

15-451/651 Algorithm Design & Analysis, Fall 2026

Homework #3

Homework #3, Due: Oral presentations September 23th to 25th

This is an oral presentation assignment. There are three problems (#1-#3). You should work in groups of three. The sign-ups will be online soon (details on Ed) and your group should sign up for a 30-minute slot. Each person in the group must be able to present every problem. The TA/Professor will randomly select who presents which regular problem, you may not choose. You may however choose which order to present the problems in.

Since this is an oral assignment, there is of course no page limit, since you will not be submitting written solutions. Keep in mind however that there is a time limit. You must be able to explain the solutions to all three problems within the 30-minute time slot. As usual, for all problems unless stated otherwise, you must explain why your algorithm is correct and provide an analysis of its runtime.

1. (ORAL, 25 pts) **It's (almost) a palindrome**

As you probably know, a palindrome is a string that is the same when reversed. In this problem, a pseudo-palindrome is a string with the property that it is not a palindrome, but when you delete one character from it, it becomes a palindrome. An example is “1001101”, because if you delete the red “0” you get “101101” which is a palindrome. Your goal is to design an algorithm that finds the longest pseudo-palindrome substring of a given binary string of length $n \geq 2$.

If your algorithm is non-deterministic and has a probability of failure, you must ensure that the probability of failure is at most $1/n$. The cost of your algorithm must be better than quadratic and for full points should be the best cost you can achieve using techniques from this class. Justify the cost with respect to the word RAM model used in class.

2. (ORAL, 25 pts) **Move to the top with SegTrees**

You want to implement a data structure supporting the following operations over a set of distinct integer keys $\{0, \dots, n-1\}$. Initially the data structure contains the keys $\{0, \dots, n-1\}$ in sorted order.

- **Find**(x): Return the position of key $x \in \{0, \dots, n-1\}$.
- **Move**(x): Move key x to the first position of the list. The relative order of every other key does not change (i.e., x is removed and inserted at the front, it is not swapped with the first element, etc.).

It is possible to solve this problem using binary search trees, but for the purpose of this problem, your solution must use SegTrees instead (No points will be awarded for a solution that is not based on SegTrees).

Suppose that you know in advance that there will be m **Move** operations ever performed. Your goal is to support the **Find** and **Move** operations in $O(\log(n+m))$ worst-case time using at most $O(n+m)$ space. You can use $O(n+m)$ time for preprocessing before receiving any operations.

3. (ORAL, 25 pts) **Splaying Problems**

- (a) A splay tree is being used to store $n + m$ items; n of them are red and m of them are blue.

Using the standard splay tree potential function (with appropriately chosen weights) prove the following two statements:

- The amortized number of splay steps (as defined in lecture) done when a red item is accessed is at most $4 + 3 \log_2 n$.
- The amortized number of splay steps (as defined in lecture) done when a blue item is accessed is at most $4 + 3 \log_2 m$.

This is curious because if there are very few red items, it seems to become very efficient to access them. However the algorithm itself does not even know which items are red and which are blue! Can you explain this?

- (b) You have a splay tree of 1,000,000 nodes. Each node has a weight of 1. Using these weights, the usual sizes and ranks of all nodes are computed. You splay a node x that is at depth 1000 in the tree. Consider the 500 splay steps that are done. Call a splay step “scary” if the rank of x does not change during the step. What is the minimum number of scary splay steps that can occur?
- (c) Continuing with the scenario in part (b), you correctly reason that the potential ought to decrease. Find the largest value of k you can such that it’s guaranteed that the potential will decrease by at least k .