ABC 145 解説

kort0n, kyopro_friends, ynymxiaolongbao

2019 年 11 月 16 日

For International Readers: English editorial starts on page 7.

A: Circle

半径 r の円の面積は、円周率を π として πr^2 になります。したがって求める答えは $\pi r^2/\pi 1^2 = r^2$ となります。
以下は C 言語でのコードです。

```
#include<stdio.h>
int main(){
    int r;
    scanf("%d",&r);
    printf("%d\n",r*r);
}
```

B: Echo

N が奇数のときの答えは, 明らかに No です.
N が偶数のとき, 0-indexed で, S の 0 文字目から始まる $N/2$ 文字分の部分文字列と, S の $N/2$ 文字目から始まる $N/2$ 文字分の部分文字列が一致していれば答えは Yes であり, そうでなければ No です.
文字列の部分文字列は, 例えば C++ であれば substr 関数を使うことで容易に取得出来ます.
C++ による解答例:https://atcoder.jp/contests/abc145/submissions/8474441

C: Average Length

解法 1

$N!$ 通りの経路を全探索し, 各経路の長さを計算し, その平均値を出力します.
$N!$ 通りの経路の全探索は, 例えば C++ では next_permutation 関数を使うと容易に実装出来ます.
時間計算量は $O(N!N)$ です.
C++ による解答例:https://atcoder.jp/contests/abc145/submissions/8474526

解法 2

次の問題を考えます.

- N 個の互いに区別出来るボールを 1 列に並べる. 特定の 2 つのボールが隣り合うような並べ方は何通りあるか.

「特定の 2 つのボール」を一纏めにして考えて, この 2 つのボールの並び順も考慮すると, この問題の答えは $2(N-1)!$ 通りであることが分かります.
これより, 元の問題における $N!$ 通りの経路において, 各町のペア間の移動が発生するような経路は, $2(N-1)!$ 個であることが分かります.
これにより, 各町のペア間の距離の総和の $\frac{2(N-1)!}{N!} = \frac{2}{N}$ 倍が答えであることが分かります.
時間計算量は $O\left(N^2\right)$ です.
C++ による解答例:https://atcoder.jp/contests/abc145/submissions/8474573

D: Knight

1 回の移動で x 座標 $+y$ 座標の値は 3 増えます。なので X+Y が 3 の倍数でないとき答えは 0 です。

3 の倍数のとき、(+1,+2) の移動の回数を n、(+2,+1) の移動の回数を m とすると、各座標の値から $n+2m=X, 2n+m=Y$ という連立方程式が得られ、n, m が求まります。$n<0$ または $m<0$ のとき答えは 0 です。

そうでないとき、計 $n+m$ 回の移動のうち、どの n 回で (+1,+2) の移動をするか決めればよいので、答えは ${}_{n+m}C_n$ です。

この値は階乗とその逆元を計算することで $O(n+m+\log \mathrm{mod})$ で求める事ができます。工夫により $O(\min\{n,m\})$ で求めることもできます。

E: All-you-can-eat

解法 1

最後の注文は必ず $T-1$ 分時点で行うとして良いです。最後にどの料理を注文するかを全探索します。i 番目の料理を最後に注文する場合、それ以外の料理は $T-1$ 分時点で完食していなければならないので「i 番目以外の料理で $T-1$ 分以内に完食できる美味しさの合計の最大値 $+$ B_i」が満足度の最大となります。これは $O(NT)$ の DP で求めることができますが、各 i について N 回繰り返していては間に合いません。そこで次のような DP を考えます。

$DP1[i][j] = 1$〜i 番目の料理で j 分以内に完食できる美味しさの合計の最大値

$DP2[i][j] =$ i〜N 番目の料理で j 分以内に完食できる美味しさの合計の最大値

これを用いると、「i 番目以外の料理で $T-1$ 分以内に完食できる美味しさの合計の最大値」は

$$\max_{0 \leq j \leq T-1} DP1[i-1][j] + DP2[i+1][T-1-j]$$

により $O(T)$ で求めることが出来ます。よって全体で $O(NT)$ で求めることができました。

解法 2

最後の注文は必ず $T-1$ 分時点で行うとして良いです。料理を注文する順番は、注文する料理のなかで A_i が最大のものを最後にする場合のみ考えれば良いです (そうでない場合、注文した料理のうち A_i が最大のものと、最後に注文したものの順番を入れ替えても満足度は減りません)。従ってあらかじめ料理を A_i の昇順に並び変えておくことで、解法 1 の $DP1$ のみを用いて答えを計算することができます。計算量は $O(NT)$ です。

F: Laminate

$K = 0$ の場合、最小の操作回数は $\sum_{i=0}^{N-1} max(0, H_{i+1} - H_i)$ (ただし $H_0 = 0$ とする) になります。このことを示します。まず、行ごとに分けて考えたとき、i 列目では白だが $i+1$ 列目では黒で塗られる予定であるような i と同じ数だけ操作が必要になります。そして、列ごとにその列番号が先程の条件を満たす i になる行数を考えると、$max(0, H_{i+1} - H_i)$ であることがわかります。最後に、これらを合計することで元の式が得られます。

$K > 0$ の場合を考えます。列の H の値を変更する際にはいつも変更するものの左隣の値と同じになるようにすることで、実質的にその列が存在しないものとして見なすことができることがわかります。この方法は明らかに最適です。

よって、この問題は以下のように言い換えられます。

> 長さ N の数列 $H_1, H_2, \ldots, H_N$ が与えられるので、K つの項を削除したとき $\sum_{i=0}^{N-K-1} max(0, H_{i+1} - H_i)$ の値が最小でいくつになるか求めてください。ただし、$H_0 = 0$ とします。

この問題は、簡単な動的計画法によって $O(N^3)$ の計算量で解くことができます。$DP[x][y]$ を、削除しない項の集合に対して、最も右の項の番号が x、サイズが y であるときのコストの最小値とします。遷移式を $DP[x][y] = \min_{i=1}^{x-1}\{DP[i][y-1] + max(0, H_x - H_i)\}$ として順次求めることができます。$\min_{i=1}^{N}\{DP[i][N-K]\}$ が答えです。

BIT を用いて動的計画法を高速化することで $O(N^2 logN)$ の計算量で解くこともできます。

A: Circle

The area of a circle of radius r is πr^2. Therefore the answer is $\pi r^2 / \pi 1^2 = r^2$.
The following is a sample code in C.

```
#include<stdio.h>
int main(){
    int r;
    scanf("%d",&r);
    printf("%d\n",r*r);
}
```

B: Echo

If N is odd, then it's obviously No.

When N is even, if the substring of S of length $N/2$ beginning at 0-th letter (0-indexed) and that of length $N/2$ beginning at $N/2$-th letter is the same, the answer is Yes, and otherwise No. The substring of a string can be easily obtained, by using substr function in C++ for example. Sample code in C++:https://atcoder.jp/contests/abc145/submissions/8474441

C: Average Length

Solution 1

Brute force all the $N!$ paths, calculate each legnth, and output their average.
When brute-forcing through $N!$ paths, for example in C++ you can use next_permutation function to implement easily.
The total time complexity is $O(N!N)$.
Sample code in C++:https://atcoder.jp/contests/abc145/submissions/8474526

Solution 2

Let's consider the following question:

- How many ways are there to align N distinguishable balls, so that specific two balls are adjacent to each other?

By regarding "2 specific balls" as one bunch, and also considering the permutations of those two balls, the answer will be $2(N-1)!$.
Therefore, within $N!$ paths in the original problem, the number of paths in which there exists a move between each pair of towns is $2(N-1)!$.
Therefore, the answer will be $\frac{2(N-1)!}{N!} = \frac{2}{N}$ times the sum of distance between each pair of towns.
The total time complexity is $O\left(N^2\right)$.
Sample code in C++:https://atcoder.jp/contests/abc145/submissions/8474573

D: Knight

In one move the value of x coordinate + y coordinate increases by 3. So if X+Y is not a multiple of 3, the answer is 0.

If it's a multiple of three, let n be the number of move (+1,+2), and m be the number of move (+2,+1), then by considering each coordinate we obtain simultaneous equations $n + 2m = X, 2n + m = Y$, so we can find n, m. If $n < 0$ or $m < 0$, the answer is 0.

Otherwise, you have to chose m moves, in which the knight will move (+1, +2), from a total of $n + m$ moves, so the answer will be ${}_{n+m}C_n$.

You can calculate this value by using factorials and its inverse in a total of $O(n + m + \log \mathrm{mod})$ time. With more effort it can be solved in a total of $O(\min\{n, m\})$ time.

E: All-you-can-eat

Solution 1

You can assume that the last order will be done $T-1$ minutes after the first order. We will brute force which dish to order at last. If he will order i-th dish at last, any other dishes should be finished until $T-1$ minutes after the first order, so the maximum happiness will be "(the maximum sum of deliciousness of dishes, except for the i-th dish, that can be eaten within $T-1$ minutes) + B_i." This can be found in an $O(NT)$ DP, but you cannot repeat it N times for each i in time. Now let's consider the following DP.

$DP1[i][j]$ =the maximum sum of deliciousness of dishes, which are chosen from 1st- to i-th dishes, such that he can finish them in j minutes

$DP2[i][j]$=the maximum sum of deliciousness of dishes, which are chosen from i-th- to N-th dishes, such that he can finish them in j minutes

By using them, "the maximum sum of deliciousness of dishes, except for the i-th dish, that can be eaten within $T-1$ minutes" can be calculated in the following formula in a $O(T)$ time:

$\max_{0 \leq j \leq T-1} DP1[i-1][j] + DP2[i+1][T-1-j]$

Therefore this problem could be solved in a total of $O(NT)$ time.

Solution 2

You can assume that the last order will be done $T-1$ minutes after the first order. Assume that you have decided which dishes to order. Among the dishses you will order, consider the dish which its A_i is maximum. Then you can always assume that that dish will be ordered at last (otherwise you can swap the order of orders so that last dish will have the maximum A_i, in which case the hapiness he will gain will not decrease). Therefore, you can sort the dishes beforehand in the increasing order of A_i, so that you will only have to calculate $DP1$ of Solution 1. The total time complexity will be $O(NT)$.

F: Laminate

If $K = 0$, we claim that the minimum number of operations is $\sum_{i=0}^{N-1} max(0, H_{i+1} - H_i)$ (where $H_0 = 0$). Here is a proof. First, for each row, the number of operations you need is equal to the number of i such that i-th column of the row will be painted white and $i + 1$-th onewill be painted black. Then, for each column, consider the number of row such that its index satisfies the condition mentioned above; it is equal to $max(0, H_{i+1} - H_i)$. Finally, we can sum them up so that we obtain the original equation.

Then let's consider $K > 0$. When you change the value H of a column, you can always set the value same to that of the column on its left, so that you can virtually ignore the column. This is obviously optimal.

Therefore, the problem can be rephrased as follows:

> You are given a sequence $H_1, H_2, \ldots, H_N$ of length N. You can remove K elements from the sequence. Find the minimum value of $\sum_{i=0}^{N-K-1} max(0, H_{i+1} - H_i)$ after removing elements. Here, we assume that $H_0 = 0$.

This problem can be solved with a simple DP in a total of $O(N^3)$ time. Let $DP[x][y]$ be the minimum cost when, for a set of elements that are not removed, the index of leftmost element of the set is x and the size of the set is y. Then each value can be calculated by a recurrence relation $DP[x][y] = \min_{i=1}^{x-1}\{DP[i][y-1] + max(0, H_x - H_i)\}$. The answer will be $\min_{i=1}^{N}\{DP[i][N-K]\}$.

You can optimize DP by using BIT so that it will be solved in a total of $O(N^2 logN)$ time.