Problem Analysis (Day 1)
ICPC Camp 2022 powered by Huawei

Problem A: Soccer match

- Let's show that we can obtain a bipartite subgraph G' of the original graph G with at least nk edges.

- First, we assign a color to each vertex of G randomly, white or black with equal probability, thus obtaining an bipartite subgraph.
    - The expected number of edges in this subgraph is $\frac{\mathrm{m}}{2}$, while the variance is $\frac{\mathrm{m}}{4}$。 Using Chebyshev's Inequality, we have a fairly low probability ($\le \frac{1}{k^2}$) of getting less than $\frac{m}{2} - k\sqrt{m}$ edges in the subgraph.

Problem A: Soccer match

- Second, we can adjust the subgraph to make sure it contains at least $\frac{m}{2}$ edges.
  - In each round, If there is a vertex which is connected to more vertices of the same color than vertices of opposite color, we will flip its color. The procedure above will end if it contains at least least $\frac{m}{2}$ edges.
  - The number of edges increases by at least 1 in each round. We can assume with high probability that, we won't flip more than $\sqrt{m}$ times. Thus the time complexity is O($m\sqrt{m}$).

Problem A: Soccer match

- Once we have obtained the bipartite subgraph, we can just repeatedly remove the vertices whose degree is $\leq k$.
  - This procedure will stop before the graph becomes empty : assume that at some moment, only $k+1$ vertices remain in the graph. By induction it should contain at least $k(k+1)$ edges. However, a graph with $k+1$ vertices apparently has no more than $\frac{k(k+1)}{2}$ edges, which leads to a contradiction.

- The time complexity of this part is O(m).

Problem B: Gachapon

- Let f[i][j] be the probability of an i-level rolling is legal, and the best card drown out is an j-star item.

- Let g[i][j] be the probability of an n-level rolling is legal， if we known a specific i-level rolling is legal, and the best drown out from it is an j-star item.
  - initially we have g[n][n]=g[n][n+1]=...=g[n][m]=1
  - the transfer between g[i][j] and g[i-1][k] are same if j!=k, so we can use prefix sum to speed up the transfer.
  - Time complexity : O($nm \log b\_i$).

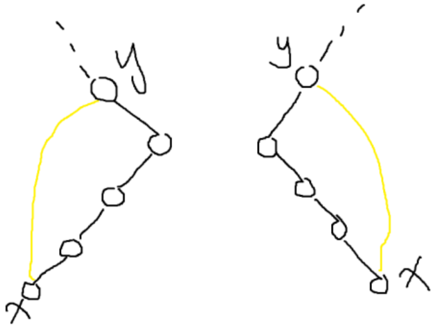
Problem C: Survey

- Due to the Linearity of Expectation, the problem is equivlent to
  - Divide m dollars to n shares.
  - A share of x dollar leads to profit $\sum_i threshold[i] \leq x$.
  - Find the maximum profit.

- Simple DP:
  - let f[i][j] be the maximum profit of dividing i dollars into j shares.
  - We have $f[i][j] = \max_{k=0}^{i/j} f[i-k][j-1] + profit[k]$ because only the smallest share needs to be enumerated.

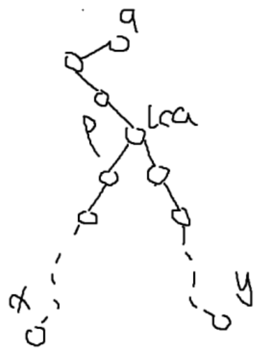
Problem D: Station

- First, we rephrase the problem as follows.
  - Adding directed edges on all (x,y) that satisfy the following conditions:
    - 1. $a[x] \le a[y]$
    - 2. $a[i] < a[x]$ $(\min(x,y) < i < \max(x,y))$
  - And the value of the edges is $l[x]$ if $x<y$, and $r[x]$ otherwise.
  - Ask q times for the length of the shortest path between s and t.
- Let's consider the case where a[i] are distinct.
- The graph also can be described as
  - First, building the Cartesian tree of a.
  - Then add edges similar to the form on the right(marked in yellow):
  - (For convenience, We denote fa2[x]=y.)
- So now let's consider the shortest path between two nodes in the tree.



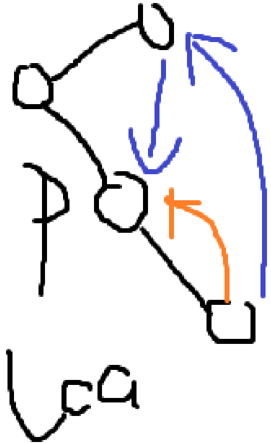
Problem D: Station

- Property 1: the closest node to the root on the shortest path from x to y must be one of the following:
  - 1. L=lca(x,y)
  - 2. the father in the Cartesian tree of lca(x,y) —— p in the picture
  - 3. fa2[lca(x,y)] —— q in the picture
- Proof. Consider the first node and the last on the path from the s to t that its distance to the root is not greater than the lca ’s one. The first one must be one of lca and p, and for the last one, it is lca and q.
- And going from x to y must pass through at least one node whose distance from the root is not greater than lca's.
- So we can only consider the shortest path between lca, p and q (It is the same if we reverse the direction).



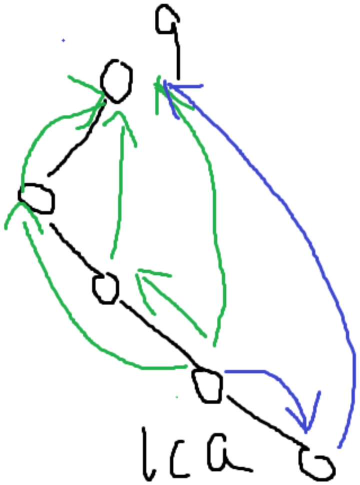
Problem D: Station

- Case 1: lca -> p
- There are two ways (Others are obviously not optimal).
  - One is marked in orange, its cost is l[lca].
  - One is marked in blue, its cost is r[lca]+l[fa2[lca]].
- Because of fa2[lca]>lca, so l[fa2[lca]]≥l[lca].
- So the path colored in blue is not optimal.



Problem D: Station

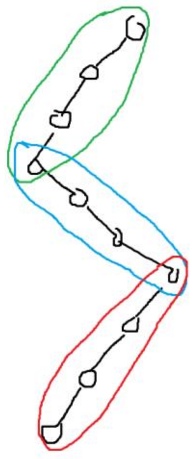
- Case 2: lca -> q
- There are some ways (Others are obviously not optimal).
- All green paths are eligible, and we can use a similar approach to case 1 to proof that blue is not optimal.



- Case 3: p -> q
- This case is either similar to case 1 or similar to case 2.

Problem D: Station

- Now the problem is how to get the length of the shortest path of from x to y, where y is an ancestor of x and we can’ t pass through points closer to the root than y.
- First, we can proof that the shortest path does not go through any point other than the path from x to y on the Cartesian tree. (It is similar to the proof of property 1.)
- We divide the path into several sub-paths like the picture on the right.
- Notice that jumping from one sub-path to another will inevitably jump to the deepest two nodes of the previous one.



Problem D: Station

- So we can get the length of the shortest path with a simple DP.
- Let f[i][0/1] be the answer of the length of the shortest path from one of the bottom two nodes of i-th sub-path.
- And the transfer of this DP can be described by a matrix, and we can use a segment tree to maintain.

Problem D: Station

- Now let's consider the case if a[i] has duplicates.
- We just need to slightly modify the process of building the Cartesian tree.
- Consider using recursion to build tree:
- Suppose now is building [l,r] and the father of the root of [l,r] is r+1, we select the rightmost position that satisfies its value as the interval max as the root of [l,r], otherwise we select the leftmost position that satisfies its value as the interval max as the root of [l,r].
    - The weight on edge(x,fa2[x]) should be recalculated due to duplicates.
- We can use a similar method to prove that the previous approach is still valid.
- Time Complexity: O((n+q) log n).

Problem E: Number guessing

- Suppose $n=3$ and $p=nk-1$ for some integer k , then H return fractional part of seed/p in ternary. If $p=nk+1$, then H return fractional part of $1-\mathrm{seed}/\mathrm{p}$. The same holds for $n=4$.

- If we ask $x=1$, then Alice must return 1 or 2. So there are at most 2 possible values of seed after 38 queries. Try to find the correct one in just one additional query.
  - $n=4$ we can ask $\mathrm{x}=10^{18}$, $n=3$ we can ask $x=1$.

- After finding the value of seed, we can have a simple binary searching, and it takes 60+38+1=99 queries at most.

Problem F: Build a City

- Let x[1..n] and y[1..n] denote the settlements' coordinates. Suppose x[1]<=x[2]<=...<=x[n].
- Let f(x,y) denote whether it is possible to transform the city into a rectangle whose upper right corner is (x,y).
- A necessary condition for f(x,y)=1 is the existence of i satisfying x[i]=x and y[i]<=y and the existence of j satisfying x[j]<=x and y[j]=y. Now we will update f(x,y) for all points satisfying the necessary condition.
- We can search the 2-D map from left to right. Each time we consider all settlements whose horizontal coordinates are the smallest. Suppose they are (x[l],y[l]) ... (x[r],y[r]), and y[l]<=...<=y[r]. Let S denote the set of the vertical coordinates which have appeared before.

Problem F: Build a City

- Now we need to update f(x[l],y[l]), … ,f(x[l],y[r]), and f(x[l],y) for all y which is not less than y[l] in the set.
- Consider f(x[l],y) for all y which is not less than y[l] in the set S. It's easy to see that for all y in the set S which is not less than y[l], the horizontal coordinates of the rightmost point satisfying the necessary condition above is non-decreasing. If we consider the same horizontal coordinate as a segment, these rightmost points can be distinguished into many segments.
- For each segment, only the points whose vertical coordinates are small enough can update to the right. Suppose the upmost point in the segment is (x',y'), we then simply use (x[l],y') to update upwards.
- It can be seen that these segments can be stored in a stack.

Problem F: Build a City

- Consider f(x[l],y[l]) … f(x[r],y[r]). Since there are only n settlements in total, it's easy to deal with this case.
- The method above can be proved to be sufficient and can be implemented using segment trees. The time complexity is O(nlogn).

Problem G: Trans

$$b_i = \sum_j (\text{popcount}(i \,\&\, j) \bmod 2) \times a_j = \frac{\sum_j a_j - \sum_j (-1)^{\text{popcount}(i\&j)} a_j}{2}$$

$$c_i = \sum_j (-1)^{\text{popcount}(i\&j)} a_j = \text{FWT}(a)_i$$

- So we can solve it in $O(2^n n)$

Problem H: Blind Box

- The question is equivalent to finding the value of the following formula taking modulo of 998244353.

$$\frac{\sum_{1\le x_1\le x_2\le\cdots\le x_n\le m} \prod_{i=1}^{n} x_i}{\sum_{1\le x_1\le x_2\le\cdots\le x_n\le m} 1}$$

- The denominator is easy to calculate, and the value is $\binom{n+m-1}{n}$

Problem H: Blind Box

- The value of the numerator is equal to the number of schemes that put n+m different balls in m identical boxes, and each box have at least one ball.
  - Consider putting the ball into the box one by one.
  - Note that there are two operations for the current ball, either put it into a empty box or put it into a non-empty box. So exactly n balls are put into boxes that already have balls.
  - If we put the ball into a non-empty box, we have chosen the value of x_i as the number of current non-empty boxes. And the way put into the box is exactly x_i。

- Therefore the numerator is S(n+m,m), which is The Stirling numbers of the second kind. It can be simply done in O($n \log n$).

Problem I: EIP

- Maintain a segment tree for the dimension of maxFee, with each node rt stores the maximum value of maxPriorityFee in its subtree maxv[rt].

- For each baseFee query, search from the root of the segment tree, assuming that it is now at node rt (the interval stored in this node is [l, r], and its left child stores [l, mid] and the right child stores [mid+1, r])
  - If maxv[rt.rson]>=mid-baseFee, then the left side is definitely not the optimal, recurse to the right son;
  - Otherwise, the optimal value on the right is maxv[rt.rson], recurse to the left and return the answer when backtracking.

- For insertion and deletion, just maintain a heap/BST for each maxFee

Problem J: Three Countries

- Consider the boundary of the convex hull. There are two possible cases.
  - An outer tangent line of two circles.
  - An arc connecting two outer tangent points of a circle.

- Let T be the set of outer tangent points excluding those that are strictly inside some circle.

- Let CT be the convex hull of T. The answer is area(CT) plus the area corresponding to all arcs.

Problem K: Cat

- We have following definition:
  - Define a cell is an exit if it locates in the boundary of the house without rock.
  - Define S[a] := the adjacent exits after the cat moves in direction a.
  - Define A := {a : a is valid and |S[a]|>=2}.
  - Define T := $\bigcap_{a \in A} S[a]$, and T := ∅ if |A|=0

Problem K: Cat

- The first six seconds, choose (9,1), (8,9), (-1,8), (-9,-1), (-8,-9), (1,-8)
- Then for every second:
  - If the cat is near an exit, choose it.
  - Else If $|A|=1$, choose the cell in the direction a(a in A) of the cat.
  - Else It can be proved that $|T|<=1$, and we can choose the unique element in T if $|T|<=1$, and an arbitrary cell otherwise.

Thank you!

icpc.foundation icpc | university commons

icpc | asia icpc | nerc