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.

```
q ← new queue()
d(i) ← ∞ for all i ∈ V
p(i) ← null for all i ∈ V
d(s) ← 0
q.add(s)
while not q.empty? do
    i ← q.remove()
    for j ∈ V such that (i, j) ∈ A do
        if d(j) > d(i) + 1 then
            d(j) ← d(i) + 1
            p(j) ← i
            q.add(j)
```

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

Exercise 1.2. ($\Longrightarrow$) 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.

($\Longleftarrow$) 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 $|P| \leq n! < \infty$. 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 $\{(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)\}$, where $v_k = v_j$ for some $j \in \{0, \ldots, k-2\}$. Then $\{(v_j, v_{j+1}), (v_{j+1}, v_{j+2}), \ldots, (v_{k-1}, v_k)\}$ is a directed cycle, a contradiction.

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

```
q ← new queue()
d(i) ← ∞ for all i ∈ V
p(i) ← null for all i ∈ V
d(s) ← 0
q.add(s)
while not q.empty? do
    i ← q.remove()
    for j ∈ V such that (i, j) ∈ A do
        if d(j) > d(i) + c(i, j) then
            d(j) ← d(i) + c(i, j)
            p(j) ← i
        if not q.contains(j) then
            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 $c : A \to \mathbb{R}$ 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) ← DAGShortest(G, −c, s)
2 d(i) ← −d'(i) for all i ∈ 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

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

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 $\Gamma^* = \arg\min_{\text{cycle } \Gamma \text{ of } G} \frac{c(\Gamma)}{|G|}$. 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 the path P' formed by appending ℓ copies of Γ^* to the end of P is also a shortest s-v path. Hence, the subpath P'' 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, $d'_n(v^*) = c'(P'') = \min_{0 \leq k \leq n-1} d'_k(v^*)$. 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

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

$$= \sum_{e \in E(P)} c'(e) + k\mu = c'(P) + k\mu = d'_k(j) + k\mu.$$

Hence, we have that

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

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

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

It is clear that $d_0(s) = 0$ and $d_0(j) = \infty$ for all $j \in V \setminus \{s\}$. Let $1 \le k \le n$. 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

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

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 $d_{k-1}(i) + c(i,j) = c(P_i) + c(i,j) \ge d_k(j)$. This implies that $\min_{(i,j) \in E}(d_{k-1}(i) + c(i,j)) \ge d_k(j)$. This proves (1). We give Algorithm 4 to compute μ and a cycle Γ such that $\mu = \frac{c(\Gamma)}{|\Gamma|}$. It is clear that the running time of Algorithm 4 is $O(nm)$. It remains to show that Γ^* returned by Algorithm 4 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 at least one cycle. 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 $d'_n(j^*) = d'_{k^*}(j^*) = \min_{0 \le k \le n-1} d'_k(j^*)$ is the length of the shortest s-j^* path. Hence, cycle Γ^* contained in P must have cost 0 with edge costs c'. Otherwise, we could have eliminated Γ^* to get a lower cost. We have that

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

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

This completes the proof.

```
d_k(j) ← ∞ for all 0 ≤ k ≤ n, j ∈ V
d_0(s) ← 0
p_k(j) ← null for all 0 ≤ k ≤ n, j ∈ V
for k ← 1, ..., n do
    for (i, j) ∈ E do
        if d_{k-1}(i) + c(i, j) < d_k(j) then
            d_k(j) ← d_{k-1}(i) + c(i, j)
            p_k(j) ← i
μ ← ∞
for j ∈ V do
    ν ← −∞
    for k ← 0, ..., n − 1 do
        ν ← max(ν, (d_n(j) − d_k(j))/(n − j))
    if ν < μ then
        μ ← ν
        j* ← j
P = {(v_1, v_2), ..., (v_{n-1}, v_n)} ← path formed by following p_n from j* backwards
for p ← 1, ..., n − 1 do
    for q ← p + 1, ..., n do
        if v_p = v_q then
            return Γ* = {(v_p, v_{p+1}), ..., (v_{q-1}, v_q)}
```

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

Exercise 1.5. (a) We give Algorithm 5. Let k be the number of iterations in the binary search of Algorithm 5. Then k is the minimum positive integer such that $\frac{(nC+1)-(-nC)}{2^k} < \frac{1}{(nT)^2}$, i.e. $2^k > (2nC+1)(nT)^2$. Hence, we have that

$$k = \left\lceil \log_2 \left((2nC+1)(nT)^2 \right) \right\rceil + 1 = O\left(\log\left(nC(nT)^2 \right) \right) = O\left(\log\left(n^3 CT^2 \right) \right)$$
$$= O(3\log n + \log C + 2\log T) = O(\log n + \log C + \log T) = O(\log(nCT)).$$

In each iteration, we invoke the negative-cost cycle detection algorithm, whose running time is $O(nm)$. Hence, the total running time of Algorithm 5 is $O(nm \log(nCT))$.

Let

$$\mu^* = \min_{\text{cycle } \Gamma \text{ of } G} \frac{c(\Gamma)}{t(\Gamma)}.$$

We will show that $\ell \le \mu^* < u$ by induction on the iterations of the binary search. Let Γ be a cycle of G. Then $|\Gamma| \le n$. Since $t(\Gamma_j) = \sum_{e \in E(\Gamma_j)} t(e) \in \mathbb{N}$, then we have that

$$\left| \frac{c(\Gamma)}{t(\Gamma)} \right| = \frac{|c(\Gamma)|}{t(\Gamma)} \le \frac{\left| \sum_{e \in E(\Gamma)} c(e) \right|}{1} \le \sum_{e \in E(\Gamma)} |c(e)| \le \sum_{e \in E(\Gamma)} C = |\Gamma| C \le nC.$$

$\ell \leftarrow -nC$
$u \leftarrow nC+1$
while $u - \ell \geq \frac{1}{(nT)^2}$ **do**
  $\mu \leftarrow \frac{\ell+u}{2}$
  Check whether G has negative-cost cycles with edge costs $c - \mu t$
  if *there exists a negative-cost cycle of G with edge costs $c - \mu t$* **then**
    $u \leftarrow \mu$
  else
    $l \leftarrow \mu$
Find a negative-cost cycle Γ^* of G with edge costs $c - ut$
return Γ^*

Algorithm 5: An algorithm for finding a cycle that minimizes $\min_{\text{cycle } \Gamma \text{ of } G} \frac{c(\Gamma)}{t(\Gamma)}$.

This shows that $-nC \leq \mu^* \leq nC < nC+1$, proving the base case. Let ℓ_0, u_0 be the values of ℓ, u in some iteration. Let ℓ', u' be the values of ℓ, u in the next iteration. If G has a negative-cost cycle Γ with edge costs $c - \mu t$, then $c(\Gamma) - \mu t(\Gamma) < 0$ and hence $\frac{c(\Gamma)}{t(\Gamma)} < \mu$; this implies that $\mu^* < \mu$. Otherwise, $c(\Gamma) - \mu t(\Gamma) \geq 0$ and hence $\frac{c(\Gamma)}{t(\Gamma)} \geq \mu$ for any cycle Γ of G; this implies that $\mu^* \geq \mu$. This completes the induction step.

Let ℓ^*, u^* be the final values of ℓ, u. We will show that the cycle Γ^* returned by Algorithm 5 satisfies $\frac{c(\Gamma^*)}{t(\Gamma^*)} = \mu^*$. Note that

$$\frac{c\left(\Gamma^*\right)}{t\left(\Gamma^*\right)} \geq \min_{\text{cycle } \Gamma \text{ of } G} \frac{c(\Gamma)}{t(\Gamma)} = \mu^*.$$

Suppose for the sake of contradiction that $\frac{c(\Gamma^*)}{t(\Gamma^*)} > \mu^*$. Then $c(\Gamma^*) > \mu^* t(\Gamma^*)$. By Algorithm 5, $c(\Gamma^*) - u^* t(\Gamma^*) < 0$, and hence $c(\Gamma^*) < u^* t(\Gamma)$. Combining these two inequalities gives that $\mu^* t(\Gamma^*) < c(\Gamma^*) < u^* t(\Gamma^*)$. Hence, we have that $\mu^* < \frac{c(\Gamma^*)}{t(\Gamma^*)} < u^*$. This implies that

$$\frac{c(\Gamma^*)}{t(\Gamma^*)} - \mu^* < u^* - \mu^* \leq u^* - \ell^* < \frac{1}{(nT)^2}. \tag{2}$$

Let

$$\Gamma' = \arg\min_{\text{cycle } \Gamma \text{ of } G} \frac{c(\Gamma)}{t(\Gamma)}.$$

For any cycle Γ of G, we have that

$$t(\Gamma) = \sum_{e \in E(\Gamma)} t(e) \leq \sum_{e \in E(\Gamma)} T = |\Gamma| T \leq nT.$$

Since $\frac{c(\Gamma^*)}{t(\Gamma^*)} > \mu^*$ and since $c(i,j), t(i,j) \in \mathbb{Z}$, then we have that

$$\frac{c\left(\Gamma^*\right)}{t\left(\Gamma^*\right)} - \mu^* = \frac{c\left(\Gamma^*\right)}{t\left(\Gamma^*\right)} - \frac{c\left(\Gamma'\right)}{t\left(\Gamma'\right)} = \frac{c\left(\Gamma^*\right) t\left(\Gamma'\right) - c\left(\Gamma'\right) t\left(\Gamma^*\right)}{t\left(\Gamma^*\right) t\left(\Gamma'\right)} \geq \frac{1}{t\left(\Gamma^*\right) t\left(\Gamma'\right)} \geq \frac{1}{(nT)^2}.$$

This contradicts (2). Hence, $\frac{c(\Gamma^*)}{t(\Gamma^*)} = \mu^*$. The proof is complete.