University of Waterloo
Department of Electrical and Computer Engineering
ECE250
Algorithms and Data Structures
Fall 2014

Midterm Examination

Instructor: Ladan Tahvildari, PhD, PEng, SMIEEE
Date: Tuesday, October 21, 2014, 10:00 a.m.
Location: EIT-1015; CPH-3602; CPH-3604
Duration: 80 minutes
Type: Closed Book

Instructions:

- There are 4 questions. Answer all 4 questions.
- Standard calculator allowed but no additional materials allowed.
- The number in brackets denotes the relative weight of the question (out of 100).
- If information appears to be missing from a question, make a reasonable assumption, state it and proceed.
- Write your answers directly on the sheets.
- If the space to answer a question is not sufficient, use overflow page.
- When presenting programs, you may use any mixture of pseudocode/C++ constructs as long as the meaning is clear.

Name	Student ID

Question	Mark	Max	Marker
1	A: B: C:	35	A: B: C:
2		10	
3	A: B: C:	35	A: B: C:
4	A: B:	20	A: B:
Total		100	

Name: **Student ID:**

Question 1: Algorithm Analysis [35]

Part A [15].

Consider the following recursive algorithm that computes minimum value out of real numbers stored in the array A on positions from l to r.

FindMin(A: array of real numbers, l: integer, r: integer)

if $l = r$ **then return** *A[l]*

$temp_1 \leftarrow FindMin(A, l, \left\lfloor \frac{l+r}{2} \right\rfloor)$

$temp_2 \leftarrow FindMin(A, \left\lfloor \frac{l+r}{2} + 1 \right\rfloor, r)$

if $temp_1 \prec temp_2$

then return $temp_1$

else return $temp_2$

Write a recurrence describing the cost of running this algorithm on n element array: *FindMin(A,1,n)*. Solve the recurrence and give the resulting running time of the algorithm.

Part B. [10]

Give asymptotic upper and lower bounds for the following recurrence. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answer.

$$T(n) = 16T(n/4) + n^5\sqrt{n}$$

Name: **Student ID:**

Part C. [10]

Prove that $f(n) = 10^7 + 7n^7 \log n + 3n^7$ is $\mathrm{O}(n^7 \log n)$ using the definition of "Big-Oh".

Question 2: Elementary Data Structures [10]

You have been provided with an implementation of a *stack-of-integers* ADT. This implementation includes the usual operations such as ***Push***, ***Pop***, and ***IsEmpty***.

Describe how you would implement the Queue ADT using two stacks. Specifically, give algorithms for the ***Enqueue*** and ***Dequeue*** operations. Give tight Big-Oh expressions for the running times of your implementation.

You may assume that ***Push***, ***Pop***, and ***IsEmpty*** are all $O(1)$.

Name: **Student ID:**

Question 3: Trees and Tree Traversals [35]

Part A. [15]

- Consider we have a BST contains the following keys: P, L, A, C, E.
  Draw this BST in such a way that the result of the "preorder tree traversal" is: C, A, L, E, P.

- Let T be an AVL tree of height 6. What is the smallest number of nodes it can store?

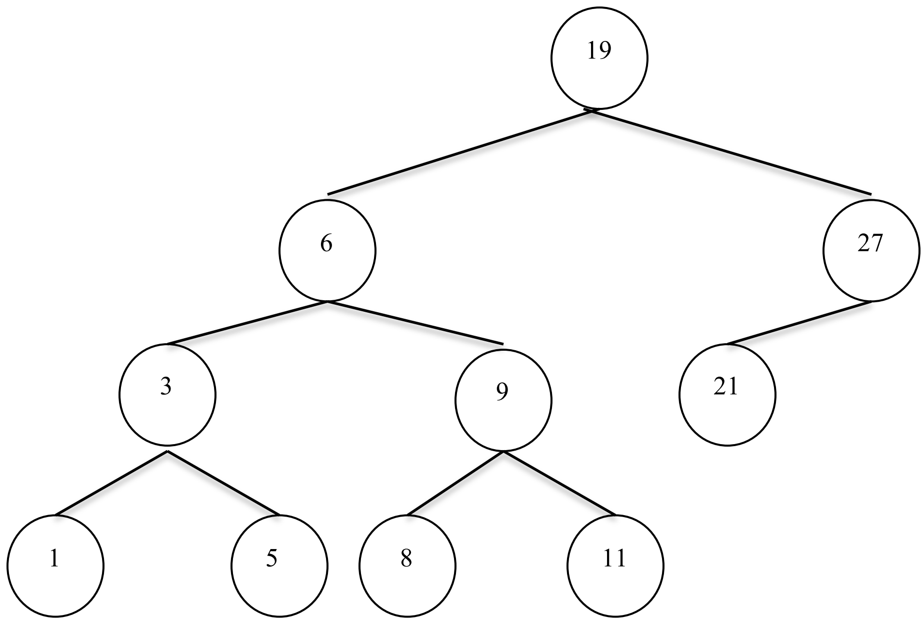
Part B. [12]

Start with an empty AVL tree and insert the following keys in the given order: 25, 35, 45, 20, 22, and 27. Draw the trees following each insertion, and also after each rotation. Specify the rotation types.

Name: **Student ID:**

Part C. [8]

Assume we have the following AVL tree. Show the tree after deleting 27. Use the successor of a node if it is needed. Draw the final tree and all intermediate trees.



Question 4: Hashing [20]

Part A. [5]

Suppose we have a hash table with N slots containing n keys. Suppose that instead of a linked list, each slot is implemented as a binary search tree. Give the worst and the best time complexity of adding an entry to this hash table. Explain your answers.

Part B. [15]

Consider a hash table of size 7. Suppose the hash function uses division method. Insert, in the given order, keys: 2, 4, 12, 19, and 20 into the hash table using:

- Linear probing to resolve the collisions. Show all your work.

- Double hashing to resolve collisions with the secondary hash function $h_2(k) = 5 - (k \mod 5)$. Show all your work.

Name: **Student ID:**

OVERFLOW SHEET [Identify the question(s) being answered.]