

Relax operation

$d(u)$: the distance from the source to the node u (getting updated during single source shortest path computation)

$\pi(u)$: the parent of u (for trace back)

$W(v,u)$: the edge weight from v to u (if there is no edge, the weight is positive infinity)

$\delta(u)$: is the shortest distance from source to the node u

Relax (u, v)

If($d(u) > d(v) + w(v,u)$)

$d(u) = d(v) + w(v, u)$

$\pi(u) = v$

Bellman-ford algorithm for graphs with negatively weighted edges

Single source shortest path (G, s)

{

For each $u \in V$

$d(u) = \text{positive infinity}$

$d(s) = 0;$

For $I = 1, \dots, |V|$

For each $(u, v) \in E$

Relax(u,v);

}

Dijkstra's Algorithms

$S \leftarrow \emptyset$

$T \leftarrow V[G]$ (initialize a priority queue)

While

$Q \neq \emptyset$

$u \leftarrow \text{extract Min}(Q)$

$S \leftarrow S \cup \{u\}$

for each $V \in \text{Adj}[u]$

relax(u,v,w)

if we use array to implement Dijkstra's algorithms:

Initialize V

extract Min $V * V$

relax $1 * E$

Therefore

$O(V + V * V * E) = O(V^2)$

Use min-heap

V

$\lg V * V$
 $\lg V * E$

Therefore

$O(V+V \lg V+ E \lg V)= O(E \lg V)$

using Fibonacci heap $O(E + V \lg V)$

All Pairs Shortest Path

Johnson's Algorithm

Johnson's algorithm contains three steps:

- (1) For a graph G , construct a new graph G' with an additional node U and create directed edges from U to each and every node in G with weight 0.
- (2) Compute the weight (potential) of the nodes in G' where the weight. The weight of a node X , $h(X)$, is the shortest distance from U to X
- (3) Update the weight of each edge, \hat{W}_{uv} , with the following formula:
$$\hat{W}_{uv} = h(u) - h(v) + W_{uv}$$

Claims: (1) if $u \dots v$ is the shortest path in G
 $u \dots v$ is the shortest path in \hat{G}
(2) $\hat{W}_{uv} \geq 0$

Implementation:

Run Bellman-ford algorithm $V(V E)$ to obtain the potential of each node

Run Dijkstra's algorithm V times to obtain all pair shortest distance. Total running time $V(E + \lg V)$