CS3 Mid Semester (max marks 40; max time 2hrs)

Indian Statistical Institute, Bangalore

September 14,2016

Q1. (1x10=10) True or False?

(a) The maximum height of a heap on an array of n elements is $\Theta(\log(n))$

(b) The maximum time to search for an element in a heap on an array of n elements is $\Theta(\log(n))$

(c) The maximum height of a binary search tree of n elements is $\Theta(\log(n))$

(d) The maximum time to search for an element in a binary search tree of n elements is $\Theta(\log(n))$

(e) For a full binary search tree with n elements ("full" means level i has 2^i elements), the time to find the maximum element in the tree is $\log(n)$.

(f) For a full min-heap with n elements on an array ("full" as defined above), the time to find the maximum is $\log(n)$.

(g) If the mergesort algorithm was given a pre-sorted array of n elements, it would still take $\Theta(\log(n))$ time.

(h) Given an array of n elements, in a modified mergesort algorithm, we dont recursively split the sub-array of k elements into two equal parts, we just ensure that the smaller of the two parts has size at least $k/3$. This modified mergesort has worst case complexity $\Theta(n^2)$

(i) If the mergesort algorithm on n elements was modified so that when the sub-array reaches size 8, the algorithm does not split it further, but sorts it by using bubble sort, then the complexity of such a modified mergesort is $\Theta(n \log(n))$.

(j) Given an n element array, if we first build a binary search tree of the given elements, and then did an inorder traversal to print them, we would get an algorithm that sorts in time $\Theta(n^2)$

Q2. (2x5=10)

(a) Given n elements in an array, what is the time to build a max-heap on the array?

(b) Given a min-heap on an array, give an algorithm (pseudo-code) to insert a new element into the array.

(c) What is the reason that quicksort is not $\Theta(n \log(n))$?

(d) Given a set of elements and a relation on them, if one wanted to list the elements in order of distance from a given element(without repetition), what data structure would one use to hold the element during the search?

(e) What data structure does one implicitly use for search when one does a recursive in-order traversal of a binary search tree?

Q3. (3+4+3=10) Give the pseudo code for an efficient algorithm for each of the following functions (assume no repeated values, and assume each node also has a pointer to its parent node):

(a) FindMax(T): Given the pointer T to the root of a BST, it finds the node in T with the maximum value.

(b) FindNext(T): Given the pointer T to the root of a BST, it finds the node in T which has the successor value to the value in the root. If no such one exists, it returns NULL.

(c) MergeTrees($T1, T2$): Given the pointers $T1$ and $T2$ to the roots of two binary search trees such that all elements in $T1$ are smaller than all elements in $T2$, it merges the two to create a new tree. This function uses the above functions if needed.

Q4. (4+2+2+2=10) Your are tasked to design a program that is constantly receiving a sequence of words, with the purpose of telling at any point of time, how many times it has seen a given word in the input sequence. Do this using hashing.

(a) Describe a data structure for the same. Describe what the hash function does, but don't design a hash function. Can collision occur? How do you handle collision? Show how the data structure may look after inserting the sequence "Good habits make good life".

(b) Write the pseudo code for a function called insert(s) which inserts a string s into the hash table, or increments the count if it already exists. What is the complexity?

(c) Write a function showcount(s) which returns the number of times the string s has appeared in the input stream. What is the complexity?

(d) Write a function showstrings() to print all the strings seen in the input stream upto that point in time and the number of times they were seen. What is its complexity?