The 2019 ICPC Asia Hong Kong Regional Contest Editorial

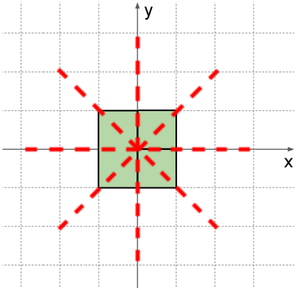
Prepared by Zhejiang University

December 1, 2019

A. Axis of Symmetry

Task

We are given n rectangles with sides parallel to the coordinate axes on the plane. The area of the intersection of any two rectangles is zero. We need to find all the axes of symmetry of the figure formed by these rectangles.



A. Axis of Symmetry

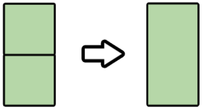
Observation

The axis can only be vertical, horizontal or one of two diagonals.

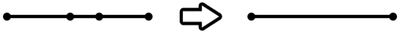
- For every possible axis, check whether it is a real axis.
- Assume no two rectangles touch each other, we can just check whether each segment on the boundary matches another segment.
- How to handle the case that multiple touched rectangles form a huge figure?

A. Axis of Symmetry

- Let's fix the boundary.
- Erase the segments belonging to two rectangles.



- Merge the connected segments to a single segment.



- After these two operations, we can just check whether each segment on the boundary matches another segment.
- Time complexity: $O(n \log n)$.

B. Binary Tree

Task

We are given a binary tree with n vertices. Two players take turns to remove a subtree T such that T is a perfect full binary tree. We are asked to determine the winner of the game.

B. Binary Tree

Observation

The number of vertices in a perfect full binary tree must be odd.

- Each move will change the parity of n.
- So when n is odd, the first player wins, otherwise the second player wins.

C. Constructing Ranches

Task

We are given a tree with n vertices. We need to count the number of simple paths such that the values of vertices on the path can form a simple polygon.

C. Constructing Ranches

Lemma

A sequence $a_1, a_2, \ldots, a_n$ can form a simple polygon if and only if:

$$\sum_{i=1}^{n} a_i > 2 \max_{i=1}^{n} \{a_i\}$$

- Using centroid decomposition, we just need to count the valid paths passing the centroid x.

C. Constructing Ranches

- Let's denote s_u as the sum of the values on the path from u to the centroid, and denote m_u as the corresponding maximum value.
- For each vertex u, we need to count the number of vertices v such that $s_u + s_v - a_x > 2\max\{m_u, m_v\}$.
- Sort u according to m_u, we can assume $m_u \geq m_v$ now, so we just need to count the number of vertices v such that s_v is greater than something, which can be done efficiently using Fenwick Tree.
- Time complexity: $O(n \log^2 n)$.

D. Defining Labels

Task

Find the x-th label in base k.

D. Defining Labels

- Very similar to number base conversion.
- Be careful of the implementation.
- Time complexity: $O(\log x)$.

E. Erasing Numbers

Task

We are given a permutation $a_1, a_2, \ldots, a_n$. Every time we can select three consecutive numbers and only keep the median until there is only one number. We are asked to report which numbers can survive in the end.

E. Erasing Numbers

- For a number x located at i, replace all the numbers a_y by 1 if $a_y > x$, and replace all the numbers a_y by 0 if $a_y < x$.
- Now we need to check whether x can survive by erasing these 01 numbers in $O(n)$ time.

E. Erasing Numbers

Lemma

If it is possible for x to survive, we can always erase the left part $[1, i-1]$ and the right part $[i+1, n]$ to 2 numbers and erase them with x in the end.

So we just need to check all the cases below:

- $0x0$ or $00x$ or $x00$
- $1x1$ or $11x$ or $x11$
- $0x1$ or $1x0$ or $01x$ or $10x$ or $x01$ or $x10$

E. Erasing Numbers

- Let's check whether a sequence can be erased to 0 or 00 for example.

Idea

Try to select 111 in each step to keep as many zeroes as possible.

- Let's consider each element from left to right and do operations greedy.
- When the last three elements are 111, erase 11.
- When the last elements are ..10..0, erase 10.

E. Erasing Numbers

- The left part is to check whether a sequence can be erased to 01 or 10.

Observation

If the occurrences of 0 and 1 are the same, we can always erase a pair of adjacent 01 to get 01 in the end.

- Note that in the previous algorithm, we are trying to maximize the number of zeroes.
- So we can run the previous algorithm until the occurrences of 0 and 1 are the same.

F. Falling Objects

Task

Report the location of each falling object.

F. Falling Objects

- Since n is small, $O(n^2)$ simulation is OK.
- Advanced 3D geometry methods are highly required.

G. Game Design

Task

For a rooted tree T, we can always block some vertices such that no leaf nodes are connected with the root. Blocking vertex i costs c_i, we need to find the optimal solution that minimizes the total cost.
Given k, we are asked to construct a tree T such that there are exactly k optimal solutions in T.

G. Game Design

- How to calculate the number of optimal solutions for a tree T?
- It can be done using dynamic programming.
- Let's denote f_x as the minimum cost for the subtree rooted at x, and denote g_x as the number of optimal solutions.
- If x is a leaf node, we have $f_x = c_x$ and $g_x = 1$.
- If x is not a leaf node, we have $f_x = \min\{c_x, \sum f_y\}$, where y is a child of x.
- $g_x = 1$ or $\prod g_u$ or $1 + \prod g_u$, where u is a child of x with the minimum value of f_u.

G. Game Design

- Now let's construct the tree for the given number k.
- If $k \le 2$, we can construct a chain easily.
- If k is even, we can construct a tree with two children, whose number of solutions are 2 and $\frac{k}{2}$.
- If k is odd, we can construct a tree with two children, whose number of solutions are 2 and $\lfloor \frac{k}{2} \rfloor$, and we can make the cost of the root equal to the optimal solution such that the number of solutions will become $1 + 2\lfloor \frac{k}{2} \rfloor = k$.
- Time complexity: $O(\log k)$.

H. Hold the Line

Task

We are asked to keep an array $h_1, h_2, \ldots, h_n$ with zeroes. Every time we will replace a zero in array h by a positive integer or be asked to find the minimum value of $|h_i - k|$ where $l \leq i \leq r$.

H. Hold the Line

- There is a straightforward $O((n+m)\log^2 n)$ on-line solution using segment tree with `std::set` on each node.
- However, the solution above is unfortunately too slow to pass the problem in such a tight constraint due to the huge constant.

H. Hold the Line

- Let's do it off-line and denote v_i as the time that h_i becomes positive.
- Solve the queries in the order of $r = 1, 2, \ldots, n$ and keep a data structure T.
- Assume we are now at r, insert r into T.
- For the i-th query, we want to find the smallest value of h_j such that $h_j \geq k$ and the largest value of h_j such that $h_j \leq k$.
- Note that j is valid if and only if $j \geq l$ and $v_j < i$.

H. Hold the Line

- Use a segment tree as T, each node $[a, b]$ of which keeps all the indices j such that $a \le h_j \le b$.
- For each query, travel on the segment tree to find the answer, we will pass by at most $O(\log n)$ nodes.
- Every time we pass by a node, we need to check whether there is a valid j such that $j \ge l$ and $v_j < i$.

Observation

If $j < k$ and $v_j > v_k$, there is no need to store j.

- For each node, keep a monotonic queue such that we can maintain the data structure in $O(n \log n)$ time.
- Now we can do a simple binary search to check whether there is a valid j.
- Time complexity: $O(n \log n + m \log^2 n)$.

I. Incoming Asteroids

Task

There are n vertices and several ICPC members with their values, each ICPC member will assign himself to at most 3 vertices. There will also be many events like "decreasing the value of all the ICPC members assigned to the x-th vertex by y". We are asked to report which ICPC members' values become non-positive in each event.

I. Incoming Asteroids

- Assume an ICPC member with value t is assigned to k vertices, set alarms to each of these vertices like "wake me up after vertex u getting $\lceil \frac{t}{k} \rceil$ points".
- For each vertex, use a heap or other data structure to keep these alarms.
- When an event like "decreasing the value of all the ICPC members assigned to the x-th vertex by y" comes, check the alarms at the x-th vertex whose target values are the minimum.
- Every time we wake up, set new alarms again until the target is reached.

I. Incoming Asteroids

Observation

Every time we wake up, the target value t will lose at least $\lceil \frac{t}{3} \rceil$.

Fact

There will be at most $O(n \log y)$ alarms.

- Time complexity: $O(n \log n \log y)$.

J. Junior Mathematician

Task

Let k be the number of digits in the decimal representation of x, and $d(x,i)$ be the i-th digit of x in its decimal representation. Then

$$f(x) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} d(x,i) \cdot d(x,j)$$

We are asked to count the number of integers x such that $x \equiv f(x) \pmod{m}$ and $L \le x \le R$.

J. Junior Mathematician

- $ans(L, R) = ans(0, R) - ans(0, L - 1)$, hence we can only consider R.
- Do dynamic programming from the highest digit to the lowest digit.

J. Junior Mathematician

- Let's denote $dp[i][j][s][0/1]$ as the number of ways to assign numbers to the highest i digits of x such that $(f(x) - x) \bmod m = j$, $\sum_t d(x, t) \bmod m = s$, where the relationship between x and R is $0 (x < R)$ or $1 (x = R)$.
- The number of states: $O(|R|m^2)$.
- The answer is $\sum_{s,t} dp[k][0][s][t]$.

State Transition

For a fixed state $dp[i][j][s][t]$, enumerate the $(i + 1)$-th digit y, we have

$$dp[i][j][s][t] \to dp[i+1][(j+y\cdot s-y\cdot 10^{k-i-1}) \bmod m][(y+s) \bmod m][t']$$

K. Key Project

Task

There are n buildings on a line from left to right. We are given two sets A and B, each of which contains m engineers. Every engineer is at one of these n buildings and we need to pay c_i dollars to hire him. If we match A_i with B_j, we need to pay $dis(A, B)$ extra dollars. Now for each $k = 1, 2, \ldots, m$, we are asked to find the optimal matching with k pairs of engineers.

K. Key Project

- Build a network as follows:
- Create source S and sink T.
- Create vertices for these n buildings, add edges between each pair of adjacent buildings with cost d_i.
- Add edges from S to the corresponding building for each engineer in A with the corresponding cost.
- Add edges from the corresponding building to T for each engineer in B with the corresponding cost.
- Unfortunately, the constraint is too tight that the standard minimum-cost-flow algorithm can't pass.

K. Key Project

Observation

We will always choose the engineer with the minimum cost in each building.

Observation

There is no need to reverse the edges linked to S or T on the augment path.

- Keep the cheapest engineer for each building.
- Find a pair of engineers A and B with the minimum cost using linear recursion.
- There will be two cases: $A \to \cdots \to B$ or $B \leftarrow \cdots \leftarrow A$.

K. Key Project

- After finding the best pair, we need to change the engineers to the next cheapest ones in the corresponding buildings.
- We also need to change the cost on the path between these two buildings like what we do in the standard flow algorithm.
- Time complexity: $O(nm)$.

Thank you!