ACM-HK Programming Contest 2022

Contest Analysis

Zhejiang University
Mysterious Oscar and her friends

06.26.2022

A. Ramen
Description

Given an array of length n, in each operation you can fold one prefix of positive numbers without exceeding the border. The numbers on the corresponding positions will be added together. Answer whether you can fold the array to length 1.

A. Ramen

Solution

Conclusion

You can fold the array to length 1 *if and only if you can fold the numbers one by one.*

Which means $\forall i \in [1, n-1], \sum_{j=1}^{i} a_j > 0.$

A. Ramen

Solution

Proof

Assume the $x = \min\{x| \sum_{i=1}^{x} a_i \leq 0\}$, *notice that what ever you fold, the number on position* x *will always be less or equal to* $\sum_{i=1}^{x} a_i$, *so it is impossible to fold* x.

Thus we just have to calculate the prefix sum. The time complexity is $\Theta(n)$.

B. Rectangle Placement

Description

Given an $n \times m$ grid, find the number of ways to place two non-intersecting rectangles on it. The rectangles are considered intersect if their edges or corners intersect.

B. Rectangle Placement

Solution

If one rectangle contains the other, the number of ways is $\binom{n}{4}\binom{m}{4}$.
Excluding this case, two rectangles are non-intersecting if and only if they do not intersect on at least one dimension.
The number of ways that they do not intersect on X dimension is $\binom{n}{4}\binom{m}{2}^2$, which means decide their X coordinates with restriction non-intersecting, equivalent to choose four lines vertical to X axis. Then choose Y coordinates for both of them without restriction.
According to symmetry, the number of ways for another case is $\binom{m}{4}\binom{n}{2}^2$.
Noticing that the number of ways that the rectangles do not intersect on any dimension is calculated twice, which we have to subtract, is $2\binom{n}{4}\binom{m}{4}$.
So the answer is $\binom{n}{4}\binom{m}{2}^2 + \binom{m}{4}\binom{n}{2}^2 - \binom{n}{4}\binom{m}{4}$, which can be calculated in $\Theta(1)$ time.

C. Infectious Disease

Description

There are n elements, each of them belongs to one set of A, B, C. Initially A,B contains 1 element, C contains $n-2$ elements.
On each day, first there will be $\min\{|A|, |C|\}$ elements chosen from set C, being moved to set A. Then there will be $\min\{2|B|, |A|+|C|\}$ elements chosen from set A, C, being moved to set B. The elements are chosen with equal probability. Count the expected number of days that set A will become empty.

C. Infectious Disease
Solution

Let's try to solve it with dynamic programming. The expected number of days until set A become empty is only related to the size of the sets. On day i, size of B will be $\min\{3^i, n\}$, thus we just have to know the size of A, the status is determined.
Let $f_{i,j} =$ the number of expected days until the set A become empty on the i-th day, and the number of elements in set A is j.

C. Infectious Disease

Solution

We have the following transitions:

- $f_{i,j} = 1, 3^{i+1} \geq n$
- $f_{i,j} = 0, j = 0$
- $f_{i,j} = 1 + \sum_{k=0}^{2j} \frac{\binom{2j}{k}\binom{n-2j-3^i}{2\cdot 3^i - k}}{\binom{n-3^i}{2\cdot 3^i}} f_{i+1,2j-k}$, otherwise

The number of days will not exceed $O(\log n)$ and the number of elements in set A is $O(n)$, so enumerating the status and do the transition both take $O(n)$ time. The time complexity is $O(n^2 \log n)$.
The dynamic programming seems to be too slow to calculate the answer.

C. Infectious Disease

Solution

We will prove that if we only enumerate useful states, this actually works in $\Theta(n^{\log_3 4})$.

The number of days which we have to do transition will not exceed $\lceil \log_3 n \rceil - 1$, which means $1 \le i \le \lceil \log_3 n \rceil - 1$, and noticing that on day i, the number of elements in set A will not exceed 2^i, so $1 \le j \le 2^i$, so the complexity is calculated as follow:

$$T(n) = \sum_{i=1}^{\lceil \log_3 n \rceil - 1} \sum_{j=0}^{2^i} (j+1) = \Theta(n^{\log_3 4})$$

By assuming $t = \lceil \log_3 n \rceil - 1$, we have $T(n) = \frac{4^{t+1}}{3}$. For the given constraints $t \le 14$, and $T(n) \approx 89\,478\,485$, which is OK to fit the given time limit.

D. Maximum Range

Description

Given a simple connected undirected graph, find the largest range cycle with non-repetitive edges in the graph.

D. Maximum Range

Solution

Lemma

Two edges can appear in one cycle with non-repetitive edges if and only if they are in the same edge bi-connected component.

Proof

Necessity*: The bridges are not in any cycle so the cycle can not contain edges in different edge bi-connected component.*
Sufficiency*: Assume the two edges are (a, b) and (c, d), we can see that changing (a, b) to (a, x) and (x, b), changing (c, d) to (c, y) and (y, d) by adding new vertices x, y will not change the edge bi-connected component according to the definition. Thus we can see that there will be a cycle containing x, y, which contains (a, b) and (c, d).*

D. Maximum Range

Solution

Then we just have to find the largest and smallest edges in each edge bi-connected component.

Use Tarjan's algorithm to find bridges and DFS to find edge bi-connected components, the time complexity is $\Theta(n + m)$.

E. Rotate Sum 2

Description

Given an convex n-gon, choose an vertex i with equal probability, and choose an edge $(j, j+1)$ with equal probability, start rotating the polygon with $(j, j+1)$ on the X-axis, and ends when $(j, j+1)$ is landing on X-axis again. Calculate the expected area swept by the vertical line starting at i ending at X-axis.

E. Rotate Sum 2

Solution

No matter which j we choose, the area will not change.
The area equals to the summation of the area of the polygon and the area of the area of the sector swept by the line connecting the vertices i and the center of rotation.
The above conclusion can be found by clipping and translating the swept area.

E. Rotate Sum 2

Solution

Then we have to calculate the sum of square of distance between some vertex i to all other vertices. Which is $\sum_{j=1}^{n}[(x_j - x_i)^2 + (y_j - y_i)^2]$. This can be calculated by maintaining $\sum x, \sum x^2, \sum y, \sum y^2$. When calculating the angles, atan2 are prefered to acos in C++, since the floating point error may cause that when calling acos(x), x might be less than -1 or greater than 1, which have to be taken care of.
The time complexity is $\Theta(n)$.

F. Smaller LCA
Description

Given a tree, for each vertex as the root, calculate the number of unordered pairs of vertices (x, y) have a lowest common ancestor z such that $z \leq x \cdot y$

F. Smaller LCA

Solution

Notice that there are only $\Theta(n \log n)$ pairs of (x, y) that $x \cdot y \leq n$, so we calculate number of pairs (x, y) such that $z > x \cdot y$.
For all pairs of vertices (x, y), the possible LCA are the vertices on the path from x to y, decided by which subtree the root is in.
So for each vertex u on the path from x to y and $u > x \cdot y$, we add 1 to the answer of all subtrees adjacent to u excluding the subtrees containing x or y.

F. Smaller LCA
Solution

Consider the subtree of v adjacent to u, the contribution to this subtree is equal to the number of pairs (x, y) where $u > x \cdot y$, u is on the path between x, y, and $x \notin v, y \notin v$.
For each pair of x, y with $x \cdot y \leq n$, we put a tag_1 (k, a, b) with $k = x \cdot y, a = x, b = y$ to all vertices on the path from x to y. Applying tag_1 to chain (x, y) is equivalent to adding tag_2 (k, a, b) at vertex x, y and deleting tag_2 twice at vertex $LCA(x, y)$, then the tag_1 of each vertex is equal to the sum of tag_2 in its subtree.
For each vertex u and one of its subtrees v, we add the number of tag_1 with $k > u, a \notin v, b \notin v$ to the answers of each vertex in subtree of v. This can be done by DFS and Fenwick Tree.
The time complexity is $\Theta(n \log^2 n)$.

G. Positive String

Description

Given a string, find the number of substrings s such that $s > s^R$, where the strings are compared lexicographically and s^R is the reverse of string s.

G. Positive String

Solution

Assume $s[i,j]$ denotes the string $s_i s_{i+1} \ldots s_j$, and pre_i denotes the prefix $s[1,i]$, suf_i denotes the suffix $s[i,n]$.
Let's enumerate the left endpoint of the substring. Assume it's i. Then for all $j \geq i$, if pre_j^R is lexicographically smaller than suf_i, then we add 1 to answer. This can be done by Suffix Array and Fenwick Tree.

G. Positive String

Solution

However, considering the string “bxxxxa” and $i = 2, j = 4$, the suf_2 is “xxxa” and pre_j^R is “xxxb”, but this is not the answer, because when comparing, we exceeded the border.

We can observe that every such illegal pairs (i, j), the string we compared leads to some pattern $s_l P s_r$, where $s_l > s_r$ and P is a palindrome. So for all palindrome $s[i, j]$, if $s_{i-1} > s_{j+1}$, then we subtract one from the answer we calculated before, thus we can get the correct answer. This can be done with Manacher’s algorithm.

The time complexity is $\Theta(|s| \log |s|)$.

Thanks!