Bellman's Algorithm for Acyclic Directed Graphs

- The Problem:

  Given a directed graph whose arcs have non-negative weights, find the least-weight directed path from vertex v to each other vertex in the digraph.

- Canonical Vertex Orderings:

  Theorem:

  A directed graph D is acyclic if and only if there exists an ordering, $v_0, v_1, \ldots, v_{n-1}$, of its vertices such that $i < j$ whenever $(v_i, v_j) \in A(D)$, where n denotes the number of vertices of D and $A(D)$ denotes the arc set of D.

  Such an ordering of the vertices of D is called a *canonical ordering*.

  Proof:

  I. Assume that a canonical ordering exists, and suppose that D contains a directed cycle, $v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_t}$ where $v_{i_1} = v_{i_t}$. Since $i < j$ whenever $(v_i, v_j) \in A(D)$, we thus have $i_1 < i_2 < i_3 < \cdots < i_t$. But $i_1 = i_t$, which is a contradiction. Hence, if a canonical ordering exists then D must be acyclic.

  II. Assume that D is acyclic. A canonical ordering of the vertices is generated by the following algorithm (the output of which is either a canonical ordering or else a directed cycle):

  1. `Let` $t = 0$ `and` $N = V(D)$.
  2. `Let` $v_t \in N$ `be a vertex such that` $\{(u, v_t) | u \in N$ `and` $(u, v_t) \in A(D)\} = \emptyset$, `assuming that such a vertex exists. If no such vertex exists, then proceed to (3); otherwise let` $N = N \setminus \{v_t\}$, $t = t + 1$, `and repeat (2).`
  3. `If` $N = \emptyset$, `then` $v_0, v_1, \ldots, v_{n-1}$ `is a canonical ordering.`
     `Otherwise,` $N \neq \emptyset$, `and we can construct a directed cycle in` D: `begin by letting` $u_0 \in N$ `and then recursively let` $u_{i+1} \in N$ `such that` $(u_{i+1}, u_i) \in A(D)$. `Since` N `is finite we will find some` $u_m = u_n$ `with` $m > n$, `and hence we will find that` $u_n, u_{m-1}, u_{m-2}, \ldots, u_{n+2}, u_{n+1}, u_n$ `is a directed cycle.`

  □

  To find a canonical ordering of the vertices of a directed graph takes time $O(n^2)$. However, the algorithm presented has time $O(n^3)$. In step (2), which is repeated up to n times, we must evaluate up to n vertices as candidates for v_t, and for each such candidate, we must compare it with each other vertex of N (of which there are n), and so we find that the algorithm's running time is $O(n^3)$. Even if we are more careful in our counting, and note that there are only $n - t$ vertices in N at each iteration, we still find that the number of operations to be performed is $O(n^3)$:

$$\sum_{t=0}^{n-1}(n-t)^2 \;=\; \frac{2n^3+3n^2+n}{6} \;=\; O(n^3)$$

In order to achieve an upper bound of $O(n^2)$, we must devise a slightly more clever algorithm:

1. Let $t=0$ and $N=V(D)$.
2. For each $v \in N$, construct the set $T_v = \{u \in N | (u,v) \in A(D)\}$.
3. Let $v_t \in N$ be a vertex such that $T_{v_t} = \emptyset$, assuming that such a vertex exists.
   If no such vertex exists, then proceed to (4).
   Otherwise let $N = N \setminus \{v_t\}$, and for each $u \in N$ let $T_u = T_u \setminus \{v_t\}$.
   Let $t = t+1$ and then repeat (3).
4. If $N = \emptyset$, then $v_0, v_1, \ldots, v_{n-1}$ is a canonical ordering.
   Otherwise, $N \neq \emptyset$, and we can construct a directed cycle in D: begin by letting $u_0 \in N$ and then recursively let $u_{i+1} \in N$ such that $(u_{i+1}, u_i) \in A(D)$. Since N is finite we will find some $u_m = u_n$ with $m > n$, and hence we will find that $u_n, u_{m-1}, u_{m-2}, \ldots, u_{n+2}, u_{n+1}, u_n$ is a directed cycle.

In this improved algorithm, we find that step (2) takes time $O(n^2)$. And each iteration of step (3) takes times $O(n)$; with up to n iterations, step (3) therefore takes time $O(n^2)$.

- Bellman's Algorithm:

  The input to Bellman's algorithm is an acyclic directed graph D with non-negative arc-weights, a canonical ordering $v_0, \ldots, v_{n-1}$ of the vertices, and a root vertex v.

  The output of the algorithm is a spanning tree of D rooted at v, showing the least-weight directed paths from v to each other vertex.

1. Suppose that $v = v_k$ in the canonical ordering.
   Let $d_i = \begin{cases} 0 & \text{if } i = k \\ \infty & \text{if } i < k \end{cases}$
   Let $t = k+1$.
2. Find $j < t$ such that $(v_j, v_t) \in A(D)$ and $d_j + wt(v_j, v_t) \leq d_h + wt(v_h, v_t)$ for all $h < t$ such that $(v_h, v_t) \in A(D)$.
   Once found, let $d_t = d_j + wt(v_j, v_t)$ and let $p_t = j$; if no such j can be found, then let $d_t = \infty$. Let $t = t+1$. If $t < n$ then repeat (2).

Bellman's algorithm takes time $O(n^2)$. The key to verifying this is to note that finding j in step (2) can be implemented as follows:

Let $j = 0$.
For $i = 0, 1, \ldots, t-1$
  if $(v_i, v_t) \in A(D)$
    if $d_i + wt(v_i, v_t) < d_j + wt(v_j, v_t)$
      let $j = i$.