The Bellman-Ford Algorithm for Directed Graphs

- The Problem:

  Given a directed graph D with weighted arcs, find the least-weight directed path from vertex v_0 to each other vertex in the digraph, provided that there is no directed cycle with negative total arc weight in the digraph.

- The Algorithm:

  The idea behind the algorithm is to iteratively calculate the least-weight path of length at most i from v_0 to each other vertex, for $i = 0, 1, 2, \ldots$

  Since any directed path containing n vertices will contain $n-1$ arcs, we can stop when $i = n-1$ (assuming that the vertices of the digraph are named $v_0, v_1, \ldots, v_{n-1}$). However, we go as far as $i = n$ so that we can test for negative-weight directed cycles.

  1. `Let` $d_j^0 = \begin{cases} 0 & \texttt{if } j = 0 \\ \infty & \texttt{if } j > 0 \end{cases}$
  2. `For` $i = 1, 2, 3, \ldots, n$
     - `For` $j = 0, 1, 2, \ldots, n-1$
       - `Find a` k `such that` $d_k^{i-1} + wt(v_k, v_j) \leq d_h^{i-1} + wt(v_h, v_j)\ \ \forall v_h \in V(D)$
       - `If` $d_j^{i-1} > d_k^{i-1} + wt(v_k, v_j)$
         - `then let` $d_j^i = d_k^{i-1} + wt(v_k, v_j)$ `and let` $p_j = k$
       - `else let` $d_j^i = d_j^{i-1}$
  3. `If` $d_j^n = d_j^{n-1}$ `for each` $j \in \{0, 1, 2, \ldots, n-1\}$`, then stop since we have found least-weight paths from` v_0 `to all other vertices.`
     `Otherwise, a directed cycle exists. We can find it by selecting some vertex` v_j `for which` $d_j^n < d_j^{n-1}$ `and then following the predecessors back from` v_j `until some vertex is repeated.`

  In step (3), if $d_j^n < d_j^{n-1}$ for some j, then there is a path from v_0 to v_j using n arcs such that this path has lower weight than the least-weight path from v_0 to v_j using at most $n-1$ arcs. But, as we've already noted, any path containing n vertices (and there are exactly n vertices in our directed graph) can contain at most $n-1$ arcs. So, if using n arcs turns out to be better, it follows that not only is there a directed cycle (since we can't use n arcs in a digraph with n vertices without repeating some vertex), but this directed cycle must have a negative weight (since it was more cost-effective to repeat a vertex, rather than just visit it at zero cost).

- The Algorithm's Running Time

  The Bellman-Ford algorithm runs in $O(n^3)$ time. In step (2), the two `For` loops contribute a factor of n, and to "`Find a` k" requires another factor of n (since we test each k from 0 to $n-1$).