会津合宿　2017 Day3

# G: ほぼ無限グリコ

- 原案　　：杉江
- 問題文　：杉江
- 解答　　：杉江・栗田・鈴木
- 解説　　：杉江

# 問題概要

- N マスからなる円環状のフィールドがある
- グリコ（階段でやる遊び）の要領でじゃんけんを K 回行う
  - 特殊なじゃんけんを用いてプレイするため、勝ち方は M 通り
  - i 番目の勝ち方で勝った場合は $p_i$ マス進み、負けた場合はその場にとどまる
- K 回のじゃんけんを終えたあとに、i 番目のマスにいる場合の数 mod 1,000,000,007 を、それぞれのマスについて求める

- $1 \leq N \leq 8 \times 10^2$
- $1 \leq M \leq 10^4$
- $1 \leq K \leq 10^9$
- $1 \leq p_i \leq 10^9$

# 想定誤解法

- 想定誤解法は、ちょっと効率の悪い DP
- dp[i][j] := i 番目のじゃんけんを終えたあとに、 j 番目のマスにいる場合の数
  - **遷移行列** D ( 任意のマス対 (u, v) に対して、 u 番目のマスから 1 回のじゃんけんで v 番目のマスへ到達する場合の数 D[v][u] を行列で表したもの ) を作ると、任意のマス x に対して
    (dp[i+1][j] += D[j][x] * dp[i][x]) %= MOD と更新可能

| | |
|---|---|
| • i 回目のじゃんけん終了時に<br>• マス x にいる | • (i+1) 回目のじゃんけん終了時に<br>• マス j にいる |

# 遷移行列

- 遷移行列の作り方 → 任意のマス対（u, v）に対して、 u 番目のマスから 1 回のじゃんけんで v 番目のマスへ到達する場合の数 `D[v][u]` を行列で表す

- 例：N = 3, 勝ち方（進めるマス数）= 1, 2, 5
  - 実はこれはまだ不十分

$$\begin{bmatrix} 0 & 2 & 1 \\ 1 & 0 & 2 \\ 2 & 1 & 0 \end{bmatrix}$$

# 遷移行列

- 遷移行列の作り方 → 任意のマス対（u, v）に対して、 u 番目のマスから 1 回のじゃんけんで v 番目のマスへ到達する場合の数 `D[v][u]` を行列で表す

- 例：N = 3, 勝ち方（進めるマス数）= 1, 2, 5
  - **「負けた時はその場にとどまる」を入れ忘れないように！**
  - 対角成分に +1

$$\begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \\ 2 & 1 & 1 \end{bmatrix}$$

# 想定誤解法

- 想定誤解法は、ちょっと効率の悪い DP
- `dp[i][j]` := `i` 番目のじゃんけんを終えたあとに、 `j` 番目のマスにいる場合の数
  - **遷移行列** D ( 任意のマス対 (u, v) に対して、 `u` 番目のマスから 1 回のじゃんけんで `v` 番目のマスへ到達する場合の数 `D[v][u]` を行列で表したもの ) を作ると、任意のマス `x` に対して `(dp[i+1][j] += D[j][x] * dp[i][x]) %= MOD` と更新可能
  - 直前の情報だけわかっていれば良いので最初の次元は配列の使い回しで削減でき、空間計算量的には O(N) なので問題ない
  - しかし、時間計算量は $O(N^2 K)$ であり、到底間に合わないため、別のアプローチが必要

# 方針 1

- さっきの DP の更新式をもう一度見てみよう
- `(dp[i+1][j] += D[j][x] * dp[i][x]) %= MOD`
- 細かいことを省略して数式で書き直すとこうなる

$$dp'_j = \sum_{x=1}^{N} D_{j,x} \times dp_x$$

- これを高速化するには、なにが使えるか？
  - **行列**を使ってみよう！

# 方針 1

- DP 配列を縦ベクトルとみなすと、先ほどの式

$$dp'_j = \sum_{x=1}^{N} D_{j,x} \times dp_x$$

- これは行列の積で表せる！

$$\mathbf{dp}' = \mathbf{D} \times \mathbf{dp}$$

$$\mathbf{dp} = [dp_1, dp_2, \ldots, dp_N]^T$$

# 方針 1

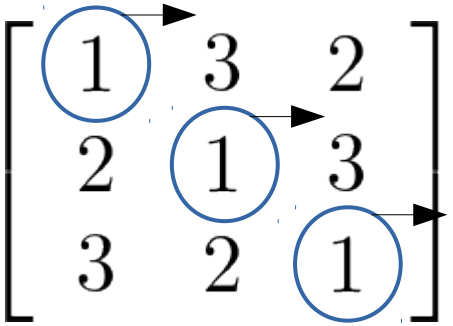
- ベクトル dp に対して、行列 D を 1 回掛けると次の状態（じゃんけんをもう 1 回した後の場合の数）に行くことがわかった
- K 回のじゃんけんを行ったあとの場合の数を知りたいのだから、D を K 回掛けたい（D を K 乗したい）気持ちになる
- これは、**行列累乗**で高速化できる！（詳細：蟻本 P.180）
- この方針をとることで **$O(N^3 \log K)$** になる

# 方針 1

- ベクトル dp に対して、行列 D を 1 回掛けると次の状態（じゃんけんをもう 1 回した後の場合の数）に行くことがわかった
- K 回のじゃんけんを行ったあとの場合の数を知りたいのだから、D を K 回掛けたい（D を K 乗したい）気持ちになる
- これは、**行列累乗**で高速化できる！（詳細：蟻本 P.180）
- この方針をとることで **$O(N^3 \log K)$** になる
  - **実はこれでもまだダメ！！**（最悪 $1.5 \times 10^{10}$ 回くらいの計算）

# 方針 2

- 遷移行列 D がどんな形の行列なのかを考えてみる
- D の例




- どのマスにいたとしても「w 番目の勝ち方をすると $p_w$ 進む」のだから、D の各行について、1 つ前の行の要素を 1 つずつずらしたものと同じになる
- このような行列を**巡回行列**という

# 方針 2

- 巡回行列の性質 → 任意の 2 つの巡回行列 A と B について、その**積 AB も巡回行列**になる
- この性質により、 dp ベクトルに掛けられる行列は常に巡回行列であることがわかる
- 巡回行列の積は、最初の行だけ普通に計算して 2 行目以降は前の行をずらして埋めることで $O(N^2)$ で計算が可能
- よって、 **$O(N^2 \log K)$** に落とすことができる！
- これが想定解法です

# Writer 解 & 統計

- **Writer 解**
  - 杉江 : 1994 bytes, 91 lines (C++)
  - 栗田 : 898 bytes, 41 lines (C++)
  - 鈴木 : 1213 bytes, 49 lines (C++)
- **Accept / Submission**
  - ?? % (?? / ??)
- **First Acceptance**
  - On-site : ???? (?? min)
  - On-line : ???? (?? min)