# Fall 24 Div Compete Week 4 - Solutions

(taken from editorials of original problems)

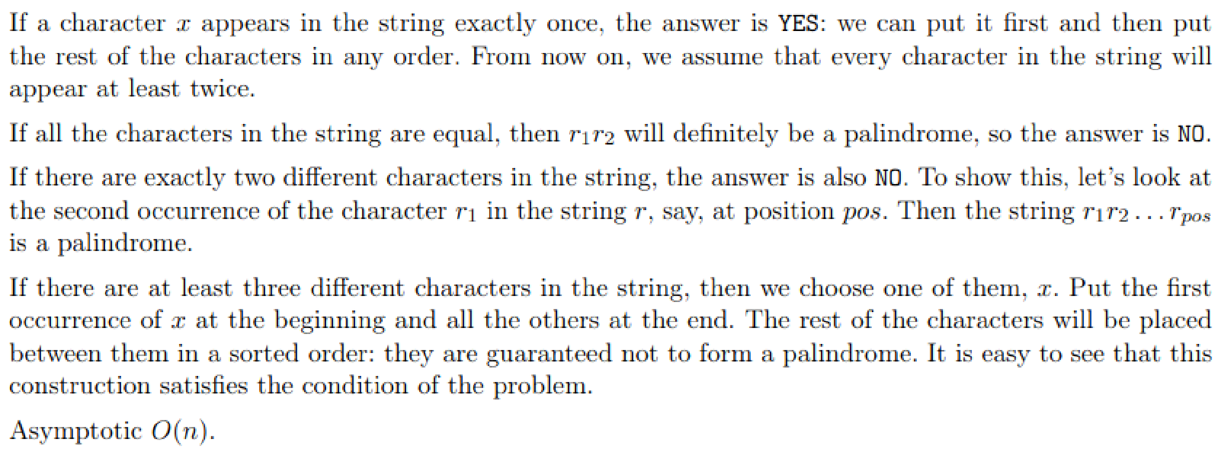
## Problem A

(Source: 2024 ICPC Asia Taiwan Online Programming Contest Problem A)

Just take the pig of largest influence and sum all of the influences of nonpigs with less influence.

## Problem B

(Source: Anton Trygub Contest 2 (The 3rd Universal Cup, Stage 3: Ukraine) Problem A)



If a character $x$ appears in the string exactly once, the answer is `YES`: we can put it first and then put the rest of the characters in any order. From now on, we assume that every character in the string will appear at least twice.

If all the characters in the string are equal, then $r_1 r_2$ will definitely be a palindrome, so the answer is `NO`.

If there are exactly two different characters in the string, the answer is also `NO`. To show this, let's look at the second occurrence of the character $r_1$ in the string $r$, say, at position $pos$. Then the string $r_1 r_2 \ldots r_{pos}$ is a palindrome.

If there are at least three different characters in the string, then we choose one of them, $x$. Put the first occurrence of $x$ at the beginning and all the others at the end. The rest of the characters will be placed between them in a sorted order: they are guaranteed not to form a palindrome. It is easy to see that this construction satisfies the condition of the problem.

Asymptotic $O(n)$.

## Problem C

(Source: 2024 ICPC Asia Taiwan Online Programming Contest Problem B)

You can reduce this to a minimum subarray sum problem. Take the sum of the absolute value of all numbers and call it M. Now, modify the array such that the negative entries are now worth 3x as much as they originally were. Find the subarray of minimum sum in this modified array. Subtract that to the original sum and that's the answer.

## Problem D

(Source: OCPC Potluck Contest 1 (The 3rd Universal Cup. Stage 6: Osijek) Problem K)

This problem is asking to repeatedly build the convex hull of a set of points and erasing one of the corners of the convex hull (points lying on an edge of the convex hull are not allowed). Let's look at the point

which minimizes $(x, y)$ lexicographically. There will not be any points with smaller $x$, and all points with the same $x$ will have bigger $y$. This means that the extreme point in the direction $(-1, -\varepsilon)$, for $\varepsilon$ tiny, will always be this point, and it also will be the unique extreme point. This means this point must be a corner of the convex hull.

So the solution becomes: sort the points in increasing order of the pairs $(x, y)$ lexicographically. Then output the first $n - 1$ points.

## Problem E

(Source: OCPC Potluck Contest 1 (The 3rd Universal Cup. Stage 6: Osijek) Problem H)

We are given a subset of special elements $S$ and need to count the number of permutations $p$ such that for each $i$, at least one of the following holds: (a) $p_i < i$ or (b) $p_i \neq i$ and $p_i$ is special.

Suppose we drop the restriction that $p_i \neq i$ in (b). Then we have a simple procedure that can generate any valid permutation. We build the permutation from left to right and maintain a pool of possible elements that $p_i$ can be legally set to. Initially, the pool consists of the special elements. For each $i$:

- Pick and remove any element from the pool, set $p_i$ to that element.
- If $i$ is not special, add $i$ to the pool.

The number of ways to pick $p_i$ is simply the size of the pool. The pool stays the same size after setting $p_i$ if $i$ is non-special, and the size is decreased by one if $i$ is special. This allows us to count the number of such permutations. If $S$ consists of elements $i_1 < i_2 < \cdots < i_k$, then the number of such permutations is

$$k^{i_1}(k-1)^{i_2-i_1}(k-2)^{i_3-i_2}\cdots 1^{i_k-i_{k-1}}0^{n-i_k}.$$

Now, let's move back to the original problem. We'll try to use the inclusion-exclusion principle to modify this solution for the original problem. For any subset $T \subseteq S$, let $m_T$ be the number of permutations $p$ such that:

- if $i \notin T$, then $p_i < i$ or $p_i$ is special;
- if $i \in T$, then $p_i = i$.

That is, $m_T$ is the number of permutations such that the $p_i \neq i$ rule is explicitly violated for a given set $T$ of special indices (but it may also be violated for other special $i$). The inclusion-exclusion principle states that then, the answer to the problem is

$$\sum_{T \subseteq S} (-1)^{|T|} m_T. \tag{1}$$

Indeed, let's take the answer given by the initial algorithm. This includes permutations where $p_{i_1} = i_1$, so subtract the number of such permutations. Similarly, subtract the number of permutations where $p_{i_2} = i_2$. Now we've doubly subtracted the permutations where both $p_{i_1} = i_1$ and $p_{i_2} = i_2$ hold. Add the number of them back and so on. Eventually you'll arrive at the formula (1).

Computing $(-1)^{|T|} m_T$ is similar to the algorithm above. Start with a pool of size $|S \setminus T|$ and set the initial answer to be $(-1)^{|T|}$. Then

- If $i \in T$, do nothing.
- If $i \in S \setminus T$, multiply the answer by the pool size and reduce the pool size by 1.
- Otherwise, multiply the answer by the pool size.

Calculating the answer for each $T \subseteq S$ is obviously too slow, but we can use dynamic programming to calculate it all "in parallel". Set $\mathrm{dp}[0][j] = 1$ if $j \equiv |S| \pmod 2$ and $-1$ otherwise. Set $\mathrm{dp}[i][j] = 0$ for $i > 0$. Now for each $i$ from $1, \ldots, n$ and each $j$ from $0, \ldots, |S|$:

- If $i \in S$, add $j \cdot \mathrm{dp}[i-1][j]$ to $\mathrm{dp}[i][j-1]$ and $\mathrm{dp}[i-1][j]$ to $\mathrm{dp}[i][j]$.
- Otherwise, add $j \cdot \mathrm{dp}[i-1][j]$ to $\mathrm{dp}[i][j]$.

The answer to the problem will be in $\mathrm{dp}[n][0]$.

## Problem F

(Source: OCPC Potluck Contest 1 (The 3rd Universal Cup. Stage 6: Osijek) Problem G)

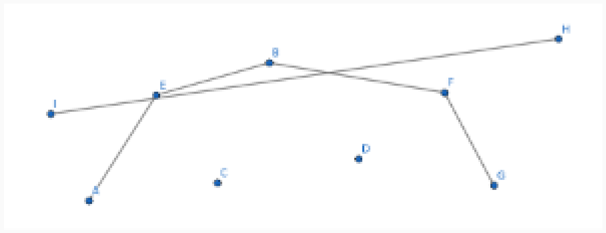
## Statement

Given $n$ points. Answer $m$ queries: given a segment, find a point that "above" this segment. $n \le 8 \cdot 10^5, m \le 3 \cdot 10^5$. Low memory.

Focus on a single query $(x_l, y_l) - (x_r, y_r)$.

Only points with $x_l \le x \le x_r$ can be the answer.

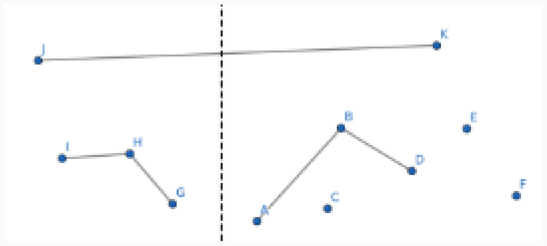
Of those, points not on the convex hull can't be the only answer.



$(x, y)$ is "above the segment" only if $ax + by \ge c$ for some $a, b, c$.

Hull points are in unimodal order w.r.t. this functional: binary search.

Divide and conquer. Split in the middle, handle queries that cross the middle.



Build hull of right side from left to right, at each prefix search for a match for appropriate queries.

How to know which queries to search on?

Build an array of all $x$-coordinates (endpoints of segments and given points).

Sort it and D&C on that.

I.e. when building a hull, if you see a given point, add it, otherwise if the other endpoint is over the split-line, do the search.