LEC14 Johnson's Algorithm

4.2 https://github.com/GUMI-21/MIT6.006_note

Previously

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

- BFS -> unweighted, maintain every level from source
- DAG Relaxation-> DAG, depends on topological order of DFS
- Dijkstra -> None-negative, maintain a extra priority queue with estimate distance
- Bellman-Ford -> full relaxation, and negative cycle detect.

All-Pairs Shortest Paths(APSP)

- Input: directed graph G = (V, E) with weights w : E → Z
- Output: δ(u, v) for all u, v ∈ V , or abort if G contains negative-weight cycle
- Just doing a SSSP algorithm |V | times is actually pretty good, since output has size O(|V |^2)
  - $|V| \cdot O(|V| + |E|)$with BFS if weights positive and bounded by $O(|V| + |E|)$
  - $|V| \cdot O(|V| + |E|)$ with DAG Relaxation if acyclic
  - $|V| \cdot O(|V| \log |V| + |E|)$ with Dijkstra if weights non-negative or graph undirected
  - $|V| \cdot O(|V| \cdot |E|)$ with Bellman-Ford (general)
- Today: Solve APSP in any weighted graph in |V | · O(|V | log |V | + |E|) time

Approach

- Idea!: Make all edge weights non-negative while preserving shortest paths!
  G' => with >=0 weights
- $Claim 1$
  We can compute distance in G from distances in G' in O(|V|(|V|+|E|))

- *Claim2*
  Not possible if G contains a negative weight cycle.
  shortest path from s to t is not simple, but shortest path in a graph with >= 0 weights are simple.

Making weights Non-negative

- Idea1
  Add large number to each edge => makes weights >= 0.
  but does not preserve shortest path, not good
- Idea2 better
  Given vertex V, -add weight h to all outgoing edges and -subtract weight to all incoming edges.
  $Claim$: Shortest Paths are preserved under this transformation
  Proof:
  -weight of every path starting at v changes by h
  -weight of every path ending at v changes by -h
  -weight of a path passing throught v does not change(locally) showed
- EVEN works with multiple vertices!
  Define a *potential function* h: V->Z, potential h(v)
  Make Graph G': same as G but edge(u,v) in E has weight
  w'(u,v) = w(u,v) + h(u) - h(v)
- $Claim$ SPs are still preserved
  Proof:
  $\pi, w(\pi)$ for v0->v1->....->vk of Graph G
  $w'(\pi) = \sum_{i=1}^{k}(w(v_{i-1}, v_i) + h(v_{i-1}) - h(v_i)) = w(\pi) + h(v_0) - h(v_k)$
  => $h(v) \leq h(u) + w(u, v)$
  every changes from v_0 to v_k by the same amout
  so any shortest path will still be short.
  -> ***v_0 and v_k are not be offseted. but evey path from v_0 to v_k has changed with h(v_0) and h(v_k), so the shortest path can be preserved!***

Algorithm

Can we find a potential function such that G' has no nagative edge weights?
• i.e., is there an h such that w(u, v) + h(u) − h(v) ≥ 0 for every (u, v) ∈ E? *(u,v) is a edge*

Idea

• Re-arrange this condition to h(v) ≤ h(u) + w(u, v), looks like *triangle inequality!*
Condition would be satisfied if *h(v) = δ(s, v) and δ(s, v) is finite, & h(u) = δ(s, u)* for some s.

Idea!

Add a new vertex s with a directed 0-weight edge to every v ∈ V ! *to detect Negative-cycle and get $\delta(s,v)$ for all v in Graph*
δ(s, v) ≤ 0 for all v ∈ V , since path exists a path of weight 0
Claim: If δ(s, v) = −∞ for any v ∈ V , then the original graph has a negative-weight cycle
Proof, (just Bellman-ford detect negative-cycle.)
-Adding s does not introduce new cycle (s has no incoming edges)
-So if reweighted graph has a negative-weight cycle, so does the original graph

- So if δ(s, v) is finite for all v ∈ V :
  – w0 (u, v) = w(u, v) + h(u) − h(v) ≥ 0 for every (u, v) ∈ E by triangle inequality!
  – New weights in G0 are non-negative while preserving shortest paths!

Johnson's Algorithm

A reduction Algorithm
-Construct G_s from G by adding vertex x connected to each vertex x in V with 0-weight edge *O(|V|+|E|)*
-Compute $\delta(s,v) all v \in V$ (e.g. by Bellman-Ford) *O(|V||E|)*
-if exist $\delta(s,v) = -\infty$: then abort
-else:
Make G' by reweighting evey edge - $(u,v) \in E$, $w'(u,v) = w(u,v) +$ $\delta(x,u) - \delta(x,v)$ -*O(|E|)*
Triangle inequality make sure w'(u,v) >= 0
means $w(u,v) + \delta(x,u) \geq \delta(x,v)$, this is defined by minimum weight!
-then For each u in V:

1. Compute shortest-path distances $\delta'(u,v)$ to all v in G' (using Dijkstra)
2. Compute $\delta(u,v) = \delta'(u,v) - \delta_x(x,u) + \delta_x(x,v)$ for all $v \in V$
   runtime: $O(|V|(|V|\log|V| + |E|))$

Correctness

Running Time

• O(|V | + |E|) time to construct Gx
• O(|V ||E|) time for Bellman-Ford
• O(|V | + |E|) time to construct G0
• O(|V | · (|V | log |V | + |E|)) time for |V | runs of Dijkstra

- O(|V | 2) time to compute distances in G from distances in G0
- O(|V | 2 log |V | + |V ||E|) time in total