

Network Flow Algorithms: Exercise Solutions

Yuchong Pan

December 22, 2020

1 Preliminaries: Shortest Path Algorithms

Exercise 1.1. Let i_k be the vertex selected at the k^{th} iteration of Dijkstra's algorithm. We prove by induction that at the beginning of the k^{th} iteration, $d(v) \in \{d(i_k), d(i_k) + 1, \infty\}$ for all $v \in V$ not marked yet. At the beginning of the first iteration, s is selected, and any $v \in V$ has $d(v) = \infty$; this proves the base case.

Suppose that at the beginning of the k^{th} iteration, $d(v) \in \{d(i_k), d(i_k) + 1, \infty\}$ for all $v \in V$ not marked yet. Let $(i_k, j) \in A$ such that j is not marked yet. If $d(i_k) + c(i_k, j) = d(i_k) + 1 < d(j)$, then we set $d(j) \leftarrow d(i_k) + 1$; otherwise, $d(j)$ remains the same, and hence $d(j) \in \{d(i_k), d(i_k) + 1, \infty\}$. If $d(i_{k+1}) = d(i_k)$, then we are done; otherwise, $d(i_{k+1}) = d(i_k) + 1$, and $d(v) \in \{d(i_k) + 1, \infty\} = \{d(i_{k+1}), \infty\}$ for all $v \in V$ not marked yet. This completes the induction step.

Now, consider the k^{th} iteration. If $d(i_k) + c(i_k, j) = d(i_k) + 1 < d(j)$ for some $(i, j) \in A$, then $d(j)$ was ∞ , and we set $d(j) \leftarrow d(i_k) + 1$. Since $d(v) \in \{d(i_k), d(i_k) + 1, \infty\}$ for all $v \in V$ not marked yet, then we can process j after any $v \in V$ not marked yet such that $d(v) < \infty$ is processed. Therefore, we can maintain a queue of all $v \in V$ not marked yet such that $d(v) < \infty$, and we push j to the tail of the queue if $d(j)$ is updated. The adapted algorithm is given in Algorithm 1. It is clear that Algorithm 1 runs in $O(m)$ time.

```
1  $q \leftarrow \text{new queue}()$ 
2  $d(i) \leftarrow \infty$  for all  $i \in V$ 
3  $p(i) \leftarrow \text{null}$  for all  $i \in V$ 
4  $d(s) \leftarrow 0$ 
5  $q.add(s)$ 
6 while not  $q.empty?$  do
7    $i \leftarrow q.remove()$ 
8   for  $j \in V$  such that  $(i, j) \in A$  do
9     if  $d(j) > d(i) + 1$  then
10        $d(j) \leftarrow d(i) + 1$ 
11        $p(j) \leftarrow i$ 
12        $q.add(j)$ 
```

Algorithm 1: Adapted Dijkstra's algorithm where $c(i, j) = 1$ for all $(i, j) \in A$.

Exercise 1.2. (\implies) Suppose for the sake of contradiction that there exists a negative-cost cycle C reachable from s . Let $v \in V(C)$. Let \mathcal{P} be the set of s - v paths. Let $P_0 \in \mathcal{P}$. Let P' be a v - v path along C . Then P_0 appended by any copy of P' is an s - v path. Since $c(P') := \sum_{e \in E(P')} c(e) < 0$, then $\{c(P) := \sum_{e \in E(P)} c(e) : P \in \mathcal{P}\}$ is not bounded below. Hence, there are no simple shortest s - v paths.

(\impliedby) Suppose that there are no negative-cost cycles reachable from s . Let $i \in V$. Let \mathcal{P} be the set of simple s - i paths. Since a simple s - i path consists of at most n distinct vertices, then $|\mathcal{P}| \leq n! < \infty$. Let $P^* = \arg \min_{P \in \mathcal{P}} c(P) := \arg \min_{P \in \mathcal{P}} \sum_{e \in E(P)} c(e)$. Let P be a non-simple s - i path. Then P contains a cycle C . Since there are no negative-cost cycles reachable from s , then $c(C) := \sum_{e \in E(C)} c(e) \geq 0$. This implies that removing all occurrences of C from P yields a simple s - i path P' with $c(P) \geq c(P') \geq \min\{c(P) : P \in \mathcal{P}\} = c(P^*)$. Hence, $c(P^*) \leq c(P)$ for any s - i path P , regardless of whether P is simple or not. This shows that P^* is the shortest s - i path, and P^* is simple.

Exercise 1.3. (a) Let $G = (V, A)$ be a DAG. Suppose for the sake of contradiction that any $v \in V$ has at least an arc directed into it. Let $v_0 \in V$. Starting from v_0 , we form a path backwards by following an edge directed into the vertices. By the pigeonhole principle, this forms a path with repeated vertices, say $\{(v_0, v_1), (v_1, v_2), \dots, (v_{k-1}, v_k)\}$, where $v_k = v_j$ for some $j \in \{0, \dots, k-2\}$. Then $\{(v_j, v_{j+1}), (v_{j+1}, v_{j+2}), \dots, (v_{k-1}, v_k)\}$ is a directed cycle, a contradiction.

(b) Let $G = (V, A)$. Let $c : A \rightarrow \mathbb{R}$ be the edge costs. We give Algorithm 2.

```

1  $q \leftarrow \text{new queue}()$ 
2  $d(i) \leftarrow \infty$  for all  $i \in V$ 
3  $p(i) \leftarrow \text{null}$  for all  $i \in V$ 
4  $d(s) \leftarrow 0$ 
5  $q.add(s)$ 
6 while  $not\ q.empty?$  do
7    $i \leftarrow q.remove()$ 
8   for  $j \in V$  such that  $(i, j) \in A$  do
9     if  $d(j) > d(i) + c(i, j)$  then
10        $d(j) \leftarrow d(i) + c(i, j)$ 
11        $p(j) \leftarrow i$ 
12     if  $not\ q.contains(j)$  then
13        $q.add(j)$ 

```

Algorithm 2: DAGShortest(G, c, s) for finding the shortest s - i path for each $i \in V$ in a DAG G .

Since G is a DAG, then G does not contain negative-cost cycles. By Exercise 1.2, there are simple shortest paths from s to each $i \in V$. We will prove that Algorithm 2 determines the length $d(i)$ of the shortest s - i path for all $i \in V$ by induction on the “passes.” Pass 0 ends after s is added to the queue, and pass k ends after any $i \in V$ such that the shortest s - i path uses at most k edges. The base case is trivial since the only s - s path is of length 0.

Suppose that after pass k for some k , $d(i)$ is the length of the shortest s - i path for any $i \in V$ such that the shortest s - i path uses at most k edges. Let $i \in V$ such that the shortest s - i path P uses $k+1$ edges. Let (j, i) be the last edge on P . Let P' be the subpath of P up to j . Then P' is the shortest s - j path. By the induction hypothesis, $c(P') = d(j)$. Therefore, $d(i)$ is set to $d(j) + c(j, i) = c(P') + c(j, i) = c(P)$ when Algorithm 2 processes j . This proves the claim.

- (c) Let $G = (V, A)$. Let $c : A \rightarrow \mathbb{R}$ be the edge costs. We give Algorithm 3. Let $i \in V$. Note that $\max\{c(P) : P \text{ is an } s\text{-}i \text{ path}\} = -\min\{-c(P) : P \text{ is an } s\text{-}i \text{ path}\}$.

1 $(d', p) \leftarrow \text{DAGShortest}(G, -c, s)$
 2 $d(i) \leftarrow -d'(i)$ for all $i \in V$

Algorithm 3: DAGLongest(G, c, s) for finding the longest s - i path for each $i \in V$ in a DAG G .

Exercise 1.4. [YP] ► We re-define $d_k(j)$ to be the length of the shortest s - j path of length k , as in R. M. Karp's original paper. ◀ Let $c' = c - \mu$. Let Γ_0 be a cycle of G . Then we have that

$$\begin{aligned} c'(\Gamma_0) &= \sum_{e \in E(\Gamma_0)} c'(e) = \sum_{e \in E(\Gamma_0)} (c(e) - \mu) = \sum_{e \in E(\Gamma_0)} c(e) - |\Gamma_0| \mu = c(\Gamma_0) - |\Gamma_0| \min_{\text{cycle } \Gamma \text{ of } G} \frac{c(\Gamma)}{|\Gamma|} \\ &\geq c(\Gamma_0) - |\Gamma_0| \cdot \frac{c(\Gamma_0)}{|\Gamma_0|} = c(\Gamma_0) - c(\Gamma_0) = 0. \end{aligned}$$

This shows that G with edge costs c' does not have negative-cost cycles. Hence, the Bellman-Ford algorithm correctly computes the shortest s - j paths for all $j \in V$. Let $d'_k(j)$ be the length of the shortest s - j path of length k with edge costs c' . By Exercise 1.2, there exists a simple shortest path P_j from s to any $j \in V$, which is of length $< n$. Hence, $c'(P_j) = \min_{0 \leq k \leq n-1} d'_k(j)$ and $c'(P_j) \leq d'_n(j)$ for all $j \in V$. This implies that $d'_n(j) \geq \min_{0 \leq k \leq n-1} d'_k(j)$ for all $j \in V$. On the other hand, let $\Gamma^* = \arg \min_{\text{cycle } \Gamma \text{ of } G} \frac{c(\Gamma)}{|\Gamma|}$. Then we have that

$$c'(\Gamma^*) = c(\Gamma^*) - |\Gamma^*| \min_{\text{cycle } \Gamma \text{ of } G} \frac{c(\Gamma)}{|\Gamma|} = c(\Gamma^*) - |\Gamma^*| \cdot \frac{c(\Gamma^*)}{|\Gamma^*|} = c(\Gamma^*) - c(\Gamma^*) = 0.$$

Let $v \in V(\Gamma^*)$. Let P be a simple shortest s - v path. Then $|P| < n$. Let $\ell \in \mathbb{N}$ be such that $|P| + \ell|\Gamma^*| \geq n$. Then P appended by ℓ copies of Γ^* , denoted by P' , is also a shortest s - v path. Hence, the subpath of P' formed by the first n edges of P' , which is an s - v^* path for some $v^* \in V$, is a shortest s - v^* path. Therefore, $d'_n(v^*) = c'(P_{v^*}) = \min_{0 \leq k \leq n-1} d'_k(v^*)$. This proves that

$$\min_{j \in V} \max_{0 \leq k \leq n-1} \frac{d'_n(j) - d'_k(j)}{n - k} = 0.$$

Let $j \in V$. Let $0 \leq k \leq n$. Let P be a shortest s - j path of length k . Then $d_k(j) = c(P)$. It is clear that P is also a shortest s - j path of length k with edge costs c' . Hence, $d'_k(j) = c'(P)$. Since $d_k(j)$ is a path of length k , then we have that

$$d_k(j) = \sum_{e \in E(P)} c(e) = \sum_{e \in E(P)} (c(e) - \mu + \mu) = \sum_{e \in E(P)} (c(e) - \mu) + |P|\mu = \sum_{e \in E(P)} c'(e) + k\mu$$

$$= c'(P) + k\mu = d'_k(j) + k\mu.$$

Hence, we have that

$$\begin{aligned} \min_{j \in V} \max_{0 \leq k \leq n-1} \frac{d_n(j) - d_k(j)}{n - k} &= \min_{j \in V} \max_{0 \leq k \leq n-1} \frac{(d'_n(j) + n\mu) - (d'_k(j) + k\mu)}{n - k} \\ &= \min_{j \in V} \max_{0 \leq k \leq n-1} \frac{d'_n(j) - d'_k(j) + (n - k)\mu}{n - k} \\ &= \min_{j \in V} \max_{0 \leq k \leq n-1} \left(\frac{d'_n(j) - d'_k(j)}{n - k} + \mu \right) \\ &= \min_{j \in V} \max_{0 \leq k \leq n-1} \frac{d'_n(j) - d'_k(j)}{n - k} + \mu \\ &= 0 + \mu = \mu. \end{aligned}$$

Next, we show that $d_k(j)$ can be computed by the following recurrence:

$$d_k(j) = \begin{cases} \min_{(i,j) \in E} (d_{k-1}(i) + c(i, j)), & k > 0, \\ 0, & k = 0, j = s, \\ \infty, & k = 0, j \neq s. \end{cases}$$

It is clear that $d_0(s) = 0$ and $d_0(j) = \infty$ for all $j \in V \setminus \{s\}$. Let $1 \leq k \leq n$. Let $j \in V$. Let P be a shortest s - j path of length k . Let (i^*, j) be the last edge of P . Then the subpath P' formed by all edges of P except (i^*, j) is a shortest s - i^* path of length $k - 1$. Hence, $c(P') = d_{k-1}(i^*)$. This implies that

$$d_k(j) = c(P) = c(P') + c(i^*, j) = d_{k-1}(i^*) + c(i^*, j) \geq \min_{(i,j) \in E} (d_{k-1}(i) + c(i, j)).$$

For all $(i, j) \in E$, if P_i is a shortest s - i path of length $k - 1$, then P_i appended by (i, j) is an s - j path, so $d_{k-1}(i) + c(i, j) = c(P_i) + c(i, j) \geq d_k(j)$. This implies that $\min_{(i,j) \in E} (d_{k-1}(i) + c(i, j)) \geq d_k(j)$. This proves the recurrence. We give Algorithm 4 to compute μ . It is clear that the running time of Algorithm 4 is $O(nm)$.

Finally, we give Algorithm 5 to compute a cycle Γ such that $\mu = \frac{c(\Gamma)}{|\Gamma|}$. It is clear that the running time of Algorithm 5 is $O(nm)$. It remains to show that Γ^* returned by Algorithm 5 satisfies $\frac{c(\Gamma^*)}{|\Gamma^*|} = \mu$. We note that $p_k(j)$ stores a shortest s - j path of length k by following $p_k(j)$ backwards. Hence, P is a shortest s - j^* path of length n . This implies that P is not simple and hence contains a cycle Γ . Since $\frac{d_n(j^*) - d_{k^*}(j^*)}{n - k^*} = \mu$, then we have that

$$\begin{aligned} \frac{d'_n(j^*) - d'_{k^*}(j^*)}{n - k^*} &= \frac{(d_n(j^*) - n\mu) - (d_{k^*}(j^*) - k^*\mu)}{n - k^*} = \frac{d_n(j^*) - d_{k^*}(j^*) - (n - k^*)\mu}{n - k^*} \\ &= \frac{d_n(j^*) - d_{k^*}(j^*)}{n - k^*} - \mu = \mu - \mu = 0. \end{aligned}$$

This implies that $d'_n(j^*) = d'_{k^*}(j^*) = \min_{0 \leq k \leq n-1} d'_k(j^*)$ is the length of the shortest s - j^* path. Hence, cycle Γ^* contained in P must have cost 0 with edge costs c' , since we could have eliminated Γ^* to get a lower cost. We have that

$$\frac{c(\Gamma^*)}{|\Gamma^*|} = \frac{\sum_{e \in E(\Gamma^*)} c(e)}{|\Gamma^*|} = \frac{\sum_{e \in E(\Gamma^*)} (c(e) - \mu + \mu)}{|\Gamma^*|} = \frac{\sum_{e \in E(\Gamma^*)} (c(e) - \mu) + |\Gamma^*| \mu}{|\Gamma^*|}$$

```

1  $d_0(s) \leftarrow 0$ 
2  $d_k(j) \leftarrow \infty$  for all  $1 \leq k \leq n, j \in V$ 
3 for  $k \leftarrow 1, \dots, n$  do
4   for  $(i, j) \in E$  do
5      $d_k(j) \leftarrow \min(d_k(j), d_{k-1}(i) + c(i, j))$ 
6  $\mu \leftarrow \infty$ 
7 for  $j \in V$  do
8    $\nu \leftarrow -\infty$ 
9   for  $k \leftarrow 0, \dots, n-1$  do
10     $\nu \leftarrow \max(\nu, \frac{d_n(j) - d_k(j)}{n-j})$ 
11   $\mu \leftarrow \min(\mu, \nu)$ 

```

Algorithm 4: An algorithm for computing the cost of the minimum mean-cost cycle.

$$= \frac{\sum_{e \in E(\Gamma^*)} c'(e)}{|\Gamma^*|} + \mu = \frac{c'(\Gamma^*)}{|\Gamma^*|} + \mu = \frac{0}{|\Gamma^*|} + \mu = 0 + \mu = \mu.$$

This completes the proof.

```

1  $d_0(s) \leftarrow 0$ 
2  $d_k(j) \leftarrow \infty$  for all  $1 \leq k \leq n, j \in V$ 
3  $p_k(j) \leftarrow \text{null}$  for all  $0 \leq k \leq n, j \in V$ 
4 for  $k \leftarrow 1, \dots, n$  do
5   for  $(i, j) \in E$  do
6     if  $d_{k-1}(i) + c(i, j) < d_k(j)$  then
7        $d_k(j) \leftarrow d_{k-1}(i) + c(i, j)$ 
8        $p_k(j) \leftarrow i$ 
9  $\mu \leftarrow \infty$ 
10 for  $j \in V$  do
11    $\nu \leftarrow -\infty$ 
12   for  $k \leftarrow 0, \dots, n-1$  do
13      $\nu \leftarrow \max(\nu, \frac{d_n(j) - d_k(j)}{n-j})$ 
14   if  $\nu < \mu$  then
15      $\mu \leftarrow \nu$ 
16      $j^* \leftarrow j$ 
17  $P = \{(v_1, v_2), \dots, (v_{n-1}, v_n)\} \leftarrow$  path formed by following  $p_n$  from  $j^*$  backwards
18 for  $p \leftarrow 1, \dots, n-1$  do
19   for  $q \leftarrow p+1, \dots, n$  do
20     if  $v_p = v_q$  then
21       return  $\Gamma^* = \{(v_p, v_{p+1}), \dots, (v_{q-1}, v_q)\}$ 

```

Algorithm 5: An algorithm for computing the minimum mean-cost cycle.