MITIT 2025 Winter Contest Editorials: Advanced Round 1

The MITIT Organizers

January 19, 2025

1 Number Reduction

Problem Idea: Michael Voigt
Problem Preparation: Michael Voigt
Analysis By: Albert Wang

1.1 Subtask 1

To solve the first subtask, we can iterate k from 1 through N and maintain a set of all valid numbers, starting with 1. When we process k, we check for each digit $2, \ldots, 9$ in its decimal representation, and set k to be valid of any of the appropriate $k/2, \ldots, k/9$ are also valid. This gives a solution in $O(N)$ time.

1.2 Subtask 2

Notice that if an integer $k > 1$ is divisible by any prime outside of $2, 3, 5, 7$ it must not be valid, as it is impossible to divide out prime factors larger than 10 from k. Therefore, all valid numbers have the form $2^a 3^b 5^c 7^d$. There are furthermore at most

$$\log_2(10^{18}) \log_3(10^{18}) \log_5(10^{18}) \log_7(10^{18}) \approx 1.23 \times 10^6$$

such numbers at most 10^{18}. So, we can draw the following graph: for any number k only divisible by primes $2, 3, 5, 7$, connect $k \to pk$ if p is in the decimal representation of pk for each $p = 2, 3, 5, 7$. A number is valid if and only if it has a path to 1 within this graph; as there are on the order of 10^6 nodes in this graph and $4 \cdot 10^6$, we may now compute the answer with a simple depth first search, for a solution in $O((\log N)^4)$.

2 Monster Fighting

Problem Idea: Albert Wang, Richard Qi
Problem Preparation: Albert Wang
Analysis By: Albert Wang

We'll represent our monsters as a pair $(p_{i,0}, p_{i,1})$ where $p_{i,j}$ is the maximum power of a type j monster that we can defeat.

Consider all type 0 monsters on the opposing team, and iterate through them in descending order of powers $q_1 \geq \cdots \geq q_k$. We will greedily assign one of our monsters to each of these opponents, beginning with q_1. This monster can be defeated by any of our monsters i with $p_{i,0} \geq q_1$. Notice that all monsters i that can defeat q_1 can also defeat all other type 0 monsters $q_2, \ldots, q_k$, as their power $p_{i,0} \geq q_1 \geq \cdots \geq q_k$. So, the values of $p_{i,0} \geq q_1$ can be considered to be effectively identical, and we can greedily choose the weakest candidate monster, i.e. one with a minimal value of $p_{i,1}$. We repeat with choosing the next monster satisfying $p_{i,0} \geq q_2$ with minimal $p_{i,1}$, and repeat until either all $q_1, \ldots, q_k$ are defeated.

After this process, all opposing type 0 monsters have been defeated, so we only need to focus on type 1 monsters, as well as type 1 powers $p_{i,1}$. We can once again do this with greedy, by iterating in descending order of the opposing monsters' powers and matching off the weakest monster on our side that can defeat them.

So, if we are able to pair off all monsters, the answer is YES, and otherwise we answer NO. This greedy can be implemented with a time complexity of $O(N \log N)$.

3 Not-So-Long Increasing Subsequence

Problem Idea: Anton Trygub
Problem Preparation: Albert Wang
Analysis By: Albert Wang

Recall that the length of the longest increasing subsequence is equal to the minimum number of decreasing subsequences that the permutation can be decomposed into. As a corollary, an array that can be decomposed into k decreasing subsequences must have an LIS of at most k. Now, the key claim is as follows:

Claim 3.1. *Let L be the length of the longest increasing subsequence of the whole permutation. The construction is possible if and only if $L + K - N \leq \frac{K+1}{2}$.*

Proof. If the inequality is not satisfied, then by Pigeonhole any subsequence of length K will intersect the longest increasing subsequence of the whole permutation in at least $\frac{K+1}{2}$ positions, and so a construction is impossible. In all other cases, decompose the original array into decreasing subsequences $D_1, \ldots, D_L$, and sort them such that their lengths are in decreasing order, $|D_1| \geq \cdots \geq |D_L|$. Consider choosing the elements of $D_1, D_2, \ldots$ until we have selected K elements in total; we claim that this gives the desired interesting subsequence b of the original array a. Let D_m be the final subsequence that we chose from.

If $D_1, \ldots, D_m$ all have length at least 2, then they give a partition of b into at most $\frac{K+1}{2}$ descending subsequences, and so b is interesting. Otherwise, D_m must have length 1. Then, $D_{m+1}, \ldots, D_L$ all have length 1, but we know that

$$N - K = L - m \implies m \leq L + K - N \leq \frac{K+1}{2}.$$

Therefore, $D_1, \ldots, D_m$ give a partition of b into at most $\frac{K+1}{2}$ decreasing subsequences, and so our construction also works in this case. We are done. □

Since we can partition a permutation into a minimal number of decreasing subsequences in $O(N \log N)$ time, our overall complexity is also $O(N \log N)$.

4 Drawing Lines

Problem Idea: Richard Qi
Problem Preparation: Richard Qi
Analysis By: Richard Qi

4.1 $N \leq 10$

Consider all 2^N possibilities, and check if there are any intersections between any pair of rays.

4.2 YES/NO

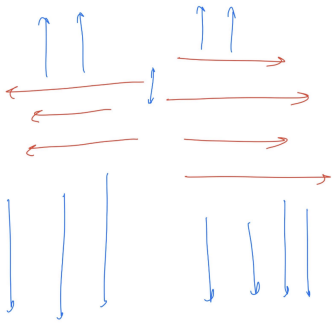
Let x_i denote a boolean variable, where if the ith ray is LR, $x_i = 1$ iff the ray points right, and similary if the ith ray is UD, $x_i = 1$ iff the ray points up. Then, for every pair of points, we enforce a condition on the OR of two (possibly negated) booleans. For example, if ray 1 is a LR at $(1, 2)$ and ray 2 is a UD at $(2, 1)$, then these two rays intersect iff ray 1 points right and ray 2 points up. This corresponds to $x_1 = 1$, $x_2 = 1$, so we need to enforce $\neg x_1 \vee \neg x_2$. In order for there to be no intersections, the bitwise AND of all of these expressions must be 1.

However, this is precisely a 2SAT! So, we can create all clauses and solve the instance for a half credit solution in $O(N^2)$.

4.3 Full Solution (Sketch)

Consider all of the LRs in order from left to right.

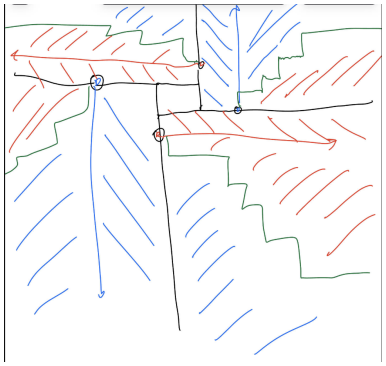
First, suppose all Ls come before all Rs. Then, let the rightmost L be L' and the leftmost R be R'. All UDs to the left of L' are forced to go in one direction by L', and similarly all UDs to the right of R' are forced to go in one direction by R'. All UDs with x coordinate in between L' and R' are free to go in either direction.



We can similarly count the number of satisfying assignments when all Ds come before all Us when sorting points by y coordinate, subtracting the overcount where all Ls come before all Rs in x coordinate at the same time all Ds come before all Us in y coordinate.

The only remaining case is when there exist an L with higher x coordinate than the the R with lowest x coordinate, and similarly for D and U. Let the rightmost L be L', the leftmost R be R', and similarly define U' and D'.

It can be shown that these 4 points must form a spiral-like shape (with two possible orientations), as shown in the below diagram.



Furthermore, once these 4 points are determined, there is at most one possible orientation for every other UD or LR.

Additionally, these four points dictate 9 regions of the plane, where for 4 rectangular regions, only UDs are allowed or only LRs are allowed, a middle rectangular region, where no points are allowed, and 4 corner regions.

For the corner regions, both LRs and UDs are allowed, and their orientations are fixed. For example, in the bottom right region in the drawn diagram (which corresponds to the leftmost R), there are Ds and Rs allowed. We can use a stack-based sweep line to check whether no Ds or Rs intersect in $O(N)$ time (after sorting).

Polynomial solutions with lower time complexity were intended to pass $N \leq 100$.

Bonus: Solve the problem in $O(N \log N)$ time.

5 Inverse Knapsack

Problem Idea: Jiang Cheng
Problem Preparation: Jiang Cheng
Analysis By: Richard Qi

For every prime $p \leq 53$, consider the terms $(p \cdot 2^k)^{-1}$ for $k \leq 7$. Using subsets these terms only, we can get $\frac{a_p}{p \cdot 128}$ for $0 \leq a_p < 128$. Observe that $2 \times 3 \times \cdots \times 53 \geq 10^{18} \geq p$.

So, we reduce to finding values a_p for each prime $p \leq 53$ such that

$$\frac{a_2}{2 \cdot 128} + \frac{a_3}{3 \cdot 128} + \cdots = \text{target} \pmod{p}.$$

Multiplying by 128 on each side, we now need to find $a_2, a_3, ...$ such that

$$\frac{a_2}{2} + \frac{a_3}{3} + \cdots = 128 \cdot \text{target} \pmod{p}$$

for some new target that is 128 times the original.

Claim 5.1. *For any $0 \leq x < 2 \times 3 \times \cdots \times 53$, you can get $c + \frac{x}{2 \times 3 \times \cdots \times 53}$ for some integer c on the LHS with $c \leq 16$ by setting each $0 \leq a_p < p$.*

Proof. Proof: Let $D = 2 \times 3 \times \cdots \times 53$. We look for

$$\frac{a_2}{2} + \frac{a_3}{3} + \cdots + \frac{a_{53}}{53} = c + \frac{x}{D}.$$

Multiplying by the D on both sides and then taking modulo D, we have $\sum_p a_p \cdot (D/p) \equiv x \pmod{D}$. If we take modulo 2 on both sides, every term on the LHS except for the first one is equal to 0 (because D/p has a factor of 2 in it), so we have $a_2 \cdot (D/2) \equiv x \pmod{2}$, and can solve for a_2. We can do this for each prime, eventually getting the values of all a_i. Then, Chinese Remainder Theorem tells us that the values must match up modulo D. To see that $c \leq 16$, each term of the sum is less than 1, and there are 16 terms. □

This allows us to construct $c + \frac{x}{D}$ for an x of our choice, but we still need to be able to clear out any variation in c. However, note that a_p can be increased by 2 up to 63 more times before exceeding the limit $a_p < 128$. So, we can add 2 to the numerator of a_p until we obtain $16 + \frac{x}{D}$ for any x of our choice.

So, we can solve $16 + x/D \equiv \text{target} \pmod{p}$, and construct the necessary a_i accordingly. Note here it is necessary for $2 \times 3 \times \cdots \times 53 \geq p$, or otherwise some values of x will be too large to obtain. Then we can write out the values of a_i in binary to obtain which $(p \cdot 2^{-k})^{-1}$ terms to take. Since we only ever take from a subset of $16 \cdot 7 = 112$ numbers we will obviously always use at most $S = 150$ of them.

The final runtime is $O(T \cdot (\#\text{ primes to multiply} \geq p)^2)$.