Lakehead
UNIVERSITY

COMP 4433: Algorithm Design and Analysis

Dr. Y. Gu

March 1, 2023 (Lecture 12)

Midterm Help Info

- TA Khaled will be available on Friday morning (March 3), 10am - 12pm.
- Instructor’s Q&A session will be on Saturday afternoon (March 4) 1pm- 3pm.
- TA Mohammad will be available on Sunday (March 5) 6-8pm.

All zoom info is posted in the announcement.

Midterm March 6, 7-8:30pm in class. Open book.

Greedy Algorithm and Shortest Paths (Part I)

Shortest-Paths Problem

In the shortest-paths problem, we are given a weighted, directed graph $G = (V, E)$, with weight function $w : E \rightarrow R$. The weight $w(p)$ of $p = \langle v_0, v_1, \ldots, v_k \rangle$ is the sum of the weights of its constituent edges:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i).$$

The shortest-path weight δ(u, v) from u to v is defined as

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \overset{p}{\leadsto} v\} & \text{if there is a path from } u \text{ to } v, \\ \infty & \text{otherwise} \end{cases}$$

A shortest path from vertex u to vertex v is defined as any path p with weight w(p) = δ(u, v).

Single Source Shortest Paths

For the shortest-path problem, we may consider single-source (or single destination, respectively) shortest path which finds a shortest path from a given source to each vertex (or to a given destination from each vertex, respectively).

We also can consider single-pair shortest path which finds a shortest path from a source vertex v to a vertex u. However, all the known algorithms for single-pair shortest path have the same worst-case asymptotic running time as the best single-source algorithms. So we mainly consider the single-source short path problem.

Single Source Shortest Paths

For the shortest-path problem, we may consider single-source (or single destination, respectively) shortest path which finds a shortest path from a given source to each vertex (or to a given destination from each vertex, respectively).

We also can consider single-pair shortest path which finds a shortest path from a source vertex v to a vertex u. However, all the known algorithms for single-pair shortest path have the same worst-case asymptotic running time as the best single-source algorithms. So we mainly consider the single-source short path problem.

Optimal Structure

To use greedy algorithm, we need some optimal substructure of the shortest path problem. We have the following lemma.

Lemma Suppose a directed graph $G = (V, E)$ with weight function $w : E \longrightarrow R$ is given. Let $p = \langle v_0, v_1, . . . , v_k\rangle$ be a shortest path from vertex v_0 to vertex v_k. For any i and j, $0 \leq i \leq j \leq k$, let $p_{ij} = \langle v_i , v_{i+1}, . . . , v_j \rangle$ be the subpath of p from v_i to v_j . Then p_{ij} is a shortest path from v_i to v_j .

Proof. If p_{ij} is not a shortest path, then there is a shortest path $p'_{ij} = \langle v_i , v'_{i+1}, . . . , v'_{j-1} , v_j \rangle$ such that $w(p'_{ij}) < w(pij)$. But $p' = \langle v_0, v_1, . . . , v_i , v'_{i+1}, . . . , v'_{j-1} , v_j , . . . v_k\rangle$ is a path from v_0 to v_k with $w(p') < w(p)$ which is impossible. □

The Bellman-Ford Algorithm

In some applications of the shortest paths problem, the graph may include some edges with negative weights.

Consider the single-source shortest path problem. If the graph contains a negative-weight cycle reachable from the source vertex s, then the shortest path weight are not well defined.

Because the path can repeat the cycle any number of times, that makes the weight smaller than any given number.

The Bellman-Ford Algorithm

So when we treat a graph with negative weight edges, we only consider those graphs that **do not** contain any **negative-weight cycle**.

A shortest path in a graph contains **no cycle.** If there is a cycle among the path with no-negative weight, then we can remove the cycle. In fact, in this case the cycle cannot has positive weight, otherwise the path would not be the shortest.

The Bellman-Ford Algorithm

For the **single-source shortest-path problem** of a weighted graph G = (V, E), we are finding a shortest-paths tree G′ = (V ′ , E′) rooted at the source vertex s, where V ′ ⊆ V, E′ ⊆ E, satisfying

1. V ′ is the set of vertices reachable from s in G,
2. G′ forms a rooted tree with roots, and
3. for all v ∈ V ′ , the unique simple path from s to v in G′ is a shortest path from s to v in G

The Bellman-Ford Algorithm

To compute shortest path, we maintain two attributes for a vertex v in the graph.

For each vertex $v \in G.V$, we define a **predecessor** $v.\pi$ that is either another vertex or NIL.

In the shortest path algorithm we set the π attributes so that the chain of predecessors originating at a vertex v runs backwards along a shortest path from s to v.

The Bellman-Ford Algorithm

We also define the predecessor subgraph $G_\pi = (V_\pi, G_\pi)$ induced by the π values. In this subgraph, V_π is the set of vertices of G with non-NIL predecessors, plus the source s:

$$V_\pi = \{v \in V : v.\pi \neq \text{NIL}\} \cup \{s\}.$$

The directed edge set E_π is the set of edges induced by the π values for vertices in V_π:

$$E_\pi = \{(v.\pi, v) \in E : v \in V_\pi - \{s\}\}.$$

Another attribute for a vertex v is v.d which is an upper bound on the weight of a shortest path from source s to v. We call v.d a shortest-path estimate.

The Bellman-Ford Algorithm

We can use the following Θ(V)-time procedure to initialize these attributes.

```
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

The Bellman-Ford Algorithm

The next procedure of relaxing an edge (u, v) consists of testing whether we can improve the shortest path to v found so far by going through u, and updating v.d and v.π.

RELAX(u, v, w)

if $v.d > u.d + w(u, v)$
  $v.d = u.d + w(u, v)$
  $v.\pi = u$

The Bellman-Ford Algorithm

The Bellman-Ford algorithm solves the single-source shortest path problem in general case in which edge weights may be negative.

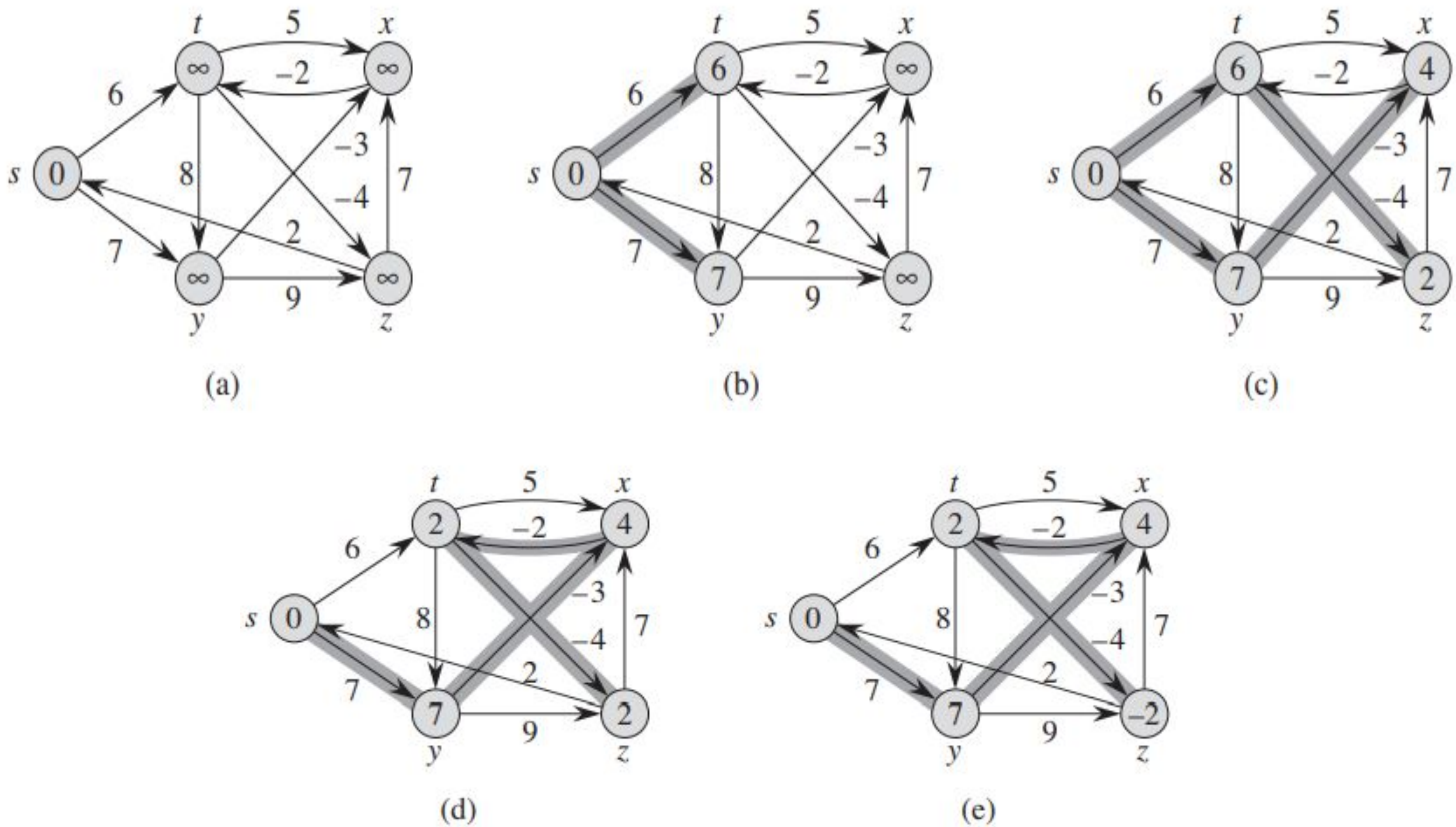
The algorithm returns a boolean value indicating if there is a negative cycle that is reachable from the source (that is, if the shortest path tree exists or not).

The Bellman-Ford Algorithm

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if v.d > u.d + w(u, v)
7          return FALSE
8  return TRUE
```

The running time for this algorithm is $O(|V|\,|E|)$. The initialization takes $\Theta(|V|)$ time, the nested for loops in line 2 execute Relax $\Theta((|V| - 1)|E|)$ times. The loop in line 5 takes $O(|E|)$ time.

Example



Correctness

Lemma 2 [Triangle inequality]

Let G be a weighted directed graph with source s. Then for all edges $(u, v) \in E$, we have $\delta(s, v) \leq \delta(s, u) + w(u, v)$.

Proof. The proof is simple and omitted. □

Correctness

Lemma 3 [Upper-bound property]

Let G be a weighted directed graph with source s. Suppose that G is initialized by Initialize-Single-Source(G, s). Then $v.d \geq \delta(s, v)$ for all $v \in V$. Moreover, once $v.d$ achieves its lower bound $\delta(s, v)$, it never changes.

Proof.

We prove the invariant $v.d \geq \delta(s, v)$ by induction. For the basis, $v.d = \infty$ after initialization for all $v \in V - \{s\}$, so $v.d \geq \delta(s, v)$, and $s.d = 0 \geq \delta(s, s)$ (note that $\delta(s, s) = -\infty$ if s is on a negative cycle).

Correctness

For the inductive step, consider the relaxation of an edge (u,v). By induction hypothesis, $x.d \geq \delta(s, x)$ for all $x \in V$ prior to the relaxation.

The only d value that may change is v.d. If it changes, we have

$v.d = u.d + w(u, v)$

$\geq \delta(s, u) + w(u, v)$ (by the inductive hypothesis)

$\geq \delta(s, v)$ (by the triangle inequality).

We have just shown that $v.d \geq \delta(s, v)$, and it cannot increase because because relaxation steps do not increase d values.

□

Correctness

Lemma 4[Convergence property]

Let G be a weighted directed graph with source s. Let $s \rightsquigarrow u \rightarrow v$ be a shortest path in G for some vertices u, $v \in V$. Suppose that G is initialized by Initialize-Single-Source(G, s) and then a sequence of relaxation steps that includes the call Relax(u, v, w) is executed on the edges of G. If u.d = δ(s, u) at any time prior to the call, then v.d = δ(s, v) at all times after the call

Correctness

Proof.

If, just prior to relaxing edge (u, v), we have $v.d > u.d + w(u, v)$, then $v.d = u.d + w(u, v)$ afterward, we have $v.d \leq u.d + w(u, v)$. Otherwise, $v.d \leq u.d + w(u, v)$ and v.d and u.d will be unchanged. By the upper-bound property, if $u.d = \delta(s, u)$ at some point prior to relaxing edge (u, v), then this equality holds thereafter. In particular, after relaxing edge (u, v), we have

$$
\begin{aligned}
v.d &\leq u.d + w(u, v) \\
&= \delta(s, u) + w(u, v) \\
&= \delta(s, v) \text{ (by Lemma 1).}
\end{aligned}
$$

However, by the upper-bound property, $v.d \geq \delta(s, v)$. Therefore,

$v.d = \delta(s, v)$ □

Correctness

Lemma 5 [Path-relaxation property]

Let G be a weighted directed graph with source s. Consider any shortest path $p = \langle v_0, v_1, \ldots, v_k \rangle$ from $s = v_0$ to v_k. If G is initialized by Initialize-Single-Source(G, s) and then a sequence of relaxation steps occurs that includes, in order, relaxing the edges $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$, then $v_k.d = \delta(s, v_k)$ after these relaxations and at all times afterward.

Correctness

Proof.

We show by induction that after the ith edge of path p is relaxed, we have $v_i.d = \delta(s, v_i)$.

For the basis, i = 0, and before any edges of p have been relaxed, we have $v_0.d = 0 = \delta(s, s)$.

By the upper-bound property, the value of s.d never changes after initialization.

For inductive step, we assume that $v_{i-1}.d = \delta(s, v_{i-1})$. By the convergence property, after relaxing this edge, we have $v_i.d = \delta(s, v_i)$, and this equality in maintained at all times thereafter.

□

Correctness

Lemma 6 Let G be a weighted directed graph with source s, and assume that G contains no negative-weight cycle that are reachable from s. Then after execute Bellman-Ford algorithm, v.d = δ(s, v) for all vertices v that are reachable from s.

Proof. Consider any vertex v that is reachable from s, and let $p = \langle v_0, v_1, \ldots, v_k \rangle$, where $v_0 = s$ and $v_k = v$, be any shortest path from s to v. Because shortest path is simple, p has at most $|V| - 1$ edges, so $k \leq |V| - 1$. Each of the $|V| - 1$ iterations relaxed all $|E|$ edges. Among the edges relaxed in ith iteration, for i = 1, 2, . . . , k, is (v_{i-1}, v_i). By the path-relaxation property, $v.d = v_k.d = \delta(s, v_k) = \delta(s, v)$. □

Correctness

Theorem 7 [Correctness of the Bellman-Ford algorithm]

Let Bellman-Ford be run on a weighted, directed graph $G = (V, E)$ with source s and weight function $w : E \rightarrow R$. If G contains no negative-weight cycles that are reachable from s, then the algorithm returns TRUE, we have $v.d = \delta(s, v)$ for all vertices $v \in V$, and the predecessor subgraph $G\pi$ is a shortest-paths tree rooted at s. If G does contain a negative-weight cycle reachable from s, then the algorithm returns FALSE.

Correctness

Proof:

Suppose that graph G contains no negative-weight cycles that are reachable from the source s.

We first prove the claim that at termination, $v.d = \delta(s, v)$ for all vertices $v \in V$. If vertex v is reachable from s, then Lemma 6 proves this claim. If v is not reachable from s, then $v.d = \infty = \delta(s, v)$ by upper-bound property. Thus, the claim is proven. Lemma 1, along with the claim, implies that G_π is a shortest-paths tree. Now we use the claim to show that Bellman-Ford returns TRUE. At termination, we have for all edges $(u, v) \in E$,

$$v.d = \delta(s, v) \le \delta(s, u) + w(u, v) \text{ (by the triangle inequality)}$$

$$= u.d + w(u, v).$$

Correctness

Now, suppose that graph G contains a negative-weight cycle that is reachable from the source s; let this cycle be $c = \langle v_0, v_1, \ldots, v_k \rangle$, where $v_0 = v_k$. Then,

$$\sum_{i=1}^{k} w(v_{i-1}, v_i) \quad < \quad 0. \qquad (3)$$

Assume for the purpose of contradiction that the Bellman-Ford algorithm returns TRUE. Thus, $v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$ for $i = 1, 2, \ldots, k$. Summing the inequalities around cycle c gives us

$$\begin{aligned} \sum_{i=1}^{k} v_j.d &\leq \sum_{i=1}^{k} (v_{i-1}.d + w(v_{i-1}, v_i)) \\ &= \sum_{i=1}^{k} v_{i-1}.d + \sum_{i=1}^{k} w(v_{i-1}, v_i). \end{aligned}$$

Correctness

Since $v_0 = v_k$, each vertex in c appears exactly once in each of the summations $\sum_{i=1}^{k} v_i.d$ and $\sum_{i=1}^{k} v_{i-1}.d$, so

$$\sum_{i=1}^{k} v_i.d = \sum_{i=1}^{k} v_{i-1}.d.$$

Moreover, v_i.d is finite for i = 1, 2, . . . , k. Thus,

$$0 \leq \sum_{i=1}^{k} w(v_{i-1}, v_i),$$

which contradicts inequality (3). We conclude that the Bellman-Ford algorithm returns TRUE if graph G contains no negative-weight cycles reachable from the source, and FALSE otherwise. □

After Class

- After class: Part VI 24.1 and 24.2