Universal cup Stage 20 : India

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 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.

Maximum Bitwise OR

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 and $x > 2^r$

Maximum Bitwise OR

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.
- Now 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. Note that j must be the only special bit of x, because there is at max one bit where we can have 1, both before and after applying the operation.
- So, we must select some $i \leq l$ such that j is the smallest bit $> l$ set to 1 in x. This works if all bits in range $[l, r]$ are 0 in x, and $j > r$.

Minimize Median

Problem

You are given two arrays A and $cost$, of lengths N and M respectively. N is odd

In one move you can:

- Pick an index $1 \leq i \leq N$ of A and an integer x between 1 and M.
- Set $A_i \leftarrow \left\lfloor \frac{A_i}{x} \right\rfloor$ for a cost of $cost_x$.

Using moves with a total cost of at most K, what's the minimum possible median of A?

Author: Nishank Suresh

Minimize Median

Solution

- Dividing by a larger number is never worse, so as a preliminary step, let $cost_i = \min(cost_i, cost_{i+1}, \ldots, cost_M)$.
- Note that if the median can be made $\leq x$, it can of course be made $\leq x + 1$. So, we can binary search for the smallest valid x.
- For a fixed x, we'd now like to compute the minimum cost to make the median $\leq x$.
- Let $A_1 \leq A_2 \leq \ldots \leq A_N$, and $N = 2k + 1$.
  Then, it's optimal to try and make $A_1, A_2, \ldots, A_{k+1}$ all be $\leq x$.
- To make A_i fall to $\leq x$, we need to divide it by at least $\left\lceil \frac{A_i+1}{x+1} \right\rceil$ in total.

Solution (continued)

- Note that for any positive integers x, y, z, floor division has the following property:

$$\left\lfloor \frac{\left\lfloor \frac{x}{y} \right\rfloor}{z} \right\rfloor = \left\lfloor \frac{x}{yz} \right\rfloor$$

  meaning that a sequence of floor divisions can instead be thought of as a single one, with the costs summed up.
- Let's compute $f(y)$, the minimum cost to divide by exactly y.
- Initialize $f(y)$ to $cost_y$, and then for each divisor d of y, relax it with $cost_d + f(y/d)$.

Solution (continued)

- Computed using dynamic programming in a sieve-like fashion, this takes $\mathcal{O}(M \log M)$ time in total.
- Note that some care is needed with values near M, because it might be optimal to divide by a larger value than M. However, we only care about the first multiple of each integer that exceeds M, so you can for example compute $f(y)$ values upto $2M$ (treat the missing *cost* values as ∞).
- Finally, since dividing by larger numbers is not worse, $f(y)$ can be set to the suffix minimum $\min(f(y), f(y+1), \ldots, f(M))$.
- Once all the $f(y)$ values are known, the cost of making the median $\leq x$ is easily calculated in $\mathcal{O}(N)$: simply sum up $f\left(\left\lceil \frac{A_i+1}{x+1} \right\rceil\right)$ across all relevant A_i.

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

Exam Requirements

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)

Exam Requirements

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.

Exam Requirements

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 $O(N \log N)$.

Central Subset

Problem

You are given a simple undirected connected graph on N vertices and M edges. You have to select a subset of vertices S, of size at most $\lceil \sqrt{N} \rceil$ such that for every vertex u in the graph there exists a vertex v in S such that $dist(u, v) \leq \lceil \sqrt{N} \rceil$. Find any such subset or output -1 if it doesn't exist.

Author: Chaithanya Shyam D

Central Subset

Solution

- It can be proven that such a subset always exists. Since problems on trees are usually easier than problems on graphs, let us convert this into a tree problem.
- Let us take any Spanning Tree of the graph. If we can find a valid subset for this tree then the same subset will be valid for the original graph, because $dist(u, v)_G \leq dist(u, v)_T$ for all u, v.
- Root the spanning tree at an arbitrary vertex. Let v be the deepest leaf of the tree. Let $anc(v)$ be the ancestor of v that is $\lceil \sqrt{N} \rceil$ levels above v, if the depth of v is less than $\lceil \sqrt{N} \rceil$ then let $anc(v)$ be the root.

Central Subset

Solution (continued)

- Notice that for each vertex u in the subtree of $anc(v)$, $dist(u, anc(v)) \leq \lceil \sqrt{N}. \rceil$. So add $anc(v)$ to S and remove the subtree rooted at $anc(v)$.
- We now have to solve the same problem for a smaller tree. We recurse until we add the root to S.
- We add exactly one vertex to S per iteration. So $|S| =$ number of iterations. In each iteration, we are removing at least $\lceil \sqrt{N} \rceil$ vertices from the tree.
- Since there are only N vertices, there can be at most $\lceil \sqrt{N} \rceil$ iterations. Therefore $|S| \leq \lceil \sqrt{N} \rceil$, and hence S is a valid subset.

Solution (continued)

- Naively running the above algorithm will give a time complexity of $O(N\sqrt{N})$. But it can be optimized if we precompute the depths of all vertices and add them to a set.
- Since we also need to maintain a DSU to get the spanning tree, the final time complexity is $O(N \log N + (N + M) \cdot \alpha(N))$, and space complexity is $O(N)$.
- There exists an alternate solution with time complexity $O((N + M) \cdot \alpha(N))$ with just a single DFS after getting the spanning tree.

Strange Keyboard

Problem

You are given N strings $S_1, S_2, \ldots, S_N$, an integer K, and a target string T.
Starting with an empty string S, in one move you can:

- Append one of the S_i to S, or
- Delete the last K characters of S (if it has at least K characters). This move cannot be used if the current length of S is less than K.

Decide if it is possible to obtain the string T via a sequence of operations; and if so, find the minimum number of moves required to do so.

Author: Jatin Yadav

Strange Keyboard

Solution

- We'll use dynamic programming.
  Let dp_i be the minimum number of moves required to form **exactly** the prefix of length i of T.
- We have the transition $dp_i = \min(dp_j + c(j+1, i))$ across all $j < i$, where $c(x, y)$ is the minimum number of moves required to form the string $T[x : y]$, starting from scratch.
- To compute the $c(x, y)$ values, we first precompute $cost(x)$: the number of moves required to delete exactly x characters from the suffix of the current string.

Solution (continued)

- Notice that we can always attempt to process deletions before appends, and so:
  - $cost(0) = 0$
  - If $x \geq K$, then $cost(x) = 1 + cost(x - K)$ by deleting K characters first.
  - If $x < K$, we can attempt to append one of the N strings we have. However, the actual string we append doesn't matter: only its length.
    So, $cost(x) = 1 + \min(cost(x + L))$ across all L that occur as the length of some S_i.
- There are only $\mathcal{O}(\sqrt{\sum |S_i|})$ distinct lengths of input strings, so all the *cost* values can be computed in $\mathcal{O}(\max(|S_i|) + K\sqrt{\sum |S_i|})$ time.

Strange Keyboard

Solution (continued)

- Finally, put all S_i into a trie, and use the *cost* array to find the minimum number of moves to form each prefix.
- This also tells us the $c(x, y)$ values we wanted to know for our initial DP, because the only relevant strings we can form are combinations of prefixes of the S_i.
- Complexity is $\mathcal{O}(K\sqrt{\sum |S_i|} + 26 \cdot \sum |S_i| + |T|^2)$
- Note that the sum of K across tests wasn't bounded, so $\mathcal{O}(K^2)$ per test would TLE.

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.

Perfect Strings

Problem

Call a string S of $2n$ characters perfect if its $2n$ indices can be partitioned into n pairs, such that:

- Each index belongs to exactly one pair
- For each pair (i, j), $S_i = S_j$
- No two pairs are entangled, that is, for any two pairs (i, j) and (k, l), $i < k < j < l$ must NOT be true.

Consider a character set σ with c characters. Count the number of perfect strings of length $2n$, each of whose characters is in σ

Author: Jatin Yadav

Perfect Strings

Solution

- Note that, in any perfect string, there must be two adjacent equal characters somewhere. Else, consider the closest pair in the partition, say (i, j). Clearly, $j > i + 1$, and any $i < k < j$ must be unmatched.
- Also, for any such pair of adjacent characters, it is optimal to pair them together. This is because if the partition had (i', i) and (j', j), then one can verify that it is also valid to pair (i, j) and (i', j') instead.
- This means that, in a perfect string, if we start from left to right and keep pairing(and removing) equal characters, we must end with the empty string in the end. This can be simulated with a stack, where we pop if current character is equal to the stack's top element, and else push the current character onto the stack.

Solution (continued)

- Let $f(n)$ be the number of perfect strings of length $2n$.
- Let $g(n)$ be the number of perfect strings of length $2n+2$ in which the first and last characters are both **a** and the stack is only empty before pushing the first character, or after all the characters are processed.
- $g(0) = 1$ and $g(n) = (c-1)\sum_{k=0}^{n-1} g(k)g(n-1-k)$ for all $n \geq 1$.
- The generating function $G(x)$ satisfies $G(x) - 1 = (c-1)xG^2(x)$
- $G(x) = \dfrac{1 - \sqrt{1-4(c-1)x}}{2(c-1)x}$

Solution (continued)

- Also, $f(0) = 1$ and $f(n) = c\sum_{k=0}^{n-1} g(k)f(n-1-k)$ for all $n \geq 1$
- So, $F(x) - 1 = cxG(x)F(x) \implies F(x) = \frac{1}{1 - cxG(x)}$
- $F(x) = \frac{1}{1 - \frac{c}{2(c-1)}(1 - \sqrt{1 - 4(c-1)x})}$, which can be rearranged in the form $\frac{\alpha + \beta\sqrt{1 - \gamma x}}{1 - \delta x}$ for some constants $\alpha, \beta, \gamma, \delta$
- Expanding this gives an $O(n)$ formula

Treelection

Problem

Given is a rooted tree. There'll be an election in which each node except the root has to vote for one of its strict ancestors. Find out which nodes can end up being the sole winner of this election

Author: Jatin Yadav

Solution

- Define $f_{t,x}$ to be the minimum number of votes that must go from subtree of x to its ancestors, if each node is allowed to get at max t votes.
  $f_{t,x} = [x \neq 1] + \max(0, t - \sum_{y \in \text{children(x)}} f_{t,y})$
- First, find the minimum value z, such that it is possible that everyone gets $\leq z$ votes. Clearly, z equals the least y for which $f_{y,1} = 0$, and can be found using binary search
- For a node x, if subtree size S_x (considering only strict descendants) of x is greater than z, then it can definitely be the sole winner of the election. This is because, we can consider the voting in which everyone gets $\leq z$ votes, and then in this voting change the vote of every node in x's subtree to x.

Solution (continued)

- Also, if $S_x < z$, it is impossible for x to be the sole winner of the election because x can receive at max $z - 1$ votes, but at least one node must receive $\geq z$ votes.
- If $S_x = z$, we want to make sure that x is the only node that gets z votes, and all other nodes get $< z$ votes.
- First find the values of $f_{z-1,y}$ for all y. In finding these values, we used the condition that each node can receive at max $z - 1$ votes. We need to make one small change: what will be the new f values if this limit is updated from $z - 1$ to z for the node x?
- So, we want to see the change in the f values if we update $f_{z-1,x}$ to 0.

Solution (continued)

- If any ancestor (including x itself) y of x has $f_{z-1,y} = 0$, then x can not be the sole winner, because one can't change the f values of the ancestors of y, and hence the root.
- Else, $f_{z-1,x} = 1$ (note that, since $S_x = z$, $f_{z-1,x}$ must be ≤ 1), and we can reduce the f values of the ancestors of x by at max 1. So, $f_{z-1,1}$ must also be 1

Disk Tree

Problem

Given N disjoint disks on the plane, draw $N - 1$ line segments connecting them such that:

- Each segment has integer endpoints between 0 and 10^9.
- Each segment touches or intersects at most two disks.
- The line segments should not touch or intersect each other, but they may share an endpoint.

Author: Jeroen Op de Beek

Disk Tree

Solution

- Sweep over x-coordinates from left to right - in particular, the events of note are the leftmost and rightmost points of the disks ($x_i - r_i$ and $x_i + r_i$, respectively).
- The leftmost points are insertions, and the rightmost points are deletions.
- When performing the sweep, keep a set S of 'active' circles, sorted by the y-coordinate of their centers.
- For a fixed x-coordinate, process in increasing order of y-coordinate; and insertions before deletions. Then, do the following:
  - When inserting a circle, use a vertical segment to join it to the closest circle in S above/below it. The closest circle can be found using binary search. Then, insert this circle into S.
  - When deleting a circle, just delete it from S.

Disk Tree

Solution (continued)

- Make sure that the vertical segments created this way don't overlap.
- At the end of the sweep, we'll have several connected components of circles, defined by some disjoint intervals along the x-axis.
- To connect everything, draw segments from the left endpoint of one interval to the right endpoint of the previous interval; these obviously don't intersect the vertical segments we created during the sweep.
- Note that some vertical segments might have negative x-coordinates; however, they can all be shifted to $x = 0$ without losing connectivity (once again, make sure they don't overlap), because the right endpoints of all circles have positive x-coordinates.

Talk That Talk

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 $1 \leq i < p, 1 \leq d \leq k$ with $i + 2d \geq p$.
- Form a new array A of length $4k$. Where $A_{2k-i} = g_{p-i}$ for $1 \leq i \leq 2k$ and $A_{2k+i} = g_i$ for $0 \leq i < 2k$
- 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 $0 \leq i < 2k$, $2k \leq i + 2d < 4k$.
- 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.

XOR Dice

Problem

Given integers $n \leq$ and $d \leq 60$, find n dice with faces labeled with integers in $[0, 10^6]$, such that:

- For each die, the numbers written on its 6 faces are distinct
- If you roll all the dice, the bitwise xor of all the n numbers on the top is always divisible by d

Author: Arul Kolla

XOR Dice

Problem

- Consider the numbers of the form $d \times (a_0 + a_1 \times 2^6 + a_2 \times 2^{12})$, where each a_i is 0 or 1. These are numbers such that bits $0 \ldots 5$ equal either 0 or d. Similarly, bits $6 \ldots 11$ equal either 0 or d, and same for bits $12 \ldots 17$
- There are 8 such numbers given choices of a_0, a_1, a_2, and one can choose any 6 for each die.

(1,2) Nim

Problem

Solve the game of nim with a small change: the second player has to make 2 consecutive moves. (With the exception that he may make one move if only one non-zero pile is remaining)

Author: Jatin Yadav

(1,2) Nim

Solution

The following statements can all be proved using induction:

- Let k be the number of piles containing more than one stone.
- If $k \geq 2$, player 1 loses.
- If $k = 1$, player 1 wins if and only if $N \neq 2 \pmod 3$
- If $k = 0$, player 1 wins if and only if the $N = 1 \pmod 3$
- If it is currently player 2's turn, then he loses if and only if all piles have a single stone and $N = 0 \pmod 3$.

Graph and Colors

Problem

Given a complete undirected graph with N nodes, color each of its edges in one of K colors, such that for each color, the corresponding subgraph is connected and has a diameter ≤ 4

Author: Aryan

Solution

- Each subgraph needs at least $N-1$ edges. So, $K \leq \frac{N}{2}$.
- This condition is also sufficient. We'll construct a solution for $K = \lfloor \frac{N}{2} \rfloor$. If $K < \lfloor \frac{N}{2} \rfloor$, we can change the colors of all edges with color $\geq K$ to the color 0 (0− indexed colors).
- First assume even N. For each $0 \leq k < \frac{N}{2}$, let vertices $2k, 2k+1$ denote vertices of color k. Color the edge $(2k, 2k+1)$ with color k
- For each $0 \leq i < j < K$, color the edges $(2i, 2j)$ and $(2i+1, 2j+1)$ with color i and the edges $(2i, 2j+1)$ and $(2i+1, 2j)$ with color j.
- For any $j \neq i$, the node $2j$ is either adjacent to $2i$ or $2i+1$. Same for the node $2j+1$, and therefore the diameter for any color i is ≤ 3.

Graph and Colors

Solution (continued)

- For odd N, first create the solution for $N-1$, and then for each $i < N-1$, color the edge $(i, N-1)$ with the color $\frac{i}{2}$

Red Black Grid

Problem

Given integers N and K, color each cell of a $N \times N$ grid red or black such that the number of pairs of adjacent cells with opposite colors on them is exactly K or claim that no such coloring exists.

Author: Jatin Yadav

Red Black Grid

Solution

- There exists no solution if $K = 1$ or $K = 2 \times N \times (N-1) - 1$
- For $N \leq 3$, one can bruteforce. We'll now assume $N \geq 4$
- Assume $0-$indexing and call a cell (i, j) odd if $(i + j) = 1 \pmod 2$, and even otherwise
- First, color all the odd cells blue.
- Let the degree of a cell be the number of cells adjacent to it. We want to make the sum of degrees of the red cells equal K.
- For each $d = 2, 3, 4$, there are a non-zero number of even cells with degree d
- One can just iterate over the even cells in decreasing order of degree and decide whether to include it greedily with a few adjustments.