1. Let A = $\{a_1, a_2, \ldots, a_n\}$be a set of distinct items totally ordered such that $a_i < a_j$ iff $i < j$. Assume that over a long sequence of m accesses to items in A, a_i is accessed $q(i) \geq 1$ times. Describe an algorithm to find a binary search tree T where each item a_i resides in a leaf of T such that if we serve the accesses using T, each time going from the root to the corresponding leaf, the total time it takes to serve the whole sequence is minimized. Prove

1) that your algorithm indeed constructs a tree that minimizes the total access time.

2) As tight upper bound as you can, on the total time it takes to serve the sequence using T.

3) An upper bound on the running time of your algorithm for constructing T. (Any polynomial time algorithm would be fine.)

2. We define the following variation on the splay algorithm. This variation looks 3 steps (edges) towards the root from the node x and applies one of the rules in Figure 1 (or their mirror image) if possible. If it is not possible to apply one of the rules in Figure 1 we apply one of the regular zig-zig, zig-zag, or zig rules (Note that zig or zig-zig would apply only if x is at distance 1 or 2 from the root, respectively).

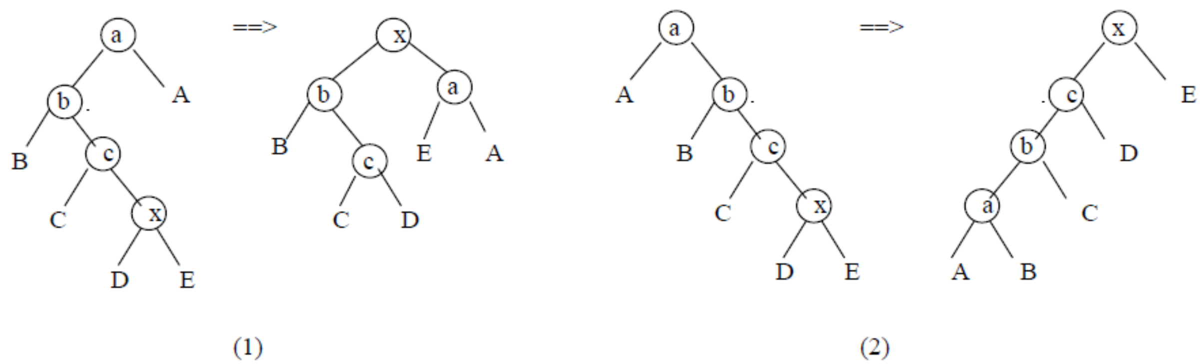

Figure 1: Splay cases in question (7)

Prove that the access lemma holds for this variation as well (with a different constant).

3. Given a string s, $|s| = n$, the suffix array, SA, of s, is a permutation of $\{1, 2, \ldots, n\}$ such that SA[j] = i if and only if the suffix of s starting with the character i, ($i = 1, \ldots$, n) is the jth when we order the suffixes lexicographically. We add a special character \$ to each suffix which is smaller than any other character so that the lexicographic order of the suffixes is well defined.

(a) Given a permutation π of 1, 2, . . . , n, is there always a string s of length n such that π is the suffix array of s? Prove your answer.

(b) Below are three suffix arrays. For each of these suffix arrays find a string s of length n, over the smallest possible alphabet Σ, such that the corresponding array is a suffix array of s. Prove that there is no string s' over a smaller alphabet such that the suffix array is a suffix array of s'.

1)

n	n-1	$\cdots$	2	1

2)

1	2	$\cdots$	n-1	n

3) Assume n is even:

n	1	n-1	2	n-3	3	$\cdots$	$n/2+1$	$n/2$

4. The recurrence for the running time of the algorithm for computing a suffix array presented in class is $T(n) = T(2n/3) + O(n)$. Show how to modify the algorithm to give one whose recurrence is $T(n)=T(3n/7) + O(n)$.

5. A string s of length n is periodic if there is a string u of length $\leq n/2$ such that $s = u^k u'$, where k is an integer ≥ 2, u^k is the concatenation of k copies of u, and u' is a prefix of u. The smallest period of s is the shortest u for which $s = u^k u'$ holds. Suppose you are given a suffix tree of s together with an LCA data structure. Show how to use it to find the smallest period of s or declare that s is not periodic.

6. Consider an implementation of Fibonacci heaps without cascading cuts (all other details are as shown in class, the only difference is that delete and decrease-key just cut the subtree and do not continue with cascading cuts). For any large enough m show a sequence of m operations on heaps of size at most n such that the average cost of an operation is as high as possible. (By m large enough we mean larger even than some function of n.)

7. For any positive integer n, give a sequence of Fibonacci heaps operations that creates a Fibonacci heap consisting of just one tree that is a linear chain of n nodes (make your sequence as short as you can).