Xmas Contest 2012

Solutions

練習ラウンド

Problem X: Say Yes

問題概要

テストケースの個数だけ YES と出力する.

解答例 (bash)

```
read t
yes YES | head -n $t
```

Problem Y: Factorization

問題概要

与えられた 2 以上 10^{18} 以下の整数を素因数分解する.

解答例 (bash)

```
read t
for i in `seq 1 $t`; do
  read n
  factor $n | sed -e 's/.*: //'
done
```

Problem A: Abundant Dice

手作業による解法

A1.in ではテストケースが 10 個のみであるから, 実際にさいころを組み立ててしまう (あるいはそれを想像する) ことで解くことは難しくない.

解法 1

さいころを回転させた状態は 24 通りである. 始めにそれらの展開図のすべて生成し, 各入力と比較すればよい.

ある状態から始めて 24 通りを生成する方法として, 例えば次が知られている. さいころは xyz 平面で座標軸の向きに置かれているとする.

for $i := 1$ **to** 6:
  for $j := 1$ **to** 4:
    さいころの状態を出力する.
    z 軸を回転軸として 90° 回転させる.
  if i が奇数 **then**:
    x 軸を回転軸として 90° 回転させる.
  else:
    y 軸を回転軸として 90° 回転させる.

解法 2

まず向かい合う面の和が 7 であるかを確認する. すると作りたいさいころかそれと鏡面対称のものに限られることがわかるので, これを利用して判定する.

例えば, ある頂点に集まる 3 面 (A, B, C) が正しい順に並んでいるかを調べればよい. 正しい順は 24 通りあるが, これらを手作業で列挙することは比較的容易である.

作りたいものであるか鏡面対称のものであるかは, 向かい合う面の目を交換することで入れ替わる. これを利用し, $\{A, B, C\} = \{1, 2, 3\}$ の場合に帰着させることで実装を簡潔にすることもできる.

Problem B: Branched DNA

塩基, DNA 鎖を単に文字, 文字列と呼び, 文字 x と貼り合わせられる文字を x' と書く. A と B を貼り合わせられる最大の長さを K とする ($K < L$, $K < M$). 条件を満たす C は, A と B を k 文字 ($1 \leq k \leq K$) 貼り合わせたもの $C^{(k)} = B'_M B'_{M-1} \cdots B'_{k+2} B'_{k+1} A'_{L-k} A'_{L-k-1} \cdots A'_2 A'_1$ で, K 通り考えられる. 以下 $K > 0$ とする.

解法 1

上記の K 通りを実際に作り, 単純な比較により辞書順最小の文字列を求める. 時間計算量は $O(K(L+M))$ となり, 実行に長い時間をかければ解くことができる.

解法 2

前計算を行うことで, $C^{(k)}$ の部分文字列の rolling hash は $O(1)$ 時間で求めることができる. 2 個の $C^{(k)}$ に対し, ハッシュが衝突しないと仮定すれば, 最初に異なる文字を二分探索で $O(\log(L+M))$ 時間で求めることができる (長さを決めたとき接頭辞のハッシュ値を比較すればよい). よって比較が $O(\log(L+M))$ 時間で行えるので, 全体で $O(K \log(L+M))$ 時間となる.

解法 3

$C^{(K)}, C^{(K-1)}, \ldots, C^{(k+1)}, C^{(k)}$ のうち辞書順最小のものを $C^{*(k)}$ とし, $C^{*(K)}, C^{*(K-1)}, \ldots, C^{*(2)}, C^{*(1)}$ を順次求めることとする (すなわち, 長さが短い順に比較していく).

$C^{(k+1)}$ と $C^{(k)}$ の先頭 $M-k$ 文字は等しいため, $C^{*(k+1)}$ と $C^{(k)}$ の先頭 x 文字が等しいならば $C^{*(k+1)}$ と $C^{(k)}$ の先頭 $\min\{x, M-k\}$ 文字も等しい, という事実を利用し, 以下のアルゴリズムを考えられる.

> $C^{*(K)} := C^{(K)}$.
> $x := 1$.
> **for** $k := K-1$ **downto** 1:
> **loop**:
>   **if** $x > M-k$ **then**:
>   **if** $C^{*(k+1)}_x C^{*(k+1)}_{x+1} \cdots < C^{(k)}_x C^{(k)}_{x+1} \cdots$ **then**: $C^{*(k)} = C^{*(k+1)}$. **else**: $C^{*(k)} = C^{(k)}$.
>   **break**.
>   **else if** $C^{*(k+1)}_x \neq C^{(k)}_x$ **then**:
>   **if** $C^{*(k+1)}_x < C^{(k)}_x$ **then**: $C^{*(k)} = C^{*(k+1)}$. **else**: $C^{*(k)} = C^{(k)}$.
>   **break**.
>   **else**:
> $x := x + 1$.

x は「何文字目まで比較したか」を表すパラメータである. $x > M-k$ となったとき x 文字目以降の文字列をまとめて比較しているが, それらはともに $A'_{L-k} A'_{L-k-1} \cdots A'_2 A'_1$ の接尾辞であるため, 事前に suffix array を構築しておく ($O(L(\log L)^2)$ 時間で十分, $O(L)$ 時間も可能である) ことで $O(1)$ 時間で処理できる. x は減少せず M 以下であるため, 時間計算量は $O(K+M)$ となっている.

Problem C: Colorful Lights

特殊な制約があるグラフの頂点彩色数とその彩色数で頂点を塗り分ける方法を数え上げる問題である．この問題は一般のグラフでは NP 困難である．

指数時間解法

グラフの頂点の部分集合 S と整数 k に対し，S を k 色で塗り分ける方法が「1 通り以上あるか」「mod 20 122 224 で何通りか」を動的計画法で求めることができる．ある 1 色で塗ることができる頂点集合は独立集合であり，それらは $O(2^N N^2)$ 時間で最初に列挙することができる．

時間計算量は，単純な方法では $O(4^N N)$ となり，部分集合を効率的に列挙する方法により $O(3^N N)$ に高速化することが可能である．

多項式時間解法

この問題のグラフに対する制約は，cograph であることと同値である．cograph には同値な定義が数多くあるが，詳細は割愛する[*1]．

cograph の重要な性質として，2 頂点以上の任意の誘導部分グラフに対し，それまたはその補グラフが非連結である，というものがある．このことを利用し，頂点集合 S で誘導される部分グラフの彩色数 $f(S)$ と，各 k に対しちょうど k 色で塗り分ける方法 $g(S,k)$ を，以下のように再帰的に求めることができる．

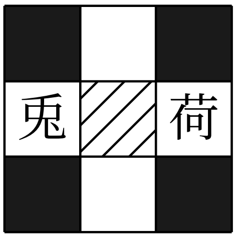
- $|S| = 1$ のとき
  $f(S) = 1$ であり，$g(S,1) = 1$, $g(S,k) = 0$ $(k \neq 1)$ である．
- $|S| = 2$ のとき
  - S が連結のとき (S の補グラフは非連結となる)
    S の補グラフの連結成分の 1 つを T とし，$T' = S \setminus T$ とすると，
    $$f(S) = f(T) + f(T'), \quad g(S,k) = \sum_{i=0}^{k} \binom{k}{i} g(T,i) g(T',k-i)$$
    である．
  - S が非連結のとき
    S の連結成分の 1 つを T とし，$T' = S \setminus T$ とすると，
    $$f(S) = \max\{f(T), f(T')\}, \quad g(S,k) = \sum_{i=0}^{k} \sum_{j=k-i}^{k} \binom{k}{i} \binom{i}{i+j-k} g(T,i) g(T',j)$$
    である．

この方法の時間計算量は $O(N^4)$ である．

[*1] 例えば `http://hos.ac/slides/20110504_graph.pdf` を参照．

Problem D: Drastic Sokoban

片づけが不可能な非自明な例としては, 以下のような場合が典型的である.



片づけが可能かどうかは, 以下の手順で判定できる.

1. 片づけ先のマスの集合を G とする. すべての荷物が既に G にあれば, 片づけは可能.
2. 荷物をすべて取り除いたとき, うさぎが到達できるマスの集合 S を求める. S にも G にも属さないマスに荷物があれば, 片づけは不可能.
3. 以下のいずれかを満たすならば片づけは可能, そうでなければ不可能.
   i) $|G| = B$ であり, ある隣接する 2 マス $x \in S \setminus G$, $y \in S \setminus G$ が存在.
   ii) $|G| = B + 1$ であり, ある隣接する 2 マス $x \in S \setminus G$, $y \in S$ が存在.
   iii) $|G| \geq B + 2$ であり, ある隣接する 2 マス $x \in S$, $y \in S$ が存在.

1. と 2. の場合の正当性は明らかであるから, 3. の場合の正当性を示す. まず $|G| = B$ のときを考える.

片づけが可能であると仮定する. 1. の場合を除外したので, 片づけが終了する直前の状態を考えることができる. このときにうさぎがいるマスも最後に動かされる荷物があるマスも $S \setminus G$ に属すため, 隣接する 2 マス $x \in S \setminus G$, $y \in S \setminus G$ が確かに存在する.

逆に, ある隣接する 2 マス $x \in S \setminus G$, $y \in S \setminus G$ が存在すると仮定する. 片づけが可能であることを, B に関して帰納的に示す. まず, うさぎは箱を適切に動かすことで S の任意のマスに移動できることに注意する. すべてのマスの個数が $|G| + 2 = B + 2$ 以上なので, 移動したいマスに荷物があっても動かせるからである. $S = \{x, y\}$ の場合, それらはうさぎがいるマスと荷物があるマスであり, $|G| = B = 1$ より片づけ先のマスが存在し, それは x, y でないから荷物を動かして片づけは完了する. そうでない場合, $z \in S \setminus \{x, y\}$ であって, z を固定家具のマスに変えたとしても, 荷物をすべて取り除いたとき, うさぎが $S \setminus \{z\}$ の任意のマスに到達できるようなものを 1 つとる (例えば, 頂点が S のマス, 辺が隣接関係を表す連結グラフの全域木を考え, x, y でない葉をとればよい). うさぎは以下の手順を行う.

- $z \in G$ かつ z に荷物がないとき, ある荷物に隣接するまで移動し, その箱を z に動かす.
- $z \notin G$ かつ z に荷物があるとき, z まで荷物を動かしながら移動したのち, z に隣接するマスに移動する.
- その他の場合, 何もしない.

すると「$z \in G$ かつ z に荷物がある」「$z \notin G$ かつ z に荷物がない」のいずれかの状態になり, 片づけは完了するか, z を固定家具のマスと考えることで, 荷物 $B - 1$ 個で片づけが可能な場合に帰着される.

以上より $|G| = B$ の場合の正当性は確認された. $|G| > B$ の場合, G のうち実際の片づけ先として使う B マス以外を G から除いて考えられるので, それぞれの必要十分条件が従う.

Problem E: Experience Point

経験値 C_i 点とボーナス D_i 点を得られる道の扱いを,

- 経験値 C_i 点を得て, ボーナスは無視する
- 経験値 $C_i + D_i$ 点を得るが, ゴールするまでに休憩地点を通らなければならない

のいずれか, として言い換えることができる. すると, チェックポイントと「ゴールするまでに休憩地点を通らなければならないかどうか」の組を頂点としたグラフを考えることができ, このグラフ上での最長経路を求めればよいことになる. グラフにサイクルがあり, そこで経験値をいくらでも多く得ることができる場合があり, そのときは 999 999 999 を出力することになる.

閉路がない場合

すべての辺が頂点番号が増える向きである場合, 頂点番号の順に, その頂点まで到達した状態で得られる経験値の最大値を動的計画法で順次計算することができる. 時間計算量は $O(N + M)$ となる.

$O(NM)$ 時間解法

Bellman-Ford 法により, 正閉路検出および正閉路が存在しない場合の最長経路を求めることができる.

$O(N + M)$ 時間解法

グラフを強連結成分に分解することで, 閉路がない場合に帰着できる. ある強連結成分内に得られる経験値が正の辺が 1 本でも含まれる場合, その強連結成分を通った場合は経験値をいくらでも多く得られる, として扱うことができる.

注意点

閉路があっても, 経験値が全く得られないもの, あるいはスタートから到達できない・ゴールに到達できないものがあり得ることに注意を要する.

また, 経験値をいくらでも多く得ることができない場合でも, 経験値を 999 999 999 を超えて得られる場合がある ($(30\,000 - 1) \times (30\,000 + 30\,000)$).

Problem F: Frill Wreath

円周 $\left(x-\frac{X}{M}\right)^2+\left(y-\frac{Y}{M}\right)^2=\left(\frac{\sqrt{P_1^{E_1}\cdots P_N^{E_N}}}{M}\right)^2$ 上の格子点 (x,y) の個数を求める問題である．これは $x'^2+y'^2=P_1^{E_1}\cdots P_N^{E_N}$, $(x',y')\equiv(X,Y) \pmod M$ を満たす整数の組 (x',y') の個数と等しく, さらに $z=x'+y'\sqrt{-1}$ とおけば条件は $z\overline{z}=P_1^{E_1}\cdots P_N^{E_N}$, $z\equiv X+Y\sqrt{-1} \pmod M$, $z\in\mathbb{Z}[\sqrt{-1}]$ と同値である．

環 $\mathbb{Z}[\sqrt{-1}]$ は UFD (一意分解整域) であり, $\mathbb{Z}$ の素数 p は以下のように $\mathbb{Z}[\sqrt{-1}]$ で素元分解される：

- $p=2$ のとき, $2=-\sqrt{-1}(1+\sqrt{-1})^2$.
- $p\equiv 1 \pmod 4$ のとき, ある素元 $\pi\in\mathbb{Z}[\sqrt{-1}]$ に対し $p=\pi\overline{\pi}$.
- $p\equiv 3 \pmod 4$ のとき, $p\in\mathbb{Z}[\sqrt{-1}]$ は素元.

よって, $\mathbb{Z}$ での素因数分解を $P_1^{E_1}\cdots P_N^{E_N}=2^e\left(\prod_i p_i^{f_i}\right)\left(\prod_j q_j^{g_j}\right)$ $(p_i\equiv 1,\ q_j\equiv 3 \pmod 4)$, $p_i=\pi_i\overline{\pi_i}$ とすると, $z\overline{z}=P_1^{E_1}\cdots P_N^{E_N}$ を満たす z はすべての g_j が偶数のときに限り存在し,

$$z=u(1+\sqrt{-1})^e\left(\prod_i \pi_i^{f_i-x_i}\overline{\pi_i}^{x_i}\right)\left(\prod_j q_j^{g_j/2}\right)\quad (u\in\{\pm1,\pm\sqrt{-1}\},\ 0\le x_i\le f_i)$$

と表される $4\prod_i(f_i+1)$ 個である．

k, a, b に対し, $\prod_{1\le i\le k}\pi_i^{f_i-x_i}\overline{\pi_i}^{x_i}\equiv a+b\sqrt{-1} \pmod M$ となる $x_1,\ldots,x_k$ の選び方が何通りあるか, を求める動的計画法によって条件を満たす z を数えることができる．素朴な方法では, 各 i について, $x_i=0,\ldots,f_i$ に対し $\pi_i^{f_i-x_i}\overline{\pi_i}^{x_i} \bmod M$ を計算する．時間計算量は $O\left(\sum_i f_i\log f_i+M^2\sum_i f_i\right)$ となる．f_i に比例する時間を使わないためには, a, b に対し $\pi_i^{f_i-x_i}\overline{\pi_i}^{x_i}\equiv a+b\sqrt{-1} \pmod M$ なる x_i が何個あるかの分布を求める．これで動的計画法の部分の計算量は $O(M^4)$ となる．

解法 1

$\left(\pi_i^{f_i-x_i}\overline{\pi_i}^{x_i}\right)_{0\le x_i\le f_i}$ は等比数列であり, $\frac{\overline{\pi_i}}{\pi_i}=\frac{\overline{\pi_i}^2}{p_i}$ より p_i が M と互いに素ならば $\bmod M$ でも等比数列でありよって周期的であることに注目する．p_i が M と互いに素でない場合でもこれらはおおよそ周期的であり, 正確には, M が p_i でちょうど l 回割り切れるとき, $\left(\pi_i^{f_i-x_i}\overline{\pi_i}^{x_i}\right)_{l\le x_i\le f_i-l}$ は周期的である ($\pi_i^l\overline{\pi_i}^l=p_i^l$ での商を考えよ)．よって $x_i\le l+M^2$ までを調べれば周期が見つかり, $\pi_i^{f_i-x_i}\overline{\pi_i}^{x_i} \bmod M$ の分布がわかる．各 i に対し $O(M^2\log f_i)$ 時間となる．なお本問の制約では $M<5^3$ より $l\le 2$ である．

解法 2

$\pi_i^{f_i-x_i}\overline{\pi_i}^{x_i} \bmod M$ の分布を f_i に対して再帰的に求める．$f_i>0$ とする．f_i が偶数の場合, $\pi_i^{f_i-1-x_i}\overline{\pi_i}^{x_i}$ の分布に π_i をかけたものと $\overline{\pi_i}^{f_i}$ を考えればよい．f_i が奇数の場合, $\pi_i^{\frac{f_i-1}{2}-x_i}\overline{\pi_i}^{x_i}$ の分布に $\pi_i^{\frac{f_i+1}{2}}$, $\overline{\pi_i}^{\frac{f_i+1}{2}}$ をかけたものを考えればよい．各 i に対し $O(M^2(\log f_i)^2)$ 時間となる．

Problem G: Galaxy View

どの 3 点も同一直線上にない平面上の 5 点の位置関係は, 凸包が三角形, 四角形, 五角形のいずれであるかで 3 通りの場合がある. 5 点が星形をなすためには, 凸包が五角形であることが必要十分であることがわかる.

凸包が三角形, 四角形, 五角形である 5 点の組合せがそれぞれ a 通り, b 通り, c 通りであるとする. これらを区別するための点の配置として,

(i) 三角形とその内部の 1 点である 4 点の組合せ
(ii) 三角形とその内部の 2 点である 5 点の組合せ

を考える. これらがそれぞれ d 通り, e 通りであるとする.

(i) の 4 点に残りの $N-4$ 点のうち 1 点を加えることを考えると, その 1 点が三角形の内部にある場合は凸包が三角形である 5 点, 外部にある場合は凸包が四角形である 5 点が生じる. 凸包が三角形である 5 点は (i) のような 4 点 4 通りから, 凸包が四角形である 5 点は (i) のような 4 点 2 通りから生じるから, $(N-4)d = 4a + 2b$ がわかる. また, (ii) は凸包が三角形である 5 点であるから, $e = a$ である. これらと $a + b + c = \binom{N}{5}$ をあわせると, $c = \binom{N}{5} - \frac{1}{2}(N-4)d + e$ として c を d, e から求めることができる.

d, e を効率よく求める方法を考える. 3 点の組合せそれぞれに対し, それらがなす三角形の内部の点の個数が求まれば十分である. 3 点の組 (A, B, C) であって, (x 座標, y 座標) が辞書順最小のものが A であり, A, B, C はこの順に正の向きであるようなものを考えることにする. 点 D が三角形 ABC の内部にあることは, $\angle BAD < \angle BAC$ かつ $\angle ABD < \angle ABC$ と言い換えられる. よって, A, B を固定し, 残りの点を C として $\angle BAC$ の小さい順に見ていくことを考えると, 各 C に対し「その時点で既に見た点 D であって $\angle ABD < \angle ABC$ を満たすもの」が数えられればよい. これは例えば binary indexed tree を用いて $O(\log N)$ 時間で問い合わせることができる (A, B を決めたとき残りの点を角度でソートしておけばよい). この方法の全体の時間計算量は $O(N^3 \log N)$ となる.

Problem H: Hidden Problem

実行に時間がかかる C 言語のプログラムの出力を求める問題である.

H1.in

$\displaystyle\sum_{i=1}^{20121224}\sum_{j=1}^{i}1$ を計算するプログラムである.

代わりに, $\displaystyle\sum_{i=1}^{20121224} i$ を計算すればよい. 元のプログラム通り, 32-bit 整数をオーバーフローさせなければならないことに注意する.

H2.in

整数の組 (x, y) を引数として受け取る関数が, $x > 0$, $y > 0$ の場合は $(x+1, y-2)$, $(x-1, y-1)$, $(x-3, y+2)$ に対して再帰呼び出しを行っている.

$x+y$ の値が常に減少するためこの再帰呼び出しは必ず停止する. メモ化を行えばよい.

H3.in

整数 n $(2 \le n \le 40)$, n 個の整数 $a_{i,j}$ $(0 \le i < n,\ 0 \le j < n,\ 0 \le a_{i,j} < 100\,000)$ が与えられたとき, n 個の整数 z_i $(0 \le i < n)$ であって $z_0 = 1$, $z_1 = 4$, $1 \le z_i \le 4$ を満たすものをうまく選んだときの, $S = \displaystyle\sum_{z_i < z_j} a_{i,j}(z_j - z_i)$ の最小値を求めるプログラムである.

全探索で書かれたプログラムに適切な枝刈りを入れることで解くこともできるが, 入力の多項式時間で答えを求める解法を紹介する.

まず, S の最小値は $z_i \ne 2$ としてもとれることがわかる. これは, $z_i = 2$ であるすべての z_i を同時に 1 または 3 に変化させることで, 少なくとも一方は S の値が減少しないからである. 同様に $z_i \ne 3$ としてよいこともわかる. つまり $S = \displaystyle\sum_{z_i=1}\sum_{z_j=4} 3a_{i,j}$ である. したがって, この問題は $v_0, v_1, \ldots, v_{n-1}$ を頂点とし, 頂点 v_i から頂点 v_j へコスト $3a_{i,j}$ の辺があるグラフの最小 v_0-v_1 カット問題に帰着される.