18.3
Shortest Paths in DAGs

Shortest Paths in a DAG

Single-Source Shortest Path Problems

Input A directed acyclic graph $G = (V, E)$ with arbitrary (including negative) edge lengths. For edge $e = (u, v)$, $\ell(e) = \ell(u, v)$ is its length.

1. Given nodes s, t find shortest path from s to t.
2. Given node s find shortest path from s to all other nodes.

Simplification of algorithms for DAGs

1. No cycles and hence no negative length cycles! Hence can find shortest paths even for negative length edges
2. Can order nodes using topological sort

Shortest Paths in a DAG

Single-Source Shortest Path Problems

Input A directed acyclic graph $G = (V, E)$ with arbitrary (including negative) edge lengths. For edge $e = (u, v)$, $\ell(e) = \ell(u, v)$ is its length.

1. Given nodes s, t find shortest path from s to t.
2. Given node s find shortest path from s to all other nodes.

Simplification of algorithms for DAGs

1. No cycles and hence no negative length cycles! Hence can find shortest paths even for negative length edges
2. Can order nodes using topological sort

Algorithm for DAGs

1. Want to find shortest paths from s. Ignore nodes not reachable from s.
2. Let $s = v_1, v_2, v_{i+1}, \ldots, v_n$ be a topological sort of G

Observation:

1. shortest path from s to v_i cannot use any node from $v_{i+1}, \ldots, v_n$
2. can find shortest paths in topological sort order.

Algorithm for DAGs

1. Want to find shortest paths from s. Ignore nodes not reachable from s.
2. Let $s = v_1, v_2, v_{i+1}, \ldots, v_n$ be a topological sort of G

Observation:

1. shortest path from s to v_i cannot use any node from $v_{i+1}, \ldots, v_n$
2. can find shortest paths in topological sort order.

Algorithm for DAGs

for $i = 1$ `to` n **do**
  $d(s, v_i) = \infty$
$d(s, s) = 0$

for $i = 1$ `to` $n - 1$ **do**
  for `each edge` (v_i, v_j) `in` $\mathrm{Adj}(v_i)$ **do**
    $d(s, v_j) = \min\{d(s, v_j), d(s, v_i) + \ell(v_i, v_j)\}$

return $d(s, \cdot)$ `values computed`

Correctness: induction on i and observation in previous slide.

Running time: $O(m + n)$ time algorithm! Works for negative edge lengths and hence can find longest paths in a DAG.

Bellman-Ford and DAGs

Bellman-Ford is based on the following principles:

- The shortest walk length from s to v with at most k hops can be computed via dynamic programming
- G has a negative length cycle reachable from s iff there is a node v such that shortest walk length reduces after n hops.

We can find hop-constrained shortest paths via graph reduction.

Given $G = (V, E)$ with edge lengths $\ell(e)$ and integer k construction new layered graph $G' = (V', E')$ as follows.

- $V' = V \times \{0, 1, 2, \ldots, k\}$.
- $E' = \{((u, i), (v, i + 1) \mid (u, v) \in E, 0 \leq i < k\}$, $\ell((u, i), (v, i + 1)) = \ell(u, v)$

Lemma 18.1.

Shortest path distance from $(u, 0)$ to (v, k) in G' is equal to the shortest walk from u to v in G with exactly k edges.

Layered DAG: Figure

THE END

...

(for now)