PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

DECEMBER 2014 EXAMINATIONS

CSC 263 H1 F

Duration — 3 hours

Examination Aids: One double-sided handwritten 8.5"×11" aid sheet.

Student Number: |_|_|_|_|_|_|_|_|_|_|

Last (Family) Name(s): ______________________

First (Given) Name(s): ______________________

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

This final examination consists of 8 questions on 17 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy is complete and fill in the identification section above.*

Answer each question directly on the examination paper, in the space provided, and use one of the "blank" pages for rough work. If you need more space for one of your solutions, use a "blank" page and *indicate clearly the part of your work that should be marked.*

In your answers, you may use without proof any theorem or result covered in lectures, tutorials, problem sets, assignments, or the textbook, as long as you give a clear statement of the result(s)/theorem(s) you are using. You must justify all other facts required for your solutions.

Write up your solutions carefully! In particular, use notation and terminology correctly and explain what you are trying to do—part marks *will* be given for showing that you know the general structure of an answer, even if your solution is incomplete.

If you are unable to answer a question (or part of a question), you will get 10% of the marks for any solution that you leave *entirely blank* (or where you cross off everything you wrote to make it clear that it should not be marked).

Remember that, in order to pass the course, you must achieve a grade of at least 40% on this final examination.

MARKING GUIDE

Nº 1: ______/12

Nº 2: ______/10

Nº 3: ______/ 5

Nº 4: ______/ 6

Nº 5: ______/ 7

Nº 6: ______/ 5

Nº 7: ______/10

Nº 8: ______/10

BONUS: ______/ 5

TOTAL: ______/65

Good Luck!

Question 1. [12 MARKS]

Part (a) [2 MARKS]

TRUE or FALSE: In a *max*-heap with n *distinct* elements, the index of the *minimum* element is always $> \lfloor n/2 \rfloor$? Briefly justify your answer.

Part (b) [2 MARKS]

TRUE or FALSE: In a *max*-heap with n *distinct* elements, the index of the *median* element is always $> \lfloor n/2 \rfloor$? Briefly justify your answer.

Part (c) [2 MARKS]

TRUE or FALSE: When we analyse the average case complexity of an algorithm, we should always use a *uniform* probability distribution over our sample space? Briefly justify your answer.

Part (d) [2 MARKS]

TRUE or FALSE: In an AVL tree, the left and right subtrees of any node always contain the same number of elements, plus or minus one? Briefly justify your answer.

Question 1. (CONTINUED)

Part (e) [4 MARKS]

Recall that when Breadth-First Search is executed on an undirected graph G starting from a vertex s, it assigns a value $d[v]$ to every vertex reachable from s, equal to the length of a shortest path from s to v in G. Prove that for every edge $\{u,v\}$ in G, $d[u]-1 \le d[v] \le d[u]+1$.

Question 2. [10 MARKS]

Consider the following algorithms that find the largest and second largest elements in a list.

```
TwoMax1(A):   # A is a list
    S ← −∞   # second largest element
    L ← −∞   # largest element
    for i ← 1,...,n:   # n = len(A)
        if A[i] > L:
            S ← L
            L ← A[i]
        else:
            if A[i] > S:
                S ← A[i]
    return (L,S)
```

```
TwoMax2(A):   # A is a list
    S ← −∞   # second largest element
    L ← −∞   # largest element
    for i ← 1,...,n:   # n = len(A)
        if A[i] > S:
            if A[i] > L:
                S ← L
                L ← A[i]
            else:
                S ← A[i]
    return (L,S)
```

Part (a) [2 MARKS]

We want to analyze the average case complexity of both algorithms, by counting the number of element comparisons performed (the statements "$A[i] > L$" and "$A[i] > S$"). Define an appropriate sample space and probability distribution for this problem.

Part (b) [4 MARKS]

Derive the expected number of element comparisons performed by algorithm TwoMax1.
HINT: Use an indicator random variable X_i with value 1 iff $A[i] > \max\{A[1],\ldots,A[i-1]\}$.

Question 2. (CONTINUED)

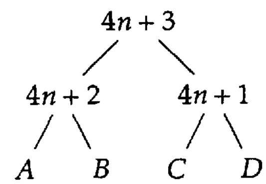
Part (c) [4 MARKS]

Derive the expected number of element comparisons performed by algorithm TwoMAX2.
HINT: Use a slightly different indicator random variable than in the previous part; define it precisely.

Question 3. [5 MARKS]

Part (a) [3 MARKS]

Let $n = 2^k - 1$ for some $k \geqslant 1$ and consider the max-heap illustrated on the right, where A contains elements $\{1, \ldots, n\}$, B contains elements $\{n+1, \ldots, 2n\}$, C contains elements $\{2n+1, \ldots, 3n\}$, and D contains elements $\{3n+1, \ldots, 4n\}$. Argue that when EXTRACT-MAX is performed on this heap, it executes a constant number of comparisons between heap elements.



Part (b) [2 MARKS]

Draw a max-heap of size 10 so that performing EXTRACT-MAX on your heap executes exactly four comparisons between heap elements.

Question 4. [6 MARKS]

Write an algorithm that takes as inputs:

- a connected, undirected graph $G = (V, E)$ with positive integer edge weights $w(e)$, for all $e \in E$,
- a minimum spanning tree $T \subseteq E$ for G, and
- a single edge $e_0 \in T$,

and that constructs a minimum spanning tree T_0 for the graph $G_0 = (V, E - \{e_0\})$. In other words, we remove edge e_0 from graph G and we want to update T so the result is still a minimum spanning tree.

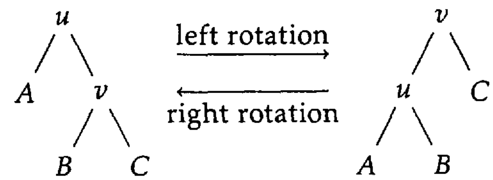
Your algorithm must run in worst-case time $\mathcal{O}(m+n)$ (where $n = |V|$ and $m = |E|$). Write your algorithm in high-level pseudocode, then briefly explain why it is correct and runs within the required time bound. HINT: Start with T and make only a few "small" changes.

Question 5. [7 MARKS]

Suppose an AVL tree is augmented so that each node x contains a pointer $Lr[x]$ to the leftmost node in its right subtree. If the right subtree of x is empty, then $Lr[x] = $ NIL.

Part (a) [1 MARK]

Explain how to update $Lr[v]$ in constant time when a left rotation is performed on u and its right child v or when a right rotation is performed on v and its left child u, as illustrated on the right.



Part (b) [2 MARKS]

Explain how to update $Lr[u]$ in constant time when a left rotation is performed on u and its right child v or when a right rotation is performed on v and its left child u, as illustrated above.

Question 5. (CONTINUED)

Part (c) [4 MARKS]

Explain how to maintain Lr in every node during an INSERT(x) operation, without affecting the $\mathcal{O}(\log n)$ worst-case runtime of INSERT.

Question 6. [5 MARKS]

Consider a party with n guests. There is one *celebrity* at the party, a guest who is known by all other guests, but who does not know any other guest.

A reporter, who is not a guest, needs to determine which guest is the celebrity. The reporter is allowed to ask any guest A whether or not they know guest B.

Part (a) [1 MARK]

If guest A says that they know guest B, what does this tell you about the celebrity status of A or B?

Part (b) [1 MARK]

If guest A says that they do not know guest B, what does this tell you about the celebrity status of A or B?

Part (c) [3 MARKS]

Prove that, in the worst case, the reporter must ask at least $n-1$ questions to determine the celebrity.

Bonus. [5 MARKS]

WARNING! This question is difficult and will be marked harshly: credit will be given only for making *significant* progress toward a correct answer. Please attempt this only *after* you have completed the rest of the final examination.

For the data structure described in Question 5, explain how to maintain Lr in every node during a DELETE(x) operation, without affecting the $\mathcal{O}(\log n)$ worst-case runtime of DELETE.

Question 7. [10 MARKS]

A *multi-set* is like a set where repeated elements matter. For example, $\{1,8,3\}$ and $\{3,3,8,1,8\}$ represent the same *set* but different *multi-sets*.

Consider the abstract data type that consists of a multi-set of integers S and supports the following operations:

- INSERT(S,x): insert integer x into multi-set S;
- DEL-TOP-HALF(S): delete the $\lceil |S|/2 \rceil$ largest integers in S and return them.
  (For example, if $S = \{3,3,8,1,8\}$, then after DEL-TOP-HALF(S) is performed, $S = \{3,1\}$.)

Design a data structure for this abstract data type so that, starting from an empty set, each operation performs an amortized constant number of comparisons between set elements. Use the accounting method to justify that the amortized performance of your data structure is constant. In particular, state and prove an explicit credit invariant.

In your answer, assume you have a deterministic algorithm MED that returns the median of any list of n integers using at most $5n$ comparisons (where the median is the $\lceil n/2 \rceil$-th largest integer in the list).

Question 7. (CONTINUED)

Question 8. [10 MARKS]

Consider the abstract data type ROW that represents the pixels on one row of a black and white screen, which is L pixels wide. An object of ROW is a subset $W \subseteq \{1,\ldots,L\}$ denoting the set of white pixels.

A *line* is a maximal subset of consecutive integers in W. For example, if $W = \{3,4,5,11,12,13,14,17\}$, then the row contains three lines: $\{3,4,5\}$, $\{11,12,13,14\}$, and $\{17\}$.

This ADT supports two operations:

- WHITE(x): Make pixel $x \in \{1,\ldots,L\}$ white, in other words, add x to W if it is not already in W.
- ENDPOINTS(x): Return the endpoints of the line containing x. If $x \notin W$, then return $(0,0)$.

Let n denote the number of white pixels in the row, i.e., $n = |W|$.
Give a data structure that implements ROW such that:

- the space complexity is $\mathcal{O}(L)$,
- the time complexity of WHITE is $\mathcal{O}(\log n)$
- the time complexity of ENDPOINTS is $\mathcal{O}(\log n)$, and
- the worst case complexity of any sequence of m operations is $\mathcal{O}(m \log^* n)$.

Extra credit will be given for a solution in which the time complexity of WHITE is $\mathcal{O}(1)$.
Briefly justify why your data structure is correct and has the required complexity.

Question 8. (CONTINUED)

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Total Marks = 65