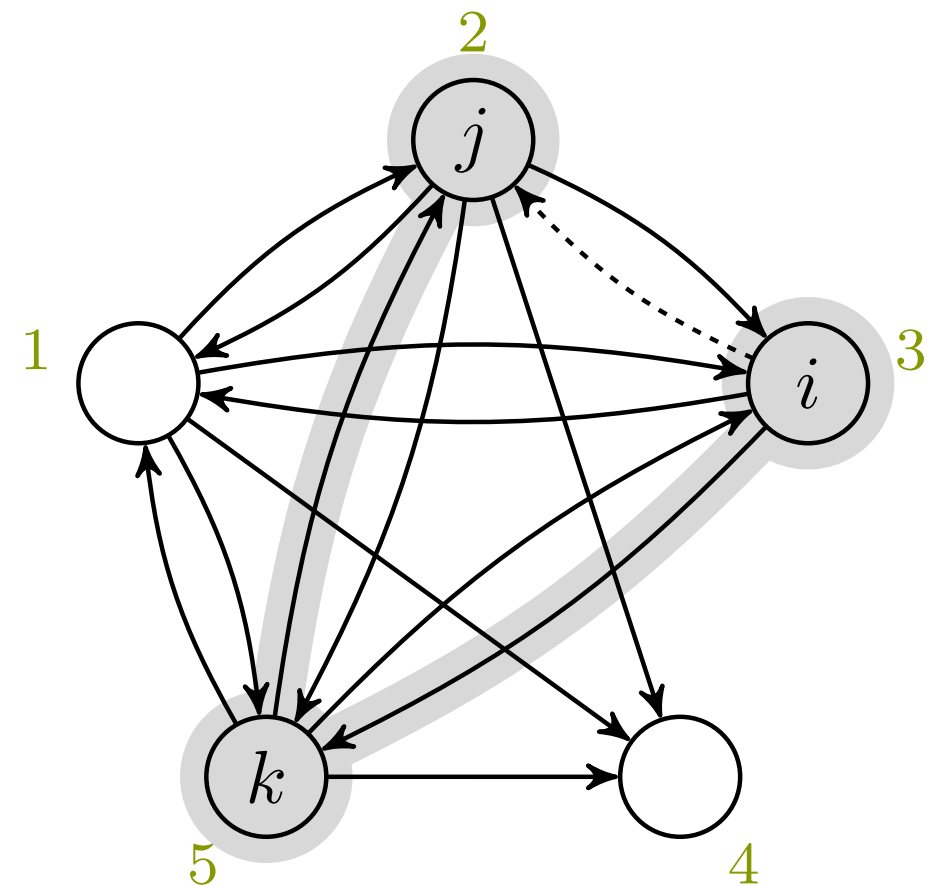
1	1	1	1	1	
2	1	1	1	1	
3			1		
4			1		
5		1	1	1	
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 3, 2$



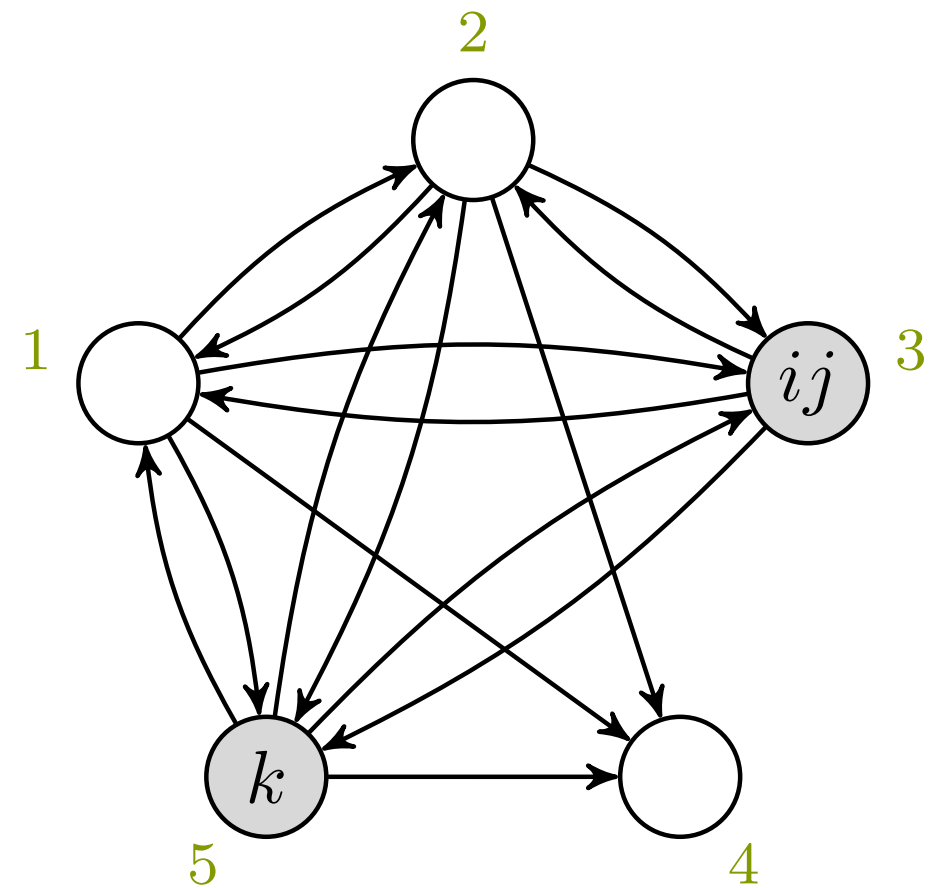
1	1	1	1	1	1
2	1	1	1	1	1
3	1		1		1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 3, 3$



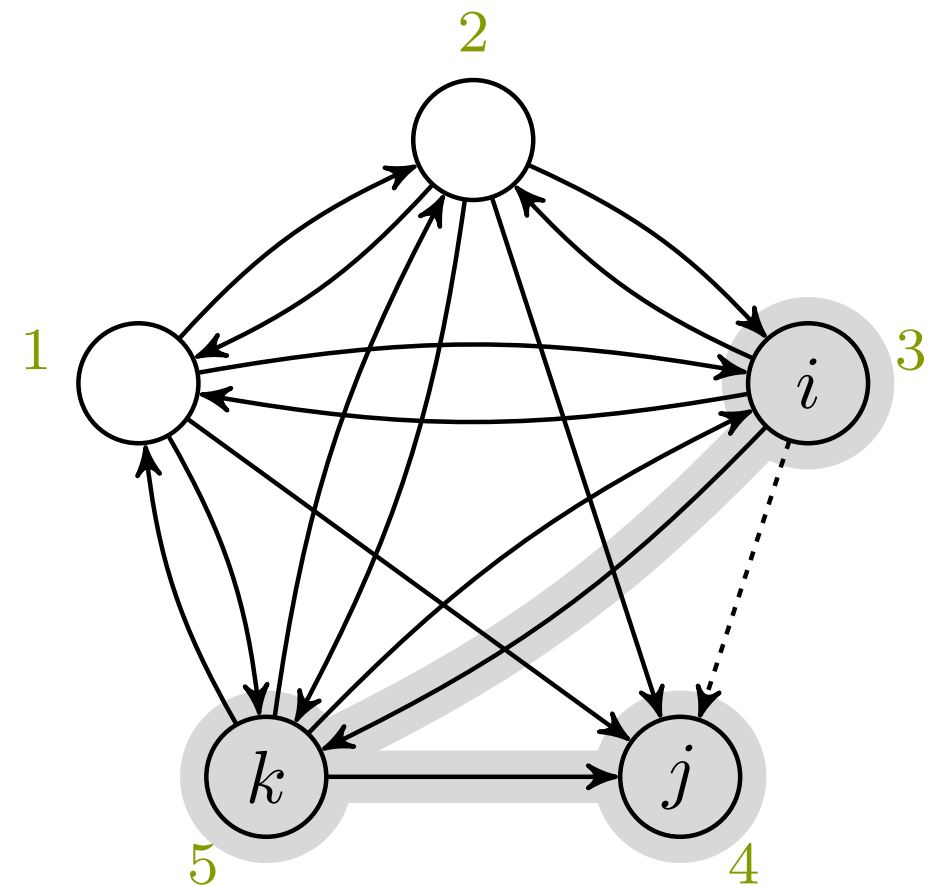
1	1	1	1	1	
2	1	1	1	1	
3	1	1	<div></div> 1	<div></div> 1	
4				1	
5	1	1	1	1	
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 3, 4$



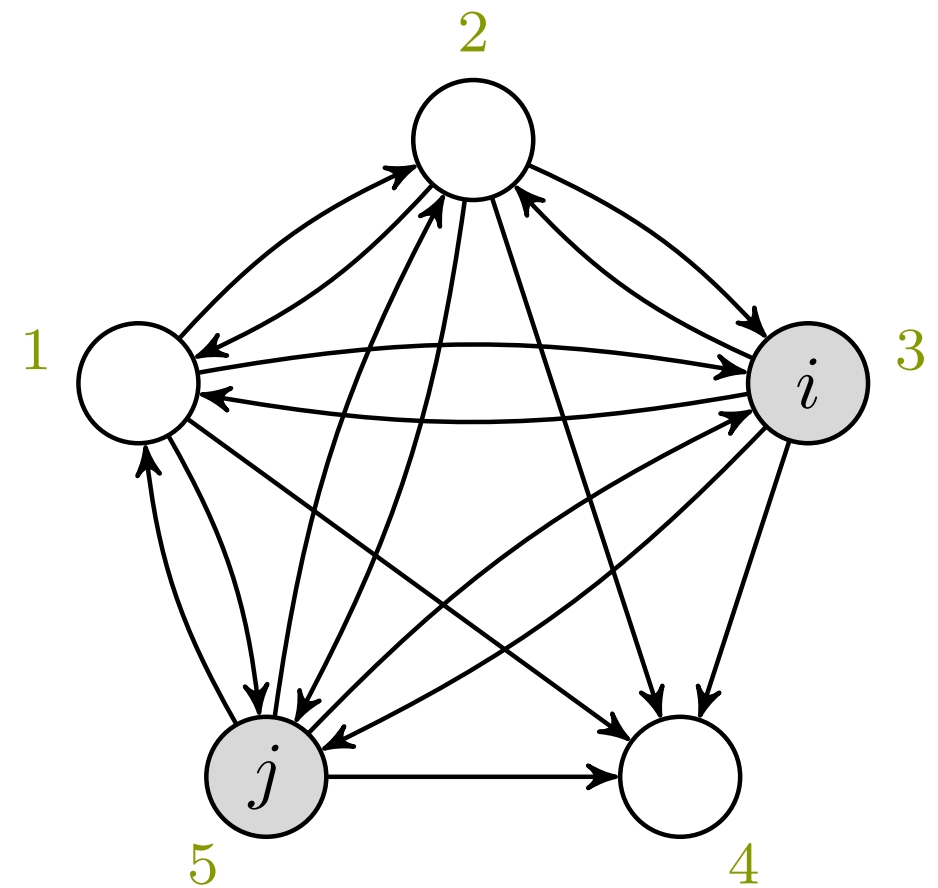
1	1	1	1	1	
2	1	1	1	1	
3	1	1	1	1	
4			1		
5	1	1	1	1	
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 3, 5$



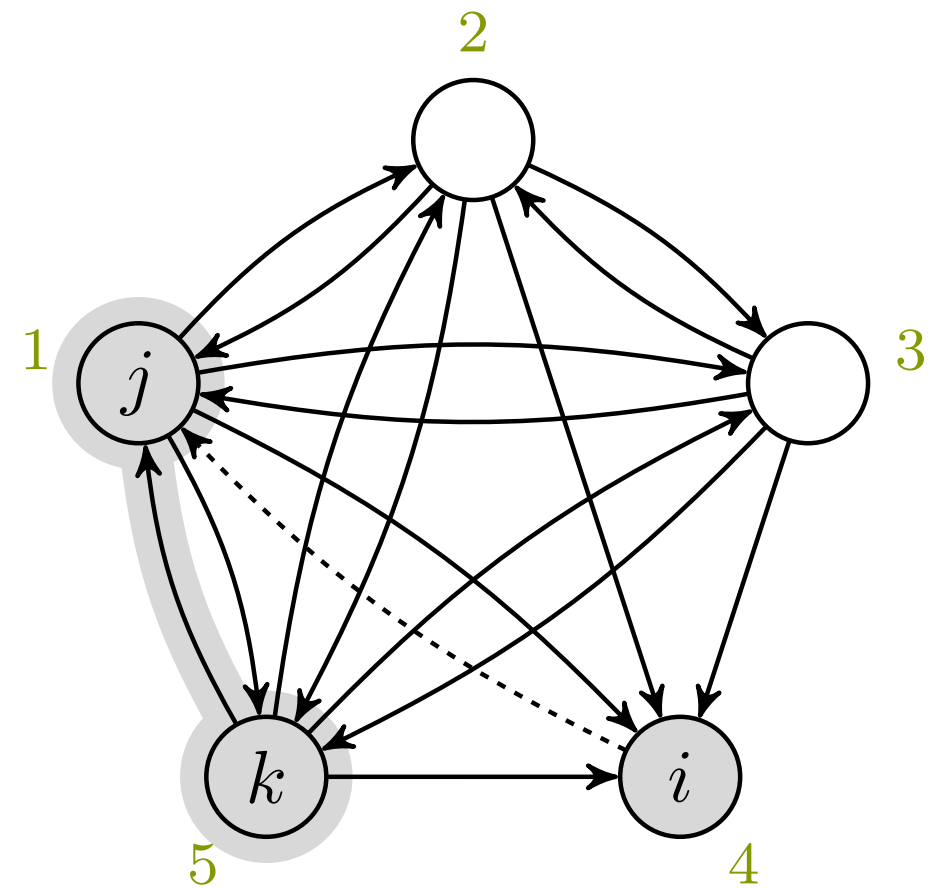
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 4, 1$



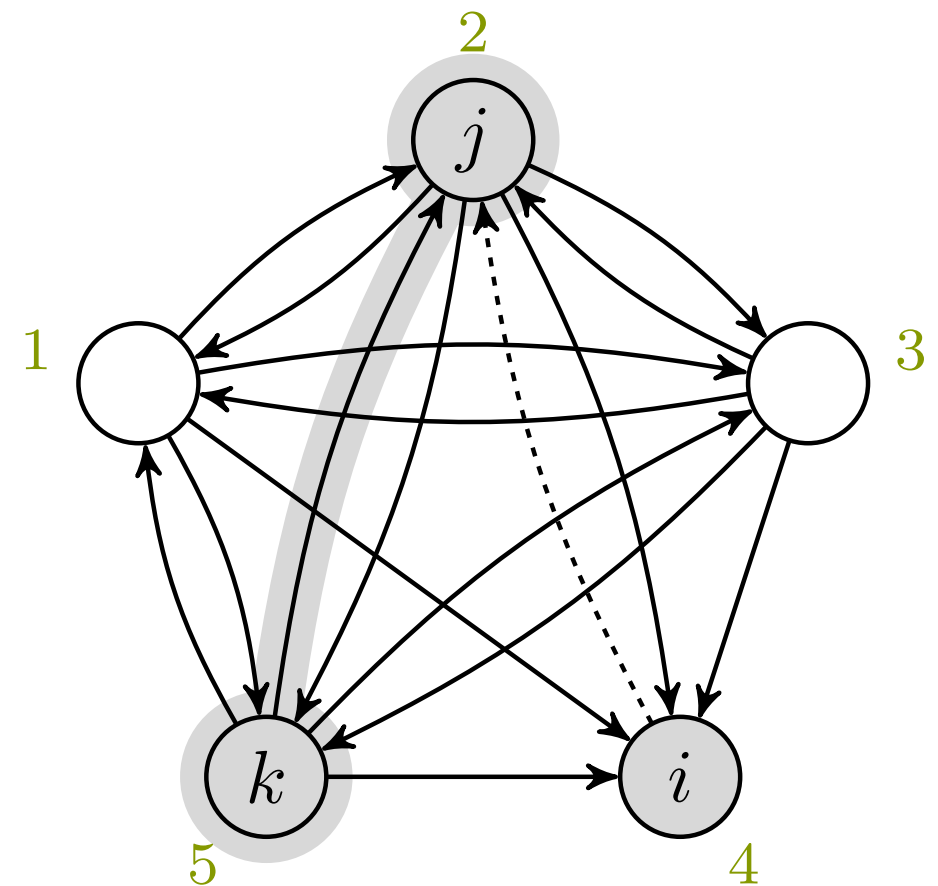
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 4, 2$



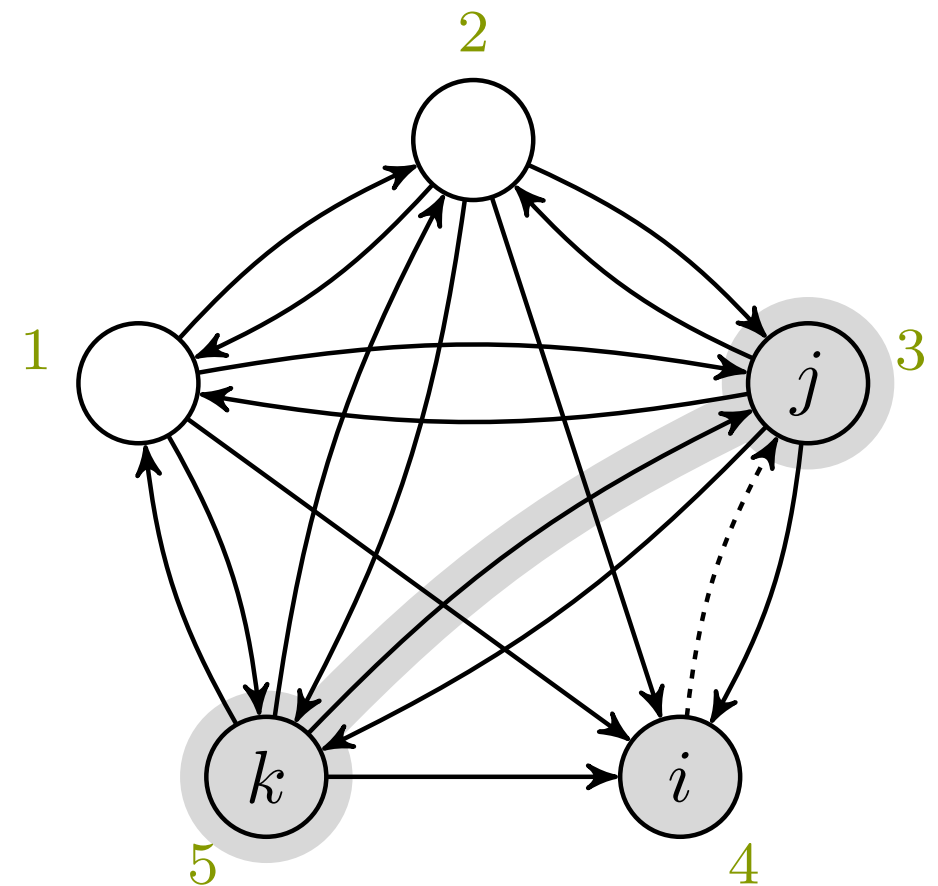
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 4, 3$



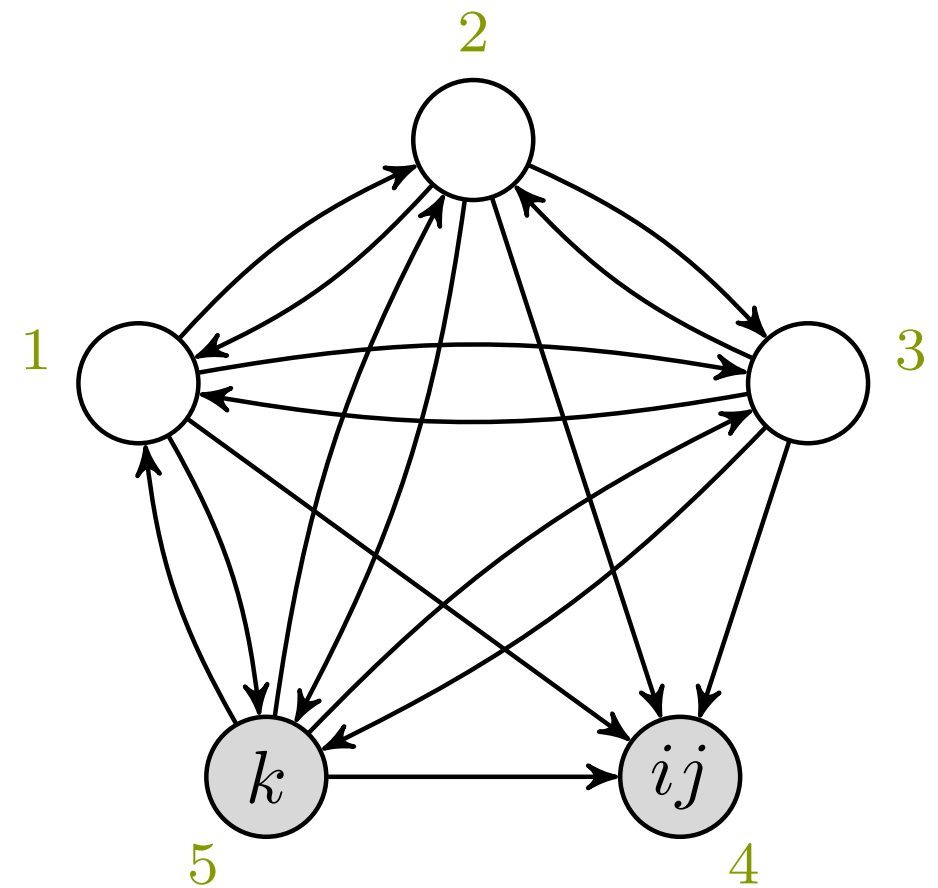
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4				1
5	1	1	1	1
	1	2	3	4

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 4, 4$



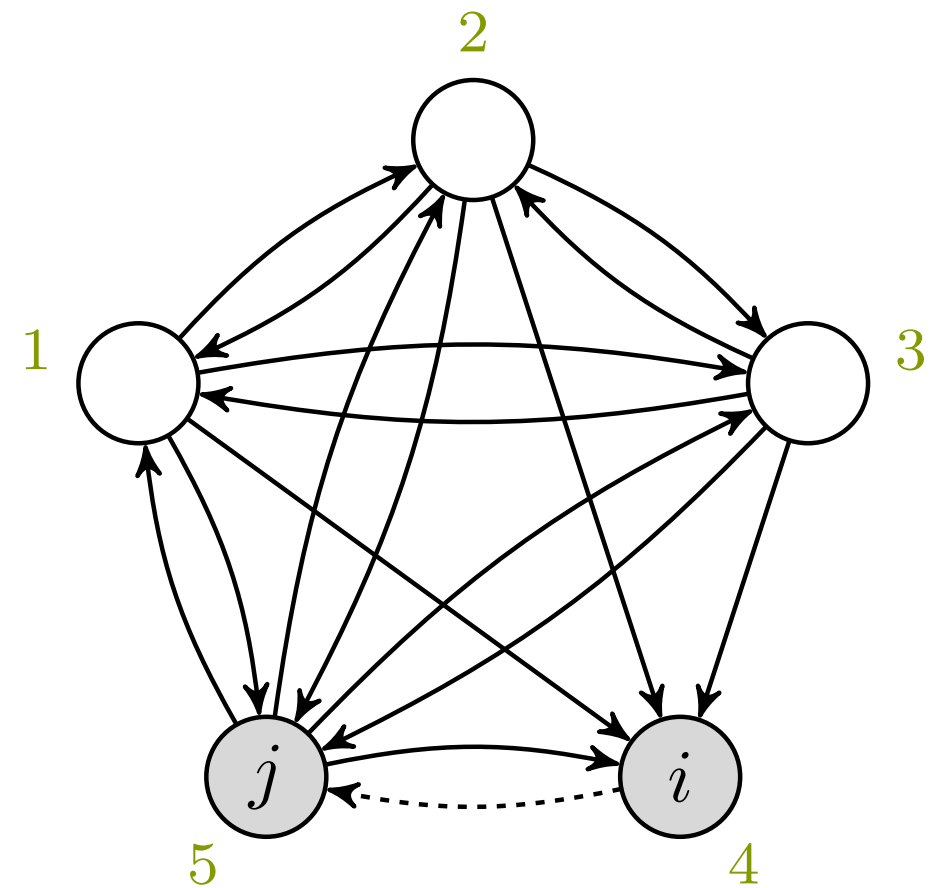
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 4, 5$



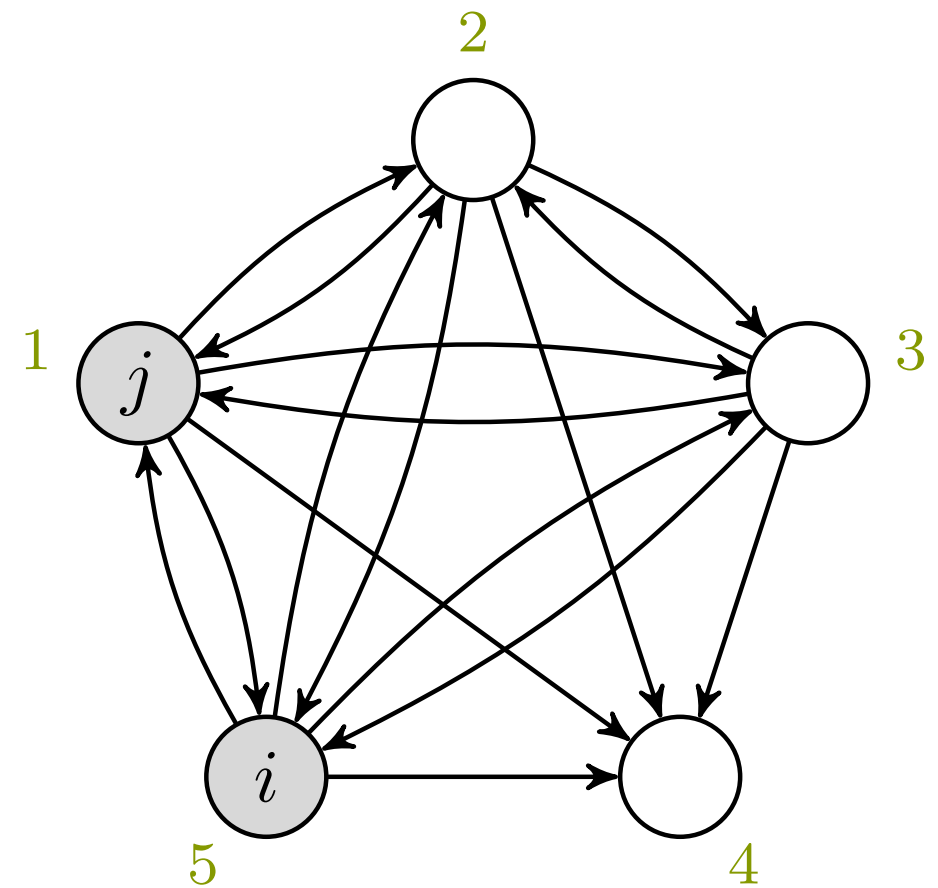
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 5, 1$



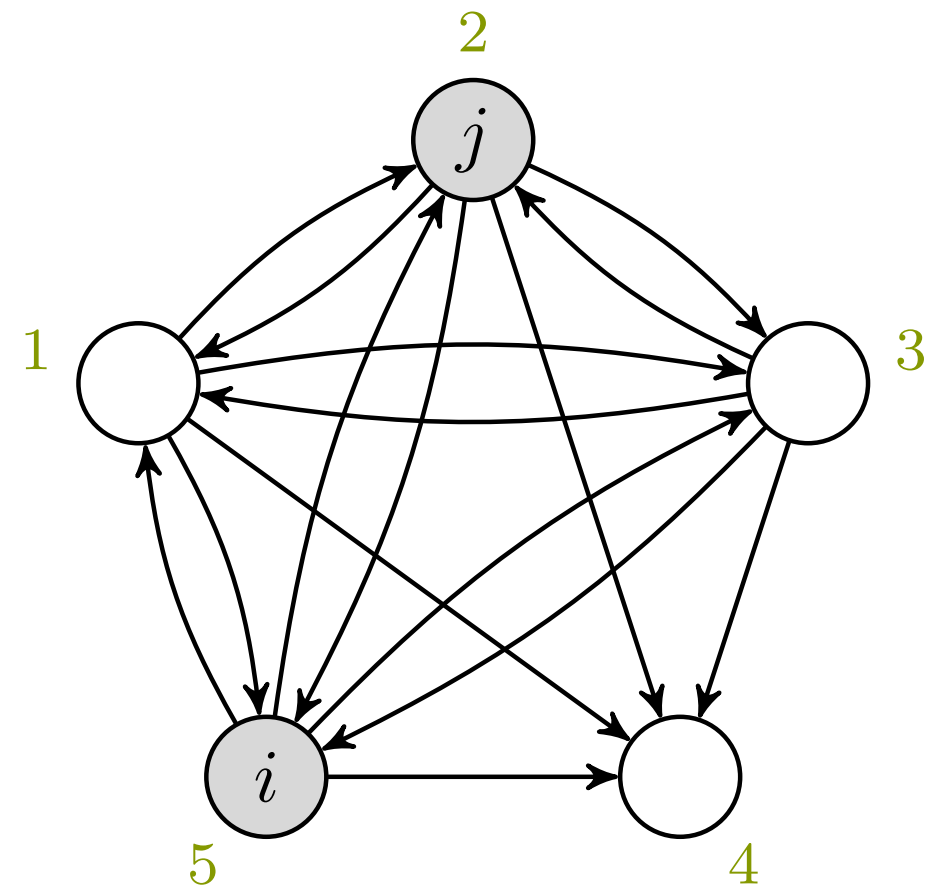
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 5, 2$



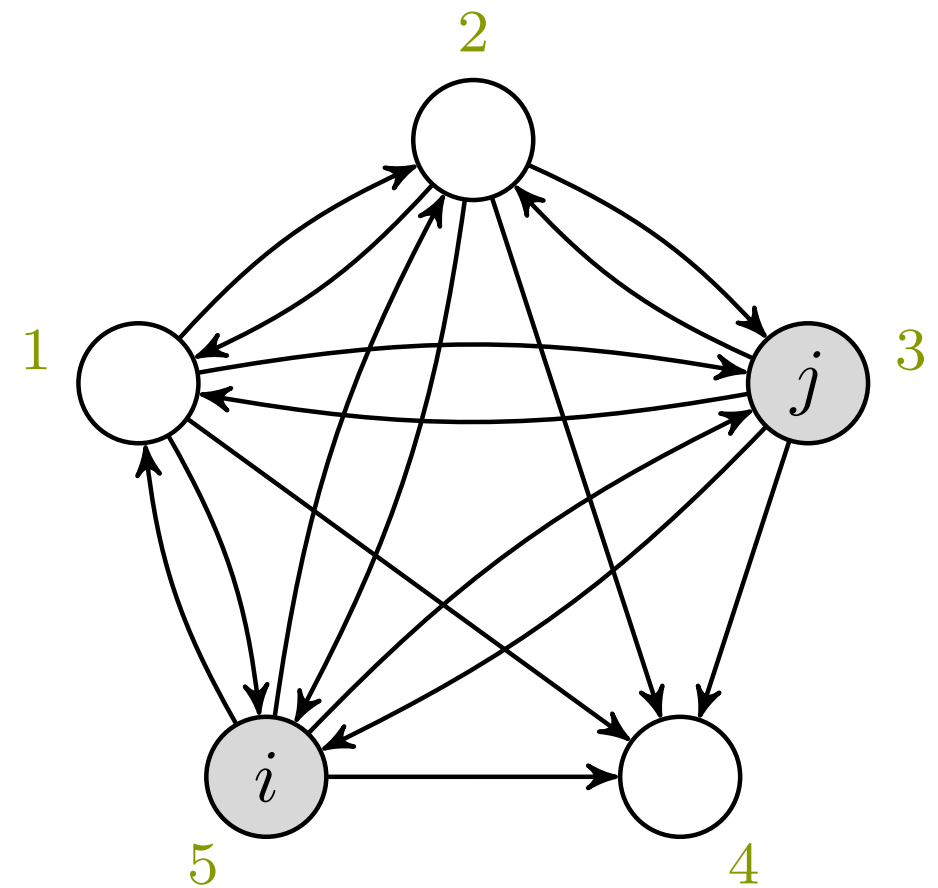
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 5, 3$



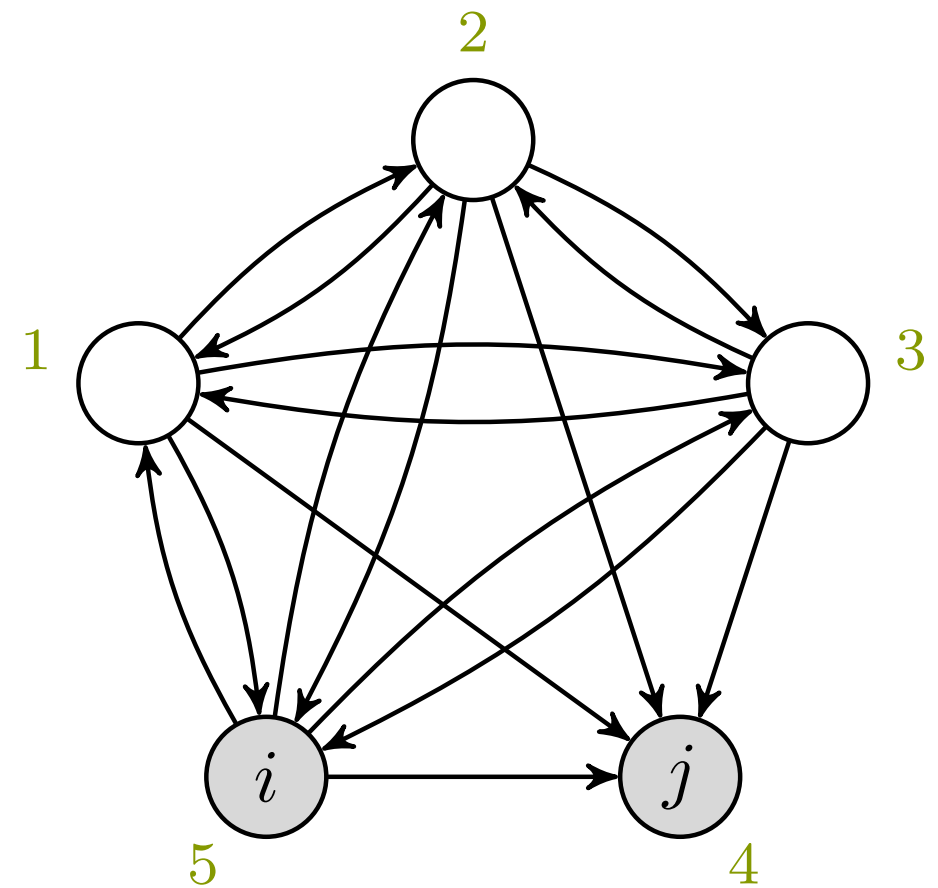
1	1	1	1	1	
2	1	1	1	1	
3	1	1	1	1	
4			1		
5	1	1	1	1	
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 5, 4$



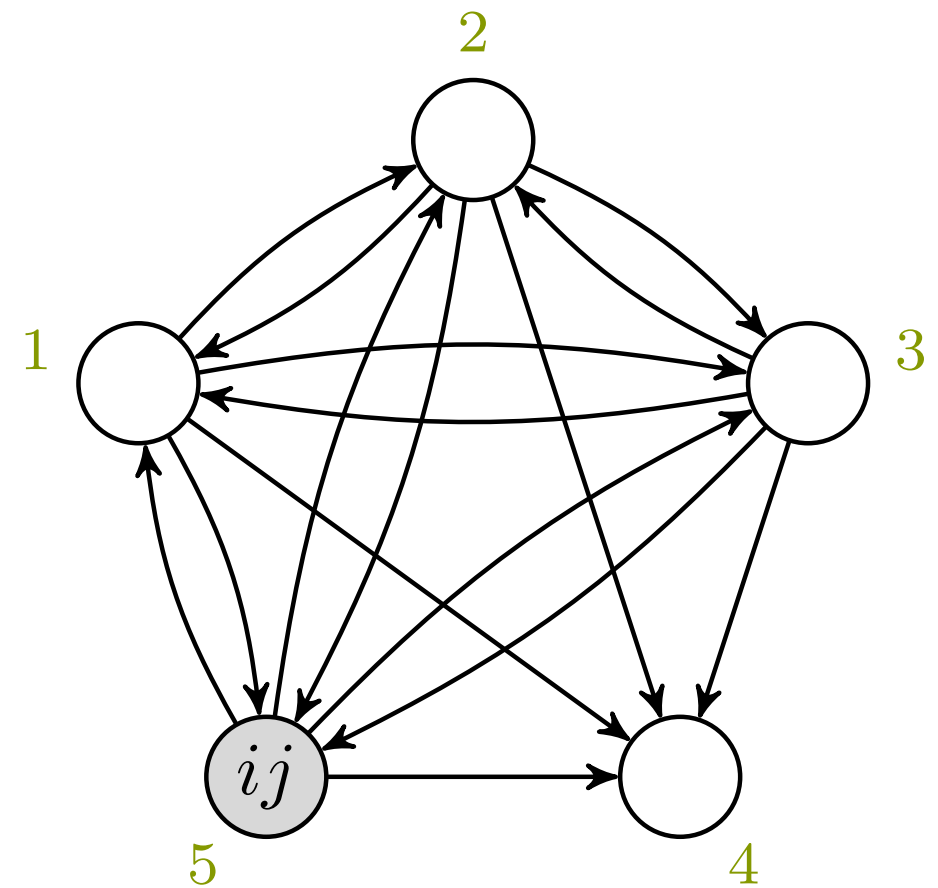
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = 5, 5, 5$



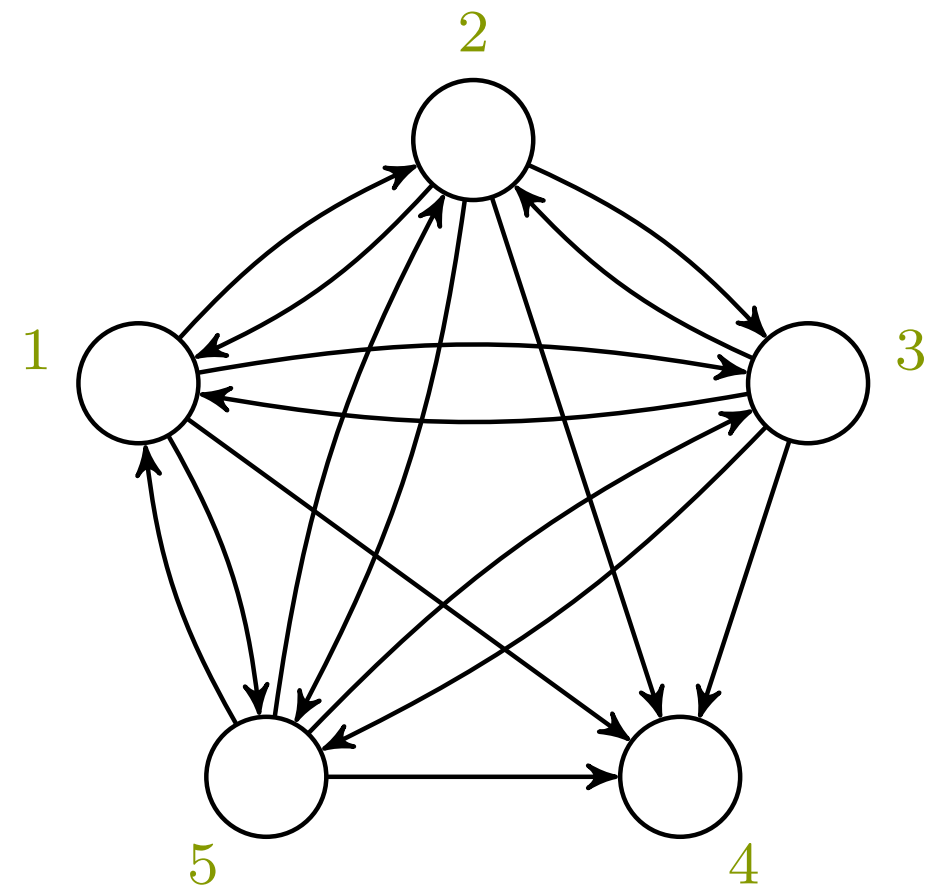
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = -, -, -$



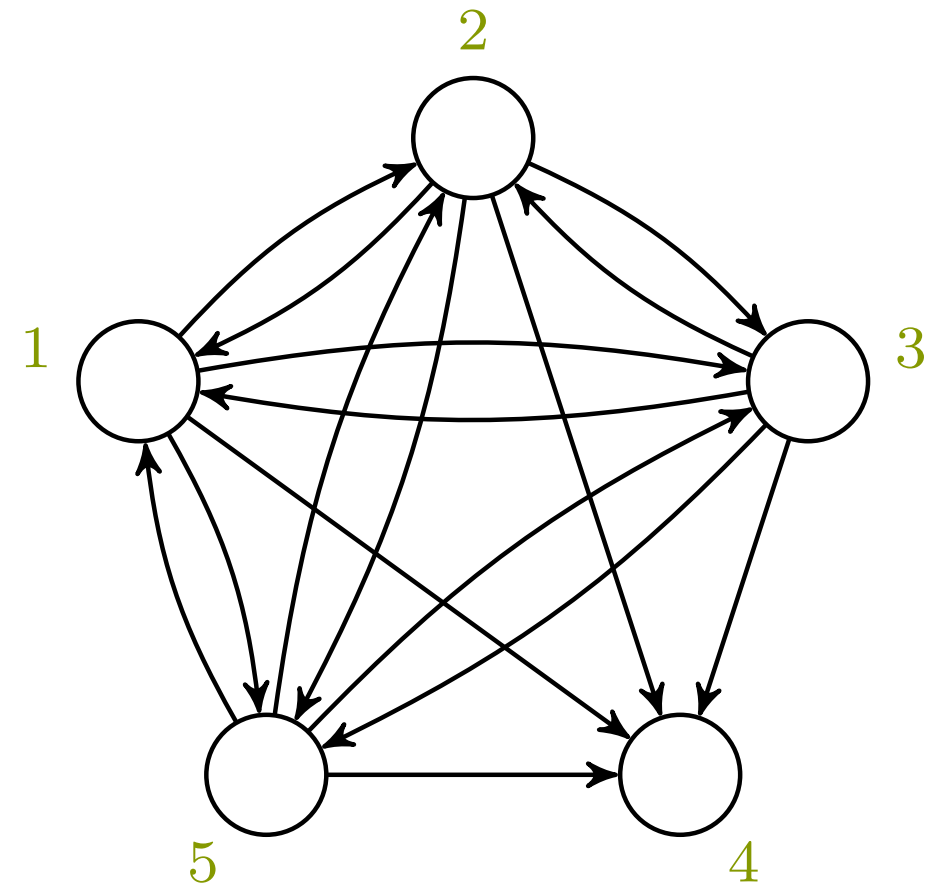
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4				1	
5	1	1	1	1	1
	1	2	3	4	5

TRANSITIVE-CLOSURE'(G)

```

1   $n = |G.V|$ 
2  initialize T
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6               $t_{ij} = t_{ij} \vee (t_{ik} \wedge t_{kj})$ 
7  return T
    
```

$k, i, j = -, -, -$



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
	1	2	3	4	5

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 ›  $O(1)$ 

```

```

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.

3:3

Floyd-Warshall

ALGORITHM 97
SHORTEST PATH
ROBERT W. FLOYD
Armour Research Foundation, Chicago, Ill.

procedure shortest path (m, n); **value** n ; **integer** $m[i, j]$ is the length of a direct link from point i of a network to point j . If no direct link exists, $m[i, j]$ is initially ∞ . At completion, $m[i, j]$ is the length of the shortest path from i to j . If none exists, $m[i, j]$ is ∞ .
SHALL, S. A theorem on Boolean matrices. *J. ACM*, 14, 1961, 317-322.

begin
integer i, j, k ; **real** inf, s ; $inf := \infty$;
for $i := 1$ **step** 1 **until** n **do**
 for $j := 1$ **step** 1 **until** n **do**
 if $m[i, j] < inf$ **then**
 for $k := 1$ **step** 1 **until** n **do**
 if $m[i, k] + m[k, j] < m[i, j]$ **then**
 $s := m[i, k] + m[k, j]$;
 begin $s := m[i, j]$ **end**;
 if $s < m[i, j]$ **then** $m[i, j] := s$;
 end;
 end;
 end;
end shortest path

Contributions to this department are invited and should be stated in the Algorithms Department (Communications, February, 1960) notation should be used (see Communications, February, 1960). Contributions should be sent in duplicate to the National Bureau of Standards, Computation Laboratory, National Bureau of Standards, Washington 25, D. C. Algorithms should be in the form of ALGOL 60 and written in a clear, concise manner. The most recent algorithms appearing in the Communications are the convenience of the printer. The convenience of the printer are delimiters to appear in the form of ALGOL 60 and written in a clear, concise manner. Although each algorithm is the property of its contributor, no warranty, expressed or implied, is made by the editor, or the publisher, or the National Bureau of Standards, or the National Bureau of Standards Machinery as to the accuracy of the algorithm and related algorithms. The reproduction of the algorithm is assumed by the contributor. The reproduction of the algorithm is assumed by the contributor. The reproduction of the algorithm is assumed by the contributor.

The reproduction of the algorithm is assumed by the contributor. The reproduction of the algorithm is assumed by the contributor. The reproduction of the algorithm is assumed by the contributor.

Fra hver node til alle andre?

Fra hver node til alle andre?

- › DIJKSTRA med tabell: $O(V^3 + VE)$

Fra hver node til alle andre?

- › DIJKSTRA med tabell: $O(V^3 + VE)$
- › ...med binærhaug: $O(VE \lg V)$

Fra hver node til alle andre?

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

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: $\Theta(V^2E)$

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: $\Theta(V^2E)$

Målsetting:

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: $\Theta(V^2E)$

Målsetting:

- › Vi vil tillate negative kanter

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: $\Theta(V^2E)$

Målsetting:

- › Vi vil tillate negative kanter
- › Vi vil ha lavere asymptotisk kjøretid...

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: $\Theta(V^2E)$

Målsetting:

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

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: $\Theta(V^2E)$

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?

ijk

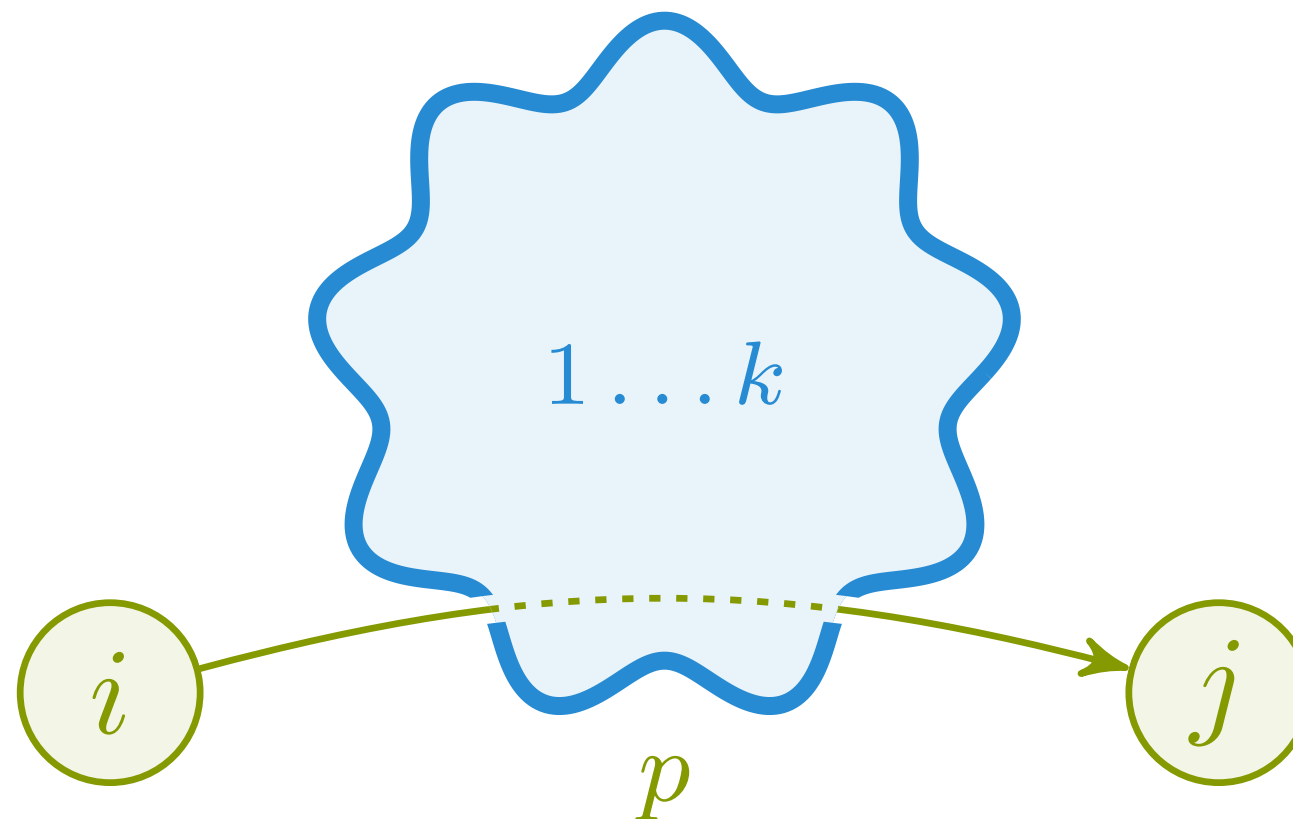
Korteste vei fra i til j via noder fra $\{1 \dots k\}$

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

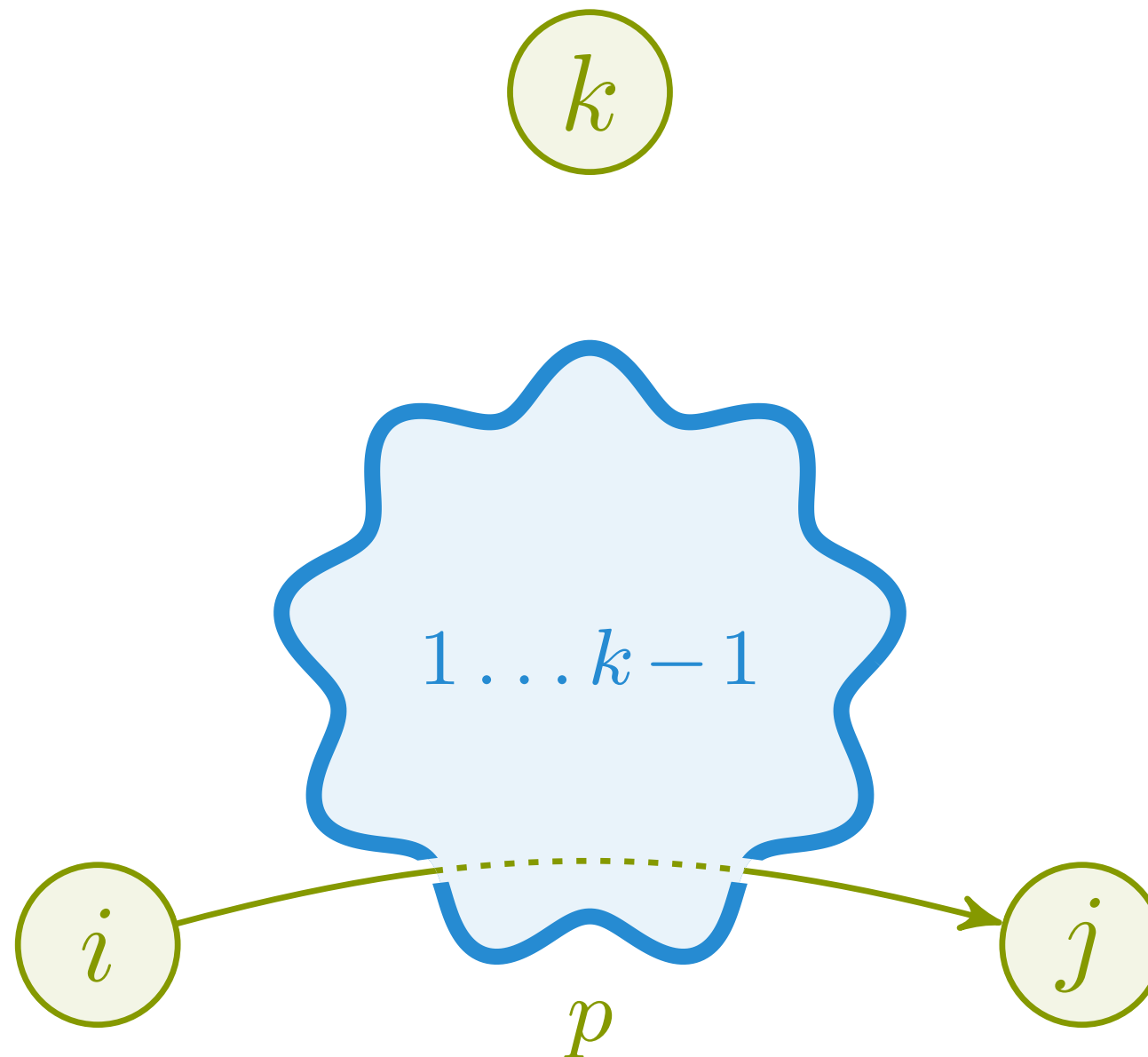
Korteste vei fra i til j via noder fra $\{1 \dots k\}$

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

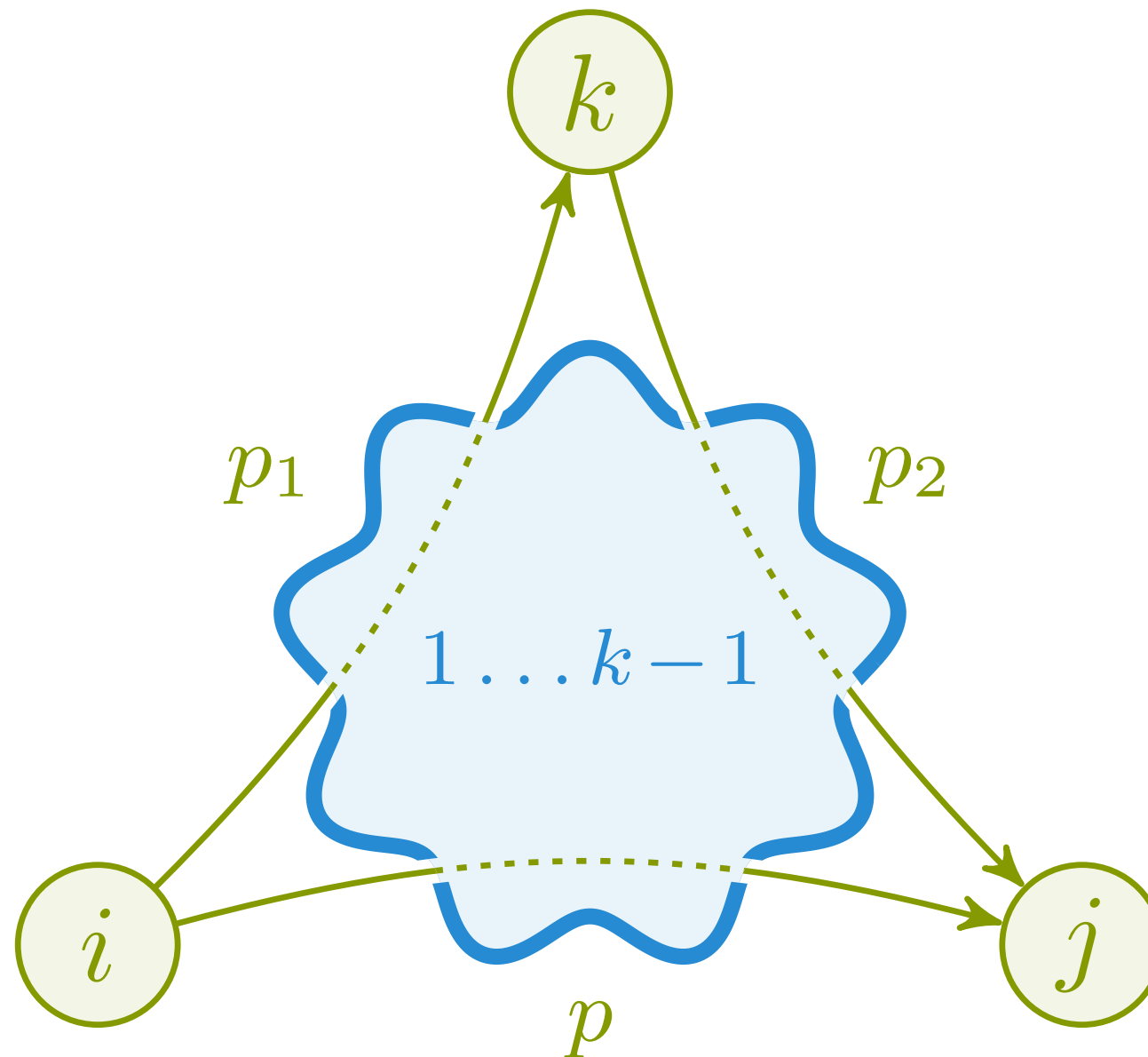
Forgjengeren til j om vi starter i i og går via noder fra $\{1 \dots k\}$



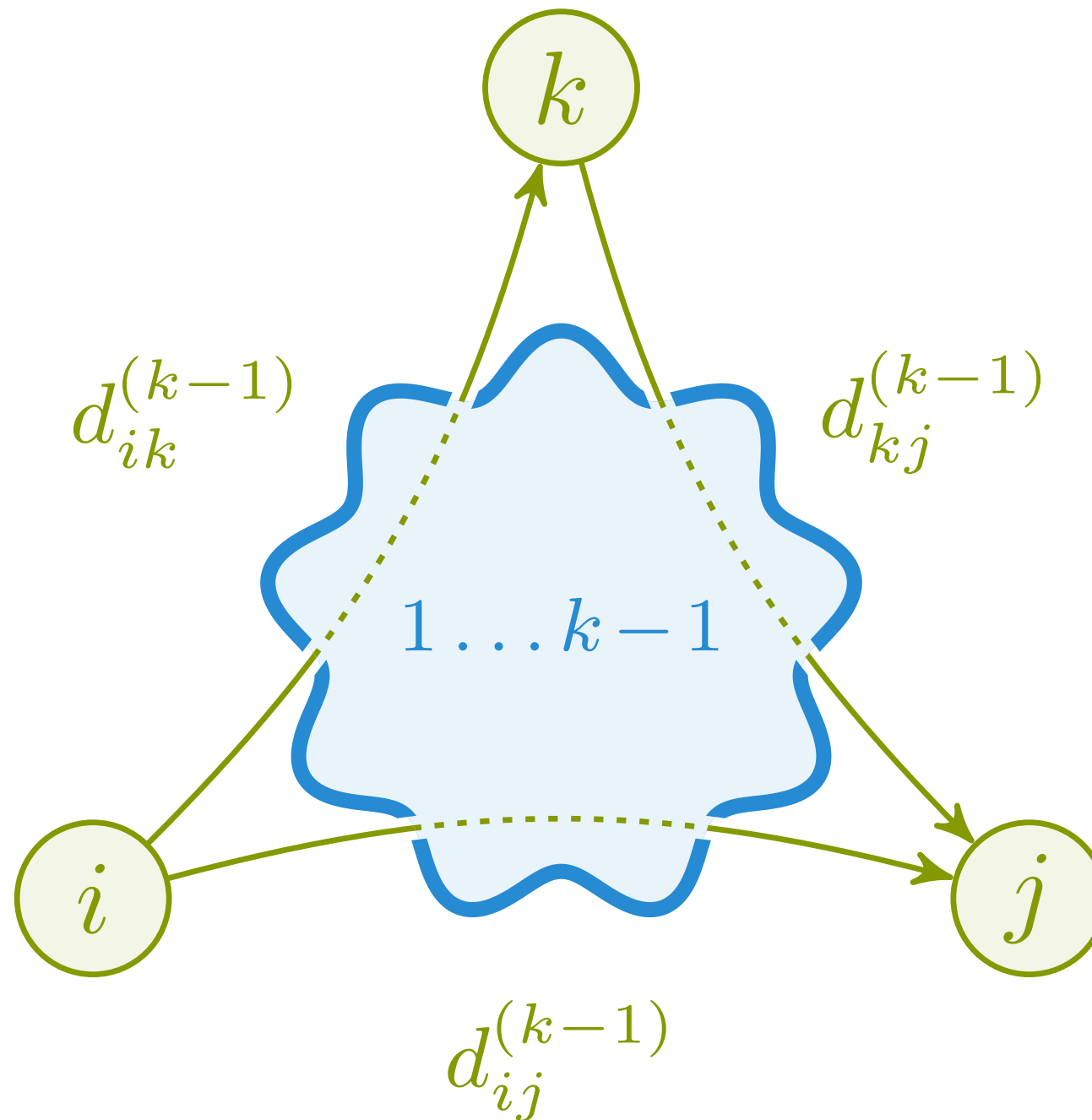
$d_{ij}^{(k)}$ = korteste vei fra i til j via noder fra $\{1 \dots k\}$



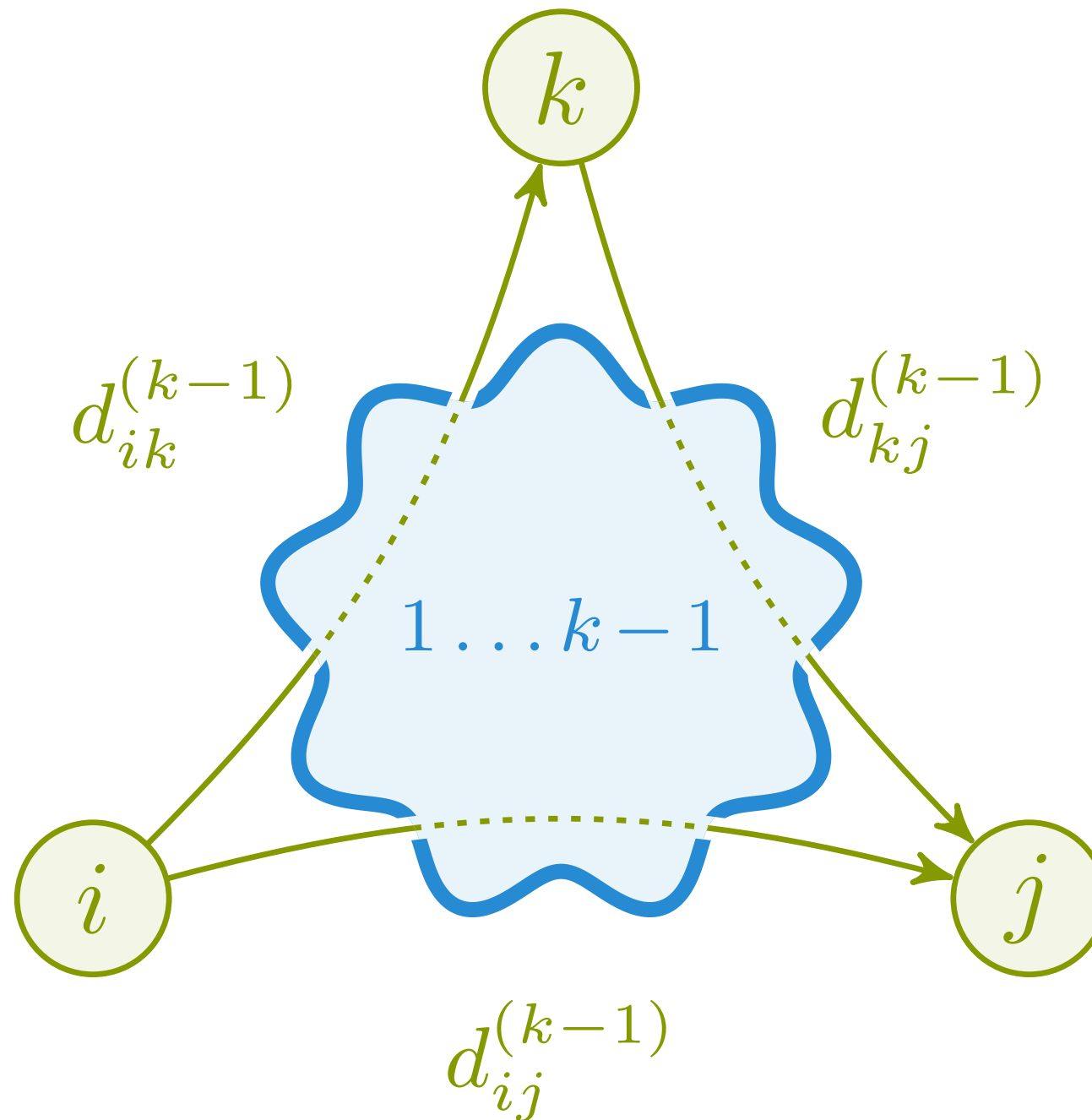
Som for ryggsekkproblemet: Skal k være med eller ikke?



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



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



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

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

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

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

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

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

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

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

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

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)

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

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 $i \rightsquigarrow j$?

FLOYD-WARSHALL(W)

1 $n = W.rows$

FLOYD-WARSHALL(W)

1 $n = W.rows$

2 $D^{(0)} = W$

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

FLOYD-WARSHALL(W)

1 $n = W.rows$

2 $D^{(0)} = W$

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

Vi får nå lov til å gå innom node k

FLOYD-WARSHALL(W)

1 $n = W.rows$

2 $D^{(0)} = W$

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

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

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

FLOYD-WARSHALL(W)

1 $n = W.rows$

2 $D^{(0)} = W$

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

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

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

For hver mulig startnode...

FLOYD-WARSHALL(W)

1 $n = W.rows$

2 $D^{(0)} = W$

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

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

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

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

For hver mulig sluttnode...

FLOYD-WARSHALL(W)

1 $n = W.rows$

2 $D^{(0)} = W$

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

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

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

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

7 $d_{ij}^{(k)} = \min (d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$

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

FLOYD-WARSHALL(W)

1 $n = W.rows$

2 $D^{(0)} = W$

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

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

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

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

7 $d_{ij}^{(k)} = \min (d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$

8 **return** $D^{(n)}$

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

FLOYD-WARSHALL'(W)

W vektor

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

FLOYD-WARSHALL'(W)
1 $n = W.rows$

W vektor
 n ant. noder

FLOYD-WARSHALL'(W)

1 $n = W.rows$

2 initialize D and Π

W vektor

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 $n = W.rows$

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)

1 $n = W.rows$

2 initialize D and Π

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

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

W vekker

n ant. noder

D avstander

Π forgjengere

k mellomlanding

i fra-node

For hver mulig startnode...

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 

```

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

For hver mulig sluttnode...

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 

```

W vektor
 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 $\{1, \dots, k - 1\}$)?

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 

```

W vektor
 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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
    
```

W vektor
 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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

W vektor
 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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 1, 1$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 1, 2$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 1, 3$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 1, 1, 4$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 1, 1, 5$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 2, 1$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 2, 2$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 2, 3$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 1, 2, 4$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 1, 2, 5$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 3, 1$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 1, 3, 2$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 1, 3, 3$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 3, 4$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 1, 3, 5$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 4, 1$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 4, 2$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 4, 2$

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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 4, 2$

D

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-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)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 4, 3$

D

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

Π

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 4, 4$

D

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

Π

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 1, 4, 5$

D

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

Π

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 4, 5$

D

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

Π

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 4, 5$

D

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

Π

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 1, 5, 1$

D

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

Π

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 1, 5, 2$

D

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

Π

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 5, 3$

D

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

Π

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 1, 5, 4$

D

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

Π

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 1, 5, 5$

D

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

Π

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 2, 1, 1$

D

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

Π

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 5, 1, 1$

D

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 5, 1, 2$

D

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 5, 1, 2$

D

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 5, 1, 2$

D

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 5, 1, 3$

D

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 5, 1, 3$

D

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 5, 1, 3$

D

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 5, 1, 4$

D

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 5, 1, 4$

D

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = 5, 1, 4$

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

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 
    
```

$k, i, j = 5, 1, 5$

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	

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = -, -, -$

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	

FLOYD-WARSHALL'(W)

```

1   $n = W.rows$ 
2  initialize D and  $\Pi$ 
3  for  $k = 1$  to  $n$ 
4      for  $i = 1$  to  $n$ 
5          for  $j = 1$  to  $n$ 
6              if  $d_{ij} > d_{ik} + d_{kj}$ 
7                   $d_{ij} = d_{ik} + d_{kj}$ 
8                   $\pi_{ij} = \pi_{kj}$ 
9  return D,  $\Pi$ 

```

$k, i, j = -, -, -$

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 ?


```
PRINT-ALL-PAIRS-SHORTEST-PATH( $\Pi, i, j$ )
1  if  $i == j$ 
```

Samme node? Kommer ingenstedsfra...

```

PRINT-ALL-PAIRS-SHORTEST-PATH( $\Pi, i, j$ )
1  if  $i == j$ 
2      print  $i$ 

```

...så bare skriv ut noden

```

PRINT-ALL-PAIRS-SHORTEST-PATH( $\Pi, i, j$ )
1  if  $i == j$ 
2      print  $i$ 
3  elseif  $\pi_{ij} == \text{NIL}$ 

```

Hvis vi ellers ikke kom fra noe sted...

```

PRINT-ALL-PAIRS-SHORTEST-PATH( $\Pi, i, j$ )
1  if  $i == j$ 
2      print  $i$ 
3  elseif  $\pi_{ij} == \text{NIL}$ 
4      print “no path from”  $i$  “to”  $j$  “exists”

```

...så finnes ingen sti!

```

PRINT-ALL-PAIRS-SHORTEST-PATH( $\Pi, i, j$ )
1  if  $i == j$ 
2      print  $i$ 
3  elseif  $\pi_{ij} == \text{NIL}$ 
4      print “no path from”  $i$  “to”  $j$  “exists”
5  else PRINT-ALL-PAIRS-SHORTEST-PATH( $\Pi, i, \pi_{ij}$ )
    
```

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

```
PRINT-ALL-PAIRS-SHORTEST-PATH( $\Pi, i, j$ )  
1  if  $i == j$   
2      print  $i$   
3  elseif  $\pi_{ij} == \text{NIL}$   
4      print “no path from”  $i$  “to”  $j$  “exists”  
5  else PRINT-ALL-PAIRS-SHORTEST-PATH( $\Pi, i, \pi_{ij}$ )  
6      print  $j$ 
```

... og deretter sluttnoden

PRINT-PATH(Π, i, j)

```

1  if  $i == j$ 
2      print  $i$ 
3  elseif  $\pi_{ij} == \text{NIL}$ 
4      print "no such path"
5  else PRINT-PATH( $\Pi, i, \pi_{ij}$ )
6      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)

1 **if** $i == j$

2 print i

3 **elseif** $\pi_{ij} == \text{NIL}$

4 print “no such path”

5 **else** PRINT-PATH(Π, i, π_{ij})

6 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)

1 **if** $i == j$

2 print i

3 **elseif** $\pi_{ij} == \text{NIL}$

4 print “no such path”

5 **else** PRINT-PATH(Π, i, π_{ij})

6 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	

$i, j = 1, 2$

PRINT-PATH(Π, i, j)

```

1  if  $i == j$ 
2      print  $i$ 
3  elseif  $\pi_{ij} == \text{NIL}$ 
4      print "no such path"
5  else PRINT-PATH( $\Pi, i, \pi_{ij}$ )
6      print  $j$ 

```

$i, j = 1, 2 \succ 1, 3$

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)

1 **if** $i == j$

2 print i

3 **elseif** $\pi_{ij} == \text{NIL}$

4 print “no such path”

5 **else** PRINT-PATH(Π, i, π_{ij})

6 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	

$i, j = 1, 2 \succ 1, 3$

PRINT-PATH(Π, i, j)

1 **if** $i == j$

2 print i

3 **elseif** $\pi_{ij} == \text{NIL}$

4 print “no such path”

5 **else** PRINT-PATH(Π, i, π_{ij})

6 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	

$i, j = 1, 2 \succ 1, 3$

PRINT-PATH(Π, i, j)

```

1  if  $i == j$ 
2      print  $i$ 
3  elseif  $\pi_{ij} == \text{NIL}$ 
4      print "no such path"
5  else PRINT-PATH( $\Pi, i, \pi_{ij}$ )
6      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	

$i, j = 1, 2 \succ 1, 3 \succ 1, 4$

PRINT-PATH(Π, i, j)

1 **if** $i == j$

2 print i

3 **elseif** $\pi_{ij} == \text{NIL}$

4 print “no such path”

5 **else** PRINT-PATH(Π, i, π_{ij})

6 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	

$i, j = 1, 2 \succ 1, 3 \succ 1, 4$

PRINT-PATH(Π, i, j)

1 **if** $i == j$

2 print i

3 **elseif** $\pi_{ij} == \text{NIL}$

4 print “no such path”

5 **else** PRINT-PATH(Π, i, π_{ij})

6 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	

$i, j = 1, 2 \succ 1, 3 \succ 1, 4$

PRINT-PATH(Π, i, j)

```

1  if  $i == j$ 
2      print  $i$ 
3  elseif  $\pi_{ij} == \text{NIL}$ 
4      print "no such path"
5  else PRINT-PATH( $\Pi, i, \pi_{ij}$ )
6      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	

$i, j = 1, 2 \succ 1, 3 \succ 1, 4 \succ 1, 5$

PRINT-PATH(Π, i, j)

1 **if** $i == j$

2 print i

3 **elseif** $\pi_{ij} == \text{NIL}$

4 print “no such path”

5 **else** PRINT-PATH(Π, i, π_{ij})

6 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	

$i, j = 1, 2 \succ 1, 3 \succ 1, 4 \succ 1, 5$

PRINT-PATH(Π, i, j)

1 **if** $i == j$

2 print i

3 **elseif** $\pi_{ij} == \text{NIL}$

4 print “no such path”

5 **else** PRINT-PATH(Π, i, π_{ij})

6 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	

$i, j = 1, 2 \succ 1, 3 \succ 1, 4 \succ 1, 5$

PRINT-PATH(Π, i, j)

```

1  if  $i == j$ 
2      print  $i$ 
3  elseif  $\pi_{ij} == \text{NIL}$ 
4      print "no such path"
5  else PRINT-PATH( $\Pi, i, \pi_{ij}$ )
6      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	

$i, j = 1, 2 \succ 1, 3 \succ 1, 4 \succ 1, 5 \succ 1, 1$

PRINT-PATH(Π, i, j)

1 **if** $i == j$

2 print i

3 **elseif** $\pi_{ij} == \text{NIL}$

4 print “no such path”

5 **else** PRINT-PATH(Π, i, π_{ij})

6 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	

$i, j = 1, 2 \succ 1, 3 \succ 1, 4 \succ 1, 5 \succ 1, 1$

PRINT-PATH(Π, i, j)

```

1  if  $i == j$ 
2      print  $i$ 
3  elseif  $\pi_{ij} == \text{NIL}$ 
4      print "no such path"
5  else PRINT-PATH( $\Pi, i, \pi_{ij}$ )
6      print  $j$ 

```

$i, j = 1, 2 \succ 1, 3 \succ 1, 4 \succ 1, 5$

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)

```

1  if  $i == j$ 
2      print  $i$ 
3  elseif  $\pi_{ij} == \text{NIL}$ 
4      print "no such path"
5  else PRINT-PATH( $\Pi, i, \pi_{ij}$ )
6      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	

$i, j = 1, 2 \succ 1, 3 \succ 1, 4$

1 5

PRINT-PATH(Π, i, j)

```

1  if  $i == j$ 
2      print  $i$ 
3  elseif  $\pi_{ij} == \text{NIL}$ 
4      print "no such path"
5  else PRINT-PATH( $\Pi, i, \pi_{ij}$ )
6      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	

$i, j = 1, 2 \succ 1, 3$

1 5 4

PRINT-PATH(Π, i, j)

```

1  if  $i == j$ 
2      print  $i$ 
3  elseif  $\pi_{ij} == \text{NIL}$ 
4      print "no such path"
5  else PRINT-PATH( $\Pi, i, \pi_{ij}$ )
6      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	

1 5 4 3


```

PRINT-PATH( $\Pi, i, j$ )
1  if  $i == j$ 
2      print  $i$ 
3  elseif  $\pi_{ij} == \text{NIL}$ 
4      print “no such path”
5  else PRINT-PATH( $\Pi, i, \pi_{ij}$ )
6      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	

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 ›  $O(1)$ 

```

```

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**