square869120Contest #6 解説

Writer: E869120, square1001

はじめに

- 今回のコンテスト、どうでしたか？？？
- 何と前回大会の s8pc #5 の 3 倍を超える 669 名に参加していただきました。
  - 運営としては驚き、嬉しいばかりです。
- 今回は最後の 1 秒までデッドヒートになりました。
- 問題が面白ければ幸いです！

講評

ID	問題名	配点	平均点	正解者数	予想難易度
A	E869120, who leaps through time	200	195.9	646	200
B	AtCoder Market	300	217.6	464	300
C	Infinite Grid	400	223.6	312	400
D	Snowballs	600	97.1	86	700
E	90-degree Rotations	800	98.1	64	800
F	Random Shuffles	1000	23.5	2	1600
G	Medals	1200	37.9	13	1200
H	Percepts of AtCoder 2	1500	29.4	8	1300
I	Garden 2	1500	122.4	0	-

A – E869120, who leaps through time

Square869120contest #5 解説

おわび

- この問題を最初に見た人が多いのではないでしょうか。
- 実は、この問題は問題名が異常に長いです。

- E869120, who leaps through time (空白含め 31 文字)
  - おそらく過去の s8pc の中で最も問題名が長いでしょう。

- という訳で、解説に進んでいきます。是非お読みください！

問題概要

- 高橋王国は、N 個の都市から成る。都市 i と都市 $i+1$ の間を移動するのに、A_i 分かかる。
- E869120 君は、時刻 0 に都市 1 を出発し、時刻 0 までに都市 N に着く必要がある。しかし当然こんなことは物理的に不可能である。
- 彼は特殊能力「タイムリープ」を使える。1 回使うと、時刻が T 秒戻る。
- 最小で何回の「タイムリープ」で目的を達成できるか？

入力例 2 の説明

時刻: 0

※ 1 回のタイムリープで 4 分戻る

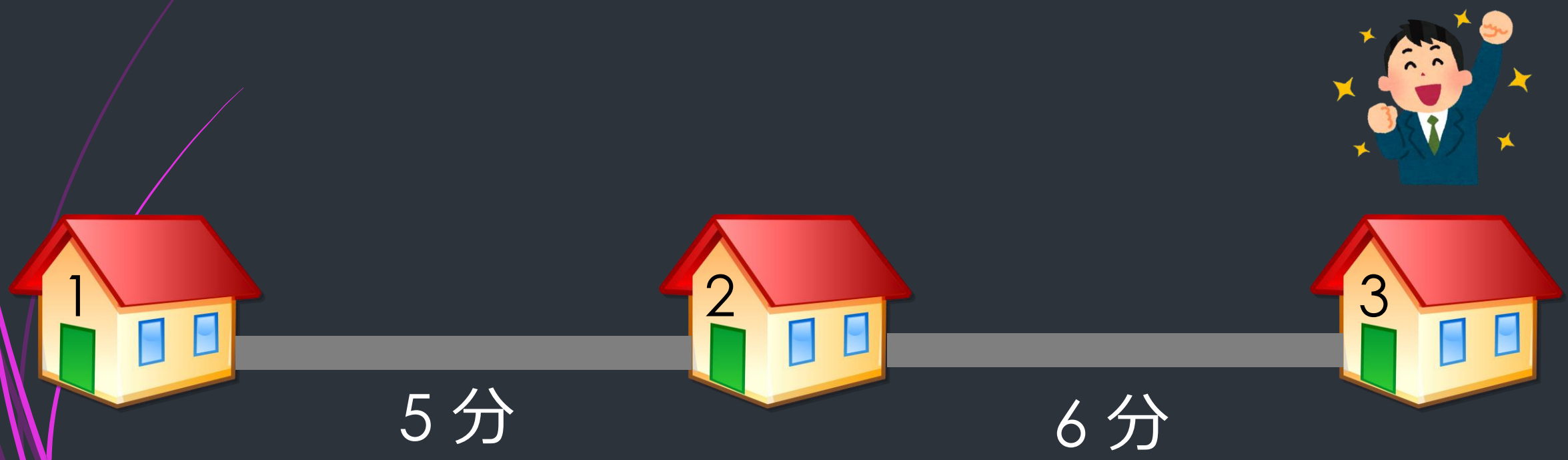
入力例 2 の説明

時刻: 0 → 5

※ 1 回のタイムリープで 4 分戻る

入力例 2 の説明

時刻: 5 → 11



※ 1 回のタイムリープで 4 分戻る

入力例 2 の説明

時刻: 11 → 7 → 3 → -1
(3 回タイムリープ)

※ 1 回のタイムリープで 4 分戻る

入力例 2 の説明

時刻: 11 → 7 → 3 → -1
(3 回タイムリープ)

よって、3 回のタイムリープで目的を達成できる

※ 1 回のタイムリープで 4 分戻る

小課題 1 (100 点)

- 特殊な制約がある
- $N = 2, T = 1$

- 都市 1 を時刻 0 に出発すると、都市 2 に時刻 A_1 に到着する
- 都市 2 でタイムリープをするとき、時刻 t から時刻 0 に戻るためにはタイムリープを t 回する必要がある
- だから、タイムリープの必要回数の最小値は A_1 回！

- この小課題は、値を入力して A_1 を出力するだけで解ける！

100 点

小課題 2 (200 点)

- 都市 1 から N までにかかる合計時間を S とし、タイムリープ 1 回で時刻を T 秒戻せる場合、目的を達成するために何回のタイムリープをする必要があるか？
- 実験してみよう！　例えば $S = 9, T = 4$ の場合：
  - 時刻 9 → 時刻 5 → 時刻 1 → 時刻 -3、つまり最小で 3 回！
- 実は最小回数は $\frac{S}{T}$ の小数点以下切り捨てなので、int 型でも $(S + T - 1)/T$ で計算できる

小課題 2 (200 点)

- 例えば入力例 2 の場合、
  - 都市 1 から N までの移動時間は $5 + 6 = 11$ 分
  - タイムリープの最小回数は $11/4$ の小数点以下切り上げ
  - $(11 + 4 - 1)/4$ の小数点以下切り捨てと同じ → 答えは 3 回

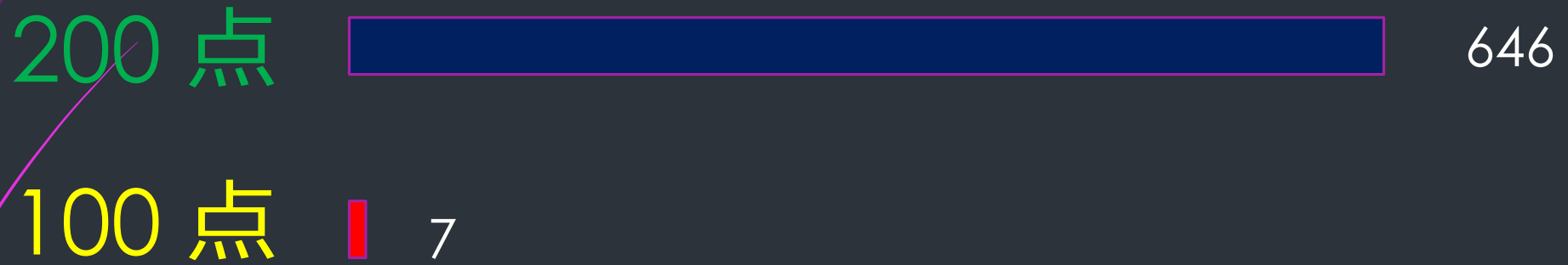
- 都市 1 から N までの移動時間は for 文を使うと簡単に求められる
  - よって 200 点獲得！　おめでとうございます！

200 点

サンプルコード

```
1. #include <iostream>
2. using namespace std;
3.
4. int N, T, S, A;
5.
6. int main() {
7.         cin >> N >> T;
8.         for (int i = 1; i <= N - 1; i++) {
9.                 cin >> A; S += A;
10.        }
11.        cout << (S + T - 1) / T << endl;
12.        return 0;
13. }
```

得点分布



平均点： 195.9 点

※ 0 点獲得者を含む

B – AtCoder Market 解説

Square869120contest #5 解説

問題概要

- 数直線があって、その上に「入口」、「出口」を 1 つ設置する
  - 入口、出口は整数座標にしか設置できない
- N 人の人がいる
  - i 番目の人は、「入口」→ (座標 a_i) → (座標 b_i) →「出口」の順で移動する
  - 移動方法は、a_i, b_i を逆にしてもよい
  - 座標 x から y に移動するときの距離は $|x - y|$
- 入口、出口をうまい位置に設置して、合計の移動距離を最短にしたい

問題概要

- 制約のリストアップ
  - $1 \leq N \leq 30$
  - $1 \leq a_i < b_i \leq 10^9$ で整数
- 部分点 (195 点)
  - $1 \leq a_i < b_i \leq 100$

考察 #1 – 実は「入口 → a_i → b_i → 出口」

- 入口の座標を s、出口の座標を t とする
  - ここで $s \leq t$ が成り立っていると、うれしいことが起こる！
- 実は、$s \leq t$ だと、どの人についても最短経路が「$s \to a_i \to b_i \to t$」になる！
  - ここでは $a_i \leq b_i$ の制約を使っていることに注意
  - $s \to \min(s, t, a_i, b_i) \to \max(s, t, a_i, b_i) \to t$ の経路と結果的に変わらないから、これが最短経路になること分かる
- つまり、最短距離は $|s - a_i| + |a_i - b_i| + |b_i - t|$ になる

小課題 1 (195 点)：$1 \leq a_i < b_i \leq 100$

- s, t がとりうる値は $1, 2, 3, \ldots, 100$ のいずれかだから、(s, t) としてありうる組み合わせは $10\,000$ 通り
- これを「全探索」する
  - この中で $|s - a_i| + |a_i - b_i| + |b_i - t|$ の合計の最小値を求める
- 計算量は $10\,000 \times N$ くらいなので、実行時間制限に間に合う！

考察 #2 – 絶対値の合計の最小化

ここで問題です：

- N 個の数 $a_1, a_2, a_3, \ldots, a_N$ があります。$|x - a_1| + |x - a_2| + \cdots + |x - a_N|$ の最小値を求めなさい。

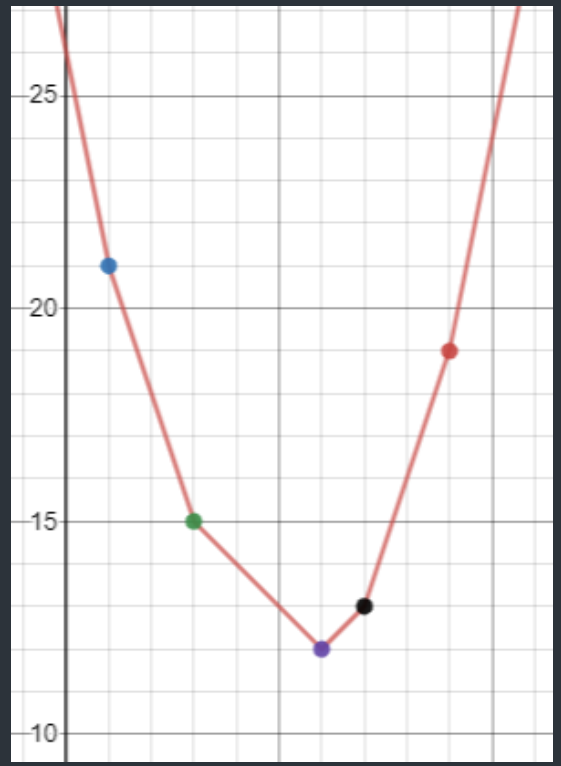
- 実は、最小値となる x は「$a_1, a_2, a_3, \ldots, a_N$ の中央値」となる！

考察 #2 – 絶対値の合計の最小化

- $a_1 \leq a_2 \leq a_3 \leq \cdots \leq a_N$ として、$y = |x - a_1| + |x - a_2| + \cdots + |x - a_N|$ のグラフを書くと、次のようになる
- $x \leq a_1$ のときの傾きは $-N$
- $a_1 \leq x \leq a_2$ のときの傾きは $-N + 2$
- $a_2 \leq x \leq a_3$ のときの傾きは $-N + 4$

  ：　　：　　：
- $a_N \leq x$ のときの傾きは N

- だから、中央値のとき最小になる



$|x - 1| + |x - 3| + |x - 6| + |x - 7| + |x - 9|$ のグラフ

満点解法

- 先ほどの式を眺めてみると、移動距離の合計は次の 3 つの合計になる
  - $|a_1 - b_1| + |a_2 - b_2| + |a_3 - b_3| + \cdots + |a_N - b_N|$
  - $|s - a_1| + |s - a_2| + |s - a_3| + \cdots + |s - a_N|$
  - $|t - b_1| + |t - b_2| + |t - b_3| + \cdots + |t - b_N|$
- 1 つ目は s, t によって変わらない
- 2 つ目、3 つ目は「考察 #2」の問題と全く同じで、「s が $a_1, a_2, a_3, \ldots, a_N$ の中央値、t が $b_1, b_2, b_3, \ldots, b_N$ の中央値」のときに最小になる
- 中央値はソートで $O(N \log N)$ で求められるので、全体の計算量は $O(N \log N)$
  - 中央値を $O(N)$ で求める方法もあるので、この問題は $O(N)$ でも解ける

満点解法 (別解)

- 「先ほどの解法は天才的で、思いつくわけがないのでは？」と思った人もいるかもしれませんが、別の解法もありますのでここで説明します

- s, t の候補は $a_1, a_2, a_3, \ldots, a_N, b_1, b_2, b_3, \ldots, b_N$ のいずれか
  - これは、入力例 2, 3 あたりで推測できそう
- つまり、合計で $2N \times 2N = 4N^2$ 通りの (s, t) の候補がある

- あとは小課題 1 の解法と同じように全探索すると、計算量 $O(N^3)$ でこの問題を解くことができる！

サンプルコード

- 満点解法
  - https://atcoder.jp/contests/s8pc-6/submissions/4966775
- 部分点解法
  - https://atcoder.jp/contests/s8pc-6/submissions/4968484

得点分布

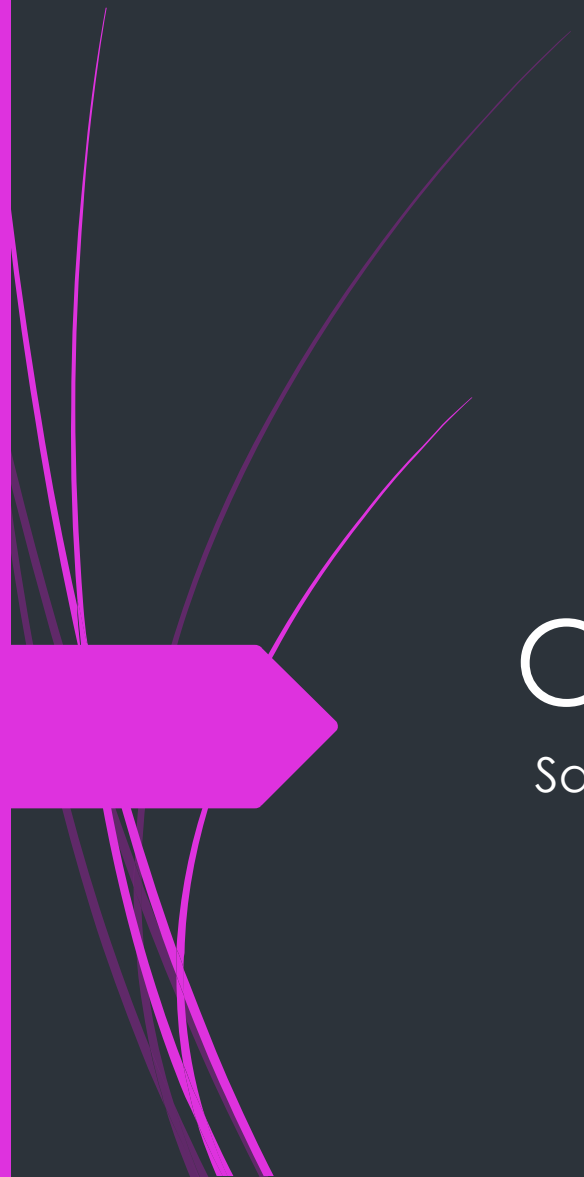
得点分布は以下の通りです

- 300 点：464 人
- 195 点：26 人

提出者数：518 人

First AC：**WA_TLE** (2 分 44 秒)

195 点がかなり少ないと思った (逆に 300 点が多くてびっくり)。これは全探索で通るので、解けなかった人は全探索の練習問題として解いてみてもいいかも？



C – Infinite Grid

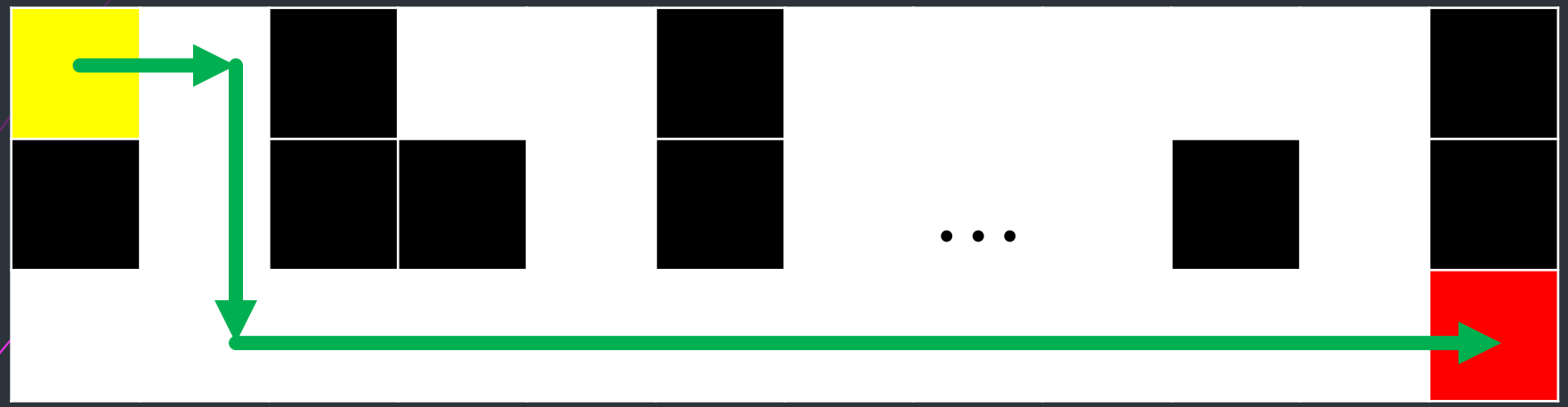
Square869120contest #5 解説

問題概要

- $H * W$ のグリッドが与えられます。
- グリッドが横に 1000000000 個繋がりました。

- この $H * 1000000000W$ のグリッドにおいて、左上から右下まで、右か下の 2 方向にしか進まずに白いマスだけを通っていけるかどうかを出力してください。

問題概要



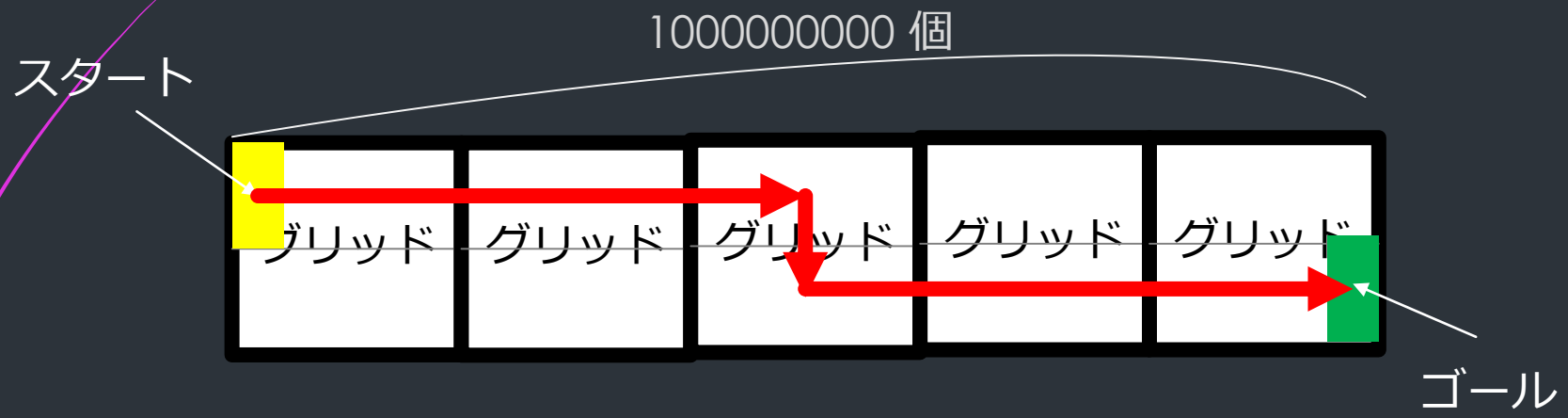
この場合は、行ける
（$H = 3, W = 3$ の例）

小課題 1 (60 点)

- $H = 1$
- つまり、グリッドは $1 * 1000000000W$
  - 縦の長さが 1 なので、右にしか動けない
- ということは…
  - どのような行き方でも全てのマスを通る
  - 1 個でも黒マスがあったらダメ
- つまり、$H * W$ のグリッドがあった場合に 1 個でも '#' のときに NO、それ以外は YES となる。簡単！

小課題 2 (60 + 130 = 190 点)

- $H = 2$
- 縦の長さが 2 の場合、どのような経路が可能か？



小課題 2 (60 + 130 = 190 点)

- $H = 2$
- 縦の長さが 2 の場合、どのような経路が可能か？

結局は、同じ $H * W$ のグリッド内で、1 列目 → 1 列目の移動と 2 列目 → 2 列目の移動のどちらかが発生する

すなわち、1 列目と 2 列目のどちらかは
全て白マスでなければいけない！

小課題 2 (60 + 130 = 190 点)

- つまり、以下の条件を全て満たせば大丈夫
  - 1 列目が全て白マスである場合 YES
    - その場合、一番最後の列までずっと 1 列目を進んで、最後の一手でしたにすすめばよい。これで行ける理由は、右下のマスは白であることが制約より保証されているから。
  - 2 列目が全て白マスである場合 YES
    - その場合、最初に 2 列目に進み、次に一番最後の列までずっと 2 列目を進めばよい。これで行ける理由は、左上のマスは白であることが制約より保証されているから。
  - そうでない場合 NO

満点解法

- $H, W \leq 100$
  - 一気に制約が大きくなるので、小課題 1・2 のように場合分けでは簡単に解けない
- 1 個大きい考察をする必要がありそう
  - 問題文には、「グリッドを横に 10^9 個繋げる」と書かれている
  - 本当に 10^9 個必要なのか？？？
  - 実は 1000 個くらいで良いのでは？？？

考察 1

- 「全て右に移動するグリッド」が 2 個連続するような動きに意味はない

満点解法

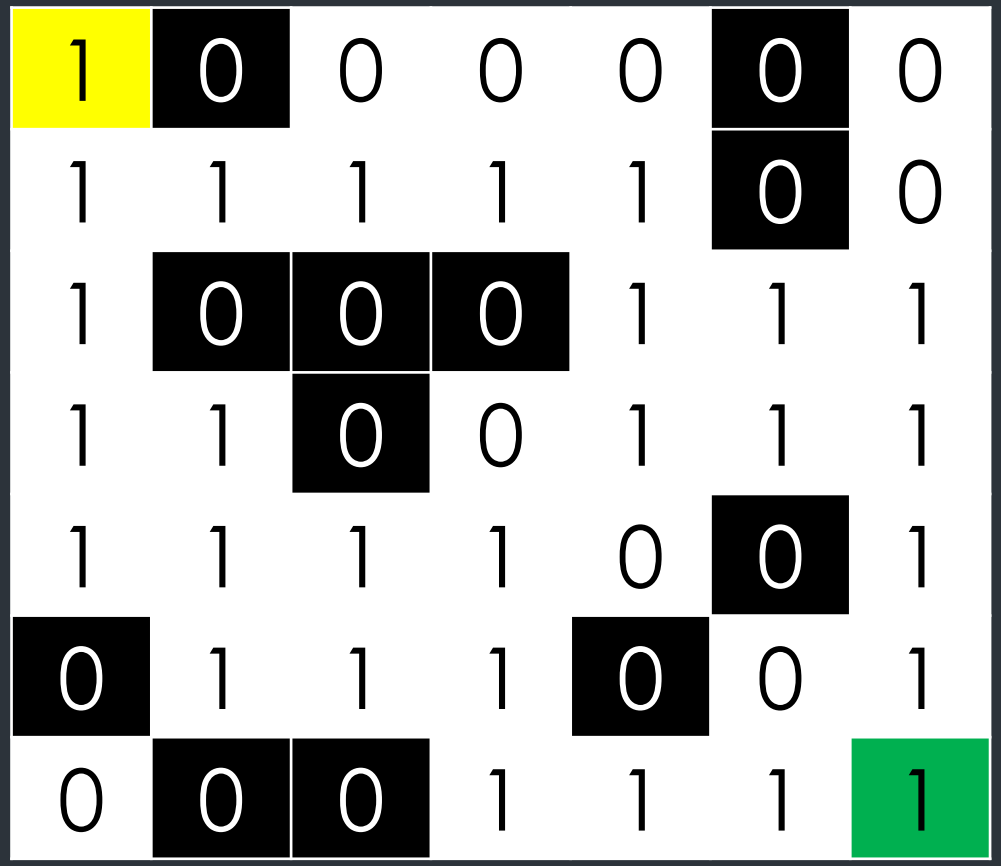
- 縦の移動は高々 $H-1$ 回
  - つまり、高々 $2(H-1)+1=2H-1$ 個のグリッドで出来る！
  - $10^9 \fallingdotseq 2H-1$
- あとは $H*W$ のグリッドを横に $2H-1$ 個繋げた $H*(2H-1)W$ のグリッドに対して、行けるかどうかの答えを求めるだけ

満点解法

- ではこの問題、解けるのでしょうか？
  - $R * C$ のグリッドが与えられる。各マスは白か黒で塗られている。左上のマスから右下のマスまで右か下方向に隣接するマスのみを辿って行けるかどうかを判定してください。
- これは動的計画法を用いて $O(RC)$ で解けます。以下のような漸化式です。
  - $dp[1][1] = 1$
  - 以下、$1 \leq i \leq R, 1 \leq j \leq C, (i,j) \neq (1,1)$ について、左上から順に計算していく
    - $dp[i][j] = 1$ [マス (i,j) が白かつ $dp[i-1][j], dp[i][j-1] = 1$ のどちらかを満たす場合]
    - $dp[i][j] = 0$ [マス (i,j) が黒または $dp[i-1][j], dp[i][j-1]$ が両方 0 の場合
    - ただし、マス目の外の DP 配列は全て 0 とする
- $dp[i][j]$ は左上からマス (i,j) に辿り着けるかを表す。$dp[R][C] = 1$ のとき、YES。

満点解法

- 図で表すとこんな感じ（$dp[R][C] = 1$ なので、答えは YES）



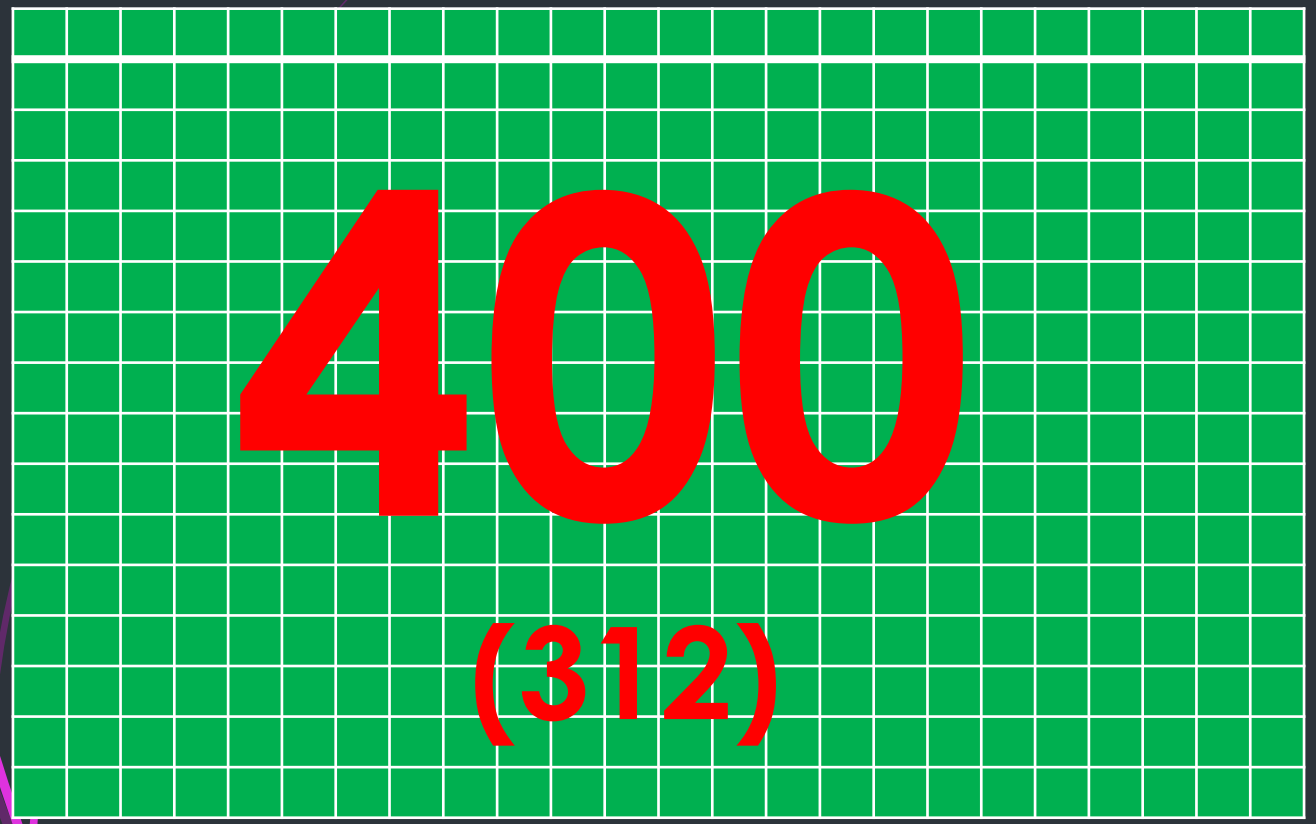
満点解法

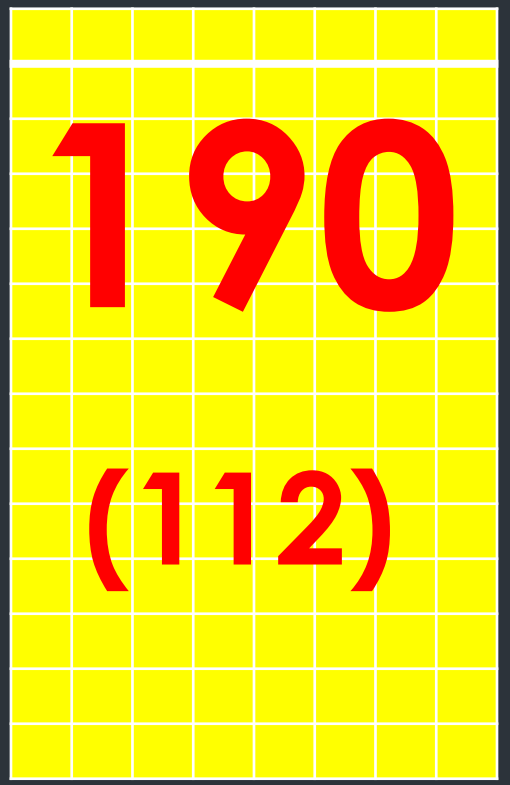
- 前述の、$H * W$ のグリッドを $2H - 1$ 個横に繋げた、$H * (2H - 1)W$ のグリッドに対して DP を行えばよい
  - $dp[H][(2H - 1)W]$ が 1 であれば ‘Yay!’ と出力する
  - そうでないなら ‘:(’ と出力する
- 計算量は $O(H(2H - 1)W)$ なので、余裕で間に合う
- 満点獲得

サンプルコード

- 満点解法（本解）
  - https://atcoder.jp/contests/s8pc-6/submissions/4934252
- 満点解法（別解）
  - https://atcoder.jp/contests/s8pc-6/submissions/4966804
- 小課題 1 (60 点)：https://atcoder.jp/contests/s8pc-6/submissions/4952271
- 小課題 2 (130 点)：https://atcoder.jp/contests/s8pc-6/submissions/4952352

得点分布





D – Snowballs 解説

Square869120contest #5 解説

問題概要

- 数直線上に N 個の雪玉があり、それぞれ (初期位置, 半径) = (X_i, R_i)
- 雪玉を転がすことができる
  - 位置を d 動かすと、半径が d 減少する (0 になったら消える)
- 同じ位置にある雪玉を合体できる
  - 半径 r_1 の雪玉と半径 r_2 の雪玉を合体すると、半径が $(r_1^3 + r_2^3)^{1/3}$ になる。
  - 体積が変わらず 1 個の球体になるという感じ
- つくれる最大の雪玉の半径はいくつか？

問題概要

- 制約は次のような感じ

1. (70 点、累計 70 点)：$N = 2$
2. (140 点、累計 210 点)：$1 \leq N \leq 15$
3. (250 点、累計 460 点)：$1 \leq N \leq 1\,000$
4. (140 点、累計 500 点)：$1 \leq N \leq 100\,000$

- それぞれの小課題について解説をしていきます！
- それでは、Let's Start!

小課題 1 (70 点)：$N = 2$

- 雪玉が 2 個 (ここでは $X_1 \leq X_2$ とする)

- やる戦略は次のいずれか

1. 雪玉 1 を「最大の雪玉」とする
2. 雪玉 2 を「最大の雪玉」とする
3. 雪玉 1, 2 を位置 x $(X_1 \leq x \leq X_2)$ にて合体させる

- 1, 2 の場合は簡単なので、3 の場合での最適解を求めたい

重要な考察

- 前提知識：
- 半径 r の雪玉の体積は、r^3 に比例
- 半径 r の雪玉の表面積は、r^2 に比例

- 半径 r の雪玉を転がしたときの「体積が減る速度」は、表面積に比例するから、すなわち **r^2 に比例する**

小課題 1 (70 点)：$N = 2$

- 次のような貪欲法を考える
- 「雪玉 1 と雪玉 2 のうち「体積が減る速度」が小さい方を、もう一方の雪玉に少し近づける」操作を、同じ位置に来るまで繰り返す
- 体積が減る速度は半径の 2 乗に比例するので、雪玉のうち半径の小さい方が、半径の大きい方に向かって動くことになる

- なぜこの方法が正しいのか？
  - 「半径が同じとき、体積が減る速度」は 2 個の雪玉で変わらないのと、体積が減る速度が単調減少だから

70 点

考察：$N \geq 3$ でも解きたい

- 今度は、2 個の雪玉が最初同じ位置にあったときに、$X = X_1 = X_2 \leq x$ を満たす座標 x に雪玉を運ぶことを考える
  - X の位置で合体させた方がいいのか？
  - それとも別の位置で合体させた方がいいのか？

- X の位置で合体させなかったとき
  - X の位置での「体積が減る速度」は $c \cdot (R_1^2 + R_2^2)$
- X の位置で合体させたとき
  - 半径は $(R_1^3 + R_2^3)^{1/3}$ となる
  - つまり、X の位置での「体積が減る速度」は $c \cdot (R_1^3 + R_2^3)^{2/3}$

考察：$N \geq 3$ でも解きたい

- さて、$c \cdot (R_1^2 + R_2^2)$ と $c \cdot (R_1^3 + R_2^3)^{2/3}$ の大小関係は？(c は比例定数なので $c > 0$)
  - 結局前者の方が後者より常に大きい (または等しい) ので、これを証明することを考える

【証明】

$R_1^2 + R_2^2 \geq (R_1^3 + R_2^3)^{\frac{2}{3}} \Leftrightarrow (R_1^2 + R_2^2)^3 \geq (R_1^3 + R_2^3)^2$

そこで、展開して両辺から $R_1^6 + R_2^6$ を引くと、$3R_1^2R_2^2(R_1^2 + R_2^2) \geq 2R_1^3R_2^3$

両辺を $R_1^2R_2^2$ で割って、$3(R_1^2 + R_2^2) \geq 2R_1R_2$

相加平均・相乗平均の不等式 $a + b \geq 2\sqrt{ab}$ に $a = R_1^2, b = R_2^2$ を代入する

すると、$3(R_1^2 + R_2^2) \geq R_1^2 + R_2^2 \geq 2R_1^2R_2^2$ が言え、不等式が証明できた

考察：$N \geq 3$ でも解きたい

- この不等式から分かること
  - 2 個の雪玉を合体させないより合体させる方が、どのような場合でも得になる
  - 雪玉を合体させるなら、できるだけ早いタイミングで合体させた方が良い
- $X_1 \leq X_2 \leq X_3 \leq \cdots \leq X_N$ として、$X_i \leq x \leq X_{i+1}$ に集めたものを「最大の大きさの雪玉」にすることを考える
- そのとき、X_i またはそれより西にあるものを合体させながら X_i に集め、X_{i+1} またはそれより東にあるものを合体させながら X_{i+1} に集めるのが良さそう
  - 逆方向に雪を進めても意味がないので
- そうすると、X_i の位置の雪玉と X_{i+1} の位置の雪玉の 2 個を合体させる問題になり、$N = 2$ の場合に帰着できる！

小課題 2 (140 点)：$N \leq 15$

- 結局、「集める場所から遠いところから順に近づけていき、ぶつかったら合体させる」のが最適であることが分かった！
- 「最大の大きさの雪玉に関与させる雪玉の選び方」は、2^N 通りある
- これを全探索して、それぞれに対して最適解を求めると、$O(2^N \cdot N^2)$ 程度の計算量で解くことができる

210 点

小課題 3 (250 点)：$N \leq 1\,000$

- 集める位置を全探索することを考える
- 先ほどの証明だと、雪玉を集める位置は $X_1, X_2, X_3, \ldots, X_N$ のいずれか

- そこで、次のような疑問が生じてくる

Question: 雪玉 $1, 2, 3, \ldots, i-1$ を位置 X_i に集めるとき、最大の半径は？

- どの段階でも半径が 0 にならなければ先ほどのアルゴリズムが使える
- 少し変更して半径が 0 になる場合も対処していきたい

小課題 3 (250 点)：$N \leq 1\,000$

- 先ほどのアルゴリズムは、「距離が遠い順から近づけていって、ぶつかったら合体させる」だが、実質的にはこれで OK
- 全ての雪玉が最終結果に関与すると思って、一番左の雪玉、一番右の雪玉から、最終的に集める場所に転がしていって、ぶつかったら合体させる
- 半径が 0 になったら次の雪玉にぶつかれないだけ (あるいはぶつかったときの半径を 0 とみなす) として、最適解が $O(N)$ で求められた
- 集める位置を全探索するので、計算量は $O(N^2)$

460 点

小課題 4 (140 点)：$N \leq 100\ 000$

- **この小課題が解ければ満点**
- 先ほどのアルゴリズムで、「雪玉 $1, 2, 3, \ldots, i-1$ を位置 X_i に持ってくるときの最適解」を求めていた
- これを高速に処理できないか？

† 動的計画法 (Dynamic Programming) †

小課題 4 (140 点)：$N \leq 100\,000$

- 雪玉 $1, 2, \ldots, i-1$ を地点 X_i に持ってきたときの最大半径を r とする
- そのとき、雪玉 $1, 2, \ldots, i$ を地点 X_{i+1} に持ってきたときの最大半径は？
  - まず半径 r の雪玉と半径 R_i の雪玉が合体して、半径 $\left(r^3 + R_i^3\right)^{1/3}$ に
  - その後、雪玉を $X_{i+1} - X_i$ 動かして、半径 $\max\left(\left(r^3 + R_i^3\right)^{1/3} - (X_{i+1} - X_i),\ 0\right)$ に
- したがって、$i = 1, 2, 3, \ldots, N$ のときの最大半径 r_i を順番に求めれば、全部が計算量 $O(N)$ で求まる！

※ただし、実数を 1/3 乗する操作が定数時間でできるものとする

小課題 4 (140 点)：$N \leq 100\ 000$

- 左から「集める位置」に持っていくときの最大半径は先ほどの動的計画法アルゴリズムによって求められた
- 右から「集める位置」に持っていくときの最大半径も、同様に求められる
- これら $2N$ 個の「最大半径」を前計算しておくと、「集める位置」の全探索の部分で計算量 $O(N)$ しかかからない
- よって、この問題は計算量 $O(N)$ で解けました！

600 点

サンプルコード

- 満点解法
  - https://atcoder.jp/contests/s8pc-6/submissions/4966862
  - （コード長 1027Byte、実行時間 222ms）

得点分布

- First AC：**hos_lyric** (18 分 56 秒)
- 満点の人数：86 人

E – 90-degree Rotations 解説

Square869120contest #5 解説

問題概要

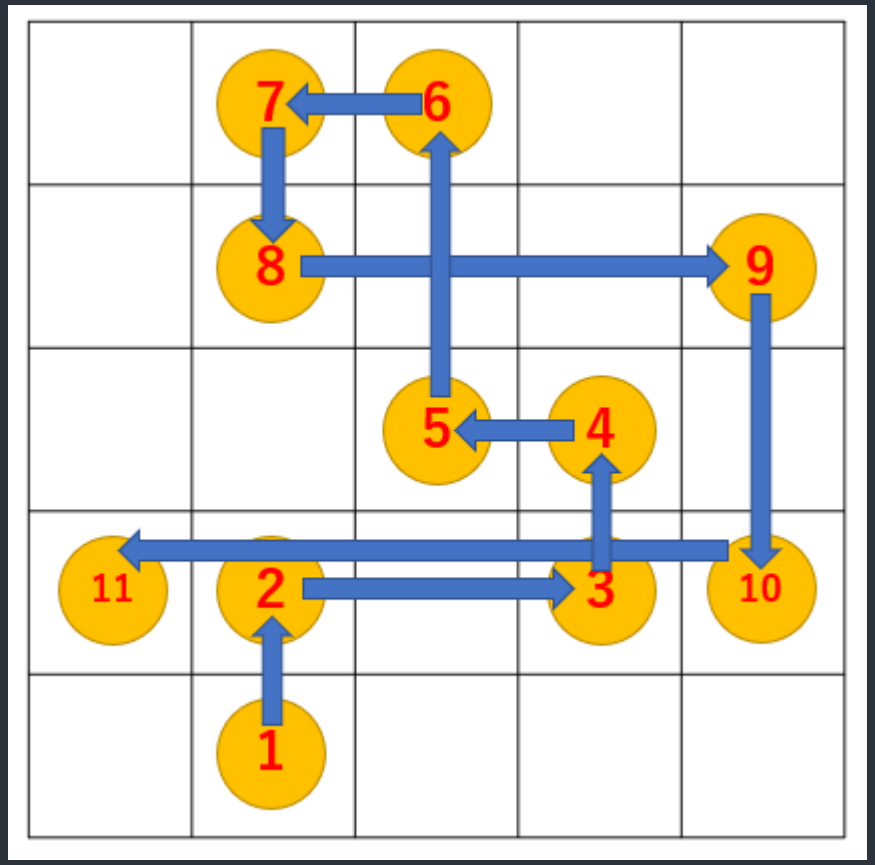
- $H \times W$ のマス目があって、各マスにはコインが 0 枚または 1 枚置かれている

1. ロボットはコインのあるいずれかのマスからスタートする
2. その後、コインを取る。コインを全部取ったらゲームクリア
3. その後、90 度右回転または左回転する
4. その後、コインがある位置までジャンプする
5. 2. にもどる

- ゲームクリアする方法はあるか？

問題概要

- 入力例 1 の場合



問題概要

- 制約は $H, W \leq 100$

- 小課題リスト

1. (100 点) : $N \leq 8$
2. (110 点) : $N \leq 16$
3. (150 点) : $H = 2$
4. (440 点) : 追加の制約はない

小課題 1 (100 点)：$N \leq 8$

- これは全探索をします
- スタートする位置・通る順番の場合の数は $N! = 1 \times 2 \times 3 \times \cdots \times N$ 通り
- 「i 個目のコインの座標が (x_i, y_i)」のように座標で管理すると良くて、そうすると判定がかなり楽に、速くできる
- 全体の計算量は $O(N! \times N)$
- 座標で管理せずに “地図上で” 全探索すると $O\big(N! \times (H + W)\big)$ かかる場合も

やったぜ、100点！！！

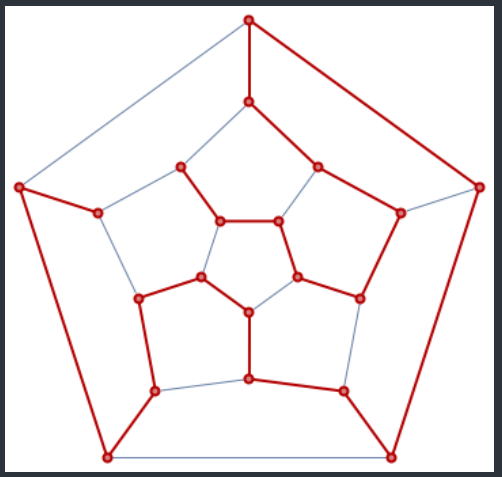
あれ・・・ほかにも小課題が・・・？

ハミルトンパス問題

- ところで、「ハミルトンパス問題」って知ってますか？

問題：「ハミルトンパス問題 (Hamiltonian Path Problem)」

N 個の都市と M 本の道路があります。i 本目の道路は、都市 a_i と b_i を双方向に結びます。そのとき、ある都市から出発してすべての都市を 1 度ずつ訪れる方法があるか判定しなさい。



- 右の図の例：都市が丸、道路が線になっていて、通る道路は赤くなってる

ハミルトンパス問題

- この問題は、動的計画法によって $O(2^N \times N^2)$ で解くことができる
- 具体的には、「（訪れた都市の集合, 最後にどの都市にいたか）の組み合わせについて、それがありうるかありえないか (true/false)」を記録
- 「訪れた都市の集合」のサイズの小さい順に求めていくと上手くいく
- （訪れた都市の集合, 最後にどの都市にいたか）の組み合わせは $O(2^N \times N)$ 通り、それぞれに対して true/false 判定するのは $O(N)$ かかるから、全体で $O(2^N \times N^2)$

小課題 2 (110 点)：$N \leq 16$

- この問題も、「すべてのコインを 1 回ずつ通らないといけない」から、ハミルトンパス問題に似てそう
- 似たような動的計画法で解きたい！！！
- 実はできて、次のような状態で「実現可能か」を true/false で管理する
  - (訪れたコインの集合, 最後にいたコインの番号, 最後に向いていた方向)
- ハミルトンパス問題とほとんど同じような DP で、計算量は $O(2^N \times N^2)$

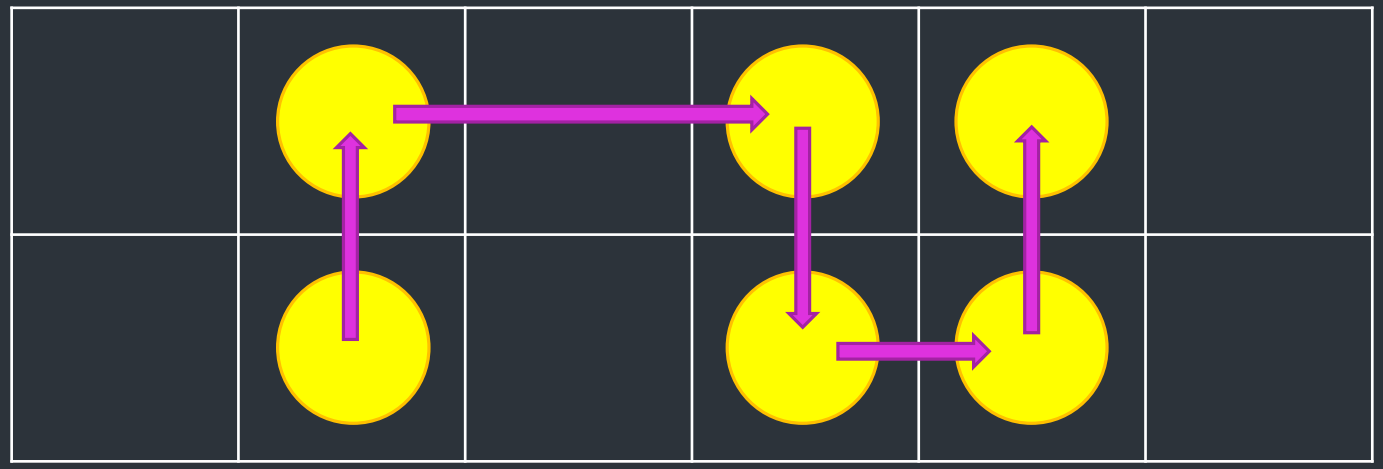
ここまでで **210 点** →

小課題 3 (150 点)：$H = 2$

- この小課題では、N の指数時間で解いたりできない
- $H = 2$ であることを利用したい！
- 考察：マス目の「上から何行目か」は周期的になりそう？
  - 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1,... みたいな感じに 2 つずつ交互に
- **これより、「1 列に 1 コイン」の列が 3 列以上あったらダメそうなことが分かる**

小課題 3 (150 点)：$H = 2$

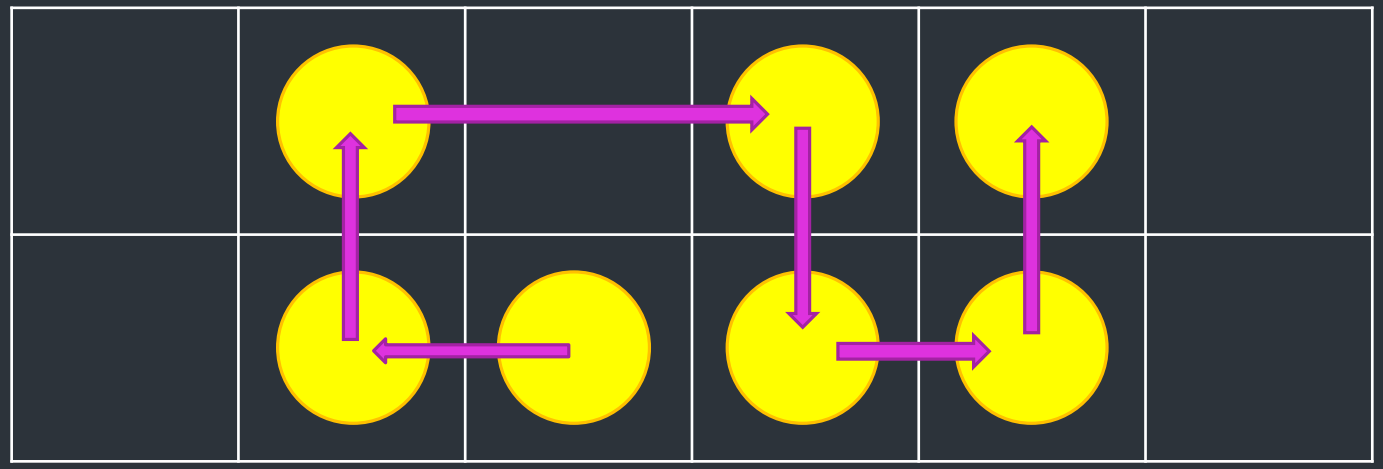
- 「1 列に 1 コイン」の列が 0 個の場合



- ジグザグに動くと、必ずすべてのコインを回収できる

小課題 3 (150 点)：$H = 2$

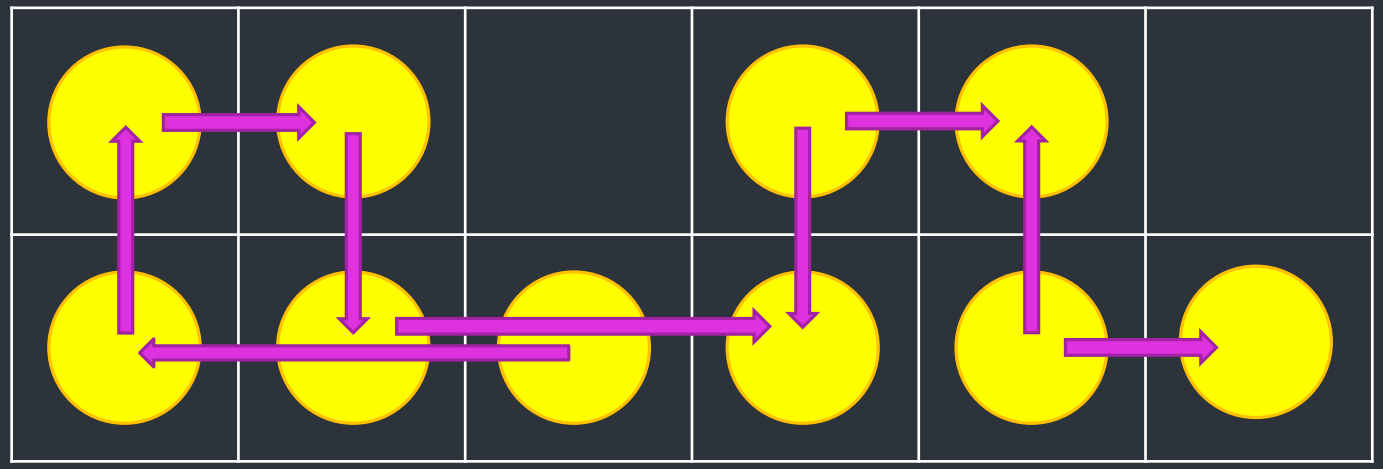
- 「1 列に 1 コイン」の列が 1 個の場合



- 「1 列に 1 コイン」の列にあるコインから始めると、「1 列に 0 コイン」の問題にすぐに帰着できるので、どんな場合でもすべてのコインを取ることが可能

小課題 3 (150 点)：$H = 2$

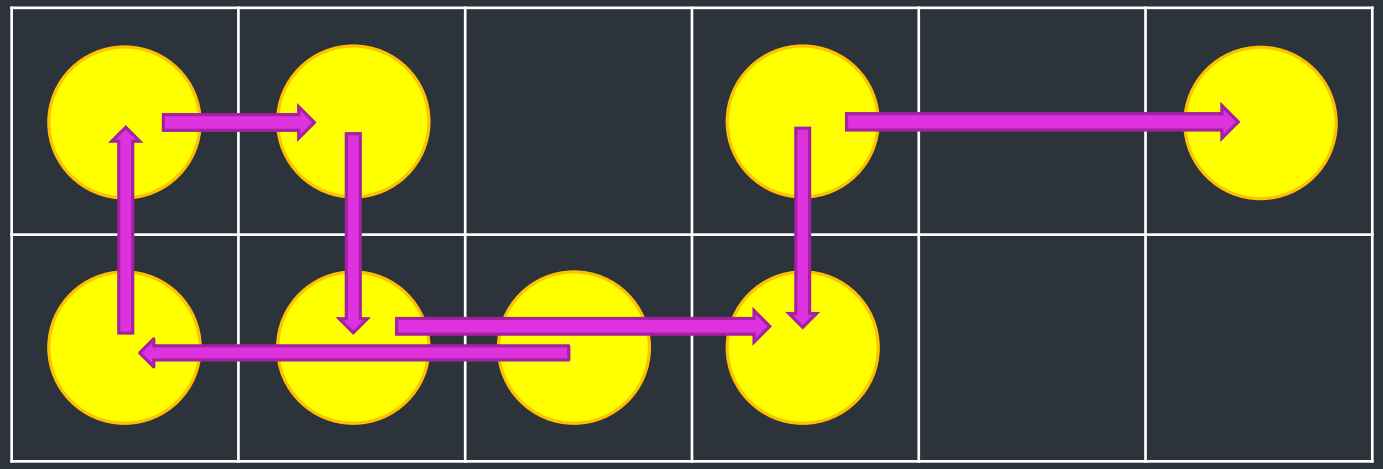
- 「1 列に 1 コイン」の列が 2 個の場合で、その 2 個のコインの行が同じ場合



- 「1 列に 2 コイン」の列の数が偶数だったら、「1 列に 1 コイン」の列のコインからスタート・ゴールをする経路で、ジグザグする感じでできる
- 「1 列に 2 コイン」の列の数が奇数だったらできない

小課題 3 (150 点)：$H = 2$

- 「1 列に 1 コイン」の列が 2 個の場合で、その 2 個のコインの行が違う場合



- 「1 列に 2 コイン」の列の数が奇数だったら、「1 列に 1 コイン」の列のコインからスタート・ゴールをする経路で、ジグザグする感じでできる
- 「1 列に 2 コイン」の列の数が偶数だったらできない

小課題 3 (150 点)：$H = 2$

- Remark: 「1 列に 1 コイン」の列が 3 列以上だとできない
- したがって、この問題の $H = 2$ の場合は $O(W)$ で解くことができた！

- 小課題 1・2 と部分点の形が全然違うので、$H = 2$ かどうかで場合分けして解法を変えると、両方の点数が獲れる！

ここまでで **360 点** →

小課題 4 (440 点)：$H \leq 100, W \leq 100$

- ハミルトンパス問題は多項式時間で解けない
- でも制約は $H \leq 100, W \leq 100$ なので、特に変わった制約はない

一体どうやったら多項式時間で解けるんだ！？

オイラーパス問題

- ところで皆さん、「オイラーパス問題」って知っていますか？

問題：「オイラーパス問題 (Eulerian Path Problem)」

N 個の都市と M 本の道路があります。i 本目の道路は、都市 a_i と b_i を双方向に結びます。そのとき、ある都市から出発してすべての道路を 1 度ずつ訪れる方法があるか判定しなさい。

- ハミルトンパス問題と違うところを黄色で記しています

- **ハミルトンパス問題は多項式時間で解けませんが、なんとオイラーパス問題は多項式時間で解けます！**

オイラーパス問題

- オイラーパス問題を多項式時間で解く方法

1. 道路が “連結” ではないならば、目標は達成できない
2. 「都市と直接つながっている道路」の数が奇数であるような都市が 3 つ以上の場合、目標は達成できない
3. 1. 2. 両方を満たさない場合、目標は達成できる

- 証明は https://mathtrain.jp/euler_graph を参照

- 1. の判定は、深さ優先探索などの方法を用いて $O(N+M)$ でき、2. の判定は簡単に $O(N+M)$ でできるから、全体の計算量は $O(N+M)$

小課題 4 (440 点)：$H \leq 100, W \leq 100$

- 実は、この問題はオイラーパス問題に帰着できる！
- 無向な二部グラフの 2 つの頂点集合を $V_1 = (A_1, A_2, A_3, \ldots, A_N), V_2 = (B_1, B_2, B_3, \ldots, B_M)$、つまり A 側に N 頂点、B 側に M 頂点のグラフを考える
- コインの位置を $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ とすると、すべての i に対して $A_{x_i} - B_{y_i}$ 間に辺を張るとして
- このグラフにオイラーパスが存在すれば「目標を達成でき」、存在しなければ「目標を達成できない」

小課題 4 (440 点)：$H \leq 100, W \leq 100$

- 90 度回転するときに、座標は $(i, j) \to (k, j) \to (k, l) \to (m, l) \to \cdots$ みたいに x 座標と y 座標が交互に変わっていく感じになる

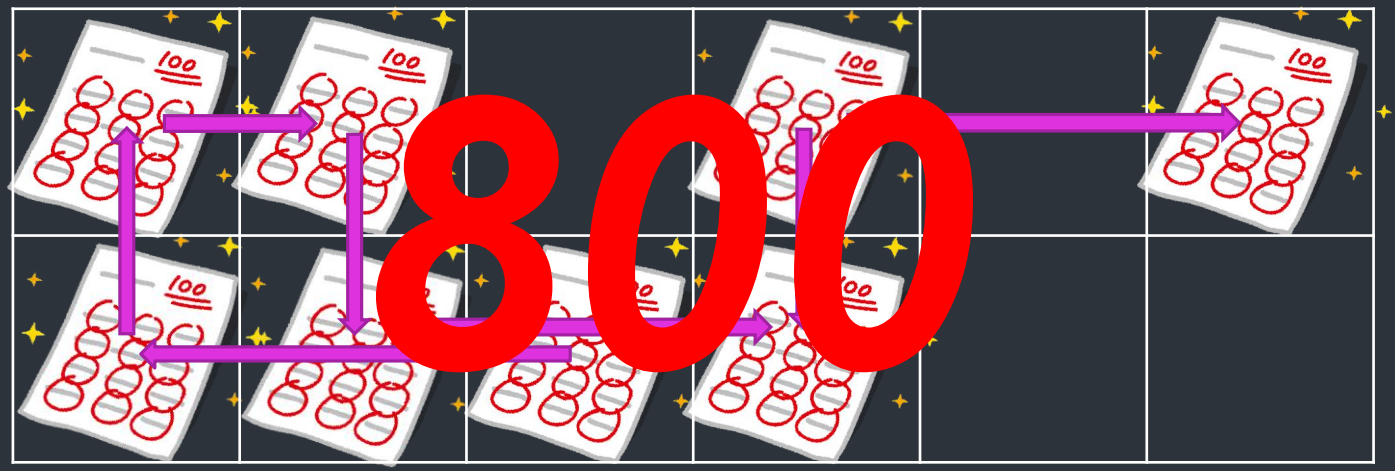
- 二部グラフのパスは、$A_{x_1} \to B_{y_1} \to A_{x_2} \to B_{y_2} \to A_{x_3} \to B_{y_3} \to \cdots$ のような感じ
- パス上の辺だけを見ると $(A_{x_1}, B_{y_1}) \to (A_{x_2}, B_{y_1}) \to (A_{x_2}, B_{y_2}) \to (A_{x_3}, B_{y_2}) \to \cdots$ になっていて、これも x と y が交互に変わっていく感じになる

- つまり、同じ問題を解いていることになる！

小課題 4 (440 点)：$H \leq 100, W \leq 100$

- グラフの頂点数が $H + W$ で、辺の数が高々 HW なので、この問題は $O(HW)$ で解くことができた！
- Yes/No 判定ではなく、やり方を求めるためには、オイラーパスを 1 つ求める必要があるが、これは Hierholzer's Algorithm などを使うと $O(HW)$ で解ける

- これで、満点を得ることができた！

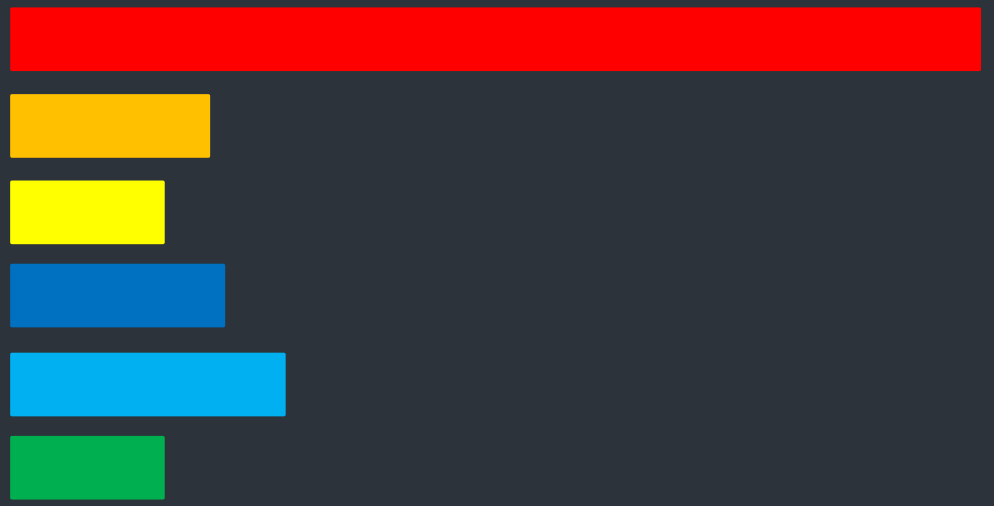


サンプルコード

- 満点解法
  - https://atcoder.jp/contests/s8pc-6/submissions/4966905
  - （コード長 849Byte、実行時間 2ms）

得点分布

- 800 点：64 人
- 360 点：13 人
- 250 点：10 人
- 210 点：14 人
- 150 点：18 人
- 100 点：10 人



F – Random Shuffles 解説

Square869120contest #5 解説

問題概要

- 問題文はかなり長いので、かなりシンプルにしてみました

- N 個のカードがあり、番号 $1, 2, \ldots, N$ のカードの色はそれぞれ $S_1, S_2, \ldots, S_N$
- D 人の参加者がいる

- $N \div D$ ラウンド行い、i ラウンド目では最初に「番号 $1, 2, \ldots, i \times D$ のカードを上から順に並べたカードの山」を作る

- その後、K 回のランダムな「隣接カードスワップ」を行う
  - $[1, N]$ のランダムな整数 i を選び、上から i 枚目のカードと $i+1$ 枚目のカードの位置を交換する。ただし $i = N$ のとき「上から N 番目と上から 1 番目のカードを交換する」。

問題概要

- それぞれのラウンドでのプレイヤー j の得点は、番号が $j, j+D, j+2D, j+3D, \ldots, j+(N-D)$ の中で、色が $j-1$ のカードの番号の合計
- 全てのラウンドの得点の合計が、最終得点になる
- D 人すべての最終得点の期待値を求めたい

問題概要

- 制約リストアップ
  - $3 \leq D \leq 10$
  - $1 \leq N \leq 3\,000\,000$ で D の倍数
  - $1 \leq K \leq 1\,000\,000\,000$
  - $S_1, S_2, \ldots, S_i, \ldots, S_N$ は $0, 1, 2, \ldots, D-1$ のいずれか

1. (50 点) : $N \leq 10, D \leq 4, K = 1$
2. (70 点) : $N \leq 10, D \leq 4, K \leq 5$
3. (190 点) : $N \leq 100, D \leq 4, K \leq 100$
4. (110 点) : $N \leq 3\,000, D \leq 4, K \leq 100$
5. (150 点) : $N \leq 3\,000, D \leq 4$
6. (170 点) : $N \leq 50\,000, D \leq 4$
7. (40 点) : $N \leq 150\,000, D \leq 6$
8. (40 点) : $N \leq 300\,000, D \leq 8$
9. (40 点) : $N \leq 500\,000, D \leq 10$
10. (40 点) : $N \leq 800\,000, D \leq 10$
11. (100 点) : $N \leq 3\,000\,000, D \leq 10$

- 小課題：やたらと多い
- https://atcoder.jp/contests/s8pc-6/tasks/s8pc_6_f を参照
- 右の画像も参照しても OK かも？

小課題 1 (50 点)：$N \leq 10, D \leq 4, K = 1$

- 全探索をします
- $K = 1$ なので、1 回しか「隣接カードスワップ」しないから、for 文を適当に書いて全探索できる

- 全部で $O(N \div D)$ ラウンドかかる
- それぞれの「隣接カードスワップ」の方法につきスコアを判定するのに $O(N)$
- 「隣接カードスワップ」の方法は N 通りある

- 全体の計算量は $O(N^3 \div D)$

小課題 1-2 (120 点)：$N \leq 10, D \leq 4, K \leq 5$

- 小課題 1 の解法とほとんど同じようにできる
- 「隣接カードスワップ」を K 回する方法は N^K 通りある
  - これは「再帰関数」などを用いて実装できる
  - Rec(残りスワップ回数, カードの並び) を再帰的に呼び出し
- 全体の計算量は $O(N^K \times N^2 \div D)$ なので間に合う

考察：全探索からの脱出

- ここからは、カードの枚数を n とする ($n = k \times D$ なので $n = N$ とは限らない)
- プレイヤー j のスコアの期待値は、「(番号 i のカード (色が $j-1$) が、位置 $j, j+D, j+2D, j+3D, \ldots, j+(n-D)$ のいずれかに来る確率) $\times i$」の合計と同じ
- あるカードが最終的にどこにいる確率がそれぞれどのくらいなのか、を求められれば、ある程度速く解けそう・・・？

考察：全探索からの脱出

- 1 回の「隣接カードスワップ」で、位置 i のカードの移動確率
  - 位置 $i-1$ に移動する確率：$1/n$
  - 位置 $i+1$ に移動する確率：$1/n$
  - 位置 i に移動する確率：$1-2/n$
  - ここでは「位置 n の後は位置 1」「位置 1 の前は位置 n」とする

- DP をすることで確率が求められる
- $dp[a][b]$ を、「隣接カードスワップ」した回数が a のときの、カードの位置が b になっている確率とする
- これは a の小さい順に $dp[a][b] = \left(1-\frac{2}{n}\right) \cdot dp[a-1][b] + \frac{1}{n} \cdot dp[a-1][b-1] + \frac{1}{n} \cdot dp[a-1][b+1]$ という式で求められる
- 計算量は $O(n \cdot K)$

小課題 1-3 (200 点)：$N \leq 100, D \leq 4, K \leq 100$

- 2 個前のスライドの通りにやると、1 回のラウンドにつき $O(n^2 \cdot K)$ の計算量で解けることが分かる
- したがって、全体の計算量 $O(N^3 \cdot K \div D)$ で解けることが分かる
- 小課題 3 の制約ではこれが間に合う！

小課題 1-4 (300 点)：$N \leq 3\,000, D \leq 4, K \leq 100$

- カード i がプレイヤー $P = S_i + 1$ に配られる条件は、位置が $P, P + D, P + 2D, \ldots, P + (n - D)$ のいずれかに来ること
- つまり、「位置を D で割った余り」で考えればよさそう！

- 先ほどの DP で「位置」を記録する部分が、「位置を D で割った余り」を記録するものに変えられるので、DP の計算は $O(D \cdot K)$ でできるようになる

- つまり、全体の計算量は $O(N^2 \cdot K)$ なので間に合うかも？
  - 定数倍高速化ができれば間に合うかもしれないけど、実行時間制限 1 秒だから厳しいかもしれない

小課題 1-4+ (300 点)：
$N \leq 3\,000, D \leq 4, K \leq 100$

- 「カードの初期位置を D で割った余り」が同じならば、DP テーブルも同じだから、カードの初期位置が $1, 2, 3, \ldots, D$ のときの DP テーブルを前計算する
- すると、各ラウンドについて前計算 $O(K \cdot D^2)$、期待値を計算するのに $O(n)$ かかるので、全体の計算量は $O\left(N \cdot (N + K \cdot D^2)\right)$
- 実行時間制限には余裕をもって間に合う

小課題 1-5 (510 点)：$N \leq 3\,000, D \leq 4, K \leq 10\text{^}9$

- 先ほどの DP を、行列累乗で高速化
  - 「行列累乗」を使って DP を高速化するテクニックは蟻本にも載っています
- 計算量は $O\left(N \cdot \left(N + D^4 \log K\right)\right)$ なので小課題 5 に通る

小課題 1-6 (640 点)：$N \leq 50\,000, D \leq 4, K \leq 10^9$

- そもそも N^2 の計算量がかかっていたらダメなので、改善したい
- そこで、「色 i で、位置を D で割った余りが j」のカードを同一視して、この番号の合計を $c_{i,j}$ とする
  - 左から順に答えを求めるか、累積和するかで $c_{i,j}$ は簡単に計算できる
- そのとき、プレイヤー $i+1$ の得点の期待値は「$c_{i,j} \times p_{j \to i}$」の合計
  - そこで、$p_{j \to i}$ は「K 回隣接カードスワップをしたときに「位置を D で割った余り」が j から i になる確率」で、これは行列累乗で計算する

小課題 1-6 (640 点)：$N \leq 50\,000, D \leq 4, K \leq 10^9$

- $p_{i \to j}$ は行列累乗 ($A_{i,j}$ を $i \to j$ の確率みたいな感じでやる) すると、全部が $O(D^3 \cdot \log K)$ の計算量で求まる

- 結局、1 ラウンドの計算量が $O(D^3 \cdot \log K)$ だから、全体で $O(N \cdot D^2 \cdot \log K)$
  - $N = 50\,000, D = 4, K = 10^9$ を代入すると 2.5×10^7 くらいなので大丈夫そう
  - $N = 150\,000, D = 6, K = 10^9$ を代入すると 1.7×10^8 くらいなのでギリギリ間に合う？
  - $N = 300\,000, D = 8, K = 10^9$ を代入すると 6.0×10^8 くらいなのでかなり厳しそう

- 定数倍高速化をすると小課題 8 にも通って 720 点が取れるかも！？
- 各ラウンドに $O(N \cdot D^4 \cdot \log K)$ かけても小課題 6 には通りそう

小課題 1-10 (800 点): $N \leq 800\,000, D \leq 10, K \leq 10^9$

- 実は $p_{i \to j}$ は $p_{0 \to (j-i)\ mod\ D}$ と同じ
- だから、$q_i = p_{0 \to i}$ として $q_0, q_1, q_2, \ldots, q_{D-1}$ を求めたい

- $K = k$ のときの $q_0, q_1, \ldots, q_{D-1}$ があったとき、$K = 2k$ のときの $q'_0, q'_1, \ldots, q'_{D-1}$ を $O(D^2)$ で求められればかなり良くて
  - 具体的には、q'_t は $q_i \cdot q_j$ $\left((i+j)\ mod\ D = t\right)$ の合計となる
  - これは普通に $O(D^2)$ で求められる

小課題 1-10 (800 点): $N \leq 800\,000, D \leq 10, K \leq 10^9$

- これで “行列累乗パート” は $O(D^2 \cdot \log K)$ まで高速化できた
- 全体の計算量は $O(N \cdot D \cdot \log K)$ なのでさっきよりかなり速い
  - $N = 300\,000, D = 8, K = 10^9$ を代入すると、7.5×10^7 くらい
  - $N = 800\,000, D = 10, K = 10^9$ を代入すると、2.5×10^8 くらい
- あまり遅くなさそうなソースコードを書けば、小課題 10 まで通って 800 点！
- 少し遅くても 760 点か 720 点は取れそう！

まとめ

- この問題は、行列累乗を使うと高速に解くことができる
  - 行列累乗の例題：AOJ 2397 Three-way Branch とか
- $O(D^2)$ や $O(D)$ の “小さな” 改善は、普段あまり気にしたことがない人も多いかもしれないが、この問題ではかなり重要な改善になってくる
- 定数倍の部分が遅くて 760 点・720 点とかしか取れなかった人も、定数倍高速化の練習だと思って下さい (もちろん、python などの言語で通すことは想定されていません)

得点分布

800	800	800	800	800	800	800	760	760	760
760	720	720	680	640	640	510	510	300	300
300	300	300	300	300	300	300	300	300	200
200	200	200	200	120	120	120	120	120	120
120	120	120	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

スライドショーの最後です。クリックすると終了します。

???

$$N \leq 3\,000\,000$$

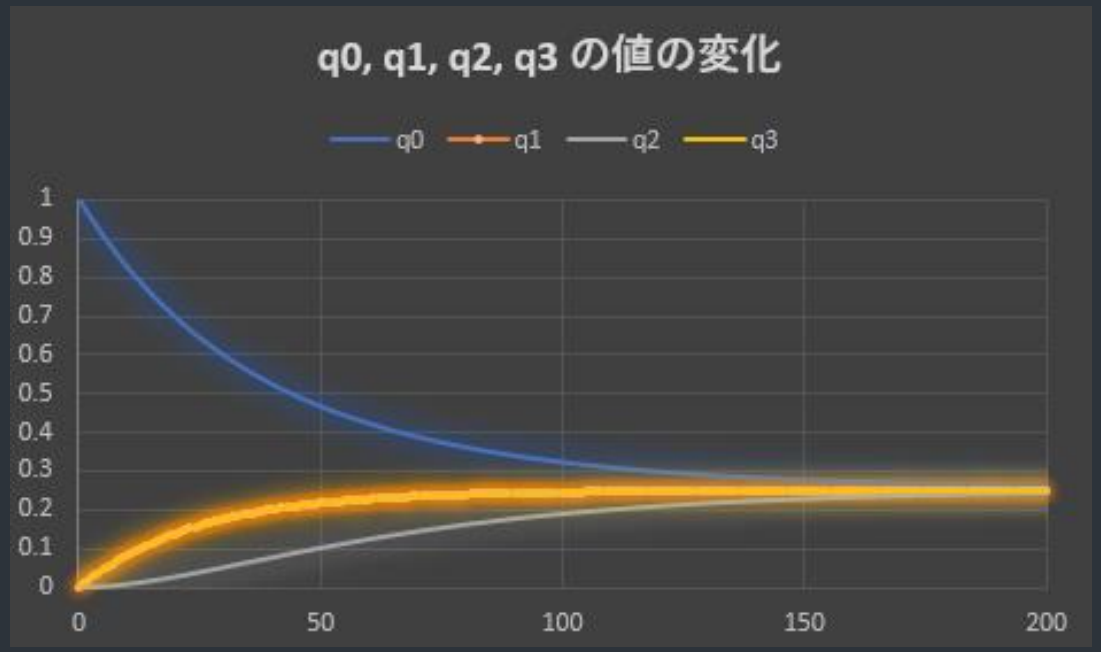
$$D \leq 10$$

こんなの定数倍高速化では通されない

- 制限時間 1 秒、$N \leq 3000000$、$D \leq 10$
  - 計算量 $O(ND \log K)$ の場合、なんと 9.0×10^8 くらい
  - 浮動小数点を扱うのに、こんなの解ける訳もない
- 誤差 10^{-5} 以下 → これを使えないか？
- 時間がかかっている操作は何？
  - 間違いなく “行列累乗パート”
  - 行列累乗パートを無理やり減らすことは出来ないのか？

誤差を解析する

- 前述の $q_0, q_1, q_2, \ldots$ の値をグラフに可視化してみる
  - q_i に関しては 8 ページ前参照
- 以下は、$D = 4$, ある場所が swap される確率 $Q = 0.01$ とするときのグラフ



誤差を解析する／近似で解を求める

- 実は偶数回目のラウンドでしか $q_0, q_1, q_2 \ldots$ の値を求める必要性がないのでは？
  - ここでは $r_{i,j}$ を i ラウンド目（カードの枚数が $i \cdot D$ 枚の時）の q_j の値とする
- この場合、例えば $r_{3,j} = \frac{r_{2,j}+r_{4,j}}{2}, r_{5,j} = \frac{r_{4,j}+r_{6,j}}{2}, \ldots$ となる
  - つまり、周りの平均を取る、ということ
  - そうすると割と近似ができそうなのではないか？
- この場合 i が小さい場合の誤差が大きそう
  - それに対して、i が大きい場合の誤差が小さそう（厳密に証明はできますが、解説の長さの都合上証明は省きます）

誤差を解析する／近似で解を求める

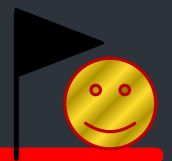
- ここで、実際に行列累乗パートで $r_{i,j}$ を求めるような i の集合を $S = \{S_1, S_2, \ldots, S_A\}$ とする。

- $S_{i+1} = max(S_i + 1, S_i \times 1.001)$ みたいなことをすると、誤差が大きそうな最初の方は全て “行列累乗パート” を行い、誤差が小さそうな後半の方は $S_{i+1} - S_i$ の値が大きく、大幅に “行列累乗パート” を行う回数を減らすことができる

- $S_i < t < S_{i+1}$ の部分の $r_{t,j}$ の値はどのようにして埋めるか？
  - $q_{t,j} = (S_{i+1} - t)q_{S_i,j} + (t - S_i)q_{S_{i+1},j}$
  - つまり、グラフに書き表すと、該当部分が直線であるとみなすということ。

満点解法

- 許容誤差は 10^{-5} 以内であり、 $S_{i+1} = max(S_i + 1, S_i \times 1.001)$ の式で “行列累乗パート” でそのまま正確な値を求めるようなラウンドを決めるとすると高々 10^{-6} 未満の誤差しか出ずに近似できることは証明できる。
  - 上の場合の S のサイズは、ラウンド数が 3000000 であっても高々 9587 である。
- そのため、計算量は以下のようになる。
  - “行列累乗パート” に、合計で $9587 \cdot D^2 \cdot \log K$ 回の計算
  - “近似で $r_{i,j}$ を求めるパート“ に、合計で ND 回の計算
  - $N = 3000000, D = 10, K = 10^9$ のとき高々 5.9×10^7 回
    - 200ms / 1000ms しか使わずに満点！



ソースコード

- 120 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4819268
- 300 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4968528
- 680 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4819275
- 800 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4819262
- 満点解法： https://atcoder.jp/contests/s8pc-6/submissions/4967221

- なお、満点解法の実行時間は 195ms、コード長は 1751Byte です。
  - 定数倍高速化の問題と思いきや実行時間の 20% も使わずに正解しています。

得点分布

- 正の得点を取った人は **27 人**で、以下の表のようになります

1000	1000	800	800	760	720	720	720	720
720	720	680	680	680	680	640	640	640
640	510	300	200	200	200	120	50	50

- 1000 点を取った人
- **yutaka1999** さん (87 分 6 秒)
- **yosupo** さん (147 分 28 秒)

G – Medals 解説

Square869120contest #5 解説

問題概要

- N 人の人がコンテストに参加する
- 1 日目の点数はそれぞれ $A_1, A_2, A_3, \ldots, A_N$ で**同点なし**ということが分かっている
- 2 日目の点数は $1, 2, 3, \ldots, N$ 点の**並び替え**であることが分かっている
- 金・銀・銅メダルは、合計点で 1 位, 2 位, 3 位だった人に与えられる
  - 同点が出た場合はその中で抽選で選ばれる
- $P = 1$ のとき、1 日目の点数は 金 > 銀 > 銅であることが分かっている
- $P = 2$ のとき、1 日目の点数は 金 > 銅 > 銀であることが分かっている

- (金メダリスト, 銀メダリスト, 銅メダリスト) としてありうる人の組み合わせは何通りあるか？

問題概要

- 制約はどのくらいかな？
  - $1 \leq N \leq 1\,000\,000$
  - $1 \leq A_i \leq 1\,000\,000\,000$
- 小課題について
  - 24 個あって、それぞれ 50 点ずつ
  - こんなのは過去の AtCoder に出題されたわけがない
  - 12 個ずつの違いはなんと **N の制約のみ**

1. (600 点) : $P = 1$
2. (600 点) : $P = 2$

1. (50 点) : $5 \leq N \leq 7$
2. (50 点) : $5 \leq N \leq 15$
3. (50 点) : $5 \leq N \leq 30$
4. (50 点) : $5 \leq N \leq 70$
5. (50 点) : $5 \leq N \leq 200$
6. (50 点) : $5 \leq N \leq 600$
7. (50 点) : $5 \leq N \leq 2\,000$
8. (50 点) : $5 \leq N \leq 8\,000$
9. (50 点) : $5 \leq N \leq 50\,000$
10. (50 点) : $5 \leq N \leq 200\,000$
11. (50 点) : $5 \leq N \leq 500\,000$
12. (50 点) : $5 \leq N \leq 1\,000\,000$

戦略 (?)

- コンテストの時間は限られている
- $P = 1$ の場合は $P = 2$ の場合よりも簡単
  - これで 600 点を狙いに行くのもあり
- 実は $O(N^4)$ くらいでも $P = 1, 2$ 両方に対応させれば 400 点くらい取れたりする
  - これである程度の点数を狙いに行くのもあり
- もちろん、満点を思いついて実装するのもあり
  - 「$O(N^7)$ から $O(N)$ に改善できるわけないだろ」、と思って考えないのはよくない
  - 「満点はどうせ実装激重w」とか思って考えないのもよくない **(実はかなり実装軽い)**

おことわり

- この問題は普通ではない小課題の付けられ方なので、解説では次のように説明

- 「小課題 A1, A5, A7 など」：$P = 1$ の小課題 1, 5, 7 など
- 「小課題 B2, B8, B11 など」：$P = 2$ の小課題 2, 8, 11 など

小課題 A1, B1 : $N \leq 7$

- 2 日目の得点は $N!$ 通り
- これを全探索して、メダリストの組み合わせとしてありうるものを探し出す

- 計算量は $O(N!) \times O(N^3) = O(N! \times N^3)$ なので間に合う

100 points (Rel. 8.33%, No Medal)

小課題 A1-2, B1-2：$N \leq 15$

- (金メダリスト, 銀メダリスト, 銅メダリスト) の組み合わせは N^3 通りくらい
  - そのすべてに対して「可能かどうか」判定したい
- 金メダリスト, 銀メダリスト, 銅メダリストの 2 日目の得点を全探索して判定
  - 残りは「1 日目の点数が高い方が 2 日目の点数が低い」のが最適
  - 組み合わせはだいたい N^3 通り
- よって、この問題は $O(N^7)$ で解けた

200 points (Rel. 16.66%, No Medal)

小課題 A1-4：$N \leq 70$
$P = 1$ の場合どうなる？

- $P = 1$ の場合、金, 銀, 銅メダリストが点数 $N-2, N-1, N$ を取るのが最適
- 証明：金, 銀, 銅メダリスト g, s, b に対して $A_g > A_s > A_b$ であることに注意する。$(A_g, A_s, A_b) = (N-2, N-1, N)$ のとき、「銅メダルボーダー」は最大化できる、つまり $A_b + N$ より大きくできない。また、残りの選手の点数は $1, 2, 3, \ldots, N-3$ だから、これも他の人の点数を下げる側からしたら最適である。
- 金, 銀, 銅メダリストの “最適な得点” が分かったので、1 スライド前の解法の「金, 銀, 銅メダリストの点数を全探索する」部分がなくなって $O(N^4)$ で解ける

300 points (Rel. 25.00%, No Medal)

小課題 A1-A6：$N \leq 600$

- 「銅メダルボーダー」は $A_b + N$ 点。さて、「次点 (4 位)」は？
- 実は、次点は「メダリスト以外で 1 日目で最も高い点数を取った人」。$A_1, A_2, A_3, \ldots, A_N$ がすべて異なっていて、2 人の 1 日目の得点は 1 点以上
- メダリスト以外の 1 日目の点数を $B_1 > B_2 > \cdots > B_{N-3}$ とすると、$B_1 + 1 \geq B_2 + 2 \geq B_3 + 3 \geq \cdots \geq B_{N-3} + (N-3)$ となる
- つまり、「次点ボーダー」は $B_1 + 1$ 点だから、$A_b + N \geq B_1 + 1$ のときだけ、このメダリストの組み合わせがありうる

小課題 A1-A6：$N \leq 600$

- 降順に並べ替えて $A_1 > A_2 > \cdots > A_N$ にしたとき、B_1 になりうるのは A_1, A_2, A_3, A_4 のいずれかなので、判定が定数時間でできる
- 結果、この問題は $O(N^3)$ で解くことができた！

400 points (Rel. 33.33%, No Medal)

小課題 A1-A12：$N \leq 1\,000\,000$

- B_1 としてありうるものは 4 通りあるので、これを全探索します
- $B_1 = A_i$ $(1 \leq i \leq 4)$ だったとき
  - 銅メダルボーダー $A_b + N$ は $A_i + 1$ 以上でなければならない
  - $g < s < b$ で、かつ g, s, b の中に $1, 2, \ldots, i-1$ がすべて含まれなければならない。それ以外は $i + 1$ 以上
- その条件下で g, s, b を選んだとして、どうやっても $A_g + (N-2) \geq A_s + (N-1) \geq A_b + N$ になるので、結局 b を固定して考えてよい

小課題 A1-A12：$N \leq 1\,000\,000$

- i が決まっているとして、b を固定したときに、先ほどの条件が成り立っていなければならない
- $i = 4$ のとき：$b = 3$ の 1 通り、それ以外ゼロ
- $i = 3$ のとき：$g = 1, s = 2$ なので、$b \geq 4$ のときだけ 1 通り
- $i = 2$ のとき：$g = 1$ で $3 \leq s \leq b - 1$ なので、$b \geq 4$ のときだけ $C(b - 3, 1)$ 通り
- $i = 1$ のとき：$2 \leq g, s \leq b - 1$ なので、$b \geq 4$ のときだけ $C(b - 2, 2)$ 通り
- ただし、$A_b + N \geq A_i + 1$ とする

- 結果的に、b を固定したときの組み合わせが定数時間で求められるので、全体の計算量は A のソートに一番時間がかかって $O(N \log N)$

小課題 A1-A12：$N \leq 1\,000\,000$

- これで、$P = 1$ の場合が解けた！
- ここまでで 700 点。$P = 1$ だけの解法を書いても 600 点が取れる

700 points (Rel. 58.33%, Bronze Medal)

小課題 B1-B3：$N \leq 70$

- $P = 1$ を解く方針と同じような感じで $P = 2$ も解きたい
- 1 日目の点数が 金 > 銀 > 銅 の場合、2 日目の点数がそれぞれ $N-2, N-1, N$ 点、とできるが、金 > 銅 > 銀の場合そうにはいかない
- $A_b > A_s$ なのに 2 日目で銀が銅以上の点数になっているので、2 日目の銀・銅の点数の差は $A_b - A_s$ 点以上
- 金メダルが 2 日目で N 点を取るよりも、銀メダリストが 2 日目で N 点を取った方が良さそう

小課題 B1-B4：$N \leq 70$

- つまり、$A_b - A_s \geq 2$ のとき 2 日目の点数は金, 銀, 銅それぞれ $N-1, N-(A_b - A_s), N$ 点とするのが最適
- $A_b - A_s = 1$ のとき 2 日目の点数は金, 銀, 銅それぞれ $N-2, N-1, N$ 点とするのが最適

- この証明は、$P = 1$ のときと同様の方針でできる

- 金・銀・銅を取った人を決め打ちすると、メダリストの 2 日目の点数が “定まる” ので、判定に $O(N)$ 時間しかかけなくてもよい
- 全体の計算量は $O(N^4)$ なので B1 – B4 に通る

800 points (Rel. 66.67%, Bronze Medal)

小課題 B1-B6：$N \leq 600$

- 今回も「次点」について考えてみよう
- 「次点」は、必ず A_1, A_2, A_3, A_4, A_5 の中に含まれるので、これを考えれば金・銀・銅メダリストが決まっていた時に定数時間で判定できる
- 全体の計算量は $O(N^3)$ なので、実装によっては小課題 B6 まで通る

900 points (Rel. 75.00%, Silver Medal)

小課題 B1-B8：$N \leq 8\,000$

- 先ほど「次点」は A_1, A_2, A_3, A_4, A_5 に必ず含まれるといいましたが、「メダリスト以外の 1 日目の点数」を $B_1 > B_2 > \cdots > B_N$ とすると、ほとんどのケースで B_1 点を取った人が「次点」になる

- ほとんどのケース・・・？
- 実は、「銅メダリストの 2 日目の点数が 2 点 (つまり、$A_b - A_s = N - 2$ の場合で、$B_1 - B_2 = 1$ の場合」のみ、B_2 が次点になる場合がある
- この場合、2 日間の合計点数は $B_1 + 1, B_2 + 3, \ldots$ となるから、B_1 と B_2 の差が 1 のときだけ $B_2 + 3$ の方が大きくなる
- しかし、その場合「銅ボーダー」がちょうど $B_2 + 3$ となることはありえないから、「$B_1 + 1$ が次点」とみなしても良いことになる！

小課題 B1-B8：$N \leq 8\,000$

- そこで、「銅メダルの 1 日目の順位」と「銀メダルの 1 日目の順位」を全探索する。状態数は $O(N^2)$
- 金メダルは「銅メダルの位置」より 1 日目の得点が上で、かつ B_1 のポジションによる条件にしか関係しないので、判定は定数時間でできそう

- (銀メダリスト, 銅メダリスト) の組み合わせが正しいのは、$A_b - A_s \leq N - 1$ であり、かつ $B_1 + c \leq A_s + N$ であるとき
- c はほとんどの場合で 1 だが、$A_b - A_s = 1$ の場合だけ $c = 2$ になることに注意

- 結果、この問題は計算量 $O(N^2)$ で解くことができた

1,000 points (Rel. 83.33%, Silver Medal)

小課題 B1-B12：$N \leq 1\,000\,000$

- あとはウィニングランです
- さっきの解法では (銀メダリスト, 銅メダリスト) の組を全探索していたが、今度は「銅メダリスト」を全探索することを考える

- 銅メダリストを固定したときに、銀メダリストとしてありうるのは (B_1 の位置にもよるけど) 大体の場合で $A_b - A_s \leq N - 1$ のとき

- そのような s を二分探索または尺取り法で求めることができる
  - $A_b - A_s = N - 1$ の場合「次点ボーダー」が 1 変化することに注意する。場合分けで対処

小課題 B1-B12：$N \leq 1\,000\,000$

- また、正しい g の値は s の選び方に依存しないので、単純に掛け算しても OK
- よって、b が固定されているときの組み合わせの数え上げが平均で定数時間 (二分探索を使った場合 $O(\log N)$) で分かるようになった
- ソートの計算量が一番多く、全体の計算量は $O(N \log N)$
- これでこの問題で満点を取ることができました！！！

1,200 points (Rel. 100.00%, Gold Medal)

サンプルコード

- 200 点解法（$P = 1$ のみ）
  - https://atcoder.jp/contests/s8pc-6/submissions/4967364
- 400 点解法（$P = 1, 2$ 両方に対応）
  - https://atcoder.jp/contests/s8pc-6/submissions/4967409
- 満点解法
  - https://atcoder.jp/contests/s8pc-6/submissions/4900638

得点分布

- **Gold (1200)** : 13 人
- **Silver (900 – 1199)** : 1 人
- **Bronze (600 – 999)** : 10 人
- **Honorable Mention (300 – 599)** : 3 人
- **Scorer (1 – 299)** : 6 人

- First AC: **Kmcode** (46 分 18 秒)

H – Percepts of AtCoder 2

Square869120contest #5 解説

問題概要

- できるだけ長さが短い、以下のような数列を 1 つ構成してください。
  - ちょうど K 種類の単調狭義増加部分列を含む。
    - 単調狭義増加部分列に関しては、問題文を参照すること。
  - ただし、違う場所から選んで取ってきた部分列であっても、部分列が同じ場合同じ種類として見做す。
  - 例えば、[1, 3, 1, 2] には 6 種類の単調狭義増加部分列を含む
    - [], [1], [2], [3], [1, 2], [1, 3]

- ただし、長さ 128 以下でなければ不正解となります。

問題概要

- 小課題 1 (30 点)：$K \leq 100$
- 小課題 2 (70 点)：$K \leq 1\,500$
- 小課題 3 (1400 点)：$K \leq 5\,000\,000\,000\,000\,000\,000\ (= 5 * 10^{18})$
  - 数列の長さの最大値を L とするとき、
  - $120 \leq L \leq 128$：125 点
  - $100 \leq L \leq 119$：$1400 * 5^{\frac{L-99}{20}}$ 点
    - 特に、$L = 114$ だと 419 点、$L = 109$ だと 626 点、$L = 104$ だと 936 点
  - $1 \leq L \leq 99$：1400 点

小課題 1 (30 点)

- $K \leq 100$
- 取り敢えず K 文字くらい使うことを考える

- 少し実験してみる
  - $8, 7, 6, 5, 4, 3, 2, 1$ の単調狭義増加部分列は 9 種類
  - $13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1$ の単調狭義増加部分列は 14 種類
  - 実は、$P, P-1, P-2, P-3, \ldots, 4, 3, 2, 1$ の単調狭義増加部分列は $P+1$ 種類なのでは？
    - これは増加部分列の長さが高々 1 であることかた簡単に証明できる

- よって、$P = K - 1$ のときの数列を出力すれば OK

30 点

考察 1

- $K \leq 1500$
- 取り敢えず、長さ L で L^2 くらいの数を表したい

- Q. 長さ a の数列と長さ b の数列を組み合わせて長さ ab の数列を作ることができるのか？

- ヒント： $[106, 105, 104, 103, 102, 101, 204, 203, 202, 201]$ だと $7 * 5 = 35$ 通り

考察 1

- A. できます
- 具体的には、長さ a の数列と長さ b の数列を組合せることにより $(a+1)(b+1)$ 通りを表すことができます
  - $[a, a-1, a-2, a-3, \ldots, 2, 1, a+b, a+b-1, \ldots, a+2, a+1]$ という構成方法
- この構成方法を用いると、あり得る最長の増加部分列の長さが 2 にしかならないので、$(a+1)(b+1)$ 通り
  - 前半部分・後半部分両方選ぶ： ab 通り
  - 前半部分・後半部分片方選ぶ： $a+b$ 通り
  - 1 個も選ばない： 1 通り

考察 1

- しかしそれだけだと K が素数の時に解けないので、$K = ab + c$ の形で表せるかどうかを考えてみる
- 考察：$K = ab$ の数列の後ろに小課題 1 のような長さ c の減少数列を付け加えればよい！
  - ただし、ちょうど c 通り増やすためには、以前の数列より小さい値が良い
- 例えば、$a = 4, b = 4, c = 7$ [通り数は $(4 + 1) * (4 + 1) + 7 = 32$ 通り] の場合
  - $[11, 10, 9, 8, 15, 14, 13, 12, 7, 6, 5, 4, 3, 2, 1]$

小課題 2 (累積 100 点)

- $N \leq 1500$ のとき、(a, b, c) を全探索した時に $a + b + c \leq 128$ を満たす組が実は存在する！

- [証明]　$(a + 1, b + 1) = (1, 1), (1, 2), (2, 2), (2, 3), (3, 3), (3, 4), \ldots, (38, 38), (38, 39)$ の時を考えればよい。その時、どのような (a, b) についても $a + b \leq 76$ を満たし、かつその中で $38 * 39 - 38 * 38 = 38$ より $c < 38$ を満たすようなものが存在する。よって、$a + b + c < 76 + 38 = 114$ が確実に満たされる。

- これで 100 点が取れました。
  - しかし、まだ得点は 7% です…

100 点

考察 2

- $K \leq 5 * 10^{18}, L \leq 128$
- 小課題 2 では L^2 くらいの数を表せたが、今度は 2^L くらいの数を表せなければならない

- [重要な考察]
- $A = \{1, 3, 5, 7\}$ だと、何種類の増加列があるか？ → 16 通り
- $A = \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19\}$ だと、何種類の増加列があるか？ → 1024 通り

- つまり、$\{1, 3, 5, 7, \ldots, 2N - 1\}$ という数列の場合、2^N 通り

考察 2

- しかし、それだけだと 2^N っぽい数しか表せない
- 前ページの数列から、後ろに 1 個整数を追加することを考える

- 例えば… $N = 7$ の場合？

考察 2

- $\{1, 3, 5, 7, 9, 11, 13\} \rightarrow 128$ 通り
- $\{1, 3, 5, 7, 9, 11, 13, 12\} \rightarrow 128 + 64 = 192$ 通り
- $\{1, 3, 5, 7, 9, 11, 13, 10\} \rightarrow 128 + 32 = 160$ 通り
- $\{1, 3, 5, 7, 9, 11, 13, 8\} \rightarrow 128 + 16 = 144$ 通り
- $\{1, 3, 5, 7, 9, 11, 13, 6\} \rightarrow 128 + 8 = 136$ 通り
- $\{1, 3, 5, 7, 9, 11, 13, 4\} \rightarrow 128 + 4 = 132$ 通り
- $\{1, 3, 5, 7, 9, 11, 13, 2\} \rightarrow 128 + 2 = 130$ 通り
- $\{1, 3, 5, 7, 9, 11, 13, 0\} \rightarrow 128 + 1 = 129$ 通り
- → 後ろに $2a$ を追加すれば増加部分列の種類の通り数が 2^a 通り増えそう？

考察 2

- 取り敢えず末尾に適当に挿入してみる
- $\{1, 3, 5, 7, 9, 11, 13\} \rightarrow 128 = 128$ 通り
- $\{1, 3, 5, 7, 9, 11, 13, 12\} \rightarrow 128 + 64 = 192$ 通り
- $\{1, 3, 5, 7, 9, 11, 13, 12, 8\} \rightarrow 128 + 64 + 16 = 208$ 通り
- $\{1, 3, 5, 7, 9, 11, 13, 12, 8, 4\} \rightarrow 128 + 64 + 16 + 4 = 212$ 通り
- $\{1, 3, 5, 7, 9, 11, 13, 12, 8, 4, 0\} \rightarrow 128 + 64 + 16 + 4 + 1 = 213$ 通り

- 確かに何回増やしても $2a$ を追加すると 2^a 増えるのには変わりがない！
  - [理由] 黄色の部分を 2 個以上含む列が存在しない

考察 2

- 何故 $\{1, 3, 5, 7, 9, 11, 13, 12, 8, 4, 0\}$ は 213 通りか？
  - 前述の通り、黄色の部分は高々 1 個しか含まない
  - 12 を含む部分増加列：前半部分に 11 までしか含まないので $2^6 = 64$ 通り
  - 8 を含む部分増加列：前半部分に 7 までしか含まないので $2^4 = 16$ 通り
  - 4 を含む部分増加列：前半部分に 3 までしか含まないので $2^2 = 4$ 通り
  - 0 を含む部分増加列：前半部分に何も含まないので $2^0 = 1$ 通り
  - 黄色を含まない部分増加列：前半部分に 13 までしか含まないので $2^7 = 128$ 通り
  - よって、$128 + 64 + 16 + 4 + 1 = 213$ 通り

225 点解法

- あとはもう簡単
- K を 2 のべき乗の和で表したときに $2^{a_1} + 2^{a_2} + \cdots + 2^{a_p}$ $(a_1 > a_2 > \cdots > a_p)$ となる時、どのようにすればよいのか？
  - 前半部分：$\{1, 3, 5, 7, \ldots, 2a_1 - 1\}$ → この時点で数列の種類数は 2^{a_1} 通り
  - 後半部分：$\{2a_2, 2a_3, 2a_4, \ldots, 2a_p\}$ → さらに $2^{a_2} + \cdots + 2^{a_p}$ 通りの種類が増える
- $2^{63} > 5 * 10^{18}$ より、長さ $63 * 2 - 2 = 124$ 以下で作れることが保証される
  - 30 + 70 + 125 = 225 点

225 点

考察 3

- L を少しだけでも良いので減らしたい。どうすれば減らせるか？

- [一般的なテク]
- 2 進数の代わりに 3 進数を使うと、若干 L の値が減るかもしれない

- [類題] パ研合宿コンペティション 2 日目 F：同一経路 (E869120 作問)
- https://atcoder.jp/contests/pakencamp-2018-day2/tasks/pakencamp_2018_day2_f

考察 3

- 取り敢えず 3 進数を無理やりでも作りたい
- $\{3, 1, 7, 5, 11, 9, 15, 13, 19, 17 \ldots\}$ という数列における増加部分列の種類の通り数は $3^{\frac{L}{2}}$ 通りなので行けそうではある

- 例えば
  - $\{3, 1, 7, 5, 11, 9\}$ → 27 種類
  - $\{3, 1, 7, 5, 11, 9, 15, 13, 19, 17\}$ → 243 種類

- これに関しても、数列の後ろに 1 個数を追加することを考える

考察 3

- $\{3, 1, 7, 5, 11, 9\} \rightarrow 27$ 通り
- $\{3, 1, 7, 5, 11, 9, 12\} \rightarrow 27 + 27 = 54$ 通り
- $\{3, 1, 7, 5, 11, 9, 10\} \rightarrow 27 + 18 = 45$ 通り
- $\{3, 1, 7, 5, 11, 9, 8\} \rightarrow 27 + 9 = 36$ 通り
- $\{3, 1, 7, 5, 11, 9, 6\} \rightarrow 27 + 6 = 33$ 通り
- $\{3, 1, 7, 5, 11, 9, 4\} \rightarrow 27 + 3 = 30$ 通り
- $\{3, 1, 7, 5, 11, 9, 2\} \rightarrow 27 + 2 = 29$ 通り
- $\{3, 1, 7, 5, 11, 9, 0\} \rightarrow 27 + 1 = 28$ 通り
- 同じように、$1 * 3^0, 2 * 3^0, 1 * 3^1, 2 * 3^1, 1 * 3^2, 2 * 3^2, \ldots$ が全てできている！

456 点解法

- 結局は黄色の部分を単調減少にすれば 3 進数にも応用できる
  - 例えば $19 = 2 * 3^2 + 3^0$ の場合、$\{2, 1, 4, 3, 5, 0\}$ (9 + 9 + 1 = 19 種類)
- L の値は結局どれくらいなのか？
  - 必要な長さは、$3^{40} > 10^{18}$ より $3 * 40 - 3 = 117$ 文字 → 429 点
  - 実装を工夫すると 116 文字 → 456 点
- これで 456 点が取れた！

456 点

486 点解法

- 2 進数の解法、3 進数の解法を両方実装して、どちらか短い方を選ぶと長さが 1 だけ短くなる
  - 3 進数で表したときに 2222222… となるようなコーナーケースに対応できるから
- $L = 115$ で、30 + 70 + 386 = 486 点
  - これで全体の 4 割！

486 点

考察 4

- 最初に言っておきますと、この考察の段階が一番難しいです。
- 取り敢えず 3 進数に増やしたが、そのままの方法で 4 進数以上で表すとどうなるのでしょうか？

進数	計算	結果
2	$2^{62} \leq K < 2^{63}$	長さ $62 * 2 = 124$
3	$3^{39} \leq K < 3^{40}$	長さ $39 * 3 = 117$
4	$4^{31} \leq K < 4^{32}$	長さ $31 * 4 = 124$
5	$5^{26} \leq K < 5^{27}$	長さ $26 * 5 = 130$

- どうやら 3 進数が最適っぽいです。

考察 4

- しかし、ここで諦めたら負けです
- 5 進数を基にして数列を作ることを考えましょう
  - 元々は、前半部分には 1 個位を増やすごとに 長さ 4 を消費していた
  - いまは、前半部分には 1 個位を増やすごとに 長さ 3 を消費することを考える
- 元々は、$\{4, 3, 2, 1, 8, 7, 6, 5, 12, 11, 10, 9, \dots\}$ というような前半部分となっているので、長さ 4 を使っている
- しかし、長さ 3 で増加部分列の種類数が 5 となるような数列も存在する
  - $\{2, 3, 1\}$ と $\{3, 1, 2\}$

考察 4

- 長さ 3 で 5 種類の増加部分列を持つような、$\{2, 3, 1\}$ の後ろに数を挿入することを考える（今回は同じ数を防ぐために $\{3, 5, 1\}$ にして、後ろに偶数を挿入する）

- $\{3, 5, 1\} \rightarrow 5$ 種類の増加部分列
  - $\{3, 5, 1, 0\} \rightarrow 5 + 1 = 6$ 種類の増加部分列
  - $\{3, 5, 1, 2\} \rightarrow 5 + 2 = 7$ 通りの増加部分列
  - $\{3, 5, 1, 4\} \rightarrow 5 + 3 = 8$ 通りの増加部分列
  - $\{3, 5, 1, 6\} \rightarrow 5 + 5 = 10$ 通りの増加部分列

- つまり、$5^n, 2 * 5^n, 3 * 5^n$ に関する加算はできそう。
  - $4 * 5^n$ に関する加算はどうする？

考察 4

- 長さ 3 で 5 種類の増加部分列を持つような、$\{3,1,2\}$ の後ろに数を挿入することを考える（今回は同じ数を防ぐために $\{5,1,3\}$ にして、後ろに偶数を挿入する）

- $\{5,1,3\} \rightarrow 5$ 種類の増加部分列
  - $\{5,1,3,0\} \rightarrow 5+1=6$ 種類の増加部分列
  - $\{5,1,3,2\} \rightarrow 5+2=7$ 通りの増加部分列
  - $\{5,1,3,4\} \rightarrow 5+4=9$ 通りの増加部分列
  - $\{5,1,3,6\} \rightarrow 5+5=10$ 通りの増加部分列

- つまり、$5^n, 2*5^n, 4*5^n$ に関する加算はできそう。
  - $3*5^n$ に関する加算はどうする？　上手く $\{2,3,1\}$ を使えたりしないか？？？？？

前提

- ここで一旦 3 進数の話に戻る
- $K = 77 = 2 * 3^3 + 2 * 3^2 + 3^1 + 2 * 3^0$ を表したい時：
  - $A = \{3, 1, 7, 5, 11, 9, 12, 10, 4, 2\}$
  - ただし、同色の前半部分と後半部分は「一対一対応している」とする

考察 4

- 取り敢えず $\{2,3,1\}$ と $\{3,1,2\}$ をなんとか合成させたい
- そこで、bit_i を 5 進数で表したときの i 桁目（5^i の位）とする
  - $bit_i = 0,1,2,3$ の時、この桁に一対一対応する前半部分は $\{2,3,1\}$ 的な感じにする
  - $bit_i = 4$ の時、この桁に一対一対応する前半部分は $\{3,1,2\}$ 的な感じにする
- ただし最上位の桁に関しては上手くやる
  - 1 ページ前の数列の白の部分に関しては、最上位の桁の数を B とするとき、取り敢えず「白い部分だけの増加部分列の個数が丁度 B 個」となるような数列にさえすればよい）

897 点解法

- 考察 4 の方針で行く
- 例えば、$K = 108$ （下のビットから 3, 1, 4）の場合
  - 0 桁目まで： $\{3, 5, 1, 2\}$ → $\{2, 3, 1\}$ 的な数列を使う
  - 1 桁目まで： $\{3, 5, 1, 7, 11, 9, 6, 2\}$ → $\{2, 3, 1\}$ 的な数列を使う
  - 2 桁目まで： $\{3, 5, 1, 7, 11, 9, 17, 13, 15, 16, 6, 2\}$ → $\{3, 1, 2\}$ 的な数列を使う
  - これで $L = 12$ で数列を作れる

897 点解法

- 結局長さはどれくらいなのか？
  - 5 進数であり、$5^{26} \leq K < 5^{27}$
  - 普通に実装をすると、$4 * 27 - 1 =$ 長さ 107 → 30 + 70 + 735 = 835 点
  - 上手く実装をすると、$4 * 27 - 2 =$ 長さ 106 → 30 + 70 + 797 = 897 点
- これで全体の 6 割程度の点数が取れた！
  - えっ…　まだ 6 割？？？

897 点

1036 点解法

- この得点となってくると L を 1 減らすことが重要
  - $L = 1$ の違いで 70 ~ 80 点は違ってくる
  - この 1 文字を削り出せ！！！

- 良さそうな解法として、「今まで使った 2, 3, 5 進数を総動員する」方法がある

進数	最小長さ	平均長さ	最大長さ
2 (225 点)	62	93	124
3 (486 点)	76	102	115
5 (897 点)	78	100	106

※ビット数が最大に限りなく近い場合

1036 点解法

- 2 進数、3 進数、5 進数の方法で答えを求め、最も数列の長さが短い場合のを出力すれば良さそう
  - ランダムケースではないが、この場合は 5 進数で表したときの桁が $\{4, 4, 4, 4, \ldots\}$ となるようなコーナーケースにも対処できそうである
- この場合、writer の実装では $L = 105$ に減少
  - 30 + 70 + 864 = 964 点獲得

964 点

考察 5

- 長さ 3 で 5 進数を表す方法はあった
  - {2, 3, 1} と {3, 1, 2} を両立することである。
- 果たして長さ 4 では最大で何進数を表す方法があるのか？
  - 実は 7 進数まで行ける！
  - {2, 4, 1, 3}, {3, 2, 4, 1}, {3, 4, 1, 2}, {4, 1, 3, 2}, {4, 2, 1, 3} を両立する
- しかし、この場合あまり良い点数が出ない
  - $L = 110$ で 30 + 70 + 578 = 678 点

考察 5

- では長さ 5, 6, 7, … だと何進数まで表せるのか？　ということを考えてみる
- これは、コンピューターを使って実験をすることで求めることができる
  - 何進数まで表せるのかは以下の表のようになる

長さ	3	4	5	6	7	8	9
最大進数	5	7	11	19	29	47	77
想定される L	106	110	107	103	103	100	99
想定される得点	897	678	835	1115	1115	1392	1500

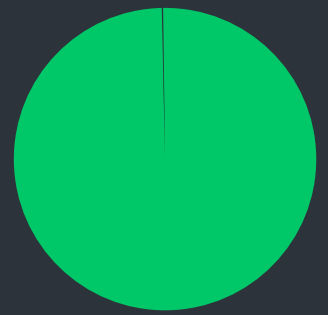
1115 点／1392 点解法

- 以下のような方法を使えば、長さ 7 あたりまで求まり 1115 点
  - まず最初に進数を全探索する。このステップに $O(BN! * N^2)$ かかる。B は後述。
  - 出来得る最大の進数を B とする。長さ 7 であり増加部分列の種類数が B であるような順列を全て vec に記録する。このステップに $O(N! * N^2)$ かかる。
  - それぞれの桁の状態（$0 \sim B-1$ まであり得る）においてどの順列を使い、どの値を追加すればよいのかを $O(P * N^2)$ で求める。ただし P は vec の要素数。

1115 点

満点解法

- 長さ 9 でも間に合うようにするために、$B = 77$ 進数を最初から自分のパソコンに求めておくと、計算量が減る
  - $O(N! * N^2)$ で、$N = 9$ のとき余裕で間に合う
- ちなみにちょっと実装をミスると $L = 100$ の 1392 点になったりします
- これでようやく満点です
  - 長いスライドを最後までお読みいただきありがとうございました！



1500 点

サンプルコード

- 225 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4821709
- 456 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4821801
- 486 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4821806
- 897 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4823101
- 964 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4823170
- 満点解法： https://atcoder.jp/contests/s8pc-6/submissions/4826081

得点分布

- 1500 点： 8 名
- 779 点： 1 名
- 456 点： 1 名
- 429 点： 1 名
- 380 点： 1 名
- 225 点： 17 名
- 100 点： 13 名
- 30 点： 10 名
- 0 点：3 名
- 未提出者： 608 名

平均点

29.4 点

(1.96%)

I – Garden 2

Square869120contest #5 解説

この問題の最初に

- I 問題のような課題は、俗に「マラソン型課題」と呼ばれているものです。
- このような問題では、解の最適さによって点数がもらえます。
  - かなり最適な解を出しても、満点がもらえない場合が多いです。
  - ただ全く最適な解から遠くても、部分点がもらえる場合があります。
- ですので、s8pc においてマラソン型課題は「初心者が逆転するためのチャンス」の問題でもあります。
- では本質に入っていきましょう。

問題概要

- $H * W$ の庭が与えられます。
- マス (i, j) には、美しさ $A_{i,j}$ の花が咲いています。

- あなたはいくつかの花を伐採し、以下のような庭を作りたいです。
  - 伐採されていない花のマス (a, b) から、別の伐採されていない花のマス (c, d) まで、上下左右に隣り合う伐採されていない花のマスのみを辿って、同じマスを二度通らずに行く方法はちょうど 1 通り存在する。
  - 言い換えると、伐採されていない花を頂点とし、ある伐採されていない花と四近傍に隣接する花を結んだものを辺とすると、グラフは連結な木になる

- 伐採されていない花の美しさの合計を出来るだけ多くしてください。

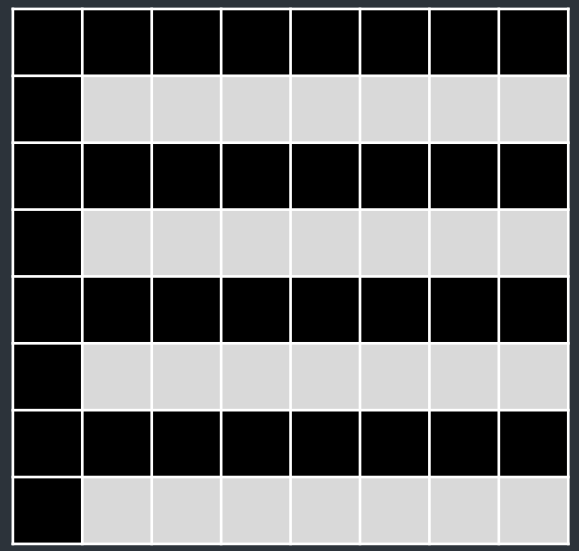
テストケースについて

- 3 ケース： $H = 5, W = 5$
- 3 ケース： $H = 5, W = 500$
- 9 ケース： $H = 500, W = 500$

- 各テストケースにつき 100 点満点で採点される
  - 詳しい配点は問題文を見てください…。

264 点解法

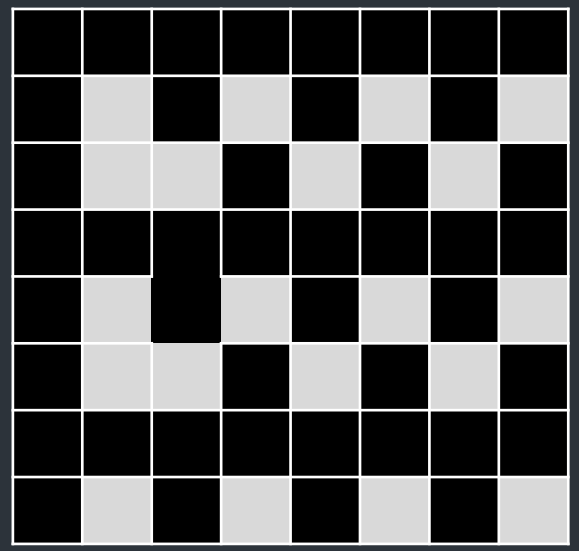
- 取り敢えず適当に「美しい庭」を作りたい
- 半分くらいの花を残すことを考えると、以下のような庭ができます
  - 基本的には、奇数行目の花を残す。残す花の場所は黒、伐採する花の場所は白。



この場合約半分の花が残ります。花の美しさの平均は 5 なので、$L' = 5 * \frac{1}{2} = 2.5$ 程度が見込めます。大きいケースでは 1 ケース当たり 11 点しか取れませんが、小さいケースではもう少し取れて合計 264 点です。

447 点解法

- もう少し多くの花を残すことは出来るのか？
- 半分くらいの花を残すことを考えると、以下のような庭ができます
  - 基本的には、3 の倍数行目を残す。間の行はできるだけ埋めることを考える。



この場合約 3 分の 2 の花が残ります。花の美しさの平均は 5 なので、$L' = 5 * \frac{2}{3} \fallingdotseq 3.33$ 程度が見込めます。大きいケースでは 1 ケース当たり 28 点しか取れませんが、小さいケースではもう少し取れて合計 447 点です。

ケース最適化を考える ①

- そこでケースを振り返ってみる

- ① 3 ケース： $H = 5, W = 5$
- ② 3 ケース： $H = 5, W = 500$
- ③ 9 ケース： $H = 500, W = 500$

- ① くらいは全探索で出来るのでは？？？

ケース最適化を考える ①

- 通り数は $2^{25} = 33554432$ 通りある
- 1 通りの場合の「美しい庭判定」に $O(HW)$ かかるため、少し厳しそう
  - しかし、枝刈り全探索をすれば大丈夫
  - 具体的に言えば、DFS で全探索行う場合は、今までの「庭の得点」の最大値を切るような庭は絶対ダメ、13 個以上 18 個以下の花が残っていないとダメ…　などの様々な枝刈りの手法が存在する
- そうすると、ケース #1 ~ #3 までの得点が 190 点上がり、合計 637 点
  - 264 点解法（1/2 を埋める）と並列して書いた場合でも 461 点まで上がる

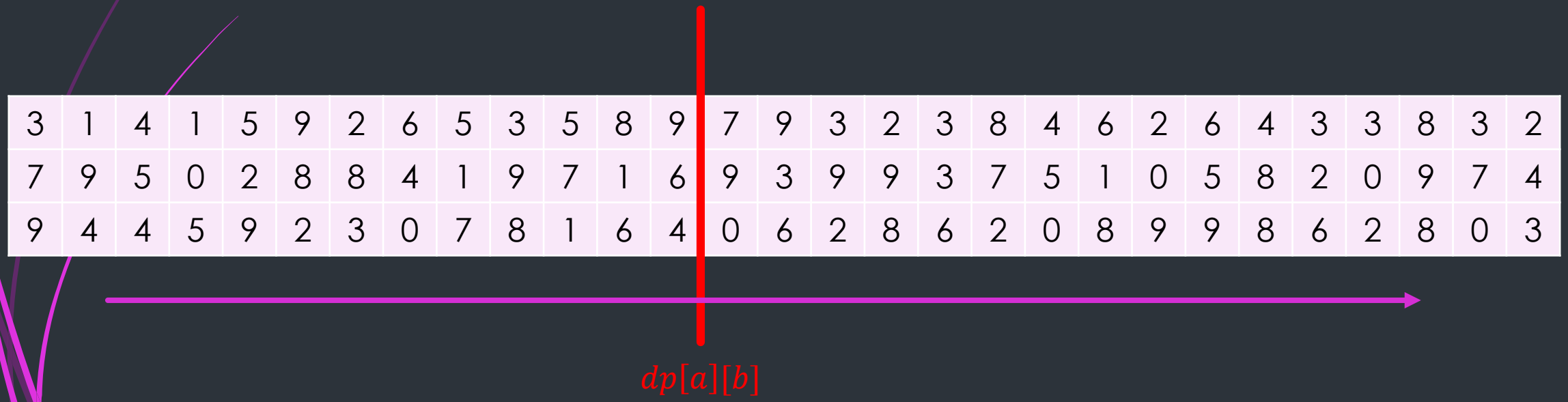
ケース最適化を考える ②

- $H = 5$ のケースが追加で 3 ケースある
- 以下の図のような、縦に短く横に長いグリッド

3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3	2	3	8	4	6	2	6	4	3	3	8	3	2
7	9	5	0	2	8	8	4	1	9	7	1	6	9	3	9	9	3	7	5	1	0	5	8	2	0	9	7	4
9	4	4	5	9	2	3	0	7	8	1	6	4	0	6	2	8	6	2	0	8	9	9	8	6	2	8	0	3

ケース最適化を考える②

- 左から右へ、「一個前の列の伐採されてる花の状態 A 」と「一個前の列について、どの花がどの花と連結であるのかの状態 B」を両方持った DP をすることを考える

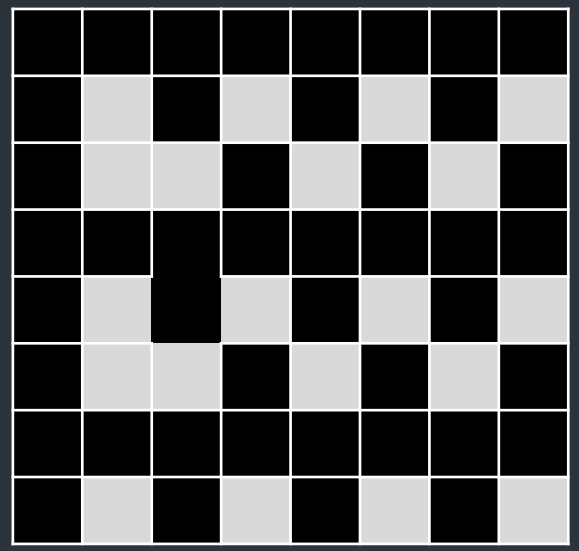


ケース最適化を考える②

- (A, B) の組の通り数はベル数などを使えば見積もることが出来るが、$H = 5$ の時高々 51 通り
- DP の計算時間は $O(51^2 * W)$ なので、$W = 500$ のとき余裕で間に合う
  - そうすると、Case #1 ~ Case #6 まで全て 100 点になる
- 合計点は、以下のようになる
  - 445 点解法 (2/3 を残す) を大きいケースに対して行うと、856 点
  - 264 点解法 (1/2 を残す) を大きいケースに対して行うと、699 点

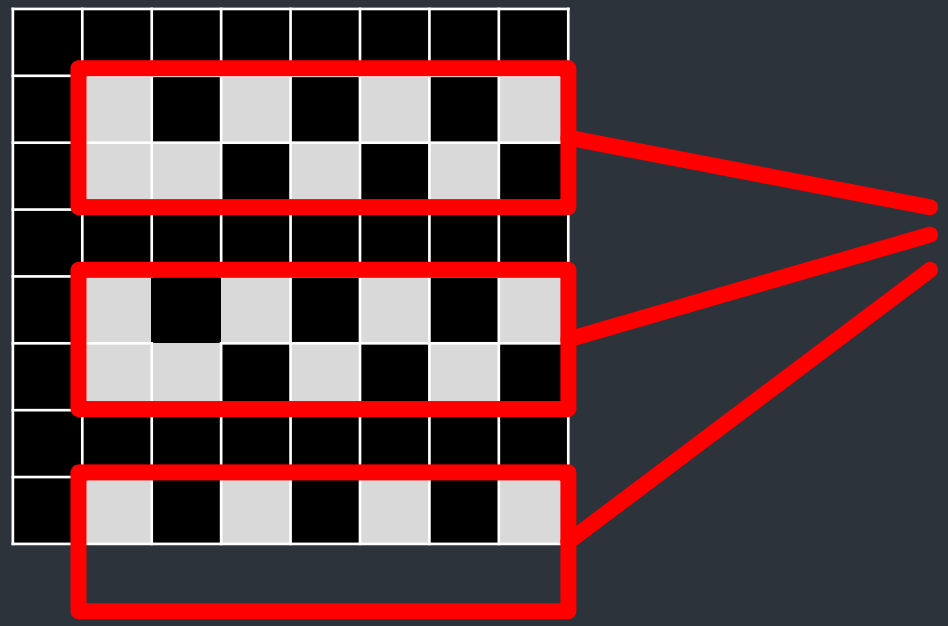
考察 (E869120 解法)

- ここからは、$H = 500, W = 500$ に対する本質的改善を行わなければならない
- 果たしてどうすればよいのか
  - 取り敢えず 445 点解法をベースとして考えてみることにした



考察 (E869120 解法)

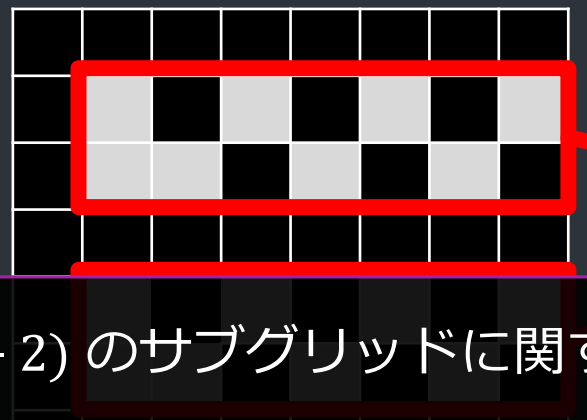
- ここからは、$H = 500, W = 500$ に対する本質的改善を行わなければならない
- 果たしてどうすればよいのか
  - 取り敢えず 445 点解法をベースとして考えてみることにした



この 2 列に対して
それぞれ最適化できないか？

考察 (E869120 解法)

- ここからは、$H = 500, W = 500$ に対する本質的改善を行わなければならない
- 果たしてどうすればよいのか
  - 取り敢えず 445 点解法をベースとして考えてみることにした



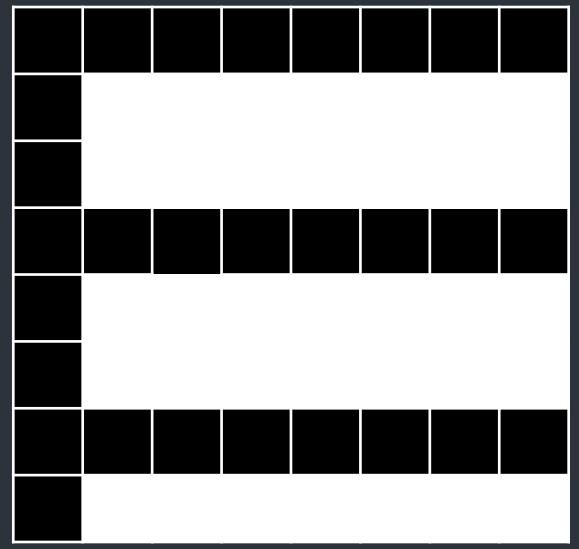
この 2 列に対して
それぞれ最適化できないか？

赤枠の $2 * (W - 2)$ のサブグリッドに関する最適化についてだが、

2 マスが連続しないような配置を考えればよいので、
左から右へ遷移を行う DP を使えばよいです。

960 点解法

- 動的計画法を用いると、合計して $O(HW)$ で、下のグリッドにおいて、まだ白いマスを黒く塗ることが出来る場合に最適解が求まる
  - 得点は 300 + 300 + 360 = 960 点



考察 (square1001 解法)

- 彼は E869120 とは違って、180 度逆の方向から考えた。
- square1001 の考え方
  - できるだけ多くの花を取る必要はない。庭の美しさを最大化すればいいのだ。
  - 1 をたくさん取るより、9 を少し取る方が良い。
- 残った花の繋がりをグラフで表すと、木構造になる。
  - グラフの最大全域木と似たようなものを作ればいいのか？
  - なお、グラフの最大全域木は、最小全域木のアルゴリズム（クラスカル法・プリム法）を使えば求めることが出来る

考察 (square1001 解法)

- しかし、クラスカル法を使う場合、一つの花に対して「残す」と決めた時に複数本辺が新たに繋がる場合がある。（4 近傍なので）

- プリム法のような貪欲的考え方を使おう！
  - プリム法を知らない人は、https://ja.wikipedia.org/wiki/%E3%83%97%E3%83%AA%E3%83%A0%E6%B3%95 を参照すること。

square1001 の解法

- 以下のようなアルゴリズムを行う
  - 最初、マス $(1, 1)$ の花を残すことに決める
  - 次に、まだ花を残すか決まっていないマスの中で、「花を残すと決めたマスに隣接するマス」かつ「追加しても木構造が保持されるマス」のうち、最も花の美しさが高いものを残す
  - 上の操作を、操作ができなくなるまで繰り返す

- 具体例（赤のマスが残すと既に決まっている花、橙のマスが今残すと決まった花）

1	5	4
7	3	9
2	8	6

square1001 の解法

- 以下のようなアルゴリズムを行う
  - 最初、マス $(1, 1)$ の花を残すことに決める
  - 次に、まだ花を残すか決まっていないマスの中で、「花を残すと決めたマスに隣接するマス」かつ「追加しても木構造が保持されるマス」のうち、最も花の美しさが高いものを残す
  - 上の操作を、操作ができなくなるまで繰り返す

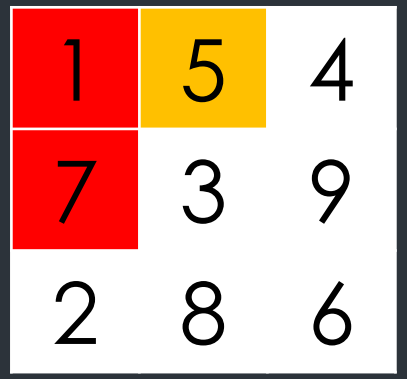
- 具体例（赤のマスが残すと既に決まっている花、橙のマスが今残すと決まった花）

1	5	4
7	3	9
2	8	6

square1001 の解法

- 以下のようなアルゴリズムを行う
  - 最初、マス $(1, 1)$ の花を残すことに決める
  - 次に、まだ花を残すか決まっていないマスの中で、「花を残すと決めたマスに隣接するマス」かつ「追加しても木構造が保持されるマス」のうち、最も花の美しさが高いものを残す
  - 上の操作を、操作ができなくなるまで繰り返す

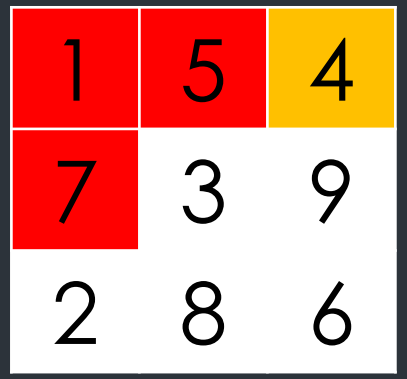
- 具体例（赤のマスが残すと既に決まっている花、橙のマスが今残すと決まった花）



square1001 の解法

- 以下のようなアルゴリズムを行う
  - 最初、マス $(1, 1)$ の花を残すことに決める
  - 次に、まだ花を残すか決まっていないマスの中で、「花を残すと決めたマスに隣接するマス」かつ「追加しても木構造が保持されるマス」のうち、最も花の美しさが高いものを残す
  - 上の操作を、操作ができなくなるまで繰り返す

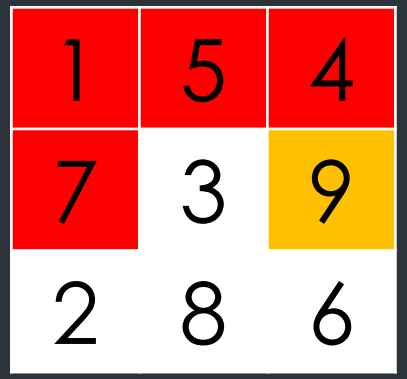
- 具体例（赤のマスが残すと既に決まっている花、橙のマスが今残すと決まった花）



square1001 の解法

- 以下のようなアルゴリズムを行う
  - 最初、マス $(1, 1)$ の花を残すことに決める
  - 次に、まだ花を残すか決まっていないマスの中で、「花を残すと決めたマスに隣接するマス」かつ「追加しても木構造が保持されるマス」のうち、最も花の美しさが高いものを残す
  - 上の操作を、操作ができなくなるまで繰り返す

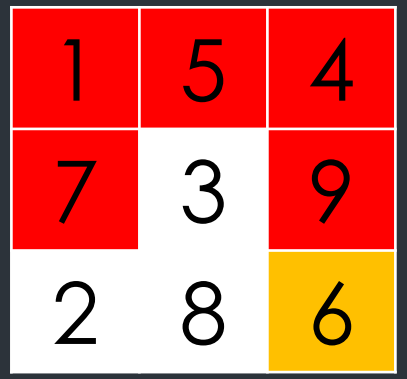
- 具体例（赤のマスが残すと既に決まっている花、橙のマスが今残すと決まった花）



square1001 の解法

- 以下のようなアルゴリズムを行う
  - 最初、マス $(1, 1)$ の花を残すことに決める
  - 次に、まだ花を残すか決まっていないマスの中で、「花を残すと決めたマスに隣接するマス」かつ「追加しても木構造が保持されるマス」のうち、最も花の美しさが高いものを残す
  - 上の操作を、操作ができなくなるまで繰り返す

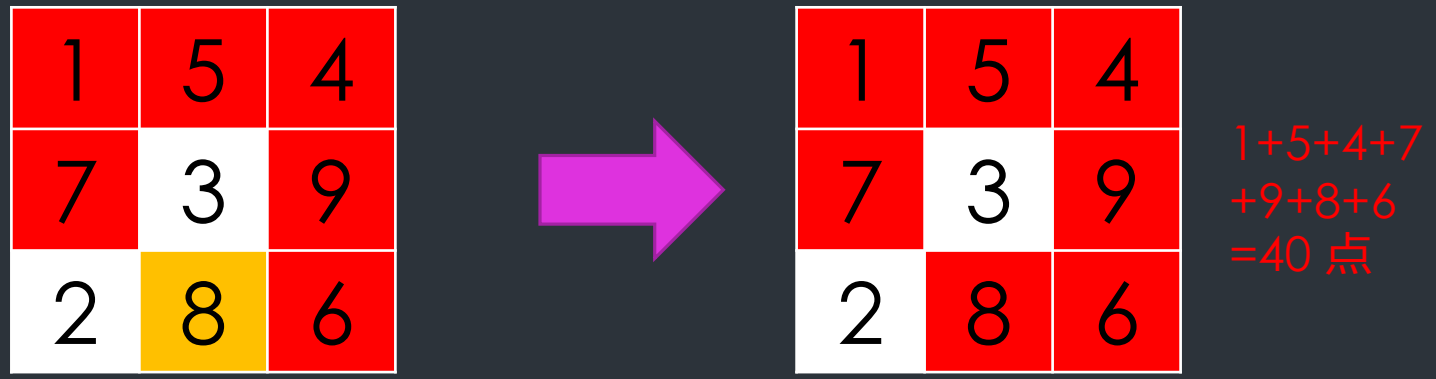
- 具体例（赤のマスが残すと既に決まっている花、橙のマスが今残すと決まった花）



square1001 の解法

- 以下のようなアルゴリズムを行う
  - 最初、マス (1, 1) の花を残すことに決める
  - 次に、まだ花を残すか決まっていないマスの中で、「花を残すと決めたマスに隣接するマス」かつ「追加しても木構造が保持されるマス」のうち、最も花の美しさが高いものを残す
  - 上の操作を、操作ができなくなるまで繰り返す

- 具体例（赤のマスが残すと既に決まっている花、橙のマスが今残すと決まった花）



square1001 の解法

- この解法は、普通に実装すると $O(H^2W^2)$ かかるが、優先度付きキューなどを用いて上手く実装すると $O(HWlogHW)$ で出来る
  - ページ数の都合上具体的な説明は省略します。すみません。実装が軽いので、サンプルコードを見てください。
- この貪欲法だけを書くと 949 点
  - 全探索（Case #1 - #3）について場合分けして対応すると 300 + 735 = 1035 点
  - 動的計画法（Case #1 - #6）について場合分けして対応すると 300 + 300 + 571 = 1171 点
- これで 7 割 5 分以上の点数が取れた！

[前提] square1001 の解法は何故成立？

- ある頂点からランダムに木を生成していくと、
  - 最小で 50% の花が残される
  - 最大で 66.666667% の花が残される
  - 平均だと 59% 程度の花が残される！！！！！！！！！
- なので、E869120 の解法（ほぼ 66.666667%）より 多少残す花の数が少なくても、平均の花の美しさが少し多ければより高得点が取れる！！！
- ただ 59% より少しだけでも多い生成方法を考えたい

さらなる改善

- 今までは純粋に「最も花の美しさが大きいもの」を追加することにしていた
  - マラソンマッチっぽく言うと、評価関数は $A_{i,j}$
- これを少し変えて、点数をどうにかして上げることは出来ないか？
  - 該当するマスの周りにある花の美しさは、小さい方が良さそう（このマスを確定させることによって、周りのマスが残される可能性が低くなるから）
  - マス (i,j) に対する評価関数を $15A_{i,j}$ −（マス (i,j) に 8 近傍で隣り合うマスの中でまだ花を残すと確定させていないマスの花の美しさの合計）とする
- Case #1 - #6 について最適化を行うと 1223 点
  - もう少しパラメーター調整を行うと、1228 点

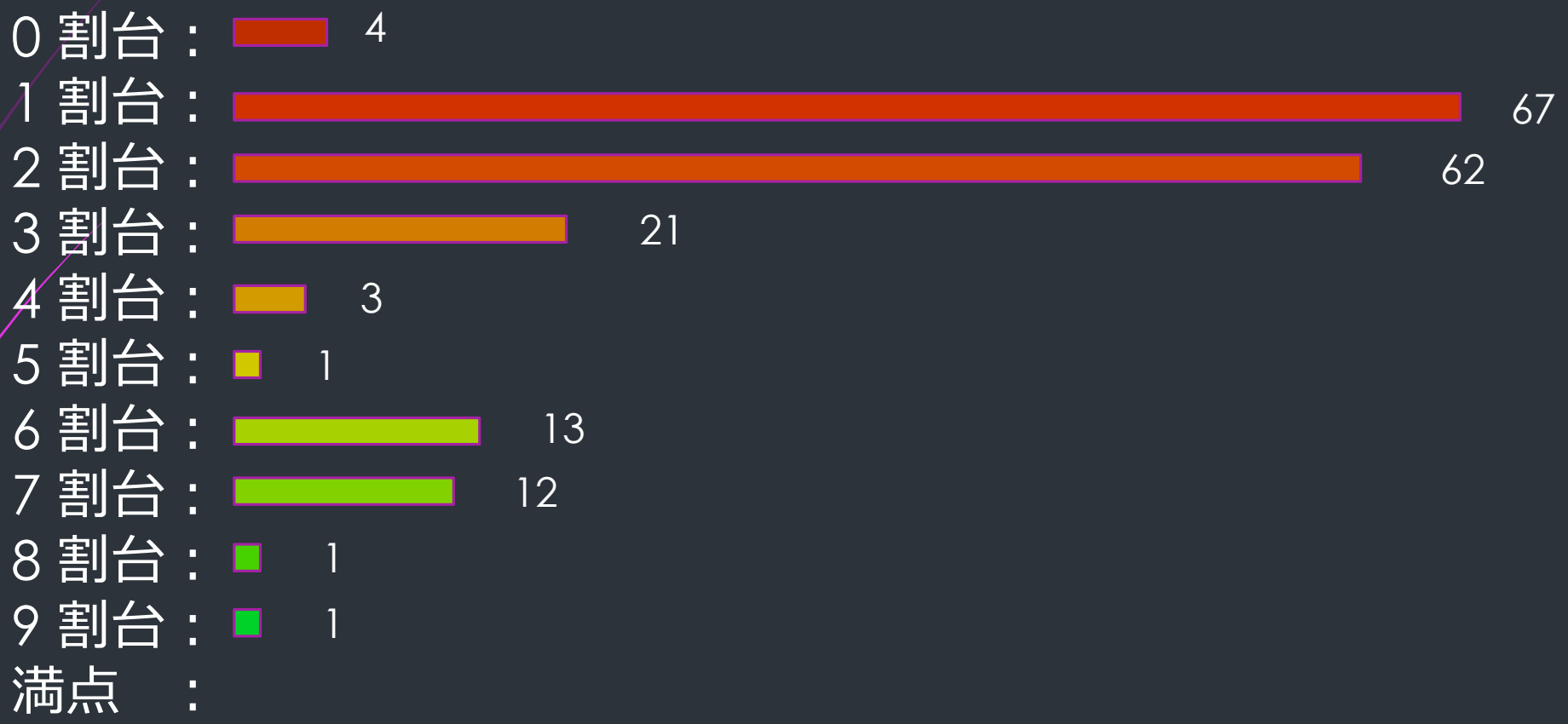
まとめ

解法	#1 - #3	#4 - #6	#7 - #9	#10-#12	#13-#15	合計点
1/2 を埋める	103	62	33	33	33	264
2/3 を埋める	110	81	85	85	84	445
全探索 [H, W=5] + 2/3	300	81	84	85	84	644
DP [H = 5] + 2/3	300	300	84	85	84	853
DP [H = 5] + 2/3 前提最適化	300	300	120	120	120	960
square1001 的貪欲	214	164	189	191	191	949
DP [H = 5] + 貪欲	300	300	189	191	191	1171
DP [H=5]+貪欲パラメタ調整	300	300	209	210	209	1228

サンプルコード

- 264 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4885633
- 447 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4885712
- 853 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4886598
- 960 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4888127
- 949 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4888429
- 1171 点解法： https://atcoder.jp/contests/s8pc-6/submissions/4888525
- square1001 最終解（$H = 5$ 場合分けなしで 1092 点）
  - https://atcoder.jp/contests/s8pc-6/submissions/4890735
- E869120 最終解（$H = 5$ 場合分けして 1228 点）
  - https://atcoder.jp/contests/s8pc-6/submissions/4890194

得点分布



おわりに

square869120Contest #6 解説

結果発表

- 1 位：yosupo さん (7085pts)
- 2 位：zscoder さん (6425 pts)
- 3 位：WA_TLE さん (6105 pts)
- 4 位：yutaka1999 さん (6000 pts)
- 5 位：heno239 さん (5226 pts)
- 6 位：hos_lyric さん (5092 pts)
- 7 位：natsugiri さん (5000 pts)

参加していただきありがとうございました！