Learn DAA : From B K Sharma

# TCS-503: Design and Analysis of Algorithms

## Graph Algorithms
## All Pairs Shortest Path Problems

Learn DAA : From B K Sharma

# Unit IV

- Graph Algorithms:
  - Elementary Graphs algorithms: BFS and DFS
  - Minimum Spanning Trees
  - Single-Source Shortest Paths
  - All-Pairs Shortest Paths
  - Maximum Flow and
  - Traveling Salesman Problem

Learn DAA : From B K Sharma

## APSP : The Floyd- Warshall Algorithm

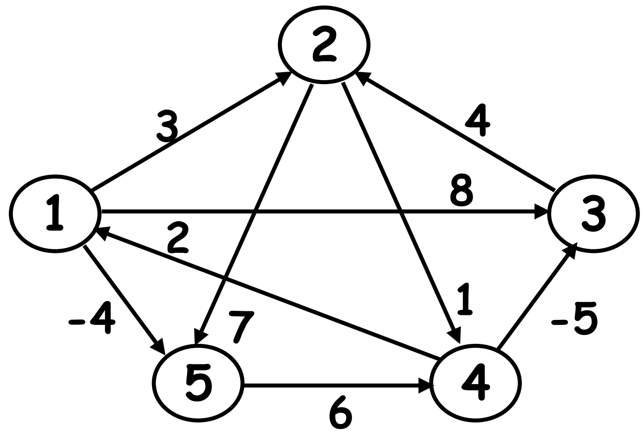
### Dynamic Programming

Vertices numbered from 1 to $n=|V|$

Adjacency Matrix of Weight of G

$$W=w_{ij} = \begin{cases} 0 & \text{if } i = j \\ \text{weight of } (i,\ j) & \text{if } i \neq j\ ,\ (i,\ j) \in E \\ \infty & \text{if } i \neq j\ ,\ (i,\ j) \notin E \end{cases}$$

$W=w_{ij}$ is an n x n matrix.



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

Learn DAA : From B K Sharma

## APSP : The Floyd- Warshall Algorithm

### Dynamic Programming

#### Recursive Solution

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

Where, $k=0, 1, \ldots, n$

$d_{ij}^{(k)}$ is the shortest path weight from *i* to *j* with intermediate vertices (excluding *i*, *j*) $\{1, 2, \ldots, k\}$ from the set

Intermediate vertex of a simple path $p = \langle v_1, v_2, \ldots, v_l \rangle$ is any vertex of *p* other than $v_1$ or $v_l$.

There are two cases:

Case 1: $k=0$, No Intermediate vertex at all, Base Case

Case 2: $k \geq 1$

# Learn DAA : From B K Sharma

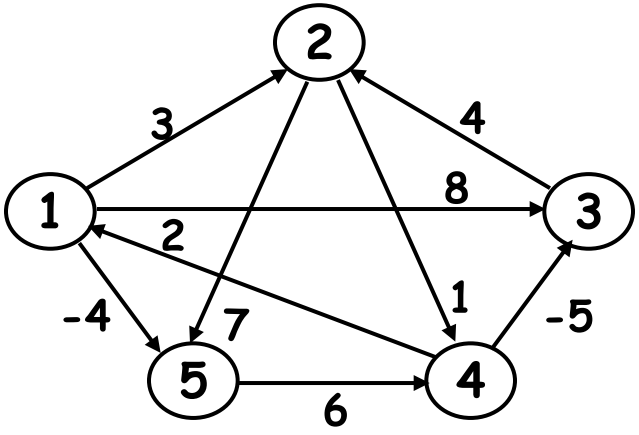
## APSP : The Floyd- Warshall Algorithm

### Dynamic Programming

#### Recursive Solution

**Case 1:** k=0, No Intermediate vertex at all, Base Case

$$d_{ij}^{(0)} = w_{ij}$$




$W=w_{ij}$

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

Learn DAA : From B K Sharma

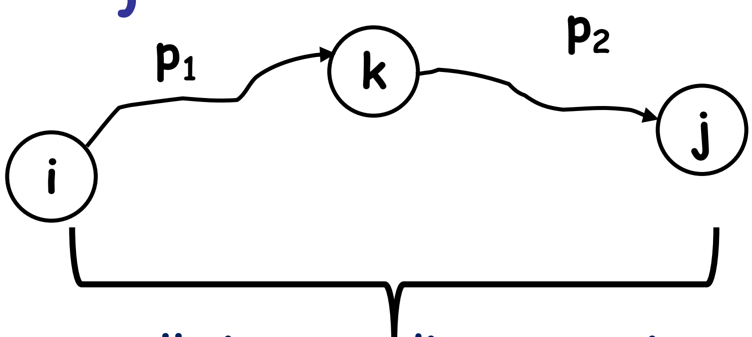
## APSP : The Floyd- Warshall Algorithm

### Dynamic Programming

#### Recursive Solution

**Case 2:** $k \geq 1$

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



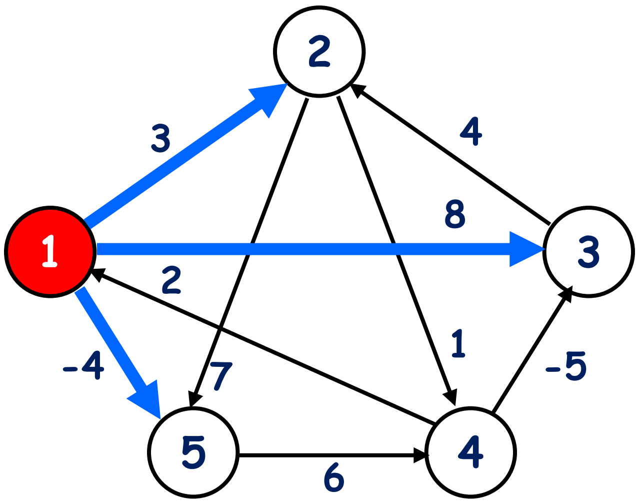

p: all intermediate vertices in [1,2.3,....k]

Learn DAA : From B K Sharma

## APSP : The Floyd- Warshall Algorithm

### Dynamic Programming

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




$D^{(0)} = W_{ij}$

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

For k=1,

$d_{11}^{(1)} = \min\{d_{11}^{(0)},\ d_{11}^{(0)} + d_{11}^{(0)}\} = \min\{0,\ 0 + 0\} = 0$

$d_{42}^{(1)} = \min\{d_{42}^{(0)},\ d_{41}^{(0)} + d_{12}^{(0)}\} = \min\{\infty,\ 2 + 3\} = 5$

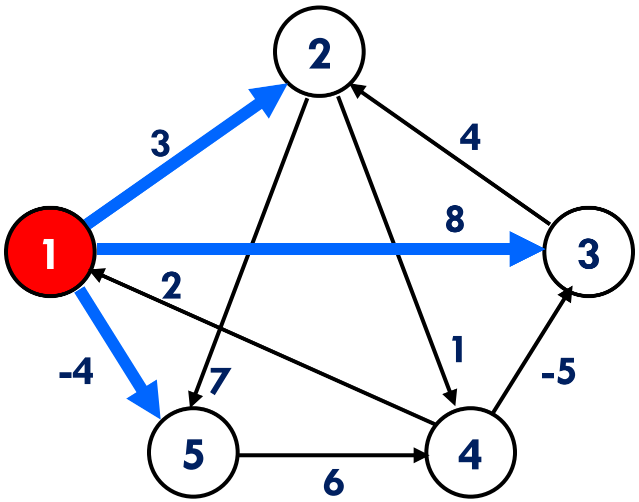
$d_{45}^{(1)} = \min\{d_{45}^{(0)},\ d_{41}^{(0)} + d_{15}^{(0)}\} = \min\{\infty,\ 2 + (-4)\} = -2$

Learn DAA : From B K Sharma

## APSP : The Floyd- Warshall Algorithm

### Dynamic Programming

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




$D^{(1)}=d_{ij}^{(1)}$

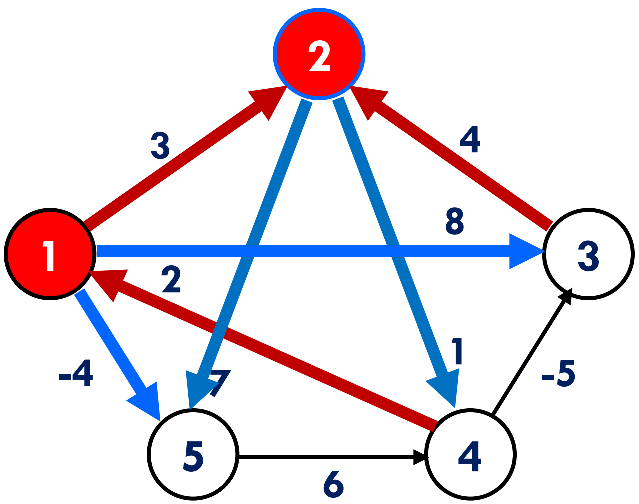
| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 3 | 8 | ∞ | -4 |
| 2 | ∞ | 0 | ∞ | 1 | 7 |
| 3 | ∞ | 4 | 0 | ∞ | ∞ |
| 4 | 2 | 5 | -5 | 0 | -2 |
| 5 | ∞ | ∞ | ∞ | 6 | 0 |

Learn DAA : From B K Sharma

## APSP : The Floyd- Warshall Algorithm

### Dynamic Programming

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




$D^{(1)}=d_{ij}^{(1)}$

| | 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 |

For k=2,

$d_{15}^{(2)}=\min[d_{15}^{(1)},d_{12}^{(1)} +d_{25}^{(1)}]=\min[-4,3 +7]=\min[-4, 10]=-4$

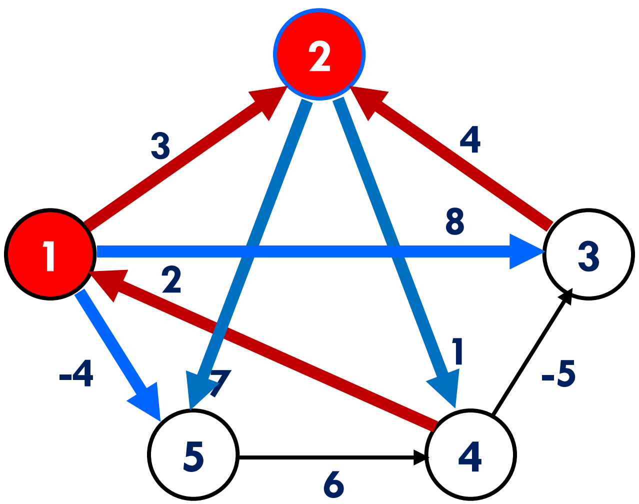
$d_{35}^{(2)} =\min[d_{35}^{(1)}, d_{32}^{(1)} +d_{25}^{(1)}]=\min [\infty, 4 +7]=\min[\infty, 11]=11$

# Learn DAA : From B K Sharma

## APSP : The Floyd- Warshall Algorithm

### Dynamic Programming

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




$D^{(2)}=d_{ij}^{(2)}$

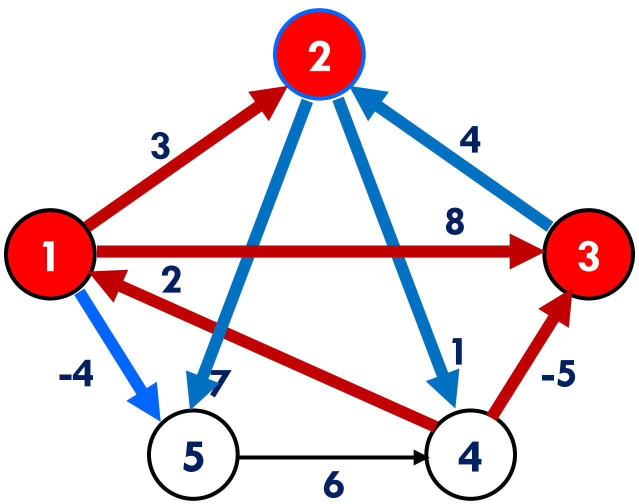
| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 3 | 8 | 4 | -4 |
| 2 | ∞ | 0 | ∞ | 1 | 7 |
| 3 | ∞ | 4 | 0 | 5 | 11 |
| 4 | 2 | 5 | -5 | 0 | -2 |
| 5 | ∞ | ∞ | ∞ | 6 | 0 |

Learn DAA : From B K Sharma

## APSP : The Floyd- Warshall Algorithm

### Dynamic Programming

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




$D^{(2)}=d_{ij}^{(2)}$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 3 | 8 | 4 | -4 |
| 2 | ∞ | 0 | ∞ | 1 | 7 |
| 3 | ∞ | 4 | 0 | 5 | 11 |
| 4 | 2 | 5 | -5 | 0 | -2 |
| 5 | ∞ | ∞ | ∞ | 6 | 0 |

For k=3,

$d_{42}^{(3)}=\min[d_{42}^{(2)},d_{43}^{(2)} +d_{32}^{(2)}]=\min[5,-5 +4]=\min[5, -1]=-1$

$d_{43}^{(3)} =\min[d_{43}^{(2)}, d_{43}^{(2)} +d_{33}^{(2)}]=\min [-5, -5 +0]=\min[-5,-5]=-5$

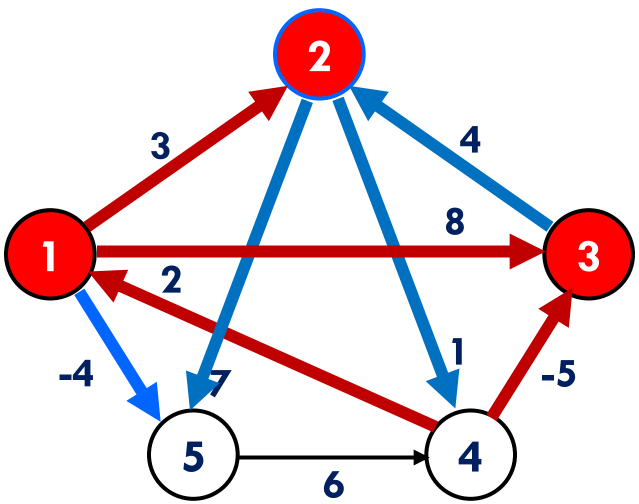
$d_{45}^{(3)} =\min[d_{45}^{(3)}, d_{43}^{(2)} +d_{35}^{(2)}]=\min [-2, -5 +11]=\min[-2, 6]=-2$

Learn DAA : From B K Sharma

# APSP : The Floyd- Warshall Algorithm

## Dynamic Programming

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

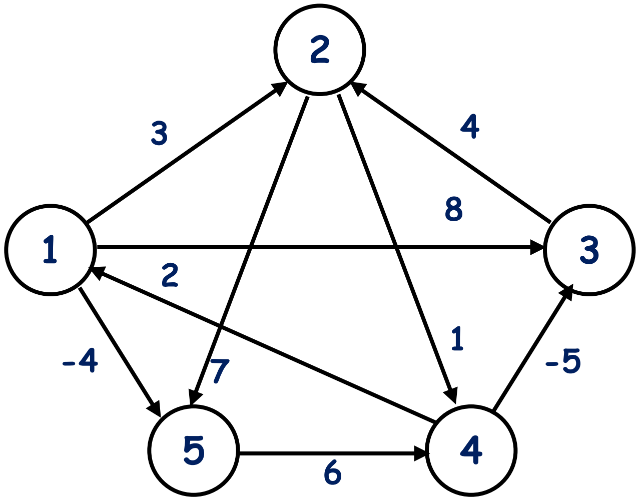



$D^{(3)}=d_{ij}^{(3)}$

| | | | | |
|---|---|---|---|---|
| 0 | 3 | 8 | 4 | -4 |
| ∞ | 0 | ∞ | 1 | 7 |
| ∞ | 4 | 0 | 5 | 11 |
| 2 | -1 | -5 | 0 | -2 |
| ∞ | ∞ | ∞ | 6 | 0 |

# Learn DAA : From B K Sharma

## APSP : The Floyd- Warshall Algorithm




$D^{(0)} = d_{ij}(0) = W_{ij}$

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

$D^{(1)} = d_{ij}^{(1)}$

| | 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 |

$D^{(2)} = d_{ij}^{(2)}$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 3 | 8 | 4 | -4 |
| 2 | ∞ | 0 | ∞ | 1 | 7 |
| 3 | ∞ | 4 | 0 | 5 | 11 |
| 4 | 2 | 5 | -5 | 0 | -2 |
| 5 | ∞ | ∞ | ∞ | 6 | 0 |

$D^{(3)} = d_{ij}^{(3)}$

| | | | | |
|---|---|---|---|---|
| 0 | 3 | 8 | 4 | -4 |
| ∞ | 0 | ∞ | 1 | 7 |
| ∞ | 4 | 0 | 5 | 11 |
| 2 | -1 | -5 | 0 | -2 |
| ∞ | ∞ | ∞ | 6 | 0 |

$D^{(4)} = d_{ij}^{(4)}$

| | | | | |
|---|---|---|---|---|
| 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 |

------

Learn DAA : From B K Sharma

## APSP : The Floyd- Warshall Algorithm

### Dynamic Programming

**Alg.:** FLOYD-WARSHALL(W)

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

## *APSP:-Johnson's algorithm*

**Improvement for sparse graphs with reweighting technique:**

**Definition (Sparse Graph):**

A sparse graph is a graph $G = (V, E)$ in which $|E| = O(|V|)$.

**Definition (Dense Graph):**

A dense graph is a graph $G = (V, E)$ in which $|E| = \Theta(|V|^2)$.

Learn DAA : From B K Sharma

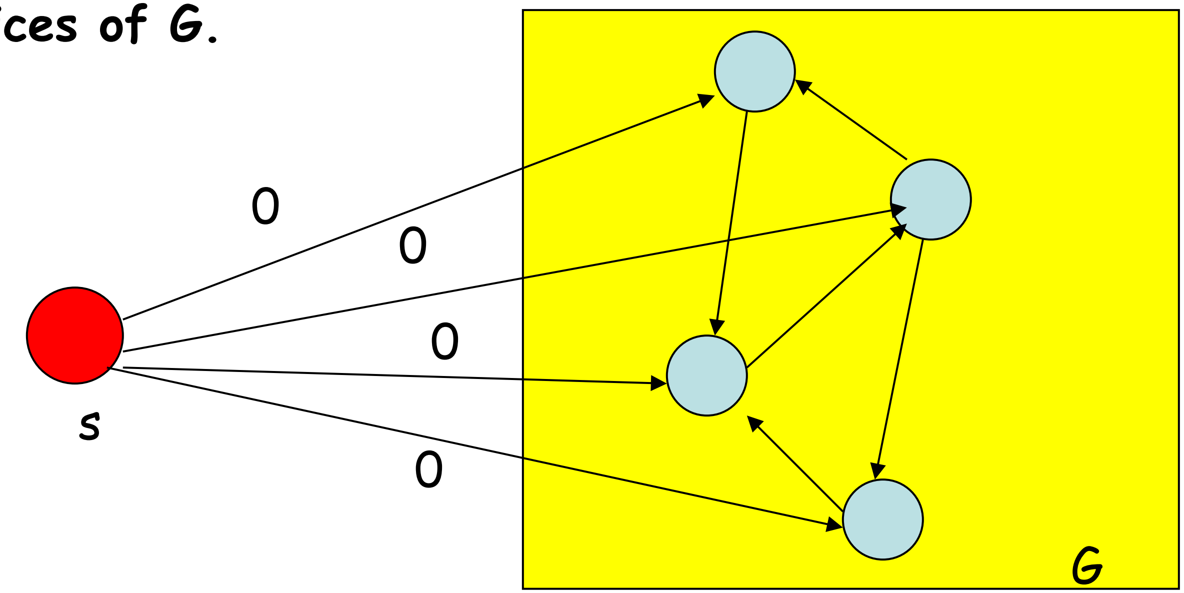
## *APSP:-Johnson's algorithm*

Uses both Bellman-Ford and Dijkstra as subroutines.

**Algorithm:**

Let the given graph be $G$.

**Step 1:** Add a new vertex s to the graph, add edges from new vertex to all vertices of $G$.




Let the modified graph be $G'$.

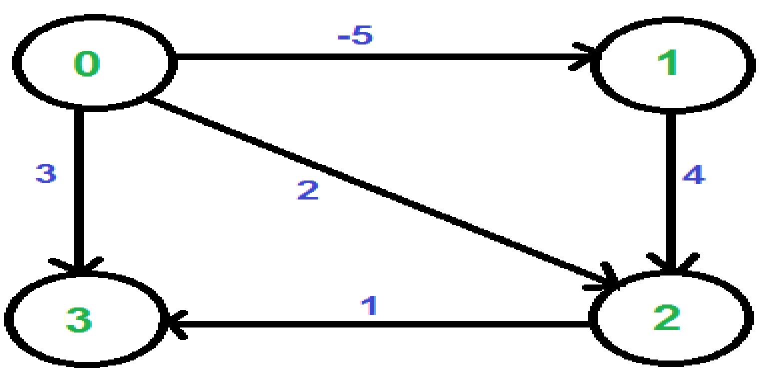
Learn DAA : From B K Sharma

*APSP:-Johnson's algorithm*

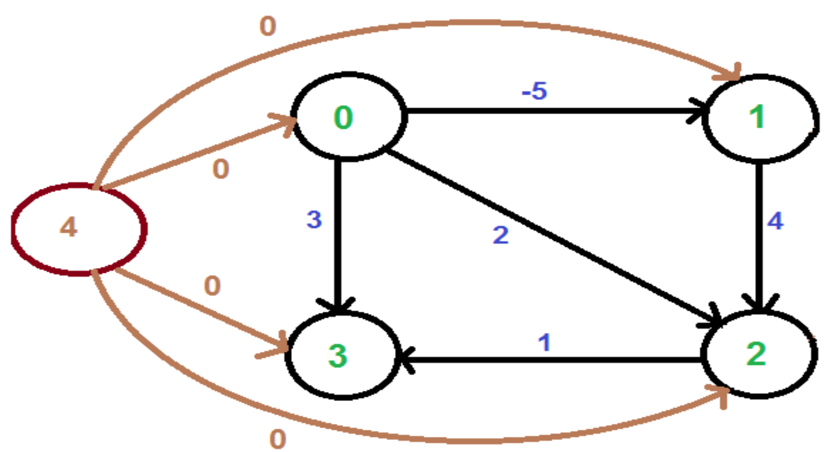
Step 2: Run Bellman-Ford algorithm on G' with s as source:

Let the distances calculated by Bellman-Ford be h[0], h[1], .. h[V-1].

If we find a negative weight cycle, then return.




Graph G




Graph G'

The shortest distances from 4 to 0, 1, 2 and 3 calculated by Bellman-Ford Algorithm

h[]={h[0], h[1], .. h[V-1]}.

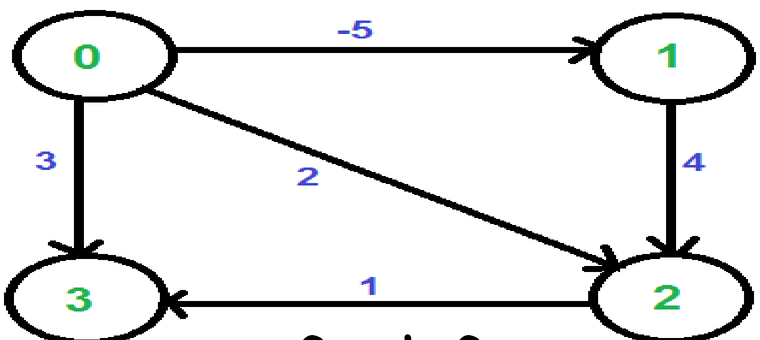
h[] = {0, -5, -1, 0}.

Learn DAA : From B K Sharma

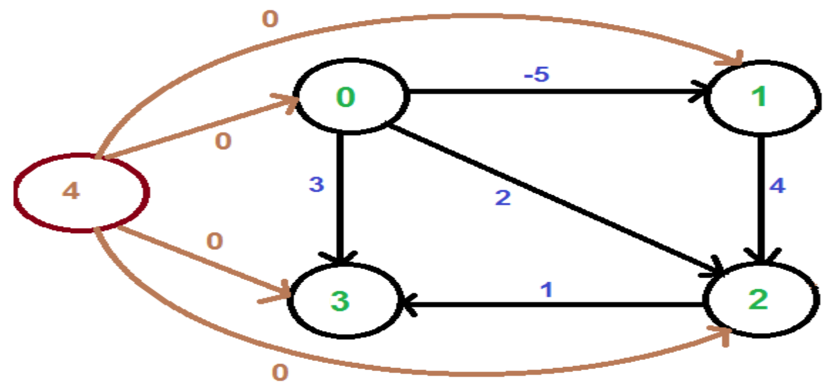
## *APSP:-Johnson's algorithm*

**Step 3:** Reweight the edges of original graph.

For each edge (u, v), assign the new weight as "original weight + h[u] – h[v]".




Graph G



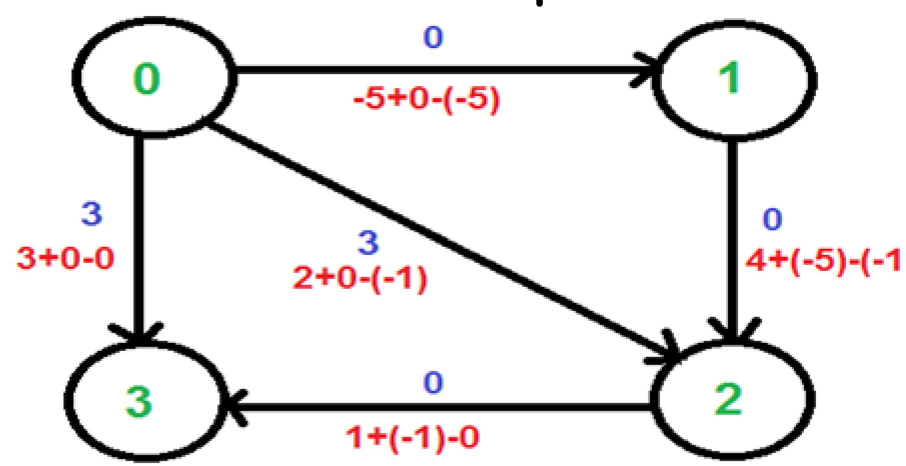

Graph G'

h[] = {h[0], h[1], h[2], h[3]}

h[] = { 0, -5, -1, 0 }.

Re-weight the edges:

w(u, v) = w(u, v) + h[u] - h[v].

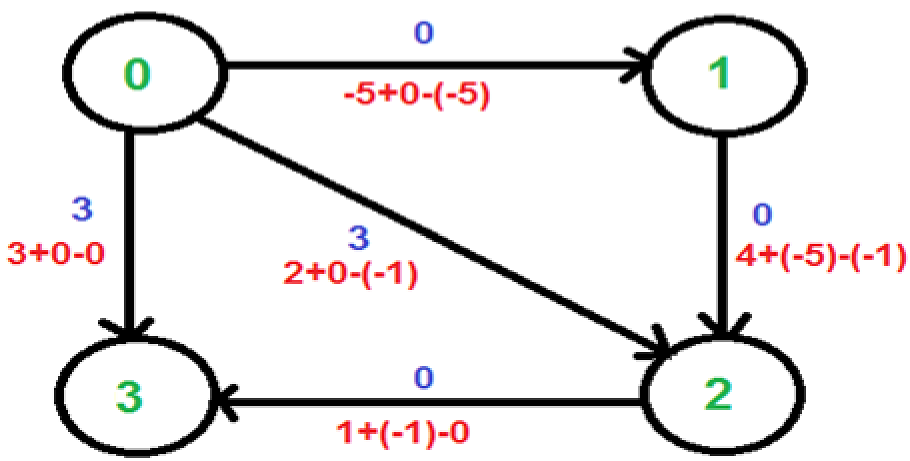


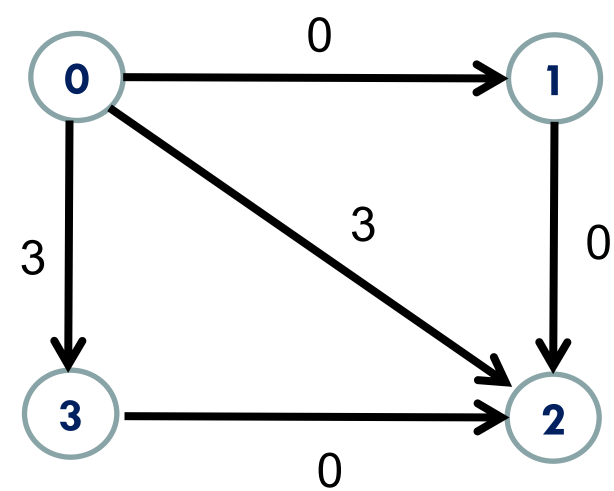
Learn DAA : From B K Sharma

*APSP:-Johnson's algorithm*

**Step 4:** Remove the added vertex s and run Dijkstra's algorithm for every vertex.

Since all weights are positive now, we can run Dijkstra's shortest path algorithm for every vertex as source.



Learn DAA : From B K Sharma

## APSP:-Johnson's algorithm

**Step 5:** Compute the actual distances by subtracting h[v]-h[u]

$$\delta(u,v)=\delta'(u,v)-h(u)+h(v)$$

$$w(p)=w'(p) - h(u) + h(v)$$

## *APSP:-Johnson's algorithm*

Time complexity of Floyd Warshall Algorithm is $\Theta(V^3)$. *Using Johnson's algorithm, we can find all pair shortest paths in $O(V^2 \log V + VE)$ time.*

## APSP:-Johnson's algorithm

**Alg.:** Johnson($G$)

1. compute $G'$, where $V[G'] = V[G] \cup \{s\}$ and $E[G'] = E[G] \cup \{(s,v) : v \in V[G]\}$
2. if Bellman-Ford($G'$,w,s)=False
3.     then print "∃ a neg-weight cycle"
4. else for each vertex $v \in V[G']$
5.         set $h(v) = \delta(s,v)$ computed by Bellman-Ford algo.
6.     for each edge $(u,v) \in E[G']$
7.         $w'(u,v) = w(u,v) + h(u) - h(v)$ "original weight + h[u] – h[v]"
8. Let $D = (d_{uv})$ be a new n x n matrix
9.     for each vertex $u \in V[G]$
10.         run Dijkstra($G$,w',u) to compute $\delta'(u,v)$
11.     for each $v \in V[G]$
12.         $d_{uv} = \delta'(u,v) - h(u) + h(v)$
13. return D

# Single-Source Shortest Path Problem

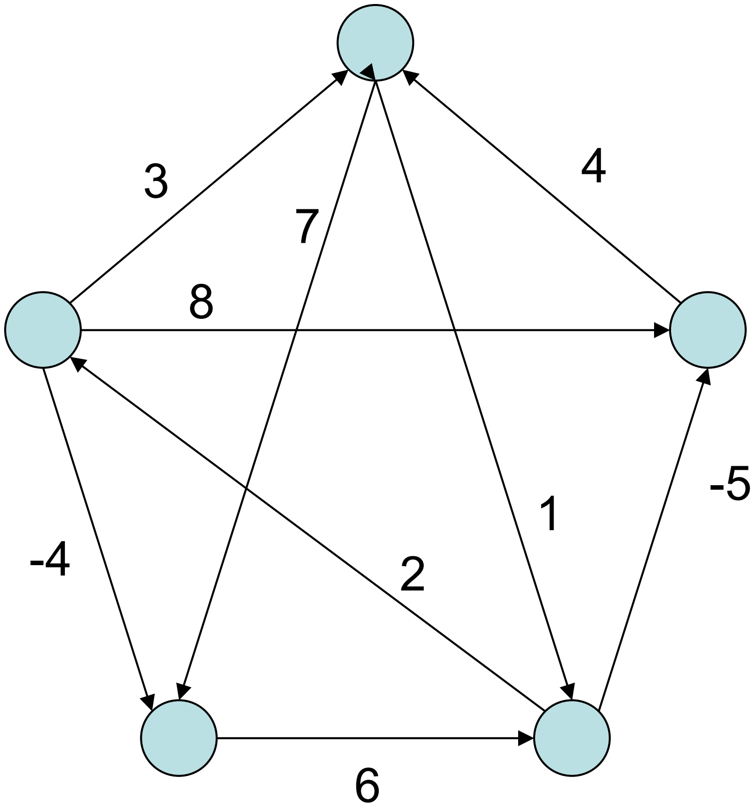
## Dijkstra's Algorithm

**Alg.: DIJKSTRA(G, w,s)**

```
INITIALIZE-SINGLE-SOURCE(G,s)
S ← ∅
Q ← V[G]
While Q ≠ ∅ do
        u ← EXTRACT-MIN(Q)
        S ← S U {u}
        For each v ∈ Adj[u] do
                RELAX(u, v, w)
```

Learn DAA : From B K Sharma

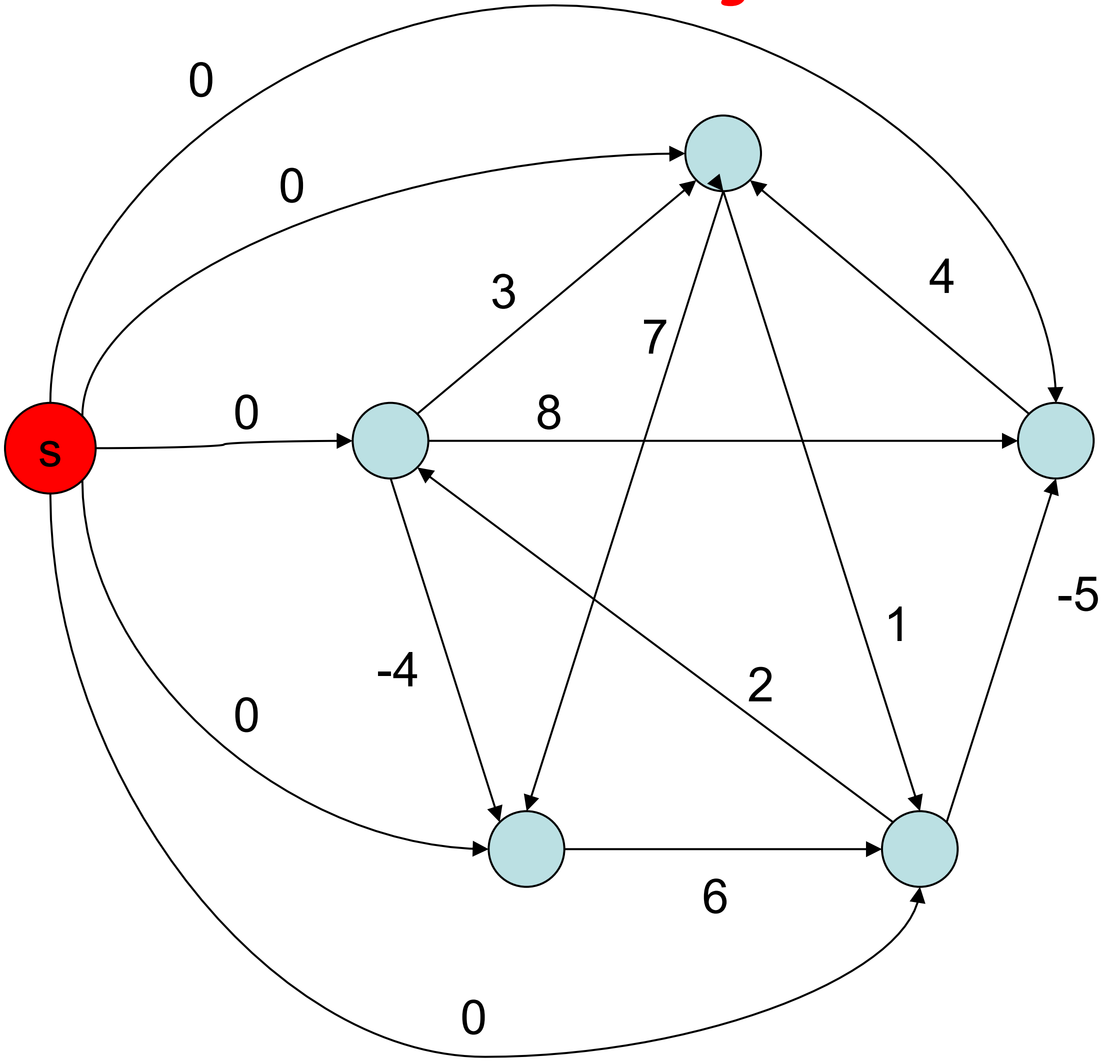
## APSP:-Johnson's algorithm: Another Example




**Graph G**

Learn DAA : From B K Sharma

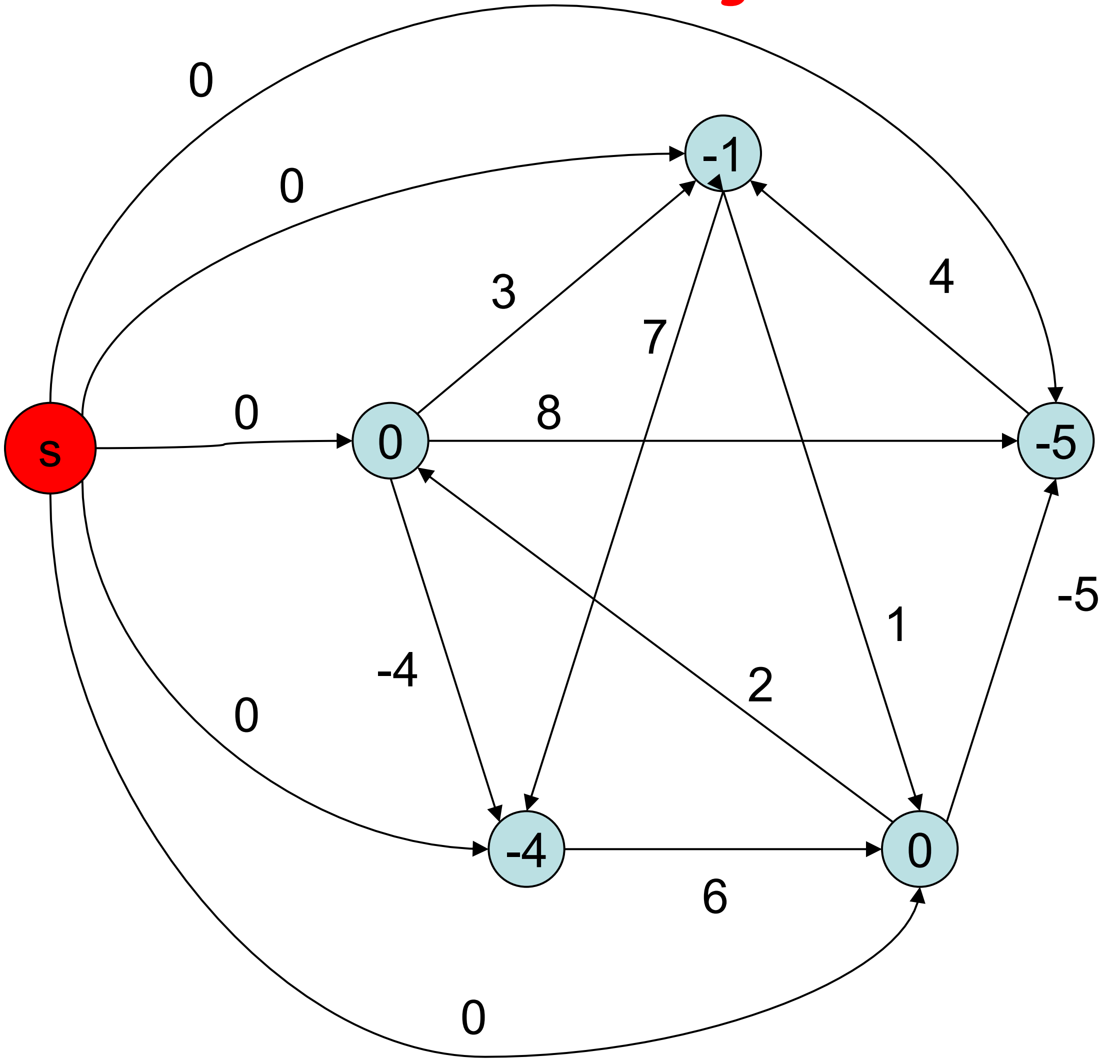
## APSP:-Johnson's algorithm: Another Example




**Graph G'**

Learn DAA : From B K Sharma

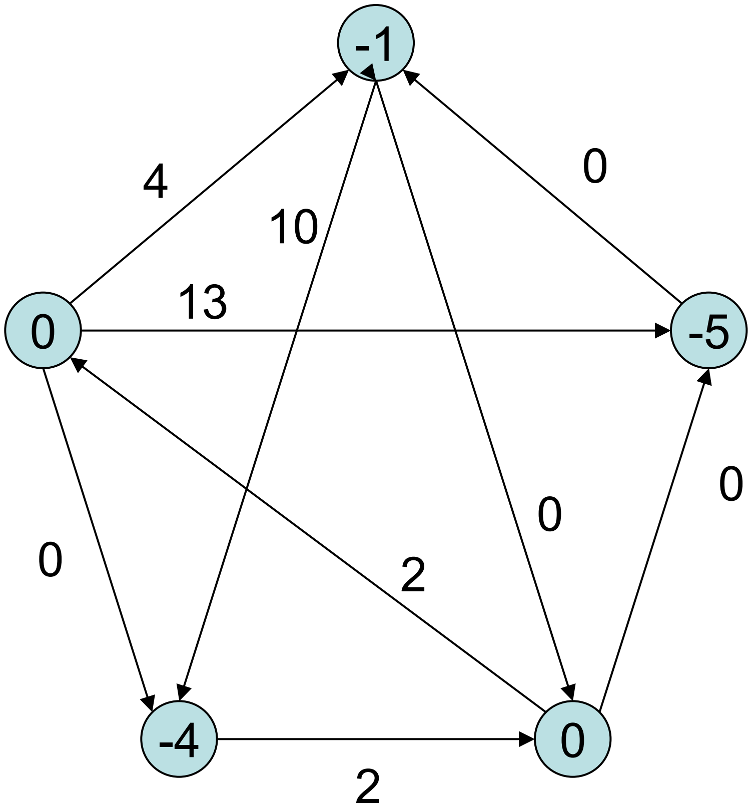
## APSP:-Johnson's algorithm: Another Example




After reweighting each edge.

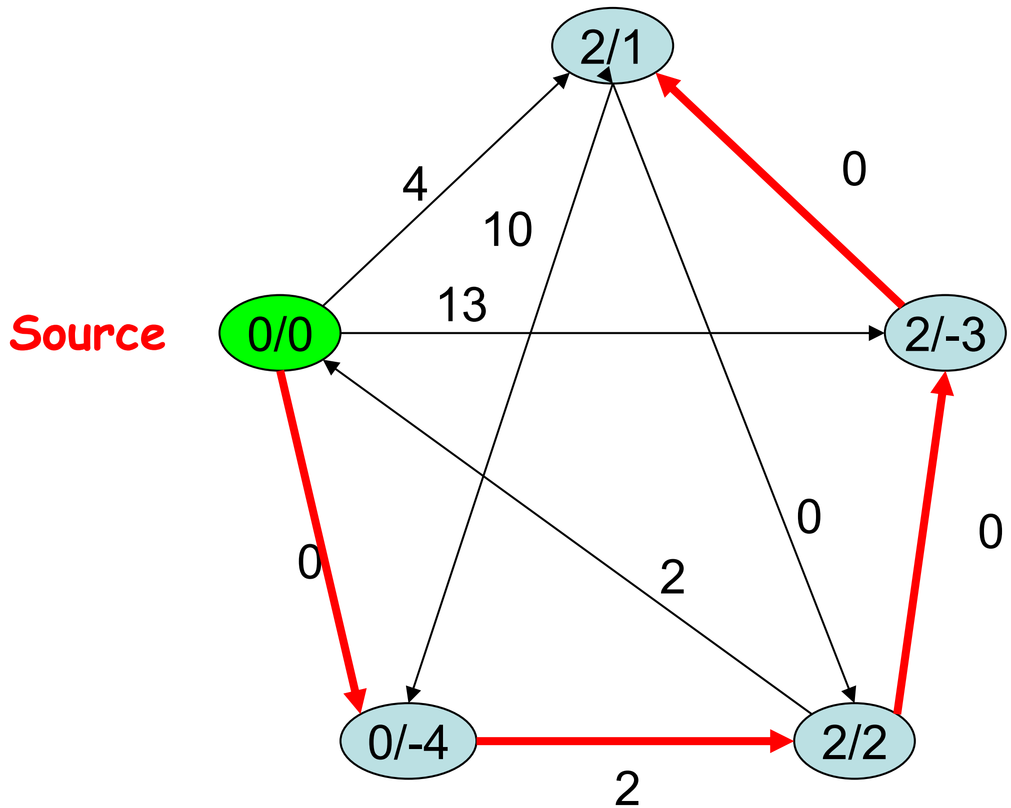
Learn DAA : From B K Sharma

## APSP:-Johnson's algorithm: Another Example



Learn DAA : From B K Sharma

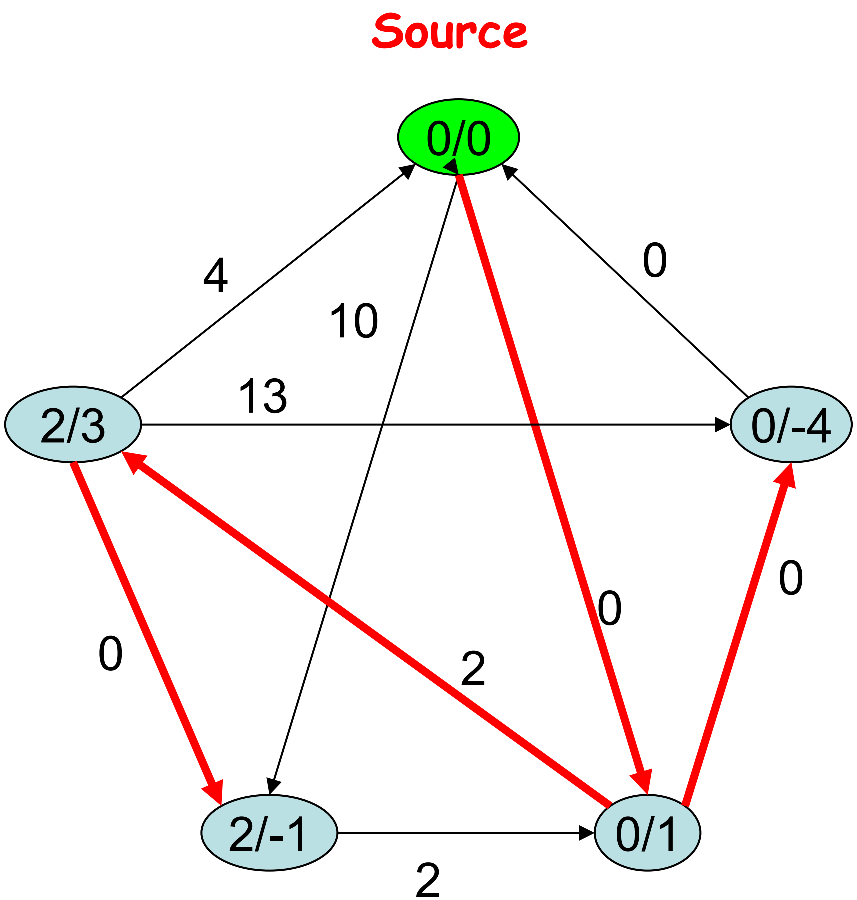
## APSP:-Johnson's algorithm: Another Example




Result of Running Dijkstra' s Algorithm

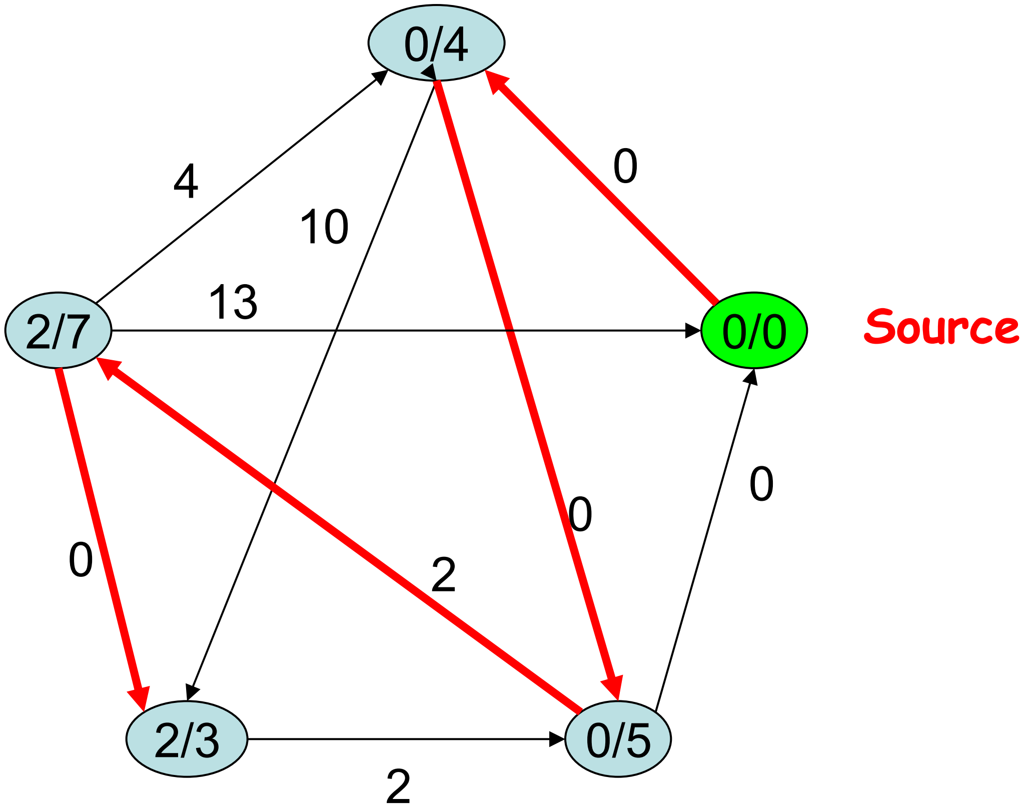
Learn DAA : From B K Sharma

## APSP:-Johnson's algorithm: Another Example



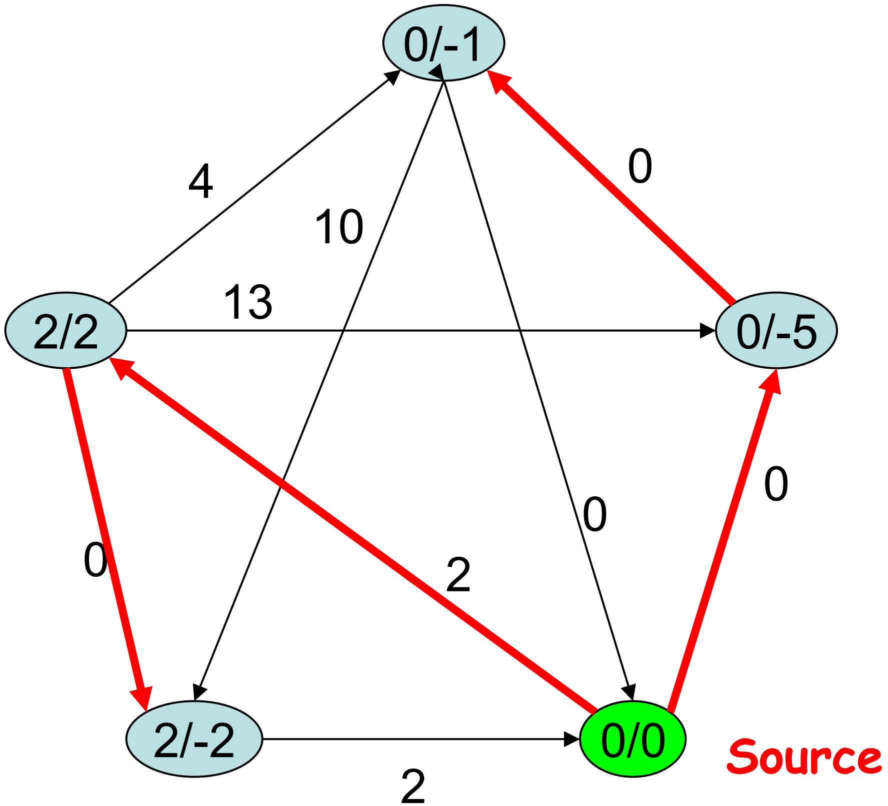
Learn DAA : From B K Sharma

## APSP:-Johnson's algorithm: Another Example



Learn DAA : From B K Sharma

## APSP:-Johnson's algorithm: Another Example



## APSP:-Johnson's algorithm: Another Example