EC 504
Final Exam, Spring 2021

May, 2021

Instructions

- Put your name on the first page of your answer sheet.
- This exam is open book/notes, but no consultations with anyone other than the instructors are permitted during the exam. Specifically, don't use your keyboard unless you are using the chat window. During the exam, you must be logged into Zoom with your camera on. Try to log in 10 minutes before the exam.
- You have one hour and 55 minutes to complete this exam. There are 8 questions. All work must be shown to be counted.
- At the end of the exam, you will have 15 minutes to scan and upload your work to Gradescope. Please do not delay as Gradescope will close at 5:10 Boston Time.

1. (20 pts) Answer True or False to each of the questions below, **AND YOU MUST PROVIDE SOME FORM OF EXPLANATION (very brief is OK)**. Each question is worth 2 points. Answer true only if it is true as stated, with no additional assumptions.

   (a) In a forward star representation of a directed graph, the edges must be sorted by start node.

   (b) Suppose T_1, T_2 are minimum spanning trees for the same graph G. Then the largest edge weight of T_1 must be the same weight as the largest edge weight of T_2.

   (c) In Bellman-Ford's shortest path algorithm, the temporary distance labels D assigned to vertices which have not yet been scanned (i.e. left the work queue) can be interpreted as the minimum distance among all paths from the start vertex to the given vertices with intermediate vertices restricted to the vertices that have already been scanned (left the work queue).

   (d) Consider a minimum spanning tree T in a weighted, undirected graph G. Suppose we decrease the weight of a single edge that is in the minimum spanning tree T. Then, it is possible that T is no longer a minimum spanning tree in the modified graph.

   (e) Consider a minimum spanning tree problem in a connected, weighted graph with positive and negative weights which can include negative weight cycles. Then, both Prim's and Kruskal's algorithms can be used to find the minimum weight spanning tree in this graph.

   (f) Consider an NP-complete decision problem. Let n denote the size of a problem instance. Then, any candidate solution for an instance can be evaluated to determine whether the answer is true or not in time $O(n^k)$ for some nonnegative integer k.

   (g) A problem is said to be NP-complete if it can be reduced to an instance of every other problem which is in class NP.

   (h) Given a weighted, directed, dense graph with non-negative weights, consider the problem of finding whether the longest simple path in the graph that starts with a given origin node o has length greater than or equal to a threshold T. This problem is in class NP.

   (i) The class NP consists of all decision problems which cannot be solved in time $O(n^k)$ for some positive integer k, where n is the size of the problem instance description in bits.

   (j) Every problem in class NP is also NP-Hard.

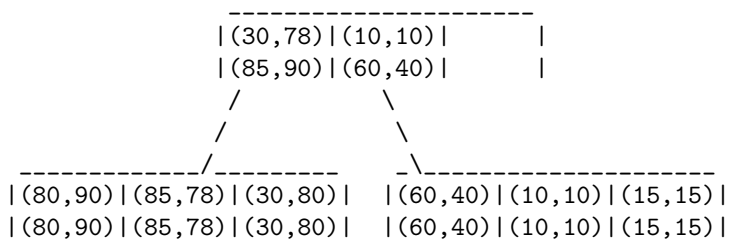
2. (15 pts) Consider the following 2-dimensional set of points:

$$(80, 90), (85, 78), (60, 40), (10, 10), (15, 15), (30, 80)$$

(a) Arrange the elements in a PR quadtree with value range 0-96 for both the first and second coordinates. Display the answer as a tree, with children arranged left-to-right in the following order: SW, SE, NW, NE.

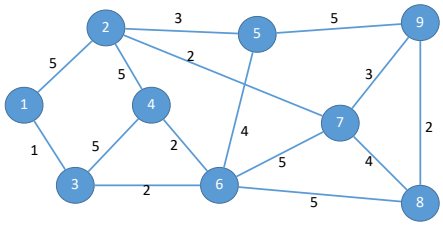
(b) Form a balanced 2-d tree using the median of the elements to split, with all the keys at the leaves of the tree. When there are an even set of points to split at a median value, select the larger of the two possible median values as the median navigation key. Assume also that when a key is equal to the navigation key, it is added to the right subtree (that is, greater than or equal).

(c) Below is an R-tree of order $M = 3, m = 1$, where the minimum number of children (and keys) is 1, formed using linear splitting, with the following variation: when splitting an area defined by points, use as seeds the two points that are farthest in distance from each other. The R-tree already contains the six points: $(80, 90), (85, 78), (60, 40), (10, 10), (15, 15), (30, 80)$. Rectangles are displayed as pairs of points corresponding to the SW and NE corners. Assume we insert into areas with smallest perimeter increase.

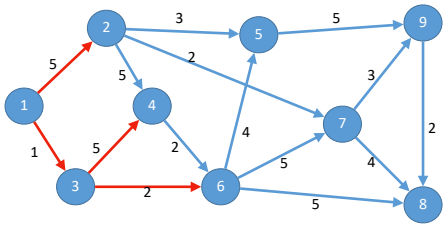


Insert the following key in the R-trees: (45,85), and display the resulting R-tree.

3. (10 points) Consider the undirected graph on the right, where the numbers on edges consist of weights. Find a minimum spanning tree for the graph on the right. Indicate what algorithm you are using, and show some of the work associated with that algorithm. Display the minimum spanning tree graphically, and show the total weight of the minimum spanning tree.



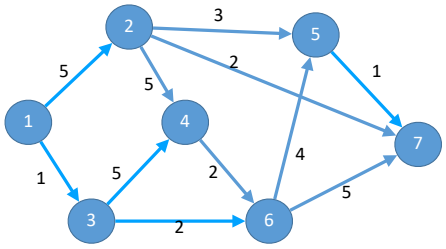
4. (10 points) Consider the directed graph on the right, where the numbers on edges consist of distances. We are going to find a shortest path tree from node 1 to every other node using Dijkstra's algorithm. The figure shows the partial tree grown after five iterations, scanning vertex 1, then vertex 3, then vertex 6, then vertex 2, and then vertex 4.



(a) What are the next two vertices to be scanned (i.e. popped from the priority queue), and what are the distances from 1 to those vertices?

(b) Draw the shortest path tree from vertex 1 to all other vertices.

5. (15 points) Consider the directed graph on the right, where the numbers on edges consist of edge capacities. Our goal is to find the max flow from vertex 1 to vertex 7. As such, we will do it using both the Ford-Fulkerson algorithm, and the Preflow-push algorithm, as guided by the questions below.



(a) Assume that we have made an augmentation of one unit of flow on path 1 - 3 - 6 - 7, and another augmentation of 2 units of flow on path 1 - 2 - 7. Construct the residual network, with residual capacities on each forward and reverse edge.

(b) On that residual network, find the shortest (least number of hops) augmenting path from vertex 1 to vertex 7, and determine its capacity.

(c) What is the max flow from vertex 1 to vertex 7 in this network?

(d) Assume we are starting over with no flow in the network, and we are going to use the Preflow-Push algorithm to find the max flow. Compute the initial vertex excesses and the initial vertex distances to vertex 7, including the initial distance label for vertex 1.

(e) For the first iteration, we start with the largest excess, and push 2 units of flow from vertex 2 to vertex 7, a saturating push. Draw the residual graph that remains after this push, showing the excess at vertices with excess.

(f) Assume we want to execute another push from vertex 2, that still has excess 3. Describe the steps required to perform this push, and draw the residual graph that remains after the push is complete, together with excesses at the vertices with excess.

6. (10 pts) Consider the following knapsack problem: We have six objects with different values and V_i and sizes w_i. The object values and sizes are listed in the table below:

Object number	1	2	3	4	5	6
Value	10	11	12	13	14	15
Size	1	1	2	3	4	3

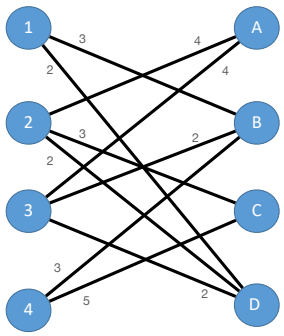
(a) Suppose we are given a knapsack with total size 11, and you are allowed to use fractional assignments. What is the maximum value that fits in the knapsack for the above tasks?

(b) Assume we are not allowed fractional assignments. For the knapsack with total size 11, the dynamic programming requires computing $Opt(j, k)$, the best value considering only objects $1, \ldots, j$, using total size k. The table below has partially computed these numbers. Compute the remaining elements in the table, $Opt(j, k), j = 5, 6; k = 5, 6, 7, 8, 9, 10, 11$.

Capacity-> Max .Task	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	10	10	10	10	10	10	10	10	10	10	10
2	0	11	21	21	21	21	21	21	21	21	21	21
3	0	11	21	23	33	33	33	33	33	33	33	33
4	0	11	21	23	33	34	36	46	46	46	46	46
5	0	11	21	23	33							
6	0	11	21	23	33							

(c) What is the maximum value of the tasks that fit in the knapsack with size 11?

7. (10 pts) Consider the sparse assignment problem shown in the figure on the right, where the costs of each edge are indicated in the figure. We are interested in computing the minimum cost assignment using the successive shortest path algorithm.



(a) To begin the algorithm, what are the initial prices assigned to vertices A, B, C, D?

(b) Given the initial prices, what are the reduced costs of the edge $\{1, B\}$ and the edge $\{4, C\}$?

(c) Now, assume we have proceeded with the algorithm, and we have made assignments of vertex 1 to vertex D, vertex 2 to vertex A, and vertex 3 to vertex D, all which have reduced cost 0, and the prices are still the same as the initial prices. Compute and show the residual network with the reduced cost distances, and find the final shortest augmenting path on the network.

(d) Using the original costs, what is the cost of the optimal assignment you found?

8. (10 points) Consider an array of non-negative integers, with positions numbered from 1 to n, where the value at a position represents the size of the maximum jump forward that can be made from that position. For instance, $[1, 2, 4, 0]$ means that starting at position 1, you can take a jump forward of size 1 to position 2; starting at position 2, you can take a jump forward of either 1 or 2 positions; starting at position 3, you can take a jump forward of 1, 2, 3, or 4 positions, and starting at position 4, you can't take any jumps forward.

(a) Given an array of nonnegative integers $[a_1, a_2, \ldots, a_n]$, describe an algorithm for finding the minimum number of jumps required to go from position 1 to position n, where the size of the jump from position i must be less than or equal to a_i.

(b) Estimate the computational complexity of the above algorithm as a function of n.

(c) Use the algorithm to compute the minimum number of jumps to go from position 1 to position 11 for the array

$$[2, 3, 1, 3, 4, 2, 1, 2, 2, 1, 0]$$