Design and Analysis of Algorithms

Mohammad GANJTABESH

mgtabesh@ut.ac.ir

School of Mathematics, Statistics and Computer Science,
University of Tehran,
Tehran, Iran.

Graph Theoretical Problems

- Basec Definitions
- Graph Representation
- Graph Traversal (BFS, DFS)
- Topological Sort
- Strongly Connected Components
- Shortest Paths
  - Single-Source All Destination (Dijkstra and Bellman-Ford Algorithms)
  - All-Pairs (Matrix Multiplication, Floyd-Warshall, and Johnson's Algorithms)
- Minimum Spanning Tree (Kruskal, Prim)

Shortest Paths

Definition

Suppose that we are given a weighted directed graph $G = (V, E)$, with weight function $w : E \mapsto R$. The weight of path $p = \langle v_0, v_1, \cdots, v_k \rangle$ is the sum of the weights of its edges:

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

We define the shortest path weight from u to v by

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

A **shortest path** from vertex u to vertex v is then defined as any path p with weight $w(p) = \delta(u, v)$.

- Single-Source All Destination shortest path (Bellman-Ford, Dijkestra)
- All-Pairs shortest path (Floyd-Warshall, Johnson)

Shortest Paths

Lemma

Given a weighted directed graph $G = (V, E)$ with weight function $w : E \mapsto R$, let $p = \langle v_1, v_2, \cdots, v_k \rangle$ be a shortest path from vertex v_1 to vertex v_k and, for any i and j such that $1 \leq i \leq j \leq k$, let $p_{ij} = \langle v_i, v_{i+1}, \cdots, v_j \rangle$ be the subpath of p from vertex v_i to vertex v_j. Then, p_{ij} is a shortest path from v_i to v_j.

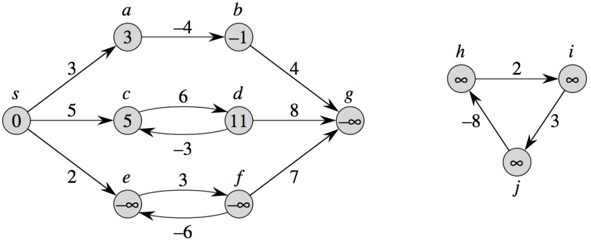
Proof.

- Decompose path p into $v_1 \overset{p_{1i}}{\rightsquigarrow} v_i \overset{p_{ij}}{\rightsquigarrow} v_j \overset{p_{jk}}{\rightsquigarrow} v_k$.
- $w(p) = w(p_{1i}) + w(p_{ij}) + w(p_{jk})$.
- Assume that there is a path p'_{ij} from v_i to v_j with weight $w(p'_{ij}) < w(p_{ij})$.
- Then, $v_1 \overset{p_{1i}}{\rightsquigarrow} v_i \overset{p'_{ij}}{\rightsquigarrow} v_j \overset{p_{jk}}{\rightsquigarrow} v_k$ is a path from v_1 to v_k whose weight $w(p_{1i}) + w(p'_{ij}) + w(p_{jk})$ is less than $w(p)$ (contradiction).

$\square$

Shortest Paths

- **Negative-weight edges:** If $G = (V, E)$ contains no negative-weight cycles reachable from s, then for all $v \in V$, the shortest path weight $\delta(s, v)$ remains well defined. If there is a negative-weight cycle reachable from s, the shortest-path weights are not well defined and we assume $\delta(s, v) = -\infty$.



- **Cycles:** Can a shortest path contain a cycle? No, since removing the cycle from the path produces a path with the same source and destination vertices and a lower path weight. Since any acyclic path in $G = (V, E)$ contains at most $|V|$ distinct vertices (and so at most $|V| - 1$ edges), Therefore, we can restrict our attention to shortest paths of at most $|V| - 1$ edges.

Shortest Paths: Intialization

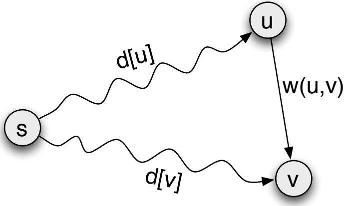
For each vertex $v \in V$, assume $d[v]$ be an upper bound on the weight of a shortest path from source s to v (we call $d[v]$ a shortest path estimate).
For initialize the following procedure is used:

INITIALIZE-SINGLE-SOURCE(G, s)

for each vertex $v \in V[G]$
    do $d[v] \leftarrow \infty$
        $\pi[v] \leftarrow$ NIL
$d[s] \leftarrow 0$

Shortest Paths: Relaxation

The process 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, if so, updating $d[v]$ and $\pi[v]$.



The following code performs a relaxation step on edge (u, v):

RELAX(u, v, w)

if $d[v] > d[u] + w(u, v)$
  then $d[v] \leftarrow d[u] + w(u, v)$
    $\pi[v] \leftarrow u$

Properties of shortest paths and relaxation

- **Triangle inequality:** For any edge $(u, v) \in E$, we have $\delta(s, v) \leq \delta(s, u) + w(u, v)$.
- **Upper-bound property:** We always have $d[v] \geq \delta(s, v)$ for all vertices $v \in V$, and once $d[v]$ achieves the value $\delta(s, v)$, it never changes.
- **No-path property:** If there is no path from s to v, then we always have $d[v] = \delta(s, v) = \infty$.
- **Convergence property:** If $s \rightsquigarrow u \to v$ is a shortest path in G for some $u, v \in V$, and if $d[u] = \delta(s, u)$ at any time prior to relaxing edge (u, v), then $d[v] = \delta(s, v)$ at all times afterward.
- **Path-relaxation property:** If $p = \langle v_0, v_1, \cdots, v_k \rangle$ is a shortest path from $s = v_0$ to v_k, and the edges of p are relaxed in the order (v_0, v_1), (v_1, v_2), $\cdots$, (v_{k-1}, v_k), then $d[v_k] = \delta(s, v_k)$.

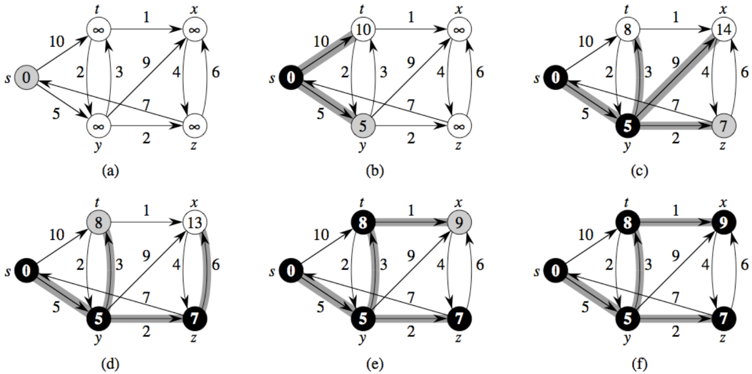
Shortest Paths: Dijkstra's algorithm

Dijkstra's algorithm maintains a set S of vertices whose final shortest path weights from the source s have already been determined. The algorithm repeatedly selects the vertex $u \in V - S$ with the minimum shortest path estimate, adds u to S, and relaxes all edges leaving u.

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S ← Ø
Q ← V[G]
while Q ≠ Ø
      do u ← EXTRACT-MIN(Q)
         S ← S ∪ {u}
         for each vertex v ∈ Adj[u]
             do RELAX(u, v, w)
```

Dijkstra's algorithm always chooses the lightest vertex in $V - S$ to add to set S, so it uses a greedy strategy.

Shortest Paths: Dijkstra's algorithm



Analysis

If the Q is implemented by an array, then the time complexity of Dijkstra's algorithm becomes $O(|V|^2)$. (How we can achieved to better performance?)

Shortest Paths: Dijkstra's algorithm

Theorem

Dijkstra's algorithm, run on a weighted directed graph $G = (V, E)$ with non-negative weight function w and source s, terminates with $d[u] = \delta(s, u)$ for all vertices $u \in V$.

Proof.

It suffice to prove that: for each vertex $u \in V$, we have $d[u] = \delta(s, u)$ at the time when u is added to set S.

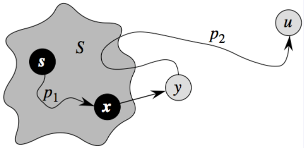
- **Initialization:** Initially, $S = \emptyset$, and so the statement is trivially true.
- **Maintenance:**
  - Let u be the first vertex for which $d[u] \neq \delta(s, u)$ when it is added to set S.
  - There must be some path from s to u (otherwise $d[u] = \delta(s, u) = \infty$ which violates $d[u] \neq \delta(s, u)$).
  - Because there is at least one path, there is a shortest path p from s to u. Prior to adding u to S, path p connects a vertex in S, namely s, to a vertex in $V - S$, namely u.
  - Let the first vertex along p be y, such that $y \in V - S$, and let $x \in S$ be y's predecessor.
  - Continue in next page...

□

Shortest Paths: Dijkstra's algorithm

Proof.

- **Maintenance (cont.):**
  - The path p can be decomposed as $s \overset{p_1}{\leadsto} x \to y \overset{p_2}{\leadsto} u$.



  - Because y occurs before u on a shortest path from s to u and all edge weights are nonnegative, we have $\delta(s,y) \le \delta(s,u)$ which implies that $d[y] = \delta(s,y) \le \delta(s,u) \le d[u]$.
  - But because both vertices u and y were in $V - S$ when u was chosen, we have $d[u] \le d[y]$.
  - So we have $d[y] = \delta(s,y) = \delta(s,u) = d[u]$.
  - Consequently, $d[u] = \delta(s,u)$, which contradicts our choice of u.
  - Therefore, we conclude that $d[u] = \delta(s,u)$ when u is added to S.
- **Termination:** At termination, $Q = \emptyset$ which implies that $S = V$. Thus, $d[u] = \delta(s,u)$ for all vertices $u \in V$.

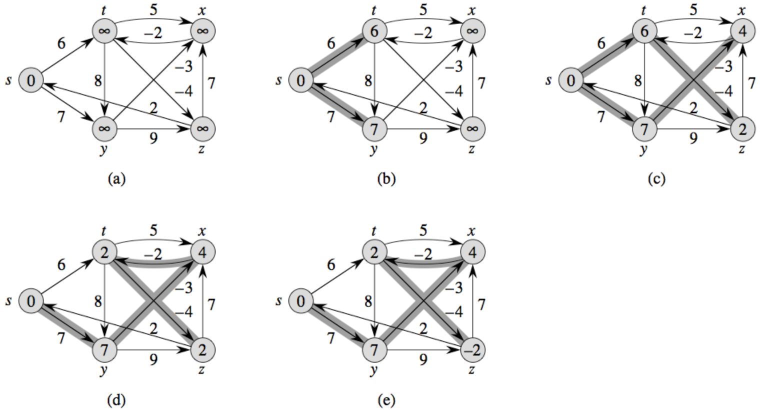
$\square$

Shortest Paths: Bellman-Ford algorithm

Given a weighted directed graph $G = (V, E)$ with source s and weight function $w : E \mapsto R$, the Bellman-Ford algorithm returns a boolean value indicating whether or not there is a negative-weight cycle that is reachable from the source. If there is such a cycle, the algorithm indicates that no solution exists. Otherwise, the algorithm produces the shortest paths and their weights.

```
BELLMAN-FORD(G, w, s)
    INITIALIZE-SINGLE-SOURCE(G, s)
    for i ← 1 to |V[G]| − 1
        do for each edge (u, v) ∈ E[G]
               do RELAX(u, v, w)
    for each edge (u, v) ∈ E[G]
        do if d[v] > d[u] + w(u, v)
             then return FALSE
    return TRUE
```

Shortest Paths: Bellman-Ford algorithm



Analysis

The time complexity of BELLMAN-FORD algorithm is $O(|V| \times |E|)$.

(How?)

Shortest Paths: Bellman-Ford algorithm

Lemma

Let $G = (V, E)$ be a weighted, directed graph with source s and weight function $w : E \mapsto \mathbb{R}$, and assume that G contains no negative-weight cycles that are reachable from s. Then, after $|V| - 1$ iterations of the for loop of lines $2 - 4$ of BELLMAN-FORD, we have $d[v] = \delta(s, v)$ for all vertices v that are reachable from s.

Proof.

- Consider any vertex v that is reachable from s.
- Let $p = \langle v_0, v_1, \cdots, v_k \rangle$, where $v_0 = s$ and $v_k = v$, be any acyclic shortest path from s to v.
- Path p has at most $|V| - 1$ edges, and so $k \leq |V| - 1$.
- Each of the $|V| - 1$ iterations of the for loop of lines $2 - 4$ relaxes all E edges.
- Among the edges relaxed in the ith iteration, for $i = 1, 2, \cdots, k$, is the edge (v_{i-1}, v_i).
- By the path-relaxation property, therefore, $d[v] = d[v_k] = \delta(s, v_k) = \delta(s, v)$.

$\square$

Shortest Paths: Bellman-Ford algorithm

Theorem

Let BELLMAN-FORD be run on a weighted, directed graph $G = (V, E)$ with source s and weight function $w : E \mapsto \mathbb{R}$. If G contains no negative-weight cycles that are reachable from s, then the algorithm returns TRUE, we have $d[v] = \delta(s, v)$ for all vertices $v \in V$. If G does contain a negative-weight cycle reachable from s, then the algorithm returns FALSE.

Proof.

- Suppose that graph G contains no negative-weight cycles that are reachable from s.
  - If vertex v is reachable from s, then the previous lemma proves the claim.
  - If v is not reachable from s, then the claim follows from the no-path property.
  - Now we use the claim to show that BELLMAN-FORD returns TRUE.
  - At termination, we have for all edges $(u, v) \in E$,

$$\begin{aligned} d[v] &= \delta(s, v) \\ &\leq \delta(s, u) + w(u, v) \quad \text{(by the triangle inequality)} \\ &= d[u] + w(u, v), \end{aligned}$$

  and so none of the tests in line 6 causes BELLMAN-FORD to return FALSE. It therefore returns TRUE.

Shortest Paths: Bellman-Ford algorithm

Proof.

- Conversely, suppose that G contains a negative-weight cycle that is reachable from s.
  - Let this cycle be $c = \langle v_0, v_1, \cdots, v_k \rangle$, where $v_0 = v_k$.
  - Then, $\sum_{i=1}^{k} w(v_{i-1}, v_i) < 0$.
  - Assume for the purpose of contradiction that the Bellman-Ford algorithm returns TRUE.
  - Thus, $d[v_i] \leq d[v_{i-1}] + w(v_{i-1}, v_i)$ for $i = 1, 2, \cdots, k$.
  - Summing the inequalities around cycle c gives us

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

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

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

    which is contradiction.

$\square$

Exercises

1. Prove all the properties of shortest paths and relaxation, i.e.:
   - Triangle inequality
   - Upper-bound property
   - No-path property
   - Convergence property
   - Path-relaxation property
2. Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \mapsto \mathbb{R}$. Give an $O(|V|.|E|)$-time algorithm to find, for each vertex $v \in V$, the value $\delta^*(v) = \min_{u \in V}\{\delta(u, v)\}$.
3. Suppose that a weighted, directed graph $G = (V, E)$ has a negative-weight cycle. Give an efficient algorithm to list the vertices of one such cycle. Prove that your algorithm is correct.
4. Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \mapsto \{1, 2, \cdots, W\}$ for some positive integer W, and assume that no two vertices have the same shortest-path weights from source vertex s. Now suppose that we define an unweighted, directed graph $G^{'} = (V \cup V^{'}, E^{'})$ by replacing each edge $(u, v) \in E$ with $w(u, v)$ unit-weight edges in series. How many vertices does $G^{'}$ have? Now suppose that we run a breadth-first search on $G^{'}$. Show that the order in which vertices in V are colored black in the breadth-first search of $G^{'}$ is the same as the order in which the vertices of V are extracted from the priority queue in line 5 of DIJKSTRA when run on G.
5. Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \mapsto \{0, 1, \cdots, W\}$ for some nonnegative integer W . Modify Dijkstras algorithm to compute the shortest paths from a given source vertex s in $O(W|V| + |E|)$ time.