Johnson's algorithm for All-pairs shortest paths

The problem is to find shortest paths between every pair of vertices in a given weighted directed Graph and weights may be negative. We have discussed Floyd-Warshall Algorithm for this problem. 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. Johnson's algorithm uses both Dijkstra and Bellman-Ford as subroutines.

If we apply Dijkstra's Single Source shortest path algorithm for every vertex, considering every vertex as source, we can find all pair shortest paths in $O(V * V log\, V)$ time. So using Dijkstra's single source shortest path seems to be a better option than Floyd-Warshall, but the problem with Dijkstra's algorithm is, it doesn't work for negative weight edge.

The idea of Johnson's algorithm is to re-weight all edges and make them all positive, then apply Dijkstra's algorithm for every vertex.

How to transform a given graph to a graph with all non-negative weight edges?

One may think of a simple approach of finding the minimum weight edge and adding this weight to all edges. Unfortunately, this doesn't work as there may be different number of edges in different paths. If there are multiple paths from a vertex u to v, then all paths must be increased by same amount, so that the shortest path remains the shortest in the transformed graph.

The idea of Johnson's algorithm is to assign a weight to every vertex. Let the weight assigned to vertex u be h[u]. We reweight edges using vertex weights. For example, for an edge (u, v) of weight w(u, v), the new weight becomes w(u, v) + h[u] - h[v]. The great thing about this reweighting is, all set of paths between any two vertices are increased by the same amount and all negative weights become non-negative. Consider any path between two vertices s and t, weight of every path is increased by h[s] - h[t], all h[] values of vertices on path from s to t cancel each other.

How do we calculate h[] values? Bellman-Ford algorithm is used for this purpose. Follwing is the complete algorithm. A new vertex is added to the graph and connected to all existing vertices. The shortest distance values from the new vertex to all existing vertices are h[] values.

procedure Johnson:

1. Let the given graph be G. Add a new vertex s to the graph, add edges from new vertex to all vertices of G. Let the modified graph

be G'.

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. Note that the negative weight cycle cannot be created be new vertex s as there is no edge to s. All edges are from s.
3. Reweight the edges of original graph. For each edge (u, v), assign the new weight as "original weight + h[u] - h[v]"
4. Remove the added vertex s and run Dijkstra's algorithm for every vertex.

How does the transformation ensure nonnegative weight edges?

h[v] <= h[u] + w(u, v)

Time Complexity

Time Complexity of Bellman-Ford is O(VE) and time complexity of Dijkstra is O(V logV). So overall time complexity is $O(V^2log \space V + VE)$