U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO
DEI DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Analysis and Synthesis of Algorithms
Design of Algorithms

Greedy Algorithms

All-Pairs Shortest Paths Problem
Johnson's Algorithm

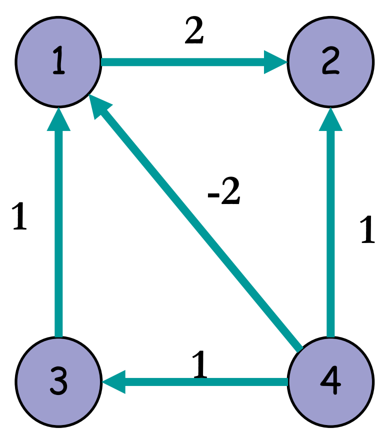
APSPs - Johnson's Algorithm

- Uses Dijkstra and Bellman-Ford-Moore Algorithms
- Based on **re-weighting of edges**
  - If all edges have non-negative weights, then use Dijkstra for each node
  - Otherwise, compute **new** set of non-negative weigths w', such that:
    - A shortest path from u to v using weight function w is also the shortest path using function w'
    - For each edge (u, v) the weigth w'(u, v) is non-negative
  - Obviously, computing w' should be efficient.

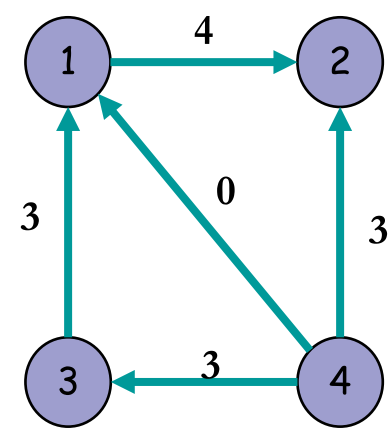
- How to reassign Weights to Edges?
  - Simple! Just add complement of smallest negative weight to all edges
  - All edges now become non-negative (≥ 0)

Simple Edge Weight Reassignment

- How to reassign Weights to Edges?
  - Simple! Just add complement of smallest negative weight to all edges
  - All edges now become non-negative (≥ 0)



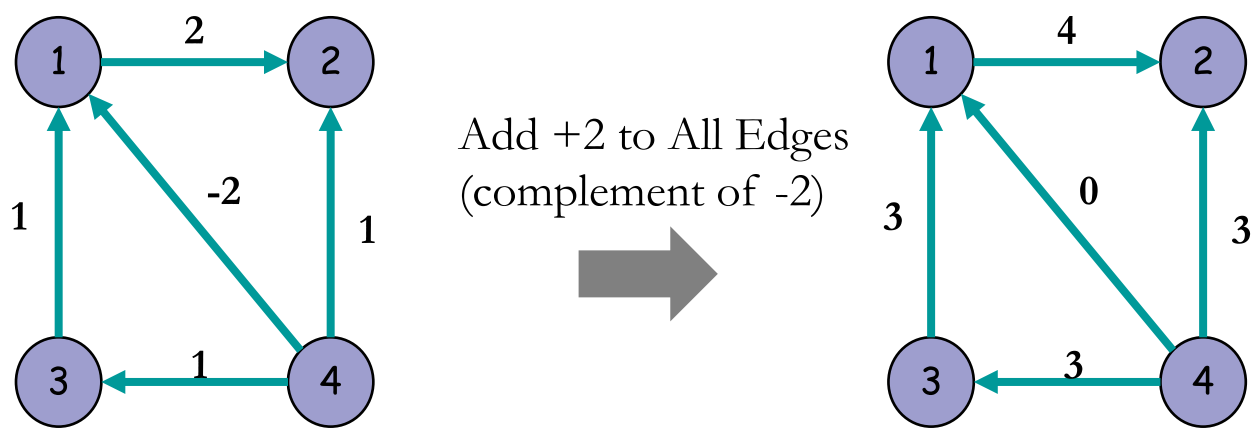
Add +2 to All Edges
(complement of -2)



Shortest path from 4 to 2: 0 via 1

Simple Edge Weight Reassignment

- How to reassign Weights to Edges?
  - Simple! Just add complement of smallest negative weight to all edges
  - All edges now become non-negative (≥ 0)

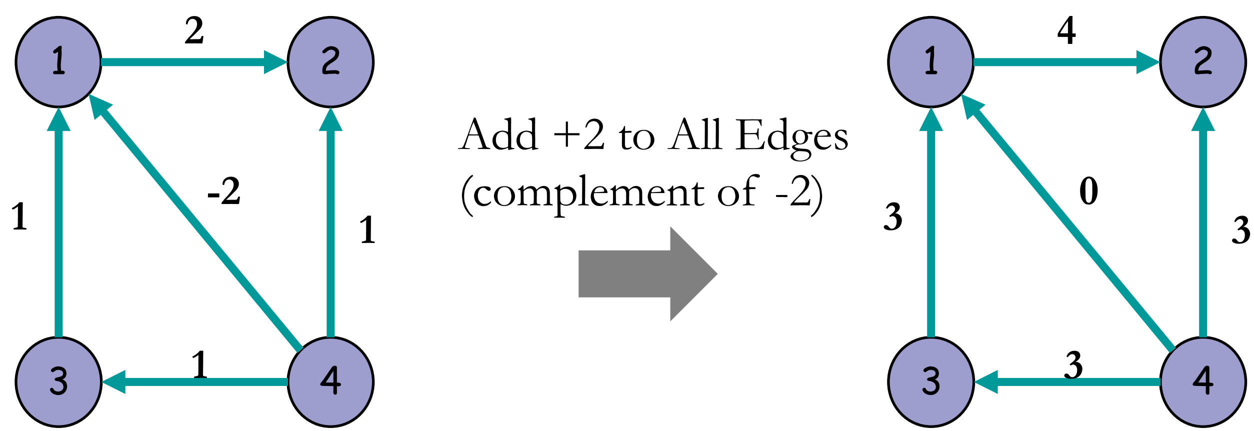


Shortest path from 4 to 2: 0 via 1

Shortest path from 4 to 2: 3 - direct

Simple Edge Weight Reassignment

- How to reassign Weights to Edges?
  - Simple! Just add complement of smallest negative weight to all edges
  - All edges now become non-negative (≥ 0)



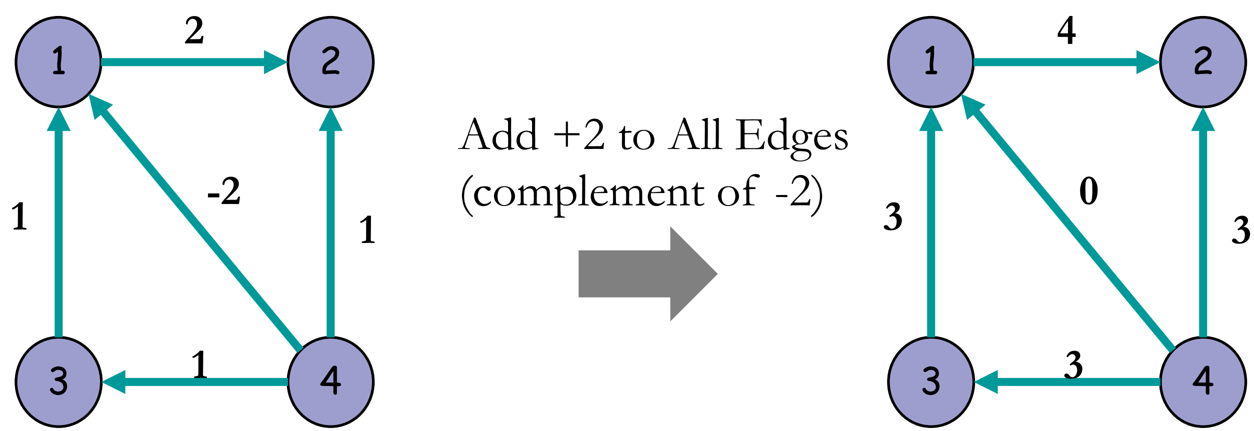
Shortest path from 4 to 2: 0 via 1

Shortest path from 4 to 2: 3 - direct

- Simple Edge reweighting Does not Work! Why?

Simple Edge Weight Reassignment

- How to reassign Weights to Edges?
  - Simple! Just add complement of smallest negative weight to all edges
  - All edges now become non-negative (≥ 0)



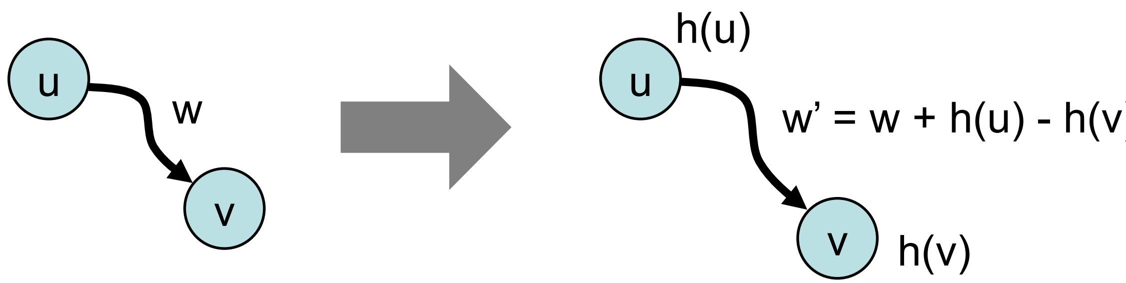
Shortest path from 4 to 2: 0 via 1

Shortest path from 4 to 2: 3 - direct

- Simple Edge reweighting Does not Work! Why?
  - Bias towards shorter paths as they have fewer contributions of the offset value...

Johnson's Algorithm: Edge Reweighting

- Given G = (V, E), and weight function w and δ as the shortest path function using w, then
  - The re-weighting function w' uses the function h: V → R, defined as w'(u, v) = w(u, v) + h(u) - h(v), with h(u) = δ(s,u), *i.e.*, shortest distance from source to u using w.



- Observations:
  - Weight of every path starting at u changes by h(u)
  - Weight of every path ending at u changes by -h(u)
  - Weight of a path passing through u does not change

Johnson's Algorithm: Edge Reweighting

- Given G = (V, E), and weight function w and δ create an augmented graph $G^* = (V+\{s\}, E + (s,u)\ \forall_{u \in E})$ and $w^*(s,u) = 0$, and $w^*(u,v) = w(u,v)$ otherwise

- Compute with G* the shortest distance (using the Bellman-Ford-Moore algorithm) between the newly added node s and every node u in G as h(u)

- Perform edge reweighting as:

$$w'(u,k) = w(u,k) + h(u) - h(k)$$

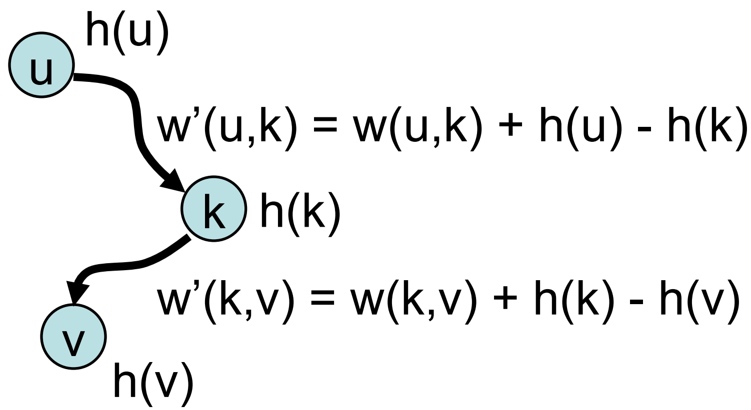
Johnson's Algorithm: Edge Reweighting

- **Observations**:
  - In G^*, $h(u) \leq 0$, since $w'(s,u) = 0$ and thus a shorter distance than 0 means the algorithm must traverse at least a negative weighted edge
  - The edge reweithing: $w'(u,k) = w(u,k) + h(u) - h(k)$ means that $w'(u,k) \geq 0$.

    Proof: Since h is the solution to the single source shortest path problem on G^*, we have $h(y) \leq h(x)+w(x,y)$. Thus $w'(x,y) = w(x,y)+h(x)-h(y) \geq 0$
- **Result:** Since $w' \geq 0$, we can use Dijkstra's algorithm n times to solve the single source shortest path problem on (G, w') using each vertex x as the source, giving us the function $\boldsymbol{\delta}'(x, y)$

Johnson's Algorithm: Edge Reweighting

- **Fact:** A shorter path with w is a shorter path with w'



$$
\begin{aligned}
w'(u,v) &= w'(u,k) + w'(k,v) = \\
&= w(u,k) + h(u) - \cancel{h(k)} + w(k,v) + \cancel{h(k)} - h(v) = \\
&= w(u,k) + w(k,v) + h(u) - h(v) \\
&= w(u,v) + \boxed{h(u) - h(v)}
\end{aligned}
$$

path independent

Johnson's Algorithm: Edge Reweighting

- **Fact:** A shorter path with w is a shorter path with w'

Proof (first part):

$$w(p) = \delta(v_0, v_k) \quad \Rightarrow \quad w'(p) = \delta'(v_0, v_k)$$

Hypothesis: If there exists a shorter path p_z from v_0 to v_k with w' then:

$$w'(p_z) < w'(p)$$

$$w(p_z) + h(v_0) - h(v_k) = w'(p_z) < w'(p) = w(p) + h(v_0) - h(v_k)$$

which means that:

$$w(p_z) < w(p)$$

Contradiction since w(p) is the shortest path with w **!**

Obs: For any paths p_1, p_2 from v_0 to v_k, the following holds

$$w(p_1) < w(p_2) \Leftrightarrow w'(p_1) < w'(p_2)$$

Johnson's Algorithm: Edge Reweighting

- **Fact:** A shorter path with w is a shorter path with w'

Proof (second part):

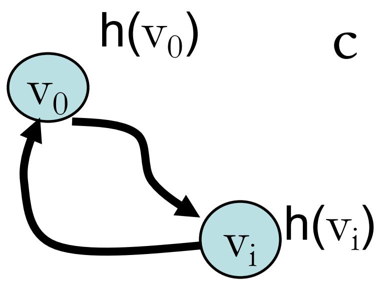
$$w(p) = \delta(v_0, v_k) \quad \Leftarrow \quad w'(p) = \delta'(v_0, v_k)$$

Similar argument:

Assume p_z as the shortest path from v_0 to v_k with w

Johnson's Algorithm: Edge Reweighting

- **Fact:** There exists a negative-weight cycle with w iff there exists a negative-wight cycle with w'



$c = \langle v_0, v_1, \ldots, v_k \rangle$ with $v_0 = v_k$

$$w'(v_0, v_0) = w(v_0, v_0) + h(v_0) - h(v_0)$$

$$= w(v_0, v_0)$$

path in G' has negative weight iff same path in G has negative weight

APSPs - Johnson's Algorithm

- Given G = (V, E), derive G' = (V',E'):
  - $V' = V \cup \{ s \}$
  - $E' = E \cup \{ (s, v) : v \in V \}$ ($\forall$ v $\in$ V, is reacheable from s)
  - $w(s, v) = 0$

- With negative-weight cycles:
  - Detected using the Bellman-Ford algorithm over G' **!**
- Without negative-weight cycles :
  - Define: $h(v) = \delta(s, v)$
  - Given that: $h(v) \leq h(u) + w(u, v)$
  - Property holds: $w'(u, v) = w(u, v) + h(u) - h(v) \geq 0$ **!**

  $\Rightarrow$ ***New edge weights w' are all non-negative***
- Execute Dijkstra algorithm for all nodes $u \in V$
  - Compute $\delta'(u,v)$, for $u \in V$
  - But also,
    - $\delta'(u,v) = \delta(u, v) + h(u) - h(v)$
    - $\delta(u,v) = \delta'(u, v) + h(v) - h(u)$

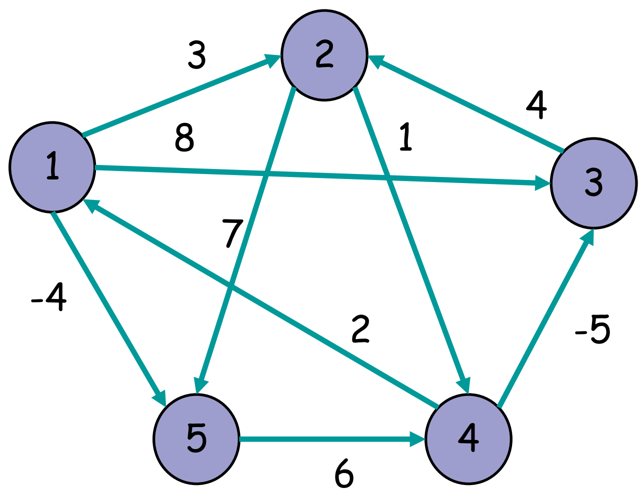
APSPs - Johnson's Algorithm

```
Johnson(G)
  derive G' by adding additonal source node s;
  if Bellman-Ford-Moore(G',w,s) = FALSE then
     print "negative cycle detected";
  else
    assign h(v) = δ(s, v), computed using Bellman-Ford-Moore;
    foreach edge (u,v) ∈ E[G] do
       compute w'(u,v) = w(u,v) + h(u) - h(v)
    foreach v ∈ V[G] do
       execute Dijkstra(G,w',v);
       compute δ'(u, v);
       backtrack values: δ(u, v); = δ'(u, v) + h(v) - h(u);
  return D
```

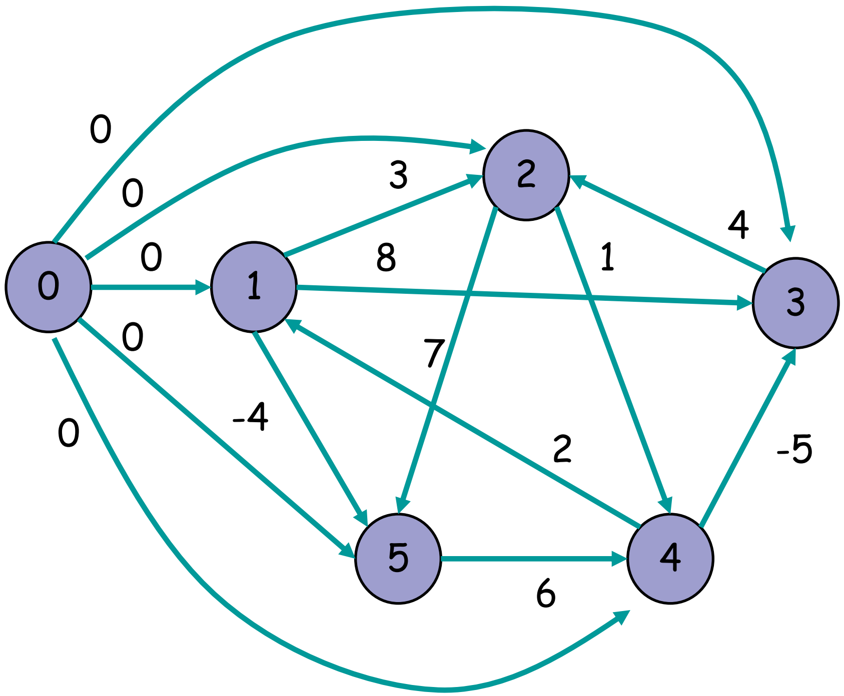
APSPs - Johnson's Algorithm

- Complexity:
  - Bellman-Ford-Moore: O(V E)
  - Execute Dijkstra for each node: O(V(V+E) logV)
    - Using binary (heap)
  - Total: O(V (V + E) log V)
    - Good for sparse graphs

Johnson's Algorithm: Example



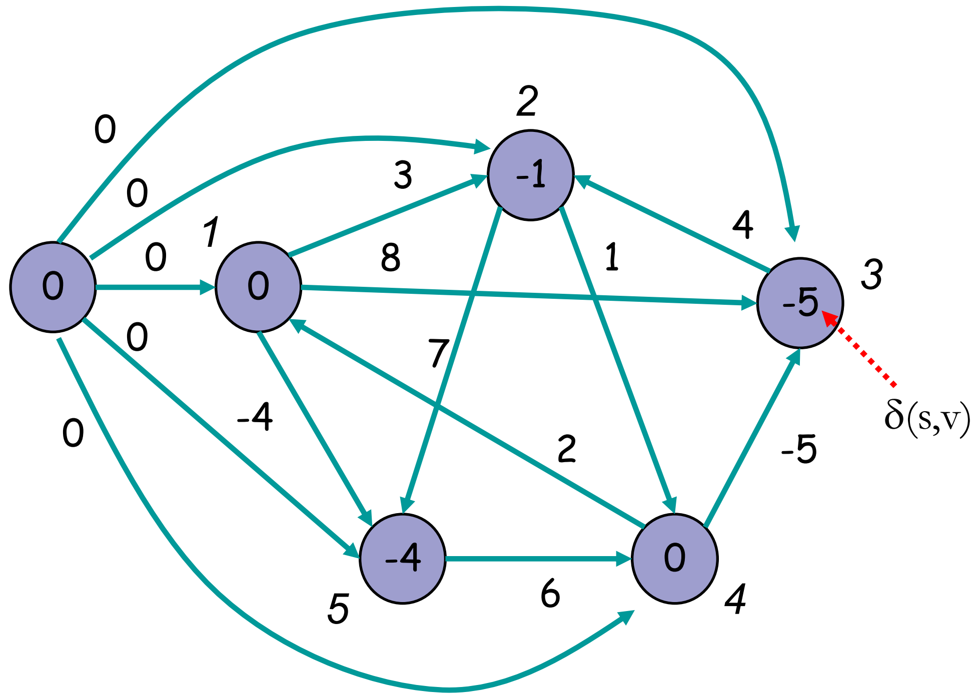
Johnson’s Algorithm: Example



New Graph G’

Negative Cycles?

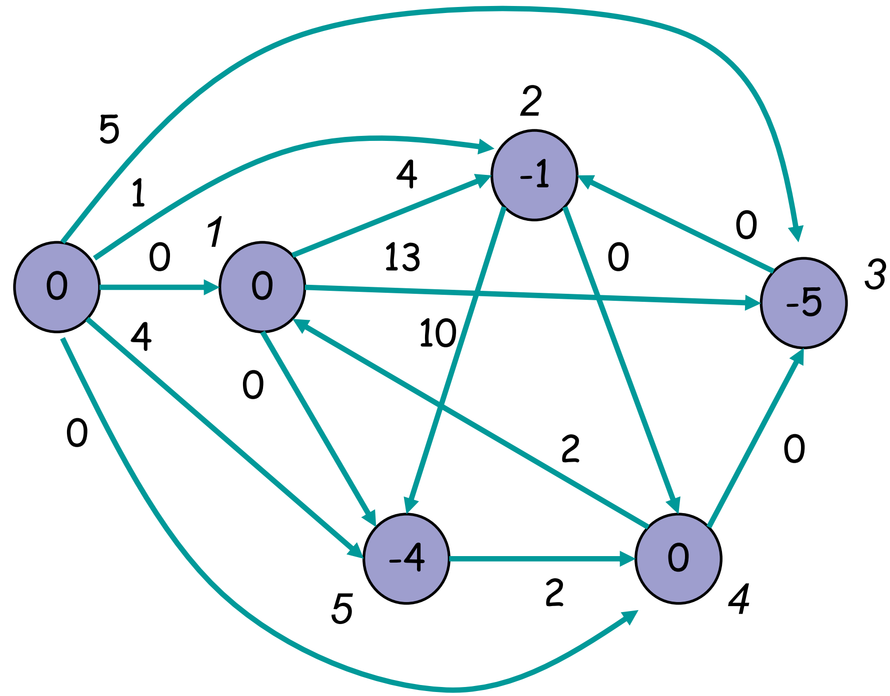
Johnson's Algorithm: Example



h(1) = 0
h(2) = -1
h(3) = -5
h(4) = 0
h(5) = -4

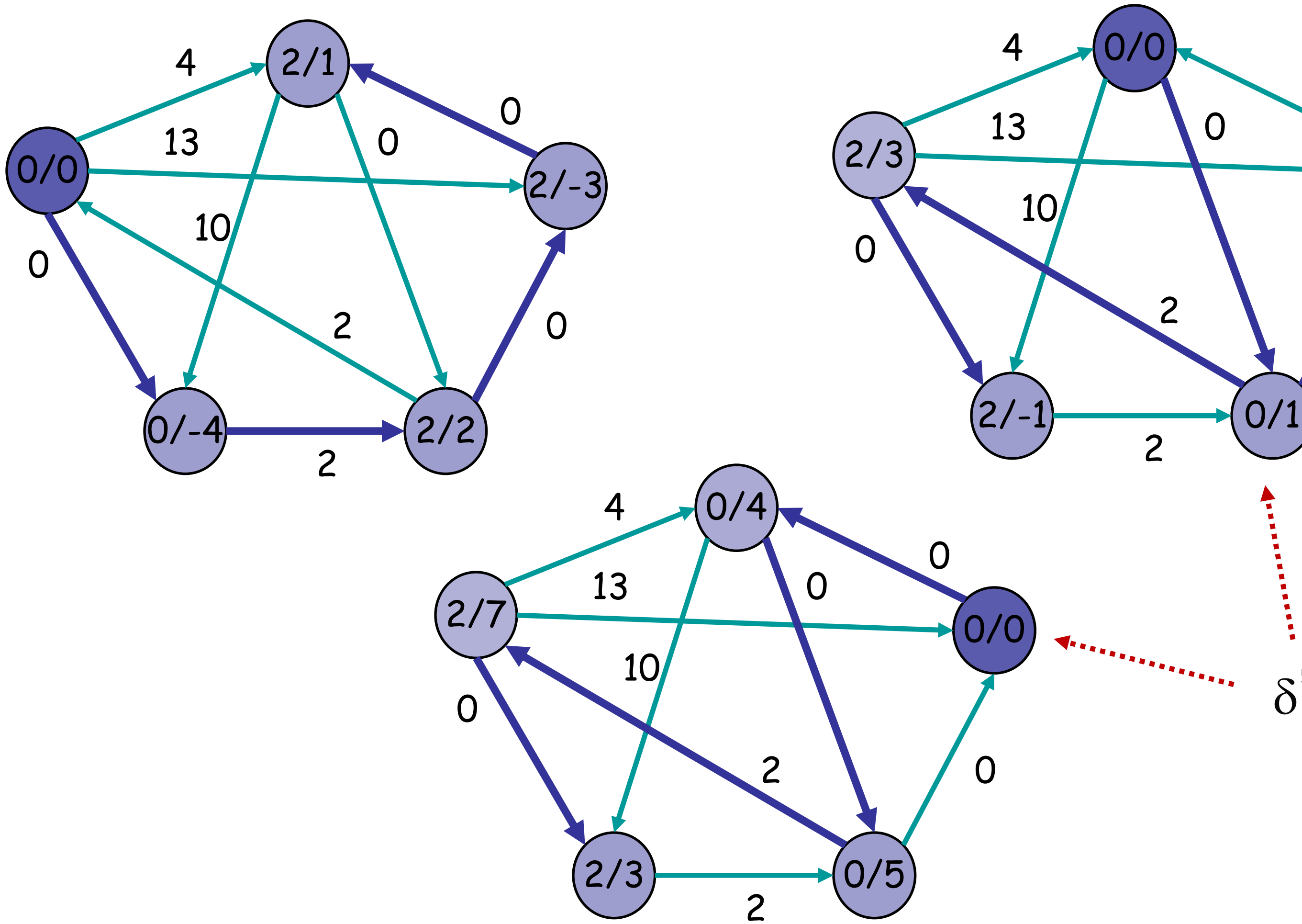
Use h(v) = δ(s,v) from Bellman-Ford

Johnson's Algorithm: Example



Reweight edges w'(u,v) = w(u,v) + h(u) - h(v)

Johnson's Algorithm: Example

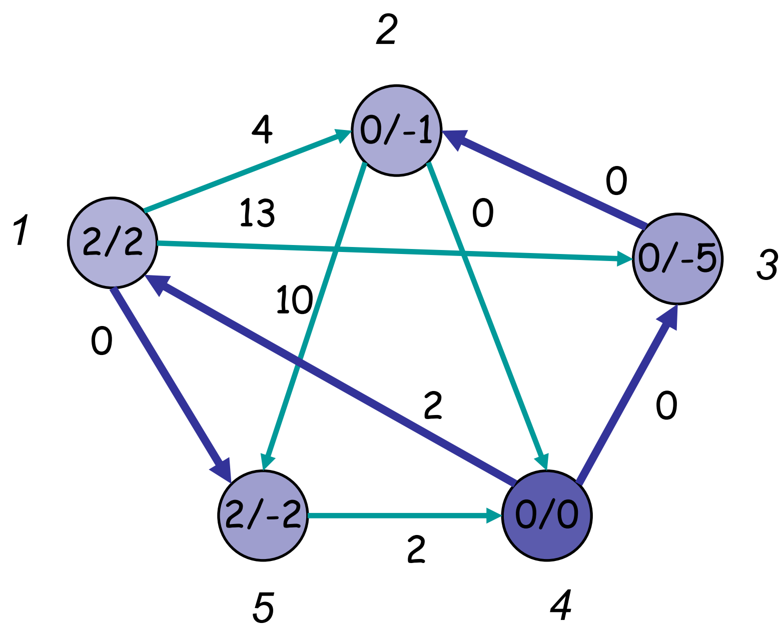


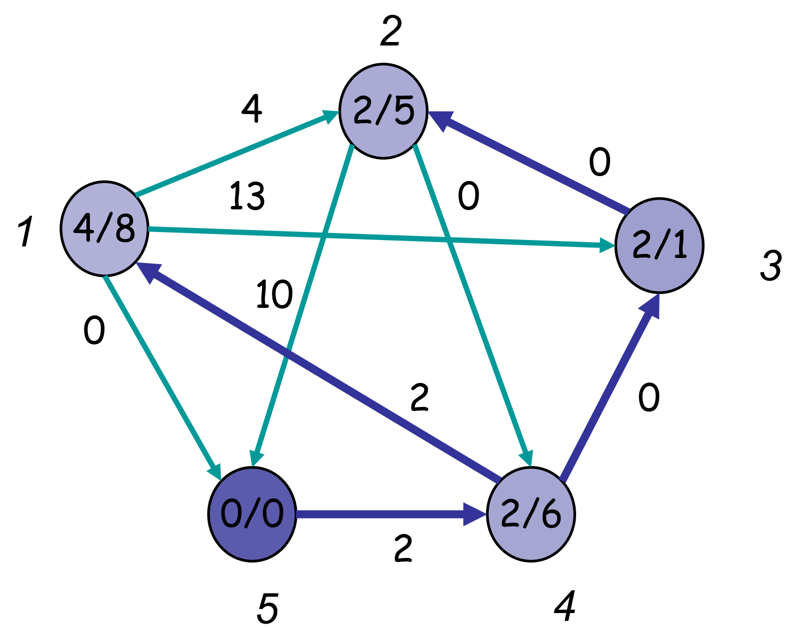
δ'(u,v)/δ(u,v)

Used Dijkstra on each node to compute δ'(u,v)

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

Johnson's Algorithm: Example





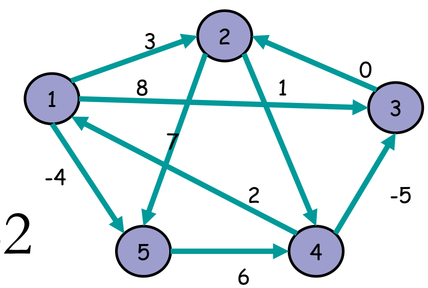
h(1) = 0
h(2) = -1
h(3) = -5
h(4) = 0
h(5) = -4

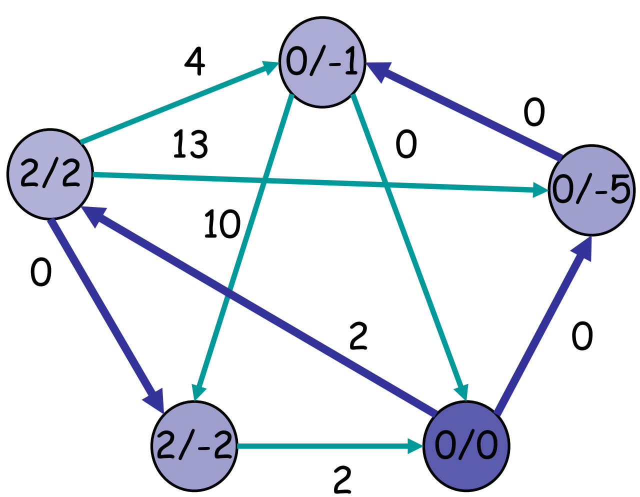
Used Dijkstra on each node to compute $\delta'(u,v)$

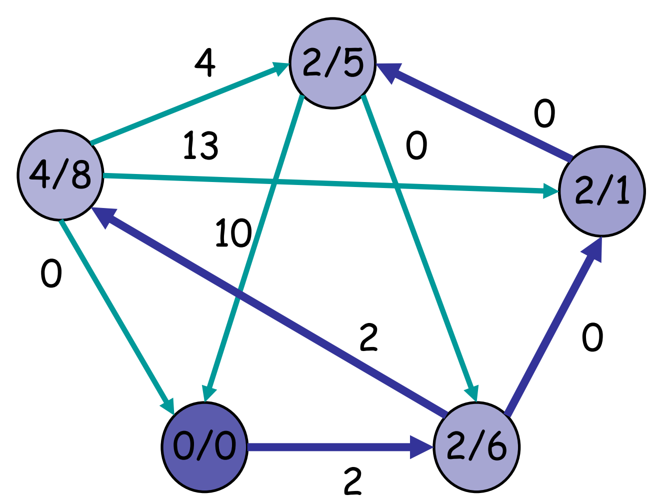
$$\delta(u,v) = \delta'(u,v) + h(v) - h(u)$$

Johnson's Algorithm: Example

- Use Shortest Paths of Reweighted Solution to Derive Shortest Distances in Original Graph
  - Use Predecessors in Original Graph
  - Examples:
    - Shortest Paths from node 4: 2, -1, -5, -2
    - Shortest Paths from node 5: 8, 5, 1, 6







Summary

- All-Pairs Shortest Paths (APSPs)
  - Edge Reweighting
  - Johnson's Algorithm