Chapter 58

Johnson's Algorithm

Johnson's algorithm solves the all-pairs shortest paths (APSP) problem. It allows for negative weights and is significantly more efficient than simply running Bellman-Ford's algorithm from each source. It uses a clever trick to tweak the weights and eliminate negative ones without altering shortest paths.

All Pairs from Single Source. One way to solve the APSP problem is by running the Bellman-Ford algorithm from each vertex. For a graph with n vertices and m edges, this gives an algorithm with total work

$$W(n, m) = O(mn) \times n = O(mn^2).$$

Johnson's algorithm improves on this by using Dijkstra's algorithm to find the shortest paths from each source vertex and by using the more expensive Bellman-Ford's algorithm to tweak the weights. Johnson's algorithm proceeds in two phases.

1. The first phase runs Bellman-Ford's algorithm and uses the result to update the weights on the edges and eliminate all negative weights.

2. The second phase runs Dijkstra's algorithm from each vertex in parallel.

For a graph with n vertices and m edges, Johnson's algorithm has the following costs:

	Work	Span
$1 \times$ Bellman Ford	$O(mn)$	$O(n \log n)$
$n \times$ Dijkstra	$n \times O(m \log n)$	$O(m \log n)$
Total	$O(mn \log n)$	$O(m \log n)$.

The work improves over the naive $O(mn^2)$ bound by a factor of $n/\log n$, and the span is no more than a single Dijkstra. The parallelism is therefore $\Theta(n)$, which is significant.

Potentials. To update the weights on the graph we will use Bellman-Ford to assign a "potential" $p(v)$ to each vertex v. We then add and subtract the potentials on the endpoints of each edge to get new weights. This adjustment change the weights of paths in the graph, but it will not change the shortest path (i.e. its sequence of edges) between any two vertices. The following lemma establishes this property, more formally.

Lemma 58.1 (Path Potentials)**.** Consider a weighted graph $G = (V, E, w)$, and any assignment $p(v) : V \to \mathbb{R}$. For

$$w'(u, v) = w(u, v) + p(u) - p(v) ,$$

and $G' = (V, E, w')$, we have that:

$$\delta_G(u, v) = \delta_{G'}(u, v) - p(u) + p(v) .$$

Proof. **Summary:** Along any path all potentials except the first and last cancel since each is subtracted from the incoming edge and added to the outgoing edge. Therefore the total weight of a path is the original weight $+p(u) - p(v)$. Since this only depends on u and v, which path is the shortest from u to v does not change, and the change in weight ($p(u) - p(v)$) can be subtracted out.

Full proof: Consider any path of edges $P = \langle (h_0, t_0), \cdots, (h_l, t_l) \rangle$ from u to v. Here h and t means head and tail of each edge, and hence $h_0 = u$, $t_l = v$ and $t_i = h_{i+1}$ for $0 \leq i < l$. In the original graph the path length is

$$w(P) = \sum_{i=0}^{l} w(h_i, t_i) .$$

In the modified graph the path length is

$$\begin{aligned} w'(P) &= \textstyle\sum_{i=0}^{l} (w(h_i, t_i) + p(h_i) - p(t_i)) \\ &= w(P) + \textstyle\sum_{i=0}^{l} p(h_i) - \sum_{i=0}^{l} p(t_i) \end{aligned}$$

Now if we look at the two sums on the right, and given that $h_{i+1} = t_i$, all but the first term of the first sum and last term of the second sum pairwise cancel out. This leaves just $p(h_0) - p(t_l) = p(u) - p(v)$. Therefore

$$w'(P) = w(P) + p(u) - p(v) .$$

Since $p(v)$ and $p(u)$ are the same for any path from u to v, this does not change which path(s) from u to v is shortest, just the weight of that path. Therefore

$$\delta_{G'}(u, v) = \delta_G(u, v) + p(u) - p(v) ,$$

giving

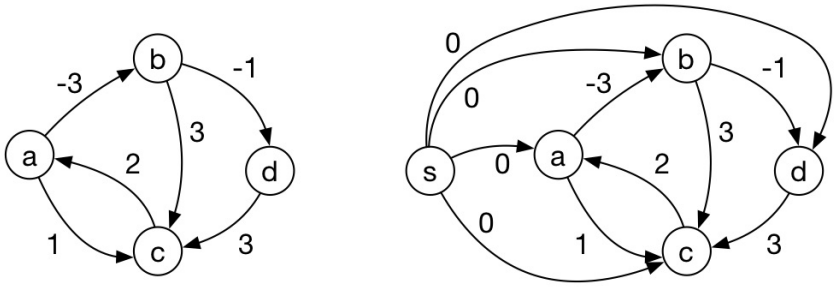
$$\delta_G(u, v) = \delta_{G'}(u, v) - p(u) + p(v) .$$

□

Algorithm 58.1 (Johnson's Algorithm)**.** The pseudo-code for Johnson's algorithm is shown below. It starts by adding a dummy source vertex s to the graph and zero weight edges between the source and all other vertices. It then runs Bellman-Ford's algorithm on this graph and adjusts the weights by using the shortest paths computed. As established by Lemma 58.2 this adjustment eliminates all negative weights from the graph. Finally, it runs Dijkstra's algorithm for each vertex u and returns the distance from u to all other (reachable) vertices.

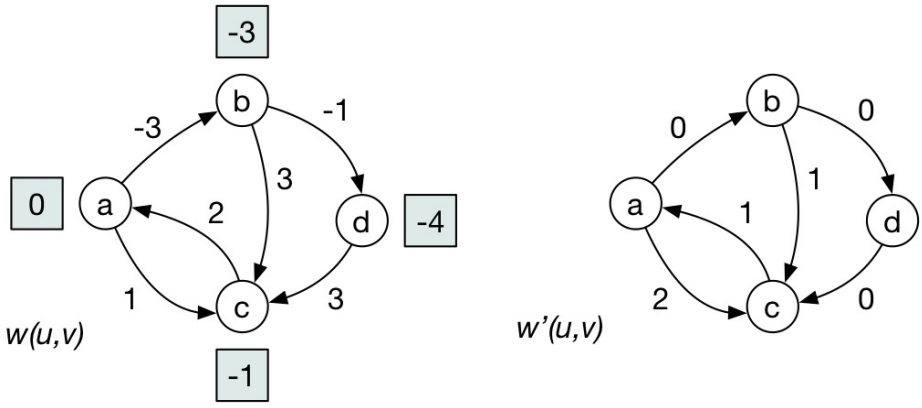
$JohnsonAPSP\ (G = (V, E, w)) =$
`let`
  $G^+ =$ Add a dummy vertex s to G,
    and a zero weight edge from s to all $v \in V$.
  $D = BellmanFord(G^+, s)$
  $w'(u, v) = w(u, v) + D[u] - D[v]$ (* $w'(u, v) \geq 0$ *)
  $G' = (V, E, w')$
  $Dijkstra'\ u =$
    `let` $\Delta_u = Dijkstra\ G'\ u$
    `in` $\{(u, v) \mapsto (d - D[u] + D[v]) : (v \mapsto d) \in \Delta_u\}$ `end`
`in` $\bigcup_{u \in V}(Dijkstra'\ u)$ `end`

Example 58.1. As an example of Johnson's algorithm consider the graph on the left.



On the right we show the graph after adding the source$s$ and the edges from it to all other vertices.

We now run Bellman-Ford's algorithm on this graph. It gives the distances shown in squares below on the left.



On the right we show the graph after the weights have been updated so that $w'(u,v) = w(u,v) + D[u] - D[v]$. This graph has no negative weight cycles.

To calculate the distance from (a,d), for example, we can use $\delta_G(a,d) = \delta_{G'}(a,d) - D[a] + D[d]$, which gives the correct distance of $0 + 0 + (-4) = -4$.

All the $\delta_{G'}(u,v)$ are calculated using Dijkstra's.

Lemma 58.2 (Non-Negative Weights)**.** For $p(v) = \delta_{G^+}(s,v)$, all edge weights $w'(u,v) = w(u,v) + p(u) - p(v)$ are non-negative.

Proof. **Summary:** For an original edge (u,v) with weight $-a$, the distance to v has to be at least a less than that to u. This difference $p(u) - p(v) \geq a$ will cancel out the negative edge.

Full proof: The sub-paths property tells us that:

$$\delta_{G'}(s,v) \leq \delta_{G'}(s,u) + w(u,v) \ .$$

Since the shortest path cannot be longer than the shortest path through u, we have that:

$$\begin{aligned} 0 &\leq \delta_{G'}(s,u) + w(u,v) - \delta_{G'}(s,v) \\ &= w(u,v) + \delta_{G'}(s,u) - \delta_{G'}(s,v) \\ &= w(u,v) + p(u) - p(v) \end{aligned}$$

□

Note. Note that the *Dijkstra* function in Johnson's algorithm readjusts the path weighs to zero out the impact of the potentials. This readjustment guarantees that the final distances are the correct distances.

Remark. Although we set the weights from the "dummy" source to each vertex to zero, any finite weight for each edge will do. In fact all that matters is that the distances from the source to all vertices in G' are non-infinite. Therefore if there is a vertex in the original graph G that can reach all other vertices then we can use it as the source and there is no need to add a new source.