CS5800 ALGORITHMS S'22 — ABHI SHELAT
DUE FRI APR 15, 2022 AT 11.59P VIA GRADESCOPE

H8.SOL

PROBLEM 1 *Number of shortest paths*

Given a graph $G = (V, E)$ with unit edge weights, and a starting node s, let $\delta(s, v)$ be the length of the shortest path between s and v (i.e., the smallest number of edges between s and v).

Define a new variable n_v which records the number of distinct shortest paths from s to v that have length $\delta(s, v)$ for every node in V. Design an algorithm that computes n_v for all nodes in V.

Solution: The key to this problem is the observation that all shortest paths from s to v consist of shortest paths from s to some node u and then an edge from u to v. Therefore, if we count the number of distinct shortest paths to u, we can do the same v.

We set $n_s = 0$. Compute BFS on G. Now for every node with $d_v = 1$, we set $n_v = 1$, since the shortest path from s in this case corresponds to the single edge from s to v. We now consider every node with $d_v = j$ for $j = 2, 3, \ldots, V$. For each such node v, we set n_v to be the

$$n_v = \sum_{u \in Adj(v), d_u = j-1} n_u$$

I.e., n_v is the sum of n_u for each neighbor u of v for which $d_u = j - 1$.

The overall running time of this algorithm is $O(V + E)$, because BFS requires time $O(V + E)$ and the summation for each $v \in V$ amounts to inspecting every edge in the graph twice. Thus, the summation phase also requires time $O(V + E)$.

PROBLEM 2 *Sparse graphs and short paths*

Let $G = (V, E)$ be a directed graph with edge weights $w(e)$ and no negative cycles.

1. State the run time of the All-pairs shortest path algorithm discussed in class.

   Solution: $O(V^3)$.

2. Consider the following algorithm.

   AnotherShortest(G, w)

   Add a new node s' to G. Add edges of weight 0 from s' to every vertex $v \in V$. Call this new graph G'.
   Run BellmanFord(G', s') to produce shortest path lengths $\delta(s', v)$. If shortest paths are not well-defined, than halt.
   For each $e = (x, y) \in E$, set $w'(e) \leftarrow w(e) + \delta(s', x) - \delta(s', y)$
   For each $v \in V$, run Dijkstra(G, v, w') to compute $\delta(v, x)$ for all $x \in V$.
   Set $d_{v,w} \leftarrow \delta(v, w) - \delta(s', v) + \delta(s', w)$

This problem will analyze what this algorithm does and why it works. The first step is to argue that the new edge weights w' that are defined in step (3) are all non-negative.

Prove that for all $e \in E$, $w'(e) \geq 0$.

Solution: Recall that edge $e = (x, y)$ connects y to x. Therefore, by definitions of shortest paths, it follows that $\delta(s', y) \leq \delta(s', x) + w(e)$. I.e., the shortest path to y is at most as long as the shortest path to x and the weight of edge e. Subtracting, we have that

$$0 \leq \delta(s', x) + w(e) - \delta(s', y)$$

which through rearranging, shows that

$$0 \leq w'(e) = w(e) + \delta(s', x) - \delta(s', y)$$

3. This explains why we can use the fast DIJKSTRA algorithm with edge weight w' in step (4) to compute shortest paths from node $v \in V$ to all other nodes in the graph. However, we must argue that the shortest paths under w' and under w will be the same shortest path.

   Prove that for any pairs of nodes $u, v \in V$, if p is a shortest path from u to v with respect to edge weight function w', then p is also a shortest path from u to v with respect to edge weight function w.

   Solution: Consider the path p from u to v, which passes through nodes $x_1, x_2, \ldots, x_k$. The length of this shortest path is therefore

$$w'(p) = w'(u, x_1) + w'(x_1, x_2) + \cdots + w'(x_k, v)$$

   Expanding these out, we have

$$\begin{aligned} w'(p) = [&w(u, x_1) + \delta(s', u) - \delta(s', x_1)] + [w(x_1, x_2) + \delta(s', x_1) - \delta(s', x_2)] + \\ &\cdots + [w(x_k, v) + \delta(s', x_k) - \delta(s', v)] \end{aligned}$$

   This sum telescopes (see how the last subtracted term from one edge weight cancels the first added term in the next edge weight) to produce

$$\begin{aligned} w'(p) &= w(u, x_1) + w(u, x_2) + \cdots + w(x_k, v) + \delta(s', u) - \delta(s', v) \\ &= w(p) + \delta(s', u) - \delta(s', v) \end{aligned}$$

   This equation implies that p must also be the shortest path from u to v with edge weights w. Suppose that there was another path p' from u to v such that $w(p') < w(p)$; this would imply that $w'(p') = w(p') + \delta(s', u) - \delta(s', v) < w'(p) = w(p) + \delta(s', u) - \delta(s', v)$ which contradicts the assumption that p was a shortest path from u to v with respect to edges weights w'.

4. What is the running time of AnotherShortest in terms of V and E? When does this algorithm run faster than the All-pairs algorithm discussed in class?

   Solution: Line 2 requires $O(VE)$ time. Line 4 requires $O(V \cdot (E \log V))$ time. Line 5 requires $O(V^2)$ time. The remaining lines all require time $O(E)$. Therefore, the overall running time of the algorithm is $O(EV \log V)$ which is better than $O(V^3)$ when $E \log V$ is less than V^2, i.e., when the graph is not dense.

PROBLEM 3 *Edmonds-Karp shortest paths*

In class, we stated that in the Edmonds-Karp maxflow algorithm, the length of shortest paths in G are monotonically increasing. However, this is not obvious because as we add augmenting paths, new edges are introduced to the graph. In this problem, we will prove the following:

Lemma 1 *For any $j > i$ and for any $u \in V$, $\delta_j(s,u) \geq \delta_i(s,u)$.*

The proof will be by contradiction. Suppose not, for the sake of contradiction. Let i be the first time that the shortest path distance to some node decreases after pushing flow along the i^{th} augmentation. Moreover, let v be the *node with the smallest* distance to s at $i+1$ for which $\delta_i(s,v) > \delta_{i+1}(s,v)$. Let p_i, p_{i+1} be respective shortest paths from s to v at times i and $i+1$.

Each answer should be roughly 1 sentence. You may refer to steps (1)–(7) in your explanations.

1. Define node u to be the node that occurs before v on path p_{i+1}. The first claim is that $\delta_{i+1}(s,u) \geq \delta_i(s,u)$. Why does this follow? (one sentence)

   Solution: Because v was assumed to be the closest to s, i.e., node with the smallest distance from s for which the inversion happens, it follows that u, which is on the shortest path, but occurs before v on the path cannot have its distances shrink like this.

2. Next, explain why $\delta_{i+1}(s,v) = \delta_{i+1}(s,u) + 1$.

   Solution: Because p_{i+1} is a shortest path and distances along shortest paths must increase by one in this way.

3. Explain why edge $e_{i+1} = (u,v)$ did not exist in the graph at time i.

   Solution: The previous step implies that edge $e_{i+1} = (u,v)$ could not have been in the graph at time i because otherwise $\delta_i(s,v)$ would be equal to $\delta_{i+1}(s,v)$.

4. Thus, the edge e_{i+1} must have been added after the i flow, which implies that the augmenting path at i took the form $s \rightsquigarrow v \rightarrow u \rightsquigarrow t$, i.e., that pushed flow from v to u. Thus at time i, we have

   $$\delta_i(s,u) = \delta_i(s,v) + 1$$

   Explain why in one sentence.

   Solution: Because this augmenting flow was a shortest path from s to t, this means it was also a shortest path from s to u.

5. Explain why this implies

   $$\delta_{i+1}(s,u) \geq \delta_i(s,v) + 1$$

   Solution: From step 1, $\delta_{i+1}(s,u) > \delta_i(s,u)$.

6. Adding one to each side, we have

   $$\delta_{i+1}(s,u) + 1 \geq \delta_i(s,v) + 2$$

   Explain why this implies that

   $$\delta_{i+1}(s,v) \geq \delta_i(s,v) + 2$$

   Solution: From step 2.

7. Explain why the previous statement is a contradiction.

   Solution: We started by assuming that $\delta_i(s,v) > \delta_{i+1}(s,v)$, but have concluded the opposite.