Chapter 25: All-Pairs Shortest-Paths

Some Algorithms

- When no negative edges
  - $O(VE + V^2 \log V)$ time (how?)
- When no negative cycles
  - Bellman-Ford [1962]: $O(V^2E)$ time
  - Floyd-Warshall [1962]: $O(V^3)$ time
  - Johnson [1977]: $O(VE + V^2 \log V)$ time
    - based on a clever combination of Bellman-Ford and Dijkstra

Notations

- Input form: matrix $W = (w_{ij})$
- $w_{ij} = 0$ if $i = j$,
- w_{ij} = the weight of the directed edge if $i \neq j$ and $(i, j) \in E$,
- Otherwise $w_{ij} = \infty$
- Output: $D = (d_{ij})$,
- $d_{ij} = \delta(i,j)$ the shortest weight path from i to j

Some Algorithms

- $G = (V, E)$ with $w: E \rightarrow R$
- Suppose that $w(e) \geq 0$ for all $e \in E$
  Using Dijkstra's algorithm: $O(V^3)$
- Using Binary heap implementation: $O(VE \lg V)$
- Using Fibonacci heap: $O(V^2 \lg V + VE)$
- Suppose negative weight are allowed

Using Bellman-Ford algorithm: $O(V^2 E) = O(V^4)$

A dynamic programming approach:

1. characterize the structure of an optimal solution,
2. recursively define the value of an optimal solution,
3. compute the value of an optimal solution in a bottom-up fashion.

The structure of an optimal solution

Consider a shortest path p from vertex i to vertex j, and suppose that p contains at most m edges. Assume that there are no negative-weight cycles. Hence $m \leq n-1$ is finite.

The structure of an optimal solution

- If i = j, then p has weight 0 and no edge.
- If i≠ j, we decompose p into $i \overset{p'}{\leadsto} k \rightarrow j$ where p' contains at most m-1 edges.
- Moreover, p' is a shortest path from i to k and $\delta(i,j) = \delta(i,k) + w_{kj}$

Recursive solution to the all-pairs shortest-path problem

- Define: $l_{ij}^{(m)}$ = minimum weight of any path from i to j that contains at most m edges.
- $l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{if } i \neq j \end{cases}$
- Then $l_{ij}^{(m)} = \min\{ l_{ij}^{(m-1)}, \min_{1 \leq k \leq n} \{l_{ik}^{(m-1)} + w_{kj}\}\} = \min_{1 \leq k \leq n} \{l_{ik}^{(m-1)} + w_{kj}\}$ (why?)

Recursive solution to the all-pairs shortest-path problem

- Since shortest path from i to j contains at most m-1 edges,

  $\delta(i,j) = l_{ij}^{(n-1)} = l_{ij}^{(n)} = l_{ij}^{(n+1)} = \ldots$
- Computing the shortest-path weight bottom up:
  - Compute $L^{(1)}, L^{(2)}, \ldots, L^{(n-1)}$ where

    $L^{(m)}=(l_{ij}{}^{(m)})$
  - Note that $L^{(1)} = W$.

EXTENDED-SHORTEST-PATHS(L, W)

- Given matrices $L^{(m-1)}$ and W return $L^{(m)}$

$n \leftarrow L.row$

Let $L' = (l'_{ij})$ be a new n x n matrix

for i = 1 to n

  for j = 1 to n

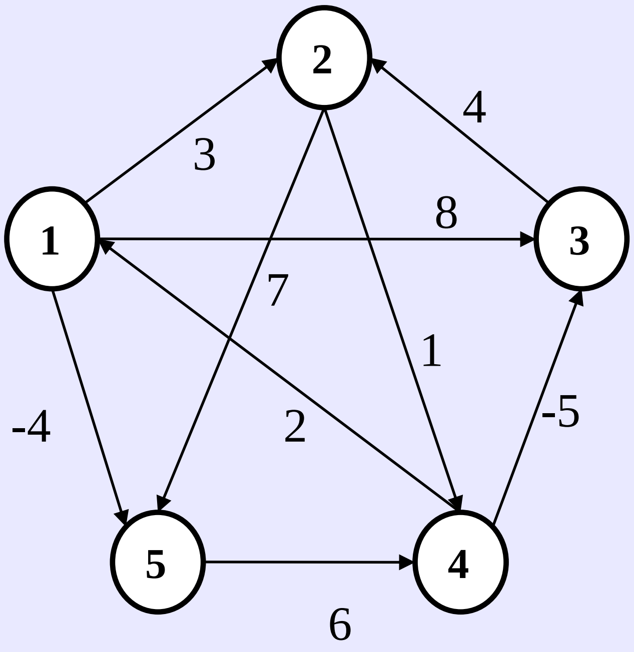
    $l'_{ij} = \infty$

    for k = 1 to n

      $l'_{ij} = \min(l'_{ij}, l_{ik} + w_{kj})$

return L'

Example:



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

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

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

Matrix Multiplication

Let $l^{(m-1)}$ -> a

w -> b

$l^{(m)}$ -> c

min -> $+$

$+$ -> $\cdot$

∞ -> zero

(time complexity : $O(n^3)$)

$$l_{ij}^{(m)} = \min_{1 \leq k \leq n} \{l_{ik}^{(m-1)} + w_{kj}\}$$

$$c_{ij} = \Sigma_{k=1 \text{ to } n} \; a_{ik} \cdot b_{kj}$$

SLOW-ALL-PAIRS-SHORTEST-PATHS(W)

$L^{(1)} = L^{(0)} \cdot W = W$

$L^{(2)} = L^{(1)} \cdot W = W^2$

$L^{(3)} = L^{(2)} \cdot W = W^3$

$\vdots$

$L^{(n-1)} = L^{(n-2)} \cdot W = W^{n-1}$

SLOW-ALL-PAIRS-SHORTEST-PATHS(W)

```
n = W.rows
L^(1) = W
for m = 2 to n-1
        let L^(m) be a new n x n matrix
        L^(m) = EXTENDED-SHORTEST-PATHS( L^(m-1), W )
return L^(n-1)
```

Time complexity : $O(n^4)$

Improving the running time

$L^{(1)} = W$

$L^{(2)} = W^2 = W \cdot W$

$L^{(4)} = W^4 = W^2 \cdot W^2$

$\vdots$

$$L^{(2^{\lceil \log(n-1) \rceil})} = W^{2^{\lceil \log(n-1) \rceil}} = W^{2^{\lceil \log(n-1) \rceil - 1}} \cdot W^{2^{\lceil \log(n-1) \rceil - 1}}$$

i.e., using repeating squaring!

Time complexity: $O(n^3 \lg n)$

FASTER-ALL-PAIRS-SHORTEST-PATHS

FASTER-ALL-PAIRS-SHORTEST-PATHS(W)

1. n = W.row
2. $L^{(1)}$ = W
3. m = 1
4. while m < n-1
5. let $L^{(2m)}$ be a new n x n matrix
6. $L^{(2m)}$ = Extend-Shorest-Path($L^{(m)}$, $L^{(m)}$)
7. m = 2m
8. return $L^{(m)}$

The Floyd-Warshall algorithm

- A different dynamic programming formulation
- **The structure of a shortest path:**

Let $V(G)=\{1,2,\ldots,n\}$. For any pair of vertices $i, j \in V$, consider all paths from i to j whose intermediate vertices are drawn from $\{1, 2,\ldots,k\}$, and let p be a minimum weight path among them.

The structure of a shortest path

- If k is not in p, then all intermediate vertices are in {1, 2,…,k-1}.
- If k is an intermediate vertex of p, then p can be decomposed into $i \overset{p1}{\sim} k \overset{p2}{\sim} j$ where p_1 and is a shortest path from i to k with all the intermediate vertices in {1,2,…,k-1} and p_2 is a shortest path from k to j with all the intermediate vertices in {1,2,…,k-1}.

A recursive solution to the all-pairs shortest path

- Let d_{ij}^{k} =the weight of a shortest path from vertex i to vertex j with all intermediate vertices in the set {1,2,…,k}.

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

$D^{(n)} = (d_{ij}^{(n)})$ if the final solution!

FLOYD-WARSHALL(W)

$n = W.rows$

$D^{(0)} = W$

for $k = 1$ to n

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

    for $i = 1$ to n

        for $j = 1$ to n

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

return $D^{(n)}$

Complexity: $O(n^3)$

Constructing a shortest path

$\Pi^{(0)}, \Pi^{(1)}, \ldots, \Pi^{(n)}$

$\pi_{ij}^{(k)}$:is the predecessor of the vertex j on a shortest path from vertex i with all intermediate vertices in the set {1,2,...,k}.

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

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

Example

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

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

$$\Pi^{(0)} = \begin{pmatrix} N & 1 & 1 & N & 1 \\ N & N & N & 2 & 2 \\ N & 3 & N & N & N \\ 4 & N & 4 & N & N \\ N & N & N & 5 & N \end{pmatrix}$$

$$\Pi^{(1)} = \begin{pmatrix} N & 1 & 1 & N & 1 \\ N & N & N & 2 & 2 \\ N & 3 & N & N & N \\ 4 & 1 & 4 & N & 1 \\ N & N & N & 5 & N \end{pmatrix}$$

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

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

$$\Pi^{(2)} = \begin{pmatrix} N & 1 & 1 & 2 & 1 \\ N & N & N & 2 & 2 \\ N & 3 & N & 2 & 2 \\ 4 & 1 & 4 & N & 1 \\ N & N & N & 5 & N \end{pmatrix}$$

$$\Pi^{(3)} = \begin{pmatrix} N & 1 & 1 & 2 & 1 \\ N & N & N & 2 & 2 \\ N & 3 & N & 2 & 2 \\ 4 & 3 & 4 & N & 1 \\ N & N & N & 5 & N \end{pmatrix}$$

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

$$\Pi^{(4)} = \begin{pmatrix} N & 1 & 4 & 2 & 1 \\ 4 & N & 4 & 2 & 1 \\ 4 & 3 & N & 2 & 1 \\ 4 & 3 & 4 & N & 1 \\ 4 & 3 & 4 & 5 & N \end{pmatrix} \qquad \Pi^{(5)} = \begin{pmatrix} N & 1 & 4 & 5 & 1 \\ 4 & N & 4 & 2 & 1 \\ 4 & 3 & N & 2 & 1 \\ 4 & 3 & 4 & N & 1 \\ 4 & 3 & 4 & 5 & N \end{pmatrix}$$

Transitive closure of a directed graph

Given a directed graph $G = (V, E)$ with $V = \{1,2,\dots, n\}$

The transitive closure of G is $G^* = (V, E^*)$ where

$E^* = \{(i, j) \mid$ there is a path from i to j in $G\}$.

Modify FLOYD-WARSHALL algorithm:

$t_{ij}^{(0)} = \;\; 0$ if $i \neq j$ and $(i,j) \notin E$

$\qquad\quad 1$ if $i = j$ or $(i,j) \in E$

for $k \geq 1$

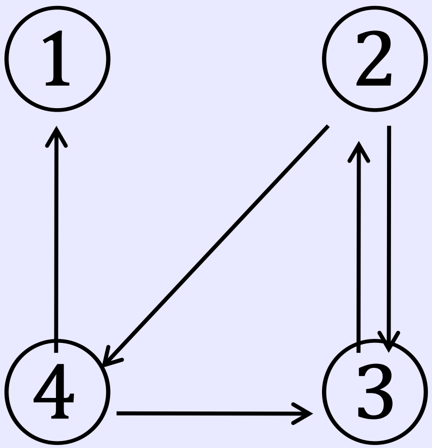
$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$

TRANSITIVE-CLOSURE(G)

$n = |G.V|$
Let $T^{(0)} = (t_{ij}^{(0)})$ be a new $n \times n$ matrix
for $i = 1$ to n
  for $j = 1$ to n
    if $i == j$ or $(i, j) \in G.E$
      $t_{ij}^{(0)} = 1$
    else $t_{ij}^{(0)} = 0$
for $k = 1$ to n
  Let $T^{(k)} = (t_{ij}^{(k)})$ be a new $n \times n$ matrix
  for $i = 1$ to n
    for $j = 1$ to n
      $t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$
return $T^{(n)}$

Time complexity: $O(n^3)$

Example



$$T^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$T^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$T^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$T^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$T^{(4)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Johnson's algorithm for sparse graphs

- If all edge weights in G are nonnegative, we can find all shortest paths in $O(V^2 \lg V + VE)$ by using Dijkstra's algorithm with Fibonacci heap
- Combine with Bellman-Ford algorithm takes $O(VE)$
- using reweighting technique

Reweighting technique

- If G has negative-weighted edge, we compute a new set of nonnegative weight that allows us to use the same method. The new set of edge weight $\hat{w}$ satisfies:
- 1. For all pairs of vertices $u, v \in V$, a shortest path from u to v using weight function w is also a shortest path from u to v using the weight function $\hat{w}$
- 2. $\forall (u,v) \in E(G)$, $\hat{w}(u,v)$ is nonnegative

Lemma: (Reweighting doesn't change shortest paths)

- Given a weighted directed graph $G = (V, E)$ with weight function $w:E \rightarrow R$, let $h:V \rightarrow R$ be any function mapping vertices to real numbers. For each edge $(u,v) \in E$, $\hat{w}(u,v) = w(u,v) + h(u) - h(v)$
- Let $P=\langle v_0,v_1,\dots,v_k\rangle$ be a path from vertex v_0 to v_k Then $w(P) = \delta(v_0,v_k)$ if and only if $\hat{w}(P) = \hat{\delta}(v_0,v_k)$ Also, G has a negative-weight cycle using weight function w iff G has a negative weight cycle using weight function $\hat{w}$.

Proof

- $\hat{w}(P) = w(P) + h(v_0) - h(v_k)$

$$\hat{w}(P) = \sum_{i=1}^{k} \hat{w}(v_{i-1}, v_i)$$

$$= \sum_{i=1}^{k} (w(v_{i-1}, v_i) + h(v_{i-1}) - h(v_i))$$

$$= \sum_{i=1}^{k} w(v_{i-1}, v_i) + h(v_0) - h(v_k)$$

$$= w(p) + h(v_0) - h(v_k)$$

Proof

- $w(p) = \delta(v_0,v_k)$ implies $\hat{w}(P) = \hat{\delta}(v_0,v_k)$

- Suppose there is a shorter path P' from v_0 to v_k using the weight function $\hat{w}$ ➔ $\hat{w}(P') < \hat{w}(P)$
- Then $w(P') + h(v_0) - h(v_k) = \hat{w}(P')$
  $< \hat{w}(P) = w(P) + h(v_0) - h(v_k)$
  ➔ $w(P') < w(P)$.
- We get a contradiction!

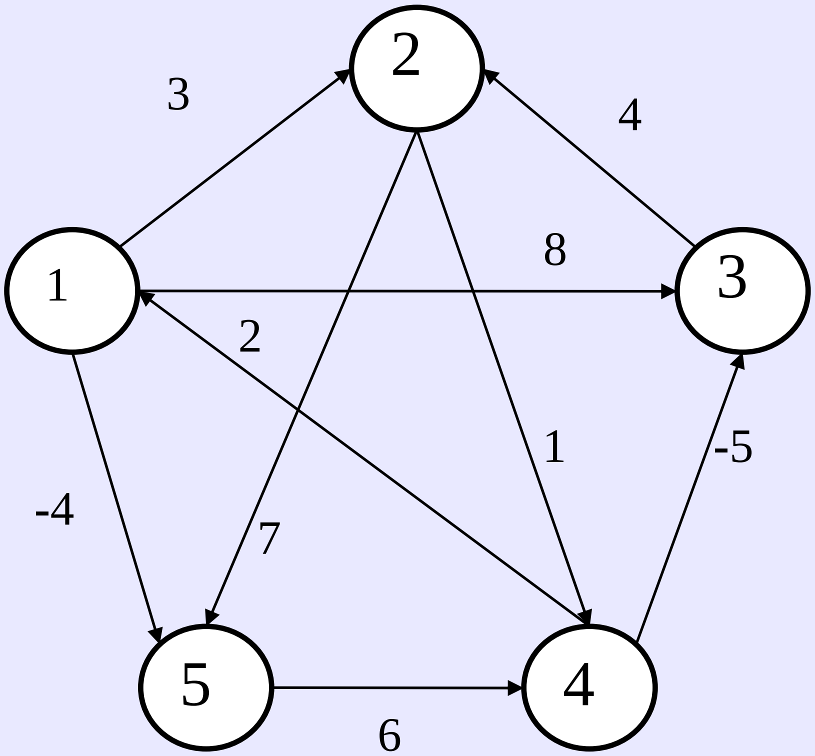
Proof

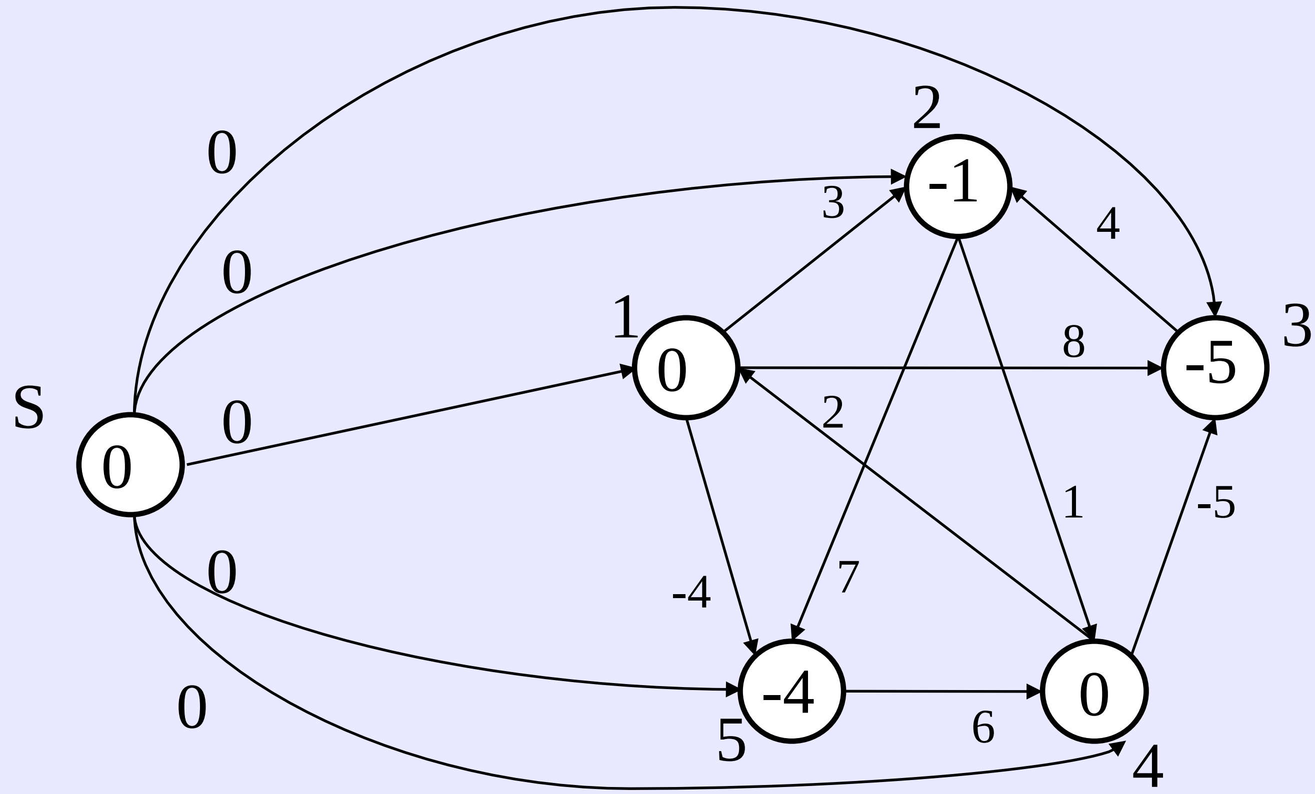
- G has a negative-weight cycle using w iff G has a negative-weight cycle using $\hat{w}$.
- Consider any cycle $C=\langle v_0,v_1,\ldots,v_k\rangle$ with $v_0=v_k$. Then $\hat{w}(C) = w(C) + h(v_0) - h(v_k) = w(C)$.
- Producing nonnegative weight by reweight

Producing nonnegative weight by reweighting

- Given a weighted directed graph $G = (V, E)$
- We make a new graph $G' = (V', E')$, $V' = V \cup \{s\}$, $E' = E \cup \{(s,v): v \in V\}$ and $w(s,v) = 0$, for all v in V
- Let $h(v) = \delta(s, v)$ for all $v \in V'$
- We have $h(v) \leq h(u) + w(u, v)$ (why?)
- $\hat{w}(u, v) = w(u, v) + h(u) - h(v) \geq 0$

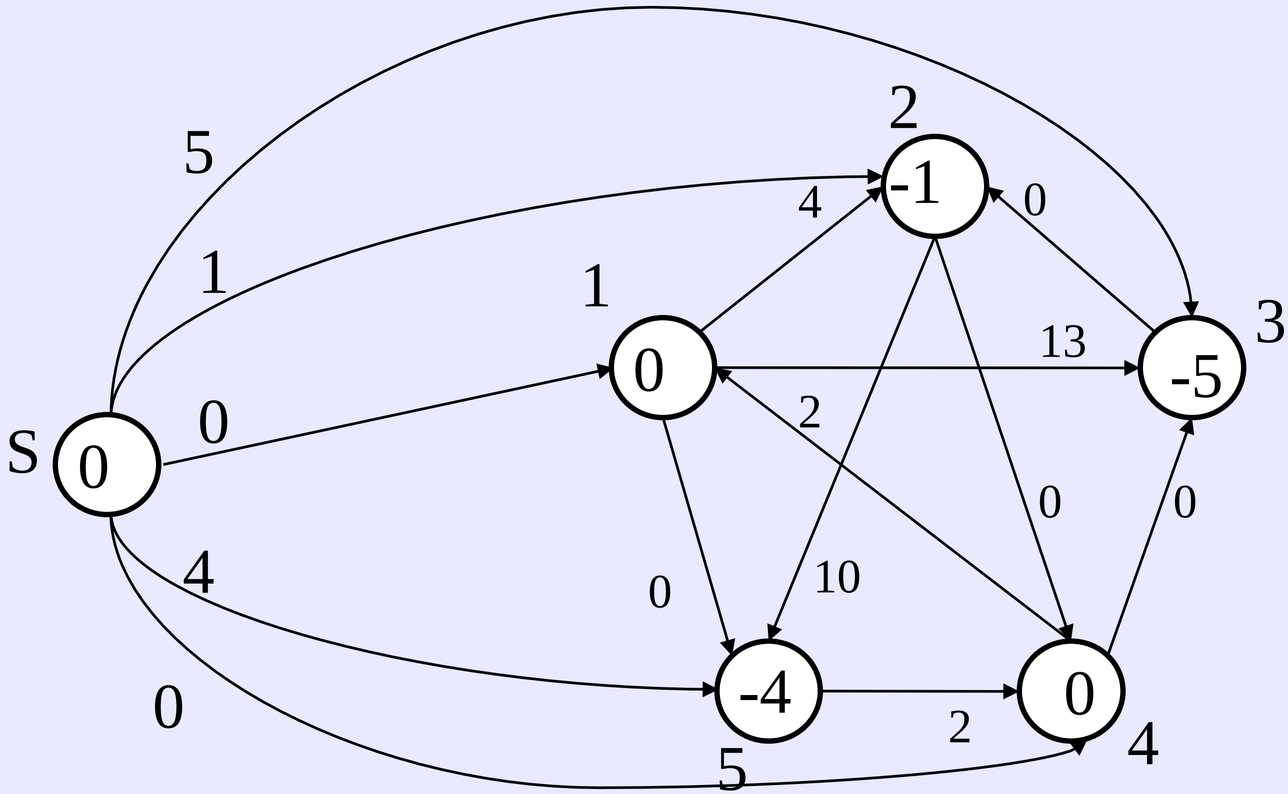
Example:




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

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

$$\hat{w}(u, v) = w(u, v) + h(u) - h(v)$$



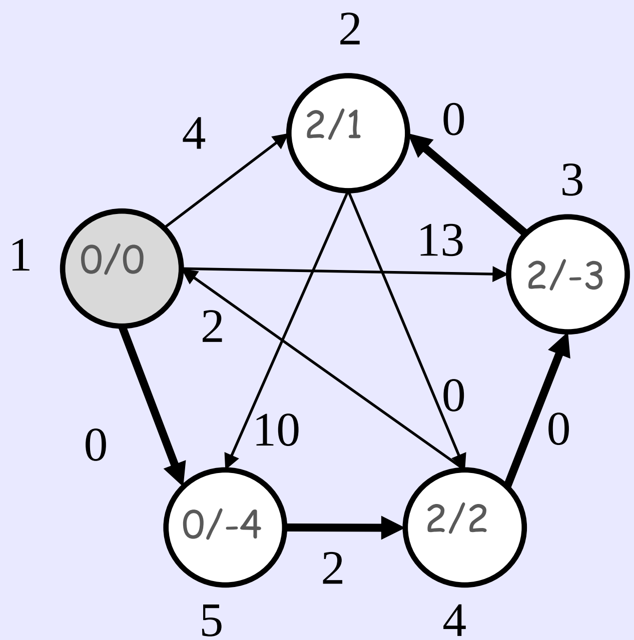
JOHNSON algorithm

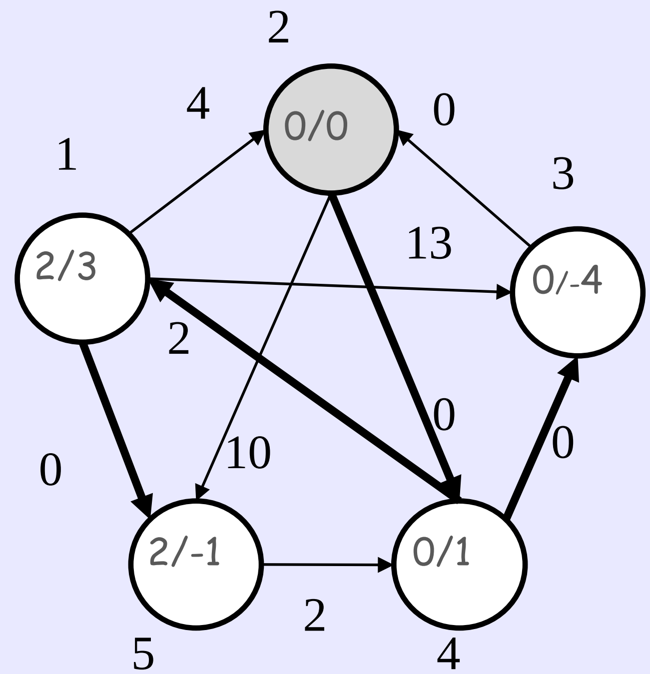
1 Computing G', where $G'.V = G.V \cup$ **{s}**
and $G'.E = G.E \cup \{(s, v): v \in G.V\}$ and $w(s, v) = 0$.
2 if BELLMAN-FORD(G', w, s)= FALSE
3 print "the input graph contains negative weight cycle"
4 else for each vertex $v \in G'.V$
5 set $h(v)$ to be the value of $\delta(s, v)$ computed by the BF algorithm
6 for each edge $(u, v) \in G'.E$

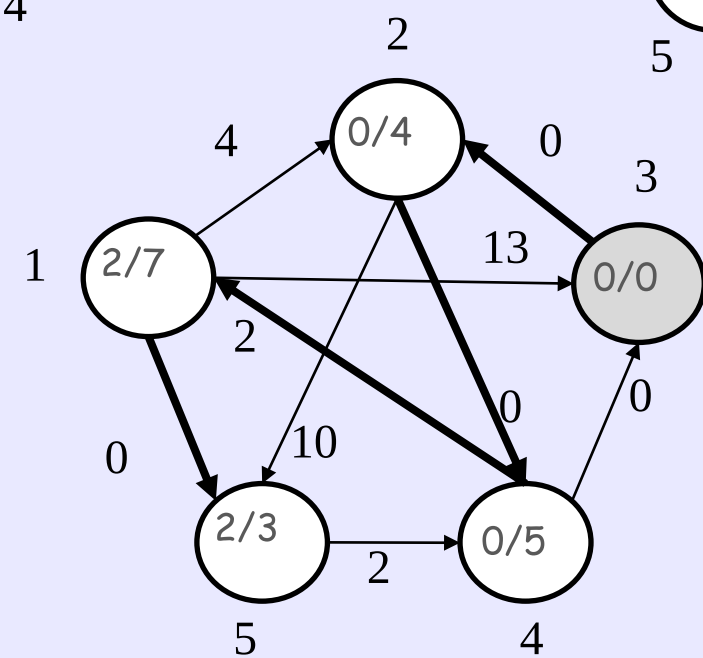
JOHNSON algorithm

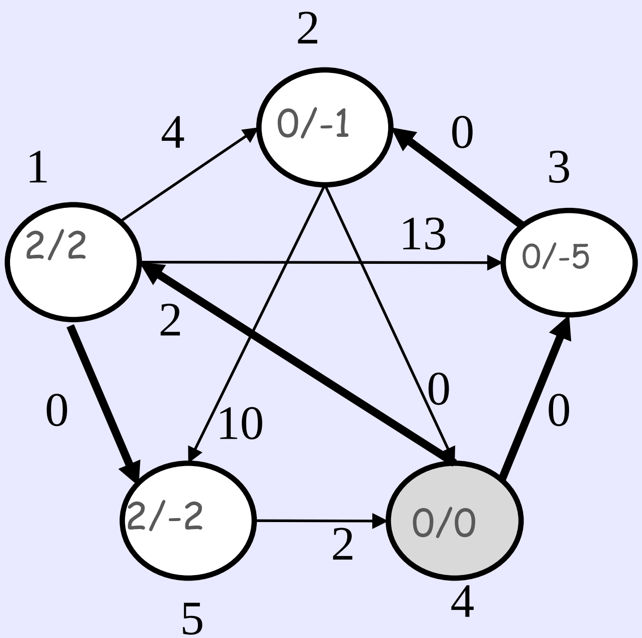
```
ŵ(u, v) = w(u, v) + h(u) – h(v)
Let D = (d_uv) be  a new n x n matrix
for each vertex u ∈ G.V
    run DIJKSTRA (G, ŵ, u) to compute δ̂ (u, v)
     for all v ∈ V[G] .
for each vertex v ∈ G.V
        d_uv = δ̂(u, v) + h(v) – h(u)
return D
```

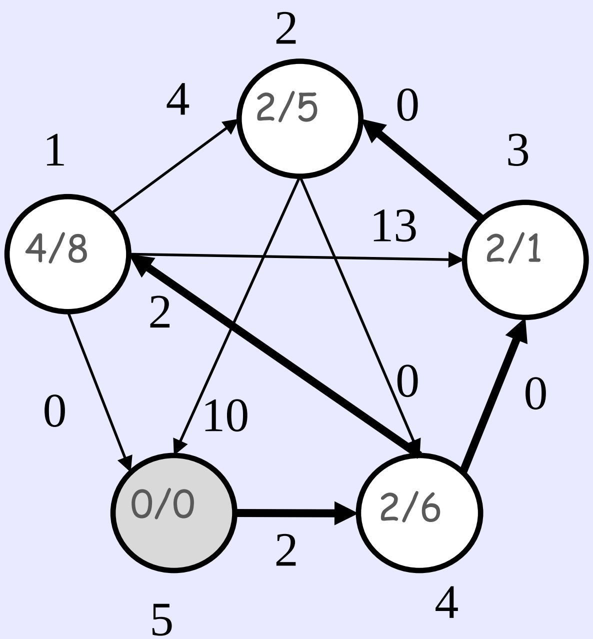
Complexity: $O(V^2 \lg V + VE)$


2
4
2/1
0
1
0/0
13
3
2/-3
2
0
10
0
0
0/-4
2/2
2
5
4


2
4
0/0
0
1
2/3
13
3
0/-4
2
0
0
10
0
2/-1
0/1
2
5
4


2
4
0/4
0
3
1
2/7
13
0/0
2
0
0
0
10
2/3
0/5
2
5
4


2
1
4
0/-1
0
3
2/2
13
0/-5
2
0
0
10
0
2/-2
0/0
2
5
4


2
4
1
2/5
0
3
13
4/8
2/1
2
0
0
10
0
0/0
2/6
2
5
4

Homework

- Exercises: 25.1-5, 25.1-9, 25.1-10
- Exercises: 25.2-2, 25.2-6
- Exercises: 25.2-8 (Due: Dec. 31)
- Exercises: 25.3-4
- Exercises: 25.3-6 (Due: Dec. 31)