水以下コンテスト 解説

Kyo_s_s, yasunori, Ackvy, えくと ,
for ループ , Nichi10p, Blueberry1001, ragna

2024/03/30

ごめんなさい

J: Counting Zig Zag Sequence にて，部分点 1 の制約で誤って部分点 2 が設定されていました．

コンテスト終了間際に発覚してしまい，申し訳ございません．

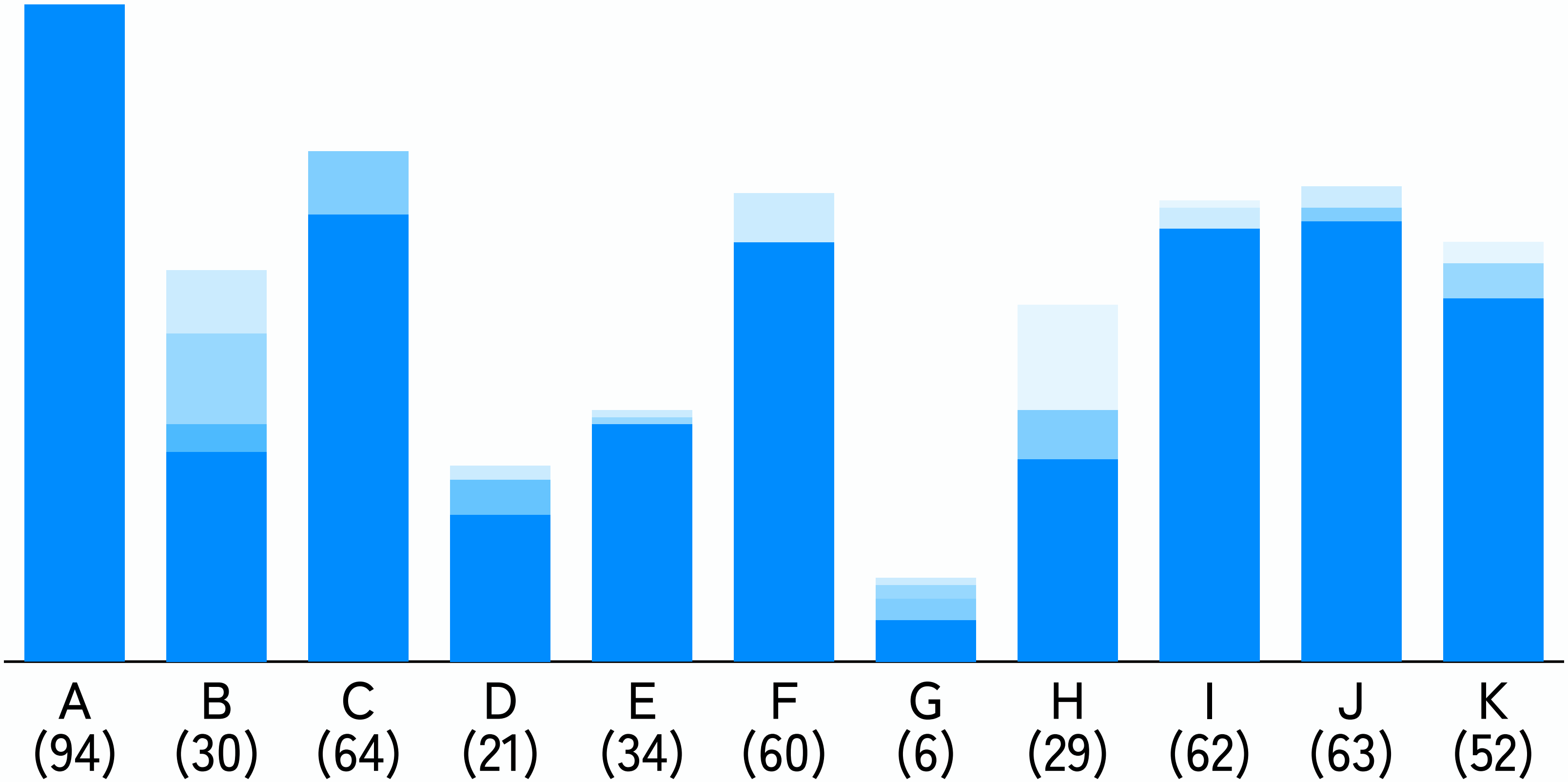
アンケートのお願い

アンケートにご協力いただけると嬉しいです.

優勝

オンサイト優勝	magepika	(980 点 , 3:40:08)
オンサイト 2 位	shinchan	(900 点 , 2:28:55)
オンサイト 3 位	ochiaigawara	(980 点 , 4:09:24)
全体優勝	ssrs	(1100 点， 1:42:19)

統計



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．

A: Cyan or Less 統計情報

AC 数	94 / 96
AC 率	94.0 %
平均点	94.0 点
オンサイト FA	shinchan (2:31)
FA	shinchan (2:31)

B: f(f(f(f(f(x)))))

Writer: Kyo_s_s

B: 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？

B: 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．

B: f(f(f(f(f(x))))) 部分点 1 解法

- $K = 1,\ \ f(x)$ に *, ^ は含まれない.

B: 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.

B: 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.

B: 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.

B: 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．

$\rightarrow$ 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．

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.

B: f(f(f(f(f(x))))) 部分点 3 解法

- $K \leq 10^4$

B: 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.

B: f(f(f(f(f(x))))) 満点解法

B: 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！

B: 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！

B: 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)．

B: f(f(f(f(f(x)))))　統計情報

AC 数	30 / 96
AC 率	30.0 %
平均点	39.80 点
オンサイト FA	takumi152 (37:20)
FA	kude (17:54)

C: Unions

Writer: yasunori

C: Unions

$N\ (2 \le N \le 10^5)$ 個の国と，複数の国による同盟が $M\ (1 \le M \le 10^5)$ 個あり，
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．
（ここで，C_i は $\sum_{i=0}^{M} C_i \le 10^5$ を満たす．）

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．

C: Unions 部分点 1 解法

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

C: Unions　部分点 1 解法

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

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．

C: 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．

C: Unions　部分点 2 解法

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

C: 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.

C: 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．

C: Unions　満点解法

C: 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！

C: 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．

C: Unions　統計情報

AC 数	64 / 96
AC 率	64.0 %
平均点	68.50 点
オンサイト FA	AngrySadEight (10:57)
FA	TKTYI (3:25)

D: X-word Database

Writer: Kyo_s_s

D: 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？

D: 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？

$X = 5,\ S =$ cyanc のとき，条件を満たす文字列は,

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

の 7 個.

D: X-word Database 部分点1解法

- $X \leq 8$

D: 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！

D: 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.

D: X-word Database　部分点 2 解法

- $X \le 10^3,\ S$ はすべて z で長さが X の文字列

D: 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.

D: 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.

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

cyan の j 文字目までを末尾に含む / すでに cyan を含む

ような文字列の個数

として DP することで, $\mathrm{dp}[i]$[cyan を含む] の値は 「cyan を連続部分列に持つ, ちょうど i 文字の文字列の個数」となる.

D: X-word Database 部分点 2 解法

- $X \le 10^3,\ S$ 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.

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

cyan の j 文字目までを末尾に含む / すでに cyan を含む

ような文字列の個数

として DP することで, $\mathrm{dp}[i]\big[$cyan を含む$\big]$ の値は 「cyan を連続部分列に持つ, ちょうど i 文字の文字列の個数」となる.

$\rightarrow$ これらの和をとり, $\sum_{i=1}^{X} \mathrm{dp}[i]\big[$cyan を含む$\big]$ が答え.

D: X-word Database 満点解法

D: 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，

$\mathrm{smaller} = 0 : X$ 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

D: 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，

$\mathrm{smaller} = 0 : X$ 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

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．

D: X-word Database 統計情報

AC 数	21 / 96
AC 率	21.0 %
平均点	24.40 点
オンサイト FA	deuteridayo (57:47)
FA	ssrs (27:13)

E: Range Rotate Query

Writer: loop0919

E: 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

E: 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

E: 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.

E: Range Rotate Query　部分点 1 解法

- $N \le 100,\ Q \le 1000,$ 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.

E: Range Rotate Query 部分点 2 解法

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

E: 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).

E: 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！

E: Range Rotate Query　満点解法

E: 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(fenwick tree) でできる！

E: 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．

$\to$ BIT(fenwick tree) 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．

E: Range Rotate Query 統計情報

AC 数	34 / 96
AC 率	34.0 %
平均点	34.60 点
オンサイト FA	porkleoi (43:27)
FA	potato167 (42:01)

F: Subset Mex

Writer: Kyo_s_s

F: Subset Mex

$N \left(1 \leq N \leq 2 \times 10^5\right)$ 枚のカードがあり，i 番目のカードには

整数 $A_i \left(0 \leq A_i \leq 2 \times 10^5\right)$ が書かれている．

N 枚のカードの中から 1 枚以上カードを選ぶ方法をすべて考え，それぞれの選び方の mex の総和を求めよ．

たとえば，$A = (0, 1, 1, 2)$ で，

- $(1, 3)$ 枚目のカードを選んだとき，$\mathrm{mex}(A_1, A_3) = \mathrm{mex}(0, 1) = 2$．
- $(1, 2, 4)$ 枚目のカードを選んだとき，$\mathrm{mex}(A_1, A_2, A_4) = \mathrm{mex}(0, 1, 2) = 3$．

このほか，すべての選び方についての mex の総和は 17 となる．

F: Subset Mex　部分点 1 解法

- $N \leq 17,\ A_i \leq 1000$

F: Subset Mex　部分点 1 解法

- $N \leq 17,\ A_i \leq 1000$

N 枚のカードから 1 枚以上選ぶ方法は $2^N - 1$ 通り．$N = 17$ のとき 131071 通りしかないので，すべての選び方を試して mex の総和を求めれば OK.

F: Subset Mex　部分点 1 解法

- $N \leq 17,\ A_i \leq 1000$

N 枚のカードから 1 枚以上選ぶ方法は $2^N - 1$ 通り．$N = 17$ のとき 131071 通りしかないので，すべての選び方を試して mex の総和を求めれば OK.

すべての選び方を列挙するには bit 全探索をすればよい．

1 から 2^N までをループで回し，立っているビットに対応するカードを選んでいるとすれば実装が楽.

F: Subset Mex　部分点 2 解法

- $A_i \leq 1000$

$A_i \leq 1000$ の制約から，どのようなカードの選び方をしても mex の最大値は 1001 以下である．

$\to x = 1, 2, ..., 1001$ に対して，mex が x になるようなカードの選び方が何通りあるか求められればよい．

mex が x になるためには ...

- $0, 1, ..., x-1$ がそれぞれ 1 枚以上選ばれている
- x が選ばれていない
- $x+1$ 以上の数は選ばれているか選ばれていないかは関係ない

F: Subset Mex　部分点 2 解法

$c(x)$ を A の中に含まれる x の個数 とすると, mex が x になるようなカードの選び方の個数は,

- $0, 1, ..., x-1$ がそれぞれ 1 枚以上選ばれている

  $\rightarrow \left(2^{c(0)}-1\right) \times \left(2^{c(1)}-1\right) \times \cdots \times \left(2^{c(x-1)}-1\right)$ 通り
- x が選ばれていない

  $\rightarrow 1$ 通り
- $x+1$ 以上の数は選ばれているか選ばれていないかは関係ない

  $\rightarrow 2^{c(x+1)} \times 2^{c(x+2)} \times \cdots \times 2^{c(1002)}$ 通り

これらの積が, mex が x になるようなカードの選び方の個数.

各 x についてこの値を求め, x を掛けたのちに足し合わせればよい.

F: Subset Mex　満点解法

部分点 2 解法の方法を高速化することを考える.

$x = 1, 2, ..., 2 \times 10^5 + 1$ に対して，以下を求めたい：

$$f(x) := \prod_{i=0}^{x-1} \left(2^{c(i)-1}\right) \times \prod_{i=x+1}^{2\times10^5+2} 2^{c(i)}$$

F: Subset Mex　満点解法

部分点 2 解法の方法を高速化することを考える.

$x = 1, 2, ..., 2 \times 10^5 + 1$ に対して，以下を求めたい：

$$f(x) := \prod_{i=0}^{x-1} \left(2^{c(i)-1}\right) \times \prod_{i=x+1}^{2\times 10^5+2} 2^{c(i)}$$

- $S_i = \prod_{j=0}^{i} \left(2^{c(j)} - 1\right)$ とすると，これは前から順に求めれば高速.
- $T_i = \prod_{j=i}^{2\times 10^5+2} 2^{c(j)}$ とすると，これは後ろから順に求めれば高速.

これより，$f(x)$ は $S_{x-1} \times T_{x+1}$ で求められる．答えは $\sum_{x=1}^{2\times 10^5+1} (x \times f(x))$.

F: Subset Mex 統計情報

AC 数	60 / 96
AC 率	60.0 %
平均点	61.40 点
オンサイト FA	magepika (10:23)
FA	magepika (10:23)

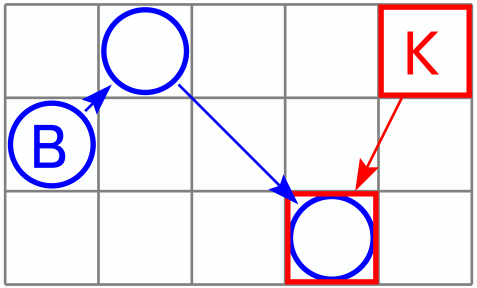
G: Long Chess Board

Writer: yasunori

G: Long Chess Board

縦 $H\ (2 \le H \le 3)$，横 $W\ \left(H \le W \le 10^9\right)$ の長方形のチェス盤があり，ナイトが (r_k, c_k) に，ビショップが (r_b, c_b) に置いてある．

ナイトとビショップを移動させ，同じマスに移動させるのに必要な最小手数を求めよ．



- ナイトを $(1, 5)$ から $(3, 4)$ に移動させる．
- ビショップを $(2, 1)$ から $(1, 2)$ に，$(1, 2)$ から $(3, 4)$ に移動させる．

この例では，答えは 3 手となる．

G: Long Chess Board 部分点 1, 2 解法

- W の総和 $\leq 10^5$

G: Long Chess Board 部分点 1, 2 解法

- W の総和 $\leq 10^5$

各マスについてナイト / ビショップを移動させる最短手数を求めればよい．
$\rightarrow$ ナイト / ビショップそれぞれ BFS をすれば求められる！

盤面の高さがとても小さいため，ビショップが 1 手で動きうるマスは少ない．ビショップの移動先を全探索しても間に合う．

G: Long Chess Board 部分点 1, 2 解法

- W の総和 $\leq 10^5$

各マスについてナイト / ビショップを移動させる最短手数を求めればよい．
$\rightarrow$ ナイト / ビショップそれぞれ BFS をすれば求められる！

盤面の高さがとても小さいため，ビショップが 1 手で動きうるマスは少ない．ビショップの移動先を全探索しても間に合う．

- $\mathrm{dist\_k}_{i,j}$:=ナイトをマス (i,j) まで移動させる最短手数
- $\mathrm{dist\_b}_{i,j}$:=ビショップをマス (i,j) まで移動させる最短手数

とすれば，答えは $\displaystyle\min_{1\leq i\leq H,\ 1\leq j\leq W}\left(\mathrm{dist\_k}_{i,j}+\mathrm{dist\_b}_{i,j}\right)$.

G: Long Chess Board 部分点 3 解法

- $H = 2$

G: Long Chess Board 部分点 3 解法

- $H = 2$

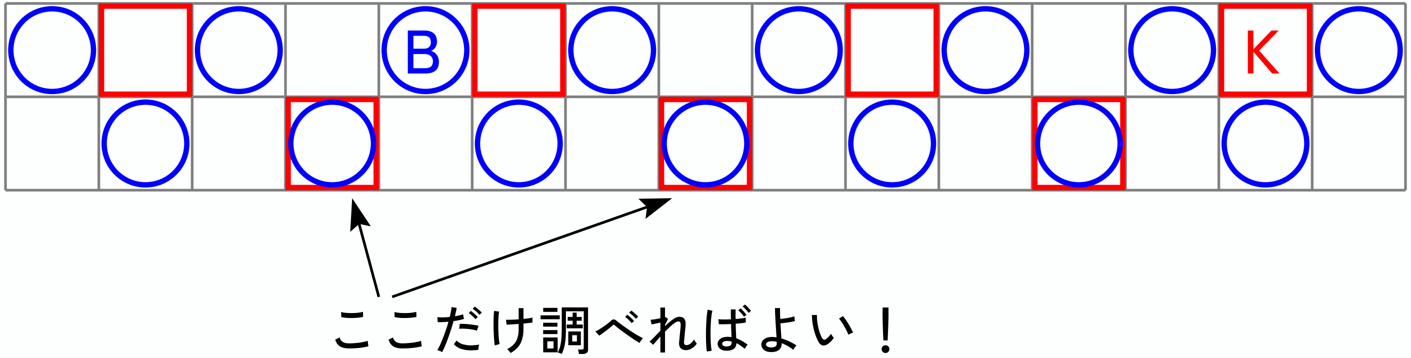
- ナイト，ビショップともに移動できるマスは一本道になっている．
- 集合しうるマスは周期 4 で存在する．
- 横方向に 4 つ進むのに，ナイトは 2 回，ビショップは 4 回かかる．

G: Long Chess Board　部分点 3 解法

- $H = 2$

- ナイト，ビショップともに移動できるマスは一本道になっている．
- 集合しうるマスは周期 4 で存在する．
- 横方向に 4 つ進むのに，ナイトは 2 回，ビショップは 4 回かかる．

$\rightarrow$ ビショップに近い集合場所を 2 つ調べればよい！



G: Long Chess Board 部分点 4 解法

- $H = 3$

G: Long Chess Board　部分点 4 解法

- $H = 3$

2 つのコマが十分離れている時を考える (近いときは BFS 解法を使えばよい).

どちらのコマも 1 回の移動による横方向の移動距離は $\max 2$.

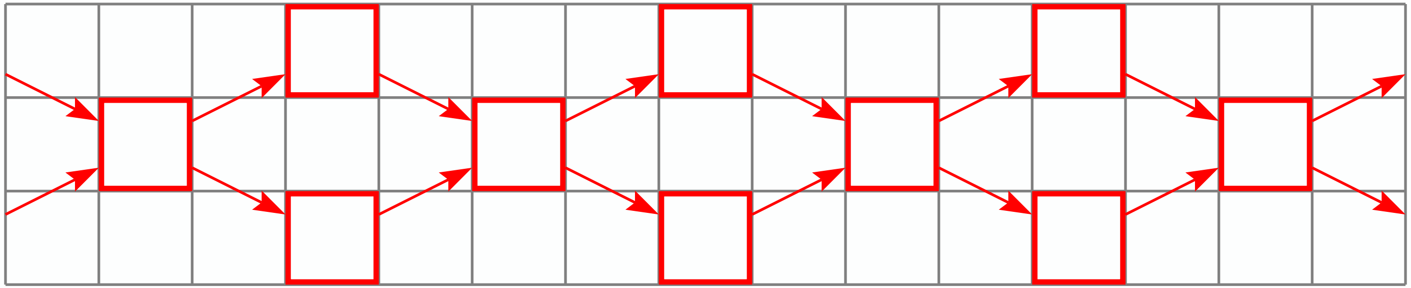
$\rightarrow$ 答えは $\left\lceil \left| c_k - c_b \right| \ / \ 2 \right\rceil + \left(\text{小さい数}\right)$ になることが分かる.

実は，殆どのケースで答えが $\left\lceil \left| c_k - c_b \right| \ / \ 2 \right\rceil$ になる！

G: Long Chess Board 部分点 4 解法

- $H = 3$

ナイトを以下のように動かすことを考えると，ナイトの位置は周期 4 になっている．

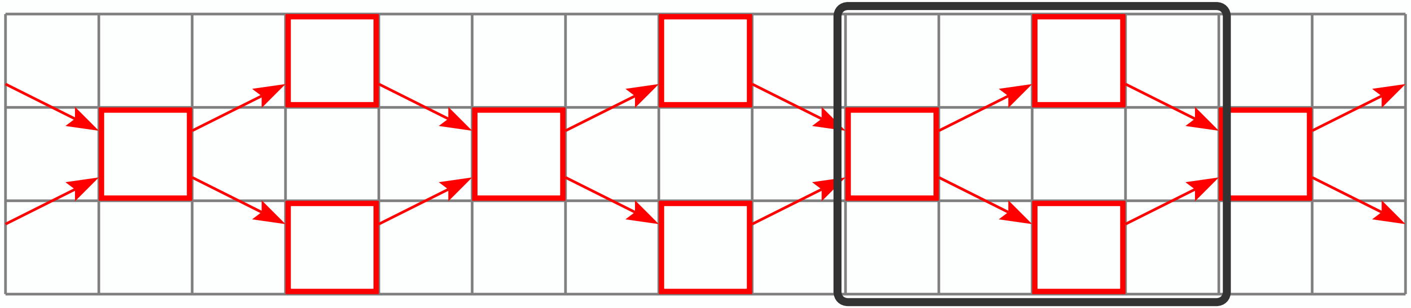


G: Long Chess Board 部分点 4 解法

- $H = 3$

ナイトを以下のように動かすことを考えると，ナイトの位置は周期 4 になっている．

$\rightarrow$ ビショップの初期位置として 3×4 ケースを考えればよい．

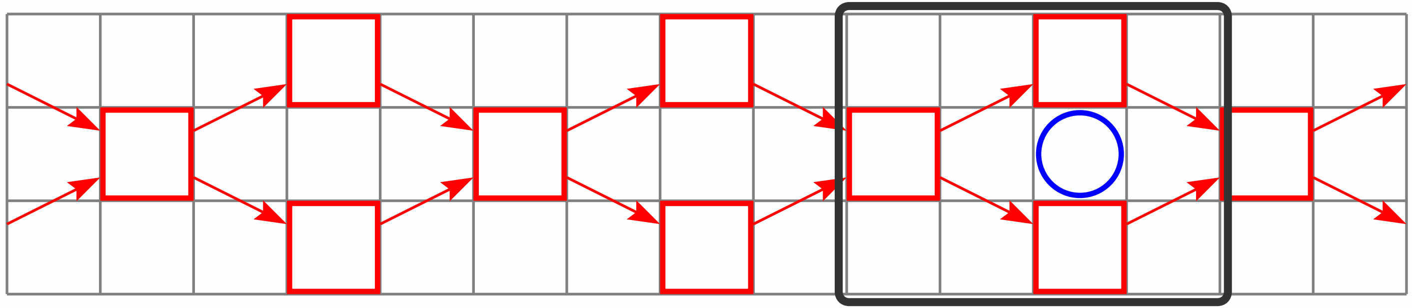


G: Long Chess Board　部分点 4 解法

- $H = 3$

ナイトを以下のように動かすことを考えると，ナイトの位置は周期 4 になっている．

$\rightarrow$ ビショップの初期位置として 3×4 ケースを考えればよい．



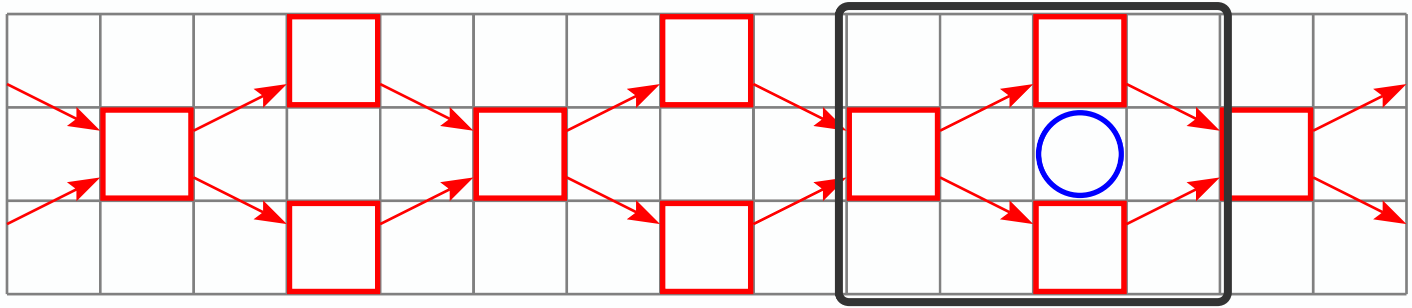
この 12 マスのうち，$\left\lceil \left| c_k - c_b \right| \;/\; 2 \right\rceil$ が達成できないのは 図の位置にビショップがある時のみ．

G: Long Chess Board 部分点 4 解法

- $H = 3$

ナイトを以下のように動かすことを考えると，ナイトの位置は周期 4 になっている．

$\rightarrow$ ビショップの初期位置として 3×4 ケースを考えればよい．



この 12 マスのうち，$\left\lceil \left| c_k - c_b \right| \ / \ 2 \right\rceil$ が達成できないのは 図の位置にビショップがある時のみ．

このとき，答えは $\left\lceil \left| c_k - c_b \right| \ / \ 2 \right\rceil + 1$，それ以外では $\left\lceil \left| c_k - c_b \right| \ / \ 2 \right\rceil$ となる．

G: Long Chess Board 統計情報

AC 数	6 / 96
AC 率	6.0 %
平均点	8.50 点
オンサイト FA	startcpp (2:47:57)
FA	ssrs (1:04:25)

H: LCM and GCD

Writer: yasunori

H: LCM and GCD

黒板に $N \left(2 \leq N \leq 10^5\right)$ 個の正整数 $A_1, A_2, ..., A_N (1 \leq A_i \leq 2 \times 10^5)$ が書かれている．

次の操作を行えなくなるまで繰り返す：

- 黒板に書かれている数を 2 つ選んで消す．消した数を $x,\ y$ として，

  $\mathrm{lcm}(x,\ y)$ と $\gcd(x,\ y)$ を黒板に書き加える．

  - ただし，操作の前後で黒板の数の組み合わせが変化しないような操作は行えない．

操作を行えなくなるまで繰り返したのち，黒板に書かれているすべての数字を昇順に並び替え，各要素を 998244353 で割った余りを求めよ．

H: LCM and GCD 部分点 1 解法

- $N = 2$

H: LCM and GCD　部分点 1 解法

- $N = 2$

操作を 2 回以上行うことはできないため，答えは

$$\{\gcd(A_1, A_2) \bmod 998244353, \mathrm{lcm}(A_1, A_2) \bmod 998244353\}$$

となる．

mod を取らないと WA となるため注意！

たとえば $A = \{199999, 200000\}$ のとき，$\mathrm{lcm}(199999, 200000) = 39999800000$.

H: LCM and GCD 部分点 2 解法

- $N \leq 1000, \ \mathrm{lcm}(A_1, A_2, ..., A_N) \leq 10^{18}$

H: LCM and GCD 部分点 2 解法

- $N \le 1000,\ \mathrm{lcm}(A_1, A_2, ..., A_N) \le 10^{18}$

実は，操作後の数列の最小値は $\gcd(A_1, A_2, ..., A_N)$ となる．

証明 : $g = \gcd(A_1, A_2, ..., A_N)$ とする．

- $A_1, A_2, ..., A_N$ はすべて g の倍数である． また，選んだ 2 つの数がともに g の倍数の時，新たに書き加える数は 2 つとも g の倍数．

  $\to$ 何度操作を行っても，黒板に g の倍数以外が書かれることはない．

- $i = 2, 3, ..., N$ に対して，(A_1, A_i) の 2 つの数を選んで操作を行うことで，g を作ることができる．

H: LCM and GCD 部分点 2 解法

- $N \leq 1000,\ \mathrm{lcm}(A_1, A_2, ..., A_N) \leq 10^{18}$

実は，操作後の数列の最小値は $\gcd(A_1, A_2, ..., A_N)$ となる．

証明 : $g = \gcd(A_1, A_2, ..., A_N)$ とする．

- $A_1, A_2, ..., A_N$ はすべて g の倍数である． また，選んだ 2 つの数がともに g の倍数の時，新たに書き加える数は 2 つとも g の倍数．

  $\rightarrow$ 何度操作を行っても，黒板に g の倍数以外が書かれることはない．

- $i = 2, 3, ..., N$ に対して，(A_1, A_i) の 2 つの数を選んで操作を行うことで，g を作ることができる．

g を除いて同じことを繰り返すことで，答えを求めることができる！

操作回数は $N(N-1)/2$ 回なので，十分高速．

H: LCM and GCD 満点解法

$$
\begin{aligned}
A_1 &= 2^{2} \times 3^{4} \times 5^{5} \\
A_2 &= 2^{6} \times 3^{3} \times 5^{7} \\
&\downarrow \\
\gcd(A_1, A_2) &= 2^{2} \times 3^{3} \times 5^{5} \\
\mathrm{lcm}(A_1, A_2) &= 2^{6} \times 3^{4} \times 5^{7}
\end{aligned}
$$

問題の操作は，素因数ごとに肩の数の大きいほうを集めると lcm に，小さいほうを集めると gcd になることが分かる．

H: LCM and GCD 満点解法

$$
\begin{aligned}
A_1 &= 2^{2} \times 3^{4} \times 5^{5} \\
A_2 &= 2^{6} \times 3^{3} \times 5^{7}
\end{aligned}
$$

$$\downarrow$$

$$
\begin{aligned}
\gcd(A_1, A_2) &= 2^{2} \times 3^{3} \times 5^{5} \\
\mathrm{lcm}(A_1, A_2) &= 2^{6} \times 3^{4} \times 5^{7}
\end{aligned}
$$

問題の操作は，素因数ごとに肩の数の大きいほうを集めると lcm に，小さいほうを集めると gcd になることが分かる．

$\rightarrow$ 肩の数を素因数ごとにソートしたものが答えになる！

H: LCM and GCD 満点解法

たとえば $A = \{20, 75, 144\}$ では，

$$\begin{aligned} A_1 &= 2^{2} \times 3^{0} \times 5^{1} & & & B_1 &= 2^{0} \times 3^{0} \times 5^{0} \\ A_2 &= 2^{0} \times 3^{1} \times 5^{2} & &\to & B_2 &= 2^{2} \times 3^{1} \times 5^{1} \\ A_3 &= 2^{4} \times 3^{2} \times 5^{0} & & & B_3 &= 2^{4} \times 3^{2} \times 5^{2} \end{aligned}$$

これより，操作後の数列は $\{1, 60, 3600\}$ となる．

H: LCM and GCD 満点解法

たとえば $A = \{20, 75, 144\}$ では,

$$\begin{aligned} A_1 &= 2^{2} \times 3^{0} \times 5^{1} & & & B_1 &= 2^{0} \times 3^{0} \times 5^{0} \\ A_2 &= 2^{0} \times 3^{1} \times 5^{2} & &\rightarrow & B_2 &= 2^{2} \times 3^{1} \times 5^{1} \\ A_3 &= 2^{4} \times 3^{2} \times 5^{0} & & & B_3 &= 2^{4} \times 3^{2} \times 5^{2} \end{aligned}$$

これより，操作後の数列は $\{1, 60, 3600\}$ となる.

2×10^5 以下の正整数について，素因数の種類は高々 6 個なため，ソートする要素の個数は最大で $6 \times N$. このため，各素因数について指数が 1 以上であるもののみソートすれば間に合う.

H: LCM and GCD 統計情報

AC 数	29 / 96
AC 率	29.0 %
平均点	34.0 点
オンサイト FA	shogo314 (20:37)
FA	TKTYI (11:29)

I: Deque Inversion

Writer: Nichi10p

I: Deque Inversion

最初，長さ N $(2 \leq N \leq 10^5)$ の数列 A $(-10^5 \leq A_i \leq 10^5)$ がある．

以下のクエリが Q $(1 \leq Q \leq 10^5)$ 個来るので，各クエリ操作後の A の転倒数を求めよ．

- 1 x : A の末尾に x を追加する．
- 2 : A の末尾を取り除く．
- 3 x : A の先頭に x を追加する．
- 4 : A の先頭を取り除く．

ここで，操作の途中で A の長さは 2 より小さくなることはない．

I: Deque Inversion　部分点 1, 2 解法

- $N \leq 100,\ Q \leq 100$,　クエリは 1, 2 のみ

I: Deque Inversion 部分点 1, 2 解法

- $N \leq 100,\ Q \leq 100$, クエリは 1, 2 のみ

末尾に追加 or 削除 クエリが飛んでくる

$\rightarrow$ 可変長配列で管理可能！

- C++ : vector の push_back と pop_back
- Python: list の append と pop

各クエリに従って配列 A を更新後，愚直に転倒数を求めれば間に合う．

I: Deque Inversion　部分点 1, 2 解法

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

末尾に追加 or 削除 クエリが飛んでくる

$\rightarrow$ 可変長配列で管理可能！

- C++：vector の push_back と pop_back
- Python: list の append と pop

各クエリに従って配列 A を更新後，愚直に転倒数を求めれば間に合う．

先頭への操作が来ても ... 配列 A を新たに作り直したりすれば OK.

転倒数を求めるパートで毎回数列 A の要素の個数の 2 乗回程度計算が行われるため，全体で計算量は悪化しない．

I: Deque Inversion　部分点 3 解法

- $N \leq 2 \times 10^4,\ Q \leq 100$

I: Deque Inversion 部分点 3 解法

- $N \leq 2 \times 10^4,\ Q \leq 100$

転倒数を高速に求めたい $\to$ BIT(fenwick tree) で求められる！

転倒数とは，「"自分より左にある，自分より大きな数の個数"の総和」

I: Deque Inversion　部分点 3 解法

- $N \leq 2 \times 10^4,\ Q \leq 100$

転倒数を高速に求めたい $\rightarrow$ BIT(fenwick tree) で求められる！

転倒数とは，「"自分より左にある，自分より大きな数の個数"の総和」

$i = 1, 2, ..., \mathrm{len}(A)$ について，

- A_i より大きい数の個数を BIT により求める．
  - いま BIT で管理しているのは「自分より左にある数」
- BIT の A_i の値を 1 増やす．

とすれば，転倒数が求められる．

このままだと負の数が出てくるが，あらかじめ 10^5 を足しておけばよい．

I: Deque Inversion 満点解法

I: Deque Inversion　満点解法

操作により転倒数がどう変化するかを考える.

- ある数 x が A の末尾に追加されるとき

  $\rightarrow$ 転倒数は「A のうち x より大きい数の個数」ぶん増える.

- ある数 x が A の末尾から取り除かれるとき

  $\rightarrow$ 転倒数は「A のうち x より大きい数の個数」ぶん減る.

- ある数 x が A の先頭に追加されるとき

  $\rightarrow$ 転倒数は「A の x より小さい数の個数」ぶん増える.

- ある数 x が A の先頭から取り除かれるとき

  $\rightarrow$ 転倒数は「A の x より小さい数の個数」ぶん減る.

I: Deque Inversion　満点解法

操作により転倒数がどう変化するかを考える.

- ある数 x が A の末尾に追加されるとき

  $\rightarrow$ 転倒数は「A のうち x より大きい数の個数」ぶん増える.
- ある数 x が A の末尾から取り除かれるとき

  $\rightarrow$ 転倒数は「A のうち x より大きい数の個数」ぶん減る.
- ある数 x が A の先頭に追加されるとき

  $\rightarrow$ 転倒数は「A の x より小さい数の個数」ぶん増える.
- ある数 x が A の先頭から取り除かれるとき

  $\rightarrow$ 転倒数は「A の x より小さい数の個数」ぶん減る.

A の各値の個数を BIT を使って管理すれば，高速にこの問題を AC できる！

I: Deque Inversion 統計情報

AC 数	62 / 96
AC 率	62.0 %
平均点	62.70 点
オンサイト FA	AngrySadEight (23:16)
FA	dyktr_06 (8:16)

J: Counting Zig Zag Sequence

Writer: Ackvy

J: 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 $\rightarrow$ 減少 $\rightarrow$ 増加 $\rightarrow \cdots$ または 減少 $\rightarrow$ 増加 $\rightarrow$ 減少 $\rightarrow \cdots$

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.

J: Counting Zig Zag Sequence 部分点 1 解法

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

J: Counting Zig Zag Sequence 部分点 1 解法

- $N \le 5,\ K \le 5$

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，

$N \le 5,\ K \le 5$ 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.

J: Counting Zig Zag Sequence 部分点 2 解法

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

J: 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．

J: 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：

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

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

J: 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．

J: 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,

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

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

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

J: 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．

J: 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，$N \le 5000,\ K \le 5000$ 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.

J: Counting Zig Zag Sequence 統計情報

AC 数	63 / 96
AC 率	63.0 %
平均点	64.60 点
オンサイト FA	n_o_n_o_n (14:31)
FA	potato167 (9:10)

K: Colorful Balls

Writer: Ackvy

K: Colorful Balls

左のほうが空いている筒と，1 以上 10^9 以下の整数で表される色が塗られたボールがたくさんある．

以下のクエリを処理せよ．

- 1 x c: 色 c のボールを左から x 個追加する．
- 2 x: 現在の筒の左から $1, 2, ..., x$ 番目のボールを取り除く．
- 3 x: 現在の筒の左から x 番目のボールの色を出力する．

K: Colorful Balls 部分点 1 解法

- $Q \leq 5000,$ クエリ 1, 2 に対して常に $x = 1$

K: Colorful Balls 部分点 1 解法

- $Q \leq 5000,$ クエリ 1, 2 に対して常に $x = 1$

ボールが 1 つの更新しかないので，そのままシミュレーションすれば OK.

クエリが高々 5000 回しか来ないので，毎回愚直に先頭にボールを追加しても間に合う.

K: Colorful Balls 部分点 2 解法

- $Q \leq 5000$

K: Colorful Balls 部分点 2 解法

- $Q \leq 5000$

一度にたくさんのボールの追加があるため，愚直にシミュレーションすると MLE / TLE してしまう．

$\rightarrow$「i 番目に，c_i 色のボールが a_i 個ある」という情報を持っておけば，愚直にシミュレーション可能！

K: Colorful Balls　満点解法

部分点 2 と同様,「i 番目に, c_i 色のボールが a_i 個ある」という情報を持っておく.

- 逆から見れば，クエリ 1 は末尾に追加の操作になるため，高速.
- クエリ 2 で消す回数は，クエリ 1 の個数で抑えられるため問題ない.

K: Colorful Balls　満点解法

部分点 2 と同様，「i 番目に，c_i 色のボールが a_i 個ある」という情報を持っておく．

- 逆から見れば，クエリ 1 は末尾に追加の操作になるため，高速.
- クエリ 2 で消す回数は，クエリ 1 の個数で抑えられるため問題ない.

クエリ 3 が問題だが ...

$\rightarrow$ ボールの個数の累積和を持っておくことで，二分探索できる！

個数の総和 $- x$ を二分探索すれば OK.

K: Colorful Balls 統計情報

AC 数	52 / 96
AC 率	52.0 %
平均点	54.30 点
オンサイト FA	cyanorless_tanto (12:14)
FA	potato167 (6:29)