Birla Institute of Technology & Science - Pilani, Hyderabad Campus
Second Semester 2013-2014
CS F211 / IS F211 / C363 / IS C363: Data Structures and Algorithms
Comprehensive Examination

Type: Part Open **Time: 180 mins** **Max Marks: 100** **Date: 10.05.2014**

Read following instructions carefully:

1. Question paper contains Part A (closed book) and Part B (open book).
2. Two answer sheets will be provided to answer part A and Part B.
3. Write name of the part (Part A / Part B) on the main page of the corresponding answer book.
4. Answers to Part A questions should not be written in answer book of Part B and vice versa.
5. **You may take text / reference books to answer Part B questions only after submitting Part A answer sheet.**
6. You may submit Part A answer book at any time during 3 Hours.

Part A (Closed Book)

1.a. . For each of the following algorithms, indicate their worst-case running time complexity using the Big-Oh notation, and give a brief (3-4 sentences each) summary of the worst-case running time analysis.

1. Preorder traversal of a binary tree of size 3n assuming each visit action takes constant time.
2. Quick-sort on a sequence of size n, assuming that the pivot is always the last element in the corresponding sequence.
3. Insertion into a red-black tree of size n. [6 Marks]

1.b Describe in pseudo-code a linear-time algorithm for reversing a queue Q. To access the queue, you are only allowed to use the methods of queue ADT. Hint: Consider using an auxiliary data structure. [3 Marks]

1.c. Insert into an initially empty binary search tree items with the following keys (in this order): 30, 40, 23, 58, 48, 26, 11, 13. Draw the tree after each insertion. [3 Marks]

1.d. Remove keys 32, 65, 76, 88, 97 (in the specified order) from the binary search tree, that is obtained by solving the above problem. Draw the tree after each removal. [3 Marks]

2.a. Suppose that each row of an n * n array A consists of 1's and 0's such that, in any row of A, all the 1's come before any 0's in that row. Assuming A is already in memory, describe a method running in O(n) time (not $O(n^2)$ time) for finding the row of A that contains the most 1's. [6 Marks]

2.b. An array A contains n integers taken from the interval [0, 4n], with repetitions allowed. Describe an efficient algorithm for determining an integer value k that occurs the most often in A. What is the running time of your algorithm? [4 Marks]

2.c. Solve the following recurrence relations using Master's theorem: [2 + 2 Marks]

i. $T(n) = 4T(n/2) + n^2 \log n$

ii. $T(n) = 8T(n/2) + n \log n$

2.d. Could a binary search tree be built using o(n lg n) comparisons in the comparison model? Explain why or why not. Note that o in o(n lg n) refer to small o. [4 Marks]

3.a. ‘Under the simple uniform hashing assumption, the probability that three specific data elements (say 1, 2 and 3) hash to the same slot (i.e., h(1) = h(2) = h(3)) is $1/m^3$, where m is a number of buckets’ - do you agree with the above statement? If yes, provide justification otherwise find out the probability. [3 Marks]

3.b. In a hash table in which collisions are resolved by chaining, a successful search takes average-case time $\Theta(1 + \alpha)$, under the assumption of simple uniform hashing. α is defined to load factor as n/m where the number of elements is n and number of slots is m. [6 Marks]

3.c. Prove that A red-black tree with n internal nodes has height at most 2 lg(n+1). [4 Marks]

3d. Consider the modified binary search algorithm so that it splits the input not into two sets of almost-equal sizes, but into three sets of sizes approximately one-third. Write down the recurrence for this ternary search algorithm and the asymptotic complexity of this algorithm. [5 Marks]

4.a. Provide an algorithm of $O(n^{1.585})$ complexity to multiply two (large) *n*-digit integers represented by arrays of their digits with the constraint that Single Digit Multiplications are only allowed. [6 Marks]

4.b. Consider two positively weighted graphs G = (V,E,w) and G^1 = (V,E,w^1) with the same vertices V and edges E such that, for any edge e € E, we have w(e) = $w(e)^2$. For any two vertices u, v € V, any shortest path between u and v in G^1 is also a shortest path in G. [3 Marks]

4.c. Devise an algorithm by using branch and bound technique to select items from the following table that maximizes benefit /value with the following constraints: [6 Marks]

1. The sum of weights of selected items should be less than or equal to 10

2. Items are indivisible, you either take an item or not.

Item	Weight	Value	Value / weight
1	4	$40	10
2	7	$42	6
3	5	$25	5
4	3	$12	4

Part B (Open Book)

5.a. Consider the Change Problem in Austria. The input to this problem is an integer L. The output should be the minimum cardinality collection of coins required to make L shillings of change (that is, you want to use as few coins as possible). In Austria the coins are worth 1, 5, 10, 20, 25, 50 Shillings. Assume that you have an unlimited number of coins of each type. Formally prove or disprove that the greedy algorithm (that takes as many coins as possible from the highest denominations) correctly solves the Change Problem. So for example, to make change for 234 Shillings the greedy algorithms would take four 50 shilling coins, one 25 shilling coin, one 5 shilling coin, and four 1 shilling coins. [4 Marks]

5.b. Consider the Change Problem in Binaryland The input to this problem is an integer L. The output should be the minimum cardinality collection of coins required to make L nibbles of change (that is, you want to use as few coins as possible). In Binaryland the coins are worth 1, 2, 2^2, 2^3, . . . , 2^{1000} nibbles. Assume that you have an unlimited number of coins of each type. Prove or disprove that the greedy algorithm (that takes as many coins of the highest value as possible) solves the change problem in Binaryland. [6 Marks]

5.c. In dynamic programming, we derive a recurrence relation for the solution to one subproblem in terms of solutions to other subproblems. To turn this relation into a bottomup dynamic programming algorithm, we need an order to fill in the solution cells in a table, such that all needed subproblems are solved before solving a subproblem. For each of the following relations, give such a valid traversal order, or if no traversal order is possible for the given relation, briefly justify why. [2 + 3 + 3 Marks]

i. $A(i,j) = F(A(i,j-1), A(i-1,j-1), A(i-1,j+1))$
ii. $A(i,j) = F(A(\min\{i,j\}-1, \min\{i,j\}-1), A(\max\{i,j\}-1, \max\{i,j\}-1))$
iii. $A(i,j) = F(A(i-2,j-2), A(i+2,j+2))$

6.a. Consider a connected weighted directed graph G = (V, E, w). Define the fatness of a path P to be the maximum weight of any edge in P. Give an efficient algorithm that, given such a graph and two vertices u, v € V, finds the minimum possible fatness of a path from u to v in G. The algorithm should be of complexity O((|E| + |V|) log|V|) or O((|E| + |V|) log|E|)). [8 Marks]

6.b. Woody the woodcutter will cut a given log of wood, at any place you choose, for a price equal to the length of the given log. Suppose you have a log of length L, marked to be cut in n different locations labeled 1, 2, . . ., n. For simplicity, let indices 0 and n + 1 denote the left and right endpoints of the original log of length L. Let d_i denote the distance of mark i from the left end of the log, and assume that $0 = d_0 < d_1 < d_2 < \ldots < d_n < d_{n+1} = L$. The wood-cutting problem is the problem of determining the sequence of cuts to the log that will cut the log at all the marked places and minimize your total payment. Give an efficient algorithm to solve this problem. [8 Marks]