牛客暑期ACM多校训练营

第 3 场-eddy1021

牛客网
NOWCODER

A-PACM Team

Main Idea: Dynamic Programming(Knapsack DP), backtracking

A-PACM Team(cont.)

Main Idea: Dynamic Programming(Knapsack DP), backtracking

DP state: dp[i][p][a][c][m]:=most knowledge points from some subset of first i groups with no more than p physics experts, a algorithm experts, etc.

DP Transition: dp[i][p][a][c][m]=
max(dp[i][p][a][c][m], dp[i-1][p-p_i][a-a_i][c-c_i][m-m_i]+g_i)

Overall Time complexity: O((36)^5)
Overall Space complexity: O((36)^5)
May need to be careful about the memory limit
(Using short or char)

A-PACM Team(cont.)

It can’t be reduced to dp[P][A][C][M] and update in place, which will make backtracking broken.

A-PACM Team(cont.)

Alternative solution: Meet in middle

Brute force all possible combination of first and last half of the groups, then try to combine the result.

Overall Time complexity: O(18*2^{18} + {36}^4)
Overall Space complexity: O(2^18 + 36^4)

The main challenge is that the constant factor in space complexity would be too large since we need to reconstruct a solution.

B-Expected Number of Nodes

Main Idea: Math, Dynamic Programming, Tree traversal, Case analysis

B-Expected Number of Nodes

Main Idea: Math, Dynamic Programming, Tree traversal, Case analysis

The expected number of nodes can be separated as sum of probability of each node remained after contracting.

Thus, we turn the problem into computing the probability of each node remained.

B-Expected Number of Nodes

Main Idea: Math, Dynamic Programming, Tree traversal, Case analysis

Considering following cases:

- A node with degree<=2: It will remain after contracting if and only if it's chosen.
- A node with degree>2: It **won't** remain after contracting if and only if all the K chosen nodes are in one or two subtree of it(when considering this node as root).

B-Expected Number of Nodes

Main Idea: Math, Dynamic Programming, Tree traversal, Case analysis

Considering following cases:

- A node with degree<=2: The probability it will remain will be (N-1, K-1)/(N, K).
- A node with degree>2: Let the sizes of its subtrees(considering it as root) be [s_1, s_2, …, s_m].
  The probabilit it will remain will be
  1-\sum_{i<j}(s_i+s_j, k)+(m-2) \sum_i (s_i, k)

B-Expected Number of Nodes

Main Idea: Math, Dynamic Programming, Tree traversal, Case analysis

Then, for each K, we can compute the first cases easily.

For second case, we just need to preprocess the parameter of each (*, k) terms. Then, we can compute the answer for each k in O(N).

Overall Time complexity: O(N^2)
Overall Space complexity: O(N)

C-Shuffle Cards

Main Idea: Data structure, Implementation

C-Shuffle Cards(cont.)

Main Idea: Data structure, Implementation

We need to keep cutting out some consecutive number and put them in front of the rest list.

Since we need to quick find (p_i)-th element, it would be too slow with linked-list.

C-Shuffle Cards(cont.)

Main Idea: Data structure, Implementation

The solution is just maintaining the list with treap(or any other balanced binary tree).

Overall Time complexity: O((N+M) \lg N)
Overall Space complexity: O(N)

D-Encrypted String Matching

Main Idea: FFT, NTT

D-Encrypted String Matching

Main Idea: FFT, NTT

Since decryption is done in normal way, it's equivalent to compare a substring decrypted in broken way and the given string decrypted in normal way.
That is, comparing E(T') and E(S).

Thus, we can decypt S in normal way directly. Then, it's equivalent to ask whether for each character of E(T') and E(S), the ascii code differs by at most one.

D-Encrypted String Matching

Main Idea: FFT, NTT

Let E(T') be X, E(S) be Y, where |X|=|Y|.

It's equivalent to check whether:

$$\sum_{i=0..|X|-1} (X_i-Y_i)^2 * ((X_i-Y_i)-1)^2 = 0$$

After expansion, the above formula can be represented as:

$$+1 * (X \dot X \dot X \dot X)+$$
$$- 4 * (X \dot X \dot X \dot Y)+$$
$$+6 * (X \dot X \dot Y \dot Y)+$$
$$- 4 * (X \dot Y \dot Y \dot Y)+$$
$$(Y \dot Y \dot Y \dot Y) - (X \dot X) +$$
$$+2 * (X \dot Y) - (Y \dot Y).$$

D-Encrypted String Matching

Main Idea: FFT, NTT

The inner product of each term can be computed by FFT or NTT after reversing E(S). Thus, we can compute all the terms in constant number of FFT or NTT.

Overall Time complexity: O((|S|+|T|) \lg (|S|+|T|))
Overall Space complexity: O(|S|+|T|)

E-Sort String

Main Idea: String, KMP, string cycle, hash

E-Sort String(cont.)

Main Idea: String, KMP, string cycle, hash

Solution 1: We can directly compute hash value of each S_i and group them by has values.
Note that the hash space should be big enough to avoid hash collision.

Overal time complexity: O(|S|)
Overal space complexity: O(|S|)

E-Sort String(cont.)

Main Idea: String, KMP, string cycle, hash

Solution 2: Notice that if S_i = S_j for some i<j, S is cycle by j-i. Thus, we just need to find the minimum cycle k. Then, S_i will be equal to S_{i mod k}.

Finding cycle of string can be done by hash or KMP.

Overal time complexity: O(|S|)
Overal space complexity: O(|S|)

F-Sum Of Digit

Main Idea: Math, data structure

F-Sum Of Digit(cont.)

Main Idea: Math, data structure

Notice that SOD(v) will be

- 0 if v is 0
- 15 if v mod 15 == 0
- v mod 15 otherwise

You can prove it by expanding v as \sum v_i*(16^i).

F-Sum Of Digit(cont.)

Main Idea: Math, data structure

Let maintain a segmet tree. For each interval, storing the number of value module 15. **Note that you sure be careful about 0 and 15.**

When merging informations from two subtree, just combine them as some convolution.

Overall time complexity: O((N+Q) \lg N 16^2)
Overall space complexity: O(16 N)

G-Coloring Tree

Main Idea: Math, BFS

G-Coloring Tree(cont.)

Main Idea: Math, BFS

If there is a oracle F answering the number of way to color the tree such that the colorness will be larger than or equal to D.

Then the answer will be F(D)-F(D+1)

G-Coloring Tree(cont.)

Main Idea: Math, BFS

Let construct such oracle F.

Colorness should be larger than or equal to D, which is equivalent to that there is no two nodes colored in the same color within distance D.

Let try to color the tree and keep the above constraint.

G-Coloring Tree(cont.)

Main Idea: Math, BFS

Let color the nodes in BFS order.

When coloring a node v, find number of nodes already colored and within distance D of v. Denote it as X.

Note that for any pair of those nodes, their distance will also be at most D. Thus, they will all be colored in different colors.

Then, the possible choice of color of v will be (K-X), multiplied it into the answer. If X>=K, it’s impossible to color the tree and result in colorness larger than or equal to D. Then, just return 0.

G-Coloring Tree(cont.)

Main Idea: Math, BFS

For each node, we need to find other nodes within distance K and already colored. It can be done by another BFS. Thus, for each node, it requires O(N) time.

Overall Time complexity: O(N^2)
Overall Space complexity: O(N)

H-diff-prime pairs

Main Idea: Math, Prime Sieve, Prefix Sum

H-diff-prime pairs(cont.)

Main Idea: Math, Prime Sieve, Prefix Sum

If gcd(i_1, j_1) is not equal to gcd(i_2, j_2), (i_1, j_1) won't be equal to (i_2, j_2).

The answer will be sum of **number of diff-prime pairs whose gcd is g** for each g.

H-diff-prime pairs(cont.)

Main Idea: Math, Prime Sieve, Prefix Sum

Iterate each g, and find two distinct prime p_1, p_2. Then,
(g p_1, g p_2) will be an answer if g p_1 <= N and g p_2 <= N.
It will be reduced to find the number of prime within (N/g).
It can be done by prime sieve and prefix sum.

Since p_1 not equal to p_2, it's obvious that gcd(g p_1, g p_2)=g

Overall Time complexity: O(N)
Overall Space complexity: O(N)

I-Expected Size of Random Convex Hull

Main Idea: Geometry, Random algo

I-Expected Size of Random Convex Hull

Main Idea: Geometry, Random algo

The shape of triangle doesn’t matter at all.

When randomly choosing N points within triangle T_1, you can always map them into N points within triangle T_2(By interpolation). Then, their property of convexity will be the same.

Thus, only the number of points chosen matters.

I-Expected Size of Random Convex Hull

Main Idea: Geometry, Random algo

For each N, randomly generate N points and compute the convex hull several times. Then, compute the average.

The number of testing times would be not enough within 1 second. But, N is only up to 10.

You can compute the answer locally. For each N, 10^8 times would be enough to converge to the answer.

Overall time complexity: O(1)
Overall space complexity: O(1)

J-Distance to Work

Main Idea: Geometry

J-Distance to Work(cont.)

Main Idea: Geometry

We can binary search the answer. For a fixed distance D, we need to compute the number of portion within the polygon.
It’s equivalent to compute the intersection area of a circle and a polygon.

We can decompose the polygon into several triangle. Then, compute the intersection of circle and triangle. The final answer will be a weighted sum of them.

Overall time complexity: O(\log(1/(10^{-6})) N M)
Overall space complexity: O(N+M)

Thanks