Network Flow Algorithms: Exercise Solutions

Yuchong Pan

December 19, 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 $d(j)$ sas ∞, 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.

$q \leftarrow$ *new queue()*
$d(i) \leftarrow \infty$ for all $i \in V$
$p(i) \leftarrow$ **null** for all $i \in V$
$d(s) \leftarrow 0$
q.add(s)
while *not q.empty?* **do**
    $i \leftarrow$ *q.remove()*
    for $j \in V$ *such that* $(i, j) \in A$ **do**
        if $d(j) > d(i) + 1$ **then**
            $d(j) \leftarrow d(i) + 1$
            $p(j) \leftarrow i$
            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.