AHC070「Ward the Village」— 解法紹介

Writer: HNN_8127

問題の紹介と、3つの解法例（2ベクトル解／ペア解法(スコア全計算)／ブロックDP※）

> 村は N×N マスに区切られている。村長は **3本**の「神力の通り道」を選び、**10000回**起こる怪異のできるだけ近くにお札を立て続け、被害（危険度）を最小化する。

※ペア解法(スコア全計算)までは人力。ブロックDPは、Writer+Claude Opus5。

問題の説明

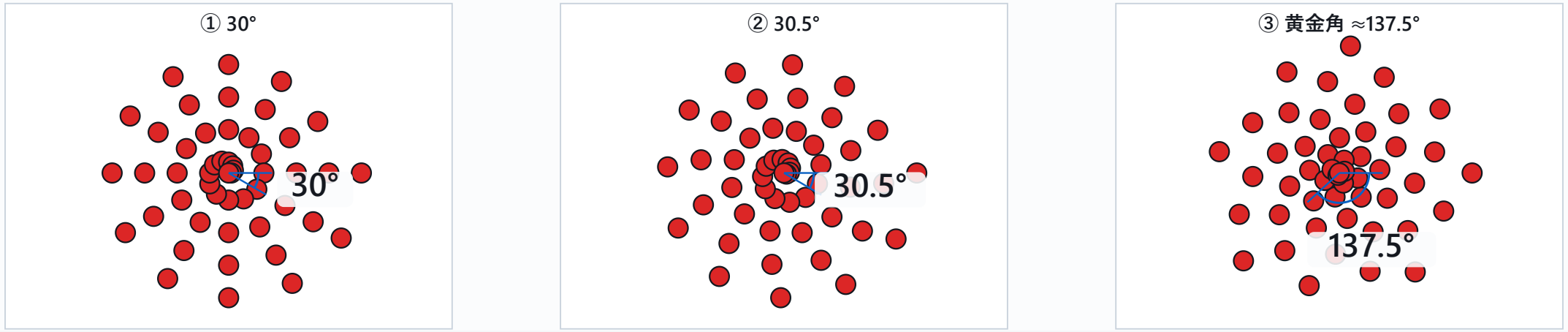
- 100×100 のトーラス盤。村長は社 (0,0) から出発
- 先に「神力の通り道」となる移動ベクトルを **3本** 決める（各i,j成分が0〜99）。
  以下、この3本を $\vec{v}_0, \vec{v}_1, \vec{v}_2$ と呼ぶ（本資料独自の呼び方。問題文にこの記号はない）
- 10000 ターン、毎回どれか1本を選んで移動 → 移動先にお札を立てる（重ねても意味ない、一度立てたら残り続ける）
- その後 (a_t, b_t) に怪異発生。危険度に $\lfloor d_t \sqrt{t+1} \rfloor$ を加算（d_t は最近傍のお札までのマンハッタン距離）
- 距離はループしない（移動はループする）。怪異の列は全マスの一様ランダム順列で、最初に全部与えられる

$$\text{スコア} = \mathrm{round}\left(\frac{10^6 N^3}{D+1}\right) = \mathrm{round}\left(\frac{10^{12}}{D+1}\right)$$

D（危険度の総和）を最小化するほどスコアが高くなる。

作問経緯

元ネタは**弾幕シューティングゲーム**。弾を発射する角度を30°と30.5°とわずかに変えるだけで、密集度やプレイ感が大きく変わることに興味を持っており、最も隙間なく均等に埋めるにはどうすればいいか考えたことがあった。



最も隙間なく均等に埋めるためには発射角を黄金角(x=137.51°、つまりxと360°-xが黄金比になるx)にするとよい。ひまわりの種もこの配置になることが知られている。黄金比は、"有理数から最も遠い無理数"(連分数展開で大きい数が出ない)のため、決まった周期ができにくい。

> → 弾の発射角度は360度でループする1次元であるが、これを2次元に拡張し、盤面を隙間なく埋める問題にした。

・「バラバラに埋めたい」という着想が先にあったが、スコアの定義決めに苦戦した。初めは「お札以外のマスに入る最大の正方形」を小さくする問題を考えたが、スコア計算自体がアルゴの問題になるため、採用できなかった。よりシンプルなマンハッタン距離を思いつき、それを採用した。

・M=2で全マス埋められることを初めから想定しており、非自明な最小の数としてM=3を採用。M=3のペア解法が作問時に見え、単純な焼きなましでは解けそうになかったため、これを採用とした。

・簡単に焼かれないように余裕をもって大きめのN=100にした。しかし、距離マップを構築しないとまともにスコアを計算できないのは、上位以外の参加者やPythonなどの言語の使用者にとってあまり良くなかったかもしれない。AHC070なので、N=70がよかったか。（奇数はダメ）

解法ごとの得点と順位

解法	得点	本番順位	パフォーマンス
解法A：2ベクトル格子を入力ごとに全探索	492,985,189	119位	1856
解法A+：解法Aの縮約部分を展開	494,081,231	107位	1907
解法B：Writer人力（ペア解法(スコア全計算)）	675,004,378	14位	2638
解法C：Writer with AI（ブロックDP）	782,279,028	4位	2983

解法A：2ベクトル格子 — 全マスを1回ずつ塗る条件

$\vec{v}_0, \vec{v}_1, \vec{v}_2$ を2つの基底ベクトル $\vec{w}, \vec{u}$ を使って

$\vec{v}_0 = \vec{w},\ \vec{v}_1 = \vec{w} + \vec{u}$ と定める($\vec{v}_2$は未使用)。**100手に1回だけ $\vec{v}_1$ を使う。**

このとき $s = 100q + r$ 手目に塗るマスは

$$\vec{p} = r\,\vec{w} + q\,\vec{u} \pmod{100}$$

全10000マスをちょうど1回ずつ塗る条件は

$$\gcd(\det(\vec{w}, \vec{u}), 100) = 1$$

記法：$\gcd(a, b)$ は a, b の最大公約数。

$\det(\vec{w}, \vec{u})$ は $\vec{w} = (w_i, w_j),\ \vec{u} = (u_i, u_j)$ から作る行列式 $w_i u_j - w_j u_i$。

$\vec{p}$の式を見ると、高速に動く成分(第一層)と低速に動く成分(第二層)に分離可能。

- 第1層：$\{r\,\vec{w}\}$ が間隔≈10 の格子（ほぼ毎ターン動く）
- 第2層：$\{q\,\vec{u}\}$ が10×10の領域を埋める（100ターンに一度だけ）

記法：$\{r\vec{w}\} = \{r\vec{w} \bmod 100 : r = 0, \ldots, 99\} = \langle \vec{w} \rangle$ — $\vec{w}$ を整数倍して作れる点の集合（$=\vec{w}$ が生成する巡回群）。

探索空間の削減

以上の議論より、条件を満たす$(\vec{w}, \vec{u})$について、盤面全体を埋め尽くす手順が存在するので、これを**全探索**したい。しかし、素朴に数えると $(\vec{w}, \vec{u})$ は 10^8 通り（=1億）、gcd 条件を満たすものだけでも 2,880万通りあり、TLEになる。

正直、適当に時間いっぱい探索しても水色〜青のパフォーマンスが得られるが、せっかく解説するので、部分群ごとに分けて真面目に取り扱う。やや煩雑なので、以下の議論はスキップして解法Bへ進んでも構わない。

ここで、第1層、第2層の格子ごとに分けて考える：

$$28{,}800{,}000 = \underbrace{180}_{\text{格子}\langle\vec{w}\rangle} \times \underbrace{40}_{\vec{w}\text{の生成元}=\varphi(100)} \times \underbrace{40}_{u\text{クラス}=\varphi(100)} \times \underbrace{100}_{|\langle\vec{w}\rangle|\text{（クラス内代表）}}$$

	個数	理由	記号
格子 ⟨w⟩	180	第1層の格子の形	–（数え上げ）
w の生成元	40	第1層の訪問の順番	φ(100)
u クラス	40	第2層の格子の形	剰余群 ≅ Z_{100} の生成元
クラス内代表	100	第2層の訪問の順番	|⟨w⟩|
(w,u) の全組み合わせ	**28,800,000**	**180 × 40 × 40 × 100**	

同じ格子でも生成元の取り方でラウンド内の順序が変わるが、影響は小さいので代表元1つに固定。この「代表元1つに固定」が本当にコスト中立か確かめた実験は解法A+へ。

解法A：入力ごとに 7,200 通りを全探索

1. gcd(w,100)=1 の w のうち、⟨w⟩ の
   辞書順最小の生成元だけ残す　　→ 180格子
2. 各格子について「全マスから⟨w⟩までの
   平均距離」を多始点BFSで求め、良い順に並べる
3. 各格子で、生成元クラスの代表 u = j·u。
   （gcd(j,100)=1 の40通り）をすべて
   実際の入力でシミュレーション
4. 危険度が最小だった (w,u) を出力

再掲：7,200通りの内訳

	個数	理由
相異なる格子 ⟨w⟩	180	1つの格子に生成元がφ(100)=40個
u の有効クラス	40	剰余群 $\cong Z_{100}$ の生成元
探索する組み合わせ	**7,200**	**180 × 40**

高速化2点

- 最初の99手は $\vec{p} = s\,\vec{w}$ で $\vec{u}$ に依存しない → 格子ごとに1回だけ計算して40通りで使い回す（作業量ほぼ半減）
- 途中で暫定ベストを超えたら打ち切り（危険度は単調増加なので厳密）

危険度（スコア）の計算：距離マップは、新しい黒マスを1点追加するたびに**改善範囲だけをダイアモンド形に差分更新**するBFSで管理する（毎回全マスから再計算はしない）。これにより7,200通り全てのシミュレーションが時間内に収まる。

結果

150ケースでのスコア合計：**492,985,189**

・入力ごとに選び直しても、固定ベクトル1組で通した場合とほとんど変わらない。

怪異の順番を「ベクトルの選択」だけに使っても、ほとんど効かない（この解法は怪異の順番と無関係にマスを塗る順番が決まってしまうため）。

・このあたりの細かい順位は、2,880万通りからどうやって探索するか、高速化ができたかによって左右されると思われる。

解法A+：同一視で畳んだ4,000通りを展開すると

アイデア

解法Aの縮約（2,880万→7,200通り）は、「**全マスをちょうど1回ずつ塗れるか**」**については無条件に正しい**（gcd条件は同値変形なので）。ただし、先述の第1層、第2層の2つの格子が等しいものの、訪問順が異なるため、**実コストDは微妙に変わってくる。**

7,200通り探索の勝者 $(\vec{w}, \vec{u})$ について、同一視で畳んでいた

$$\underbrace{40}_{\text{同じ格子の生成元}} \times \underbrace{100}_{\text{同じ}\vec{u}\text{クラスの代表}} = 4{,}000 \text{ 通り}$$

を全展開し、実際に入力でシミュレーションし直して最小Dを再探索した。評価回数は7,200+4,000=11,200回になる。

結果

	score（150ケース合計）
解法A（7,200通りの最善）	492,985,189
解法A+（4,000通り展開後の最善）	**494,081,231**

scoreで約0.22%改善。（492,985,189→494,081,231）

→ 伸び悩みの原因は、解法A(+)が**3本目のベクトル $\vec{v}_2$ を使わず、怪異の順番（入力）も見ずに**固定パターンで塗る順序を決めてしまうこと。$(\vec{w}, \vec{u})$ をどう選び直しても、この2点を変えない限り伸びしろは小さい。

行動列そのものを怪異の順番に合わせるには、3本目のベクトルを使って入力を見る必要がある⇒解法B・C

・1ターン目だけ3番目の移動を使うなどして小さい工夫を行うと、スコア500M程度まで伸びるようです。順位は62位、パフォーマンスは2141相当です。

解法B：ペア解法(スコア全計算) —3番目の移動を使った「2点交換」の仕組み

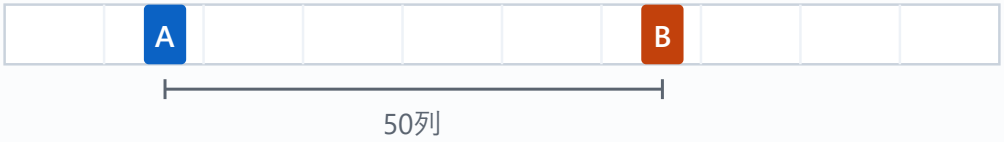
使うベクトルは $\vec{v}_0, \vec{v}_1$ を適当に固定（解法Aで良い値を前計算）。
例えば、$\vec{v}_0 = (57, 77),\ \vec{v}_1 = (37, 64)$とする。基本パターン（100手に1回だけ $\vec{v}_1$）で移動する。

ここで $\vec{v}_2$ を $\vec{v}_0 + (0, 50)$ で定義する（2回足すと (0,100)≡(0,0) に戻る、**位数2の元**）。

また、$\vec{v}_0, \vec{v}_1$ のみの基本パターンだけだと 5000手で位置がちょうど (0,50) ずれる、つまり時刻 s と 時刻 $s + 5000$ は、必ず同じ行で50列離れた「ペア」のマスに対応する。
これを利用すると、**jターン目とj+5000ターン目に訪問する2マスだけの訪問順の交換**ができる：

> $\vec{v}_0 \Leftrightarrow \vec{v}_2$ の反転 を **j, j+1, j+5000, j+5001 の4箇所セット**で打つと、
> そのペア1組だけが前半・後半で入れ替わり、**残り 9,998 マスは全く動かない。**
> 入替なので全マス被覆も崩れない。

① 同じ行で50列離れた2マスが「ペア」



$v_2 - v_0 = (0,50)$ は**位数2**（2回で元に戻る）

② 前半と後半で、ペアを1つずつ塗る

③ ビット f[s] を1つ反転 ＝ このペアだけ入替

- どちらを選んでも全10000マスがちょうど1回ずつ塗られる
- 怪異が先に来る方を前半に置くほど危険度が下がる
- 4,950ビットの山登りで、この入替を繰り返す

> $(0, 50)$以外に$(50, 0), (50, 50)$をも試し、よいものを採用することが考えられるが、実行時間が3倍になるわりに探索空間は3倍にしかならないので、却下した。

解法B：ペア解法(スコア全計算) — なぜ4箇所セットで入れ替えるのか

$\vec{v}_2$ を1回打つと、**それ以降ずっと** (0,50) ズレたままになる。

でも $\vec{v}_2 - \vec{v}_0$ は位数2なので、**すぐ次の手でもう1回打てば元に戻る。**

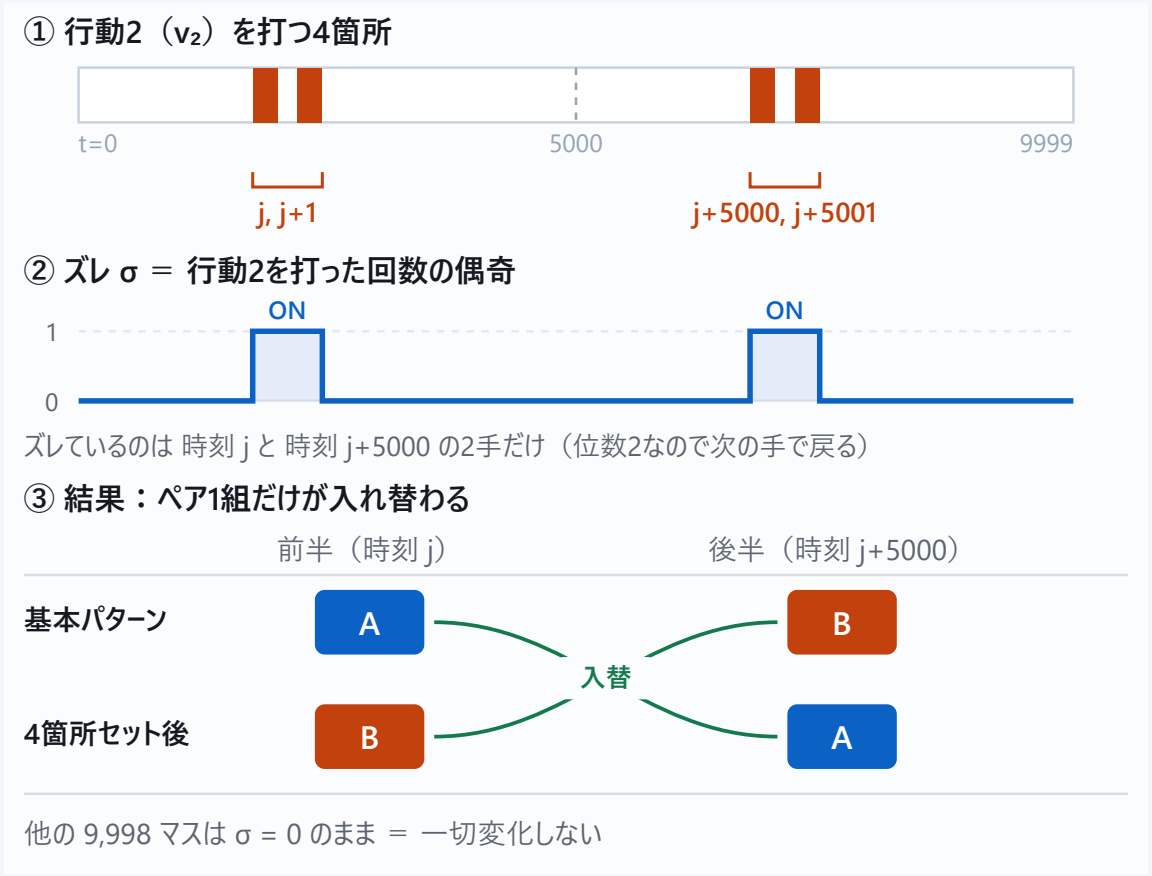
```
m[j] = 2,  m[j+1] = 2
  → 時刻 j の 1手だけ 一時的にズレる
```

ただし前半だけではこれで壊れる。時刻 j も 時刻 j+5000 も B を塗ってしまい、**A が誰にも塗られなくなる**（全マス被覆が崩れる）。

```
m[j+5000] = 2,  m[j+5001] = 2
  → 後半も同じだけズラす
```

4箇所そろって初めて A と B がちょうど入れ替わる。残り 9,998 マスは一切動かない。

単純な山登り版はこの4箇所セットを1組ずつ試す。**Writer解はこれをビット配列で一般化**したもので、f[j] = 1 が「ペア j を入れ替える」を直接表し、行動2は **1 が連続する区間の両端**にだけ立つ。孤立した 1 がちょうどこの4箇所セットに一致する。



①②は模式図（j と j+1 は実際には隣り合う1ターン差）。時刻の縮尺は正確ではない。

解法B：ペア解法(スコア全計算) — 山登りと結果

ビット配列 f を1つずつ反転する山登り法：

```
for i in 0..4999:
    f[i] を反転
    ビット列から行動列を作り直す
      （f の境界 j ごとに m[j]=m[5000+j]=2）
    全10000ターンをゼロから完全に再計算
    改善すれば採用、悪化すれば元に戻す
```

実装上の配列長は 5000 だが、効くのは 4,950 ビット。
$\vec{v}_0, \vec{v}_1$のうち、$\vec{v}_0$にしか適用できないため。

スコアは毎回、**工夫した全計算**を行う（距離マップの差分更新はP6と同じ）： 今回は1ビット反転のたびに盤面全体が変わるため、**t<600** は少数の黒マスを保持し、マスの数だけ愚直に反復／ **t=600** の瞬間だけ盤面全体をBFSで距離マップを1回構築／ それ以降は差分更新に切り替え。これで実効4,950回の全計算が1秒未満で収まる。

差分更新だけの実装でも数千回回り、そこそこの点数が得られる。

f[j] は「ペア j をどちらの順で塗るか」を直接表し、行動2は f の境界にだけ立つ （1ビット反転＝**ちょうど1組のペアの前後入替**）。

150ケースでのスコア合計：**675,004,378**

解法Aに比べ、スコアが**約36.9%高い**（492,985,189 → 675,004,378）。「どのマスをいつ塗るか」を怪異の順番に合わせただけでこれだけ動く。

さらに伸ばす余地：持ち方を工夫すると、工夫した全計算ではなく、この入れ替えによる**スコアの差分だけ**を計算できる。これにより山登りを焼きなましに変えられ、**さらに5%程度**スコアが伸びる見込み（AI禁止の4時間で実装するのは厳しい）。

解法C：ブロックDP ① なぜ (位相, オフセット) に帰着するか

t 手目までに $\vec{v}_0, \vec{v}_1, \vec{v}_2$ をそれぞれ c_0, c_1, c_2 回使ったとすると、位置 $\vec{p}$ は

使った順番によらず

$$\vec{p} = c_0\vec{v}_0 + c_1\vec{v}_1 + c_2\vec{v}_2 \pmod{100}$$

$\vec{\delta}_i = \vec{v}_i - \vec{v}_0$ と置いて整理すると

$$\vec{p}_s = s\,\vec{v}_0 + X_s\,\vec{\delta}_1 + Y_s\,\vec{\delta}_2$$

（X, Y は単調非減少：1手で高々どちらかが+1）

> つまり時刻 s に置けるマスは $s\,\vec{v}_0 + \langle \vec{\delta}_1, \vec{\delta}_2 \rangle$ という剰余類に限られる——動ける自由度は $\vec{\delta}_1, \vec{\delta}_2$ が生成する部分群の中だけ。

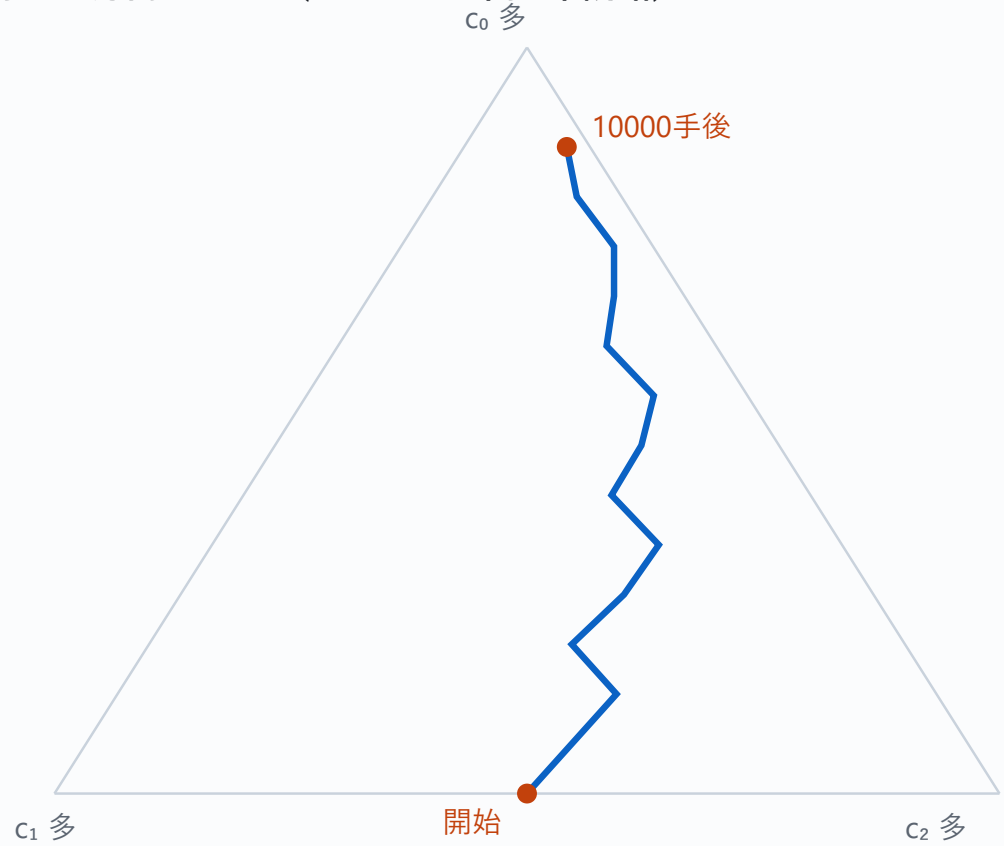
記法：$\langle \vec{\delta}_1, \vec{\delta}_2 \rangle = \{\, i\vec{\delta}_1 + j\vec{\delta}_2 \pmod{100} : i, j \in \mathbb{Z} \,\}$ — $\vec{\delta}_1$ と $\vec{\delta}_2$ の整数倍の和で作れるベクトル全体（「$\vec{\delta}_1, \vec{\delta}_2$ で生成される部分群」）。

$\langle \vec{\delta}_1, \vec{\delta}_2 \rangle$ が位数100の巡回群になるように$\vec{v}_i$を選べば（②参照）、「**高速に回る基底** $\vec{v}_0$」＋「**ゆっくり動くオフセット**」の**2層構造**として全マスが (r, g) で一意に書ける：

$$\vec{p} = r\,\vec{w} + g\,\vec{u}$$

$r = s \bmod 100$（時刻で強制）、$g = g_s$（選べる：オフセット）⇒ g の1手あたりの変化は $0, \alpha, \beta$ のみ

1手 = 3方向のどれか（＝カウント空間の単調路）



同じ格子点＝同じマス。訪問順の自由度はこの路の形だけ。

解法C：ブロックDP ② $\langle\vec{\delta}_1, \vec{\delta}_2\rangle$が位数100の巡回群になるように選ぶ

$\vec{\delta}_1, \vec{\delta}_2$ は $\vec{v}_1, \vec{v}_2$ の選び方次第で**どんな2ベクトルにもなりうる**——このままでは $\langle\vec{\delta}_1, \vec{\delta}_2\rangle$ が位数100になる保証はない。

そこで $\vec{v}_1, \vec{v}_2$ を、**解法Aと同じ格子基底** $\vec{w} = \vec{v}_0,\ \vec{u}$ を使って

$$\vec{v}_1 = \vec{w} + \alpha\vec{u}, \qquad \vec{v}_2 = \vec{w} + \beta\vec{u}$$

の形に**限定して選ぶ**（α, β は整数）。すると

$$\vec{\delta}_1 = \vec{v}_1 - \vec{v}_0 = \alpha\vec{u}, \qquad \vec{\delta}_2 = \vec{v}_2 - \vec{v}_0 = \beta\vec{u}$$

なので $\langle\vec{\delta}_1, \vec{\delta}_2\rangle = \langle\alpha\vec{u}, \beta\vec{u}\rangle = \langle\vec{u}\rangle$ (α, β の最大公約数が1のとき)。

> $\vec{u}$ が**位数100の元**（解法Aと同じ $\gcd(\det(\vec{w}, \vec{u}), 100) = 1$）であれば、$\langle\vec{u}\rangle$ は**ちょうど位数100の巡回群**になる。

つまり $\vec{\delta}_1, \vec{\delta}_2$ を独立に選ぶのではなく、**両方とも同じ $\vec{u}$ の整数倍**になるように設計するのがポイント。

具体例（実際に使われた値）

$$\vec{w} = (89, 29), \quad \vec{u} = (6, 89), \quad \alpha = 54, \quad \beta = 11$$

$$\vec{v}_0 = \vec{w} = (89, 29)$$

$$\vec{v}_1 = \vec{w} + 54\vec{u} = (13, 35)$$

$$\vec{v}_2 = \vec{w} + 11\vec{u} = (55, 8)$$

$\gcd(54, 11) = 1$、$\gcd(\det(\vec{w}, \vec{u}), 100) = 1$ をどちらも満たす（解法Aの全探索で見つけた良い格子と同じ条件）。この $\vec{v}_0, \vec{v}_1, \vec{v}_2$ が実際にP15の出力と一致する。

> **解法Bとの関係：** 解法Bの $\vec{v}_2 - \vec{v}_0 = (0, 50)$ も同じ発想の特殊ケース——**位数2**の元を使い、オフセットは「ズレなし／(0,50)ズレ」の2通りだけだった。
> 解法Cはこれを**位数100**に拡張し、α, β の2方向のジャンプで100通りのオフセットを数手で行き来できるようにしたもの。
> ただし自由度が上がった分、**全マス被覆はもう保証されない**（解法Bは常にペア交換で被覆を保っていた）。それでも問題ない——目的は被覆ではなく**危険度の最小化**である。そのため、これを達成する適切な評価関数を設計する必要がある（③参照）

解法C：ブロックDP ③ 何を最適化するか

狙い：空打ちを減らす

怪異はマスごとに**1回だけ**。すでに怪異が済んだマスにお札を立てても、危険度はあまり下がらない。

非適応の解だとラウンド m で $m/100$ の割合の手が「怪異済みのマス」に落ちてしまい、

全体で約半分（4950手）が無駄になっている。

「まだ怪異が来ていないマス」を優先して塗れれば、その分がまるごと得になる。

評価関数：マス $\vec{p}$ にお札を1枚置いたときの距離場の改善量

$$\mathrm{val}(\vec{p}) = \sum_{\vec{x}} \max(0,\ d(\vec{x}) - |\vec{x} - \vec{p}|) \cdot W(\vec{x})$$

$d(\vec{x})$ は、$\vec{p}$ を置く前の時点での現在の距離場（つまり$\vec{x}$ から最寄りの既存のお札までの距離）。

$W(\vec{x}) = \sqrt{\tau_x}$（$\tau_x$ は $\vec{x}$ に怪異が来る時刻。まだ来ていないマスのみ）。

ここに割引を入れるのが決定的：

$$W(\vec{x}) = \sqrt{\tau_x} \cdot \exp\left(-\lambda \frac{\tau_x - t}{N^2}\right)$$

「今 x を塗る価値」は、**放っておいても後で別の手が塗ってしまえば消える。**

遠い未来の報酬を素で足すと過大評価になり、かえって悪化する。

縦=オフセット g、横=位相 r。灰の破線は非適応（g を100手に1回だけ動かす直線）。青の折れ線が適応解で、α/β のジャンプでまだ怪異が来ていないマス（○）を拾いに行く。

α, β の選び方も効く。β = −α（±1）だと g が滑らかにしか動けず狙ったマスに行けない。α=54, β=11 のように「跳べる」組にすると数手で任意のオフセットに到達できる。

解法C：ブロックDP ④ なぜ「幅100のDP」で足りるのか

$\vec{v}_1, \vec{v}_2$ をそれぞれ X 回・Y 回使ったとすると、時刻 s に居るマスは

$$\vec{p} = r\,\vec{w} + g\,\vec{u}$$

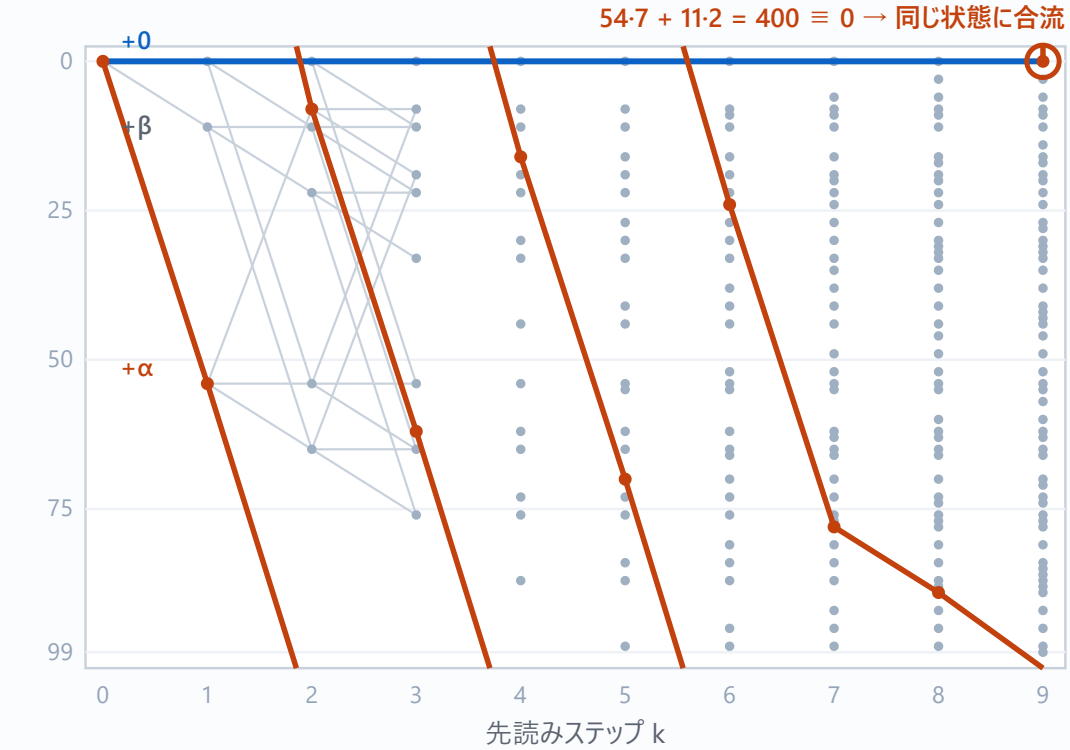
$r = s \bmod 100$（時刻で強制、選べない）、$g = (\alpha X + \beta Y) \bmod 100$（選べる：オフセット）

> **状態は g だけ（(a,b,c) や (X,Y) ではない）。位置 (r,g) のうち r は時刻で固定**されるので自由なのは g のみ：100×100 ではなく 100。
> (X,Y) の組み合わせは1,431通りあっても、g はそれを**100で割った余り**でしかない——100目盛りの時計の針のように、何周させても行き先は100通り。入力がいくら多くても mod 100 を取った時点で答えは0〜99のどれかに収まる（右図の「合流」）。

L=52 手先を読むとき	通り
行動列 3^{52}	約 6.5×10^{24}
使用回数の組 (a,b,c), a+b+c=52	1,431
区別できる状態 g ∈ Z_{100}	**100**

遷移も $g \to g,\ g+\alpha,\ g+\beta$ の3通りだけ。DPの更新は $100 \times 3 \times 52 \approx 1.6$**万回**で終わる。

3^k 通りに枝分かれしても、縦は 100 行しかない



青 = v_0 を9回 (X,Y)=(0,0)　橙 = v_1を7回・v_2を2回 (X,Y)=(7,2)　灰 = 到達可能な g

同じ k の中では衝突しない（g = 11n + 43X で 43 は mod 100 で可逆）。異なる使用回数どうしの合流は k = 9 から始まり、k = 19 で 100 個すべてが埋まる。

解法C：ブロックDP ⑤ 先読みの作り方と結果

```
for b0 = 0, C, 2C, ... :
  # ① 割引つき重みを更新
  W[x] = √τ_x · exp(−λ(τ_x − b0)/N²)

  # ② 報酬表（L×100 回の probe）
  rew[k][g] = val( p((b0+k+1) mod 100, g) )

  # ③ DP（幅100 × 遷移3）
  dp[k+1][g] = rew[k][g] +
      max( dp[k][g], dp[k][g−α], dp[k][g−β] )

  # ④ 復元して先頭 C 手だけ確定
  #     （receding horizon）
```

報酬表 rew はブロック先頭の盤面で1回だけ作る＝「窓の中で自分が塗った影響」を無視した近似。DPはこの近似目的に対しては厳密だが、先を読むほど実際とズレる。だから **L=52 手読んでも、確定するのは先頭 C=3 手だけ**にして毎回作り直す。

ローカルで調整を行い、以下のパラメータを採用。

$\vec{w} = (89, 29),\ \vec{u} = (6, 89)$

$\alpha = 54,\ \beta = 11,\ L = 52,\ C = 3,\ \lambda = 6.966$

$\vec{v}_0 = (89, 29),\ \vec{v}_1 = (13, 35),\ \vec{v}_2 = (55, 8)$

結果

> 150ケースでのスコア合計：**782,279,028**

塗ったマスは 10000 中 7,938 マスだけ。

塗らない約2000マスは「すでに怪異が済んだマス」なので、塗る価値がゼロ。**全マス被覆は目的ではない**、というのが適応解の要点。

まとめ

3つの解法は、扱う探索空間の大きさも、それを扱う方法も大きく異なる。

解法	探索空間の大きさ	実際の探索方法	score（150ケース合計）
A：2ベクトル全探索	格子 (w, u)：2,880万通り	対称性で7,200通りに縮約し**全探索**	492,985,189
A+：Aの縮約部分を展開	格子 (w, u)：2,880万通り	Aに加え、Aの勝者1点の周辺4000通りを**全探索**	494,081,231
B：ペア解法(スコア全計算)	(A) × ビットマスク 2^{4950} 通り $\approx 10^{1490}$ 通り	格子は固定し、マスク空間を**山登り(焼きなまし)**で局所探索	675,004,378
C：ブロックDP	(A) × 行動列 3^{10000} 通り $\approx 10^{4771}$ 通り	$\langle\delta_1,\delta_2\rangle$の設計の工夫で状態を100個に圧縮し、**動的計画法**で多項式時間に	782,279,028

格子を決めるとAの行動列は完全に決まる（自由度なし）。Bは4,950ビットのマスクが、Cは10000手全体で1手ごとに自由な3択が上乗せされる（格子の2,880万倍という係数は3^{10000}の桁数の前では無視できるほど小さい）。scoreは本番150ケースでの合計（実測値）。

> 探索空間が桁違いに大きくなるが、**全探索→局所探索→動的計画法**とアルゴリズムを工夫することで実行時間内に収め、結果としてスコアも伸びている。