CS21003: Algorithms - 1
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur
Midsem Examination, Autumn 2015

Total Marks: 100 **Time: 2 Hours**

Answer ALL Questions. Answers should be brief and to the point.

1. **(a)** Given two non-negative functions f(n) and g(n), when can we say that f(n) = θ(g(n))? (4)
(b) Define the load factor of a hash table. Typically, what should be the maximum load factor if you use hashing (no explanation is needed)? (3 + 2)
(c) Name two sorting algorithms that are both stable and in-place (no explanation is needed). Why do you think a stable sorting algorithm can be useful (other than radix sort)? (2 + 4)

2. **(a)** Define a Red-Black tree. (5)
(b) Suppose the numbers 6, 19, 17, 11, 3, 12, 8, 20, 22, 23, 13, 18, 14, 16, 1, 2, 24, 25, 4, 26, 5 are inserted one by one (in this order) into an empty B-tree of minimum degree 2. Show only **(i) the trees just before and just after each splitting of a node, and (ii) the final tree after all insertions**. No other intermediate tree needs to be shown and no explanation is needed. (10)
(c) Consider a hash table of size 23. Open addressing with linear probing is used to hash elements in the table, with the hash function used for a key k being h(k) = k % 23. The following numbers are inserted into the hash table in order: 33, 21, 58, 80, 56, 68, 43, 66, 35, 23. For each insertion, just show the index at which the value is finally inserted and the number of probes needed. Do not draw the whole table at each step and no explanation is needed. (10)

3. Given two arrays X and Y containing real numbers, and another real number k, design an algorithm to find if there exists a pair of elements, one from X and one from Y, whose sum is equal to k. Your algorithm should run in O($n$lg$n$) time, where n is the sum of the number of elements in X and the number of elements in Y. Justify the time complexity of your algorithm in 1-2 sentences. (12)

4. Consider two sets of n elements each stored in two n-element arrays A and B. You are also given an empty array C of size $2n$. Design an O($n$lg$n$) time algorithm to compute the union of the sets A and B and store it in C, with the size of the union set stored in a variable m. Other than these three arrays, you can use only O(1) additional space. Write the pseudocode. (12)

5. Design an ADT that supports the following three operations: (i) insert an element if it is not already there, (ii) delete an element if it is there, and (iii) find the k-th smallest element. All the operations should be done in O(lg$n$) time, where n is the number of elements in the ADT. (20)

6. Consider an array A of n arbitrary integers x_1, x_2, ...,x_n. Consider another array B of n distinct integers a_1, a_2, ..., a_n from 1 to n (i.e., a_1, a_2, ...,a_n is a permutation of 1,2,...n). Design an O($n$lg$n$) time algorithm to order the integers in A according to the order imposed by the permutation in B (i.e., for each k, x_k should finally be placed in A[a_k]). For example, if A = 17, 5, 1, 9 and B = 3, 2, 4, 1, then the A array should finally contain 9, 5, 17, 1. You can use only O(1) additional space. List the steps clearly, do not write pseudocode. (16)