Problem Analysis
The 2026 ICPC Asia Japan Online First-Round Contest

Problem A: 最強のカードを見つけろ

原案: 稲葉一浩　　主担当: 稲葉一浩

様々な解き方がある．例えば，入力に 2 が含まれていれば 2，そうではなく 1 が含まれていれば 1，そうでなければ最大値を返す．あるいは，入力された数値どうしを直接比較するのではなく，(入力 + 10) を 13 で割った余りで比較して最大の要素を返す．

Problem B: 自動販売機

原案: 楠本 充　　主担当: 楠本 充

自販機を左から貪欲に置くことで $O(n)$ 時間で解ける．具体的には，最左の扉の位置 x に対して $x+d$ に自販機を置き，この自販機によってカバーされた扉を除去することを，扉がなくなるまで行えばよい．部屋の長さは 10^8 までなので $x+d$ が 10^8 を超えるときは，自販機を $x+d$ ではなく 10^8 に置く必要があるが，答えには影響しない．

n の上限が小さいので，$O(n^2)$ 時間などの実装も正答になる．

Problem C: 残った水

原案: 石畑 清　　主担当: 山口 文彦

底が段々になっている水槽を作って水を入れたら，両端の板が外れて水が流れ出てしまった．残った水の量を求めよ，という問題である．

区間 k に水が残る条件は，区間 k よりも浅い (深さが小さい) 区間が，区間 k の両側にあることである．すなわち，$\min_{1\le i<k}(d_i) < d_k$ かつ $\min_{k<i\le n}(d_i) < d_k$ を満たすことである．水が残る場合，水が流れ出たあとの区間 k の水面の高さは，$\max(\min_{1\le i<k}(d_i), \min_{k<i\le n}(d_i))$ となる．これが d_k よりも小さいときに，その差の分だけ水が残っている．この値を k について合計すればよい．

k のそれぞれについての繰り返しの中で，区間 k の両側の区間の底の深さの最小値を求める手順の漸近計算量は，$\mathcal{O}(n^2)$ となるだろうが，$n \le 100$ という制約があるので，これでも問題なく正解となる．

すべての区間について，それぞれの区間 i よりも水槽の一方の端に近い区間の深さの最小値を求める (すなわち，$1 \leq i \leq n$ のすべてについて，$\min_{1 \leq j \leq i}(d_j)$ を求める) 手順は，最小値を求める手順の暫定最小値を記録していくと考えれば，$\mathcal{O}(n)$ である．同様に，区間 i よりも他方の端に近い区間の深さの最小値も $\mathcal{O}(n)$ で，すべての区間について求められる．各区間 k について，これらの大きい方と d_k の差を求めれば，全体の漸近計算量は $\mathcal{O}(n)$ となる．

Problem D: 回数数列

原案: 岡 智洋　　主担当: 岡 智洋

この問題では，数列の規則性を上手く見つけ出すことが重要である．

「何が何回出現したか」の配列の変化を追っていくと，s が出現するたびにパターンが変わっていると考えることができる．

例として，$s = 5$ の場合の数列は以下のようになる．

$$
\begin{aligned}
&\underline{5},\\
&1,1,2,1,3,1,4,1,\underline{5},\\
&2,2,3,2,4,2,\underline{5},\\
&3,3,4,3,\underline{5},\\
&4,4,\underline{5},\\
&\underline{5},\\
&6,1,6,2,6,3,6,4,6,\underline{5},\\
&7,1,7,2,7,3,7,4,7,\underline{5},\\
&\vdots
\end{aligned}
$$

$i+1$ 回目 $(0 \leq i \leq s)$ に出現する s は $s^2+1-(s-i)^2$ 項目にあり，さらにその後は $s^2+1+2sj$ $(1 \leq j)$ 項目の値が s になる．s の出現した直後で区切って，区間には順番に 0 から始まる番号をつける．

第 k 項がどの区間に入っているかを求める．k が s^2+1 以下の場合は，$s^2+1-(s-i)^2$ が k 以上になる最小の i を二分探索または i についてのループで求めることができる．k が s^2+1 より大きい場合は，$k-s^2-2$ を周期の $2s$ で割ってその整数部分から求めることができる．

区間 0 について，これは a_1 のみの区間で，その解は s である．区間 i $(1 \leq i \leq s)$ について，この区間の長さは $2s-2i+1$ である．この区間を $b_1, \ldots, b_{2s-2i+1}$ としたとき，b の奇数番目の値は i から 1 ずつ増加する．偶数番目の値は全て i である．つまり $b = (i, i, i+1, i, i+2, \ldots, s-1, i, s)$ である．区間 i $(s+1 \leq i)$ について，この区間の長さは $2s$ である．この区間を $c_1, \ldots, c_{2s}$ としたとき，c の奇数番目の値は全て i である．偶数番目の値は 1 から順番に増加する．つまり $c = (i, 1, i, 2, \ldots, i, s)$ である．

Problem E: 買い物上手

原案: 米田 優峻　　主担当: 米田 優峻

まず，特典ジェムの個数 b_i の合計が $n-1$ 以上である場合を考える．最初に $b_i \geq 1$ である酒を最初にコインで購入すると，必ず追加のコイン無しで，すべての酒を購入することができる．具体的には，得られた特典ジェムを使って，$b_i \geq 1$ の酒を優先的に買い続けると良い．

次に，特典ジェムの個数 b_i の合計が $n-2$ 以下である場合を考える．合計を S として，最初にコインで $n-S$ 本の酒を購入する必要がある．たとえば合計が $n-2$ の場合は 2 回である．ここで最初にコインで買う酒は，最低 1 つは $b_i \geq 1$ であるべきだが，それ以外は $b_i = 0$ でも良い．逆にそれさえ満たせば，必ず追加のコイン無しで，すべての酒を購入する方法が存在する (証明は合計 $n-1$ の場合と同様に考えればよい)．

以上をまとめると，次のプログラムを実装すればよい．基本的には最も安い $n-S$ 本の酒をコインで購入するが，もし買った酒がすべて $b_i = 0$ となった場合，1 本を最も安い $b_i \geq 1$ の酒に置き換える．ただし，b_i が全部 0 の場合，$b_i \geq 1$ の商品が存在しないため，適切に場合分けする必要がある．なお，答えが最大で 10^{14} となるため，long long 型などを使う必要がある．

Problem F: 地図アプリの高速化

原案: 米田 優峻　　主担当: 米田 優峻

この問題は，一見典型的な最短経路問題に見えるが，一辺の長さ n が最大 10^9 と大きい．そのため，単に計算量 $O(n^2)$ の幅優先探索を使うのは現実的ではない．重要な考察は，図 1 中央に示す通り，建物の境界となる x 座標および y 座標で区切ることである．建物は全部で m 個あるので，およそ $2m \times 2m$ 個の領域に区切られる．このようなテクニックは座標圧縮と呼ばれる．

すると，中央駅から，図 1 で青色で示された「各領域の四隅」までの最短距離を求めておけば，答えを出力することができる．なぜなら，目的地がある領域 A の内部であった場合，中央駅から目的地までの最短経路のうち，領域 A の四隅のいずれかを必ず通るものがあるからである．ここで，青色部分までの最短距離を求めるには，単に青色の区画を頂点とし，縦横で隣り合う頂点同士を辺で結んだグラフ上で，ダイクストラ法を行えばよい．計算量は $O(m^2 \log m)$ となる．

この問題の最大の関門は実装である．大きさが 1×1 である中央駅の処理を含め，国内予選にしては実装量がやや多いだけでなく，建物同士が辺や角で接する場合や，目的地が領域の境界上にある場合など，特殊なケースにも対応できる正確なプログラムを実装することが要求される．したがって解法にたどり着いても，時間内に正解を出せなかったチームが多いと考えられる．

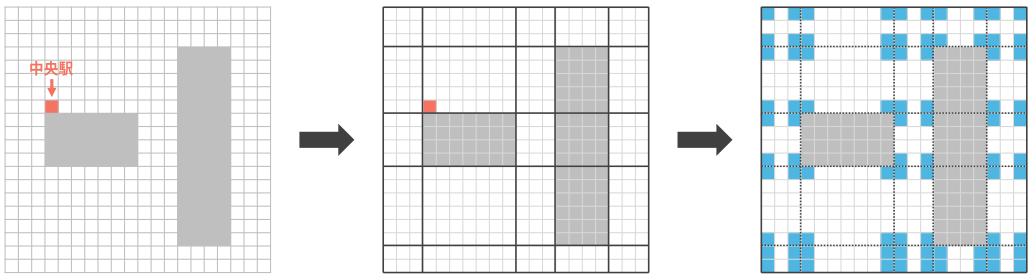

図 1　境界となる座標で区切る例．青色で示された区画までの最短距離を求めればよい．

Problem G: 衝突回避

原案: 天海 大志　　主担当: 天海 大志

2 つの駒の経路を固定したとき，2 つの駒が同じ時刻に同じマスに存在しうるとすれば，そのマスは必ず $h := \left\lfloor \frac{n}{2} \right\rfloor + 1$ 行目にある．白い駒の経路が h 行目で通る列を $a, a+1, \ldots, b$，黒い駒の経路が h 行目で通る列を $c, c+1, \ldots, d$ とする．このとき，2 つの駒が同じマスに存在してはならないので，これらの区間は交わってはならない．すなわち，条件を満たす経路は $a \leq b < c \leq d$ または $c \leq d < a \leq b$ が成り立つ必要がある．

あらかじめ，$(1,1)$ から $(h-1,j)$ への経路数 U_j，$(h+1,j)$ から (n,m) への経路数 V_j，$(n,1)$ から $(h+1,j)$ への経路数 X_j，$(h-1,j)$ から $(1,m)$ への経路数 Y_j を求めておく．a,b,c,d を固定したときの答えへの寄与は，h 行目の $a,\ldots,b$ 列目および $c,\ldots,d$ 列目にすべて障害物がない場合に限り $U_a V_b X_c Y_d$ である．

条件を満たすすべての a,b,c,d に対する寄与の総和は状態 DP によって計算できる．$a \leq b < c \leq d$ の場合なら，左から右へ走査しながら「まだ a を決めていない」「a まで決めた」「b まで決めた」「c まで決めた」「d まで決めた」というように，どこまで決めたかを状態として持てばよい．

時間計算量は $O(nm)$ である．なお，U,V,X,Y を求める処理を関数化しておくと，実装が簡潔になるうえ，添字や向きの取り違えによるミスも減らせる．ただし，実装方法によっては $n=2$ がコーナーケースになりうるので注意すること．

Problem H: 浮き輪の色分け

原案: 佐藤 遼太郎　　主担当: 佐藤 遼太郎

はじめに，どの棒にどの色を集めるかをあらかじめ決め打ちした場合の解法を考える．この場合，各棒の一番下から連続している，最終的にその棒に集める色の浮き輪は全く動かす必要がない．そして，それ以外の浮き輪にはそれぞれ 1 回以上の操作を行う必要がある．ここで，これらの動かす必要がある浮き輪をいったんすべて置き場に置き，その後置き場から 1 個ずつ戻す，という単純な戦略が存在する．この戦略は浮き輪 1 個あたり 2 回の操作で済む．また，浮き輪 1 個に 3 回以上の操作を費やすメリットがないこともわかる．したがって，操作 1 回だけで済む浮き輪の個数を最大化する問題を解けばよい．

ある浮き輪が 1 回の操作で済むのは，その浮き輪を移動する時点で，移動先の棒から余計な浮き輪がすべて取り除かれている場合である．したがって，どの棒から余計な浮き輪を追い出していくかという順序を最適化すればよい．これは，すでに追い出しを終えた棒の集合をキーに持つ DP で計算できる．次に追い出す棒を 1 本選び，その棒から取り除く浮き輪のうち，行き先の棒がすでに追い出し済みであるものは操作回数を 1 回，そうでないものは操作回数を 2 回として足せばよい．各棒に含まれる各色の個数を前計算しておけば，この DP は $O(2^n n)$ 時間で行える．

色の決め打ちを行わない（本来の問題の）場合も，どの棒にどの色を割り当てるか決めながら同様の DP を行える．具体的には，$dp[S][T]$ を，すでに追い出しを終えた棒の集合が S，すでに使っ

た色の集合が T であるときの，それまでの操作回数の最小値とする．まだ整理していない棒 $i \notin S$ を，まだ使っていない色 $c \notin T$ の棒にするとき，棒 i の下から連続する色 c の浮き輪の個数と，棒 i に含まれる色集合 T の浮き輪の個数がわかっていれば，$dp[S \cup \{i\}][T \cup \{c\}]$ の更新は $O(1)$ で行える．最終的な答えは $dp[\{1,\ldots,n\}][\{1,\ldots,n\}]$ である．

DP の状態数は，$|S| = |T|$ の状態のみを考えればよいので $\sum_{i=0}^{n} \binom{n}{i}^2 = \binom{2n}{n} \in O(4^n/\sqrt{n})$ である．よって全体の計算量は $O(nm + \binom{2n}{n}n^2)$ であり，実装上も十分高速に動作する．なお，遷移をうまく計算することで更に $O(nm + \binom{2n}{n}n)$ に改善できるが，本問題では必須ではない．

Problem I: 制限速度

原案: 米田 優峻　　主担当: 米田 優峻

使えるコストを $L = 2, 4, 6, 8, \ldots$ と 2 ずつ上げていくことを考える．使えるコストが 2 増えると，ある区間の時速を 1km/h 上げることができるが，その中で制限時間を遵守しつつ，最も合計時間が短くなるようなものを選択し続けるという貪欲法を考える．たとえば $a = (4, 4, 1, 4, 4, 5, 4, 4)$ および $L = 10$ の場合，表 1 のように速度を上げていくことになる．特にコストを 8 から 10 に上げる際は，左側の時速 2km/h の区間を速くすると約 0.33 時間，右側の時速 3km/h の区間を速くすると約 0.42 時間短くなるため，右側を速くしている．

表 1　コストの増加に対する各区間の時速の変化．時速が増加した部分を強調している．単位は km/h.

コスト	0-1km	1-2km	2-3km	3-4km	4-5km	5-6km	6-7km	7-8km	所要時間
2	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	8.00
4	1	1	1	**2**	**2**	**2**	**2**	**2**	5.50
6	**2**	**2**	1	2	2	2	2	2	4.50
8	2	2	1	**3**	**3**	**3**	**3**	**3**	3.67
10	2	2	1	**4**	**4**	**4**	**4**	**4**	3.25

実は，この貪欲法が必ず最適解を出すことが証明できる (後述)．しかし，本問題は $L \leq 10^{10}$ である一方，単純な解法では $O(nL)$ 時間を要するため，実行時間制限に間に合わない．そこで，コストを 2 増やした際の時間の短縮分 (以下，**差分**と記す) で二分探索することを考える．すなわち，差分が実数 x 以上となるような改善をすべて行った際に，コストが何必要であるかを高速に求める関数を作成し，それがちょうど L となるような実数 x を二分探索する．

コストの計算を行う方法を，$a = (40, 40, 10, 40, 40, 50, 40, 40)$ および $x = 0.01$ の場合をベースに説明する (a は前述の例の 10 倍である)．まず，貪欲法において，最初は高速道路の全区間を増やすことになるが，制限速度の最小値の 10km/h まで増やしても，差分が x を超えるため 10km まで増やす．すると，まだ速度を伸ばせる区間が左側の 2km，右側の 5km に分断される．

- 左側については，制限速度の最小値が 40km/h である．また，14km/h から 15km/h まで増やす段階で差分が x を下回るため，時速を 14km/h まで増やす．
- 右側については，制限速度の最小値が 40km/h である．また，22km/h から 23km/h まで増

やす段階で差分が x を下回るため，時速を 22km/h まで増やす．

ここで，14km/h などの境界は，二次方程式によって簡単に計算できる．最終的に，各区間の時速が $(14, 14, 10, 22, 22, 22, 22, 22)$ となり，コストが 52 であると分かる．このように，今見ている区間の時速を増やせるまで増やし，もし差分の限界に達する前に制限速度の最小値に当たったら，区間を分割して再帰的に解いていくと，コストを計算することができる．制限速度の最小値についてはセグメント木を用いると $O(\log n)$ で計算することができるため，コスト計算には $O(n \log n)$ 時間を要する．二分探索を加味しても，実行時間制限には間に合う．

■注意点　実装上の注意点として, 差分が同じ場合の処理が挙げられる. たとえば $a = (2, 1, 2), L = 4$ の場合，コストを 2 から 4 に上げるタイミング，コストを 4 から 6 に上げるタイミングの両方で差分が 0.5 となるため，コストがちょうど L となるような差分 x が存在しない．そのような場合はコストが L 以上となるような最小の x を計算したうえで，L を超えた分について所要時間を補正する必要がある．また，二分探索の範囲および回数にも注意が必要である．

■貪欲法の証明　最後に貪欲法の最適性を証明する．まず，制限速度の上限をヒストグラムのような形にし，それを横で切った図 2 左のようなものを考える．また，実際の走行速度も，ヒストグラムのような形で青で塗ることを考える．そのとき，横で切ったパーツの一部しか青で塗られない図 2 右のような方法は明らかに最適ではない (そのようなパーツの中で一番下のものを全部塗ると，コストを変えずに所要時間を短くすることができる)．ここで，横で切った各パーツ全体を塗るごとにコスト 2 が必要となるが，塗ることで所要時間がどれくらい短くなるかを考える．すると，下の方が影響が大きく，上の方が影響が小さいという単調性が成り立つため，貪欲法が最適となる．

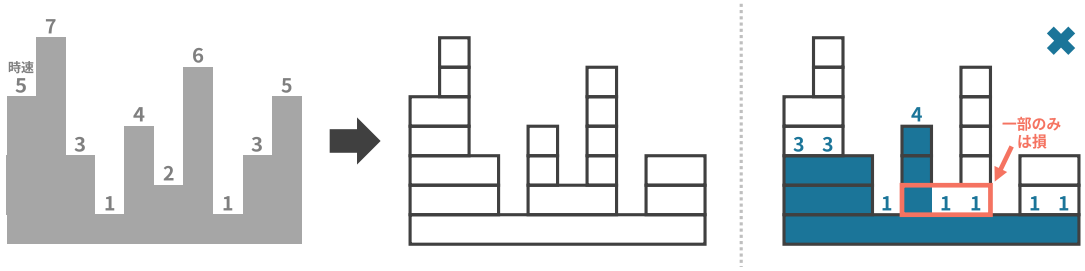

図 2　証明のスケッチ．左: 速度のヒストグラムを横で切る例．右: パーツの一部のみを青で塗るような走行速度の例.

Problem J: 最大の倍率

原案: 楠本 充　　主担当: 楠本 充

拡大率 s は二分探索することにして，以降は値を固定して考える．Q' ではなく P を平行移動させて Q' が P に含まれるようにできるかを考えても問題は変わらないので，そのように考える．

P の周上および内部の領域は，n 個の半平面の共通部分により表すことができる．具体的には P の各辺を伸ばして直線にして，P を含む方の半平面を取ればそのようにみなせる．

これらの半平面のうち一つを $ax + by \leq c$ とする．この半平面を x 方向に r，y 方向に s だけ動かすと

$$a(x - r) + b(y - s) \leq c$$

つまり

$$ax + by - c \leq ar + bs$$

となる．

P が Q' を含む必要十分条件は，すべての n 個の半平面たちが Q' の頂点をすべて含んでいることである．m 個の対 $(x, y) = (x'_1, y'_1), \ldots, (x'_m, y'_m)$ を上の不等式に代入すると，1 つの半平面についての条件は

$$\max_{j=1,\ldots,m} (ax'_j + by'_j - c) \leq ar + bs$$

に等しいことがわかる．左辺はの値は係数 a, b, c が既知なら $O(m)$ 時間で求められる．これにより，1 つの半平面に対して平行移動成分 (r, s) の存在領域 (これも半平面である) が分かるので，すべての半平面について (r, s) の存在領域を求め，共通部分が空であるかどうかを見れば元の問題は解ける．共通部分が存在するかどうかの判定は色々な方法が存在する．効率の良い $O(n \log n)$ 時間の手法も存在するが，この問題では凸多角形の切断などの手法を使って $O(n^2)$ 時間で処理しても許容される．

全体として，

- P のそれぞれの辺について，半平面の係数 a, b, c を求める: $O(n)$ 時間
- 二分探索は要求精度 ε に対して $O(\log(1/\varepsilon))$ 回行う
- (r, s) の存在領域を表す半平面の式を n 個求める: $O(nm)$ 時間
- (r, s) の存在領域が空であるかを求める: $O(n^2)$ 時間

とすることで，$O(\log(1/\varepsilon)n(m + n))$ 時間で問題を解ける．