UNIVERSITY OF TORONTO
Faculty of Arts and Science

April 2014 Examinations

CSC 263H1S

Duration: 3 hours

No aids allowed.

- **Check that your exam book has 19 pages** (including this cover page and 5 blank pages at the end of this book). Please bring any discrepancy to the attention of an invigilator.
- Answer all questions, and write your answers in the spaces provided in this booklet.
- For rough work, use the backs of the pages, or the last 5 pages; ***these will not be marked.***
- This exam will be graded according to the following criteria:
  1. correctness and completeness,
  2. clarity, precision and conciseness.
- Course policy reminder: a mark of at least 40% on the final exam is necessary to pass the course.

Good luck!

Last (Family) Name	
First (Given) Name	
Student Number	

Problem	Marks Received	Marks Worth
1.		15
2.		18
3.		16
4.		12
5.		10
6.		16
7.		14
8.		16
9.		18
10.		15
11.		20
TOTAL		170

QUESTION 1. (15 marks)

The array shown below stores a MIN heap. The X'ed entries of the array do not contain heap elements.

a. (5 marks) Show the array that results after applying an ExtractMin operation to this heap.

0	4	1	5	8	2	3	6	7	☒	☒	☒	☒

b. (5 marks) Show the array that results after applying an Insert(3) operation to this heap.

1	4	2	6	10	9	5	8	7	☒	☒	☒	☒

c. (5 marks) You are given the following array A of 8 integers: [8,3,5,12,9,11,19,14]. Apply the Build-Max-Heap() algorithm, which was described in the textbook and tutorial, to the array A and *show the resulting array*. Do **not** show any intermediate result.

Double-check your answers above carefully by solving these questions twice!

QUESTION 2. (18 marks)

In the following, we consider three basic data structures to store elements with distinct integer keys, and several operations that we may want to do. For each data structure, and each operation, write in the table below the worst-case time complexity of executing the operation on the data structure. Choose your answer from this list: $\Theta(1)$, $\Theta(\log n)$, $\Theta(n)$, $\Theta(n \log n)$, where n is the number of elements stored in the data structure.

Do *not* justify your answer.

Operation	Search(k)	Find-Min	Find-Max	Delete(x)	Successor(x)	Merge(D, D')
MIN-HEAP						
BINOMIAL MIN-HEAP						
AVL TREE						

In the table above, k is a key and x is a pointer to an element in the data structure.

Search(k): returns the element with key k (without removing it).

Find-Min: returns the element with minimum key (without removing it).

Find-Max: returns the element with maximum key (without removing it).

Delete(x): remove the element pointed at by x.

Successor(x): returns the element with the smallest key that is greater than the key of the element pointed at by x (returns $\perp$, if no such element exists).

Merge(D, D'): merges any two instances D and D' of the data structure into a single one. Here n is maximum of the sizes of D and D'.

QUESTION 3. (16 marks)

Consider the Randomized Quicksort algorithm that we described in class. Suppose that we execute this algorithm on input array $[5, 11, 7, 8]$. In the following questions, do **not** justify your answers.

a. (4 marks) What is the probability that elements 5 and 8 are compared to each other?

b. (3 marks) What is the probability that elements 11 and 8 are *not* compared to each other?

c. (5 marks) What is the *expected* total number of comparisons (between the elements of the array)? Your answer should be an exact number.

d. (4 marks) Now suppose we execute the algorithm on input array $[0, 1, 2, 3]$. What is the *expected* total number of comparisons (between the elements of the array)? Your answer should be an exact number.

QUESTION 4. (12 marks)

In this question, we consider executing a sequence σ containing $n-1$ Unions and $m \geq n$ Finds (starting from singleton elements $1, 2, \ldots n$) in the Linked List and Forest implementations of the disjoint sets ADT that we saw in class and tutorials. In the table below, give the worst-case time to execute: (1) a Find operation, (2) a Union operation, and (3) the entire sequence σ of operations, for each disjoint sets implementation indicated in the table. Your answers in the table should be *good asymptotic upper-bounds* on the *worst-case* execution times written in terms of n and m only (use the O notation).

Note: WU denotes the Weighted Union (by size) rule and PC is the Path Compression rule.

Implementation	Find(x)	Union(r_1,r_2)	sequence σ
Linked List with WU			
Forest no heuristics			
Forest with both PC and WU			

In the above, x is a (pointer to a) set element, and r_1 and r_2 are (pointers to) set representatives.

QUESTION 5. (10 marks)

We want to store a set S of distinct positive integers and support the following operations on S:

Insert(a): insert integer $a \notin S$ into S.

Delete(a): delete integer $a \in S$ from S.

FindNext(a, k), where $a \in S$ and k is an integer: if S consists of some elements $a_1 < a_2 < \cdots < a_n$, and $a = a_i$ (for some i, $1 \leq i \leq n$), then return a_{i+k} (for simplicity, assume that $i + k \leq n$).

The worst-case time complexity of each operation should be $O(\log n)$ where n is the size of S.

An augmented data structure that we learned in class can be used (without any modification) "as a black box" to achieve the above goal: ***using the operations that this data structure provides***, the algorithms for performing the above three operations are quite simple.

Give an algorithm for the FindNext(a, k) operation ***that uses only the operations of this data structure and consists of at most 5 lines of pseudo-code***.

Do ***not*** justify your answer, and do ***not*** describe the implementation of the data structure that you use.

QUESTION 6. (16 marks)

In this question, we are interested in the amortised complexity of incrementing by one a ***ternary*** counter $C[0..7]$ that contains numbers represented in ***base 3*** notation. Instead of using bits (0 and 1) as in base 2 notation, the counter uses ***trits***, 0, 1 and 2, to represent these numbers.

We define the ***time*** required to increment by one the 8-trits counter $C[0..7]$ to be the number of trits in C that change when we increment C by one. For example, the time to increment $C = 12201222$ by one is exactly 4, because incrementing C by one gives 12202000, so 4 trits changed; the time to increment $C = 00002222$ by one is exactly 5, because incrementing C by one gives 00010000, so 5 trits changed; and the time to increment $C = 22222222$ by one is exactly 8, because incrementing C by one gives 00000000, so 8 trits changed (note that when the 8-trits counter C "overflows" it goes back to 00000000).

Let $A(n)$ be the ***amortised*** time per increment operation when we perform a sequence of n successive increments by one, starting from the counter $C = 00000000$.

a. (4 marks) What is $A(6)$? Your answer should be an exact number. Do not justify your answer.

b. (12 marks) Give a good upper bound on $A(n)$. More precisely:

Consider the list $\frac{4}{3}, \frac{17}{12}, \frac{3}{2}, \frac{7}{4}, 2, \frac{5}{2}$ (note that the numbers in this list are sorted from smallest to largest). What is the *smallest* number c in this list such that $A(n) \leq c$ *for all* $n \geq 1$?

Prove the correctness of your answer. Make sure you justify why your answer is the *smallest* such c from the list.

QUESTION 7. (14 marks)

Consider a Depth-First Search (DFS) of a directed graph G. Let $d[z]$ and $f[z]$ denote the discovery and finishing times of a node z, as computed by this DFS of G, and let (u, v) be an edge in G.

a. (5 marks) Suppose the DFS classifies (u, v) as a *back edge*.

- What is the color of v *just before* edge (u, v) is explored? Circle the correct answer:
  - (a) WHITE
  - (b) GREY
  - (c) BLACK
  - (d) it depends on G and the DFS
- List $d[u]$, $f[u]$, $d[v]$ and $f[v]$ in *increasing order*:

b. (4 marks) Suppose the DFS classifies (u, v) as a *forward edge*.

What is the color of v *just before* edge (u, v) is explored? Circle the correct answer:

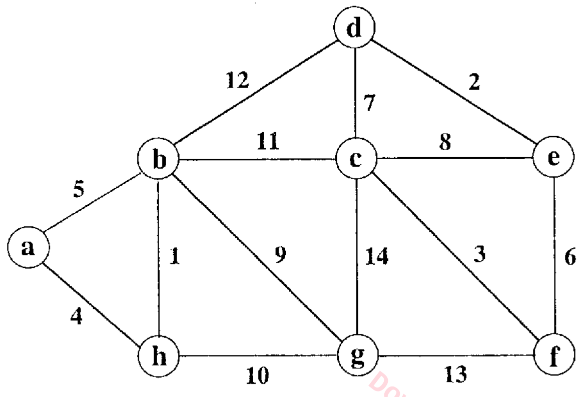
- (a) WHITE
- (b) GREY
- (c) BLACK
- (d) it depends on G and the DFS

c. (5 marks) Suppose the DFS classifies (u, v) as a *cross edge*.

- What is the color of v *just before* edge (u, v) is explored? Circle the correct answer:
  - (a) WHITE
  - (b) GREY
  - (c) BLACK
  - (d) it depends on G and the DFS
- List $d[u]$, $f[u]$, $d[v]$ and $f[v]$ in *increasing order*:

QUESTION 8. (16 marks)

This question is about the Minimum Spanning Tree (MST) of the following weighted undirected graph G (G has 8 nodes, a, b, c, d, e, f, g, h, and 14 edges, weighted from 1 to 14):



a. (8 marks) Perform Kruskal's MST Algorithm on G, and list the edges of the MST found by the algorithm, *in the order in which the algorithm finds them.* Give *only* this sequence of edges (and nothing else).

b. (8 marks) Perform Prim's MST Algorithm on G starting **from vertex** c, and list the edges of the MST found by the algorithm, *in the order in which the algorithm finds them.* Give *only* this sequence of edges (and nothing else).

QUESTION 9. (18 marks)

If you do not know how to solve this question, and you write "I do not know how to solve this", you will receive 20% of the marks of this question. If you just leave it blank, you get 0.

Describe an algorithm that, given any unsorted array $A[1..n]$ of n positive integers, determines whether the array contains duplicates (i.e., whether some integer appears more than once in the array). The ***expected*** running time of your algorithm should be $O(n)$ under some assumptions discussed in class.

Note: Do ***not*** assume that the numbers in the array are small, or they have a small range, e.g., they could be arbitrary integers. So a solution based on a direct access table or on linear time sorting is ***not*** acceptable.

a. (12 marks) Describe your algorithm clearly and concisely, and give the pseudo-code.

b. (4 marks) Explain why the ***expected*** running time of your algorithm is $O(n)$. Explicitly state every assumption that you need for your complexity analysis.

[question continued on the next page...]

c. (2 marks) What is the ***worst-case*** running time of your algorithm? Use the Θ notation and justify your answer.

QUESTION 10. (15 marks)

If you do not know how to solve this question, and you write "I do not know how to solve this", you will receive 20% of the marks of this question. If you just leave it blank, you get 0.

In the following, B_1 and B_2 are two binary search trees such that every key in B_1 is smaller than every key in B_2.

Describe an algorithm that, given pointers b_1 and b_2 to the roots of B_1 and B_2, merges B_1 and B_2 into a single binary search tree T. Your algorithm should satisfy the following two properties:

1. Its worst–case running time is $O(min\{h_1, h_2\})$, where h_1 and h_2 are the heights of B_1 and B_2.
2. The height of the merged tree T is at most $max\{h_1, h_2\} + 1$.

Note that the heights h_1 and h_2 are ***not*** given to the algorithm (in other words, the algorithm does not "know" the heights of B_1 and B_2). Note also that B_1, B_2 and T are ***not*** required to be balanced.

Describe your algorithm in ***clear and concise*** English (no pseudo-code), and ***briefly*** justify its correctness and worst-case running time.

NOTE: Partial credit may be given for an algorithm that runs in $O(max\{h_1, h_2\})$ time.

QUESTION 11. (20 marks)

If you do not know how to solve this question, and you write "I do not know how to solve this", you will receive 20% of the marks of this question. If you just leave it blank, you get 0.

Let I_n be the set of n integers $\{1, 2, \ldots, n\}$ where n is some power of 2.

Note that we can easily use an n-bit vector (i.e., an array of n bits) $B[1..n]$ to maintain a subset S of I_n and perform the following three operations (where j is any integer in I_n) in constant time each:

INSERT(j): insert integer j into S.

DELETE(j): delete integer j from S.

MEMBER(j): return **true** if $j \in S$, otherwise return **false**.

Describe a data structure that supports all the above operations ***and*** also the following operation

MAXIMUM: return the greatest integer in S

such that:

- The worst-case time complexity of operations INSERT(j), DELETE(j), and MAXIMUM is $O(\log n)$ each. The worst-case time complexity of MEMBER(j) is $O(1)$.
- The data structure uses only $O(n)$ bits of storage.

  Note that the binary representation of an integer i where $1 \leq i \leq n$ takes $\Theta(\log n)$ bits. Assume that any pointer also takes $\Theta(\log n)$ bits.

A solution that does not meet ***all*** the above requirements may not get any credit.

HINT: Combine an n-bit vector (i.e., an array of n bits) with a complete binary tree, and avoid using pointers.

a. Describe your data structure by drawing it for $n = 8$ and $S = \{1, 2, 6\}$, and by explaining this drawing briefly and clearly. Your drawing must be very clear.

[question continued on the next page...]

b. Explain how the operations INSERT(j) and MAXIMUM are executed in $O(\log n)$ time in the worst-case. Be brief and precise.

c. Explain how the operation MEMBER(j) is executed in $O(1)$ time in the worst-case. Be brief and precise.

THE END

[Page for rough work only — not graded]

[more space, if needed, available on the next page]

[Page for rough work only — not graded]

[more space, if needed, available on the next page]

[Page for rough work only — not graded]

[more space, if needed, available on the next page]

[Page for rough work only — not graded]

[more space, if needed, available on the next page]

[Last page for rough work only — not graded]