Johnson's Algorithm

- Floyd-Warshall algorithm takes $O(V^3)$ time.
- If Dijkstra's algorithm is run on every vertex of the same graph having non-negative weights the complexity would be $O(V^2 \log V + VE)$.
- If the graph is not dense, then Dijkstra's algorithm asymptotically outperforms the Floyd–Warshall algorithm.
- Johnson's algorithm uses both Dijkstra and Bellman-Ford's algorithm.
- It outperforms Floyd-Warshall algorithm when the graph is sparse.

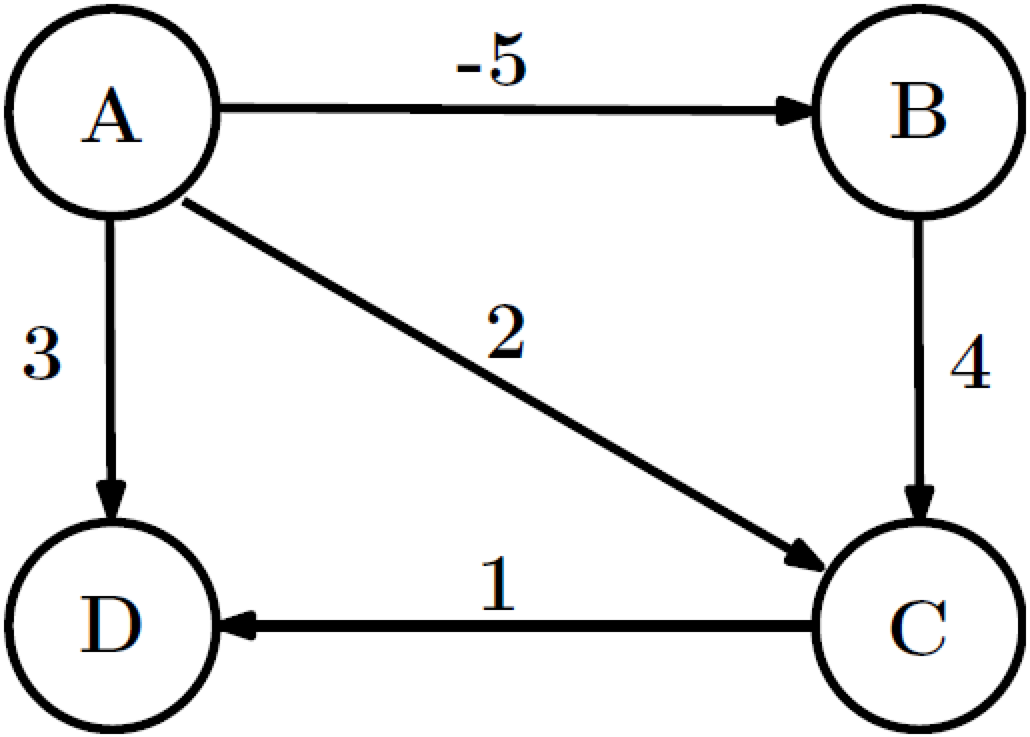
Working of the Algorithm

- If a graph G contains all non-negative edges, then Johnson's algorithm uses Dijkstra's algorithm on every vertex to find the all pair shortest path.

- If the graph G contains negative edges, then Johnson's algorithm uses Bellman-Ford algorithm to reweight the edges, so that all negative edges gets converted to non-negative edge.

- This is followed by Dijkstra's algorithm on every vertex to find the all pair shortest path.

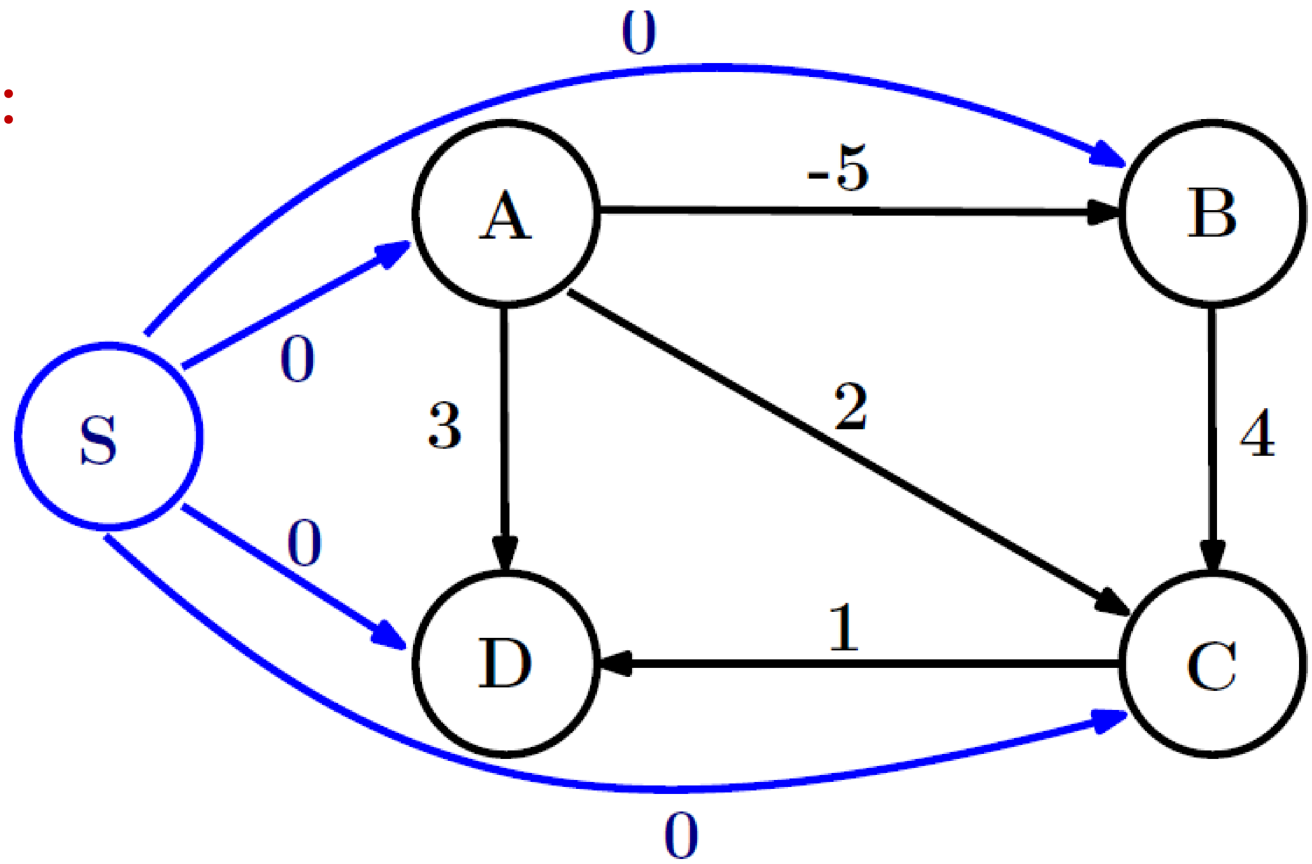
Initial Graph (G)



Graph G containing negative edge

Vertex Addition

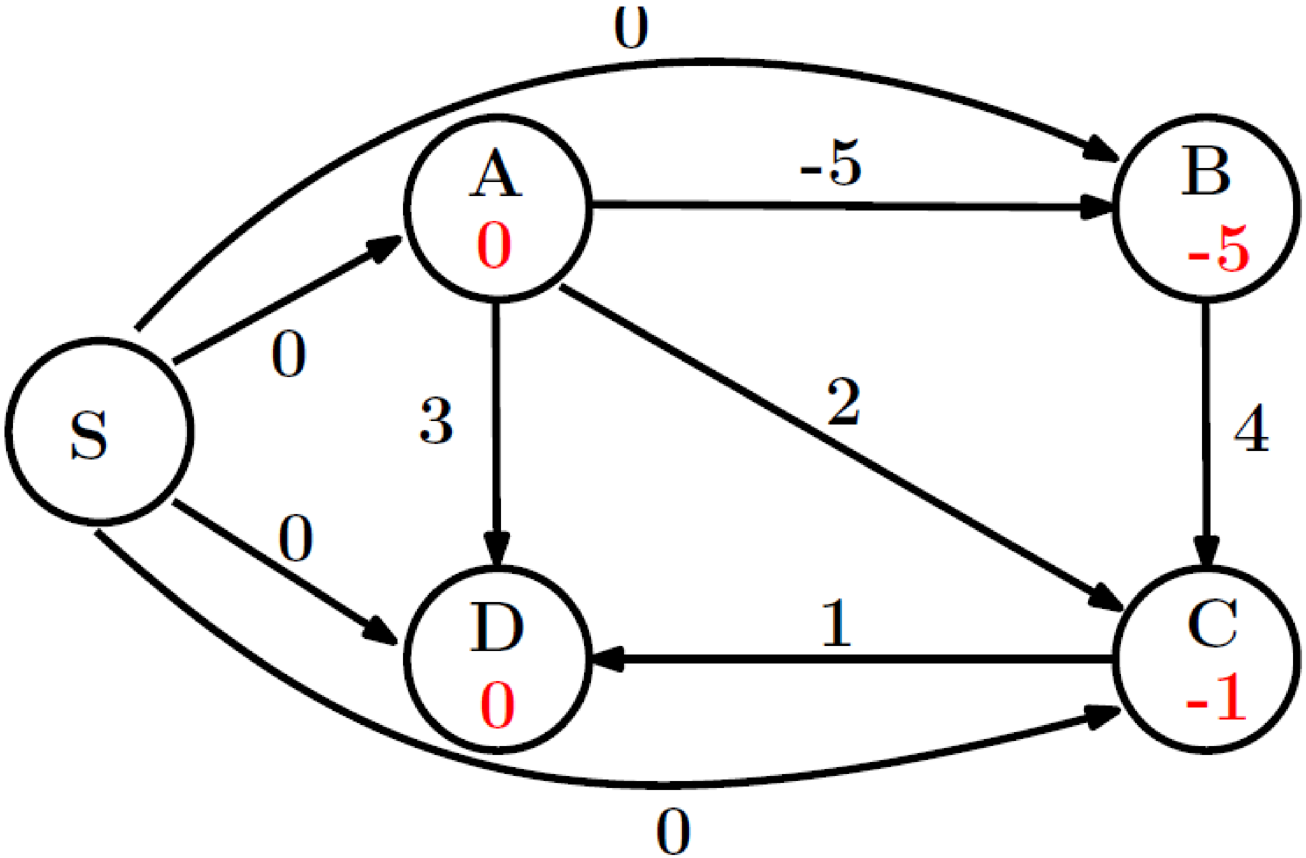
Step 1:



A vertex ‘S’ is added, and it is connected with all other edges through outbound edges and the weight of such edges are set to ‘0’.

Edge Weight Reweighting I

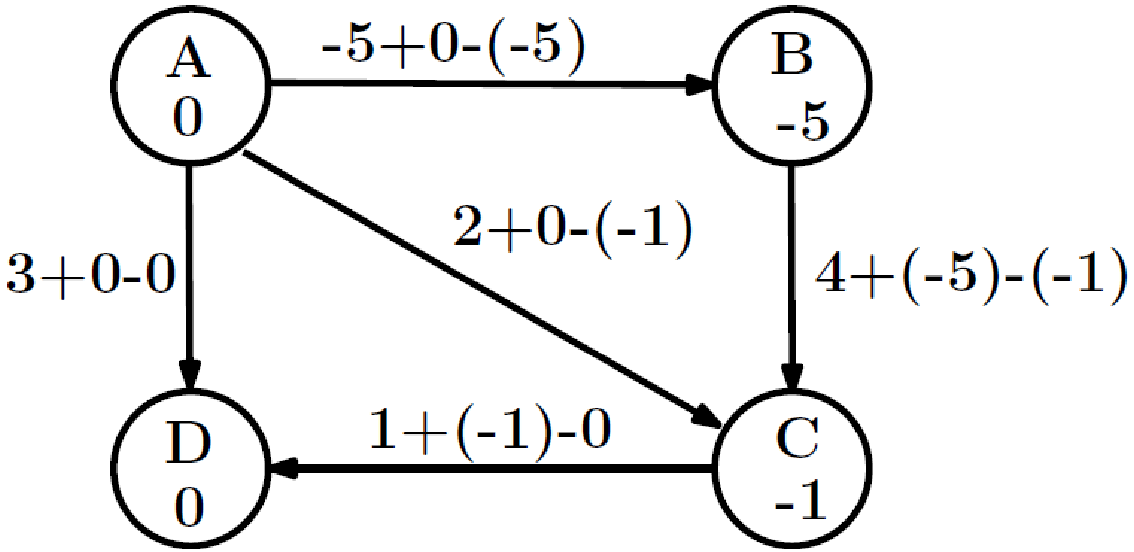
Step 2:



Bellman-Ford algorithm is applied with start vertex ‘S’, and the shortest path to each vertex is shown in red color.

Edge Weight Reweighting II

Step 3:

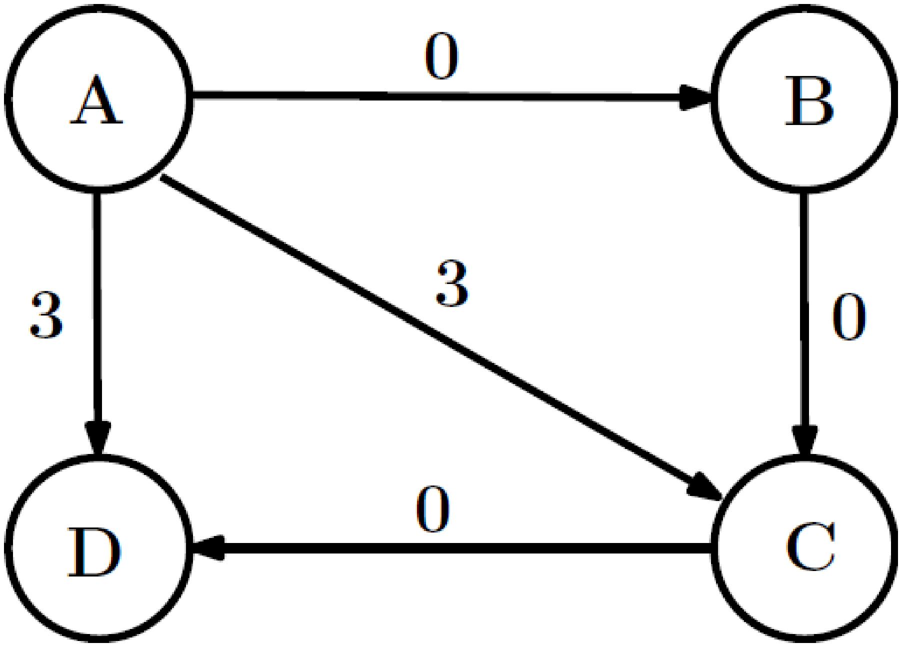


w(s,d)=w(s,d)+h(s)-h(d)

Given equation is applied on each edge weight to reweight the edges to non-negative values.

Edge Weight Reweighting III

Step 4:



Dijktra's algorithm is applied on each vertex of the reweighted graph to find the all pair shortest path.

Complexity

- O(VElogV) for dense graph
- O(V^2logV) for sparse graph