Xmas Contest 2014

Solutions

Problem A: A+B Problem

ループ処理を記述できないため, 足し算のアルゴリズムの時間計算量がプログラムの行数に大きく影響する. まず, 部分点解法におけるいくつかの重要な操作を示す.

- m を十分大きい値とする. $x \geq 1$ のとき, $\left\lfloor \frac{mx}{m+1} \right\rfloor = x - 1$ を用いて 1 を引く操作ができる.
- $x \geq 3$ のとき, $\left\lfloor \frac{x^2}{x-1} \right\rfloor = x + 1$ を用いて 1 を足す操作ができる.
- $y > 0$ のとき, $\left\lfloor \frac{x}{y} \right\rfloor$ の値を調べることで $x \geq y$ かどうかを判定できる.

$O(A + B)$

例えば次のような方法が考えられる.

- 最初に $C := A + 3$ として (場合分けを要する), $i = 1, 2, \ldots, 100$ に対して「$B \geq i$ なら $C := C + 1$ とする」として, 最後に $C := C - 3$ とする.
- $A \leq B$ となるように入れ替える. $C := 2B$ とし, $A < B$ である間 $B := B - 1$, $C := C - 1$ とする (実際はプログラムに 100 回書く). この方法はループ中で 1 を足す操作を避けている.

$O(\log A + \log B)$

$2^{31}x$ のオーバーフローを利用し x の偶奇を判定できる. これを利用し, 例えば次の方法が考えられる.

- 最初に $C := 2$ として, $i = 6, 5, \ldots, 0$ に対して「$C := 2C$ とする, $\left\lfloor \frac{A}{2^i} \right\rfloor$ が奇数ならば $C := C + 1$ とする, $\left\lfloor \frac{B}{2^i} \right\rfloor$ が奇数ならば $C := C + 1$ とする」として, 最後に $C := C \bmod 2^8$ とする (2^{24} をかけてオーバーフローさせて 2^{24} で割る).

$O(1)$

重要なアイデアは, 式 $(sA \pm t)(sB \pm t) = s^2AB \pm st(A + B) + t^2$ を用い, s^2AB や t^2 をオーバーフローまたは切り捨てで消すことである. 例えば次のように実現できる (どちらも 8 行のプログラムが書ける).

- $s = 2^8$, $t = 1$ と選ぶ. $D := \left\lfloor \frac{(2^{24} - 2^8)A}{2^{16}} \right\rfloor = 2^8A - 1$, $E := \left\lfloor \frac{(2^{24} - 2^8)B}{2^{16}} \right\rfloor = 2^8B - 1$, $F := \left\lfloor \frac{DE}{2^8} \right\rfloor = 2^8 - (A + B)$, $C := \frac{(2^{32} - 2^{24})G \bmod 2^{32}}{2^{24}} = A + B$.
- $s = 1$, $t = 2^{16}$ と選ぶ[*1]. $D := \frac{(2^{32} - 2^{16})A \bmod 2^{32}}{2^{16}} = 2^{16} - A$, $E := \left\lfloor \frac{2^{32} - 1}{D} \right\rfloor = 2^{16} + A$, $F := \frac{(2^{32} - 2^{16})B \bmod 2^{32}}{2^{16}} = 2^{16} - B$, $G := \left\lfloor \frac{2^{32} - 1}{F} \right\rfloor = 2^{16} + B$, $C := \left\lfloor \frac{EG \bmod 2^{32}}{2^{16}} \right\rfloor = A + B$.

[*1] コンテスト中の semiexp さんの解法による.

Problem B: Bad Sort?

クイックソートの一種であるが, 数列に同じ要素があるときに性能が悪くなるソートアルゴリズムに関する問題である.

出力形式に関しては, 基本的には整数を $\bmod P$ で計算し, 割り算を行うときに, a で割る代わりに a の $\bmod P$ での乗法の逆元 ($ab \equiv 1 \pmod P$ となる b) をかければよい. $P = 10\,007$ は素数であり, 割り算の除数は N 以下であるから, これは常に行える. 予め $\bmod P$ での乗法の逆元を計算しておくとよい.

B1

問題文に書かれている関数 Sort を少し修正し, かかる時間の期待値を返す関数にすればよい. 具体的には, p を一様ランダムに選ぶ代わりにすべての p を試して, 再帰呼び出しの結果の平均をとるようにする.

B2

数列 $(x_1, \ldots, x_n)$ が $x_1 \leq \cdots \leq x_n$ を満たすとき, 関数 Sort が選んだ p に対して, $x_{p'} = x_p$ となる最小の p' をとると, $A = (x_1, \ldots, x_{p'-1})$, $B = (x_{p'+1}, \ldots, x_n)$ となる.

このことを利用すると, 動的計画法による解法を考えることができる. 入力の $(X_1, \ldots, X_N)$ は昇順にソートしておく. $f(i,j)$ を $(X_i, X_{i+1}, \ldots, X_{j-1}, X_j)$ に対する関数 Sort のかかる時間とする. $i > j$ のとき $f(i,j) = 0$ であり, $i \leq j$ のときは

$$f(i,j) = (j-i+1) + \frac{1}{j-i+1} \sum_{k=i}^{j} (f(i, l_{i,k}-1) + f(l_{i,k}+1, j))$$

である. ただし, $l_{i,k}$ は $X_l = X_k$ かつ $i \leq l$ なる最小の l とする.

この式を用いて $j-i$ の小さい順に $f(i,j)$ を求めていくことができ, 答えは $f(1,N)$ である. 各 $f(i,j)$ の計算は $O(j-i)$ 時間で行えるので, 全体の時間計算量は $O(N^3)$ となる.

B3

上の $f(i,j)$ の式の和の部分を高速に計算したい. $f_1(i,j) = \sum_{k=i}^{j} f(i, l_{i,k}-1)$, $f_2(i,j) = \sum_{k=i}^{j} f(l_{i,k}+1, j)$ とおく. $j-i$ の小さい順に, $f_1(i,j)$, $f_2(i,j)$, $f(i,j)$ を並行して計算していく.

$f_1(i,j)$ は $f_1(i,j-1)$ から効率よく求められる. $f_1(i,j) = f_1(i,j-1) + f(i, l_{i,j}-1)$ である. $l_{i,k} = \max\{l_{1,k}, i\}$ なので, 予め $l_{1,k}$ を $1 \leq k \leq N$ についてすべて求めておけばよい.

$f_2(i,j)$ は $f_2(i+1,j)$ から効率よく求められる. $i+1 \leq k \leq j$ なる k について, $X_i < X_k$ ならば $l_{i,k} = l_{i+1,k}$ であり, $X_i = X_k$ ならば $l_{i,k} = i$, $l_{i+1,k} = i+1$ である. よって, $X_i = X_k$ である k の個数を m をおくと, $f_2(i,j) = f_2(i+1,j) - mf(i+2,j) + (m+1)f(i+1,j)$ である. ここで, $m = r_{i,j} - i$ となる. ただし, $r_{k,j}$ は $X_k = X_r$ かつ $r \leq j$ なる最大の r とする. $r_{k,j} = \min\{l_{k,N}, j\}$ なので, 予め $r_{k,N}$ を $1 \leq k \leq N$ についてすべて求めておけばよい.

以上により各 $f_1(i,j)$, $f_2(i,j)$, $f(i,j)$ の計算を定数時間で行えるので, 全体の時間計算量は $O(N^2)$ となる.

Problem C: Count Your Way

$P = 2\,014\,122\,419$ は素数である. 以下の解説は P が一般の素数の場合に成り立つ.

与えられた有向グラフの長さ P の閉ウォークの個数を $\bmod P$ で求める問題である. グラフの隣接行列を A とおく (すなわち, A は N 次正方行列であり, (u,v) 成分は u を始点, v を終点とする辺の本数である). このとき, 非負整数 k に対し, A^k の (u,v) 成分は u を始点, v を終点とする長さ k のウォークの個数に等しい (行列の積の定義を考えよ). よって, A^P の $(1,1)$ 成分, ..., (N,N) 成分の和を $\bmod P$ で求めればよい.

C1

$A^P \bmod P$ を繰り返し二乗法で求めれることができる ($A^1, A^2, A^4, A^8, \ldots$ を順次求める). 積をとるたびに各成分を $\bmod P$ で求めればよく, 行列積を $O(N^3)$ 時間で行えば, 全体で $O(N^3 \log P)$ 時間で計算できる.

C2

求めたい値は $(\mathrm{tr}(A^P)) \bmod P$ である (tr は行列のトレース). 実は, 以下で示すように $\mathrm{tr}(A^P) \equiv \mathrm{tr}A \pmod P$ が成り立つ. これは Fermat の小定理の一種の拡張と考えることができる.

A の各成分が整数なので $\mathrm{tr}A$ は整数であり, Fermat の小定理から $(\mathrm{tr}A)^P \equiv \mathrm{tr}A$ である. よって, $(\mathrm{tr}A)^P \equiv \mathrm{tr}(A^P) \pmod P$ を示せばよい. A の固有値を重複度を込めて $\lambda_1, \ldots, \lambda_N \in \mathbb{C}$ とする. A^P の固有値は重複度を込めて $\lambda_1^P, \ldots, \lambda_N^P$ である (例えば Jordan 標準形を考えよ). トレースは固有値の和であるから, 示すべき式は

$$(\lambda_1 + \cdots + \lambda_N)^P \equiv \lambda_1^P + \cdots + \lambda_N^P \pmod P$$

となる. P が素数であるから, $(\lambda_1 + \cdots + \lambda_N)^P$ を展開したとき現れる係数 $\dfrac{P!}{e_1! \cdots e_N!}$ (ただし $e_1 + \cdots + e_N = P$) は, $e_1, \ldots, e_N < P$ のとき P の倍数である. よって, ある整数係数 N 変数多項式 f によって

$$(\lambda_1 + \cdots + \lambda_N)^P = (\lambda_1^P + \cdots + \lambda_N^P) + P \cdot f(\lambda_1, \ldots, \lambda_P)$$

と書ける. ここで $\lambda_1, \ldots, \lambda_N$ は代数的整数 (最高次係数が 1 の整数係数多項式の根) であるから, $f(\lambda_1, \ldots, \lambda_N)$ も代数的整数である. 一方, $\mathrm{tr}A$, $\mathrm{tr}(A^P)$ は整数であるから, 上の式より $f(\lambda_1 + \ldots + \lambda_N)$ は有理数である. これらより $f(\lambda_1, \ldots, \lambda_N)$ が整数であることが従い, 所望の合同式を得る.

以上のことより, 本問の答えは $(\mathrm{tr}A) \bmod P$, すなわちグラフの自己ループの個数を P で割った余りである.

Problem D: Distinguished Furry Animals

与えられた 2 個の文字列を区別する決定性有限オートマトン (DFA) のうちの状態数の最小値を求める問題である.

2 個の文字列が等しいとき, そしてその時に限り, それらは区別できないので答えは -1 となる.

D1

文字 1 種類しかないとき, 遷移できる状態はグラフで表すと ρ 字の形をになる. 正確には, ある非負整数 a と正整数 m について, a 回の遷移後に長さ m の周期に入る. このとき, 長さ L_1 の文字列と長さ L_2 の文字列が区別できるための条件は $L_1 < a$ または $L_2 < a$ または $L_1 \not\equiv L_2 \pmod{m}$ である. よって区別するための状態数の最小値として $L_1 + 2$, $L_2 + 2$, $L_1 - L_2$ を割り切らない最小の正整数 m が候補となり, これらのうち最も小さいものが答えとなる. あるいは, 適切な範囲の a, m をすべてに対して条件を調べてもよい ($1, \ldots, 9$ の最小公倍数が 2520 なので, $m \leq 8$ を調べればよい).

D2

文字が 2 種類のとき, 長さ 18 以下の異なる 2 個の文字列は, 高々 5 状態の DFA で区別できる. この上限が小さいことは, 例えば以下のようなことから推測できるかもしれない.

- 文字が 1 種類の場合の考察より, 長さ, あるいはいずれかの文字を含む個数が異なる場合, その差は 17 以下であるから, 5 状態で区別できる.
- 先頭 3 文字が異なる場合は 5 状態で区別できる.
- 末尾 4 文字が異なる場合は 5 状態で区別できる (パターンマッチオートマトンを構成すればよい).

よって, 4 状態以下のオートマトンをすべて試せばよい. n 状態のオートマトンは n^{2n} 通りであるから, 現実的な時間で実行することができる (実装にもよるが, 数分程度でできる). 遷移できない状態があるオートマトンを予め除いておく, といった多少の高速化も可能である.

数分のプログラムの実行で, 長さ 18 以下のすべての文字列を 4 状態以下のオートマトンで区別できるかという基準で分類することもできる. 4 状態で区別できない組は, 2 つ組が 58 個, 3 つ組が 4 個あり (すべて入力に含めてある), これらが 5 状態で区別できることは上述のいずれかの場合としてわかる.

Problem E: Economic Plan

空港を頂点, 廃止が決定していない航空便を辺としたグラフを考えると, 求めるものはグラフの辺連結度 (あるいは全域最小カット) から 1 を引いた値である.

E1

一般のグラフの全域最小カットを $O(N^3)$ 時間で求めるアルゴリズムを適用することができる. あるいは, 頂点 1 を始点, 他の頂点を終点とする最大流を求めることで $N-1$ 回の最大流計算で求めてもよい.

E2

制約から, 各頂点の次数は $\left\lfloor \frac{1}{2}N \right\rfloor$ 以上である. この条件下では, 全域最小カットは次数の最小値に等しいことを示す.

グラフを $G=(V,E)$ とし, 次数の最小値を $\delta \geq \left\lfloor \frac{1}{2}N \right\rfloor$ とする. カット $(S, V \setminus S)$ を考える. ここで $|S| \leq \left\lfloor \frac{1}{2}N \right\rfloor$ としてよい. するとこのカットのコストは,

$$\sum_{u \in S} \deg u - 2 \cdot |\{uv \in E \mid u,v \in S\}| \geq \delta|S| - |S|(|S|-1) \geq (\delta - |S|)(|S|-1) + \delta \geq \delta$$

と評価できる. 一方, S として 1 頂点 $\{u\}$ をとればカットのコストは $\deg u$ となるから, コスト δ の最小カットは存在する.

以上のことより, 本問の答えは $\delta - 1$ である. 特に, 与えられるグラフはすべて連結である.

なお, サンプルから, あるいは上の式からわかる通り, $|S| = \left\lfloor \frac{1}{2}N \right\rfloor$ のときにもカットのコストが δ になることはある.

Problem F: Fake Distribution

しろうさの分布は平均 0 分散 $\frac{1}{2}$ の正規分布である (Box-Muller 法として知られている).

一方, くろうさの分布は平均 0 分散 $\frac{1}{2}$ であるが, 正規分布ではない. 重要な特徴の 1 つとして, 値は開区間 $(-3, 3)$ に分布することが挙げられる. なお, 一様分布の和は Irwin-Hall 分布とも呼ばれている.

ヒューリスティックな解法

実際に分布をヒストグラムなどで図示することで, 2 つの分布は平均の周りの尖りが異なることがわかる. 確率変数 X の尖度は, $\mu = \mathbb{E}[X]$, $\mu_2 = \mathbb{E}[(X-\mu)^2]$, $\mu_4 = \mathbb{E}[(X-\mu)^4]$ としたとき値 $\frac{\mu_4}{\mu_2^2} - 3$ であるが, 実際, しろうさの分布 (正規分布) の尖度は 0, くろうさの分布 (一様分布 6 個の和) の尖度は $-\frac{1}{5}$ である.

与えられた $X_1, \ldots, X_N$ の標本尖度を求め, 尖度が大きい半分のテストケースに `White` と, 残りに `Black` と答える, という解法が考えられ, これで正解することができる.

コンテスト中には, より高次のモーメント ($\mathbb{E}[X^k]$) に注目する方法, あるいは絶対値が一定以上の値の個数に注目する方法, など様々なものが見られた.

尤度計算による解法

本問では, 正解の確率を最大化することができる.

しろうさの分布, くろうさの分布の確率密度関数をそれぞれ p_1, p_2 とする. これは, 各分布で確率変数の値が $[x, x+dx)$ に含まれる確率が $p_1(x)dx$, $p_2(x)dx$ であることを意味する. よって, N 個の値 $X_1, \ldots, X_h$ が得られる確率の比は $\prod_{i=1}^{N} p_1(X_i)$ と $\prod_{i=1}^{N} p_2(X_i)$ の比に等しい (これらの値は尤度と呼ばれる. 確率分布が離散的な場合はこれらが直接になるのでわかりやすいだろう). 制約から, しろうさの分布とくろうさの分布は等確率で選ばれたと考えてよいので, 値 $X_1, \ldots, X_N$ が得られたという条件下での元の分布が選ばれた確率の比もこの比に等しい.

正規分布の確率密度関数はよく知られており, $p_1(x) = \frac{1}{\sqrt{\pi}} \exp(-x^2)$ である.

$p_2(x)$ を求めよう. 累積分布関数 F_2 ($F_2(x)$ は値が x 以下である確率) を経由する. $F_2(x)$ は 6 次元立方体 $(0,1)^6$ の $u_1 + \cdots + u_6 \leq x + 3$ の部分の体積に等しいが, 包除原理により $F_2(x) = \sum_{k=0}^{6} (-1)^k \binom{6}{k} \frac{1}{6!} (\max\{(x+3) - k, 0\})^6$ と求められる ($(0,1) = (0,\infty) \setminus (1,\infty)$ と考えよ). したがって $p_2(x) = \frac{d}{dx} F_2(x) = \sum_{k=0}^{6} (-1)^k \binom{6}{k} \frac{1}{5!} (\max\{(x+3) - k, 0\})^5$ である.

以上を用いて, $\prod_{i=1}^{N} p_1(X_i)$, $\prod_{i=1}^{N} p_2(X_i)$ を計算できる. 実際には $\sum_{i=1}^{N} (\log p_1(X_i) - \log p_2(X_i))$ を計算するのがよいだろう (ただし $p_2(X_i) = 0$ となる場合に注意せよ).

各テストケースに対して尤度が大きい方の分布を答えた場合, 入力 F1 ではすべてのテストケース, F2 では 100 個中 84 個のテストケースに正解することができる. なお, 実際に確率を計算をすると 96.4 % の確率で 80 個以上のテストケースに正解すると見積もることができる.

Problem G: Geometry Lover

与えられた 3 本の紐による組み紐群 (braid group) B_3 の元 2 個が等しいかどうか判定する問題である. $i = 1, 2$ に対し, σ_i を i 本目の紐を $i+1$ 本目の紐をひねる操作とすると (逆向きは σ_i^{-1}), B_3 は σ_1, σ_2 で生成され, 関係式は $\sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_1\sigma_2$ である : $B_3 = \langle \sigma_1, \sigma_2 \mid \sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_1\sigma_2 \rangle$.

入力は 3 行目と 7 行目のみを読み, 重なり方を σ_i または σ_i^{-1} に読み替えればよい.

手作業

実際に紐を 3 本用意して実験すればよい. 入力 G1 に対して正解できるだろう.

Reidemeister 移動の繰り返し

Reidemeister 移動 II に対応する $\sigma_1\sigma_1^{-1} = \sigma_2\sigma_2^{-1} = 1$ および Reidemeister 移動 III に対応する $\sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_1\sigma_2$ を用いて w から w' に変形できるかどうかを探索する. 入力 G1 に対して正解できるだろう.

群論ライブラリ

数式処理システム Sage には, 組み紐群を処理するライブラリがある.

```
B.<a, b> = BraidGroup(3)
print 'YES' if (a * b * a == b * a * b) else 'NO'
```

標準形

$\Delta = \sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_1\sigma_2$ とおく (この元は half twist と呼ばれる). Δ^2 (full twist と呼ばれる) は, 紐の位置の置換を考えると恒等置換になっている (実は $B_3 \twoheadrightarrow S_3$ の核は Δ^2 で生成される), B_3 の任意の元と可換である (実は B_3 の中心は Δ^2 で生成される), などの特徴をもつ.

σ_i, σ_i^{-1} たちの積として表示された元に以下の手続きを行うことを考える.

1. $\sigma_1^{-1} = \Delta^{-1}\sigma_2\sigma_1$, $\sigma_2^{-1} = \Delta^{-1}\sigma_1\sigma_2$ により σ_i^{-1} を消す.
2. $\sigma_1\Delta^{-1} = \Delta^{-1}\sigma_2$, $\sigma_2\Delta^{-1} = \Delta^{-1}\sigma_1$ により Δ^{-1} を前に括り出す.
3. 先頭から $\sigma_1\sigma_2\sigma_1$ または $\sigma_2\sigma_1\sigma_2$ の並びを探し, $\sigma_1\Delta = \Delta\sigma_2$, $\sigma_2\Delta = \Delta\sigma_1$ により Δ を前に括り出す.

この操作により, B_3 の任意の元を $\Delta^k x$ (k は整数, x は σ_1, σ_2 たちの積で途中に Δ を含まない) という形に書くことができる. この形は, 等しい元に対しては一意的であることが知られているので, 直接比較することができる. 時間計算量は $O(N^2)$ である.

群の表現

B_3 については Burau representation という忠実な表現が知られている[*2] (紐が 4 本以上のときは同様の方法は使えない).

群準同型 $B_3 \to GL(M_2(\mathbb{Z}[X, X^{-1}]))$ (M_2 は 2×2 行列全体) を $\sigma_1 \mapsto \begin{pmatrix} -X & 1 \\ 0 & 1 \end{pmatrix}, \sigma_2 \mapsto \begin{pmatrix} 1 & 0 \\ X & -X \end{pmatrix}$ で定めると, これは well-defined な単射になることが知られている. X に適当な値を代入し, 十分大きな素数 p をとって $\bmod p$ で与えられた元に対応する行列を計算することで, 高い確率で正しく判定できる.

[*2] コンテスト中の rng_58 さんのアイデアを参考にした.

Problem H: Hack Me

入力として与えられるファイルはモノクロの BMP (Windows ビットマップ) であり, 元のファイル中の各数字をそれぞれ別の全角記号に対応させてから画像としたものである. フォントはＭＳ ゴシック 24 pt, 解像度は 300 ppi である (したがって画像においては 1 文字あたり 100 px × 100 px である). 行と行の間には 75 px の空白が空いている.

このことは例えば以下の方法で気づくことができるかもしれない.

- バイナリファイルであることが明記されているので, バイナリエディタで開く. 先頭 2 バイトが `BM` になっており, これは BMP ファイルの識別子である.
- `file` コマンドを利用する.
- 一部 OS のプレビュー機能を利用して画像であることを発見する.
- 一部ウェブブラウザで開いて確認する.

H1

画像であることがわかっていれば, 対応するサンプル出力は D 問題のものとわかる.

画像であることに気づかずとも, 配布されたサンプル出力ファイルのいずれかを選んでそのまま提出すれば, 高々 7 回の提出で正解できることが保証されている.

H2

まず全角文字の画像を扱いやすいデータ (整数を割り当てて配列にする, あるいはテキストファイルにするなど) にする必要がある. これは例えば以下の方法でできる.

- 画像を開いて目で見てテキストに起こす.
- (各文字の領域が整数ピクセルであることを確認した上で,) 実際に BMP ファイルをプログラムで読み取る. 一部の言語ではライブラリを使用することもできるし, 無圧縮の BMP は比較的単純な構造なので, 直接バイナリファイルを読む方法でもよい (4 バイト境界のパディングに特に注意せよ).
- ウェブ上の OCR サービスを利用する[*3].

その後数字への対応を求めることになるが, H1 の解答からわかる 2 および 3 の対応, および, 制約からわかる 2 桁以上なら先頭が 0 ではないこと, 値が 2^{32} 未満であることを用いて特定できる. 10! 通りの対応をプログラムで総当たりして条件を満たすかどうか調べるのが最も簡単だろう.

正解の対応を以下に示しておく.

0	1	2	3	4	5	6	7	8	9
□	☆	▽	△	※	§	○	〒	◎	◇

[*3] コンテスト中の gachizei_DDR さんの解法による.