Petrozavodsk Summer Training Camp 2022

ZJU Contest 2 Analysis

Zhejiang University

08.30.2022

A. Mode

Description

Given an array a of length n, select a subarray and add k to the elements in it. Maximize the number of occurence of the mode, and output the possible modes.

A. Mode

Solution

First let's rewrite the problem as "select an subarray [l,r], maximize $f(\{a_l, a_{l+1}, \ldots, a_r\}) + f(\{a_1, a_2 \ldots, a_{l-1}, a_{r+1}, a_{r+2}, \ldots, a_n\})$, where $f(S)$ is the number of occurence of modes in set S".

A. Mode

Solution

Assume the mode of the selected interval is i, and the number of existance of i in the original array as cnt_i. Let's consider for some threshold S:
If $cnt_i \leq S$, then we enumerate the number of occurence of i in the interval, and use two pointers to maintain the minimum interval. Notice that the number i will be the answer for the second query only when the interval is expanding, we can also maintain answer for the second question.

A. Mode

Solution

If $cnt_i \geq S$, the number of such values is less than $\frac{n}{S}$, then we can enumerate i, and enumerate the mode outside the interval j. Replace all occurrences of i in the original array with $+1$, and replace the occurrences of j with -1, and replace other elements with 0, then we have to find the maximum sub-segment. Notice that the endings of the maximum sub-segment must be next to some -1, we can prelude the prefix sum of $+1$ and enumerate all positions of j.
The complexity is $\Theta(nS + \frac{n^2}{S})$, choose $S = \sqrt{n}$ then we solved it in $\Theta(n\sqrt{n})$.

B. Tree

Description

For two trees generated in some given way, calculate the number of pair of trees such that the leaf set of the trees T_1, T_2 satisfy that $L(T_1) \bigcup L(T_2) = \{1, 2, \ldots, n\}, L(T_1) \bigcap L(T_2) = \varnothing$. Calculate answer for all $n \in [2, N]$.

First we can know the following conclusion:

Conclusion

The number of ways that $L(T_1) = S$ and $L(T_2) = \{1, 2, \ldots, n\} \setminus S$ are same.

Proof

Let's use inclusion-exclusion principle, we now try to proof that way of $L(T_1)$ *contains* S*, and* $\{1, 2, \ldots n\} \setminus L(T_2)$ *containing* S *are same.*
Let $S = \{a_1, a_2, \ldots, a_k\}$*, the ways for* T_1 *is trivial, which equals to* $(n-1)! \cdot \frac{a_1-1}{n-1} \cdot \frac{a_2-2}{n-2} \cdot \cdots \cdot \frac{a_k-k}{n-k} = (n-k-1)! \cdot (a_1-1) \cdot (a_2-2) \cdot \cdots \cdot (a_k-k)$*.For* T_2*, let's subtract the leaf containing* $\{b_1, b_2, \ldots b_m\}$ *from* $(n-1)!$*, which also use inclusion-exclusion principle, now we are trying to proof*

$$(n-k-1)! \cdot (a_1-1) \cdot (a_2-2) \cdot \cdots \cdot (a_k-k)$$

$$= \sum_{\{b_1, b_2, \ldots, b_m\} \subset \{a_1, a_2, \ldots, a_k\}} (-1)^m (n-m-1)! \prod_{i=1}^{m} (n+1-b_i-(m+1-i)) \tag{1}$$

Proof

$$= \sum_{\{b_1,\ldots,b_m\}\subset\{a_2,\ldots,a_k\}} [(n-(m+1)-1)!(a_1-(n-m))+(n-m-1)!]$$
$$\prod_{i=1}^{m}(b_i-(n-m+i))$$
$$= (a_1-1)\sum_{\{b_1,\ldots,b_m\}\subset\{a_2,\ldots,a_k\}}((n-1)-m-1)!\prod_{i=1}^{m}(b_i-(n-m+i)) \quad (2)$$

Proof

$$= (a_1 - 1) \sum_{\{b_1,\ldots,b_m\} \subset \{a_2-1,\ldots,a_k-1\}} ((n-1) - m - 1)!$$

$$\prod_{i=1}^{m} (b_i - ((n-1) - m + i))$$

$$= (a_1 - 1)((a_2 - 1) - 1) \sum_{\{b_1,\ldots,b_m\} \subset \{a_3-2,\ldots,a_k-2\}} ((n-2) - m - 1)! \quad (3)$$

$$\prod_{i=1}^{m} (b_i - ((n-2) - m + i))$$

$$\ldots$$

$$= (a_1 - 1)(a_2 - 2) \cdots (a_k - k)[(n - k - 1)!]$$

B. Tree

Solution

Then let's rewrite the problem as, we have two trees T_1, T_2 where they are both generated by method 1, and the number of ways that they have the same leaf set.

Use dynamic programming. Let $f_{i,j,k}$ be the number of ways that we considered i vertices, and in T_1 there are j undecided sons, and in T_2 there are k undecided sons.

We have two transitions:

1. vertex i is leaf:
   $f_{i,j-1,k-1} \leftarrow f_{i-1.j,k} \cdot j \cdot k$
2. vertex i is not leaf, and they have p, q sons in two trees:
   $f_{i,p+j-1,q+k-1} \leftarrow f_{i-1,j,k} \cdot j \cdot k \cdot \frac{1}{p!} \cdot \frac{1}{q!}$

B. Tree

Solution

Let $F_i(x, y)$ be the generating function of f_i. Then the transition is:
$F_i(x, y) = \frac{\partial}{\partial x}\frac{\partial}{\partial y}F_{i-1}(x, y) + [\frac{\partial}{\partial x}\frac{\partial}{\partial y}F_{i-1}(x, y)] \cdot (e^x - 1) \cdot (e^y - 1) =$
$\frac{\partial}{\partial x}\frac{\partial}{\partial y}F_{i-1}(x, y) \cdot (e^{xy} - e^x - e^y + 2)$
Rewrite $F_i(x, y) = \sum a_{i,p,q} \cdot (e^x)^p \cdot (e^y)^q$, then
$\frac{\partial}{\partial x}\frac{\partial}{\partial y}(e^x)^p \cdot (e^y)^q = p \cdot q \cdot (e^x)^p \cdot (e^y)^q$, we have
$a_{i,p,q} = a_{i-1,p-1,q-1} \cdot (p-1) \cdot (q-1) - a_{i-1,p-1,q} \cdot (p-1) \cdot (q) -$
$a_{i-1,p,q-1} \cdot (p) \cdot (q-1) + 2 \cdot a_{i-1,p,q} \cdot (p) \cdot (q)$.
The answer is $F_n(0, 0) = \sum a_{n,p,q} \cdot (e^0)^p \cdot (e^0)^q = \sum a_{n,p,q}$.
You can also use some inclusion-exclusion principle method to get similar DP function.
The time complexity is $\Theta(N^3)$.

C. 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.

Conclusion

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

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

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 than 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 $\Theta(n)$.

D. 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.

D. 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 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.

D. 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 $\Theta(n)$.

E. 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 $z \leq x \cdot y$

E. 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.

E. 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 $\Theta(n \log^2 n)$.

F. Noodle

Description

Given a sequence of length n, where n is even, for some position x, answer q queries, each query asks for g_x^k, where:

$$g_x^0 = a_x$$

$$g_x^k = \frac{1}{2} \times (g_{\lceil \frac{x}{2} \rceil}^{k-1} + g_{n-\lceil \frac{x}{2} \rceil+1}^{k-1})$$

F. Noodle

Solution

For convenience, we change the transform into:

$$g_x^0 = a_x$$

$$g_x^k = g_{\lceil \frac{x}{2} \rceil}^{k-1} + g_{n-\lceil \frac{x}{2} \rceil+1}^{k-1}$$

And for each query we answer $\frac{g_x^k}{2^k}$

F. Noodle

Solution

Let's say, $t = \max\{x | n \equiv 0 \pmod{2^x}\}$, and see what happens when we do $1, 2, \ldots$ oprations to the sequence.

After one operation, the sequence looks like $a, a, b, b, \ldots, x, x, y, y, z, z$。

After two operations, $a, a, a, a, b, b, b, b, \ldots, x, x, x, x, y, y, y, y, z, z, z, z$。

...

After t operations, $\underbrace{a, a, \ldots, a}_{2^t}, \underbrace{b, b, \ldots, b}_{2^t}, \ldots, \underbrace{z, z, \ldots, z}_{2^t}$。

After $t+1$ operations, $\underbrace{a, a, \ldots, a}_{2^{t+1}}, \underbrace{b, b, \ldots, b}_{2^{t+1}}, \ldots, \underbrace{z, z \ldots, z}_{2^t}$

The last segment only have half of the length of other segments.

Then whatever how many operations are performed, the sequence looks like this pattern, let's try to maintain it.

F. Noodle

Solution

Assume the pattern is $c_1, c_2, \ldots, c_m$, let $mid = \lceil \frac{m}{2} \rceil$.
After one operation the pattern is
$c_1 + c_m, c_1 + c_{m-1}, c_2 + c_{m-1}, c_2 + c_{m-2}, \ldots, c_{mid-1}, c_{mid}, 2c_{mid}$。
We see that one of the two ends are shrinking one position inward at a time. Consider the difference sequence, let $d_i = c_{i+1} - c_i$。
The difference sequence is $d_1, d_2, d_3, \ldots, d_{m-1}$。
And after one operation it becomes $-d_{m-1}, d_1, -d_{m-2}, d_2, \ldots, d_{mid-1}$.
Rewrite the difference sequence as
$d_1, d_2, d_3, \ldots d_{m-1}, -d_{m-1}, -d_{m-2}, \ldots, -d_2, -d_1$, then we can see one operation permutes the extended difference sequence.

F. Noodle

Solution

Consider how to calculate a_x through difference sequence, Assume the sum of the initial sequence is s, after k operations the difference now is p, Then $a_1 = \frac{s \times 2^k - \sum_{i=1}^{m-1} p_i c_i}{n}$, where c_i is some constant that p_i contributes to $s.a_x = a_1 + \sum_{i=1}^{x-1} p_i$. We can rewrite the form that $a_x = t \cdot s + \sum_{i=1}^{m-1} p_i g_i$, where g is some contstant array.

Now we can see that, for some permutation f and some constant array g, we are calculating $\sum_{i=1}^{m-1} f_i g_i$, $\sum_{i=1}^{m-1} (f^2)_i g_i$, $\sum_{i=1}^{m-1} (f^3)_i g_i \ldots$.

Let's take all cycles in the permutation, and the corresponding positions in g, if we permute them as the order they are on the cycle, and reverse it, we are calculating circular convolution, which can be solved with FFT.

F. Noodle

Solution

Consider that the number of different length of cycles is $O(\sqrt{n})$, we can prelude their sums and answer each query in $O(\sqrt{n})$, which is not enough. However, we have some further conclusion.

Conclusion

All the length of the cycles are divisors of the maximum cycle, if the permutation have length n, the largest length of the cycle equals to the order of 2 under the multiplicative group of integers modulo $n + 1$.

Proof

Assume the permutation length is $2t$, present the index as $i = k \cdot (t+1) + r (0 \le k < 2, 0 \le r \le t)$, then after one operation the index becomes $j = r \cdot 2 + k$, we can see that $j \equiv 2i \pmod{2t+1}$, so the largest length of the cycle equals to the order of 2 under the multiplicative group of integers modulo $2t+1$, and each number is in the cycle which length is a divisor of the largest one.

Hence we can precalculate the power of 2^{-1}, and power of 2^{-B}, and put the answer of cycles with same length all together and add to the largest one, we can answer each query in (1) times. The total complexity is $\Theta(n \log n + q)$.

G. Geometry

Description

Given a special two-dimensional coordinate system: the angle between the positive half-axis of the X-axis and the positive half-axis of the Y-axis is 60 degrees.

Consider the following graph. The vertices are all integer coordinates (x, y) such that at least one of x, y is odd and $-2a+1 \leq x \leq 2a-1$, $-2b+1 \leq y \leq 2b-1$, $-2c+1 \leq x+y \leq 2c-1$. The edges from (x, y) go to $(x, y+1)$, $(x, y-1)$, $(x+1, y)$, $(x-1, y)$, $(x+1, y-1)$, and $(x-1, y+1)$.

Find the size of the maximum independent set of vertices in this graph. Additionally, find the number of such sets modulo $998\,244\,353$.

G. Geometry

Solution

Without loss of generality, assume $a \geq b \geq c$. Notice that when $a > b + c$ the set of points are the same with $a = b + c$, so we can change a into $\min(a, b + c)$.

Add edges to all pairs of points that cannot be colored, we will get the line graph of hexagonal grid graph. The maximum independent set has a bijection to the maximum matching of the original graph. So the problem changes into calculating the maximum matching of the hexagonal grid graph.

The dual graph of hexagonal grid is triangular grid. The matching of original graph has a bijection to the face matching of the dual graph. So the problem changes into calculating the maximum face matching of the triangular grid graph.

G. Geometry

Solution

For each maximum face matching of the triangular grid, we can erase the lines between each pair of matched faces and form a rhombus tiling of the grid. The rhombus tiling can also be observed as the isometric view of many unit cubes placed at a corner.

The problem now changes into placing multiple unit cubes at a corner such that the maximum coordinates are less than or equal to $(x = a + b - c, y = c + a - b, z = b + c - a)$ and the number of unit cubes are monotonic in three directions.

Applying Lindstrom-Gessel-Viennot Lemma will be able to calculate the number of maximum independent sets in $O(\min(n, m, k)^3)$.

We can also calculate the size of the maximum independent set by this observation.

If it is still hard to observe the non-intersecting paths, we can see the cube pile from positive direction of z-axis. After marking the height of each position, we can obtain a x by y matrix with each element in $[0, z]$ and the values are monotonic.
For each $i \in [1, z]$ draw the boundary between the numbers $< i$ and the numbers $\geq i$ and translate the i-th boundary along the vector $(i, -i)$.
This will create the bijection between unit cube placement and non-intersecting paths.

$$
\begin{aligned}
\#\text{maxIS} &= \begin{vmatrix}
C_{x+y}^{x} & C_{x+y}^{x+1} & \cdots & C_{x+y}^{x+z-1} \\
C_{x+y}^{x-1} & C_{x+y}^{x} & \cdots & C_{x+y}^{x+z-2} \\
\vdots & \vdots & \ddots & \vdots \\
C_{x+y}^{x-z+2} & C_{x+y}^{x-z+3} & \cdots & C_{x+y}^{x+1} \\
C_{x+y}^{x-z+1} & C_{x+y}^{x-z+2} & \cdots & C_{x+y}^{x}
\end{vmatrix} \\
&= \begin{vmatrix}
C_{x+y+1}^{x} & C_{x+y+1}^{x+1} & \cdots & C_{x+y+1}^{x+z-1} \\
C_{x+y+1}^{x-1} & C_{x+y+1}^{x} & \cdots & C_{x+y+1}^{x+z-2} \\
\vdots & \vdots & \ddots & \vdots \\
C_{x+y+1}^{x-z+2} & C_{x+y+1}^{x-z+3} & \cdots & C_{x+y+1}^{x+1} \\
C_{x+y}^{x-z+1} & C_{x+y}^{x-z+2} & \cdots & C_{x+y}^{x}
\end{vmatrix} \\
&= \cdots
\end{aligned}
$$

$$= \begin{vmatrix} C_{x+y+z-1}^{x} & C_{x+y+z-1}^{x+1} & \cdots & C_{x+y+z-1}^{x+z-1} \\ C_{x+y+z-2}^{x-1} & C_{x+y+z-2}^{x} & \cdots & C_{x+y+z-2}^{x+z-2} \\ \vdots & \vdots & \ddots & \vdots \\ C_{x+y+1}^{x-z+2} & C_{x+y+1}^{x-z+3} & \cdots & C_{x+y+1}^{x+1} \\ C_{x+y}^{x-z+1} & C_{x+y}^{x-z+2} & \cdots & C_{x+y}^{x} \end{vmatrix}$$

$$= \prod_{i=0}^{z-1}(x+y+i)! \times \begin{vmatrix} \frac{1}{x!(y+z-1)!} & \frac{1}{(x+1)!(y+z-2)!} & \cdots & \frac{1}{(x+z-1)!y!} \\ \frac{1}{(x-1)!(y+z-1)!} & \frac{1}{x!(y+z-2)!} & \cdots & \frac{1}{(x+z-2)!y!} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{(x-z+2)!(y+z-1)!} & \frac{1}{(x-z+3)!(y+z-2)!} & \cdots & \frac{1}{(x+1)!y!} \\ \frac{1}{(x-z+1)!(y+z-1)!} & \frac{1}{(x-z+2)!(y+z-2)!} & \cdots & \frac{1}{x!y!} \end{vmatrix}$$

$$= \frac{\prod_{i=0}^{z-1}(x+y+i)!}{\prod_{i=0}^{z-1}(y+i)!} \times \begin{vmatrix} \frac{1}{x!} & \frac{1}{(x+1)!} & \cdots & \frac{1}{(x+z-1)!} \\ \frac{1}{(x-1)!} & \frac{1}{x!} & \cdots & \frac{1}{(x+z-2)!} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{(x-z+2)!} & \frac{1}{(x-z+3)!} & \cdots & \frac{1}{(x+1)!} \\ \frac{1}{(x-z+1)!} & \frac{1}{(x-z+2)!} & \cdots & \frac{1}{x!} \end{vmatrix}$$

$$= \frac{\prod_{i=0}^{z-1}(x+y+i)!}{\prod_{i=0}^{z-1}(y+i)!} \times \begin{vmatrix} \frac{z-1}{(x+1)!} & \frac{1}{(x+1)!} & \cdots & \frac{1}{(x+z-1)!} \\ \frac{z-2}{x!} & \frac{1}{x!} & \cdots & \frac{1}{(x+z-2)!} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{(x-z+3)!} & \frac{1}{(x-z+3)!} & \cdots & \frac{1}{(x+1)!} \\ 0 & \frac{1}{(x-z+2)!} & \cdots & \frac{1}{x!} \end{vmatrix}$$

$$= \cdots$$

$$= \frac{\prod_{i=0}^{z-1}(x+y+i)!}{\prod_{i=0}^{z-1}(y+i)!} \times \begin{vmatrix} \frac{z-1}{(x+1)!} & \frac{z-1}{(x+2)!} & \cdots & \frac{z-1}{(x+z-1)!} & \frac{1}{(x+z-1)!} \\ \frac{z-2}{x!} & \frac{z-2}{(x+1)!} & \cdots & \frac{z-2}{(x+z-2)!} & \frac{1}{(x+z-2)!} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{(x-z+3)!} & \frac{1}{(x-z+4)!} & \cdots & \frac{1}{(x+1)!} & \frac{1}{(x+1)!} \\ 0 & 0 & \cdots & 0 & \frac{1}{x!} \end{vmatrix}$$

$$= \frac{\prod_{i=0}^{z-1}(x+y+i)!}{\prod_{i=0}^{z-1}(y+i)!} \times \frac{(z-1)!}{x!} \times \begin{vmatrix} \frac{1}{(x+1)!} & \frac{1}{x!} & \cdots & \frac{1}{(x+z-1)!} \\ \frac{1}{x!} & \frac{1}{(x-1)!} & \cdots & \frac{1}{(x+z-2)!} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{(x-z+3)!} & \frac{1}{(x-z+4)!} & \cdots & \frac{1}{(x+1)!} \end{vmatrix}$$

$$= \cdots$$

$$= \frac{\prod_{i=0}^{z-1}(x+y+i)!}{\prod_{i=0}^{z-1}(y+i)!} \times \frac{(z-1)!}{x!} \times \frac{(z-2)!}{(x+1)!} \times \cdots \times \frac{0!}{(x+z-1)!}$$
$$= \frac{f(x+y+z) \times f(x) \times f(y) \times f(z)}{f(x+y) \times f(y+z) \times f(z+x)}$$

where $f(x) = \prod_{i=0}^{x-1} i!$。

H. 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.

H. 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.

I. 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.

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 to determine the status.
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.

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.

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 $1 \leq i \leq \lceil \log_3 n \rceil - 1$, and noticing that on day i, the number of elements in set A will not exceed 2^i, so $1 \leq j \leq 2^i$, 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 $t \leq 14$, and $T(n) \approx 89\,478\,485$, which is OK to fit the given time limit.

J. 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.

J. Positive String

Solution

Assume $s[i, j]$ denotes the string $s_i s_{i+1} \dots s_j$, 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.

J. Positive String

Solution

However, considering the string "bxxxa" 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 $\Theta(|s| \log |s|)$.

K. DFS

Description

Given a rooted tree, each vertex have some value a_x, for all x and y in the subtree of x, calculate $\sum f(x, y)$, where $f(x, y) =$ the expectation of minimum a_p visited when using DFS to find y in the subtree of x, visiting arbitrary son with equal probability one by one.

Consider some trivial tree DP, let $f_{i,j}$ be the sum of probability that visiting some vertex in the subtree of i, the minimum value is j. Assume mn_v is the minimum value in subtree of v, rk_s is the rank of mn_s in sons of i.
We have transitions:

$$f_{i,j} \leftarrow f_{v,j} \times \frac{1}{(\sum_{s \in son_x} [mn_s \leq j]) - [mn_v < j]}$$

$$f_{i,mn_s} \leftarrow f_{v,j} \times \frac{1}{rk_s \cdot (rk_s + 1)}, mn_s \leq j \ \ \& \ \ s \neq j$$

K. DFS

Solution1

We can see that the second dimension is divided into several segments by the minimum value of the subtree, and the transition of each segment is the same.
Thus we use segment tree to maintain the DP values and when doing the transition, split it into several trees and multiply them and then merge.
The complexity is $\Theta(n \log n)$.
Noticing the fact that $\sum \min(deg_i, sz_j)$ is $O(n \log \log n)$ and not doing the split operation, but use big small trick which leads to a solution with $O(n \log n \log \log n)$ complexity with small constant, which can also pass.

K. DFS

Solution2

Consider enumerating i, and calculate the sum of probability that we go from all vertices, ending at some vertices, and the minimum value is $\geq i$.
The process is like every time we paint one vertex in the tree, maintain for all legal pairs (x, y), the sum of probability not visiting painted vertices $p(x, y)$.
Painting one vertex u have two types of effects:

1. All (x, y) can not pass u
2. When reaching some vertex p, if u and y are both in subtree of p, and their LCA is p, then we can not go into subtree of u.

K. DFS

Solution2

Consider how to maintain two types of effects:

1. painting some vertex u, for all ancestor x of u and y in subtree of u, set $p(x, y)$ to 0.
2. Let x be ancester of p, y is in subtree of p. When p have s painted sons, If y is not in subtree of some painted vertex, The coefficient to $p(x, y)$ is $\frac{1}{s+1}$, If y is in some subtree of a painted son, The coefficient is $\frac{1}{s}$. We can see every vertex will affect its ancestors only when it is painted. Then we can just change the coefficients on the path from it to its nearest painted ancestor one by one, and maintain $p(x, y)$.

We can see the transition can be written into matrix multiplication. Thus we can use the idea of splay to decompose the tree and maintain the transition with balanced trees(you can find some details here), the complexity is $\Theta(n \log n)$.

Thanks!