水以下コンテスト 解説

運営メンバーのなまえ

2024/03/30

A: Cyan or Less

Writer: Kyo_s_s

A: Cyan or Less

カラーコードが文字列で与えられる.

彩度を以下で定義する：

- r, g, b をカラーコードの RGB 値とする.

$$\text{彩度} = \frac{\max(r, g, b) - \min(r, g, b)}{\max(r, g, b)}$$

与えられたカラーコードが表す色は水色 #00c0c0 の彩度以下か判定せよ.

A: Cyan or Less　解法

step1. 与えられたカラーコードを RGB 値に変換する.

- 2 桁の 16 進数を 10 進数に変換するには, 1 つ目の数字を 16 倍して 2 つ目の数字を足せばよい.
  - a は 10 に, b は 11 に, ..., f は 15 に変換してから足す.

A: Cyan or Less 解法

step1. 与えられたカラーコードを RGB 値に変換する.

- 2 桁の 16 進数を 10 進数に変換するには, 1 つ目の数字を 16 倍して 2 つ目の数字を足せばよい.
  - a は 10 に, b は 11 に, ..., f は 15 に変換してから足す.

step2. 彩度を計算する.

- 彩度の定義通りに計算すれば OK.

A: Cyan or Less 解法

step1. 与えられたカラーコードを RGB 値に変換する.

- 2 桁の 16 進数を 10 進数に変換するには, 1 つ目の数字を 16 倍して 2 つ目の数字を足せばよい.
  - a は 10 に, b は 11 に, ..., f は 15 に変換してから足す.

step2. 彩度を計算する.

- 彩度の定義通りに計算すれば OK.

step3. 彩度が水色の彩度以下か判定する.

- 水色 #00c0c0 も同じ方法で彩度を計算しておけば, 大小関係を比較するだけ.

A: Cyan or Less 余談

実は，水色 #00c0c0 の彩度は 1

$\rightarrow$ 彩度がこれより大きくなる色は存在しない！

このため，すべてのケースで Yes と正解すればよいです．

一旦没になったのですが，丁寧に書いてもよいし，気づけば１行で解けるので面白くない？となり出題されました．

X: f(f(f(f(f(x)))))

Writer: Kyo_s_s

X: f(f(f(f(f(x)))))

整数 K $(1 \leq K \leq 10^{18})$ と x についての関数 $f(x)$ が与えられる． 最初 $x = 1$ として，次の操作を K 回繰り返す：

- x を $f(x)$ で更新する

最終的な $x \bmod 998$ は？

X: f(f(f(f(f(x)))))

整数 K $(1 \leq K \leq 10^{18})$ と x についての関数 $f(x)$ が与えられる．最初 $x = 1$ として，次の操作を K 回繰り返す：

- x を $f(x)$ で更新する

最終的な $x \bmod 998$ は？

構文解析をがんばる必要がある ...？

$\rightarrow$ めちゃめちゃな式は与えられないため，そんなに頑張らなくてよい．

X: f(f(f(f(f(x))))) 部分点 1 解法

- $K = 1, \quad f(x)$ に *, ^ は含まれない.

X: f(f(f(f(f(x)))))　部分点 1 解法

- $K = 1,\quad f(x)$ に *, ^ は含まれない.

$f(x)$ は, x もしくは 1 以上 10^9 未満の整数 の和で表されているので,

step 1. $f(x)$ を $+$ で分割する

step 2. 分割したそれぞれの文字列を数値に変換する (x なら 1)

step 3. すべて足し合わせる

を実装すれば OK.

X: f(f(f(f(f(x))))) 部分点 2 解法

- $K \leq 10^4$, $f(x)$ に ^ は含まれない.

X: f(f(f(f(f(x))))) 部分点 2 解法

- $K \leq 10^4, \quad f(x)$ に ^ は含まれない.

部分点 1 解法で,「数値 / x と演算子 $+$ のみからなる数式」を計算した.
$\rightarrow$ 改造すると,「数値 / x と 演算子 * のみからなる数式」を計算できる.

X: f(f(f(f(f(x))))) 部分点 2 解法

- $K \leq 10^4$, $f(x)$ に ^ は含まれない.

部分点 1 解法で,「数値 / x と演算子 $+$ のみからなる数式」を計算した.

$\to$ 改造すると,「数値 / x と 演算子 * のみからなる数式」を計算できる.

step 1. $f(x)$ を $+$ で分割する

step 2. 分割したそれぞれの文字列は「数値 / x と 演算子 * のみからなる数式」なので，それぞれを計算する

step 3. すべて足し合わせる

そのままだとオーバーフローするため，和 / 積の計算時に $\mathrm{mod}\ 998$ を取れば OK.

X: f(f(f(f(f(x))))) 部分点 3 解法

- $K \leq 10^4$

X: f(f(f(f(f(x))))) 部分点 3 解法

- $K \leq 10^4$

「数値 / x と 演算子 *, ^ のみからなる数式」を計算できれば OK．これは，

step 1. $f(x)$ を * で分割する

step 2. 分割したそれぞれの文字列は「数値 / x もしくは，演算子 ^ のみからなる数式」なので，

- ^ を含む : (数値 or x)^(数値) の形になっているので計算する
- ^ を含まない : そのまま数値に変換する

step 3. すべて掛け合わせる

とすれば実装できる．愚直に K 回シミュレーションすれば OK.

X: f(f(f(f(f(x))))) 満点解法

X: f(f(f(f(f(x))))) 満点解法

関数 $f(x)$ の形から，x を更新するときに $x \bmod 998$ で更新してもよい． つまり，関数 f は，

$$f : \mathbb{Z}/998\mathbb{Z} \to \mathbb{Z}/998\mathbb{Z}$$

とみなせる．$\to$ ダブリングできる！

X: f(f(f(f(f(x))))) 満点解法

関数 $f(x)$ の形から，x を更新するときに $x \bmod 998$ で更新してもよい． つまり，関数 f は，

$$f : \mathbb{Z}/998\mathbb{Z} \to \mathbb{Z}/998\mathbb{Z}$$

とみなせる. $\to$ ダブリングできる！

前処理として $x = 0, 1, ..., 997$ に対する $f(x)$ を計算しておくことにより，$x = 0, 1, ..., 997$ に対する $f(f(x))$ の値が計算でき， さらにこの値を用いて x に対する $f(f(f(f(x))))$ の値が計算でき，...

これを繰り返し，必要な部分を適用することで K 回操作した後の値を求めることができる！

X: f(f(f(f(f(x))))) 余談

- 原案では 引き算 - と 括弧 () も含まれていたのですが，さすがにやりすぎということで無くしました．
  - 括弧が入ってくるとちゃんと再帰的な処理をする必要があります．

- ^ もなくていいじゃん， と言われていたのですが, ^ が無いと Python の eval を使うことで構文解析をサボれてしまうのでやむなく入れました．
  - 部分点 2 までは eval をやるだけで通せます．
  - 部分点 3 も，re.sub(r'(x|\d+)\^(\d+)', r'pow(\1,\2,998)', S) と置換することで，構文解析パートは eval で済ませることができます (ダブリングはする必要があります)．

X: Unions

Writer: yasunori

X: Unions

N 個の国と，複数の国で構成される同盟が M 個ある．i 個目の同盟には C_i 個の国 $A_{i,1}, A_{i,2}, ..., A_{i,C_i}$ の国が所属している．

同じ同盟に所属している国同士は直接行き来でき，同盟 i に属している国同士は D_i 分で移動できる．同じ同盟に属していない国同士は直接行き来できない．

国 $2, 3, ..., N$ について，国 1 から移動するのにかかる時間の最小値を求めよ．

- $2 \leq N \leq 10^5$
- $1 \leq M \leq 10^5$
- $\sum_{i=1}^{M} C_i \leq 10^5$

X: Unions 部分点 1 解法

- $C_i = 2 \ (1 \leq i \leq M)$

X: Unions 部分点 1 解法

- $C_i = 2 \ (1 \leq i \leq M)$

すべての同盟がちょうど 2 個の国からなるため，同盟 i は「$A_{i,1}$ と $A_{i,2}$ は移動に D_i 分かかる道でつながっている」と言い換えることができる．

X: Unions 部分点 1 解法

- $C_i = 2 \ (1 \leq i \leq M)$

すべての同盟がちょうど 2 個の国からなるため，同盟 i は「$A_{i,1}$ と $A_{i,2}$ は移動に D_i 分かかる道でつながっている」と言い換えることができる．

つまり，国 1 から国 $2, 3, ..., N$ への最短距離を求める問題に帰着できる．

$\rightarrow$ このグラフは頂点の数が N，辺の数が M であるため，ダイクストラ法で解ける！

国 1 を始点として，ダイクストラ法を使って各国への最短距離を求めることで，部分点 1 に正解できる．

X: Unions　部分点 2 解法

- $\sum_{i=1}^{M} C_i \leq 10^3$

X: Unions　部分点 2 解法

- $\sum_{i=1}^{M} C_i \leq 10^3$

同じ同盟に属する国同士は「D_i 分かかる道でつながっている」と言い換えることができる.

X: Unions 部分点 2 解法

- $\sum_{i=1}^{M} C_i \leq 10^3$

同じ同盟に属する国同士は「D_i 分かかる道でつながっている」と言い換えることができる．

$\sum_{i=1}^{M} C_i \leq 10^3$ の制約から，同じ同盟に属する国同士のペアすべてに D_i の重みを持つ辺を張っても十分間に合う．実際，辺数を $|E|$ とすると，

$$|E| = \sum_{i=1}^{M} \frac{C_i(C_i - 1)}{2} < \frac{1}{2} \sum_{i=1}^{M} C_i^2 \leq \frac{1}{2} \left(\sum_{i=1}^{M} C_i \right)^2 \leq 5 \times 10^5$$

となる．

部分点 1 と同様にダイクストラ法を用いれば，部分点 2 に正解できる．

X: Unions　満点解法

X: Unions　満点解法

部分点 2 の方法だと，張る辺の本数が多すぎて間に合わない．

超頂点を導入することで間に合う！

X: Unions　満点解法

部分点 2 の方法だと，張る辺の本数が多すぎて間に合わない．

超頂点を導入することで間に合う！

各国 $1, 2, ..., N$ と，同盟 $1, 2, ..., M$ を頂点とするグラフを考える．

$i = 1, 2, ..., M,\ j = 1, 2, ..., C_i$ について，

- 国 $A_{i,j}$ から 同盟 i に，重み D_i の辺を張る
- 同盟 i から 国 $A_{i,j}$ に，重み 0 の辺を張る

とすると，同じ同盟に属する国同士は同盟の頂点を経由することで D_i 分のコストで移動できる．

このグラフの辺数は $2 \times \sum_{i=1}^{M} C_i$ であるため，このグラフ上でダイクストラ法をすればよい．

X: X-word Database

Writer: Kyo_s_s

X: X-word Database

整数 $X \left(4 \leq X \leq 10^5\right)$ と文字列 S $(1 \leq |S| \leq X)$ が与えられる．

以下の条件を満たす文字列を**よい文字列**と呼ぶ：

- 文字列の長さが X 以下
- 辞書順で S 以下
- 連続部分列に cyan を含む

よい文字列は何個ある？

X: X-word Database

整数 $X\ \left(4 \le X \le 10^5\right)$ と文字列 $S\ (1 \le |S| \le X)$ が与えられる．

以下の条件を満たす文字列を**よい文字列**と呼ぶ：

- 文字列の長さが X 以下
- 辞書順で S 以下
- 連続部分列に cyan を含む

よい文字列は何個ある？

$X = 5,\ S =$ cyanc のとき，条件を満たす文字列は,

acyan, bcyan, ccyan, cyan, cyana, cyanb, cyanc

の 7 個.

X: X-word Database 部分点 1 解法

- $X \leq 8$

X: X-word Database　部分点 1 解法

- $X \leq 8$

cyan を含む長さ 8 以下の文字列を全て試せば OK.

たとえば長さが 8 で cyan を含む文字列は,

****cyan, ***cyan*, **cyan**, *cyan***, cyan****

のどれかの形なので, * に入るアルファベットを全て試してそれぞれがよい文字列かどうかを判定すればよい！

X: X-word Database　部分点 1 解法

- $X \leq 8$

cyan を含む長さ 8 以下の文字列を全て試せば OK.

たとえば長さが 8 で cyan を含む文字列は,

****cyan, ***cyan*, **cyan**, *cyan***, cyan****

のどれかの形なので, * に入るアルファベットを全て試してそれぞれがよい文字列かどうかを判定すればよい！

このままだと $X = 8$ で cyancyan を 2 回数えてしまうのでそこだけ注意.

X: X-word Database　部分点 2 解法

- $X \leq 10^3,\ S$ はすべて z で長さが X の文字列

X: X-word Database　部分点 2 解法

- $X \leq 10^3,\ S$ はすべて z で長さが X の文字列

長さが X 以下で cyan を含むような文字列はすべてよい文字列になる.

X: X-word Database 部分点 2 解法

- $X \leq 10^3,\ S$ はすべて z で長さが X の文字列

長さが X 以下で cyan を含むような文字列はすべてよい文字列になる.

長さが K で cyan を含む文字列の個数は，包除原理を使うと,

$$\sum_{i=1}^{\lfloor K/4 \rfloor} (-1)^{i+1} \cdot 26^{K-4i} \cdot {}_{i+1}H_{K-4i}$$

で求められる (ここで，${}_nH_k$ は重複組合せ).

X: X-word Database 部分点 2 解法

- $X \leq 10^3,\ S$ はすべて z で長さが X の文字列

長さが X 以下で cyan を含むような文字列はすべてよい文字列になる．

長さが K で cyan を含む文字列の個数は，包除原理を使うと，

$$\sum_{i=1}^{\lfloor K/4 \rfloor} (-1)^{i+1} \cdot 26^{K-4i} \cdot {}_{i+1}H_{K-4i}$$

で求められる (ここで，${}_nH_k$ は重複組合せ)．よって，答えは，

$$\sum_{K=4}^{X} \sum_{i=1}^{\lfloor K/4 \rfloor} (-1)^{i+1} \cdot 26^{K-4i} \cdot {}_{i+1}H_{K-4i}$$

X: X-word Database　満点解法

X: X-word Database　満点解法

桁 DP で解ける！

$\mathrm{dp}[i][\mathrm{smaller}][j]$:= X の上から i 番目まで見て，

smaller $= 0$: X と同じ / 1 : X より小さく，

cyan の j 文字目までを末尾に含む / すでに cyan を含む

文字列の個数

X: X-word Database 満点解法

桁 DP で解ける！

$$\mathrm{dp}[i][\mathrm{smaller}][j] := X \text{ の上から } i \text{ 番目まで見て，}$$
$$\mathrm{smaller} = 0 : X \text{ と同じ} \ / \ 1 : X \text{ より小さく，}$$
$$\text{cyan の } j \text{ 文字目までを末尾に含む} \ / \ \text{すでに cyan を含む}$$
$$\text{文字列の個数}$$

上から DP をしていくことで，X 字以下の文字列の個数も求められている．

$$\sum_{i=0}^{X}\Big(\mathrm{dp}[i][0]\big[\text{cyan を含む}\big] + \mathrm{dp}[i][1]\big[\text{cyan を含む}\big]\Big)$$

が答え．

X: Range Rotate Query

Writer: loop0919

X: Range Rotate Query

二次元平面上に N $(3 \leq N \leq 10^5)$ 個の点がある．

以下のクエリを Q $(1 \leq Q \leq 50000)$ 個処理せよ：

- 原点からのユークリッド距離が $\sqrt{l}$ 以上 $\sqrt{r}$ 以下の点すべてを反時計回りに θ 度回転させる
- 点 a, b, c を頂点とする三角形の面積を出力する

X: Range Rotate Query 部分点１解法

- $N \leq 100,\ Q \leq 1000,$ 面積を求めるクエリのみ

X: Range Rotate Query 部分点 1 解法

- $N \leq 100,\ Q \leq 1000,$ 面積を求めるクエリのみ

3 点 $\left(x_1, y_1\right), \left(x_2, y_2\right), \left(x_3, y_3\right)$ を頂点とする三角形の面積を求めたい．

X: Range Rotate Query　部分点 1 解法

- $N \leq 100,\ Q \leq 1000,$ 面積を求めるクエリのみ

3 点 $\left(x_1, y_1\right), \left(x_2, y_2\right), \left(x_3, y_3\right)$ を頂点とする三角形の面積を求めたい.

このとき，三角形の面積は,

$$\frac{1}{2} \cdot \left| x_1 y_2 + x_2 y_3 + x_3 y_1 - x_1 y_3 - x_2 y_1 - x_3 y_2 \right|$$

で求められる (証明はここでは略).

クエリごとにこれを計算すれば OK.

X: Range Rotate Query　部分点 2 解法

- $N \leq 100,\ Q \leq 1000$

X: Range Rotate Query 部分点 2 解法

- $N \leq 100,\ Q \leq 1000$

ある頂点 (x, y) を θ 度反時計回りに回転させせると，その座標は，

$$\left(x\cos\frac{\theta\pi}{180} - y\sin\frac{\theta\pi}{180},\ x\sin\frac{\theta\pi}{180} + y\cos\frac{\theta\pi}{180}\right)$$

で求められる (証明はここでは略．回転行列を用いればよい).

X: Range Rotate Query　部分点 2 解法

- $N \leq 100,\ Q \leq 1000$

ある頂点 (x, y) を θ 度反時計回りに回転させせると，その座標は，

$$\left(x\cos\frac{\theta\pi}{180} - y\sin\frac{\theta\pi}{180},\ x\sin\frac{\theta\pi}{180} + y\cos\frac{\theta\pi}{180} \right)$$

で求められる (証明はここでは略．回転行列を用いればよい).

各点について，何度回転したかを記録しておけば，

- 回転クエリ : 距離が $\sqrt{l}$ 以上 $\sqrt{r}$ 以下の点の回転した度数を θ だけ増やす
- 面積クエリ : 3 点それぞれの今の座標を求めたのち，面積を計算する

とすれば部分点 2 に正解できる.

毎回回転させせると誤差で落ちるので注意！

X: Range Rotate Query　満点解法

X: Range Rotate Query 満点解法

各点をユークリッド距離でソートしておくと，回転クエリは，「$\sqrt{l}$ 以上 $\sqrt{r}$ 以下の区間に θ を加算する」という操作になる．

- 実際に加算する区間は二分探索をしたりすれば求められる．

区間加算一点取得ができればよい．

$\rightarrow$ BIT でできる！

X: Range Rotate Query　満点解法

各点をユークリッド距離でソートしておくと，回転クエリは，「$\sqrt{l}$ 以上 $\sqrt{r}$ 以下の区間に θ を加算する」という操作になる．

- 実際に加算する区間は二分探索をしたりすれば求められる．

区間加算一点取得ができればよい．

$\rightarrow$ BIT でできる！

- 区間 $[l, r)$ に x を加算：
  - l 番目の要素に x を加算，r 番目の要素に $-x$ を加算
- i 番目の要素を取得：
  - $[0, i)$ の総和が求めたい値

BIT 上で imos 法をするイメージ．

X: Counting Zig Zag Sequence

Writer: Ackvy

X: Counting Zig Zag Sequence

正整数 N $(1 \leq N \leq 5000)$, K $(1 \leq K \leq 5000)$ が与えられる.

次のような数列を**ジグザグ数列**と呼ぶ:

- 連続する 2 項が異なる
- 各 i $(1 \leq i \leq N-2)$ について, $(A_i - A_{i+1})(A_{i+1} - A_{i+2}) \leq 0$ を満たす
  - 増加 $\to$ 減少 $\to$ 増加 $\to \cdots$ または 減少 $\to$ 増加 $\to$ 減少 $\to \cdots$

長さが N, 各要素が 1 以上 K 以下の整数からなるジグザグ数列の個数を求めよ.

たとえば, $(2, 7, 1, 8, 2, 8)$ はジグザグ数列だが, $(3, 1, 4, 1, 5, 9)$ はジグザグ数列ではない.

X: Counting Zig Zag Sequence　部分点 1 解法

- $N \leq 5,\ K \leq 5$

X: Counting Zig Zag Sequence 部分点 1 解法

- $N \leq 5,\ K \leq 5$

長さが N, 各要素が K 以下の正整数からなる数列を全探索し，ジグザグ数列かどうかを判定すれば OK.

長さが N, 各要素が K 以下の正整数からなる数列は合計で K^N 個あり，
$N \leq 5,\ K \leq 5$ の制約であればすべての数列を試すことができる.

0 から K^N まで, K 進数として考えると実装が楽.

X: Counting Zig Zag Sequence 部分点 2 解法

- $N \leq 5000,\ K \leq 50$

X: Counting Zig Zag Sequence　部分点 2 解法

- $N \leq 5000,\ K \leq 50$

$N = 1$ のとき，定義よりジグザグ数列は K 個．$N \geq 2$ について考える．

X: Counting Zig Zag Sequence 部分点 2 解法

- $N \leq 5000,\ K \leq 50$

$N = 1$ のとき，定義よりジグザグ数列は K 個．$N \geq 2$ について考える．

次のような DP を考える：

$$\mathrm{dp}[i][j][\mathrm{increased}] := i \text{ 番目の要素が } j \text{ で，}$$
$$\text{次の要素が } \mathrm{increased} = 0 : \text{減少} \ / \ 1 : \text{増加}$$
$$\text{するべきなジグザグ数列の個数}$$

初期値は $\mathrm{dp}[1][\star][\star] = 1$.

X: Counting Zig Zag Sequence 部分点 2 解法

- $N \leq 5000,\ K \leq 50$

$\mathrm{dp}[i+1][j][0]$ は，「i 番目の要素が j より小さく，次に増加する」ような dp 配列の要素の和で求められる．

- 「i 番目の要素が j より小さく，次に増加する」ような数列は，$i+1$ 番目の要素を j にしてもジグザグ数列を保つ．

X: Counting Zig Zag Sequence 部分点 2 解法

- $N \leq 5000,\ K \leq 50$

$\mathrm{dp}[i+1][j][0]$ は，「i 番目の要素が j より小さく，次に増加する」ような dp 配列の要素の和で求められる．

- 「i 番目の要素が j より小さく，次に増加する」ような数列は，$i+1$ 番目の要素を j にしてもジグザグ数列を保つ．

$\mathrm{dp}[i+1][j][1]$ も同様に考えると，遷移は，

$$\mathrm{dp}[i+1][j][0] = \sum_{k=1}^{j-1} \mathrm{dp}[i][k][1]$$

$$\mathrm{dp}[i+1][j][1] = \sum_{k=j+1}^{K} \mathrm{dp}[i][k][0]$$

となる．答えは，$\sum_{j=1}^{K} (\mathrm{dp}[N][j][0] + \mathrm{dp}[N][j][1])$.

X: Counting Zig Zag Sequence 満点解法

部分点 2 解法の DP:

$$\mathrm{dp}[i+1][j][0] = \sum_{k=1}^{j-1} \mathrm{dp}[i][k][1]$$

$$\mathrm{dp}[i+1][j][1] = \sum_{k=j+1}^{K} \mathrm{dp}[i][k][0]$$

を愚直に書くと TLE してしまうが, $\sum$ の部分を累積和で前計算しておくことにより, $N \leq 5000,\ K \leq 5000$ の制約で高速に答えを求めることができる.

X: Counting Zig Zag Sequence 満点解法

部分点 2 解法の DP:

$$\mathrm{dp}[i+1][j][0] = \textstyle\sum_{k=1}^{j-1} \mathrm{dp}[i][k][1]$$

$$\mathrm{dp}[i+1][j][1] = \textstyle\sum_{k=j+1}^{K} \mathrm{dp}[i][k][0]$$

を愚直に書くと TLE してしまうが, $\sum$ の部分を累積和で前計算しておくことにより, $N \leq 5000,\ K \leq 5000$ の制約で高速に答えを求めることができる.

増加 $\rightarrow$ 減少 $\rightarrow$ 増加 $\rightarrow$ $\cdots$ となるようなジグザグ数列の個数を求めたのち, 答えを 2 倍してもよい.

- こうすると, 次に増加 / 減少する を状態として持たなくてもよくなる.
- 偶奇で場合分けすれば OK.