(PRINT) Name: _____ Student No.: _____

Signature: _____ Total Mark: _____ /175

UNIVERSITY OF TORONTO

Faculty of Arts and Science

August 2016 EXAMINATIONS

CSC263H1 —Data Structures and Analysis

Duration - 3 hours

No aids are permitted.

This examination has 19 pages and 10 questions. Check that you have a complete paper. There are three pages at the back for rough work.
To pass the course you must achieve a mark of at least 35% on this exam. Failure to do so will result in automatic failure.
If you do not attempt to answer a question you will receive 20% for that question.
Answer all questions on this paper in the spaces provided. If your answer is outside the space provided, please provide a clear indication of the location of your answer.
Total Marks: 175

Please do not write below this line.

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

1. (*15* pts) Short answer questions. You may use the next page for rough work.

(a) (5 pts) Consider an array A of 11 integers. Say we call $BuildHeap(A)$, the $O(n)$ time heap building algorithm, how many swaps are made in the worst case? **Do not** provide asymptotic notation, provide an exact number.

(b) (5 pts) Consider some *AVL* tree with 5 nodes. How many nodes may we examine in an unsuccessful search in the worst case? **Do not** provide asymptotic notation, provide an exact number.

(c) (5 pts) Consider the following hash function $h(k) = \lfloor k \cdot m \rfloor$, where k is our key and m is the number of buckets. For an arbitrary input distribution, is $h(k)$ a good hash function? Why or why not? Under what conditions would you say it is a good hash function? Do not worry about having a value of m that is too small. Assume each $k \in [0, 1]$, that is the hash function will always return a valid bucket.

Rough Work space for question 1.

2. *(15 pts)* Binomial Heaps. You may use the next page for rough work.

(a) (5 pts) Consider a Binomial Heap with 23 nodes, say we insert a single element. How many trees will be in the new heap? Provide an exact number do not use asymptotic notation.

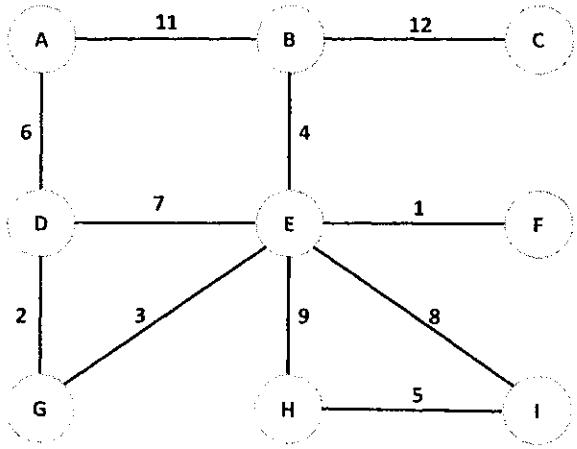
(b) (5 pts) Consider a Binomial Max Heap with 1029 nodes, say we perform a search for the maximum element. How many elements must we examine? You may not assume we have a pointer to the maximum element. Provide an exact number do not use asymptotic notation.

(c) (5 pts) Consider a Binomial Max Heap with 27 nodes, say we extract the maximum element. What is the maximum number of tree mergers that may take place when reconstructing the Binomial Heap? What is the minimum number of tree mergers? Provide exact numbers do not use asymptotic notation.

Rough Work space for question 2.

3. *(10 pts)* Pick one of **Prim's** or **Kruskal's** algorithms to find Minimum Spanning Trees. Using the algorithm you pick, find the MST for the following graph G. For **Prim's**, assume you are starting with E as the initial vertex. For your chosen algorithm in what order are the edges added to the MST? You may use the next page for rough work.

Figure 1: The graph G



Rough Work space for question 3.

4. *(20 pts)* Consider the dynamic array that we studied in class. For this question, assume that we only ever perform append operations and never deletes. When we grow the array instead of doubling its size, we grow it by 25%. That is, if our array A currently has capacity k and it is full, the next insertion causes the array to grow to capacity $\lceil k \cdot 1.25 \rceil$. Use the accounting method to show that the amortized cost of n appends is $O(1)$. You must justify why your charge per append is enough.

5. *(20 pts)* Depth First Search.

(a) *(5 pts)* Consider the following information recorded about the nodes in a graph G during the execution of $DepthFirstSearch(G)$.

	Discovery	Finish
A	1	8
B	3	4
C	2	7
D	5	6

Is the information in the table valid? That is, does there exists a graph G where $DepthFirstSearch(G)$ would record the above information. If so, provide such a graph and annotate the nodes with the appropriate information. Otherwise, explain why the information is incorrect.

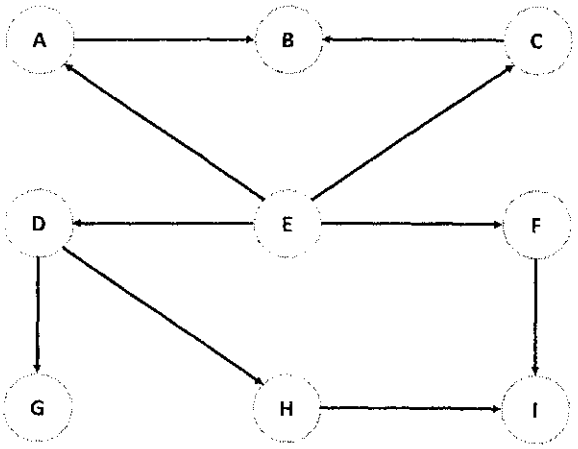
(b) *(10 pts)* Let G be a DAG. Is it possible when exploring an edge (u, v) during $DepthFirstSearch(G)$ we have that $u.discovery = 4, v.discovery = 3, u.finish = v.finish = \infty$? If so provide a graph for which this happens, otherwise explain why it is impossible. Recall that if a node has a finishing time of ∞, that means $DepthFirstSearch(G)$ is still exploring that node.

Question 4 continued on next page

Question 4 continued

(c) *(5 pts)* Consider the following directed graph. Each node in our graph represents a task that needs to be done. Here a directed edge (u, v), that is a directed edge from u to v, means that task v requires that task u be completed prior to beginning it. For the following graph, give a sequence of the nodes which is a valid ordering to complete the tasks in. If no such ordering exists, explain why.

Figure 2: The graph G



6. *(15 pts)* Shortest Paths. For this question, recall that Dijkstra's algorithm computes the lowest cost path to each node in a weighted graph from a special source node s.
(a) *(5 pts)* Consider a directed graph with negative weighted edges. Does Dijkstra's still find the optimal path to each node? If so, briefly justify why. If not, provide a graph where it finds non optimal paths.

(b) *(10 pts)* Consider a directed graph with negative weighted edges, but only coming out of the source. That is, if s is the source node there can be some edges (s, v) where $w_{(s,v)} < 0$. Does Dijkstra's still find the optimal path from s to each node? If so, briefly justify why. If not, provide a graph where it finds non optimal paths.

7. *(10 pts)* Consider the following method to determine if two n bit binary integers $x = x_1x_2...x_n$ and $y = y_1y_2...y_n$ are the same. Starting from $i = 1$, the method checks if $x_i = y_i$, if they are different the procedure stops (concluding x and y are different), otherwise it continues onwards to $i + 1$. If the procedure makes it to $i = n$ and determines $x_n = y_n$ it stops and determines that indeed x and y are equal. Define the running time of the procedure to be the final value of i.

Assuming each possible value of x and y are equally likely, what is the expected running time of the procedure. Provide an exact value, do not simplify it.

8. *(20 pts)* For this question, you are given two Binary Search Trees with integer keys, T_1 and T_2, where the number of nodes in each tree is $|T_1| = n_1$ and $|T_2| = n_2$. All keys are unique, that is no key appears in both trees. Show how to make a new Binary Search Tree T_3 which contains, and only contains, all the nodes of T_1 and T_2. Your technique must run in $O(n_1 + n_2)$ time. Briefly justify why your technique is correct and meets the required runtime.
Hint: T_3 can be any height, although if you are especially clever you can make it perfectly height balanced.

9. *(20 pts)* Consider the following game that proceeds in rounds. Given an undirected graph G, each node is marked *water* or *land*. You begin standing on a special *land* node called the *start* node. You can move from any *land* node to any other adjacent *land* node (that is there is an edge between the two *land* nodes). The goal is to get to a special *land* node called the *finish* node.
At the start of the game there is no valid path (consisting only of *land* nodes) from the *start* to *finish* node. In each round one of the *water* nodes becomes a *land* node (the reverse, *land* becomes *water* never happens). You do not control which node changes in each round, instead, given a pointer to the changed node, you must efficiently determine if the change has created a valid *start* to *finish* path.
To do this you will use a **disjoint-set** data structure. Assume the $Union(x_1, x_2)$ operation never takes more than T_U time, the $MakeSet(x)$ operation never takes more than T_M time and the $FindSet(x)$ operation never takes more than T_F time.
G is given in the adjacency list format and each node is labeled with *water* or *land*. A pair of coordinates indicating the start and finish node are also given.

(a) (5 pts) Before the initial round how will you populate your **disjoint-set**? What are the elements of your set? What is stored in the separate sets? How long will this take in the worst case?

(b) (10 pts) After a node is changed how will you determine if there is now a path from the *start* node to *finish* node. How long will this take?

Question 9 continued on next page

Question 9 continued

(c) (5 pts) Briefly justify why your method is correct.

10. *(30 pts)* For this question, you must augment an *AVL* tree. We are given an array $A = [a_1, \ldots, a_n]$ of n unique integers. You must return an array $B = [b_1, \ldots, b_n]$, where b_i is the sum of all elements in A which are greater than a_i and occur after a_i. For example if $A = [7, 8, 11, 2]$ we would return $B = [19, 11, 0, 0]$. $b_1 = 19$ since both 8 and 11 are larger than 7 and occur later on in A, while $b_4 = 0$ since any element larger than 2 occurs before it in A. Your technique must run in time $O(n \log(n))$. You may reuse whatever you have seen in course the or assignment, simply quote it and do not reprove its correctness or provide its pseudo-code.

(a) *(5 pts)* What will you store in the *AVL* tree and what is your augmentation?

(b) *(20 pts)* How will you calculate B?

(c) *(5 pts)* Justify why your technique fits the required runtime.

This is the final question on the exam

Rough Work space.

Rough Work space.

Rough Work space.