Homework 11

Graded

Student

Nick Zhu

Total Points

100 / 100 pts

Question 1

Problem 1

20 / 20 pts

✔ **– 0 pts** Correct

Question 2

Problem 2

20 / 20 pts

✔ **– 0 pts** Correct for a， AB, CF, CD, AG, AF, CE

✔ **– 0 pts** Correct for b, AB, AG, AF, CF, CD, CE

Question 3

Problem 3

30 / 30 pts

✔ **– 0 pts** Correct

Question 4

Problem 4

30 / 30 pts

✔ **– 0 pts** Correct

Question assigned to the following page: 1

Problem 0 (Course Evaluation, 0 points)

If you haven't done so already, please fill out the course evaluation at `https://go.blueja.io/N1IjMHTspU-bSehiAVM6_A` or by scanning the QR code below.

Problem 1 (MST Updates, 20 pts)

Let G be an undirected and connected graph with positive weights, and T be a minimum spanning tree T of the graph.

(a) Suppose that we decrease the weight of one of the edges in T. Will T still be an MST of G? If yes, provide a proof; if no, provide a counterexample.

(b) Suppose that we **increase** the weight of one of the edges **not** in T. Will T still be an MST of G? If yes, provide a proof; if no, provide a counterexample.

Solution

(a) Suppose we decrease the weight of some edge $e \in T$. Let the original weight be $w(e)$ and the new weight be $w'(e) = w(e) - \delta$ for some $\delta > 0$. For any spanning tree T' of G, write

$$\mathrm{wt}(T) = \sum_{x \in T} w(x), \quad \mathrm{wt}'(T) = \sum_{x \in T} w'(x), \quad \mathrm{wt}(T') = \sum_{x \in T'} w(x), \quad \mathrm{wt}'(T') = \sum_{x \in T'} w'(x).$$

There are two cases:

- If $e \notin T'$, then $\mathrm{wt}'(T) = \mathrm{wt}(T) - \delta$ while $\mathrm{wt}'(T') = \mathrm{wt}(T')$. Since T was an MST, $\mathrm{wt}(T) \leq \mathrm{wt}(T')$, so

$$\mathrm{wt}'(T) = \mathrm{wt}(T) - \delta < \mathrm{wt}(T) \leq \mathrm{wt}(T') = \mathrm{wt}'(T'),$$

  and hence $\mathrm{wt}'(T) < \mathrm{wt}'(T')$.
- If $e \in T'$, then both $\mathrm{wt}'(T)$ and $\mathrm{wt}'(T')$ decrease by δ, so the original inequality $\mathrm{wt}(T) \leq \mathrm{wt}(T')$ becomes $\mathrm{wt}'(T) \leq \mathrm{wt}'(T')$.

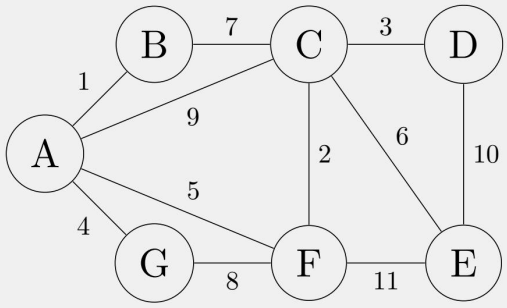
In either case, no spanning tree can beat T under the new weights, so T remains an MST.

(b) Suppose we increase the weight of some edge $f \notin T$. Let the original weight be $w(f)$ and the new weight be $w'(f) = w(f) + \delta$ for some $\delta > 0$. By the cycle property of MSTs, adding f to T creates a unique cycle C. In the original MST, every edge $e \in C \cap T$ satisfies $w(e) \leq w(f)$. After increasing f, we have $w(e) \leq w(f) < w'(f)$. Now consider any spanning tree T' that uses f. Removing any edge $e \in C \cap T$ from $T' \cup \{e\}$ yields a spanning tree of strictly smaller weight (since $w(e) < w'(f)$). Thus no spanning tree that includes f can be optimal under the new weights, and since T never used f, it is still an MST.

Question assigned to the following page: 2

Problem 2 (MST Algorithms, 20 pts)

Consider the following undirected graph $G = (V, E)$.



(a) Illustrate a run of Kruskal's algorithm on this graph by filling in the table below. State at each step which edge is added to the tree. We have filled the first step in for you. Here, **we only want the edge added to the tree and not every edge that is considered.**

Step	1	2	3	4	5	6
Edge Added	AB					

(b) Illustrate a run of Prim's algorithm on this graph starting from vertex A by filling in the table below. State at each step which edge is added to the tree. We have filled the first step in for you. Here, **we only want the edge added to the tree and not every edge that is considered.**

Step	1	2	3	4	5	6
Edge Added	AB					

Solution

(a) **Kruskal's algorithm.**

Step	1	2	3	4	5	6
Edge Added	AB	CF	CD	AG	AF	CE

(b) **Prim's algorithm.**

Step	1	2	3	4	5	6
Edge Added	AB	AG	AF	CF	CD	CE

Question assigned to the following page: 3

Problem 3 (Pizza Delivery, 30 points)

Alice, who lives at a vertex s of a directed, weighted graph $G = (V, E)$ (with non-negative weights), is going to her friend's house at vertex h for dinner. Naturally, Alice wants to get from s to h as soon as possible, but it appears that her friend is terrible at cooking, so along the way she wants to get a pizza from a pizza store just in case her friend cooks something inedible. Let's say the pizza stores form a subset of the vertices $B \subset V$. Thus, starting at s, Alice must go to some vertex $b \in B$ of her choice, and then head from b to h using the shortest overall route possible (assuming she wastes no time at the pizza store). We can help Alice reach h as soon as possible, by solving the following sub-problems.

1. Compute the shortest distance from s to all pizza stores $b \in B$.
2. Compute the shortest distance from every pizza store $b \in B$ to h. Note that this is the dual of the single-source shortest path where we are now asking for the shortest path from every node to a particular destination.
3. Combine part 1 and 2 to solve the full problem.

A straightforward solution is to run Dijkstra's algorithm twice (once in part 1 and once in part 2). In this problem, you will improve this solution by running Dijkstra's algorithm only **once**. Specifically, you should define a new graph G' on $2|V|$ vertices and at most $2|E| + |V|$ edges (and appropriate weights for these edges), so that the original problem can be solved using a single Dijkstra call on G'. Briefly argue correctness and runtime of your proposed solution. (*Hint: This is another problem on reductions. You might want to refresh your memories on Problem 2 in Homework 8, which is also a reduction problem.*)

Solution

We build a new directed graph $G' = (V', E')$ with vertex set and edge set as follows:

- $V' = \{\, v^1, v^2 : v \in V\}$. Think of v^1 as "before picking up pizza" and v^2 as "after picking up pizza."
- For each original edge $(u \to v) \in E$ of weight w, include

$$(u^1 \to v^1) \text{ with weight } w, \quad (u^2 \to v^2) \text{ with weight } w.$$

- For each pizza-store vertex $b \in B$, include a "switch" edge

$$(b^1 \to b^2) \quad \text{of weight } 0.$$

Now run Dijkstra's algorithm once on G' from source s^1. The distance to h^2, denoted $\text{dist}'(s^1, h^2)$, equals

$$\min_{b \in B} \big(\text{dist}_G(s, b) + \text{dist}_G(b, h)\big),$$

because any path in G' from s^1 to h^2 must travel along "before"-edges from s^1 to some b^1, cross the zero-cost edge into b^2, and then follow "after"-edges to h^2. Thus the shortest path in G' corresponds exactly to the optimal choice of pizza-store b.

Question assigned to the following page: 3

Runtime.

$$|V'| = 2\,|V|, \quad |E'| = 2\,|E| + |B| \;\le\; 2\,|E| + |V|.$$

Dijkstra on G' takes

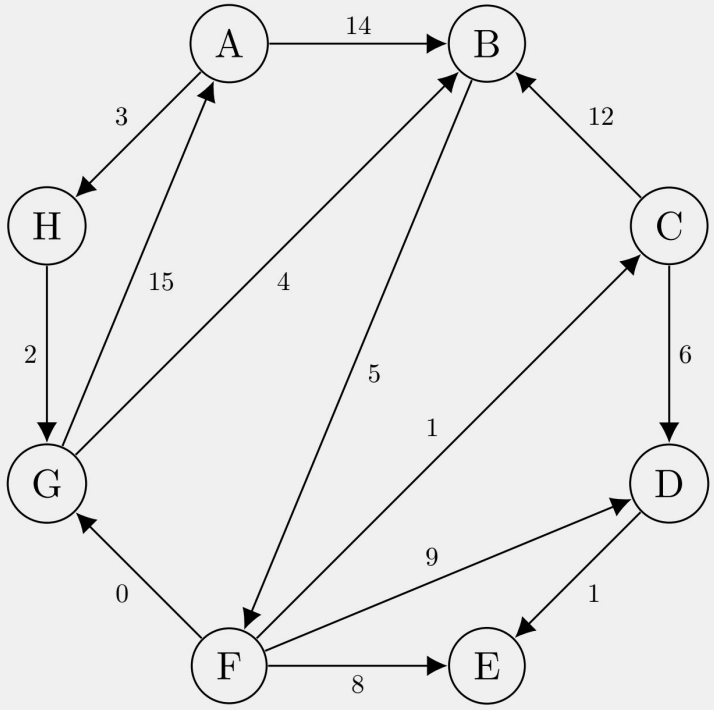
$$O\big(|E'|\log|V'|\big) = O\big((E+V)\log V\big),$$

Correctness. Each path from s^1 to h^2 in G' encodes a path in G that first goes from s to some pizza store $b \in B$ (using only "before"-edges), switches at cost zero to "after"-mode at b, and then proceeds from b to h (using only "after"-edges). No other routes are possible in G'. Hence the shortest-path computation on G' exactly minimizes $\mathrm{dist}_G(s,b) + \mathrm{dist}_G(b,h)$ over $b \in B$, as required.

Question assigned to the following page: 4

Problem 4 (Dijkstra Practice, 30 points)

Show the running of Dijkstra's algorithm for finding the distance from A to all other vertices in the following graph. You should show the updated distances to all the vertices (i.e., the key of the vertex in the priority queue) after each step in the algorithm.



For the ease of representing your solution, simply complete the following table. The first row (Step 0) represents the initial values.

	Vertex extracted from PQ	dist(A)	dist(B)	dist(C)	dist(D)	dist(E)	dist(F)	dist(G)	dist(H)
0	-	0	∞	∞	∞	∞	∞	∞	∞
1	A								
2									
3									
4									
5									
6									
7									
8									

Question assigned to the following page: 4

Solution

Step	Vertex extracted	dist(A)	dist(B)	dist(C)	dist(D)	dist(E)	dist(F)	dist(G)	dist(H)
0	–	0	∞	∞	∞	∞	∞	∞	∞
1	A	0	14	∞	∞	∞	∞	∞	3
2	H	0	14	∞	∞	∞	∞	5	3
3	G	0	9	∞	∞	∞	∞	5	3
4	B	0	9	∞	∞	∞	14	5	3
5	F	0	9	15	23	22	14	5	3
6	C	0	9	15	21	22	14	5	3
7	D	0	9	15	21	22	14	5	3
8	E	0	9	15	21	22	14	5	3