Problem Analysis
The 2025 ICPC Asia Yokohama Regional Contest

Problem A: Tatami Renovation

原案: 佐藤 遼太郎　　主担当: 佐藤 遼太郎　　解説: 佐藤 遼太郎

タイルの枚数が奇数の場合，明らかに敷き詰めは不可能である．一方，タイルの枚数が偶数の場合，敷き詰めは常に可能である．以下，タイルの枚数は偶数とする．

動かす必要があるタイルの枚数の下界を以下の方法で与える．長い廊下を，各区間にタイルが偶数枚含まれるような区間たちにできるだけ細かく分割する．分割後の区間のうちタイルが 1 枚のみ含まれる列から始まるようなものそれぞれについて，その区間に置かれていたタイルのうち動かす必要があるものの枚数の下界を以下の要因から求める．

- ある列にタイルが 2 枚置かれているならば，明らかにそのうち 1 枚以上を動かさなければならない．なぜならば，もしどちらも動かさない場合，廊下全体がその列を境界として奇数枚のタイルを持つ 2 つの矩形に分かれてしまうからである．
- グリッドを白黒の市松模様に塗り分けたとき，各区間について，最初にその区間に置かれていたタイルのうち最終的に白マスに置かれているものと黒マスに置かれているものの枚数が一致しなければならないことが証明できる（証明において，分割した区間の境界を跨ぐようなタイルの動かし方も考慮すべき点には注意が必要である）．

これらに基づいて区間ごとに求めた下界の総和がもとの問題の解の下界となる．ここで，この下界を達成するような動かし方が常に存在するので，この下界がその元の問題の解である．想定される計算量は入力された値のソートがボトルネックとなり $O(n \log n)$ である．

別解として，キーとして現在の列と次の列のタイルの置き方，値としてこれまで動かしたタイルの最小枚数の情報を持って一列目から順に処理していく動的計画法も考えられるが，実装は少々煩雑になるだろう．

Problem B: Minimizing Wildlife Damage

原案: 楠本 充　　主担当: 松下謙太郎　　解説: 松下謙太郎

解の構造について考える。d 回の操作で取り除かれるものは「$a_i = 0$ になるまで取り除かれるも

の」、「$a_i > 0$ で残るが、一回以上取り除かれるもの」、「一度もの取り除かれないもの」の三種類に分けることができる。それぞれを、ブロック 1, ブロック 2, ブロック 3 と呼ぶ。ブロック 1 では、(初期状態から $a_i = 0$ である場合を除いて) 0 が三つ並ぶことはない。なぜならば、間を 0 にしない方が d をより多く消費できるからである。また、$a_i > 0$ が二つ以上連続することはないとしてよい。これも同様の理由から、片方を 0 にしても解が悪化しないようにできるからである。このとき、d を消費できる量は単純な DP によって求まる。

ブロック 2 は、K 個の連続する $a_t, a_{t+1}, \ldots, a_{t+K-1} > 0$ から $\min(a_t, \ldots, a_{t+K-1})$ 未満の回数等しく取り除いた状況になる。

重要な観察として、ブロック 1 は d を超えない範囲でなるべく長く保たれていることが min の条件からわかる。よって、パリティも含めるとブロック 1 の長さの候補は $O(1)$ である。

最後に、クエリに解答することを考える。ブロック 2 をどこで切るかに任意性が残るが、これは Convex Hull Trick を用いて列を後ろから走査しながら一次方程式を管理することによって、ブロック 1 の途切れる位置の候補に到達したときに最大値を $O(\log n)$ で求めることができる。

Problem C: Seagull Population

原案: 城下 慎也　　主担当: 城下 慎也　　解説: 城下 慎也

まず，総カモメ数の自明な下界として b_i の最大値 $D_{\max} = \max(b_1, \ldots, b_n)$ が考えられる．これは，1 日に観測されるカモメ数は必ず総カモメ数以下であることから分かる．同様に，カモメが増えた回数の総和 $S_{\mathrm{inc}} = \sum_{i=1,\ldots,n} \max(b_i - b_{i-1}, 0)$ (ただし $b_0 = b_n$) も総カモメ数の下界となる．これは，$b_i > b_{i-1}$ ならば，i 日目に島にやって来るカモメが少なくとも $b_i - b_{i-1}$ 羽はいることから示すことができる．よって，$M_{\mathrm{opt}} = \max(D_{\max}, S_{\mathrm{inc}})$ は解の下界であることがわかる．以下，M_{opt} が最適であることを構成例とともに示す.

まず，いずれかの日にカモメがいないケースを考える．対称性のため，$b_n = 0$ としてよい．このとき，1 日目から順に，図のように先に来たカモメの滞在期間ができるだけ長くなるよう貪欲に決定すると S_{inc} 羽の解を構成できる．以降の図では，縦の列はそれぞれの日を，アルファベットは各カモメの滞在期間を，数値はその日に観測されるカモメの総数を表すものとする．このケースでは $S_{\mathrm{inc}} \geq D_{\max}$ が成り立つため M_{opt} は最適である.

```
AAAAAAAAAA.
..BBBBBB...
...CC.DD...
-----------
11233233110
```

次に，すべての日に 1 羽以上カモメがいるケースを考える．b_i の最小値を $D_{\min} = \min(b_1, \ldots, b_n)$ とおいたとき，通年のカモメは $D_{\min}$ 羽とは限らず，図のようにカモメの滞在期間を重複させるこ

とで節約できる．

```
AAAAA....AAAAAA
..BBBBBBBBBB...
---------------
112221111222111
```

すべての b_i を $D_{\min}$ ずつ減らしたうえで最初のケースのように貪欲な構成例を考える．このとき，最初の図で同じ高さとなったカモメのグループ（新しく訪れた時にすでに滞在しているカモメの羽数が同じグループ）は互いに滞在期間が重複しない．このグループに対し，次のように区間の終点を巡回するように交換することで最大（グループのカモメの羽数 -1）羽まで通年のカモメを節約できる．

```
..AAA...............     ..AAAAAAAAA.........     ..AAAAAAAAAAAAAAAAA..
.......BBBB..........    .......BBBBBBBBBBBB..    BBBBB..BBBBBBBBBBBBBB
..............CCCC.. ->  CCCCC..........CCCCCC -> CCCCCCCCCCC....CCCCCC
--------------------     ---------------------    ---------------------
00111001111000011110 0   112221122221111222211    223332233332222333322
```

以上のことを全ての高さで実施すると，合計でちょうど $S_{\mathrm{inc}} - (D_{max} - D_{min})$ 羽の通年のカモメを節約できる．$S_{\mathrm{inc}} \geq D_{\max} \Leftrightarrow S_{\mathrm{inc}} - (D_{\max} - D_{\min}) \geq D_{\min}$ であれば，ちょうど $D_{\min}$ 回節約することで S_{inc} 羽の解を構成できる．反対に $S_{\mathrm{inc}} < D_{\max} \Leftrightarrow S_{\mathrm{inc}} - (D_{\max} - D_{\min}) < D_{\min}$ の場合は，通年のカモメが $D_{\min} - (S_{\mathrm{inc}} - (D_{\max} - D_{\min})) = D_{\max} - S_{\mathrm{inc}}$ 羽必要であり，総カモメ数は $S_{\mathrm{inc}} + (D_{\max} - S_{\mathrm{inc}}) = D_{\max}$ 羽となる．いずれのケースにおいてもちょうど M_{opt} 羽の解を構成できているため，M_{opt} は最適解である．

Problem D: Decompose and Concatenate

原案: 稲葉 一浩　　主担当: 山口 文彦　　解説: 山口 文彦

正整数値が一つ与えられる．以下では，これを s とする．$s = a + b$ となる二つの正整数値 a, b の十進表記を文字列として連結してできる数値（の表現が表す整数値）のうち，最大のものを求める問題である．ここでは文字列の連結を . で表すことにして，求める数値の表現を $a.b$ と書くことにする．

最大の正整数値を求めたいので，$a.b$ の桁数をできるだけ大きくしたい．a と b の桁数を s と同じにできる場合には，それが最大である．これは s の最上位桁の数字が 2 以上の場合である．このとき，a をできるだけ大きく（$b = s - a$ を小さく）したい．n 桁で最小の正整数値は 10^{n-1} なので，b を s を超えない最大の 10 のべき乗とすればよい．

しかし，s の最上位桁の数字が 1 の場合には（その場合だけは），s を同じ桁数の整数の和に分解することができない．s の上から 2 桁目の数字が 0 でない場合（たとえば $s = 110$），b を s と同じ桁数の値，すなわち s を超えない最大の 10 のべき乗（この例では 100）としてよい（$110 = 10 + 100$）．

最後に，s の最上位の 2 桁が 10 の場合（たとえば 102）について考える．b を s を超えない最大の 10 のべき乗（この例では 100 ）としてしまうと，a の桁数が s の桁数よりも 2 つ以上小さくなってしまって都合が悪い．この場合には b を s よりも一桁小さな 10 のべき乗（この例では 10 ）とすればよい $(102 = 92 + 10)$.

s の範囲が 10^{17} 以下と大きいので，固定長の整数表現を使うときには注意が必要である．

Problem E: Cutting Tofu

原案: 丸茂 直貴　　主担当: 北川 宜稔　　解説: 北川 宜稔

切り出す立方体の一辺の長さを x とする．三辺が a, b, c である直方体から，この大きさの立方体は $\lfloor a/x \rfloor \cdot \lfloor b/x \rfloor \cdot \lfloor c/x \rfloor$ 個作ることができる．ここで，$\lfloor \cdot \rfloor$ は床関数で，整数部分を表す関数である．よって，この問題は $\lfloor a/x \rfloor \cdot \lfloor b/x \rfloor \cdot \lfloor c/x \rfloor \geq k$ となるような $x > 0$ の最大値を求める問題と言い換えられる．ただし，x の最大値は有理数になるので，既約分数で答えよという条件が付いている．

最後の既約分数で答えよという条件がなく，ある程度の誤差を許すような設定であれば，x は二分探索で求めることができる．なぜなら，$\lfloor a/x \rfloor \cdot \lfloor b/x \rfloor \cdot \lfloor c/x \rfloor$ は x に関して単調減少な関数だからである．x を浮動小数点数で二分探索しても厳密解が得られるとは限らないため工夫する必要がある．

分数で厳密に求める方法を考える．$\lfloor a/x \rfloor$ は a/x が整数でないなら x を微小に増加させても値は変化しない．$\lfloor b/x \rfloor$, $\lfloor c/x \rfloor$ についても同様である．よって，$a/x, b/x, c/x$ のどれかが整数のときに限り，x を微小に増加させると $\lfloor a/x \rfloor \cdot \lfloor b/x \rfloor \cdot \lfloor c/x \rfloor$ が減少する．以上より，x が求める最大値ならば $a/x, b/x, c/x$ のどれかは整数でなければならない．つまり，最大値 x は $a/i, b/i, c/i$ (i は正の整数) のいずれかの形になる．$x = a/i$ という形の場合

$$\lfloor a/x \rfloor \cdot \lfloor b/x \rfloor \cdot \lfloor c/x \rfloor = i \cdot \left\lfloor \frac{b}{a} i \right\rfloor \cdot \left\lfloor \frac{c}{a} i \right\rfloor$$

となり，これが k 以上になる最小の i を求めるには，i に関して二分探索すればよい．$x = b/i$ や $x = c/i$ の場合も同様なので，x の形に応じた三通りの二分探索を行い，それらの結果の最大値が答えとなる．

Python では問題にならないが，他の言語の場合 $i \cdot \left\lfloor \frac{b}{a} i \right\rfloor \cdot \left\lfloor \frac{c}{a} i \right\rfloor \geq k$ をチェックするときにオーバーフローする可能性があるので注意されたい．また，i を探索する範囲も間違えやすい．$x = a/i$ の形の場合，最適な i は 0 より大きく $\max(a, k)$ 以下という範囲にある．

別解法

$a/i, b/i, c/i$ という形の有理数は，最小公倍数 $\mathrm{lcm}(a, b, c)$ を用いて，まとめて $\mathrm{lcm}(a, b, c)/i$ という形に表される．よって，$x = \mathrm{lcm}(a, b, c)/i$ という形に限定して i を二分探索すれば答えが求まる．

素直に x を二分探索する方針でも答えを求めることができるが，その場合，分数を復元できる程度に十分な精度で計算する必要がある．$x = n/2^{64}$ (n は正の整数) の形での最大値を求めた後，$n/2^{64}$ の連分数展開を計算すれば最適値を求めることができる．いずれにしてもこれらの別解法では多倍長整数が必要になる．

Problem F: Astral Geometry

原案: 丸茂 直貴　　主担当: 楠本 充　　解説: 楠本 充

星 i の座標を (x_i, y_i, z_i) とし、また $\vec{r_i} = [x_i, y_i, z_i]^\top$ とする。最初に地球と星星との距離がわかっていることは $\|\vec{r_i}\|^2$ が分かっていることに相当する。また、星 i と星 j の距離を計測することは $\|\vec{r_i} - \vec{r_j}\|^2$ を特定することに相当するが、これは展開すると $\|r_i\|^2 + \|r_j\|^2 - 2\vec{r_i} \cdot \vec{r_j}$ でもあるから、内積 $\vec{r_i} \cdot \vec{r_j}$ を特定することでもある。

$3 \times n$ 行列 M を $M = [\vec{r_1} \dots \vec{r_n}]$ と置く。知りたいものは行列積 $S = M^\top M$ の各要素の値である。上記の議論を踏まえると、インタラクションの初期段階では S の対角成分のみが分かっていて、1 回の計測で S の 1 要素が分かるということである。問題の目的は S のすべての要素を $3n$ 回で特定するということに言い換えられる。なお、S は対称行列であることに留意されたい。

問題を解くために、$\mathrm{rank}(S) \leq 3$ が成り立つ ($\mathrm{rank}(M) \leq 3$ であるため) ことを利用する。アプローチとしては M において線形独立な基底ベクトル ($\vec{r_1}, \dots, \vec{r_n}$ から $\mathrm{rank}(M)$ 個) を取り、それを元にして S の要素をすべて特定する方針を取る。

まず、問題の仮定としてすべての星は原点にはないという仮定があるので、$\vec{r_1}$ は基底に入れてよい。次に、$\vec{r_1}$ と線形独立なベクトルが無いかを確認したい。もし、ある $\vec{r_a}$ が $\vec{r_1}$ と線形独立であるならば、部分行列

$$\begin{bmatrix} S_{11} & S_{1a} \\ S_{a1} & S_{aa} \end{bmatrix}$$

の行列式は非ゼロである。逆にそうでないならこの 2 つは線形従属である。このように線形独立なベクトルは星 1 と他の各星について $n-1$ 回の計測を行うことで発見できる。

もし線形独立なベクトル $\vec{r_a}$ が見つかった場合、さらにもう一つ線形独立なベクトルがないか探すことを考える。同じように、ある $\vec{r_b}$ が $\vec{r_1}, \vec{r_a}$ と線形独立なのだとすると

$$\begin{bmatrix} S_{11} & S_{1a} & S_{1b} \\ S_{a1} & S_{aa} & S_{ab} \\ S_{b1} & S_{ba} & S_{bb} \end{bmatrix}$$

の行列式が非ゼロなはずである。星 a と他の各星について $n-2$ 回の計測を行うことでこのような $\vec{r_b}$ があるかどうかを判定できる。

このようにして基底ベクトルを 3 つ取れたとする。さらに $n-3$ 回の追加の計測を行い、星 b と他の各星について計測を行うことにしよう。すると $i \neq j$ に対し、行 $1, a, b, i$ と列 $1, a, b, j$ を取り出した部分行列

$$\begin{bmatrix} S_{11} & S_{1a} & S_{1b} & S_{1j} \\ S_{a1} & S_{aa} & S_{ab} & S_{aj} \\ S_{b1} & S_{ba} & S_{bb} & S_{bj} \\ S_{i1} & S_{ia} & S_{ib} & S_{ij} \end{bmatrix}$$

の行列式はゼロでなければならないはずである。そうでないなら S のランクが 4 以上ということになってしまうためである。この行列式は S_{ij} を未知変数と見なすと単なる 1 次方程式である。S_{ij} 以外の要素の値は分かっているので、1 次方程式を解けば S_{ij} の値が分かることになる。よって、任

意のペア i, j についてこれを行えば S の要素が求まるので、あとは内積を距離に変換して答えれば良い。

基底の数が 1 つや 2 つの場合でも同じように部分行列の行列式を考えて 1 次方程式を解くことで S_{ij} の値を求めることができる。計算時間は $O(n^2)$ であり、必要になる計測は $3n-6$ 回である。

実装においては、S の各要素は最大で $3 \times 4000^2 \simeq 5 \times 10^7$ 程度の大きい値になる点に注意を要する。特に S で 3×3 や 4×4 行列の行列式を求める際に 64 bit 整数を使うとオーバーフローしてしまう。これに対処するには 128 bit 以上の精度をもつ整数型を使うか、あるいは複数個の素数を用いて有限体上で値を計算するアプローチが有効である。

Problem G: Charity Raffle

原案: 住谷 達哉　　主担当: 住谷 達哉　　解説: 住谷 達哉

i 番目の種類の景品数を a_i と表記する. 最大値が m 以下であるような非負整数列 $A = (a_1, \ldots, a_n)$ であって，あり得るものの個数を求めればよい．明らかに $\sum_i a_i = k$ を満たす必要はあるため，その中で列 A としてあり得ないものの個数を数えることにする.

景品の選択の方針から，一つの種類の景品だけを多く獲得しているということはあり得ない．より正確な条件として，列 A の最大値を $r\ (\leq m)$ として，以下が満たされていると仮定する.

- $a_i = r$ となる i はただ一つである（$a_{i_0} = r$ とする).
- $1 \leq i < i_0$ なる全ての i に対して $a_i \leq r-1$ が成り立ち，$i_0 < i \leq n$ なる全ての i に対して $a_i \leq r-2$ が成り立つ.

このとき，i_0 番目の景品を $r-1$ 個から r 個に増やすことはできないため，このような列 A はあり得ないということが分かる．そうでない場合は，$a_i = r$ かつ $r-1 \leq a_j \leq r$ であるような i, j の組 $(1 \leq i < j \leq n)$ が存在する．最初に種類 i, j の組が $a_i + a_j$ 回 pick up され，その後各 $l\ (\neq i, j)$ に対して種類 i, l の組が a_l 回ずつ pick up されるというシナリオで列 A を実現することができる.

上記条件を満たす（すなわち不可能である）列 A を数える方法として，以下 2 つの解法がある.

解法 1

$a'_{i_0} = r-1, a'_i = a_i\ (i \neq i_0)$ として非負整数列 $A' = (a'_1, \ldots, a'_n)$ を作ると，

- $\sum_i a'_i = k-1$
- $\max A' = r-1$

が満たされる．逆に，このような任意の列 A' に対して, $a'_i = r-1$ の中で一番 i が大きい要素を 1 増やすという操作を考えると，不可能な条件を満たす列 A を作ることができる．これらの変換は全単射であることを示せるため，列 A' の個数を数える問題に帰着できる．r に関して総和をとるので，総和が $k-1$ で最大値が $m-1$ 以下であるような列の個数を数えればよく，$O(n+k)$ の計算量でこの問題を解くことができる.

解法 2

(r, i_0) を固定した際の上記条件を満たす列 A の個数は包除原理より，

$$\sum_{S \subset \{1,2,\ldots,n\} \setminus \{i_0\}} (-1)^{|S|} f(i_0, S, r) \tag{1}$$

と算出できる．ここで，$f(i_0, S, r)$ は次の条件を満たす非負整数列 $A' = (a'_1, \ldots, a'_n)$ の個数を表す．

- $\sum_i a'_i = k$
- $a'_{i_0} = r$
- $i \in S$ に対し，$i < i_0$ ならば $a'_i \geq r$ であり，$i > i_0$ ならば $a'_i \geq r - 1$ である

S の要素のうち i_0 未満のものの個数を $c(i_0, S)$ とすると，$f(i_0, S, r)$ は総和が $k - r - (r-1)|S| - c(i_0, S)$ であるような長さ $n - 1$ の非負整数列の個数として算出できる．

列 A としてあり得ないものの個数を求めるためには r, i_0 に関して上記の総和をとればよく，

$$\sum_{r=1}^{m} \sum_{i_0=1}^{n} \sum_{S \subset \{1,2,\ldots,n\} \setminus \{i_0\}} (-1)^{|S|} f(i_0, S, r) \tag{2}$$

と算出できる．ただしこの表式のままでは計算量が大きすぎる．ここで，$S' = S \cup \{i_0\}$ と置くことで，上式を

$$\sum_{r=1}^{m} \sum_{S' \subset \{1,2,\ldots,n\}} (-1)^{|S'|-1} \sum_{i_0 \in S'} f(i_0, S' \setminus \{i_0\}, r) \tag{3}$$

と変形できる．$i_0 \in S'$ における総和の部分では，$c(i_0, S' \setminus \{i_0\})$ は 0 から $|S'| - 1$ までの値を 1 つずつとる．したがって，$\sum_{i_0 \in S'} f(i_0, S' \setminus \{i_0\}, r)$ は総和が $k - r|S'|$ 以上 $k - (r-1)|S'| - 1$ 以下であるような長さ $n - 1$ の非負整数列の個数に等しい．この値は $r, |S'|$ のみに依存し，また，二項係数を用いて $O(1)$ で算出できる．$|S'|$ として考慮すべき値は高々 $\min(n, \frac{k-1}{r-1})$ までであることを合わせると，全体で $O(n + k \log k)$ の計算量でこの問題を解くことができる．

Problem H: U-Shaped Panels

原案: 楠本 充　　主担当: 鵜川 始陽　　解説: 鵜川 始陽

U 字の形をよく見てみよう．問題文でも述べられている通り，U 字には 1 辺だけ開口部がある．制約である $k \geq 5$ を満たす場合，他の U 字をどのように配置しても開口部を完全には埋められない．このことに気づけば，この問題は解けたも同然である．

$k \geq 5$ を満たす場合，開口部の間口は 3 マス以上になる．しかし，他の U 字をどのように配置しても，それらが重ならない場合は開口部を高々 2 マスまでしか埋められない．したがって，少なくとも 1 マスは空いたままになる．ちなみに，2 マスを埋めるには図 1 のように開口部に別の 2 個の U

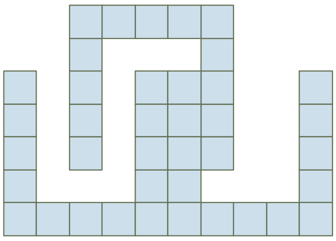

図 1　開口部を 2 マス埋める配置

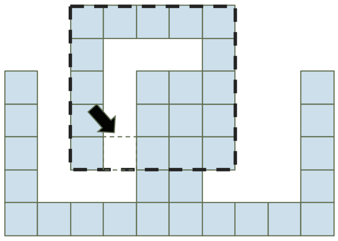

図 2　U 字の向きの判定

字の縦の辺を差し込むしかない．したがって，もしある $k \times k$ の正方形領域をバウンディングボックスとする U 字が存在するなら，その U 字の向きは一意に決まる．つまり，そのような U 字が存在するなら，図 2 のように，正方形の 4 辺の中で埋まっていないマスがある辺がちょうど一つだけあるはずで，その辺が開口部になる．

もし与えられた計画が実現可能であれば，U 字を一つ取り除いた計画も実現可能である．また，板が一つもない計画も実現可能である．したがって，一つずつ U 字を取り除き，一つも板が残らない状態にできれば計画は実現可能と判断できる．

この U 字を取り除いていく処理をシミュレートすればよい．U 字のバウンディングボックスの正方形の 4 隅にはいずれも板 (#) があるので，池を端から順に走査したとき，最初に見つかった板は U 字のバウンディングボックスの隅のはずである．その中の U 字を取り除く．U 字の向きは一意に決められるので，バックトラックは必要ない．シミュレーションの途中で，バウンディングボックスの中に U 字がない状況 (開口部が二つ以上あったり，残る 3 つの隅のいずれかに板がない場合が相当する) になれば，この計画は実現できない．

なお，U 字を一つ取り除いた後に次の板を探すとき，また池の端から探し始めると計算量が $O((nm)^2)$ になって間にあわない．取り除いた U 字より前には板がないことが分かっているので，続きから次の U 字を探せば十分である．

Problem I: Game of Names

原案: 住谷 達哉　　主担当: 隈部 壮　　解説: 住谷 達哉

ゲームの初期時点で Alice と Bob の名前が書かれているセルの個数をそれぞれ m_A, m_B として，また最終的なそれぞれのセルの個数を n_A, n_B とする．Alice の勝利条件は $n_A - m_A > n_B - m_B$ すなわち $n_A - n_B > m_A - m_B$ である．

ゲームの決着がついても勝者が可能な限り自分の名前を盤面に書き込むことにしても勝敗は変わらない．一番最後の盤面の状態を考えると，空きマスを飛ばして Alice と Bob の名前が交互に並ぶことがわかる（そうでなければ，どちらかの名前をまだ盤面に書き込むことができるため）．そのため $n_A - n_B$ は両端のマスに応じて以下のようになる（両端のマスは必ず埋まる）．

- 両端のマスを Alice がとったとき 1
- 両端のマスを Bob がとったとき -1
- それ以外のとき 0

したがってこのゲームの勝敗は両端の 2 マスをどちらのプレイヤーが取れたかのみで決まり，適切な場合分けにより解くことができる．

Problem J: ICPC Board

原案: 丸茂 直貴　　主担当: 丸茂 直貴　　解説: 丸茂 直貴

問題文中の仮説を満たすような盤面を**良い**盤面と呼ぶことにする．盤面が良い盤面であるための必要十分条件は，以下のいずれかの形であることである:

- 「`C` と `I` が交互に並ぶ行」と「`C` と `P` が交互に並ぶ行」が交互に縦に並ぶ,
- 「`C` のみからなる行」と「`I` と `P` が交互に並ぶ行」が交互に縦に並ぶ,
- 上記いずれかの形の転置である．

この必要条件が得られたら，判定も構築も計算量 $O(nm)$ で可能となる．以下では，これが必要十分条件であることを確認する．`I` or `P` を `X` と書く．

1 行目に `C` と `X` が交互に並ぶ場合，良い盤面であるためには，2 行目にも `C` と `X` が交互に並ぶ必要があることがわかる．帰納的に，すべての行に `C` と `X` が交互に並ぶこととなる．

1 行目に `CC` もしくは `XX` が含まれる場合，それらの列には `CC` と `XX` が交互に縦に並ぶこととなる．他の列も `C` と `X` が交互に並ぶことがわかる．

以上より，良い盤面であるためには，各行もしくは各列に `C` と `X` が交互に並ぶ必要があることがわかった．以下では，各行に `C` と `X` が交互に並ぶ場合のみを考える．各行は以下のいずれかである:

(a) `CXCXCX...`
(b) `XCXCXC...`

■盤面に (a) と (b) がともに存在する場合　以下のように (a) と (b) が上下に隣接する箇所がある:

```
CXCXCX...
XCXCXC...
```

この部分に注目すると，「`C` と `I` が交互に並ぶ行」と「`C` と `P` が交互に並ぶ行」が交互に縦に並ぶ必要があることがわかる．例えば以下のような具合である:

```
CICICI...
CPCPCP...
ICICIC...
CPCPCP...
```

■盤面が (a) のみからなる場合　盤面は以下のようになる:

```
CXCXCX...
CXCXCX...
```

```
CXCXCX...
CXCXCX...
```

すなわち C のみからなる列と X のみからなる列が交互に並ぶ．X のみからなる列は I と P が交互に並ぶ必要がある．以下のような具合である:

```
CICICP...
CPCPCI...
CICICP...
CPCPCI...
```

(b) のみからなる場合も同様である．

以上で，冒頭の条件が，良い盤面であるための必要十分条件であることが確認できた．

Problem K: Membership Structure of a Secret Society

原案: 稲葉 一浩　　主担当: 稲葉 一浩　　解説: 稲葉 一浩

問題を数学的に解釈すると，以下の 3 タイプの論理式が複数与えられるので，

- $x_i \in x_j$
- $x_i \notin x_j$
- $x_i = x_j \cap x_k$

すべてを満たすよう変数に集合を割当てることは可能か？ を判定する問題となる．(推薦者集合が同じならば同一人物という問題設定なので，解説では推薦者集合とその推薦する人物を同一視する．)

問題を難しくしているのは，$x_i = x_j \cap x_k$ タイプの論理式である．これらからは，変数の指す集合同士が等しい，という関係が帰結してしまうことがある．たとえば $x_1 = x_2 \cap x_3$ と $x_4 = x_3 \cap x_2$ からは，$x_1 = x_4$ が導出される．この時，他に $x_1 \in x_5$ と $x_4 \notin x_5$ という論理式があったとすると，構文的な見た目からは一見問題ないが，意味的には矛盾しており，集合割当てが不可能になる．

まずはこの問題を解決しよう．$x_i = x_j \cap x_k$ 型の論理式を処理して，その論理的帰結として導出される変数同士の $=$ 関係をすべて求めておくのである．まず仮に，各変数の指す集合が異なる要素を何かしら持っていると仮定してみよう．

$$c_1 \in x_1, \qquad c_2 \in x_2, \qquad \ldots, \qquad c_m \in x_m$$

この仮定と，与えられた論理式を合わせると，連鎖的に，他の関係が導出される．すなわち，

- $c \in x_i$ かつ $x_i = x_j \cap x_k$ ならば，$c \in x_j$ と $c \in x_k$ が成り立つ．
- $c \in x_j$ かつ $c \in x_k$ かつ $x_i = x_j \cap x_k$ ならば，$c \in x_i$ が成り立つ．

グラフの幅優先探索の要領で同じ導出を繰り返さないよう注意すれば，各 c_i について $O(n)$，全体で $O(n^2)$ 時間でこのような帰結がすべて求まる．こうして $c_a \in x_b$ と $c_b \in x_a$ が導出されたとすると，x_a に属する要素はすべて x_b に属し，また逆も成り立つことがわかるため，$x_a = x_b$ がわかる．逆に

この等号条件が導出されなかったペアは，$x_i = \{c_a \mid c_a \in x_i \text{が導出された}\}$ という割当てをすれば，入力の $\cap$ に関する論理式はすべて満たしつつ，異なる集合を割当てる割当てを具体的に構成できる．

以上の計算で得られた同値類分解を使って，変数名を同値類の代表元に付け替えると，変数名が違えば必ず集合としても違う，と想定してよくなり，矛盾のチェックは簡単になる．以下のステップで，この問題全体を $O(n^2)$ の計算時間で解くことができる．

1. まず $x_i = x_j \cap x_k$ 型の論理式を処理して，変数同士の $=$ 関係をすべて導出する．
2. 上で得られた同値類分解を使って，変数名を同値類の代表元に付け替える．
3. $x_i \in x_j$ 型の論理式から $x_i = x_j \cap x_k$ 型の論理式を使って導出される $\in$ 関係をすべて求める．（$=$ 関係の導出の際の幅優先探索的な議論と同様の計算．）
4. 同じ変数ペアに対して $\in$ と $\notin$ の両方の論理式がないことをチェックする．
5. $\in$ によるループがないかチェックする．

Problem L: Common Tangent Lines

原案: 楠本 充　　主担当: 岡 智洋　　解説: 岡 智洋

4 本の直線を 2 本の内接線と 2 本の外接線に分けるような組み合わせを全て試すとする。それらを平行移動して 4 本の共通接線を持つような 2 個の円が存在するかを確認して、そのときの平行移動の最小距離を求める。

共通接線を持つような 2 個の円が存在する必要条件として、2 本の外接線は平行であってもよいが、2 本の内接線は平行であってはならない。また、内接線と外接線が平行になることもない。以降では、この必要条件を満たしているケースだけを考える。

2 本の内接線は 2 つの円の中心を結ぶ直線で線対称の位置に存在するはずである。線対称となる軸はその交点の角の二等分線であり、2 通り存在する。同様に 2 本の外接線も 2 つの円の中心を結ぶ直線で線対称の位置に存在する。2 本の外接線がある 1 点で交差する場合は内接線と同様に 2 通りの角の二等分線が存在する。2 本の外接線が平行なときは、2 つの円が直線で挟まれているので、線対称の軸は外接線の中間を通るような平行な等距離線が存在する。

2 本の直線 $(a_i, d_i), (a_j, d_j)$ の対称軸は

$$\alpha := \frac{a_i + a_j}{2}$$
$$\beta := \frac{a_i - a_j}{2}$$

を用いて以下のように得られる。

$$x \cos\left(\frac{\pi\alpha}{180}\right) + y \sin\left(\frac{\pi\alpha}{180}\right) = \frac{d_i + d_j}{2\cos\left(\frac{\pi\beta}{180}\right)}$$

2 本の直線が交点を持つときはもうひとつの対称軸が次の式で得られる。

$$x \cos\left(\frac{\pi(\alpha + 90)}{180}\right) + y \sin\left(\frac{\pi(\alpha + 90)}{180}\right) = \frac{d_i - d_j}{2\sin\left(\frac{\pi\beta}{180}\right)}$$

与えられた直線を平行移動するとき、この対称軸も平行移動するので、内接線の対称軸と外接線の対称軸で角度が等しい組が存在するのならば、4 本の直線が共通接線となるように平行移動することが可能である。

1 本の内接線を平行移動させるとその距離に比例して内接線の対称軸も平行移動する。同様に外接線とその対称軸の平行移動の距離は比例している。そのため、対称軸を一致させるための最小距離となるような平行移動は、1 本の直線だけを平行移動させれば良い。$d_1, \cdots, d_4$ それぞれについて、増減させて対称軸を重ねるような距離を求める。その最小値が求めるべき平行移動の距離である。