Full Name: Alex Miller CNetID: 12177451

CS162: Honors Introduction to Computer Science II (Winter '19)

Exam II

March 13, 2019

Instructions

- This exam is closed-book, with one double-sided handwritten sheet of notes permitted.
- You have 50 minutes to complete the exam.
- There are a total of 5 problems, on 12 pages.
- Please read each problem carefully and write down your answer in the space provided under the question.
- Good luck!

Full Name: Alex Miller CNetID: 12177451

	Question	Points	
Prob 1	True/False	12	6
Prob 2	Heap	7	7
Prob 3	MST	10	10
Prob 4	Union-Find	19	18
Prob 5	Dijkstra's algorithm	15	15
	Total:	63	56

Full Name: Alex Miller CNetID: 12177451

1. **True/False**

For each question, **circle** the correct answer, and write a *short explanation* ($\leq$ 2 sentences) if the statement is true, or give a *counterexample* if the statement is false.

2

(a) (2 points) Any directed graph with no cycles has a topological ordering.

TRUE FALSE

Any DAG has a topological ordering since nodes can only be accessed sequentially; The point at which ~~whether~~ a node can be visited depends on when the node before it.

2

(b) (2 points) In a *min*-heap, the key with *second* minimum priority value is *always* located at the first level directly underneath the root (i.e. as a direct child of the root).

TRUE FALSE

It can't be the case at the 2nd minimum value is anywhere else; since All parents are less on their children, The values that extend from the nodes of the first level are greater than the nodes of the second level. Therefore one of the nodes at the level beneath the root must be the 2nd minimum priority value

0

(c) (2 points) If we add a positive constant number to the weight of every edge in a weighted graph with all positive weights, it does not change the shortest paths.

TRUE FALSE

The ordering of the weights of the edges would not change as they are all shifted upwards equally. Therefore the order in which edges are generated does not change

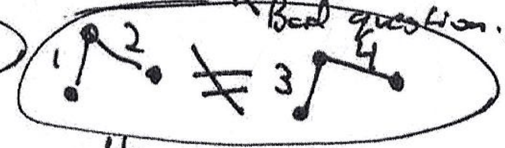
0

(d) (2 points) If we add a positive constant number to the weight of every edge in a weighted graph with all positive weights, it does not change the minimum spanning tree.

Bad question..

TRUE FALSE

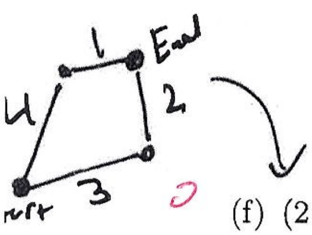
The total weight of the MST changes but not the structure of the MST; It does not mean they are equal.



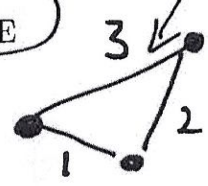
2

(e) (2 points) The smallest three edges in a graph with unique edge weights *always* appear in the minimum spanning tree.

TRUE FALSE



should not be in MST

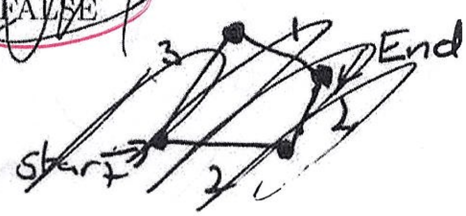


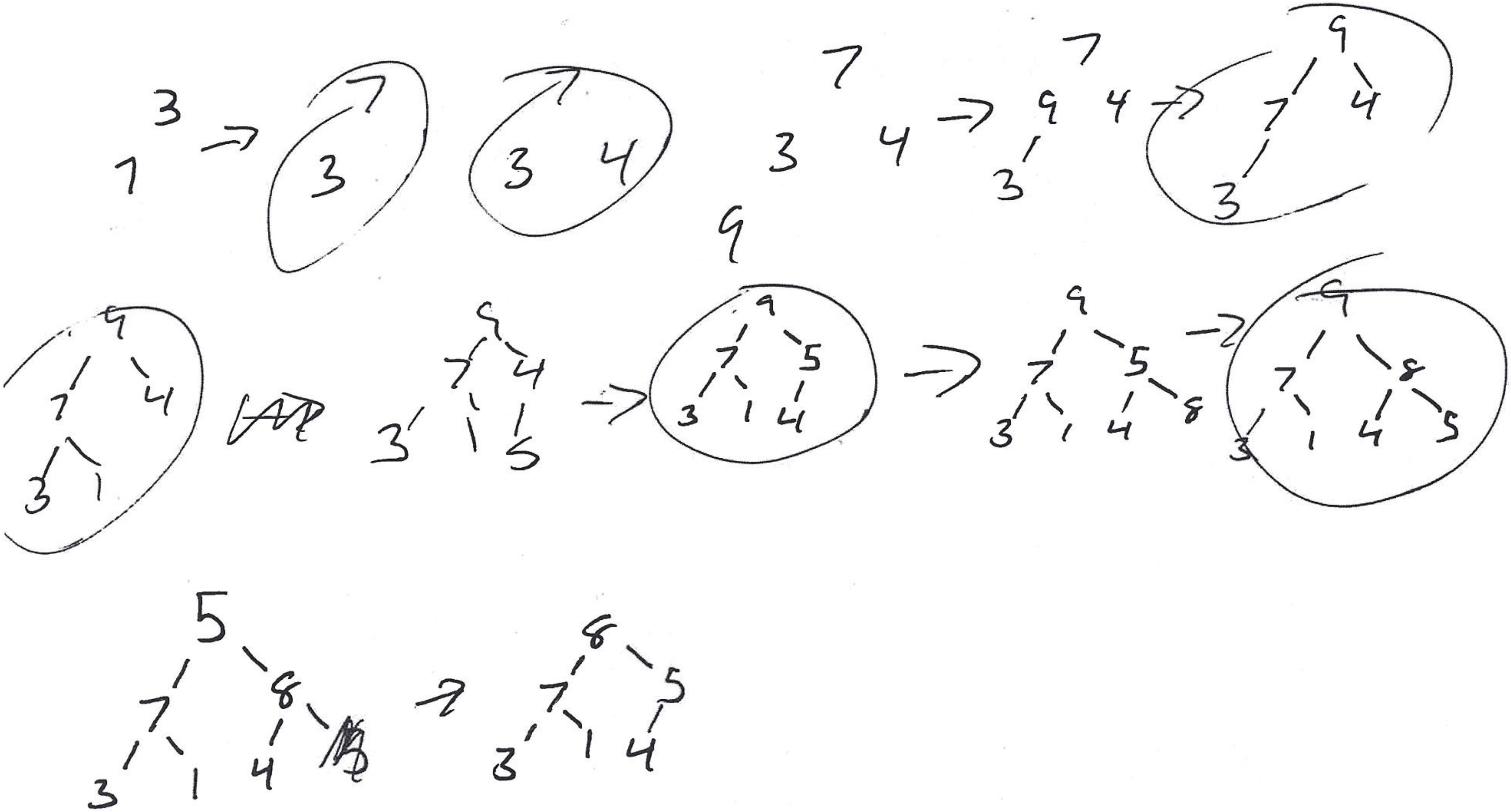
0

(f) (2 points) In a graph with unique edge weights, the shortest path between two vertices is unique.

TRUE FALSE

There can exist multiple shortest paths, But each shortest path is itself unique since edge weights are unique





Full Name: Alex Miller CNetID: ____________

2. **Heap**

A binary heap is a complete binary tree which satisfies the heap ordering property (for either max-heap or min-heap).

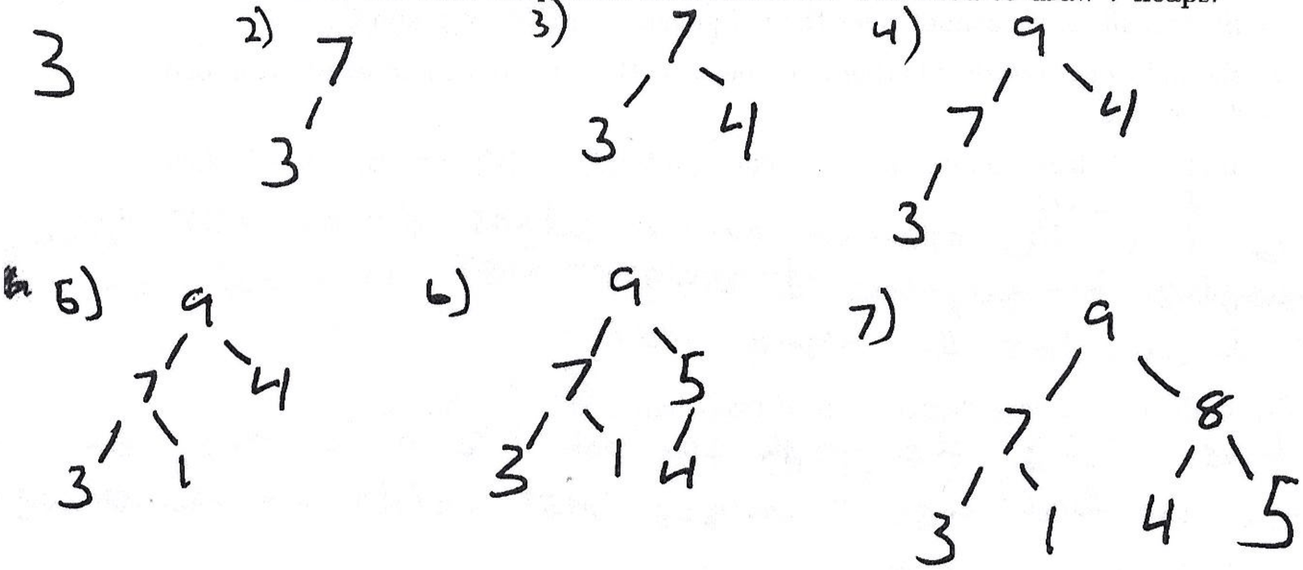
Throughout this section, the word "heap" will always refer to a binary heap.

5

(a) (5 points) Starting from an empty *max*-heap, draw the *max*-heap that results from inserting the following numbers in the order given:

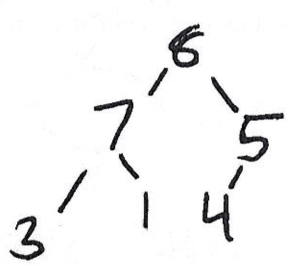
`3 7 4 9 1 5 8`

Show the *max*-heap after each insertion. You need to draw 7 heaps.



2

(b) (2 points) Draw the final state of the *max*-heap after remove the *max* number on this heap from previous task.



Full Name: ______________________ CNetID: ______________

3. **Minimum Spanning Trees (MST)**

You are given a graph $G = (V, E, w)$, where V is the set of vertices, E is the set of edges, and w is the weights for the edges.

5

(a) (5 points) Consider the problem of deciding the uniqueness of minimum spanning trees for an *un-directed connected* graph $G = (V, E, w)$. Suppose we have the following three statements:

- Statement A: graph G must have a unique MST.
- Statement B: graph G must have at least two MSTs.
- Statement C: we cannot infer the uniqueness of MSTs of graph G.

For the following, decide whether statement A, B, or C holds, and **explain** in one *short* sentence.

(i) If graph G has n nodes and $n-1$ edges in total, i.e. $|V| = n, |E| = n-1$, then ___A___ holds.

G has the minimum number edges for an MST, + is connected ~~therefore it only has 1 unique MST~~ is already an MST + only has 1 unique MST

(ii) If all vertices have degree at most two, then ___C___ holds.

If all v have $d \leq 2$, the graph can look like either 

or 

. In the first case, 1 unique MST exists. In the second, 3 unique MSTs exist

(iii) If all edges in graph G have distinct weights, then ___A___ holds.

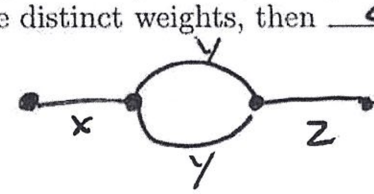
ll edges have distinct weights, Therefore they can be ordered on-arbitrarily + choices in edges must be unique

(iv) If graph G has exactly one pair of edges that have the same weight, and all the rest edges have distinct weights, then ___C___ holds.

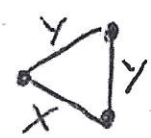
1 unique MST this case

e pair of edges w/ same weight might not connect the same nodes, or they might oth be necessary in the graph. However they might also connect the same vertices. Therefore there are two possible unique MSTs in this case

(v) If graph G contains a cycle with exactly one pair of edges along the cycle having the same weight, and all the rest edges have distinct weights, then ___C___ holds.

The graph could be something like 

in which case there ∃ two unique MSTs

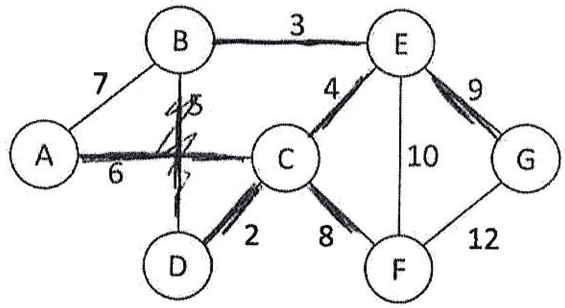
it could also look line 

with $x >> y$, in which case there ∃ only 1 unique MST.

Full Name: ______________________________ **CNetID:** ____________________

5

(b) (5 points) For the following tasks, use this graph:



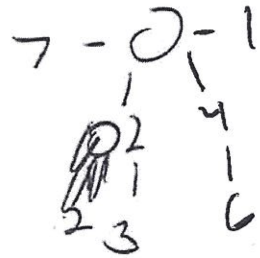
2
3
4
6
8
5

Show the order in which edges will be added to the MST if we run Kruskal's algorithm on the graph. You may use the edge weights as labels.

2
3
4
6
8
9

0 — 1   2 — 3   ~~4-6~~ ~~2→7~~

4 — 5   6 — 7



~~10-12~~

0→1   2→3   4→5   6→~~9~~

2→3   0

↓

4

↓

5

Full Name: ______________________ **CNetID:** ______________

4. **Union-Find**

Consider the *tree-based* union-find data structure on the set of N integers. Note that in this implementation, each node is labeled from $0, \ldots, N-1$, and together they form a *forest*. Each tree in the forest corresponds to a disjoint set, where the root of the tree is the canonical representative of the elements in the tree.

Suppose we perform union-find operations with union-by-height technique, but *without* any path-compression. Recall from lectures and labs, we have the following operations:

Find(v): Starting from node v, we walk upward in the tree until we reach the root.

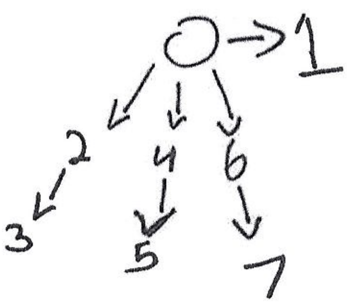
Union(u, v): First we find the representatives of u and v, i.e. $r_u = \text{Find}(u)$ and $r_v = \text{Find}(v)$ respectively. If $H(r_u) \neq H(r_v)$, the root with larger height (say r_u) becomes the new root (i.e. r_u will be the parent of r_v). Otherwise, we pick the root whose index is smaller. Then update the height of the tree accordingly.

By convention, we say the **height** of a node $H(v)$ is the number of nodes (inclusive) along the *longest* path from the node v to a leaf. And the height of a tree $H(T)$ is the height of its root. So a singleton tree has height 1.

5

(a) (5 points) Consider running union-find on singleton sets of 8 elements $(0, 1, \ldots, 7)$. Give a sequence of union operations that will result in a *single* tree with height 3. And then draw the final tree.

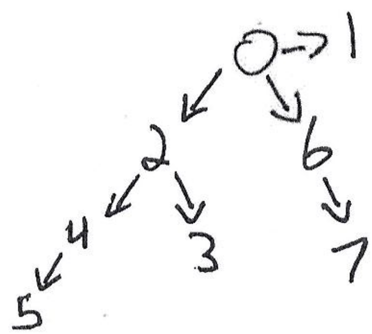
Union(0,1)
Union(2,3)
~~Union(3,4)~~
Union(4,5)
Union(6,7)
Union(1,3)
Union(1,4)
~~Union(6,7)~~ ~~Un~~ Union(1,6)



5

(b) (5 points) Similarly as above, give a sequence of union operations that will result in a *single* tree with height 4. And then draw the final tree.

Union(0,1)
Union(2,3)
Union(4,5)
Union(6,7)
Union(3,5)
Union(1,7)
Union(1,3)



5 nodes

Full Name: Alex Miller **CNetID:** 12177451

what is h?

(c) (5 points) In a forest of N nodes resulted from union-find operations, what is the *maximum* number of nodes that have height $\geq h$? And explain why. (Hint: write down the answer in terms of N and h.)

4

To grow height of ~~the~~ a tree by 1, you must union 2 trees of same height
Therefore to generate node of heigh h, you need 2h nodes

$\frac{N}{2h^{-1}}$

0-1
1
2-7
1
3

0-1
1

1 → 1
2 → 2
3 → 5
4 → 8

Kruskal's algorithm for finding the minimum spanning tree of a graph G can take advantage of Union-Find in order to keep track of all of the trees in the current forest. More specifically with Union-Find, when adding an edge, we can check if adding this edge will generate a cycle and we can easily merge the two connected components together.

(d) (2 points) With this implementation of Kruskal's algorithm, what condition should be checked (in terms of Find and Union operations) before adding an edge (u, v) to the current forest. Explain briefly.

2

Check if the nodes U & V have the same Root; If they have the same root, creating edge (u,v) will create a cycle which violates MST definition

(e) (2 points) With this implementation of Kruskal's algorithm, what is the *exact* number of calls to Union which must be made? (Note: do not use big-O notation.)

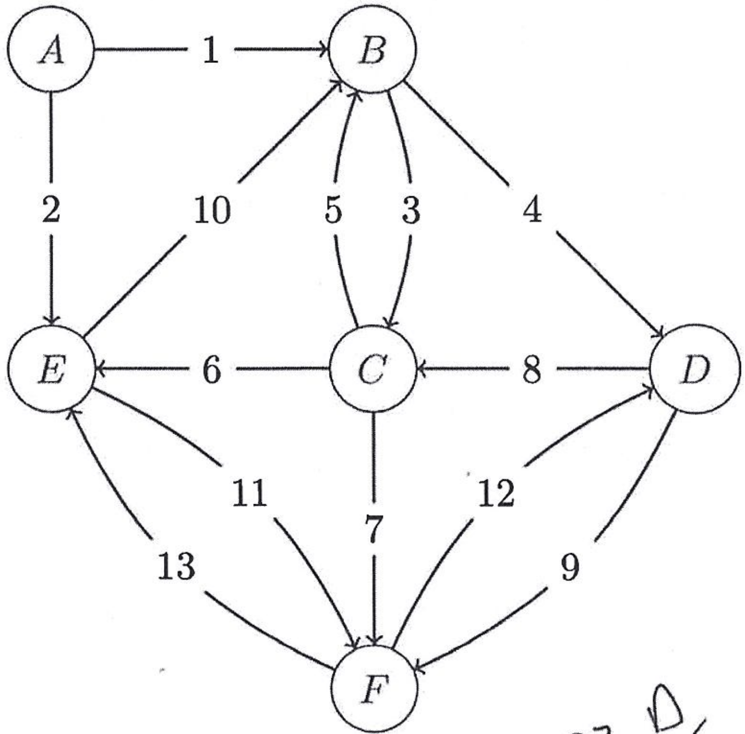
2

|# of Vertices| − 1

Full Name: Alex Miller CNetID: 12177451

5. **Dijkstra's Algorithm**

In this problem you will perform Dijkstra's Algorithm on the graph below, starting with vertex A, and finding the shortest path to every node in the graph.



14 F
13, C

23 D
24, E

(a) (10 points) You will fill out the table (which begins on the next page) as you go – just like on the homework. The first row is already filled in for you, as is the final distance to A. On the last line, the third column should be empty, and there may be extra space in the chart. Also, fill out the table on the right with the final distances to all nodes.

5

(b) (5 points) When Dijkstra's algorithm finishes, we have a table of distances from A to each other node, as you did in the previous part. However, the actual path is not recorded. Using no additional space (you can't add arrays or additional variables for example), describe at a high level how you could use this table of distances as as well as the original graph to find the exact path from A to another node.

Mark the path node to every node ~~when its cost is generated~~ backwards to the node used to generate it's cost only when that cost is less than its current cost.

Full Name: CNetID:

10,E

8

Time	Vertices Visited So Far	State of Priority Queue
0	–	$(0, A)$
1	A	(1,B),(2,E)
2	A B (0,A),(1,B)	(2,E),(4,C),(5,D)
3	A B E (0,A)(1,B)(2,E)	(4,C)(5,D)(12,B),(13,F)
4	A B E C (0,A)(1,B)(2,E)(4,C)	(5,D),(9,B),(10,E),(11,F)(12,B)(13,F)(14,F) 2
5	A,B,E,C,D	(9,B)(10,E)(11,F)(12,B)(13,F)(13,C)
6	A,B,E,C,D,	(10,E),(11,F)(12,B)(13,F)(13,C)(14,F)
7	A,B,E,C,D.	(11,F)(12,B)(13,F)(13,C)(14,F)
8	A,B,E,C,D,F	(12,B)(13,F)(13,C)(14,F)(23,D)(24,E)
9	A,B,E,C,D,F	(13,F)(13,C)(14,F)(23,D)(24,E)
10	A B E C D F	(13,C)(14,F)(23,D)(24,E)
11	A B E C D F	(14,F)(23,D)(24,E)
12	A B E C D F	(23,D)(24,E)
13	A B E C D F	(24,E)
14	AB~~[illegible]~~ECDF	—
15		
16		

	Distance
A	0
B	1
C	4
D	5
E	2
F	11

0
1
4
5
2
11

Scrap Paper

Full Name: ______________________ CNetID: ______________

Scrap Paper