Fall 23 Div I Week 1 - Solution Sketches

(taken from editorials of original problems)

Problem A

(Source: Codeforces Round 870 (Div. 2) Problem A)

Let's iterate over the number x of liars in the group. Now everyone, who says $l_i > x$ is a liar, and vice versa. Let's count the actual number of liars. If those numbers match, output the answer.

Now that we've checked all possible x's, we can safely output -1, since no number of liars is possible.

Problem B

(Source: ICPC German Collegiate Programming Contest (GCPC 2023) Problem I)

Solution

- Maintain the current positions of each frog and an ordered set S of currently free positions.
- Now the events can be simulated quickly. For a jump of frog i, currently at position p:
  - find $\min\{p' \in S : p < p'\}$ in $\mathcal{O}(\log |S|)$ using operations from the standard library.
  - update S and the position of frog i
- Total time complexity: $\mathcal{O}(n \log |S|)$

Problem C

(Source: ICPC German Collegiate Programming Contest (GCPC 2023) Problem B)

Solution

- If we have at most k points the answer is obviously Yes.
- If we select $k+1$ points, one line has to go through two of those points.
- Given k and $n > k$ points solve the problem recursively:
  - Select $k+1$ points and try all lines through two points.
  - For each line remove all covered points.
  - Check recursively with $k-1$ and the remaining points.
- Time complexity for $(k = 3)$: $n \cdot \prod_{i=1}^{k} \binom{i+1}{2} = 18 \cdot n$

Another solution (I like it better)

More Observations

- There must be a line which covers at least a third of all points.
- There must be a line which covers at least half of all remaining points.
- There must be a line which covers all remaining points.

Solution 2

- Recursively select a random line through two points.
- At step k check if the chosen line covers $\frac{1}{k}$ of all points.
  Yes: recursively continue with $k-1$ and the remaining points.
  No: try another line or abort after sufficient many tries $(\sim 5 \cdot k)$.
- Time complexity for $(k = 3)$: $n \cdot 5 \cdot k! = 30 \cdot n$

Problem D

(Source: 43rd Petrozavodsk Programming Camp (2022 Summer) Day 7. HSE Koresha Contest Problem F)

Let's consider some integer d and periodic array $\{1, 2, \ldots, d, 1, 2, \ldots, d, \ldots\}$ with period d. For this array for any subsegment of length at most d the answer is negative (all elements are different), for others is positive.

Of course, it is not always possible to select such d, that exactly m segments have length at most d, so let's upgrade this array a bit.

Let's sort all segments by their length (in case of equal length by left bound). Let d be the length of $(m+1)$-st segment, r be it's right bound.

- If $d = 1$, the answer is impossible, because there are $\geq m+1$ segments of length 1, for them the answer will be always negative.
- If $d \geq 2$ the answer always exists. Let's consider an array $\{1, 2, \ldots, d, 1, 2, \ldots, d, \ldots\}$. From the position r let's change the period from d to $d-1$. It means, that $a_i = a_{i-d}$ for $i < r$ and $a_i = a_{i-d+1}$

  for $i \geq r$. For such array the answer for the first m segments in the sorted order will be negative, for others will be positive.

Time complexity: $O(n + m \log m)$.

Problem E

(source: 43rd Petrozavodsk Programming Camp (2022 Summer) Day 7. HSE Koresha Contest Problem I)

Required tricks:

- Binary search by the answer.
- Radewoosh trick (https://codeforces.com/blog/entry/62602).
- dp by angle to find the convex polygon (with fixed lowest point).

Let's fix the lowest point of the sun s and make a binary search. We want to check if there exists a sun with lowest point A_s, such that $S - xP \geq 0$ for given x.

Let's make a standart dp. Let's sort all pairs of points (A_i, A_j), such that the triangle $A_sA_iA_j$ doesn't contain other points. Pairs will be sorted by angle of vector $A_j - A_i$. Let dp_p equal to the maximum score of the path from s to p ($dp_p = -\infty$ initially). After that we iterate pairs (i, j) in the sorted order and update:

$$dp_j = \max\left(dp_j, dp_i + area(A_sA_iA_j) - x|A_iA_j|\right)$$

Pairs (i, j) can be sorted before iteration of s and the binary search once, so the complexity of this check is $O(n^2)$. At the end we should check, that $dp_s \geq 0$.

But this solution does not take into account points outside of the sun. How we could add them? For each point A_p outside of the sun let's consider it's projection to the perimeter of the sun.

- If this projection is on some side A_iA_j, it is easy to add such points into dp. For each pairs of points A_i, A_j let's calculate the sum of $\min\left(|A_iA_p|, |A_jA_p|\right)$, for all p such that the projection of A_p to the line A_iA_j lies on the segment A_iA_j and A_p lies on the right side to the vector from A_i to A_j. After that just change $|A_iA_j|$ to the $|A_iA_j| + sum_{i,j}$ in dp formula.
- This projection is some point A_i. To add this case to our dp let's add events corresponding to it. Let's add a second type of events for every pair (i, p). In the moment $angle(A_p - A_i) + \frac{\pi}{2}$ let's add this event. To update dp for event (i, p) of this type we should make:

$$dp_i -= x|A_iA_p|$$

The time complexity of one check is still $O(n^2)$.

Now let's iterate all s in random order and make a binary search only if the answer can be increased on this step (it can be checked with one check). The total number of checks will be $O(n + \log n \log A)$.

So, the total time complexity is $O(n^3 + n^2 \log n \log A)$.