UNIVERSITY OF TORONTO
FACULTY OF ARTS AND SCIENCE

August 2017 Examinations

CSC263H1Y

Duration: 2 Hours

Aids allowed: one A4-sized aid sheet, handwritten on both sides.

No electronic aids allowed.

First (given) name(s): ______________________________

Last (family) name: ______________________________

Student Number: ______________________________

10 questions, 11 pages (including this cover page and 2 blank pages at the end.) The last blank pages are for rough work only, they will not be marked.
Please bring any discrepancy to the attention of an invigilator.
Total Mark: 65
You need to get **at least 35%** in this exam to pass the course.
Answer all questions. WRITE LEGIBLY!
If you need to make any additional assumptions to answer a question, be sure to state those assumptions in your test booklet.
Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, use the empty page on the back.

Question:	1	2	3	4	5	6	7	8	9	10	Total
Points:	15	3	6	6	6	5	8	5	6	5	65
Score:											

1. (15 points) *SHORT ANSWER QUESTIONS*

(a) **What is the best asymptotic characterization (upper bound) of the following function:** $f(n) = 25 + 5n^3 \log n + 26n^2 + 100n^4$ $f(n) = O(\quad\quad)$

(b) **Given a hash table** T **with 25 slots that stores 1000 elements, the load factor** α **for** T **is:** ______________________

(c) **What is the main advantage of the adjacency list representation over the adjacency matrix representation of a graph?**

____

(d) **Depth-first search can be used to classify the edges of a directed graph to four types (tree, back, forward and cross). An <u>undirected</u> graph has <u>only</u>** ______________________ **and** ______________________ **type edges.**

(e) **What are the appropriate data structures** (Stack, Queue, Priority Queue, Union Find) **for the following algorithms?(a data structure can be used more than once.)**

- Breadth First Search:
- Depth-first Search:
- Prim's Minimum Spanning Tree:
- Kruskal's Minimum Spanning Tree:

(f) **The recurrence for worst-case of QuickSort is** $T(n) =$
and the time complexity in worst-case is $T(n) = O(\quad\quad)$
and the expected worst-case for the randomized version is $T(n) = O(\quad\quad)$.

(g) **Depth-first search on a directed graph always produces the same number of tree edges.** ○ *TRUE* ○ *FALSE*

(h) **After running Kruskal's algorithm on a single graph, we always get the same set of edges.** ○ *TRUE* ○ *FALSE*

(i) **An undirected graph** G **has** n **nodes. Its adjacency matrix is given by an** $n * n$ **square matrix whose diagonal elements are 0 and non-diagonal elements are 1. which one of the following is true?**
○ Graph G has no minimum spanning tree (MST).
○ Graph G has a unique MST of cost $n - 1$
○ Graph G has multiple distinct MSTs, each MST has cost of $n - 1$.
○ Graph G has multiple spanning trees of different costs.

(j) **Let T be an AVL tree of height 3. What is the smallest number of entries it can store? Note that a tree with one node (only the root) has the height of zero and stores one element.**

2. (3 points) What is the minimum number of comparisons required to find the minimum value in a *MAX-HEAP* with n elements? Justify your answer.

3. (6 points) **Let D be a data structure whose operations have an amortized cost of at most t.**

   (a) Can you give a guaranteed bound on the cost of a sequence of m operations on D? If so, state this bound. Justify your answer.

   (b) Can you give a guaranteed bound on the cost of an individual operation on D? If so, state this bound. Justify your answer.

   (c) If t is not the amortized but the expected cost of operations on D, can you give any of the above guarantees? If so, which ones. Justify your answer.

4. (6 points) Given an integer array with n elements, write an algorithm that finds the minimum k elements where k and n are two integers $k < n$. You need to give the most efficient algorithm for this question. Analyze the time complexity of your algorithm.

5. (6 points) **Consider a hash table of size 7 storing entries with integer keys. Suppose the hash function is $h(k) = (2k + 3) \mod 7$.**

   - Insert, in the given order, entries with keys $0, 1, 4, 7, 8, 14$ into the hash table using chaining to resolve collisions.

0	1	2	3	4	5	6

   - Insert, in the given order, entries with keys $0, 1, 4, 7, 8, 14$ into the hash table using linear probing to resolve collisions.

0	1	2	3	4	5	6

6. (5 points) The following algorithm uses a randomized strategy to search for a value x in an unsorted array A consisting of n elements.
   The strategy: Choose a random index i into a given array A. If $A[i] = x$, then we return i; otherwise, we continue the search by picking a new random index into A. We continue picking random indices into A until we find an index j such that $A[j] = x$ or until we have checked every element of A. Note that we pick from the whole set of indices each time, so that we may examine a given element more than once. This strategy is implemented in Algorithm 1.

Algorithm 1 *RANDOMIZED SEARCH*(A, n, x)

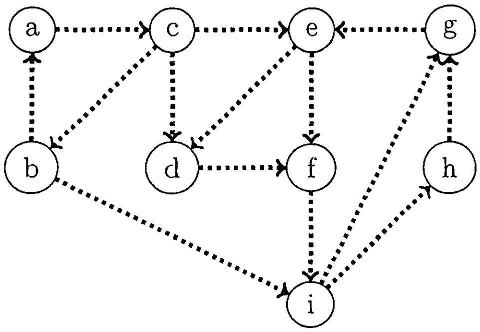
```
Initialize count = 0 and i = 0
Initialize a new array Flag with n elements of 0
while count ≠ n do
  Uniformly choose i ∈ {1, ..., n}.
  if (A[i] == x) then
    return i
  end if
  if (Flag[i] == 0) then
    Flag[i] = 1
    count = count + 1
  end if
end while
return "A does not contain x"
```

Assume there is exactly one index i such that $A[i] = x$. What is the expected number of indices into A that we must pick before we find x and *RANDOMIZED SEARCH* terminates?

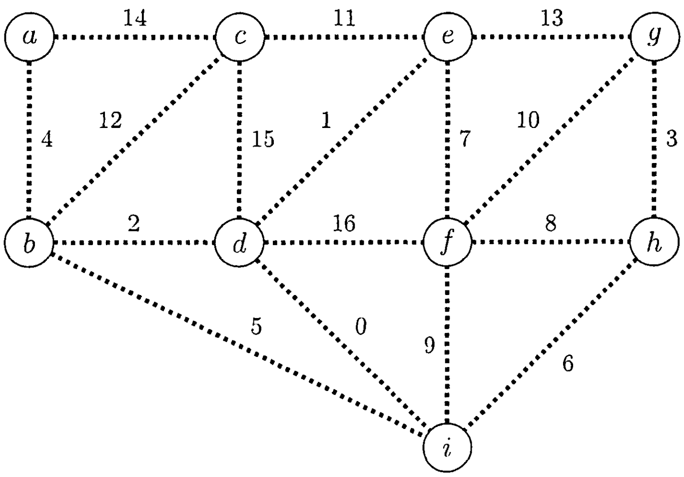
Remark: Recall that: $\sum_{k=0}^{n} x^k = \frac{1-x^{n+1}}{1-x}$, $\sum_{k\geq 1} kx^{k-1} = \frac{1}{(1-x)^2}$

7. (8 points) (a) **Breadth-First Search:** Execute the BFS algorithm on the following directed graph starting from vertex a. Assume that the vertices are listed in increasing order in the adjacency list of the graph: For example, for vertex c, the list of vertices are b, d, e (in this order) in the graph's adjacency list.

- Thicken the edges that belong to the resulting *BFS tree* on the graph.
- What is the *distance* of the vertex h from vertex a computed by this BFS?



(b) **Kruskal's Algorithm.** Execute Kruskal's algorithm on the following graph. The number associated with each edge is its weight.



List the edges of the MST in the order they are found by the algorithm.

8. (5 points) A group of friends are waiting in line at the movie theater.

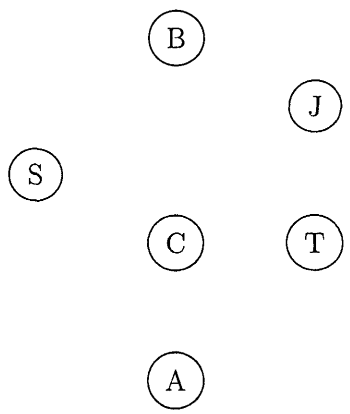
   Joe buys his ticket before **Bob**.
   Bob buys his ticket before **Cathy**.
   Cathy buys her ticket after **Sarah**.
   Sarah buys her ticket before **Joe**.
   Alexa buys her ticket before **Cathy** but after **Sarah**.
   Thomas buys her ticket after **Alexa**.

   In what order do the 6 friends buy their tickets? Which well-known algorithm in the course would give you the order?

   What is the run time complexity of this algorithm?

   Draw a directed graph using the following nodes that shows the initials of the name (e.g. S stands for Sarah). Run the algorithm on the graph and show in which order they buy their ticket.
   You do not need to write the psudocode for the algorithm, show the application the algorithm on your graph and list the names in the order that your algorithm suggests.

   Representing graph:



9. (6 points) (a) Let T be an AVL tree with distinct keys, and k be the key of a *leaf* of T. Note that a search for k partitions the sets of keys of T into three disjoint sets:

- the set $Left$ of keys *to the left of* the search path,
- the set Mid of keys *on* the search path,
- the set $Right$ of keys *to the right of* the search path.

claim: "For every $a \in Left$, $b \in Mid$, and $c \in Right$, $a \leq b \leq c$."

Is this claim correct? If it is, briefly argue why it is correct. If it is not, give a small counter-example.

(b) Let T be an AVL tree and k be a key that is not in T. Prove or disprove:

> If we insert k into T and then delete it (using the AVL insert and delete algorithms), then T after the operations is *exactly* the same as T before the operations.

10. (5 points) Let $G = (V, E)$ be a connected, undirected graph with edge-weight function $w : E \to \mathbb{R}$, and assume all edge weights are distinct. Consider a cycle $v_1, v_2, ..., v_k, v_{k+1}$ in G, where $v_{k+1} = v_1$, and let (v_i, v_{i+1}) be the edge in the cycle with the largest edge weight. Prove that (v_i, v_{i+1}) does not belong to the minimum spanning tree T of G.

SCRATCH PAPER **Your name:**

Use this page only as a scratch paper. This page will be removed and anything on this page will **NOT** be graded.

SCRATCH PAPER **Your name:**

Use this page only as a scratch paper. This page will be removed and anything on this page will **NOT** be graded.