University of Waterloo
Department of Electrical and Computer Engineering
ECE250
Algorithms and Data Structures
Fall 2017

Midterm Examination

Instructor: Dr. Ladan Tahvildari, PEng, SMIEEE
Date: Wednesday, October 25, 2017, 8:45 p.m.
Location: RCH-301 and RCH-302
Duration: 75 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 writing an algorithm, 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:	30	A: B:
2		15	
3	A: B: C:	35	A: B: C:
4		20	
Total		100	

Question 1: Algorithm Analysis [30]

Part A [15].

Remember the *MERGE* procedure discussed in the lecture. Briefly speaking a call to $MERGE(A, p, q, r)$ merges two sorted sub-arrays $A[p..q]$ and $A[q+1..r]$ into a sorted sub-array $A[p..r]$, using linear time. We use this procedure in the following sorting algorithm:

$SORT(A : array, p : \text{int}, r : \text{int})$

    if $p \geq r$

        then return

    $m \leftarrow \left\lfloor \frac{r-p-1}{3} \right\rfloor$

    $SORT(A, p, p+m)$

    $SORT(A, p+m+1, r-m-1)$

    $SORT(A, r-m, r)$

    $MERGE(A, p, p+m, r-m-1)$

    $MERGE(A, p, r-m-1, r)$

Write a recurrence for the worst-case running time of the *SORT* algorithm and find a tight asymptotic bound on the worst-case running time.

Part B. [15]

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.

$$T(n) = 3T(\frac{n}{4}) + n^2 \lg n$$

Question 2: Elementary Data Structures [15]

Propose a stack data structure that supports the following three operations:

a) **Push (Stack s, x):** This operation adds an item x to stack s
b) **Pop (Stack s):** This operation removes the top item from the stack s
c) **Concatenate (Stack s1, Stack s2):** This operation combines two stacks; the content of stack s2 goes on the top of stack s1

Time complexity of all above operations should be $O(1)$.

Question 3: Trees and Tree Traversals [35]

Part A. [10]

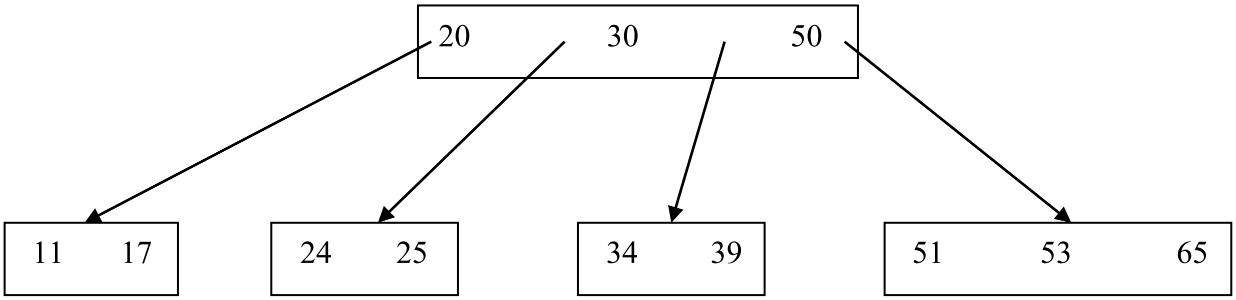
Start with an empty AVL tree and insert the following keys in the given order: 45, 55, 65, 40, 42, and 47. Draw the trees following each insertion, and also after each rotation. **Specify the rotation types.**

Part B. [12]

Let B_1 and B_2 be two binary search trees (BST) that together store keys $k_1, k_2, \ldots, k_n$. Suppose we know that every key in B_1 is smaller than every key in B_2 (according to some comparison operator). Write an algorithm that merges B_1 and B_2 into a single BST in $O(\min\{h_1, h_2\})$ time, where h_1 and h_2 are the heights of B_1 and B_2, respectively. Pointers b_1 and b_2 to the roots of each BST are given.

Part C. [13]

Consider the following B-Tree of order t=2.



Draw the tree after inserting 60. **Show all your work.**

Question 4: Hashing [20]

Consider a hash table of size 11. Suppose the hash function uses division method. Insert, in the given order, keys: 10, 22, 31, 4, 15, 28, 17, 88 and 59 into the hash table. Double hashing technique with the secondary hash function $h_2(k) = 1 + \left(k \bmod (m-1)\right)$ is used to resolve collisions. **Show all your work.**

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