The 2020 ICPC Asia Macau Regional Contest Editorial

Prepared by Zhejiang University

May 29, 2021

A. Accelerator

Task

We are given n accelerators $a_1, a_2, \ldots, a_n$. The final velocity is

$$((\ldots((a_1 + 1)\, a_2 + 1) \cdots + 1)\, a_{n-1} + 1)\, a_n$$

We are asked to calculate the expected final velocity when the order sequence $a_1, a_2, \ldots, a_n$ is randomly shuffled.

A. Accelerator

Observation

$$((\dots((a_1+1)\,a_2+1)\cdots+1)\,a_{n-1}+1)\,a_n$$
$$=a_1a_2a_3\dots a_n+a_2a_3\dots a_n+\cdots+a_{n-1}a_n+a_n$$

- Assume we pick k unordered items $b_1, b_2, \dots, b_k$ from the sequence a.
- They can form $k!$ suffixes of the shuffled sequence, the left $n-k$ items can form $(n-k)!$ prefixes.
- So the contribution to the answer is $(b_1b_2\dots b_k)\times\left(\frac{k!\times(n-k)!}{n!}\right)$.
- The sum of the left part $(b_1b_2\dots b_k)$ is $[x^k]\prod_{i=1}^{n}(1+a_ix)$, which can be computed using divide & conquer and FFT in $O(n\log^2 n)$.

B. Boring Problem

Task

We are given n strings $T_1, T_2, \ldots, T_n$ of length m and a string R. For each i $(1 \leq i \leq |R|)$, we are required to compute the expected length of the final string when it contains any string in T as substring if we repeat appending the i-th $(1 \leq i \leq k)$ lowercase letter with with probability p_i to the end of the current string, here the initial string is $R[1..i]$.

B. Boring Problem

- W.L.O.G, we can assume $T_1, T_2, \ldots, T_n$ are pairwise distinct.
- Insert $T_1, T_2, \ldots, T_n$ into Trie T', build the Aho-Corasick Automaton G on T', there will be $O(nm)$ vertices.
- Let's denote $g(i, j)$ as the vertex we will go to when we receive letter j at vertex i in G.
- Let's denote $E(x)$ as the expected number of random steps to reach any leaf from the x-th vertex in G.
  - If vertex x is a leaf in T', $E(x) = 0$.
  - Otherwise $E(x) = 1 + \sum_{1 \le j \le k} E(g(x, j))p_j$.
- If we pre-compute all the values of $E(\cdot)$, we can find the answer in $O(|R|)$ time by walking on G.
- There is a straightforward approach to compute $E(\cdot)$ using Gauss Elimination in $O(n^3 m^3)$, which is unfortunately too slow to pass.

B. Boring Problem

- Consider all the vertices according to their depth in T', from root to the leaves.
- Let's look at a vertex x in T', which is not a leaf.
  - If $dep(g(x,j)) > dep(x)$, $g(x,j)$ must be a child of x in T'.
  - Otherwise $dep(g(x,j)) \leq dep(x)$, so $g(x,j)$ will be considered before x.

Idea

Assume x has t $(t \geq 1)$ children $g(x,c_1), g(x,c_2), \dots, g(x,c_t)$ in T', we can know $E(g(x,c_1))$ if we know $E(x)$ and all other values of $E(g(x,j))$ according to

$$E(g(x,c_1)) = \frac{E(x) - 1 - \sum_{j, j \neq c_1} E(g(x,j)) p_j}{p_{c_1}}$$

B. Boring Problem

Observation

If we treat $E(g(x, c_2)), E(g(x, c_3)), \ldots, E(g(x, c_t))$ as unknown variables, we can know $E(g(x, c_1))$ because $dep(g(x, j)) \leq dep(x) < dep(g(x, c_1))$, all other vertices are considered before.

- The value of the root $E(root)$ should also be treated as unknown.
- Equations on leaves (i.e. $E(x) = 0$) are unused.
- Hence we will get n unknowns and n equations, run Gauss Elimination in $O(n^3)$, then we can get all the values of $E(\cdot)$.
- Overall time complexity is $O(n^3 + n^2 mk + |R|)$.

C. Club Assignment

Task

We are given n positive integers $w_1, w_2, \ldots, w_n$, divide them into two groups such that

$$\min_{1 \le i < j \le n,\ \texttt{i and j are in the same group}} \{w_i \oplus w_j\}$$

is maximized.

C. Club Assignment

- Build a graph G with n vertices $1, 2, \ldots, n$, the weight of the edge between i and j is $w_i \oplus w_j$.
- Initially the graph contains no edges. Add all edges in non-decreasing order by weight.
- Assume here comes an edge (u, v) with weight $w_u \oplus w_v$:
  - If u and v are not connected: Link them with this edge, they should be in different groups.
  - If u and v are connected, and they should be in different groups: Ignore this edge.
  - If u and v are connected, and they should be in the same groups: Conflict arises, the current assignment is the answer.

C. Club Assignment

Lemma

Find the minimum spanning tree of G, assign adjacent vertices into different groups is optimal.

- The algorithm to find the XOR minimum spanning tree is a well-known recursive approach.
- Divide all numbers into two groups according to their highest digit, solve recursively for these groups, then add the minimal possible edge between the groups.
- Finding the minimal possible edge can be done using 01-Trie.
- Time complexity: $O(n \log^2 w)$.

D. Artifacts

Task

We are given 5 artifacts, and we are asked to calculate the expected damage.

D. Artifacts

- Read the whole statement.
- Parse the input carefully.
- Calculate the expected damage.

E. Mountain

Task

There is a mountain

$$(0,0) - (1,h_1) - (2,h_2) - \cdots - (n,h_n) - (n+1,0)$$

At each point (i,h_i), a picture is taken, which covers all the points (x,y) where $i - W \le x \le i + W$ and $h_i - H \le y \le h_i + H$.
For $k = 1, 2, \ldots, n$, keep exactly k pictures, maximize the total area of the mountain which is covered by at least one picture.

E. Mountain

Observation

The picture taken at (i, h_i) can only coincide with those pictures taken at (j, h_j) where $|i - j| \leq 2W - 1$.

- Let's denote $dp[i][j][S]$ as the maximum total covered area such that we choose to keep j pictures in the first i points, where S is a $(2W - 1)$-bit binary mask denoting the kept pictures in the previous $2W - 1$ points.
- The number of states: $O(n^2 2^{2W-1})$.
- The transition can be done in $O(1)$ if we pre-compute the expanded covered area.

F. Fixing Networks

Task

Construct a simple undirected graph with n vertices and c components, where the degree of each vertex is d.

F. Fixing Networks

- $d = 0$ or $d = 1$ are trivial.
- The cases for no solution:
  - $c(d+1) > n$: Number of vertices is not enough.
  - Both n and d are odd: The sum of degrees is not even.
- When there is a solution, we can construct $c - 1$ components by assign each as a complete graph with $d + 1$ vertices.

F. Fixing Networks

- Now we only need to handle the case $c = 1$.
- Let $k = \lfloor \frac{d}{2} \rfloor$, construct a cycle, link each vertex to the previous k vertices and the next k vertices.
- When d is odd, link each vertex to the extra vertex opposite to it on the cycle, because n is always even in this case.

G. Game on Sequence

Task

There is a sequence $A_1, A_2, \ldots, A_n$ with non-negative integers. Initially a token is at position k. Two players take turns moving the token from i to a position j on the right side such that A_j differs from A_i on at most one bit in binary representation.
We are required to perform two types of operations:

- Append an integer at the end of the sequence A.
- Predict the winner when the token is at position k.

G. Game on Sequence

- For each position k, determine the value of f_k: Can the current player win when the token is at k?
- $f_k = \texttt{OR}(\texttt{NOT } f_j)$, where $j > k$ and A_j differs from A_k on at most one bit in binary representation.
- Assume there are two positions i and j, where $i < j$ and $A_i = A_j$:
  - If $f_j = \texttt{False}$: By the definition of f_i, we can know $f_i = \texttt{True}$.
  - If $f_j = \texttt{True}$: There exists a position k such that $k > j$ and $f_k = \texttt{False}$, so $k > i$ and $f_i = \texttt{True}$.

G. Game on Sequence

Lemma

For two positions i and j, where $i < j$ and $A_i = A_j$, $f_i = \texttt{True}$ always holds.

- Assume the token is at k:
  - If k is not the rightmost position matches A_k: We can claim $f_k = \texttt{True}$.
  - Otherwise keep only the rightmost position for each value and run brute-force dynamic programming to find the answer.
- Time complexity: $O(mA \log A)$.

H. Fly Me To The Moon

Task

There is an infinity grid and n types of spacecrafts $d_1, d_2, \ldots, d_n$. Assume we are at (x, y), we can choose a type of spacecraft d_i and fly to $(x + dx, y + dy)$ $(dx, dy \geq 0)$ where $0 < dx^2 + dy^2 \leq d_i^2$. There are m broken points we can't touch. We are required to count the number of ways to reach $(1000, 1000)$ from $(0, 0)$.

H. Fly Me To The Moon

- Assume the number of ways to reach (x, y) from $(0, 0)$ is $ways(x, y)$ when there are no broken points.
- Add $(1000, 1000)$ into the set of broken points for convenience.
- Let's denote f_i as the number of ways to reach the i-th broken point from $(0, 0)$ such that we never visit any other broken point.
- $f_i = ways(x_i, y_i) - \sum_{j, j \neq i} f_j \times ways(x_i - x_j, y_i - y_j)$, where $x_j \leq x_i$ and $y_j \leq y_i$.
- If we pre-compute $ways(x, y)$ for all pairs of (x, y) $(0 \leq x, y \leq 1000)$, we can find the answer in $O(m^2)$ time.

H. Fly Me To The Moon

- Now we are going to pre-compute $ways(x, y)$.
- Ignore all broken points, assume the number of ways to reach (x, y) from $(0, 0)$ in a single step is $one(x, y)$.
- Let's denote $G(x, y) = \sum_{i \geq 0} \sum_{j \geq 0} one(i, j) x^i y^j$.
- $$ways(i, j) = [x^i y^j] \sum_{k \geq 0} G(x, y)^k = [x^i y^j] \frac{1}{1 - G(x, y)}$$
- Here $\frac{1}{1-G(x,y)}$ mod x^{1001} mod y^{1001} can be computed using FFT in $O(1000^2 \log 1000)$.

I. Nim Cheater

Task

There is a sequence of piles of stones, Bob can pay to remove some piles from the game.
Initially the sequence is empty. We are required to perform three types of operations:

- Append a new pile at the end of the sequence.
- Delete the rightmost pile.
- Find the cheapest way for Bob to cheat such that the first player of the Nim game will lose.

The memory limit is 8MB.

I. Nim Cheater

- The first player will lose iff the XOR sum of all numbers is zero.
- We need to find a subset of numbers such that the XOR sum is zero and the total cost is maximized, and remove the left part to cheat.

Observation

The modifications form a rooted tree.

- Construct the rooted tree, we need to find the answer for each vertex.
- Let's denote $dp[i][j]$ as the maximum total cost such that the XOR of chosen numbers is j if we choose numbers among the path from the root to the i-th vertex.
- Time & space complexity: $O(na)$, which is unfortunately unable to fit in the 8MB limit.

I. Nim Cheater

Idea

Try different DFS orders won't affect the answer.

- Find the heavy light decomposition of the tree.
- For each vertex, DFS its light children first, and finally DFS its heavy child, give its space to its heavy child.
- Now $dp[i][j]$ can be stored in $f[cnt][j]$, cnt denoting the number of light edges on the path from the root to the i-th vertex.

Lemma

There will be at most $O(\log n)$ light edges on a path, so the space complexity is $O(a \log n)$.

J. Jewel Grab

Task

Given a sequence of jewels, each jewel has its color and its value. We are required to perform two types of operations:

- Modify a jewel in the sequence.
- Find the maximum total value of taken jewels if we start at position s and move right, skipping at most k jewels, such that no two taken jewels share the same color.

J. Jewel Grab

- For the i-th jewel, find the first jewel pre_i with the same color on the left side of it.
- For each query, let's move right from s step by step, assume we are at j with color c_j and value v_j:
  - If $pre_j < s$, it is the first time we meet a jewel in such color, take it.
  - Otherwise $pre_j \geq s$, we should compare it with the previous one, greedy keep the one with larger value, and skip one of them.
- The second part takes place at most k times, which is acceptable for $k \leq 10$.
- To speed up the first part, we need to find the next position j such that $pre_j \geq s$, and find the sum in this range, which can be done by traveling on the segment tree in $O(\log n)$.
- Time complexity: $O(k \log n)$ per query.

K. Candy Ads

Task

Given n advertisement posters, we are required to choose some of them such that no two chosen advertisement posters will occupy the same pixel at the same time.
We are also given m extra conditions, each of which requires us to keep at least one of two listed advertisement posters.

K. Candy Ads

- For the i-th ad poster, introduce a boolean variable x_i, which is true iff we choose the i-th ad poster.
- For two pairs of ad posters (i, j), if they occupy the same pixel at the same time, we have $x_i \wedge x_j = \texttt{False}$.
- For an extra condition (u, v), we have $x_u \vee x_v = \texttt{True}$.
- We can build a graph with $2n$ vertices and $O(n^2 + m)$ edges to solve the above 2-SAT problem, which is unfortunately too slow to pass.

K. Candy Ads

- To solve the 2-SAT problem, we need to find the strongly connected components in the corresponding graph.
- We can find SCC using the Kosaraju Algorithm by running two similar DFS procedures on the graph.
- When we are visiting a vertex in DFS, we need to know the adjacent vertices, and we should skip visited vertices in a clever way.

Idea

Use Bitset to find the adjacent vertices and skip visited vertices.

- Time complexity: $O(\frac{n^2}{w} + m)$.

L. Random Permutation

Task

An integer sequence $a_1, a_2, \cdots, a_n$ is generated randomly, and the probability of being $1, 2, \cdots, n$ are all $\frac{1}{n}$ for each a_i.
We are required to calculate the expected number of permutations $p_1, p_2, \cdots, p_n$ from 1 to n such that $p_i \leq a_i$ holds for each $i = 1, 2, \cdots, n$.

L. Random Permutation

Observation

No matter what the permutation p is, the number of the corresponding sequence a is always $n!$.

- There are $n!$ permutations, and n^n possible sequences a.
- So the answer is $\frac{n!n!}{n^n}$.

Thank you!