The 19th Zhejiang Provincial Collegiate Programming Contest

April 2022

A. JB Loves Math

The answer will not exceed 3 .
Case study.

B. JB Loves Comma

Implement everything.

C. JB Wants to Earn Big Money

Implement everything.

D. The Profiteer

When the interval expands, the expected profit will only decrease.
For each l, let f_l be the minimum r such that after raising the prices of interval $[l, r]$, the expected profit is less than E.
The final answer will be $\sum_{i=1}^{n} n - f_i + 1$.
It is not hard to find out that $f_1 \leq f_2 \leq \cdots \leq f_{n-1} \leq f_n$.
Using two-pointers and 0-1 knapsack will result in time complexity $O(n^2 k)$, which cannot pass this problem.

D. The Profiteer

Use parallel binary search to calculate all f_i.
Let $solve(l, r, dl, dr, V)$ be a process to calculate the values of $f_l, f_{l+1}, \ldots, f_r$, where each value are in the interval $[dl, dr]$, and all items outside $[l, r] \cup [dl, dr]$ are already in the knapsack V.
Let $dm = \lfloor \frac{dl+dr}{2} \rfloor$.
Since $f_i \leq f_{i+1}$, we can do binary search in $[l, r]$ to find the maximum m satisfying $f_m \leq dm$.
In other words, we need to binary search for m and determine whether raising the prices of $[m, dm]$ will cause the expected profit to exceed E.

D. The Profiteer

Since V is only lacking items in $[l, r] \cup [dl, dr]$, we can add these items into the knapsack by brute force.
The naive approach needs $O((r - l + dr - dl) \log n)$ times of adding an item into the knapsack.
Notice that since the items in $[dl, dr]$ and not in $[l, r]$ have constant prices, they can be added into the knapsack prior to the binary search process.
While binary searching for m and found m was too small, then the items in $[l, m]$ will also have constant prices, and can be added into the knapsack before continuing the binary search process.
Similarly, when m is too large, items in $[m, r]$ can also be added into the knapsack before continuing the binary search process.
After optimization, the new process needs $O(r - l + dr - dl)$ times of adding an item into the knapsack.

D. The Profiteer

After calculating the maximum m such that $f_m \leq dm$, the two subproblems are $solve(l, m, dl, dm - 1, V_1)$ and $solve(m + 1, r, dm + 1, dr, V_2)$.
For each item in $[l, r] \cup [dl, dr]$, if it is not considered in some subproblem, its price will be constant in that subproblem, so we can add it into the corresponding knapsack.
solve will recurse $O(\log n)$ times, and the $[l, r]$ and $[dl, dr]$ does not intersect in each level of recursion, so the sum of $r - l + dr - dl$ is $O(n)$ in each level.
Total time complexity $O(nk \log n)$.

E. Easy Jump

Firstly, let's consider the cases where $S = 0$ or $T1 \geq T2$, Grammy does not need to use "Focus" to heal.

To minimize the expected time, the best strategy is to challenge the stages until 1 unit of hp is left, and heal by 1.

Let $g_{i,j}$ be the minimum expected success time for Grammy at stage i having j hp.

- If $j > 2$, then $g_{i,j} = 1 + p_i g_{i+1,j} + (1 - p) g_{i,j-1}$.
- If $j = 2$, then $g_{i,j} = 1 + p_i g_{i+1,j} + (1 - p)(g_{i,j} + T2)$.

Note that $g_{i,j}$ may be on both sides of some transitions, solving the equation may be needed.

E. Easy Jump

For cases where $S > 0$ and $T1 < T2$, Grammy need to use "Focus" to heal. If Grammy is located at a stage without a soul totem, the best strategy is to challenge the stage until 1 unit of hp is left, and prioritize "Focus" over "Hiveblood".

Let $f_{i,j,k}$ be the minimum expected success time for Grammy at stage i, having j mp and k hp.

- If $k > 2$, then $f_{i,j,k} = 1 + p_i f_{i+1,j,k} + (1-p) f_{i,j,k-1}$.
- If $k = 2$ and $j > 0$, then $f_{i,j,k} = 1 + p_i f_{i+1,j,k} + (1-p)(f_{i,j-1,k} + T1)$.
- If $k = 2$ and $j = 0$, then $f_{i,j,k} = 1 + p_i f_{i+1,j,k} + (1-p)(f_{i,j,k} + T2)$.

E. Easy Jump

If Grammy is located at a stage with a soul totem, she can fill up her mp at this stage for future usage.
We can enumerate the amount of mp recovered for transition.
Since mp can be recovered indefinitely in this stage, all of the dp states have full mp value.
Total time complexity $O(nH(H+S))$.

F. Easy Fix

Use Fenwick tree to preprocess the A values and B values of each position.
Without loss of generality, let's assume $u < v$, then C_u and C_v can be recalculated on each query.
For positions to the left of u or to the right of v, there C values will not change after modification.
For positions in $[u+1, v-1]$, their A values and B values will change equally. Each type of contribution can be calculated offline using Fenwick tree.
Total time complexity $O((n+m)\log n)$

G. Easy Glide

Use Dijkstra.
Total time complexity $O(n^2)$

H. A=B

Fun fact: This language is Turing Complete.
In each round, copy the two given strings to the end of the string, and check if T is a prefix of S. Delete the first character of S after each round. If S is empty, return 0.
Copying a string needs $O(|S|L)$, and there are a total of $O(|S|)$ rounds, so the total time complexity is $O(L^3)$, which cannot pass this problem.

H. A=B

Since this language can only check or modify some local information, it is hard to do random accesses. So we should not copy the strings too frequently.
This hints us to move the characters to adjacent positions before comparing, for example `abcdefSxyz` can be moved into `xaybzcdefS`

H. A=B

After moving the characters, we can compare the adjacent characters to check if T is a prefix of S.
After comparison, we can move each character of T one position to the right and delete the first character of S.
For example `xaybzcdefS` will become `xbyczdefS` after movement and deletion.
Repeat above process until some character of string T are adjacent to the last letter `S`, then return 0.
Total time complexity $O(L^2)$

I. Barbecue

Use Hash or Manacher's algorithm to detect whether the starting string is a palindrome.
After that, the string will not be palindrome before each player rips off a character.
If at some moment a string will change into a palindrome no matter deleting the first character or the last character, the operating player must lose.
It is not hard to discover that such string can only be like
$ab, abab, ababab, \ldots$.
This means the final state where a player must lose can only have even length, so the winner is determined by the parity of the length of the starting string if it is not a palindrome.
Total time complexity $O(n + q)$.

J. Frog

Without loss of generality, assume $S = (1, 0)$, and T is not below the x-axis.

Let $\overset{\frown}{ST}$ be d degrees.

If $d = 0$, then the answer is 0.

If $0 < d \leq 90$, then the answer is 2.

If $d > 90$, then if 3 steps are enough, the unit-length tangent segments through S and T must not exceed 1, solving the triangle yields $d \leq 131$.

If $d > 131$, then the frog can jump two times to $(0, 1)$ and use two extra steps to get to T.

K. Dynamic Reachability

We split operations into size-k groups and process each group in order.
At most k edges will change its color in one group, so we mark these edges as critical edges.
We also mark the vertices incident to all edges in this group (including modifications and queries) as critical vertices.

K. Dynamic Reachability

We can calculate $f_{i,j}$ as whether i can reach j using only black non-critical edges utilizing some SCC algorithm and std::bitset in $O(n + m + \frac{(n+m)k}{w})$ time.
Next, we can build a new graph G. i have an edge to j in G if and only if $f_{i,j} = True$.
After obtaining a shrinked graph G, we can add $O(k)$ black critical edges into G and run BFS to find a path from u to v.
The BFS process can also speed up to $O(\frac{k^2}{w})$ using std::bitset.

K. Dynamic Reachability

Total time complexity is
$O(\frac{q}{k}(n+m+\frac{(n+m)k}{w})+q\frac{k^2}{w}) = O(\frac{q(n+m+k^2)}{w}+\frac{q(n+m)}{k})$.
When $k = 32 = \frac{w}{2}$, the total time complexity is $O(\frac{q(n+m)}{w}+qw)$.
The actual running time will decrease when k slightly increases, because the preprocess time cost will decrease.

L. Candy Machine

Assume the final average is not greater than k, then we can choose all candies with sweet value less than k, and choose some of the smallest remaining sweet values.
It is not hard to find out that after sorting the sweet values, the final subset must be a prefix. So we can enumerate the prefixes and use two-pointers to calculate the amount of candies JB can get.
Total time complexity $O(n + sort(n))$.

M. BpbBppbpBB

Assume Grammy has used x type C stamps and y type S stamps.
The number of black cells is $146x + 100y$.
The number of holes is $2x + y$.
Solving the equation will yield x and y.
Total time complexity $O(nm)$.
Note that using a 4×4 square pattern to find a hole is wrong.

Thank you!