Design and Analysis of Algorithms

Homework # 3

Q1) [10+5+5 = 20 Marks]

(a) In Data Structures, you studied binary heaps. Binary heaps support the insert and extractMin
functions in $O(lgn)$, and getMin in $O(1)$. Moreover, you can build a heap of n elements in just $O(n)$. Refresh your knowledge of heaps from chapter no. 6 of your algorithms text book.
Now implement Merge Sort, Heap Sort, and Quick Sort in C++ and perform the following experiment:
1. Generate an Array A of **10^8** random numbers. Make its copies B and C. Sort A using Merge Sort, B using Heap Sort, and C using Quick Sort.
2. During the sorting process, count the total number of comparisons between array elements made by each algorithm. You may do this by using a global less-than-or-equal-to
function to compare numbers, which increments a count variable each time it is called.
3. Repeat this process 5 times to compute the average number of comparisons made by each algorithm.
4. Present these average counts in a table. These counts give you an indication of how the different algorithms compare asymptotically (in big-O terms) for a large value of n.

(b) Now compare the same algorithms in terms of practical time, i.e. the actual running time. Simply, repeat the previous example but use the chrono library to compute the actual times taken by each algorithm, and report the average value of the time for each algorithm.

(c) The sorting algorithm available in the C++ STL is called IntroSort. Here is what Wikipedia says about it: *"Introsort or introspective sort is a hybrid sorting algorithm that provides both fast average performance and (asymptotically) optimal worst-case performance. It begins with quicksort, it switches to heapsort when the recursion depth exceeds a level based on (the logarithm of) the number of elements being sorted and it switches to insertion sort when the number of elements is below some threshold."* In light of your experiments above, explain why IntroSort does what it does. You may take a look at its pseudo-code here

Q2) A singly linked list contains n - 1 strings that are binary representations of numbers from the set $\{0, 1, \ldots, n-1\}$ where n is an exact power of 2. However, the string corresponding to one of the numbers is missing. For example, if $n = 4$, the list will contain any three strings from 00, 01, 10 and 11. Note that the strings in the list may not appear in any specific order. Also note that the length of each string is *lgn*, hence the time to compare two strings in *O*(*lgn*). Write an algorithm that generates the missing string in *O*(*n*). [10 Marks]

Q3) Given a sorted array of n distinct integers A[1] · · · A[n], describe an O(log n) divide-and-conquer algorithm to find out whether there is an index i such that A[i] = i. Why does your algorithm run in the claimed time bound? [10 Marks]

Q4) You are given an infinite array A in which the first n cells contain integers in sorted order and the rest of the cells are filled with ∞. You are not given the value of n. Describe an O(log n) algorithm that takes an integer x as input and finds a position in the array containing x, if such a position exists. [10 Marks]

Q5) Given an unsorted array of integers, *A*, its size *n*, and two numbers *x* and *y* both elements of *A*, write an algorithm that returns the *distance* between *x* and *y*. The distance between two numbers of an array is the number of elements that lie between them in the sorted order. Achieve the asymptotically fastest time for this problem. [10 Marks]