ICPC Amritapuri Regionals 2022 Solutions

Final Over

Problem

Given the number of runs (X) required by a team to win in the final over of a cricket match, and the results of the first five balls of the over, decide if they can win on the last ball.

Author: Janmansh

Final Over

Solution

- Here in the problem, we are asked whether the batting team can win or not if it needs X runs in final over.
- We know that in 1 ball maximum 6 runs can be made and an over has 6 balls.
- So, we just need to check if the difference of X and sum of runs made in first 5 balls is less than or equal to 6 or not.

Problem

Given an array A, find the MEX of the medians of all its subarrays.

Author: Aryan

Solution

- According to the definition, the median will always be one element of the chosen subarray.
- We can choose all length one subarray to get all elements present in the given array.
- All the other subarray median will be one among these.
- So, finding the MEX of a given array is enough.

Longest strictly increasing sequence

Problem

Given an array b of length n, find an array a of length N such that b_i is the length of the longest increasing subsequence of $[a_1, a_2, \ldots, a_i]$.

Author: Aryan

Longest strictly increasing sequence

Solution

- There are many valid solutions.
- One of the solutions is to choose $a = b$, and then check if it satisfies the given LIS conditions.
- It can be proven that if the answer is YES, the $a = b$ will also satisfy this.

Knockout Miracles

Problem

N teams play a knockout tournament, with team i being the i-th strongest team.
You're given the bracket of the tournament. For each team i, find the smallest number of miracles (i.e, matches where a weaker team beats a stronger one) required for it to win the whole tournament.

Author: Aryan

Solution

- Let's say we want to find the minimum number of miracles for team i to win.
- By exchanging arguments, we can prove that it's always optimal for miracles to happen in matches team i plays if required.
- We can do a dfs first to win winners of each subtree, assuming no miracles.
- This gives us $O(N^2)$ solution where we visit all ancestors of team i where miracle must happen.

Solution (continued)

- Now, we need a list of all winners of other siblings subtree of ancestors.
- In this list we need no of elements $< i$.
- We can maintain this list in ordered set/pbds.
- We should left subtree winner, while moving down to the right subtree during DFS, and remove it while up the DFS.
- Vice versa when moving up/down in left subtree.
- When we reach the leaf node, we can query no of elements less than $< i$, in pbds.

Rectangle Intersection

Problem

Given N and $0 \leq K \leq \binom{N}{2}$, find a multi-set of N rectangles, such that there are exactly K unordered pairs of rectangles that intersect.

Author: Jatin Yadav

Rectangle Intersection

Solution

- There exist a lot of different approaches.
- One way is to find a solution recursively:
- If $K < N - 1$, let's make the rectangle $1 \leq i \leq K + 1$ have corner cells $(1, i) and (1, i + 1)$. For $i > K + 1$, let the rectangle i be the single cell (i, i) .
- If $K \geq N - 1$, let one rectangle have corners $(1, 1)$ and $(1000, 1000)$, and then recursively find $N - 1$ rectangles with $K - N + 1$ intersections.

No Enemy

Problem

Given a $2-D$ grid and N people, every person has an initial position and direction. Every minute they rotate 90 degrees anti-clockwise. You have to perform operations such that no two people see each other. In one operation you can rotate a person by 180 degrees. Minimize the number of operations.

Author: Chaithanya Shyam

Solution

- Let us say two people are "adjacent" if there is a horizontal or vertical line from one to the other such that no other person is in between them. Let us say L, R are horizontal directions and U, D are vertical directions.
- Here are a few observations:
  - Two non-adjacent people won't become enemies.
  - If there are two adjacent people and one of them is facing in a horizontal direction and the other is facing in a vertical direction then they won't become enemies.
  - If two adjacent people are in the same horizontal/vertical direction but they are facing the same way then they won't become enemies.
  - If two adjacent people are in the same horizontal/vertical positions and they are facing opposite ways then they will become enemies at some point in time.

Solution (continued)

- Now we can model the problem as a graph.
- Every person corresponds to a vertex.
- We add an edge between two people if they are adjacent and facing the same horizontal direction or the same vertical direction.
- For every connected component in this graph let there be x people facing a particular direction and y people facing the opposite direction.
- We need to make every person in this component face the same direction, because if this is not the case then there will be two adjacent people that are facing opposite directions. These two people will become enemies at some point in time.
- So we necessarily need at least $min(x, y)$ operations.

Solution (continued)

- It can easily be seen that $min(x, y)$ is also sufficient to prevent any enemies for every component to get the minimum number of operations.
- Since every component is independent we only need to get the sum of $min(x, y)$ for every component.
- This can easily be done in linear time using BFS/DFS/DSU.
- To add edges to the graph we can store the points based on their x/y coordinates in a vector of vectors after using coordinate compression OR use a map of vectors.
- Overall time complexity is $O(N \log N)$

Problem

You have an array A with n elements. In one move, you can choose a non zero element x from A, and subtract x from all elements $\geq x$ in A. What is the minimum number of moves needed to make all elements of A equal to 0.
$1 \leq N \leq 42$, $1 \leq A_i \leq 42$.

Author: Vaibhav Tulsyan

Solution

- In one move, we can always reduce the highest set bit in the binary representation of some element of A.
- If the highest set bit is k, choose $x =$ the smallest value $\geq 2^k$ in A.
- So, we can always solve the problem in 6 moves.
- Let $F(h, l)$ be the maximum number of reachable states starting at an array with maximum value h, and making l moves.
- When you apply a move x, the new maximum becomes $\leq \max(x - 1, h - x)$
- Therefore, $F(h, l) \leq 2(F(h-1, l-1) + F(h-2, l-1) + ..)$
- $F(h, l) \leq 2^l \times \binom{h}{l}$

Make It Zero

Solution(continued)

- An array can be made zero in one move if and only if it has at max one distinct non zero element.
- An array can be made zero in two moves if and only if it has ≤ 2 distinct non zero elements, or has 3 distinct non zero elements $a, b, a+b$, for some a, b.
- So, let's run a bfs upto depth 3,and from there we can figure out whether answer can be 4 or 5. Else, the answer is 6.
- The number of states is $F(42, 3) \leq 2^3\binom{42}{3} < 10^5$.

Finding the best possible partners

Problem

A kingdom has N men and N women. Initially, the i-th man is matched with the i-th women.
You are given each man's preference order of the women. Man i can be matched with woman j only if woman j is not a worse match than his initial one.
Find, for each i, the best possible match man i can have in some valid matching.

Author: Mukesh

Finding the best possible partners

Solution

- Let $P_{i,j}$ denote the j-th highest preference of man i.
- Clearly, if k is the index such that $P_{i,k} = i$, all $j > k$ can be ignored.
- Model this as a graph on $2N$ vertices; N each for the men and women. Add a directed edge from man i to woman j if they can be matched.
- Note that if man i is to be matched with woman j, then:
  - If man j can be matched with woman i, there's no issue.
  - Otherwise, man j must be matched with some other woman, say k.
  - This then necessitates a new match for man k, and so on until some man is matched with woman i.

Finding the best possible partners

Solution (continued)

- Let this sequence of 'replacement' matches be $i \to x_1 \to x_2 \to \ldots \to x_m \to i$, where:
  - Man i is matched with woman x_1
  - Man x_j is matched with woman x_{j+1} for $1 \leq j < m$
  - Man x_m is matched with woman i
- Notice that this is possible only when each of the $m + 1$ corresponding directed edges exist in the graph we created initially.
- Analyze the structure of the edges and note that this is quite similar to a directed cycle.
- In fact, if we add a directed edge from woman i to man i for each i, this is indeed a directed cycle!

Finding the best possible partners

Solution (continued)

- In particular, note that if there exists a directed cycle containing man i and woman j, then there exists a valid matching where man i is paired with woman j.
- So, our task reduces to finding, for each man, his highest preference woman that such that a cycle contains them both.
- Given the specific structure of our graph, man i and woman j lie a common directed cycle if and only if they lie in the same strongly connected component.
- So, compute the SCCs of the directed graph we created, then find for each man the highest preference woman who lies in his SCC.
- Overall time complexity is $\mathcal{O}(N^2)$.

Maximum Bitwise OR

Problem

Given an array $A = A_1, A_2, \ldots A_N$ and Q queries. Each query consists of X, Y. Let B to be the subarray $A_X, \ldots A_Y$. In one move you can choose a value x from B, and replace x by $x \oplus (x - 2^i)$ for some i with $2^i \leq x$.
You want to maximize the bitwise OR of all values in B, and determine the minimum number of moves required for this purpose.

Authors: Jatin Garg, Jatin Yadav

Maximum Bitwise OR

Solution

- When we apply a move $x \to (x \oplus (x - 2^i))$, we get 2^i if the bit i is set in x, else we get a value that has exactly bits $[i, j]$ set, where j is the smallest bit $> i$ that is set in x.
- If the highest set bit over all values is k, then no matter what we do, we can not set any bit $> k$.
- But we can always set all the bits $\leq k$. To do so, choose any element x with k^{th} bit set. First, replace x by $x \oplus (x - 2^k) = 2^k$. Now replace 2^k by $2^k \oplus (2^k - 2^0) = 2^{k+1} - 1$
- So the maximum possible answer is $2^{k+1} - 1$, and is achievable in ≤ 2 moves.

Solution(continued)

- If initial bitwise OR is $2^{k+1} - 1$ itself, then number of moves required is 0. Else, we need to decide if we can get away with only 1 move.
- Let l be the lowest bit and r be the highest bit $\leq k$ not set in any value.
- Let's call a value special if there is a bit that is set in only that value. We'll treat special values separately.
- It is easy to find all the special values in $O(\log M)$, by precomputing the next position containing a given bit for every index.
- We can succeed by applying a move on a non-special value x, if and only if all bits in the range $[l, r]$ are 0 in x.

Solution(continued)

- Determining whether there exists such a value is doable by storing for every r, a segtree with min operation over an array whose i^{th} value is the largest bit $b \leq r$ contained in A_i.
- Consider a special value x. If a bit j is only contained in x, then we need to make sure that the j^{th} bit remains set even after the operation.
- So, we must select a value $i \leq l$ such that j is the smallest bit $> l$ set to 1 in x.

Minimal Increasing Graph

Problem

Given a permutation P, find the minimum possible edges in a directed graph with N nodes, such that node j is reachable from node i in the graph if and only if $i < j$ and $P_i < P_j$

Authors: Vaibhav Gosain, Jatin Yadav

Minimal Increasing Graph

Solution

- Clearly, an edge (i, j) is mandatory if $i < j$, $P_i < P_j$, and there doesn't exist a $i < k < j$, with $P_i < P_k < P_j$.
- Also the set of mandatory edges is sufficient to satisfy the reachability criteria. This can be proved using induction on $j - i$.
- If $i < j$ and $P_i < P_j$, either (i, j) is mandatory and we are done or there exists a $i < k < j$ with $P_i < P_k < P_j$, and we can go $i \to k \to j$.
- So, the problem reduces to finding the number of mandatory edges.

Minimal Increasing Graph

Solution (continued)

- We use divide and conquer to solve this efficiently.
- Suppose we want to find the number of mandatory edges (i, j) satisfying $L \le i < j \le R$.
- Let $M = \left\lfloor \frac{L+R}{2} \right\rfloor$. First, recursively find the number of mandatory edges in $[L, M]$ and $[M+1, R]$ respectively.
- Now, we need to find the number of mandatory edges (i, j) with $L \le i \le M$, and $M < j \le R$.
- For each $L \le i \le M$, let $F_i = \min\{P_k : i < k \le M, P_k > P_i\}$, that is the minimum value greater than P_i to its right. These values can be found in $O(M - L + 1)$ by iterating over the indices i in decreasing order of P_i, and finding suffix minimums.

Solution (continued)

- Similarly, for each $M < j \leq R$, let $G_j = \max\{P_k : M < k < j, P_k < P_j\}$
- Note that (i, j) is mandatory if and only if $G_j < P_i < P_j < F_i$.
- For each $L \leq k \leq R$, store $H_k = F_k$, if $k \leq M$, and $H_k = P_k$ otherwise.
- Iterate on k in decreasing order of H_k. Doing this, when we reach a j, we ensure that we only consider i with $F_i > P_j$. So, now we only need to count i with $G_j < P_i < P_j$, which can be done using a fenwick tree.
- Using coordinate compression on the values in range $[L, R]$, this takes $O((R - L + 1)\log(R - L + 1))$ overall.

Minimal Increasing Graph

Solution (continued)

- Overall time complexity is therefore $T(N) = 2T(N/2) + O(N \log N) = O(N \log^2 N)$.

Exam Requirements

Problem

There are N exams where each exam i happens continuously from time S_i to E_i. To pass an exam, you need to attend it in its entirety. You can only attend non-overlapping exams. To graduate, there M requirements which you need to fulfill, each requirement is of the form: pass at least one of exams A or B. Constraints: $N \leq 10^5$, $M \leq 10^5$.

Author: Vaibhav Gosain

Solution

- We can model the decision to attend or not attend each exam as a separate Boolean variable.
- For every requirement A,B, the following boolean expression must hold true: $A|B$.
- For every pair of exams X,Y that intersect, the following boolean expression must hold true: $X'|Y'$.
- All such boolean expressions must hold true in order to graduate with a valid exam attendance. Hence, the overall boolean expression which we need to satisfy is of the form: $(A_1|B_1)\&(A_2|B_2)\&...\&(X_1'|Y_1')\&(X_2'|Y_2')\&...$ which is a boolean expression in conjunctive normal form (2-CNF)

Solution (continued)

- Checking satisfiability for a 2-CNF boolean expression is a well-known problem, also known as 2-satisfiability (2-SAT): https://en.wikipedia.org/wiki/2-satisfiability . This can be solved by modelling the boolean expression as a graph, and adding implication edges for each "or" condition in the expression.
- Because there can be $O(N^2)$ intersection ranges, there can be $O(N^2)$ "or" conditions in our boolean expression. To reduce the number of implication edges in the 2-SAT graph, we can create dummy segment-tree like nodes which are responsible for a "segment" of a contiguous range of exams, sorted by start or end times.

Solution (continued)

- Now for each exam, instead of making direct implication edges to other intersecting exams, we can make edges to $O(\log N)$ segment tree dummy nodes which are responsible for ranges of exams which start or end in this exam's range. This reduces the number of edges in the graph to $O(N \log N)$ and overall time complexity is also $O(N \log N)$.

Problem

Given is a prime p and a threshold k. Consider a binary string w, where $w_i = 1$ if there exists some $0 \leq z < p$ with $z^2 = i \pmod{p}$, and 0 otherwise. Find the number of triples of indices (i, j, k) with $j - i = k - j$, $1 \leq j - i \leq k$, and $w_i = w_j = w_k$.

Author: Harris Leung

Solution

- Consider some difference value $d = j - i$.
- We need $1 \leq i \leq p - 1 - 2d$.
- Let's first allow the whole range $1 \leq i < p$ and then we'll remove the contribution of all i for which we need a wraparound $(p - 2d \leq i < p)$
- Let $g_i = i^{\frac{p-1}{2}}$, and $h_i = (g_i g_{i+d} + g_{i+d} g_{i+2d} + g_{i+2d} g_i + 1)/4$
- Note that $h_i = 1$ if $w_i = w_{i+d} = w_{i+2d}$, else $h_i = 0$.
- So, the number of triples for a given d is $\sum_{i=1}^{p-1} h_i$
- $\sum_{i=1}^{p-1} i^r = 0 \pmod{p}$ for all $1 \leq r < p - 1$, and $p - 1 \pmod{p}$ for $r = p - 1$.

Solution(continued)

- $g_i g_{i+d} = (i(i+d))^{\frac{p-1}{2}} = i^{p-1}$ + some weighted sum of lower non-zero powers of i.
- $\sum_{i=1}^{p-1} g_i g_{i+d} = p - 1 \pmod{p}$
- Now, we have to just remove the contribution of all $1 \le i < p, 1 \le d \le k$ with $i + 2d \ge p$.
- Form a new array A of length $4k$. Where $A_{2k-i} = g_{p-i}$ for $1 \le i \le 2k$ and $A_{2k+i} = g_i$ for $0 \le i < 2k$
- In this array, we need to find the sum of $A_i A_{i+d} + A_{i+d} A_{i+2d} + A_{i+2d} A_i + 1$ over all i and d satisfying $0 \le i < 2k$, $2k \le i + 2d < 4k$.
- This can be done in $O(k)$ using prefix and suffix sums stored for both odd and even indices.

Talk That Talk

Solution(continued)

- The overall time complexity is $O(k \log p)$, for finding the array A.