COMPUTER SCIENCE 311
SPRING 2020
FINAL EXAM

Instructions

- You have **30** hours to complete the exam, including the uploading time of your solutions.
- There are four questions and one bonus question. The maximum possible score is 70; the number of points for each problem is indicated on the next page.
- Read all questions carefully before starting.
- Since the exam is conducted remotely, questions should be asked on **Piazza only**. Note that we do **NOT** answer questions through emails.
- All questions should be for clarification purposes only. Please keep your questions concise. We CANNOT have lengthy discussions like we usually answer homework questions on Piazza.
- Due to the challenges of conducting the exam remotely, we can only try to answer the questions in time when we see them. However, we cannot guarantee how quickly we can respond. If you have questions, please ask early.
- We expect clear and concise answers. Think carefully before you write; make sure that every word you write counts.
- If you do not know the answer to a problem, write "*I don't know*" and (except where otherwise stated) you will earn approximately 20% of the points for that problem.
- When asked to provide an algorithm, remember to
  - describe the algorithm clearly,
  - analyze the running time of the algorithm.
- For any algorithm, your grade will depend on the efficiency of your algorithm.

Write Your Name and Your Recitation Section Number Here

Your Score

Problem	Points	Score
1	15	
2	15	
3	15	
4	15	
5 (Extra)	10	
Total	60 + 10 = 70	

PROBLEM 1: SHORT QUESTIONS (NO CREDIT FOR WRITING "I DON'T KNOW") (15 POINTS)

(a) (3 points) **True or False:** Every directed graph can have more than one minimum spanning tree. If true, give a short explanation. If false, give a counterexample.

`True`

Its possible to have multiple paths with the same weight. Suppose we have to spanning paths with the same weight then if one of theses paths is a minimum spanning path they both must be minimum spanning. This gives more then one minimum spanning path.

(b) (3 points) Suppose G is a directed graph with n vertices and m edges, and let s be any vertex in G.

True or False: *If every vertex of G is reachable by a directed path from s, then* $m \geq n - 1$ *edges.* If true, give a short explanation. If false, give a counterexample.

`True`

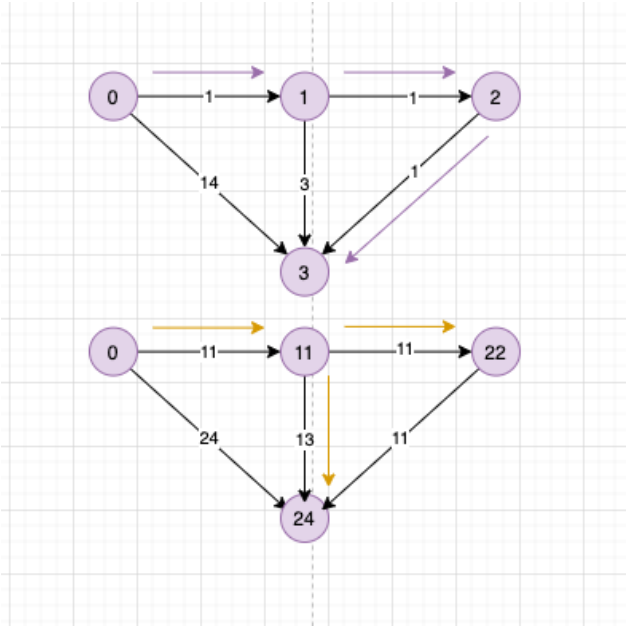
Proof. For a directed graph G lets supports all edges have a weight of 1. Then for there to be a minimal spanning tree. There needs to exist a path from a vertex s $\in G$ to all other nodes. Suppose there isnt a incoming reference to s then there is a n-1 nodes to be connected to s thus there are n-1 outgoing edges in G to have all vertices connected from s. giving us the minimum about of edges needed to have a spanning tree.

thus you'll need n-1 edges in graph G to have a connected graph. w □

(c) (3 points) Suppose we are given an undirected graph $G = (V, E)$ where every edge $e \in E$ has a distinct positive weight w_e. Let T be a minimum spanning tree of G. Now, suppose we add 10 to the weight w_e of every edge.

True or False: *T is still a minimum spanning tree in G with the new edge weights.* If true, give a short explanation. If false, give a counterexample.

`False`

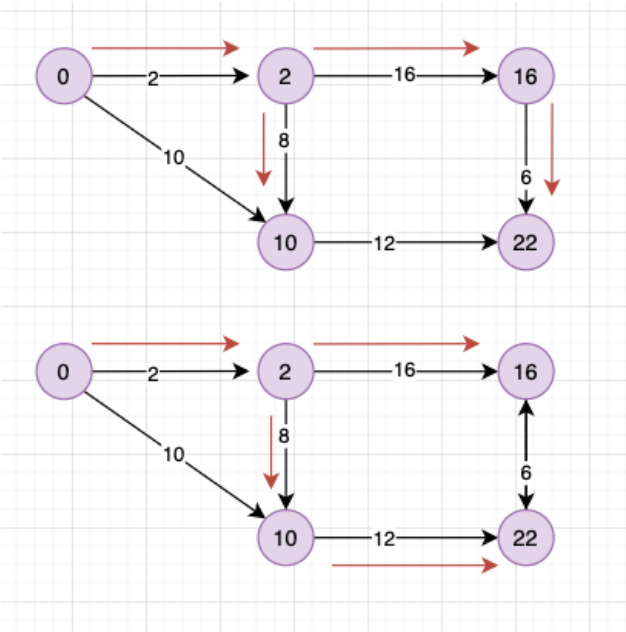


Seen above, by adding 10 to all edges between graph 1 and 2 the minimum spanning tree changes and so the addition doesn't don't preserve the minimum spading tree.

(d) (3 points) Suppose we are given a directed graph $G = (V, E)$ where every edge $e \in E$ has a distinct positive length ℓ_e, and s is a specified source vertex in V. Assume that every vertex in V is reachable from s.

True or False: *G has at most one shortest-paths tree.* If true, give a short explanation. If false, give a counterexample.

`False`: consider.



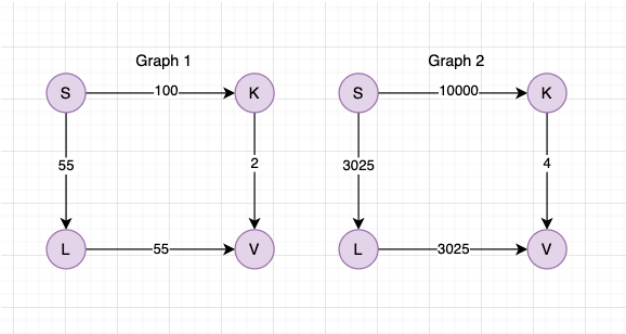
since each path holds the same weight both are considered a minimum path from s to v making this a valid minimum path tree.

(e) (3 points) Suppose we are given a directed graph $G = (V, E)$ where every edge $e \in E$ has a distinct positive length ℓ_e. Let s and t be two nodes in G and let P be a shortest $s \rightsquigarrow t$ path in G. Now, suppose we replace the length ℓ_e of each edge e by $\ell'_e = \ell_e^2$.

True or False: *P is still a shortest $s \rightsquigarrow t$ path in G with the new edge lengths.* If true, give a short explanation. If false, give a counterexample.

`False`

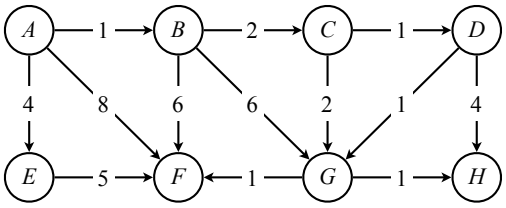
suppose we have the graph.



Let P be the shortest path between S and V. In graph 1 the shortest path from S to V is obviously is $S- > K- > V = 102$. Now let $\ell'_e = \ell_e^2$. in graph 2 the shortest path is from S->L->V = 6050 $< 100^2 + 2^2$ thus is not the same path after the operation.

Problem 2: Shortest Paths (15 Points)

Consider the following directed graph, where every edge has a specified length:



(a) (10.5 points) Run through Dijkstra's algorithm step-by-step to calculate shortest paths from A to every other node. Show your steps in the following table as the algorithm proceeds. For example, in the row corresponding to Iteration 1, B.d denotes the distance from A to B in iteration 1, and B.p denotes the preceding node of node B in iteration 1. Whenever there is a tie in choosing from a set of nodes, break the tie in alphabetical order.

Iter. #	A.d	A.p	B.d	B.p	C.d	C.p	D.d	D.p	E.d	E.p	F.d	F.p	G.d	G.p	H.d	H.p
Init.	0	Nil	∞	Nil	∞	Nil	∞	Nil	∞	Nil	∞	Nil	∞	Nil	∞	Nil
1	0	Nil	1	A	∞	Nil	∞	Nil	4	A	8	A	∞	Nil	∞	Nil
2	0	Nil	1	A	**3**	**B**	∞	NIL	4	A	**7**	**B**	**7**	**B**	∞	Nil
3	0	Nil	1	A	3	B	**4**	**C**	4	A	7	B	**5**	**C**	∞	Nil
4	0	Nil	1	A	3	B	4	C	4	A	**6**	**G**	5	C	**6**	**G**
5	0	Nil	1	A	3	B	4	C	4	A	6	G	5	C	6	G

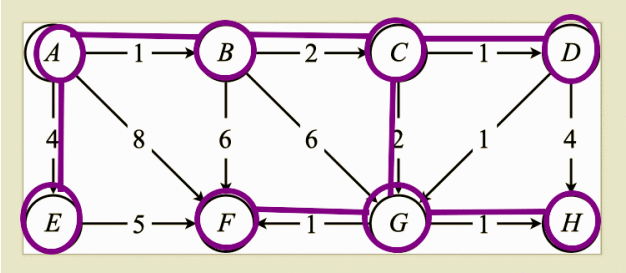
Extra details. Bold numbers are updated nodes at that level. Now I did not include nodes that did not update any other nodes. For example after 1 we check each B,F,E for other connections and by starting at B we update F in init 2 and then look at F and E. Since F has the lowest path so fare we don't update anything and its left out of the iterations. Below is each node corresponding the its iteration.

1 A
2 B
E,F nothing happens
3 C
4 G
D nothing happens because H is already updated from G. 5 H, done

shortest path A->B->C->G->H □

(b) (4.5 points) Based on the table you filled in for part (a), draw the tree of the shortest paths from A to every other node in the graph.

The minimal spanning tree of the graph.


A
B
C
D
E
F
G
H
1
2
1
4
8
6
6
2
1
4
5
1
1

Problem 3: Greedy Algorithms (15 Points)

Consider a country road that runs in a straight line from East to West with houses scattered very sparsely along it. Our goal is as follows.

Solution similar to the interval partitioning this problem can be done by treating cell towers as classes and we want to minimize the number of cell towers.

> Find a way to place cell phone base stations at certain points along the road, so that every house is within four miles of one of the base stations.

(a) (8 points)

Algorithm 1: Greedy Solution(T,H)

```
1  //sort H from least to greatest letting 0 indicate the left most house using Merge. T = List of
     Towers H = MergeSort(H)
2  int index = 0;
3  while |H| > 0 do
4  |   T_new = new Tower(H[index].position+4_miles)
5  |   T.add(T_new )
6  |   //remove all nodes in H that are within the bounder of of T_new
7  |   h = H[index]
8  |   for i = index ; |H| > 0 and h.position <= T_new.position ; i++ do
9  |   |   remove h from H
10 |   |   h = H[i] index++;
11 return T
```

(b) (4 points) Justify the correctness of your algorithm (i.e., explain why your algorithm finds the placement that uses the fewest base stations).

Observation: Each tower is a circle of radius 4 miles by at the closes 8 miles from anther tower. This will gives an maximal area covered

Proof of correctness.

Let T = $\{t_1, ..., t_n\}$ be the set of solution created by my algorithm and let Let T' = $\{t'_1, ..., t'_k\}$ be a set of solutions. let both of else list be sorted from left to right in starting at index 0 being the left most tower. Now suppose Greedy Solution id not result in the optimal solution. Then there is a $t \in T'$ such that t is not 4 miles from a tower. But we can swap this tower with a tower exactly 4 miles from house t. Over all the T' this can be done to achieve a equivalent solution or better solution. This Greedy Solution creates a optimal solution

□

(c) (3 points) Analyze the running time of your algorithm.

Solution. There are two worst case scenario one is if all nodes are within the first tower, and the second is if all nodes are more then 4 miles apart.

scenario 1:

Then the running time starts with mergeSort taking O(nlogn). Then the while loop is accessed once. with in the while loop the remove loop removed all nodes running in O(n) since inside the this loop runs in consent time. Thus running in O(nlogn+n) = O(nlogn)

scenario 2: Then the running time starts with mergeSort taking O(nlogn). then this while loop runs n times each time deleting one element. making the run time O(nlogn). As seen there is a compilation between removing elements inside the while loop and the Max number of iterations the while loop and make.

Thus the worst case is always O(nlogn).

□

Problem 4: Unique Shortest Paths (15 Points)

We are given an undirected graph $G = (V, E)$ where each edge $e \in E$ has a length $\ell_e > 0$, and a starting vertex $s \in V$. Let $n = |V|$ and $m = |E|$. Suppose that, in addition to `dist` and `pred` fields, each node $u \in V$ has a Boolean field u.`usp`. Give a $O((m+n)\log n)$-time algorithm that, for each node $u \in V$, sets u.`usp` to `true` if there is a unique shortest path from s to u; otherwise, it sets u.`usp` to `false`. (*Note:* s.`usp` == `true`.)

(a) (8 points) Pseudocode of your algorithm:

We can achieve this algorithm by modifying dijkstra's algorithm with a binary heap.

Algorithm 2: usp(node s)

```
1  List<priority queue> queue
2  for each vertex u do
3  |   u.dist = ∞
4  |   u.pred = null;
5  queue.inset(s)
6  s.distance = 0
7  while queue is not empty do
8  |   u = queue.ExtractMin()
9  |   for iterate threw all children of u do
10 |   |   node child = adjacent vertices
11 |   |   if child.dist != ∞ then
12 |   |   |   // check shortest path and usp
13 |   |   |   if child.dist > u.dist + weight then
14 |   |   |   |   child.dist = u.dist + weight
15 |   |   |   |   minheapify(queue)
16 |   |   |   |   child.pred = u
17 |   |   |   |   child.ups = True
18 |   |   |   else if child.dist = u.dist + weight then
19 |   |   |   |   child.ups = False
20 |   |   else
21 |   |   |   child.dist = u.dist + weight
22 |   |   |   queue.insert(u)
23 |   |   |   child.pred = u
24 |   |   |   child.ups = True
```

(b) (4 points) Brief justification for your algorithm:

Threw Dijkstra's algorithm we visited every node and every path within the graph. Then all we need to do is update all nodes usp while visiting nodes if we visit a node with the same path its path is currently not unique and will be set to false. This value will be override if we find a shorter path to x node making x.usp = True.

By setting ups True when first seeing a node we can guarantee later on that if we find a path with the same dist we change ups to false.

(c) (3 points) Running time analysis for your algorithm:

By using a priority queues with a binary heap we know that Insert, DecreaseKey, ExtractMin , and heapify all run in $O(logV)$). so " V + V calls to insert + V calls to ExtractMin() + E calls to minheapify" giving us $O(v + (2V + E)logv) = O((2n + m)logn) = O((n + m)logn)$

Minheapify is called when every a child dist is changed inside the queue thus it can only happen with respect the number of Edges inside the graph.

Problem 5: Dynamic Programming (10 Extra Credit Points)

Note. *For this problem, there is **no** partial credit for answering "I don't know."*

Andy's (a pizza chain) is considering opening a series of restaurants along Scenic Valley Highway (SVH). The n possible locations are along a straight line, and the distances of these locations from the start of SVH are, in miles and in increasing order, $m_1, m_2, \ldots, m_n$, where each m_i is a positive integer. The constraints are as follows:

- At each location, Andy's may open at most one restaurant. The expected profit from opening a restaurant at location i, $i \in \{1, 2, \ldots, n\}$, is p_i, where $p_i > 0$.
- Any two restaurants should be at least k miles apart, where k is a positive integer.

Give an efficient dynamic programming algorithm to compute the maximum expected total profit subject to the given constraints.

(a) (3 points) Derive the Bellman equation for this problem:

(b) (2 points) Brief justification for your algorithm:

(c) (3 points) Pseudocode of your algorithm based on the derived Bellman equation in Part (a):

(d) (2 points) Running time analysis for your algorithm: