R14 Johnson's Algorithm

Single Source Shortest Paths Review

Restrictions		SSSP Algorithm	
Graph	Weights	Name	Running Time $O(\cdot)$
General	Unweighted	BFS	$\lvert V\rvert + \lvert E\rvert$
DAG	Any	DAG Relaxation	$\lvert V\rvert + \lvert E\rvert$
General	Non-negative	Dijkstra	$\lvert V\rvert \log \lvert V\rvert + \lvert E\rvert$
General	Any	Bellman-Ford	$\lvert V\rvert \cdot \lvert E\rvert$

Tolve shortest paths problems, you must first define or construct a graph related to your problem, and then running an SSSP algorithm on that grap in a way that solves your problem.

- Also can solve other problems
  例えば, count connect components in a graph using Full-DFS or Full-BFS
  topologically sort vertices in a DAG using DFS
  detect negative weight cycles using Bellman-Ford.

All Pairs Shortest Paths

APSP problem asks for the minimum weight $\delta(u, v)$ of any path from u to v for every pair of vertices u,v in V.
A straight-forward way to solve this problem is to reduce to solving an SSSP problem |V | times, once from each vertex in V .

Johnson's Algorithm

The idea behind Johnson’s Algorithm is to reduce the ASPS problem on a graph with arbitrary edge weights to the ASPS problem on a graph with non-negative edge weights.
Then finding shortest paths in the re-weighted graph using |V | times Dijkstra will solve the original problem.
change the weight of each edge (a, b) from w(a, b) to w'(a, b) = w(a, b) + h(a) − h(b), to form a new weight graph G' = (V, E,w').

- proof in LEC NOTE

Find a vertex assignment function h

add a new node x to G with a directed edge from x to v for each vertex v ∈ V to construct graph G∗, letting h(v) = δ(x, v). This assignment of h ensures that w'(a, b) ≥ 0 for every edge (a, b).

Claim:

If $h(v) = \delta(x, v)$ and $h(v)$ is finite, then $w'(a, b) = w(a, b) + h(a) - h(b) \geq 0$ for every edge $(a, b) \in E$.

- Proof
  This Claim is equivalent to claiming $\delta(x, b) \leq w(a, b) + \delta(x, a)$ for every edge $(a, b) \in E$, i.e. the minimum weight of any path from x to b in G* is not greater than the minimum weight of any path from x to a than traversing the edge from a to b, which is true by definition of minimum weight.(This is simply a restatement of the triangle inequality.)

Algorithm process

1. Johnson's algorithm computes h(v) = δ(x, v), negative minimum weight distances from the added node x, using Bellman-Ford. If δ(x, v) = −∞ for any vertex v, then there must be a negative weight cycle in the graph, and Johnson's can terminate as no output is required.
2. Otherwise, Johnson's can re-weight the edges of G to w'(a, b) = w(a, b) +h(a)−h(b) ≥ 0 into G' containing only positive edge weights.
3. Then we can run Dijkstra (O(|V|log|V| + |E|)) |V|times on G' to find a single source shortest paths distance from each vertx u in G'.
4. Then we can compute each δ(u, v) by setting it to δ'(u, v)−δ(x, u)+δ(x, v).
5. Running Time
   Johnson's takes O(|V ||E|) time to run Bellman-Ford on x to every vertex in graph.
   and O(|V |(|V | log |V | + |E|)) time to run Dijkstra |V | times,
   so this algorithm runs in $O(|V|^2 \log |V| + |V||E|)$ time, asymptotically better than $O(|V|^2|E|)$ which run |V| times $Bellman - Ford$.