CSCI-GA.1170-001/002 Fundamental Algorithms December 13, 2019

Solutions to Problem 1 of Homework 12 (16 points)

Name: jingshuai jiang (jj2903) *Due: 5pm on Thursday, December 12*

Collaborators: NetID1, NetID2

You are given a map $G = (V, E)$ with cities V connected by roads E. Each road (edge) is labeled with a weight which is a *real number*. You are located in city $s \in V$. You are also given an array A of boolean values that tells you if there is a toll on that road. More precisely, for road $e \in E$ we have $A[e] = 1$ if and only if there is a toll on e. Being the low budget traveler that you are, your budget allows for at most one such toll to be payed for any given trip (path).

Let $d(s, t)$ denote the minimum possible sum of weights of a path from s to t for any $s, t \in V$. If there is no path from s to t, then $d(s, t) = \infty$, and if there is a path from s to t that contains a negative cycle, then $d(s, t) = -\infty$ (since one can cycle along the path an arbitrary number of times).

Similarly, let $c(s, t)$ denote the minimum possible sum of weights of a path from s to t that passes through at most 1 toll.

(a) (6 points) Construct a graph $G' = (V', E')$ and mappings $f : V \mapsto V'$, $g : V \mapsto V'$ such that $|V'| = 2|V|$, $|E'| \leq 2|E| + |V|$ and for any $s, t \in V$, $c(s, t) = d(f(s), g(t))$. Namely, you reduce the "constrained" problem on G to "unconstrained" problem in G'. Remember to consider the case when $c(s, t)$ is $-\infty$, and prove the correctness of your solution. (**Hint**: Use a strategy similar to the one used for a previous homework question.)

Solution: New graph G': The vertices in G' is consists of two parts. A and B. Their vertices are the same as vertices in G. If edge in G is connected without a toll. Then draw all these edges in A and B. If there is toll edge in G and its two nodes are c and d. Then we draw an edge from A.c to B.d. And do this to all the toll edges. Function f will return a vertex in A, g return a vertex in B. **Correctness:**

Since both A and B have the same vertices in G. Then obviously the number of vertices in G' is $|V| + |V| = 2|V|$

The edges in G' is $2|E| - N_a \leq 2|E| + |V|$.

f(s) returns a node in A and g(t) returns a node in B. $c(s, t) = d(f(s), g(t))$ returns the path from a vertex in A to a vertices in B. □

(b) (3 points) Give an algorithm that takes as input s and finds a shortest path with at most one toll road from s to all cities in V. Analyze the running time of your algorithm. (**Hint**: Use an algorithm discussed in class.)

Solution: **Algorithm:** Construct a graph G' as described in (a).

Use Bellman-Ford algorithm to run on (G',s) to assign the shortest path for s to all other vertices.

If we detect there is a negative cycle in the bellman-ford algorithm. Then we

Runtime: The runtime of bellman-ford algorithm is $O(VE)$. And the following DFS takes time □

(c) (3 points) Now assume your friend who works at a major airlines company has given you a free plane ticket to any city in V, meaning you can start your trip at any node. As before, once you start your road trip, you are still refusing to pay more then a single toll on any such trip. To help plan the trip, your job is to give an algorithm to find a shortest path with at most one toll road between *all pairs* of cities in V. Analyze the running time of your algorithm. (**Hint**: Use another algorithm discussed in class. How many edges are there?)

Solution:

□

(d) (4 points) Here you will solve the problem in part (c) directly on graph G, without constructing the helper graph G'. Let $W = \{w(i,j)\}$ be the original edge weight matrix and $W' = \{w'(i,j)\}$ be the same edge matrix except we replace $w'(i,j) = \infty$ if $A(i,j) = 1$ (i.e., never use toll roads in W'). For simplicity, assume W' is pre-computed for you. For $0 \le k \le n$, let

- D^k be the matrix of all shortest distances w.r.t. W' which only use nodes $\le k$ as intermediate nodes.
- C^k be the matrix or all shortest distances w.r.t. W which only use nodes $\le k$ as intermediate nodes, but *also use at most one toll.*

Fill in the blanks below to directly modify the Floyd-Warshall algorithm to compute the correct answer for problem (c) in time $O(n^3)$. Argue the correctness of your algorithm.

```
FLOYD-WARSHALL($W, W'$)
    $n = W.rows$
    $D^0 = ......$
    $C^0 = ......$
    for $k = 1$ to $n$ do
        $C^k = (c^k_{i,j})$ be a new $n \times n$ matrix
        $D^k = (d^k_{i,j})$ be a new $n \times n$ matrix
        for $i = 1$ to $n$ do
            for $j = 1$ to $n$ do
                $d^k_{i,j} = \min(..............................................................)$
                $c^k_{i,j} = \min(..............................................................)$
    return .............
```

Solution: INSERT YOUR SOLUTION HERE □

CSCI-GA.1170-001/002 Fundamental Algorithms December 13, 2019

Solutions to Problem 2 of Homework 12 (19+4 points)

Name: jingshuai jiang (jj2903) *Due: 5pm on Thursday, December 12*

Collaborators: NetID1, NetID2

You are the grader for a class evaluating a programming assignment. The goal of this assignment is to implement an algorithm $SSSP$ that takes as input a graph $G = (V, E)$ where each edge $e \in E$ has an associated weight $w_e \in \mathbb{R}$, the set of real numbers. Further, $s \in V$ is designated as the source. The algorithm returns the $v.d, v.\pi$ values for each $v \in V$.

You decide to test only one of the submissions, because it looks weird. So, you generate the graph $G = (V, E)$. You run the algorithm on this G and get the outputs $v.d, v.\pi$ for each v. Now, you need to efficiently test if the answers are indeed right. To help you, we have given you the algorithm.

1. Verify that $s.d = 0$ and $s.\pi = nil$. If fail, output "wrong".
2. Verify that
   (a) $\forall v \neq s$ such that $v.\pi \neq$ **nil** , there exists $(v.\pi, v) \in E$. If fail, output "wrong".
   (b) Else define a parent graph $G' = (V, E')$ such that E' consists of edges of the form $(v.\pi, v)$ for $v \neq s$ and $v.\pi \neq$ **nil** .
   (c) Run DFS-Visit(s) on G'. Output "wrong" if you either reach a node with $v.d = \infty$ or fail to reach a node with $v.d < \infty$.
3. Verify that for all $v \neq s$ such that $v.d < \infty$, $v.d = v.\pi.d + w(v.\pi, v)$. If fail, output "wrong".
4. Run one pass of the Bellman-Ford algorithm (i.e., try to relax each edge of G). If any relaxation is successful, output "wrong".
5. Otherwise, output "correct".

(a) (2 points) What is the run time of the above algorithm? Show that this algorithm accepts the right answer always.

Solution: for step1 , the runtime is $O(1)$.

for step2 , the step (a) will take time$O(|V|)$,for step(bc),for worst cases. The number of edges in the new graph G' is $|E'| = |E|$. Then The DFS will take time $O(|V| + |E|)$

for step3 , the runtime for one pass of bellman-ford is $O(|V|)$.

for step4 , the runtime is $O(|E|)$.

for step5 , the runtime is $O(1)$.

The total runtime should be $O(|V| + |E|)$.

correctness: for step1, since it is the starting node. its distance should be zero. And had no parent. for step2, if $v! = s$ then the edge of connecting it to its parent node should be in the edge set of E. Since we are running sssp. Then after the valid solution, we should reach any vertices whose distance is less than infinity. Then step2 is fine. for step3, for any reachable node. the walk from their parents to themselves is the shortest path. Then $v.d = v.parent.d + w(v.parent, v)$. for step4, since it is already the shortest path. Then after one pass of bellman ford no distance should be updated. Then in this way. The algorithm accepts the right answer. □

For the next several parts, we will prove the correctness in the opposite direction. More formally, we will prove that if the algorithm returns correct, then the answer is indeed right. In part (a), you proved that if the answer is right, the algorithm returns "correct".

To prove the converse, let us assume that the answer is wrong. We will then show that one of the checks 1-4 will catch the error and consequently return "wrong". For simplicity, we will not verify that the parent values are correct when all distances are correct, and instead focus on the case where at least one distance is wrong (with parents being arbitrary). Concretely, assume that there is some node $v \in V$ s.t. $\delta(s, v) \neq v.d$. Clearly, if $v = s$, and $v.d \neq \delta(s, s) = 0$, then the first check would catch the error.

So, we have that $v \neq s$ and we split the proof in four parts. For each case, state which check(s) 2-4 are used in the proofs of the corresponding part.

(b) (2 points) $\delta(s, v) = \infty$ but $v.d < \infty$. Which check(s) will catch this? Justify your answer.

Solution: step 2 checks this problem. if $\delta(s, v) = \infty$, then this node is not reachable from the source node s. Running DFS on this graph from s will lead to a failure to reach this v with $v.d < \infty$. □

(c) (3 points) $\delta(s, v) < \infty$ but $v.d = \infty$. Which check(s) will catch this? Justify your answer.

Solution: This is checked by step2. If $v.d = \infty$ and $\delta(s, v) < \infty$. By running DFS, i will reach a node v with $v.d = \infty$ which is wrong in 2(c). And it can be checked by step3. for all node v. $v.d < \infty$ but this node is not correct. □

(d) (3 points) $\delta(s, v) < v.d < \infty$. Which check(s) will catch this? Justify your answer.

Solution: This is checked by step4. if $\delta(s, v) < v.d < \infty$. Then it means this v.d is not the shortest path. And this edge will be relax in the bellman-ford algorithm. Then it is wrong. □

(e) **(Extra Credit:)**(4 points) $v.d < \delta(s, v) < \infty$. Which check(s) will catch this? Justify your answer.

Solution: INSERT YOUR SOLUTION HERE □

You now realize that the student's algorithm is indeed correct. To make matters easier for future grading, you decide to standardize this by creating a set of sample input, output test cases. However, you do not want to come up with new graphs. Instead, you decide to reuse the same graph as before but merely change the source from s to s'. Your task is now to come up with an $O(|E|\log|V|)$ algorithm that produces the values for the new source s'. More formally, you have $v.d$ values for $v \in V$ (which is the measure of the shortest path from s to v). You will now need to compute $v.d'$ (which is the measure of the shortest path from s' to v). You do not need to compute $v.\pi'$, i.e, the parent node. Ideally, one can run Dijkstra's right off the bat but weights can be negative.

(f) (3 points) You will construct a new graph G' with different edge weights. You will use the values of $v.d$ which the student's algorithm has generated and your algorithm has verified. Let w'_e be the new edge weight of an edge $e = (u, v)$ and w_e be the weight in graph G. What is the new w'_e? Fill in the blank below. Also, argue that $w'_e \geq 0$. (**Hint**: Use ideas of Johnson's algorithm to define the new edge weight, which must be non-negative.)

$$w'_e = \underline{\hspace{6cm}}$$

Solution: INSERT YOUR SOLUTION HERE ☐

(g) (3 points) Formally, let P be the min-cost path from s' to an arbitrary node x. Let $w(P)$ denote the cost of P under w and $w'(P)$ denote the cost under w'. Fill in the blanks below for $w'(P)$. Justify the correctness of your equation. Use this equation to argue that the min-cost path in G is not affected by the new weights.

$$w'(P) = w(P) + \underline{\hspace{6cm}}$$

Solution: INSERT YOUR SOLUTION HERE ☐

(h) (3 points) Finally, put the last two parts together to construct an algorithm that takes as input G and the correct values from the student's algorithm to produce the shortest distance to s'. Argue why your running time is $O(|E|\log|V|)$. You can write this in words. (**Hint**: Use Dijkstra, as alluded to earlier.)

Solution: INSERT YOUR SOLUTION HERE ☐