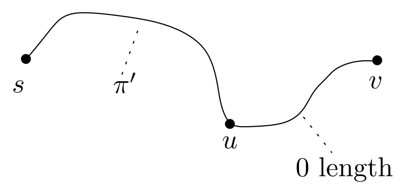
CSCI3160: Regular Exercise Set 9

Prepared by Yufei Tao

Problem 1*. Prove the correctness of Dijkstra's algorithm (when the edges have non-negative weights).

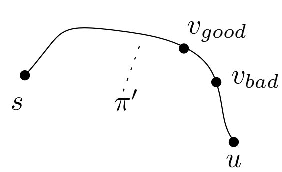
Solution. We argue that, *every time a vertex v is removed from S (note: each time the algorithm removes from a vertex v from S with the smallest $dist(v)$; see the algorithm's description in the lecture slides), we must have $dist(v) = spdist(v)$*. We will do so by induction on the order that the vertices are removed. The base step, which corresponds to removing the source vertex s, is obviously correct. Next, assuming correctness on all the vertices already removed, we will prove the statement on the vertex v removed *next*.

Let π be an arbitrary shortest path from s to v. Identify the first vertex u on π (in the direction from s to v) such that $spdist(u) = spdist(v)$. In other words, all the edges on π between u and v have weight 0. Let π' be the prefix of π that ends at u. Note that π' must be a shortest path from s to u.



Claim 1: When v is to be removed from S, all the vertices on π' — except possibly u — must have been removed from S.

Proof of Claim 1: Suppose that the claim is not true. Define v_{bad} as the first vertex on π' that is still in S when v is to be removed from S. Let v_{good} be the vertex right before v_{bad} on π; note that v_{good} definitely exists because v_{bad} cannot be s. By how u is defined, we must have $spdist(v_{bad}) < spdist(u) = spdist(v)$.

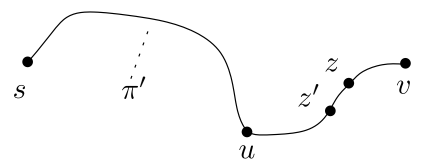


By our inductive assumption, when v_{good} was removed from S, we had $dist(v_{good}) = spdist(v_{good})$. We must have relaxed the edge (v_{good}, v_{bad}), after which we must have

$$\begin{aligned} dist(v_{bad}) &= dist(v_{good}) + w(v_{good}, v_{bad}) \\ &= spdist(v_{good}) + w(v_{good}, v_{bad}) = spdist(v_{bad}). \end{aligned}$$

The value $dist(v_{bad})$ never increases during the algorithm. Hence, when v is to be removed from S, we must have $dist(v_{bad}) = spdist(v_{bad}) < spdist(u) = spdist(v) \le dist(v)$. But this contradicts the fact that v has the smallest *dist*-value among all the vertices still in S. □

Consider the moment when v is to be removed from S; define z as the first vertex on π that has *not* been removed from S. Note that z is well defined because v itself is still in S at this moment.



Claim 2: When v is to be removed from S, $dist(z) = spdist(z)$.

Proof of Claim 2: Let z' be the vertex right before z on π. Note that z' is well defined because z cannot be earlier than u on π (Claim 1) and z cannot be s.

By our inductive assumption, when z' was removed from S, we had $dist(z') = spdist(z')$. We must have relaxed the edge (z', z), after which we must have

$$dist(z) = dist(z') + w(z', z) = spdist(z') = spdist(z).$$

□

It now follows that, when v is to be removed from S, we have $dist(v) \leq dist(z) = spdist(z) = spdist(v)$. As $dist(v)$ cannot be larger than $spdist(v)$, we must have $dist(v) = spdist(v)$.

Problem 2. Consider again your proof for Problem 1. Point out the place that requires edge weights to be non-negative.

Solution. We used this assumption in the proof of Claim 1: look for the sentence: "By how u is defined, we must have $spdist(v_{bad}) < spdist(u) = spdist(v)$".

Problem 3. Consider a directed simple graph $G = (V, E)$ where each edge $e \in E$ has an arbitrary weight $w(e)$ (which can be negative). It is known that G does not have negative cycles. Prove: given any vertices $s, t \in V$, at least one shortest path from s to t is a simple path (i.e., no vertex appears twice on the path).

Remark: This implies that the path must have at most $|V| - 1$ edges.

Solution. Let π be a shortest path from s to t that uses the *least* number of edges. We will prove that π must be a simple path. Let us list all the vertices on the path from s to t as $u_1, u_2, ..., u_t$, where $u_1 = s$, $u_t = t$, and $t - 1$ is the number of edges on π. If π is not a simple path, then there must exist $1 \leq i < j \leq t$ with $u_i = u_j$. Thus, the sub-path $u_i, u_{i+1}, ..., u_{j-1}, u_j$ is a cycle. The length of the cycle must be non-negative. By removing this sub-path, we obtain another path π' from s to t: $u_1, u_2, ..., u_i, u_{j+1}, u_{j+2}, ..., t$. The new path π' cannot have a length greater than π, but uses strictly fewer edges. This contradicts the definition of π.

Problem 4* (SSSP in a DAG). Consider a simple acyclic directed graph $G = (V, E)$ where each edge $e \in E$ has an arbitrary weight $w(e)$ (which can be negative). Solve the SSSP problem on G in $O(|V| + |E|)$ time.

Solution. Let s be the source vertex. For each vertex $v \in V$, define $spdist(v)$ as the shortest path length from s to v. Also, define $\mathrm{IN}(v)$ as the set of in-neighbors of v. Observe that:

$$spdist(v) = \begin{cases} 0 & \text{if } v = s \\ \infty & \text{if } \mathrm{IN}(v) = \emptyset \\ \min_{u \in \mathrm{IN}(v)}(spdist(u) + w(u, v)) & \text{if } v \neq s \text{ and } \mathrm{IN}(v) \neq \emptyset \end{cases}$$

We can compute *spdist*(v) in $O(|V| + |E|)$ time based on a topological order of V, which can also be obtained in $O(|V| + |E|)$ time (see Prof. Tao's CSCI2100 homepage). The shortest path tree of s can then be obtained using the piggyback technique without increasing the time complexity.

Problem 5. Let $G = (V, E)$ be a simple directed graph where each edge $e \in E$ carries a weight $w(e)$, which can be negative. It is guaranteed that G has no negative cycles. Prove: given any vertices $s, t \in V$, at least one shortest path from s to t is a simple path (i.e., no vertex appears twice on the path).

Solution. Consider a shortest path π from s to t that has the least number of edges. We argue that π must be simple. Otherwise, at least one vertex v appears twice on π. Identify any two consecutive occurrences of v — call the first occurrence v_1 and the second v_2. Thus, the subpath of π from v_1 to v_2 is a cycle. As G does not have any negative cycle, that subpath must have a non-negative length. We can now remove the subpath from π to obtain another shortest path from s to t that has fewer edges than π.

Problem 6.** Let $G = (V, E)$ be a simple directed graph where the weight of an edge (u, v) is $w(u, v)$. Prove: the following algorithm correctly decides whether G has a negative cycle.

algorithm negative-cycle-detection
1. pick an arbitrary vertex $s \in V$
2. initialize *dist*$(s) = 0$ and *dist*$(v) = \infty$ for every other vertex $v \in V$
3. **for** $i = 1$ **to** $|V| - 1$
4.   relax all the edges in E
5. **for** each edge $(u, v) \in E$
6.   **if** *dist*$(v) >$ *dist*$(u) + w(u, v)$ **then**
7.   **return** "there is a negative cycle"
8. **return** "no negative cycles"

Solution. We will prove two directions.

Direction 1: If the inequality of Line 6 holds for any edge (u, v), then there must be a negative cycle. The lecture proved that, in the absence of negative cycles, Bellman-Ford's algorithm correctly finds all shortest path distances (from s) after $|V| - 1$ rounds of edge relaxations. This means that, if there are no cycles, when we come to Line 6, the value *dist*(v) must be the shortest path distance from s to v, for every $v \in V$. If Line 6 holds for some edge (u, v), however, it means that an even shorter path from s to v has just been discovered. Therefore, G must contain a negative cycle.

Direction 2: If there is a negative cycle, then the inequality of Line 6 must hold for at least one edge (u, v). Suppose that the negative cycle is $v_1 \to v_2 \to ... \to v_\ell \to v_1$. Hence:

$$w(v_\ell, v_1) + \sum_{i=1}^{\ell-1} w(v_i, v_{i+1}) \quad < \quad 0. \tag{1}$$

Assume that Line 6 does not hold on any edge in E. This indicates:

- for every $i \in [1, \ell - 1]$, $dist(v_{i+1}) \leq dist(v_i) + w(v_i, v_{i+1})$;
- $dist(v_1) \leq dist(v_\ell) + w(v_\ell, v_1)$.

These two bullets lead to:

$$\begin{aligned}\sum_{i=1}^{\ell} dist(v_i) &\leq \left(\sum_{i=1}^{\ell} dist(v_i)\right) + w(v_\ell, v_1) + \sum_{i=1}^{\ell-1} w(v_i, v_{i+1}) \\ \Rightarrow 0 &\leq w(v_\ell, v_1) + \sum_{i=1}^{\ell-1} w(v_i, v_{i+1})\end{aligned}$$

which contradicts (1).