Review of Floyd Warshall

Computes all possible paths between each pair of verticies in $\theta(|V|^3)$. Let there be a function $shortestPaths(i, j, k)$ that returns the shortest path from i to j using only the verticies from $1, 2, \cdots, k$.

Go through an example, with the adjacency graph:

$$\begin{bmatrix} & JFK & SFO & ORD & LAX \\ JFK & 0 & 500 & 200 & 900 \\ SFO & 500 & 0 & 600 & 50 \\ ORD & 200 & 600 & 0 & 700 \\ LAX & 900 & 50 & 700 & 0 \end{bmatrix}$$

First you only want direct flights so, the best you can do is:

$$JFK - SFO \Rightarrow 500$$

$$JFK - ORD \Rightarrow 200$$

$$JFK - LAX \Rightarrow 900$$

$$ORD - SFO \Rightarrow 600$$

$$ORD - LAX \Rightarrow 700$$

$$SFO - LAX \Rightarrow 50$$

You are a poor college student, and can't afford the \$900 flight from JFK to LAX, so you look for other paths that might involve lay-overs. You notice there's a cheaper flight from JFK to SFO and thenf from SFO to LAX, since $500 + 50 < 900$. You feel really clever at this point, and decide to start a service where you find the cheapest flights from any destination to any other destination (but with potentially a lot of layovers).

For every pair of distances you try to figure out if adding one more layover stop would help you find a cheaper flight.

```
for each LAYOVER in V:
   for each START in V:
       for each DESTINATION in V:
           if currentBestPrice[START -> DESTINATION] >
                  currentBestPrice[START -> LAYOVER] +
                  currentBestPrice[LAYOVER -> DESTINATION]:

              currentBestPrice[[START -> DESTINATION] =
                  currentBestPrice[START -> LAYOVER] +
                  currentBestPrice[LAYOVER -> DESTINATION
```

And that's basically Floyd-Warshall.

Review of Johnson's Algorithm

Let $G = (V, e, w)$ be a weighted directed graph. Let $(u, v) \in E$ represent edges $u \to v$, and $w(u, v)$ be the edge weights.

Recall that Dijkstra's Alg had a run-time of $O(|E| + |V| \log |V|)$ using Fibonacci Heaps. Running Dijkstra's from every node solves the All-Pairs-Shortest-Path problem in $O(|E||V| + |V|^2 \log |V|)$ if all $w(u, v)$ are nonnegative. Johnsons enables us to run Dijkstra's on any graph, even those with negative weights, by transforming $G = (v, E, w)$ to $G' = (V, E, w')$ such that all $w'(u, v) > 0$.

This transformation must be chosen so that the shortest paths and their values can be retrieved on the original graph G from G'.

Wrong Idea: Add Absolute value of the minimum weight edge to all edges in G to get G'
This does get you non-negative weight edges but the shortest paths could be different in G', since paths with more edges in gain more total weight by this transformation than do paths with shorter edges.

Insight 1: Telescoping Sums
Let $w'(v_j, v_k) = w(v_j, v_k) + h(v_j) - h(v_k)$. The weight of any path P between any pari of verticies v_j and v_k in G', where P is the set of edges $(v_j, v_{j+1}, \cdots, (v_{k-1}, v_k)$:

$$w'(P) = \sum_{i=j+1}^{k} w'(v_{i-1}, v_i)$$

$$w'(P) = \sum_{i=j+1}^{k} w(v_{i-1}, v_i) + h(v_{i-1}) - h(v_i)$$

$$w'(P) = w(P) + h(v_j) - h(v_k)$$

We saw this last recitation when talking about potential functions. Because the weight of each path between v_j and v_k is offset by the same constant amount, $h(v_j) - h(v_k)$, the shortest paths in G are composed of the same edges as those in G'. We can get the original path weights, $w(P)$ by noting that $w(P) = w'(P) - (h(v_j) - h(v_k))$.

Insight 2: Triangle Inequality
Insight 1 gives us a strategy to modify the weights to produce G' and recover the shortest paths in G from G'. Our original goal, however, was to make all $w'(u, v) > 0$. Therefore, we must choose $h(v)$ such that the following holds: $w'(u, v) \geq 0 \Rightarrow w(u, v) + h(u) - h(v) \geq 0 \Rightarrow h(v) \leq h(u) + w(u, v)$.

The last inequality should look familiar it is the triangle inequality that holds for shortest path distances.

Claim from Lecture: If there is a negative weight cycle in the input then no such solution to the h's exist.
Let there be a negative weight cycle from $v_0 \to v_1 \to \cdots \to v_k \to v_0$. If there exists a valid h

mapping:

$$h(v_1) - h(v_0) \leq w(v_0, v_1)$$
$$h(v_2) - h(v_1) \leq w(v_2, v_1)$$
$$\ldots$$
$$h(v_k) - h(v_{k-1}) \leq w(v_{k-1}, v_k)$$
$$h(v_0) - h(v_k) \leq w(v_0, v_k)$$

The sum of the inequalities above yields $0 \leq w(cycle)$. This contradicts our initial statement that the cycle was a negative weight cycle.

If there are no negative weight cycles, we can solve the difference constraints
Let $\delta(u, v)$ denote the distance of the shortest path between u and v. To generate an appropriate set of $h(v)$ values, we can simply add an extraneous vertex s connected to all vertices $v \in V$ via edges $s \to v$ with arbitrary weights (for simplicity, Johnsons chooses weights of 0),and run Bellman-Ford from s to produce $h(v) = \delta(s, v)$ values for all vertices. By definition of shortest path, $\delta(s, v) \leq \delta(s, u) + w(u, v) = h(v) \leq h(u) + w(u, v)$, as desired.

Summary
Putting all this together, Johnsons Algorithm is as follows:

1. Introduce an extra vertex s and connect it to all $v \in V$ (as specified above.)
2. Run Bellman-Ford from s to produce $h(v) = \delta(s, v)$ values.
3. Transform $G \to G'$ by setting $w'(u, v) = w(u, v) + h(u) - h(v)$.
4. Run Djikstras from every vertex in G' (now possible since all $w'(u, v) \geq 0$.
5. . Compute the distance for every shortest path P between pairs of vertices (u, v) in G via the relation $w(P) = w'(P) - h(u) + h(v)$.

The total run-time (broken down by steps above) is then $\theta(|V| + |V||E| + |E| + (|V||E| + |V|^2 log|V|) + |V|^2) = \theta(|V||E| + |V|^2 log|V|)$.