Data Structures and Algorithms : CS210A/CS210A

Practice-sheet 6

Date: *31st October, 2014*

1 Median finding, Binary heap, average running time of quick sort

1. Describe an $O(n)$ time algorithm that given a set S of n distinct numbers and a positive number $k \leq n$, determines the k numbers in S that are closest to the median of S.

2. Thoroughly understand the proof for the average running time of quick sort. In particular, you must fully internalize the way the recurrence was formulated for $T(n)$ and solved using induction. What can you say about the average running time for insertion sort and selection sort ?

3. Recall the *QuickSelect* algorithm discussed during the lecture of $O(n)$ time median finding algorithm. This algorithm was derived from *QuickSort*. Show that average running time of *QuickSelect* is $O(n)$.

4. Recall the algorithm for 2-majority element. Suppose elements in the given set are numbers. Extend this algorithm to compute k-majority element that uses $O(nk)$ time. Try to improve it to $O(n \log k)$.

5. There are ℓ sets $S_1, S_2, \ldots, S_\ell$ storing distinct numbers. These sets are sorted. The total number of elements in $\cup_i S_i$ is n. Design an $O(n \log k)$ time algorithm to output elements of $\cup_i S_i$ in sorted order.

6. What can be the worst case time for searching in a binary heap of size n?

7. Derive an alternate algorithm for finding k-majority element using median finding algorithm. What will be its time complexity ?

2 Greedy Algorithms

1. A security company needs to obtain licenses for n different pieces of a cryptographic software. Due to regulations, they can only obtain these licenses at the rate of at most one per month.

   Each license is currently selling for a price of \$100. However, they are all becoming more expensive according to exponential growth curves: in particular, the cost of license j increases by a factor of $r_j > 1$ each month, where r_j is a given parameter. This means that if license j is purchased t months from now, it will cost $\$100 r_j^t$. We will assume that all the price growth rates are distinct, that is $r_i \neq r_j$ for licenses $i \neq j$ even though they start at the same price of \$100. The question is the following:

Given that the company can only buy at most one license a month, in which order should it buy the licenses so that the total amount of money it spends is as small as possible ?

Give an algorithm that takes the n rates of price growth $r_1, r_2, \cdots, r_n$ and computes an order in which to buy the licenses so that the total amount of money spent is minimized. The running time of the algorithm should be polynomial in n.

2. Sequence A is said to be subsequence of another sequence B if there is a way to delete zero or more elements from A so that the resulting sequence turns out to be B. For example $\langle a, b, a, a, c, b, d\rangle$ is a subsequence of $\langle c, a, a, b, a, b, a, b, c, a, c, b, b, d\rangle$. Let S and S' be two sequences of characters. Let m and n be respectively the lengths of sequences S and S'. You may assume that S and S' are given in the form of arrays. Getting inspiration from greedy approach, design an $O(m + n)$ time algorithm to detect whether S is a subsequence of S'. Prove correctness of the algorithm as well.

3. You are working with a group of security consultants who are helping to monitor a large computer system. There is a particular interest in keeping track of processes that are labeled "sensitive". Each such process has a designated start time and finish time, and it runs continuously between these times; the consultants have a list of the planned start and finish times of all sensitive processes that will run that day.

   As a simple first step, they have written a program called `status_check` that, when invoked, runs for a few seconds and records various pieces of logging information about all sensitive processes running on the system at that moment. (We'll model each invocation of `status_check` as lasting for only this single point of time.) What they'd like to do is to run `status_check` as few times as possible during the day, but enough that for each sensitive process P, `status_check` is invoked at least once during the execution of process P.

   Give an efficient algorithm that, given the start and finish times of all the sensitive processes, finds as small a set of times as possible at which to invoke `status_check`, subject to the requirement that `status_check` is invoked at least once during each sensitive process P.

4. The manager of a large student union on campus comes to you with the following problem. She is in charge of a group of n students, each of whom is scheduled to work one shift during the week. There are different jobs associated with these shifts (tending the main desk, helping with package delivery, rebooting cranky information kiosks, etc.), but we can view each shift as a single contiguous interval of time. There can be multiple shifts going on at once.

   She is trying to choose a subset of these n students to form a *supervising committee* that she can meet once a week. She considers such a committee to be complete if, for every student not on the committee, that student's shift overlaps (at least partially) the shift of some student who is on the committee. In this way, each student's performance can be observed by at least one person who is serving on the committee.

   Give an efficient algorithm that takes the schedule of n shifts and produces a complete supervising committee containing as few students as possible. You must also prove the correctness of the algorithm.

3 Minimum Spanning Tree

1. Suppose we are given a connected graph G, with edge costs that are all distinct. Prove that G has a unique minimum spanning tree.

2. There is a connected graph $G = (V, E)$, with edge costs that are all distinct. You are given a minimum spanning tree T for G. Let v be a leaf node in T. Weight of one of the edges incident on v in the graph has decreased. This might lead to update T. You have to determine if T has to be updated, and in case, the MST is updated, you have to compute the new minimum spanning tree in $O(n)$ time only. (Note that there could be $O(n)$ edges incident on v in the original graph.)

3. Given an undirected weighted graph $G = (V, E)$ in the form of Adjacency matrix. Design an $O(n^2)$ time algorithm for computing minimum spanning tree of G.

4. Give complete proof of cut-property and cycle property for MST discussed in the class.

5. Prove or give a counter-example for the following assertions in the context of MST of a given weighted graph.

   - The max-weight edge of a cut can not be in MST ?
   - The minimum weight edge of a cycle has to be in MST ?

6. A spanning tree of a graph is said to be a maximum spanning tree if its weight is maximum among all spanning trees of the graph. Design efficient algorithms for computing maximum spanning tree. How similar are these algorithms to the algorithms of minimum spanning tree ? Are there graph theoretic properties of maximum spanning trees similar to cut-property and cycle-property of minimums spanning tree ?