数え上げテクニック Hard

tsutaj (@_TTJR_)

Hokkaido University M2

Oct 28, 2019

1. はじめに

2. 問題解説
   - 実は単純な組み合わせになる問題
   - 挿入 DP
   - $\mathbb{F}_2$ 上の線形代数
   - Hall の結婚定理
   - フック長の公式

3. 類題

はじめに

- 前の活動で扱った「写像 12 相」は、数え上げ問題において重要な概念
  - 1 問目で軽く復習します
- それ以外にもテクニックはいろいろ
  - DP、包除原理、母関数など・・・
- 今回は数え上げに関するテクニックの一部を扱います

123 Pairs

123 Pairs (CODE FESTIVAL 2016 Grand Final J) Link

1 以上 $2N$ 以下の整数が 1 つずつある。これらを、以下の条件を全て満たすように N 個のペアに分ける方法は何通りあるか？

- 1 以上 $2N$ 以下の整数はそれぞれちょうど 1 つのペアに含まれる
- 差が 1 であるようなペアがちょうど A 個ある
- 差が 2 であるようなペアがちょうど B 個ある
- 差が 3 であるようなペアがちょうど C 個ある

制約

- $1 \leq N \leq 5,000$
- $0 \leq A, B, C$
- $A + B + C = N$

123 Pairs

DP だとどうやっても間に合わなさそうなので、方針を変えてみよう
とりあえず、ありえるペア分けの方法を試してみる

- 隣り合う 2 個を、「差が 1 であるペア」にする
  - 例: $(7, 8)$
- 隣り合う 6 個を、3 つの「差が 3 であるペア」にする
  - 例: $(1, 4)$, $(2, 5)$, $(3, 6)$
- 隣り合う $4 + 2x$ 個を、「差 2 のペア」、「差 3 のペア」、「差 3 のペア」、$\cdots$ 、「差 3 のペア」、「差 2 のペア」にする
  - 例: $(1, 3)$, $(2, 5)$, $(4, 7)$, $(6, 9)$, $(8, 11)$, $(10, 12)$
- 隣り合う 4 個を、「差 3 のペア」、「差 1 のペア」にする
  - 例: $(3, 6)$, $(4, 5)$

書いてみるとわかりますが、パターンはこれしかありません

123 Pairs

以下、「差 x のペア」を $\boxed{x}$ と表すことにし、値の小さい数字から順に、ペアの差がどのようになるか列で表現

- たとえば $\boxed{1}\boxed{2}\boxed{3}\boxed{3}\boxed{2}$ は、$(1,2), (3,5), (4,7), (6,9), (8,10)$ を表す

この問題は以下のように言い換えられる

問題の言い換え

以下の条件を全て満たすような N 個の四角の列は何通りあるか？

- $\boxed{1}$ がちょうど A 個ある
- $\boxed{2}$ がちょうど B 個ある
- $\boxed{3}$ がちょうど C 個ある
- 列は $\boxed{1}$ or $\boxed{3}\boxed{3}\boxed{3}$ or $\boxed{2}\boxed{3}\boxed{3}\cdots\boxed{2}$ or $\boxed{3}\boxed{1}$ の連結

123 Pairs

考えやすくしよう (以下のようにおく)

- $\boxed{1}$ が p セット
- $\boxed{3}\ \boxed{3}\ \boxed{3}$ が q セット
- $\boxed{2}\ \boxed{3}\ \boxed{3}\ \cdots\ \boxed{3}\ \boxed{2}$ が r セット
- $\boxed{3}\ \boxed{1}$ が s セット

整理すると

- $A = p + s$
- $B = 2r$
- $C = 3q + s + x$ (x は $\boxed{2}$ と $\boxed{2}$ の間にある個数)

ここからどうする？　→ いくつかの変数を固定しよう！

123 Pairs

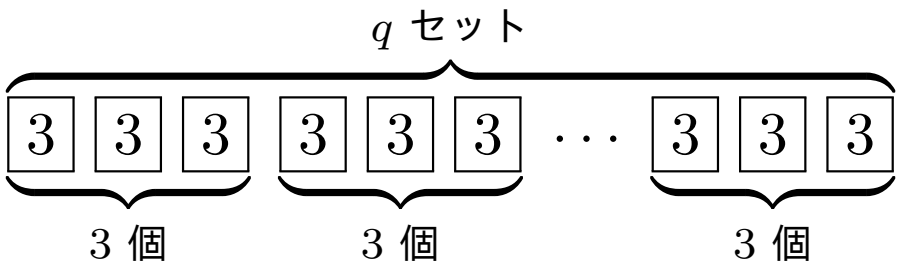
q と s を固定してみる

- $A = p + s$
- $B = 2r$
- $C = 3q + s + x$ (x は $\boxed{2}$ と $\boxed{2}$ の間にある個数)

固定すると他の変数もすべて決まる！ あとは通り数を求めればよい
どうやったら求められるか考えてみよう

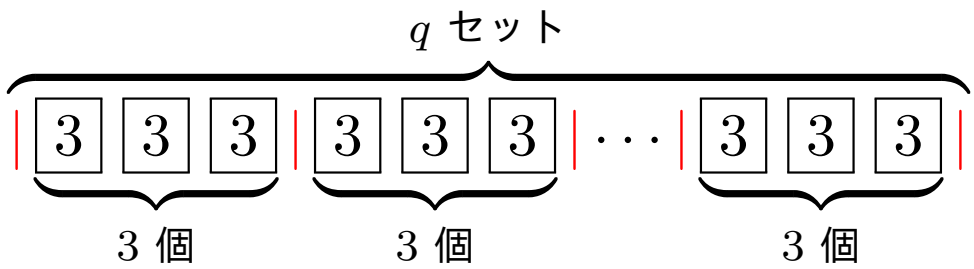
123 Pairs

まず、$\boxed{3}\,\boxed{3}\,\boxed{3}$ を q セット配置



123 Pairs

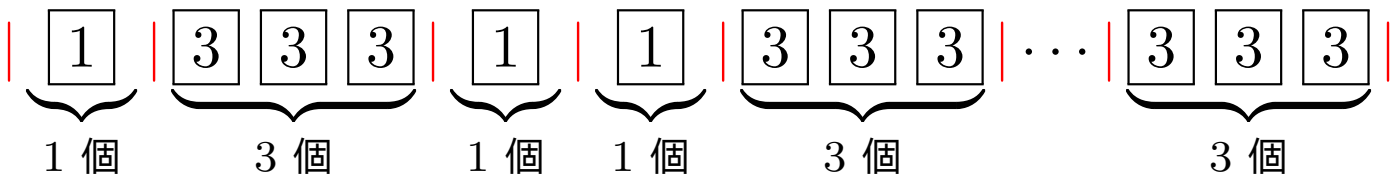
次に、$\boxed{1}$ を p セット配置



入れられるのは赤線部 (重複組み合わせ、${}_{q+1}H_p$ 通り)

123 Pairs

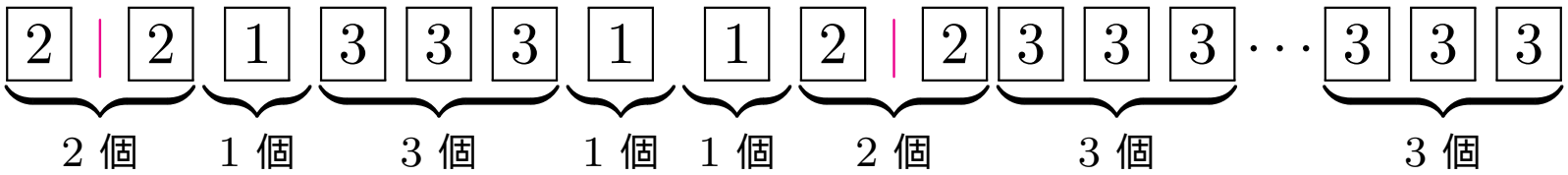
次に、$\boxed{3}\,\boxed{1}$ を s セット配置



入れられるのは赤線部 (重複組み合わせ、${}_{p+q+1}H_s$ 通り)

123 Pairs

- 次に、$\boxed{2}\ \boxed{3}\ \boxed{3}\ \cdots\ \boxed{3}\ \boxed{2}$ を r セット配置
  - ${}_{p+q+s+1}H_r$ 通り)
- そして、残っている $\boxed{3}$ を $\boxed{2}$ と $\boxed{2}$ の間にいれる
  - 紫線部に入れる、${}_rH_x$ 通り



123 Pairs

組み合わせの計算は、事前に階乗・階乗の逆元を $O(N)$ で計算することで
それぞれ $O(1)$ で可能
よって、以下の通りに解ける

まとめ

- 手順
  - q と r をある値に固定する: $O(N^2)$
  - 組み合わせを計算: $O(1)$
  - これをすべての q, r に対して行う
- 計算量は $O(N^2)$ となるので、間に合う！

123 Pairs

実装例 ▸ Link

```cpp
void init() {
    fact[0] = 1;
    for(int i=1; i<MAXN; i++) fact[i] = fact[i-1] * i % MOD;
    finv[MAXN-1] = mod_pow(fact[MAXN-1], MOD-2);
    for(int i=MAXN-2; i>=0; i--) finv[i] = finv[i+1] * (i+1) % MOD;
}

ll nCr(ll n, ll r) {
    return (n < r or r < 0) ? 0 : fact[n] * finv[r] % MOD * finv[n-r] % MOD;
}
ll nHr(ll n, ll r) {
    return (n == 0 and r == 0) ? 1 : nCr(n+r-1, r);
}

int main() {
    init();
    int N, A, B, C; cin >> N >> A >> B >> C;
    if(B % 2) { puts("0"); return 0; }

    ll ans = 0;
    for(int q=0; 3*q<=C; q++) {
        for(int s=0; 3*q+s<=C and s<=A; s++) {
            int x = C - 3*q - s, p = A - s, r = B / 2;

            ll add = 1;
            (add *= nHr(q+1, p)) %= MOD;
            (add *= nHr(p+q+1, s)) %= MOD;
            (add *= nHr(p+q+s+1, r)) %= MOD;
            (add *= nHr(r, x)) %= MOD;
            (ans += add) %= MOD;
        }
    }
    cout << ans << endl;
    return 0;
}
```

Sequence Growing Hard

Sequence Growing Hard (AGC024 E) ▸ Link

次の条件を満たす数列の組 $(A_0, A_1, \ldots, A_N)$ としてあり得るものの個数を M で割ったあまりを求めよ。

- 全ての i に対し、A_i は 1 以上 K 以下の整数からなる長さ i の数列
- 全ての i に対し、A_{i-1} は A_i の部分列
- 全ての i に対し、A_i は辞書順で A_{i-1} より大きい

制約

- $1 \leq N, K \leq 300$
- $2 \leq M \leq 10^9$

Sequence Growing Hard

- 空文字列に対して、1 文字ずつ挿入していくことを考える
- 文字列 s に文字 x を挿入できる条件は？
  - 便宜上、どの文字よりも小さいもの ϵ が文字列の末尾に含まれているものとする (つまり、はじめの文字列は “ϵ”)

$$s = \boxed{\text{hokkaido } \epsilon}$$

Sequence Growing Hard

文字 x を追加できる条件は以下の 2 通りある

- 追加する位置の直後の文字が、x より小さい

$$s = \boxed{\text{hokkai}}\ \boxed{\text{k}}\ \boxed{\text{do } \epsilon}$$

- 追加する位置の直後の文字が x と等しく、その後に登場する x でない文字のうち最も近いものが x より小さい

$$s = \boxed{\text{hok}}\ \boxed{\text{k}}\ \boxed{\text{kaido } \epsilon}$$

Sequence Growing Hard

ところで、以下の 2 つは生成される列が同じ

$$s = \boxed{\text{hok}}\ \boxed{\text{k}}\ \boxed{\text{kaido } \epsilon}$$

$$s = \boxed{\text{hokk}}\ \boxed{\text{k}}\ \boxed{\text{aido } \epsilon}$$

これを考えると実は 1 つめのルールだけ考えれば OK で、直後の文字が真に小さいときのみ追加を許せば被りも抜けもなく状態を考慮できる

Sequence Growing Hard

ここまでの議論から、問題を言い換える

問題の言い換え

以下の条件を全て満たすような $N+1$ 頂点からなる根付き木は何通り？

- 頂点は 1 から $N+1$ の番号付き
- それぞれの頂点には文字が書かれており、親の番号および文字は常に自身より小さい
  - 頂点 u の親が頂点 v である $\leftrightarrow$ u 番目に追加した文字は、頂点 v に書かれた文字の直前に挿入した
- 根に書かれた文字は ϵ

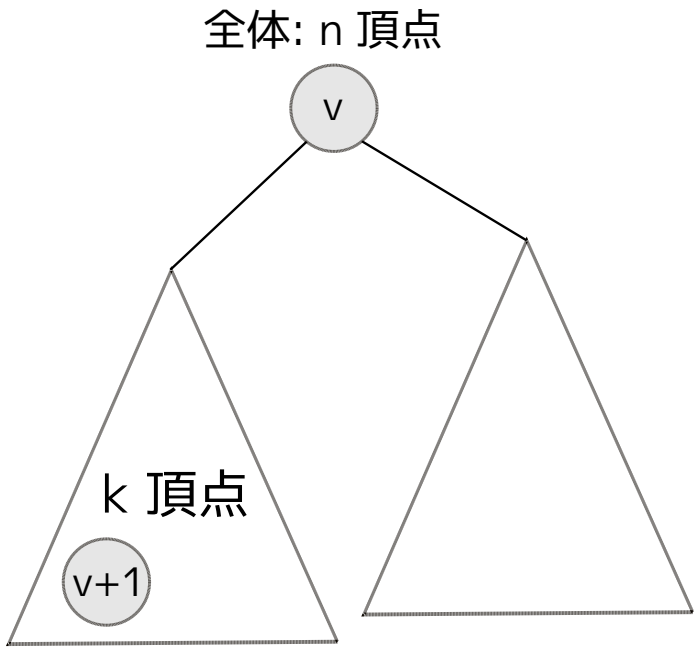
- 小さいものに挿入が起こって大きなものになるので、木構造による表現が効果的

どのように解けるか？

Sequence Growing Hard

木 DP をしよう

- $\mathrm{dp}[n][x] :=$ 根に文字 x を書くとき、n 頂点の木を作る通り数
- 根の頂点番号 $+1$ の頂点が含まれる部分木のサイズを k と決め打ちして数える
  - $\mathrm{dp}[n][x] = \sum_{k=1}^{n-1} \sum_{d>x} \mathrm{dp}[k][d] \times \mathrm{dp}[n-k][x] \times {}_{n-2}C_{k-1}$
  - ${}_{n-2}C_{k-1}$ は、下図の $v, v+1$ 以外の頂点のラベリングの通り数
  - このままだと 4 乗だが、累積和を取ると $O(N^2K)$ になる



Sequence Growing Hard

実装例 ▸ Link

```
ll solve(ll n, ll x) {
    if(x > K) return 0;
    if(dp[n][x] != -1) return dp[n][x];
    if(n == 1) return dp[n][x] = K-x+1;

    ll ans = 0;
    for(int k=1; k<n; k++) {
        ll v1 = solve(k, x+1);
        ll v2 = solve(n-k, x);
        ll v3 = solve(n-k, x+1);

        ll m1 = v1, m2 = (v2 - v3 + MOD) % MOD, m3 = nCr(n-2, k-1);
        (ans += m1 * m2 % MOD * m3 % MOD) %= MOD;
    }
    return dp[n][x] = (ans + solve(n, x+1)) % MOD;
}

int main() {
    cin >> N >> K >> MOD; init();
    fill(dp[0], dp[N+2], -1);
    cout << (solve(N+1, 0) - solve(N+1, 1) + MOD) % MOD << endl;
    return 0;
}
```

$\mathbb{F}_2$ 上の線形代数

次に説明する問題のために最低限説明します

有限体 (finite field)

有限個の元からなる体 (四則演算が定義されて閉じている有限集合)

- $\mathbb{F}_2 = \{0, 1\}$ は $0, 1$ のみで構成された体のことで、競技プログラミングではよく出てくる
- $\mathrm{mod} 2$ で加法と乗法が定められている、とも読み替えられる
- 加法は xor、乗法は and、とも読み替えられる

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

Table: $\mathbb{F}_2$ 上での加法と乗法

Odd Subrectangles

Odd Subrectangles

- 要素が $0, 1$ のみからなる $N \times M$ の行列 $A = \{a_{i,j}\}$ が与えられる
- 行の部分集合 X と列の部分集合 Y の組 2^{N+M} 通りのうち、以下の条件を満たすものの個数を 998244353 で割った余りを求めよ
  - X に属する行と Y に属する列の交わりに属する $|X||Y|$ 個のマスに書かれた整数の総和が奇数である

制約

- $1 \leq N, M \leq 300$
- $0 \leq a_{i,j} \leq 1$

ヒント: 「総和が奇数」を少し言い換えよう

Odd Subrectangles

いくつか言い換えをしよう

- 「総和が奇数」$\leftrightarrow$「該当要素どうしの xor が 1」
  - 先程述べたように、和は実質 xor
- 行を並べ替えてもよい
  - 集合なので行の順序は関係ない
- x 行目を y 行目に加算してもよい
  - xor の重要な性質「逆元が自分自身」: 自身と xor を取ると 0 になる
  - x 行目を y 行目に加算した状態、つまり c を列インデックスとして $a'_{y,c} = a_{x,c} \oplus a_{y,c}$ とした場合を考える
    - x 行目だけ選択する状態は $a_{x,c}$ (操作後の x 行目を選択)
    - y 行目だけ選択する状態は $a_{x,c} \oplus a'_{y,c}$ (操作後の x, y 行目を選択)
    - x, y 行目両方選択する状態は $a'_{y,c}$ (操作後の y 行目を選択)
  - それぞれ以上のように対応が取れるので行加算しても問題ない

Odd Subrectangles

- 行を並び替えてもよい
- 行加算してもよい
- 言い換えれば、「行列を行基本変形してもよい」！
  - 通常、基本変形には「行を定数倍」や「他の行にある行を定数倍したものを加算」という操作もありますが、$\mathbb{F}_2$ の場合はこれだけです
- なので、与えられた行列を行基本変形してみよう
- 同様の理由で列基本変形もできるので、それもしよう

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \xrightarrow{\text{行基本変形}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \xrightarrow{\text{列基本変形}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

操作後は「左上に単位行列があり、他は全て 0」という形になる

Odd Subrectangles

- 変形すると、すべての要素が 0 である行・列が発生するときがある
- このような行および列は、選んでも選ばなくても xor が変わらない！
- 単位行列のサイズを r とするとき、$2^{N-r} \times 2^{M-r}$ 通りは自由

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

- 次に、単位行列の箇所に関して考える

Odd Subrectangles

- 選ぶ行を適当に固定する
  - x 行選ぶ (赤色)
  - 選んだ行のインデックス集合を R とする

$$x \text{ 行選ぶ} \left\{ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \right.$$

- 列のインデックス c について、以下のように選ぶと総和が奇数になる
  1. $c \in R$ から奇数個選ぶ
  2. $c \notin R$ から任意個選ぶ

Odd Subrectangles

- 選ぶ行を適当に固定する
  - x 行選ぶ (赤色)
  - 選んだ行のインデックス集合を R とする

$$x \text{ 行選ぶ} \left\{ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \right. \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- 列のインデックス c について、以下のように選ぶと総和が奇数になる
  1. $c \in R$ から奇数個選ぶ (緑色)
  2. $c \notin R$ から任意個選ぶ (黄色)

Odd Subrectangles

計算してみよう (全体の行列サイズが $N \times M$、単位行列のサイズが r)

- x 行選ぶ通り数: ${}_rC_x$
- x 列から奇数個選ぶ通り数: 2^{x-1}
  - 「偶数個選ぶ」通り数と「奇数個選ぶ通り数」が等しいことを利用
- $r-x$ 列から任意個選ぶ通り数: 2^{r-x}
- 自由に選択できる行と列の通り数: $2^{N-r} \times 2^{M-r}$

$$
\begin{aligned}
\text{答え} &= \sum_{x=1}^{r} {}_rC_x 2^{r-1} 2^{N+M-2r} \\
&= (2^r - 1) 2^{N+M-r-1} \\
&= 2^{N+M-1} - 2^{N+M-r-1}
\end{aligned}
$$

- 単位行列のサイズ r とは、元の行列の rank に等しい
- ガウスの消去法などを用いて求められる (参考: Link)

Odd Subrectangles

実装例 ▸ Link

```cpp
using ll = long long int;
const ll MOD = 998244353;
ll mod_pow(ll n, ll k) {
    ll res = 1;
    for(; k>0; k>>=1) {
        if(k & 1) (res *= n) %= MOD;
        (n *= n) %= MOD;
    }
    return res;
}

int main() {
    int N, M; cin >> N >> M;
    BinaryMatrix mat(N, M);
    for(int i=0; i<N; i++) {
        for(int j=0; j<M; j++) {
            int val; cin >> val;
            mat[i][j] = val;
        }
    }

    int rank = gaussianEliminationBinary(mat);
    int v1 = mod_pow(2, N + M - 1);
    int v2 = mod_pow(2, N + M - rank - 1);
    cout << (v1 - v2 + MOD) % MOD << endl;
    return 0;
}
```

就職活動

就職活動 ▸ Link

- N 人の就職希望者にす社、ぬ社、け社それぞれに対して就職したいかアンケートを取った
  - 「誰がどこに就職したいかどうか」というパターンは 2^{3N} 通りある
- す社、ぬ社、け社にはそれぞれ最大で X 人、Y 人、Z 人までの人が入社できる
- 2^{3N} 通りのうち、全ての人が希望する就職先へ就職できるのは何通りかを求め、$10^9 + 7$ で割った余りを求めよ
  - どの会社も希望しない人は、希望する就職先へ就職できないものとする

制約

- $1 \leq X, Y, Z \leq N \leq 150$

Hall の結婚定理

この問題を解くための前提知識

Hall の結婚定理 ▸ Link

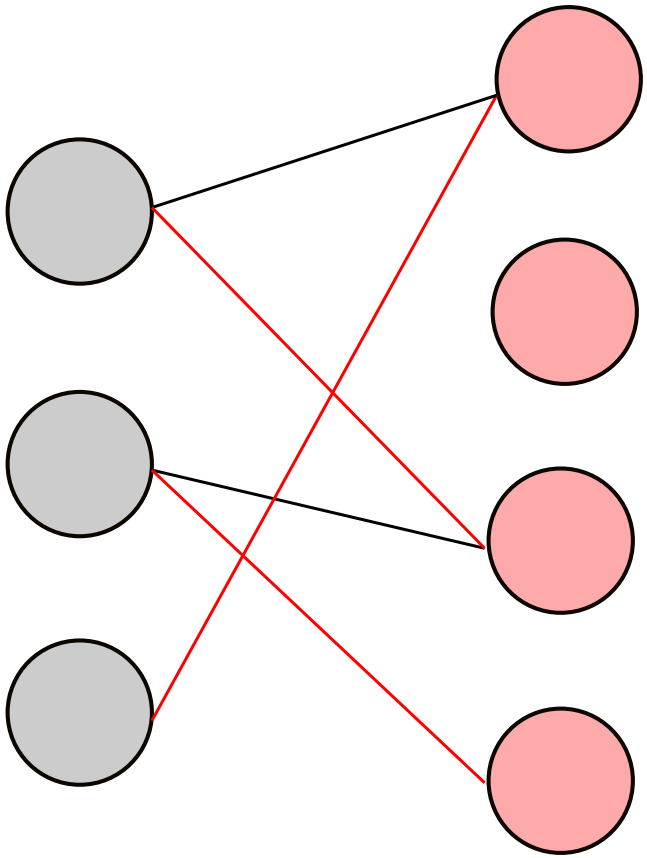
二部グラフ $G = (U + V, E)$ に対して、以下は同値である

1. U の頂点を全てカバーできるマッチングが存在
   - U の頂点を全てカバーするとは、U 内の任意の頂点 u がマッチングの端点として採用されていること
2. 任意の U の部分集合 A に対して $|A| \leq |\Gamma(A)|$
   - ただし、$\Gamma(A)$ は A と辺でつながっている頂点の集合

必要十分である、というのがポイント！

Hall の結婚定理

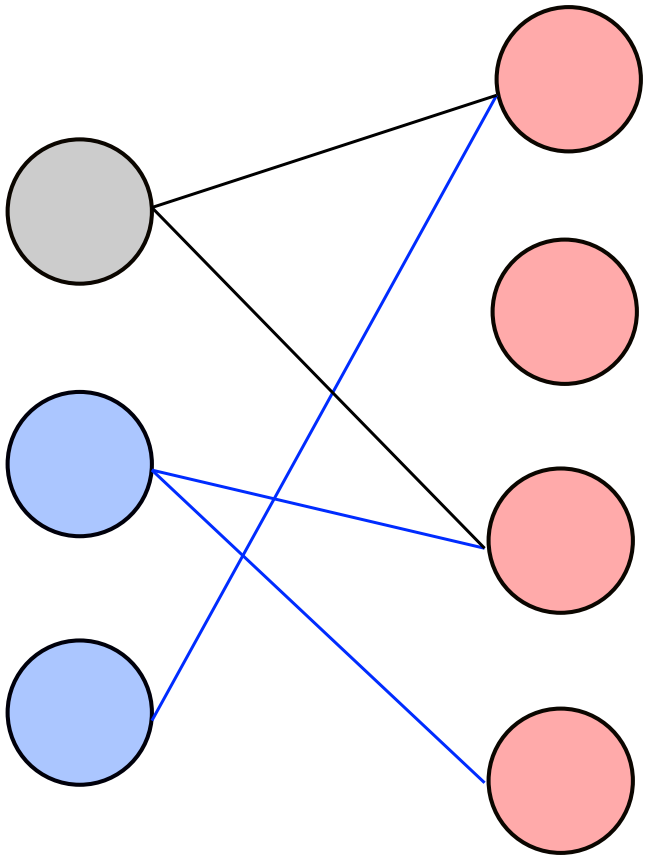
1. の気持ち



黒い頂点は U に、赤い頂点は V に含まれる頂点
黒い頂点はすべてマッチングの端点として採用されている

Hall の結婚定理

2. の気持ち



青い頂点数 (2) $\leq$ 青辺とつながっている赤い頂点数 (3)
このような大小関係が、全ての U の部分集合に対して成立

Hall の結婚定理

さらに、以下も成り立つことが分かる

Hall の結婚定理 (重み付き？)

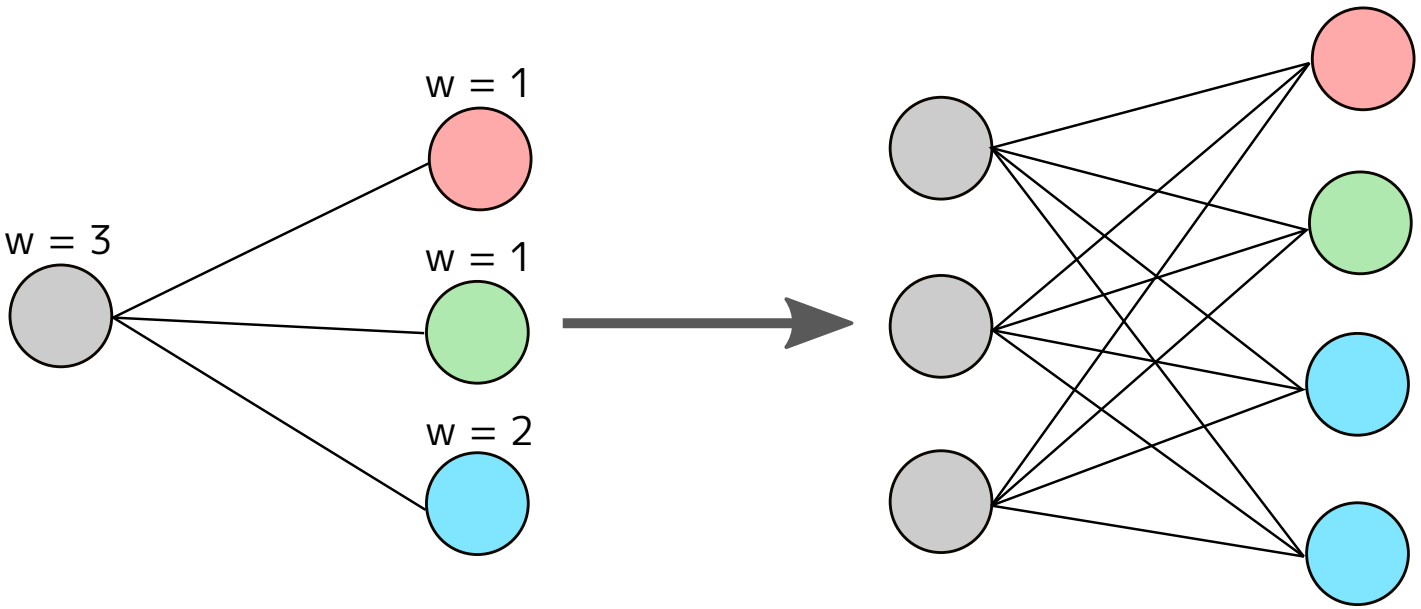
- 二部グラフ $G = (U + V, E)$ があり、U, V の各頂点 x には「その頂点を端点とする辺をいくつまで採用できるか」を表した量 $w(x)$ が定まっているものとする
  - 多重辺でもいい (流量っぽく考えてください)
- 以下は同値である

1. $\sum_{u \in U} w(u)$ 辺すべてを採用することができる
2. 全ての $A \subseteq U$ について、$\sum_{u \in A} w(u) \leq \sum_{v \in \Gamma(A)} w(v)$

Hall の結婚定理

以下のように U, V をバラせば、先程の結婚定理よりも少しきつい条件が成り立っていることが分かる

- バラした後の U 側の頂点において元々同じ頂点だったものは、最低でも 1 個選んでしまえば何個選んでも $\Gamma(A)$ が変わらない
- 全部選んだときが最も条件がきつく、この場合でも結婚定理の条件が成り立つというのが先程の主張



(自分の理解が間違っていたら教えてください)

就職活動

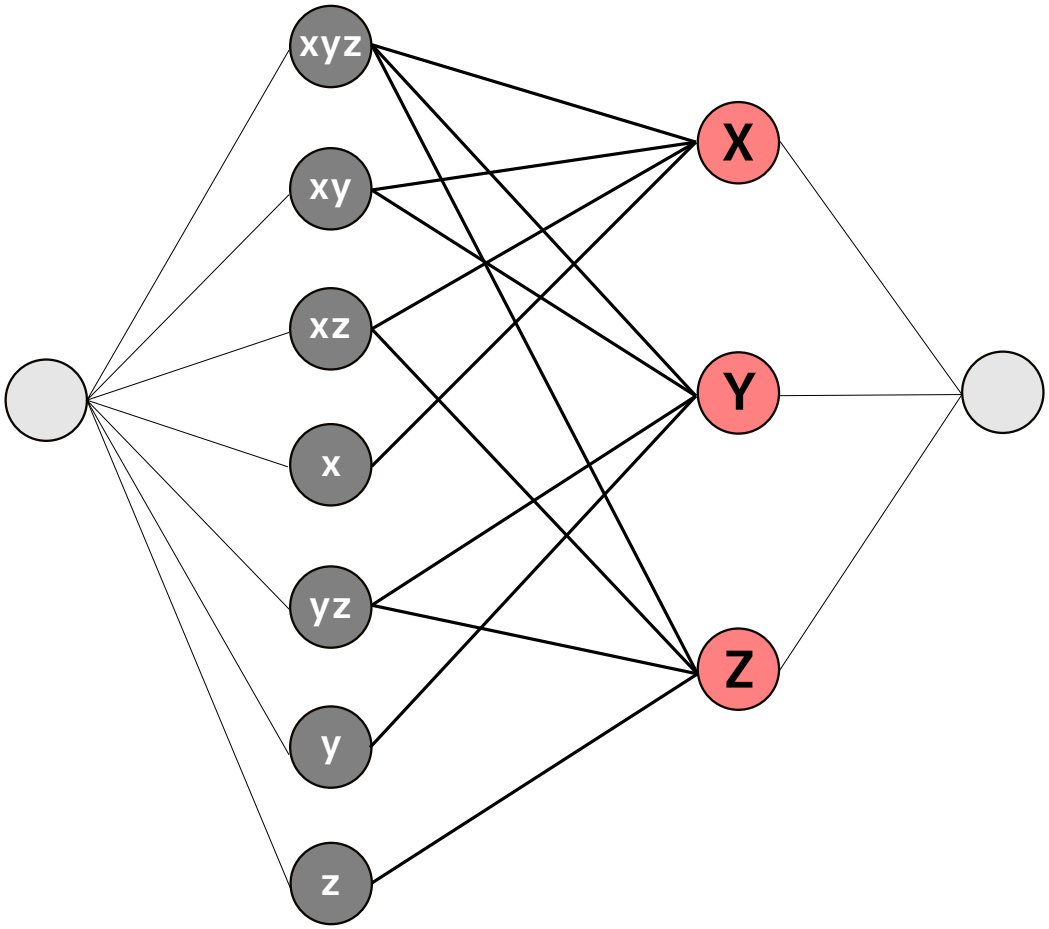
- とりあえず数え上げではなく Yes / No を考える
- 人をそれぞれの希望に応じて以下の 7 通りに分類
  - 各変数は、そのような希望を出した人数を表すものとする
- 全員が、希望する就職先へ就職できるか？
  - ただし、$x + y + z + xy + yz + xz + xyz = N$

す社	ぬ社	け社	変数名
○	○	○	xyz
○	○	×	xy
○	×	○	xz
○	×	×	x
×	○	○	yz
×	○	×	y
×	×	○	z

Table: 就職希望と変数の対応表

就職活動

各変数を固定して Yes / No を判定するなら、以下のようなグラフに流量 N 流れるかどうかを調べれば良い



しかし、全部の変数を固定していては間に合わない

就職活動

- 先程のグラフに Hall の定理を用いる
- 条件が厳しいところのみ抜き出すと以下のような式になる
  1. $x \leq X$
  2. $y \leq Y$
  3. $z \leq Z$
  4. $x + y + xy \leq X + Y$
  5. $y + z + yz \leq Y + Z$
  6. $x + z + xz \leq X + Z$
  7. $x + y + z + xy + yz + xz + xyz = N$

就職活動

- x, y, z, xy を固定して、定数とみなす
  1. $yz \leq Y + Z - y - z = c_1$
  2. $xz \leq X + Z - x - z = c_2$
  3. $yz + xz + xyz = N - x - y - z - xy = c_3$
- 残りの変数が取り得る値の範囲は c_1, c_2, c_3 という定数で表されることが分かる！

就職活動

- c_1, c_2, c_3 を固定したとき、条件を満たす yz, xz, xyz が何通り存在するかを計算しよう
- たぶん色々求め方はあります
  - yz, xz の条件が不等式ではなく等式であるとして、累積和を取る
  - 包除原理 (c_1 を超える、c_2 を超える、どちらも超える)
- 各 (c_1, c_2, c_3) について上記の通り数を前処理で求めておけば、x, y, z, xy を固定したときに $O(1)$ で値を持ってこれる
- よって、この問題は $O(N^4)$ で解ける！
  - 定数で割られるのでわりと速いです

実装例 (1/2) ▸ Link

```
ll calc(int a, int b, int c) {
    return fact[a+b+c] * finv[a] % MOD * finv[b] % MOD * finv[c] % MOD;
}

int main() {
    init();
    int N, X, Y, Z; cin >> N >> X >> Y >> Z;
    if(N > X + Y + Z) {
        puts("0");
        return 0;
    }

    // 前処理パート
    for(int c1=0; c1<=2*N; c1++) {
        for(int c2=0; c2<=2*N; c2++) {
            for(int c3=0; c3<=2*N; c3++) {
                if(c3 >= c1 + c2 and c3 <= N) {
                    int c0 = c3 - c1 - c2;
                    C[c1][c2][c3] += calc(c0, c1, c2);
                }
                if(c1 > 0) C[c1][c2][c3] += C[c1-1][c2][c3];
                if(c2 > 0) C[c1][c2][c3] += C[c1][c2-1][c3];
                if(c1 > 0 and c2 > 0) C[c1][c2][c3] -= C[c1-1][c2-1][c3];
                C[c1][c2][c3] %= MOD;
                if(C[c1][c2][c3] < 0) C[c1][c2][c3] += MOD;
            }
        }
    }
```

実装例 (2/2) ▸ Link

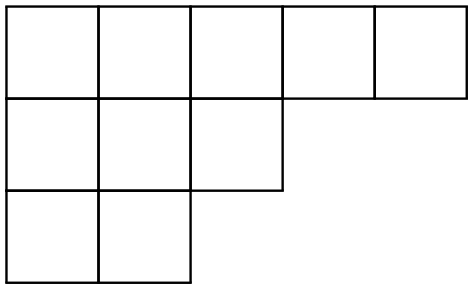
```
// 変数決め打ち
ll ans = 0;
for(int x=0; x<=X; x++) {
    for(int y=0; y<=Y; y++) {
        for(int z=0; z<=Z; z++) {
            for(int xy=0; x+y+xy<=X+Y; xy++) {
                int c1 = Y + Z - y - z;
                int c2 = X + Z - x - z;
                int c3 = N - x - y - z - xy;

                if(c3 < 0) continue;
                ll mul = C[c1][c2][c3], s = c3;
                (mul *= nHr(s+1, x)) %= MOD; s += x;
                (mul *= nHr(s+1, y)) %= MOD; s += y;
                (mul *= nHr(s+1, z)) %= MOD; s += z;
                (mul *= nHr(s+1, xy)) %= MOD;
                (ans += mul) %= MOD;
            }
        }
    }
}
cout << ans << endl;
return 0;
}
```

フック長の公式

ヤング図形 (Young diagram) Link

- 自然数 n の分割 $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_r)$
  - $\lambda_k \in \mathbb{N}, \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r, \sum_k \lambda_k = n$
- 分割 λ から、以下のルールで図を作れる
  - k 行目に λ_k 個の正方形を左詰めに並べる
  - それぞれの正方形を cell と呼び、i 行 j 列目の cell を (i, j) と書くことにする

例 (分割 $\lambda = (5, 3, 2)$ に対するヤング図形)

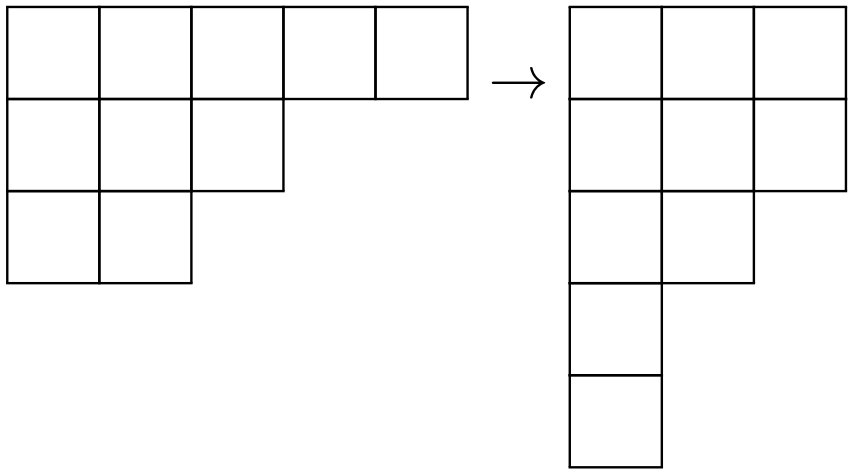


フック長の公式

ヤング図形 (Young diagram) Link

- 分割 λ に対するヤング図形の転置
  - 時計回りに 90 度回して、左右反転
  - 転置すると、λ' に対するヤング図形が得られる
- ヤング図形を知っていると、色々直感的にわかる
  - 例: 自然数 n を k 個の自然数に分割する通り数と、使用する整数の最大値が k である n の分割の通り数は等しい
  - ヤング図形の転置を使って一対一対応が取れてお互いに戻せる

例 (分割 $\lambda = (5, 3, 2)$ のヤング図形の転置)

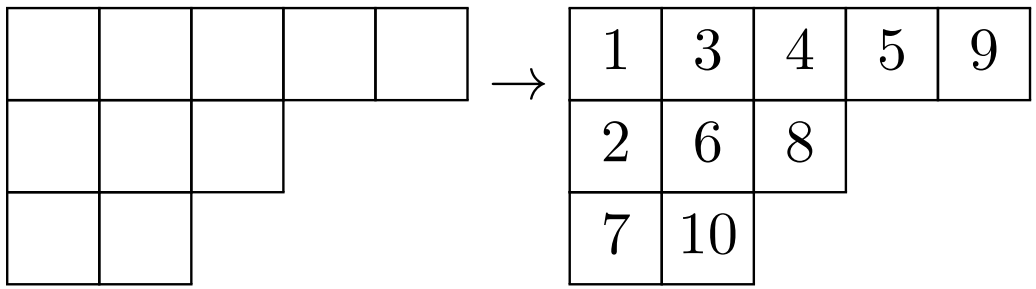


フック長の公式

標準ヤング盤 (standard Young tableau)

- λ を自然数 n の分割、Y_λ をそれに対応するヤング図形とする
- 以下の条件に従い、Y_λ の各 cell に 1 から n までの整数を 1 つずつ書き込んだ図形 T を標準ヤング盤という
  1. 各行は左から右に単調増加
  2. 各列は上から下に単調増加
- 標準ヤング盤のベースとなった λ を、標準ヤング盤の「形」という
- 形が λ の標準ヤング盤の総数を SYT_λ とおく

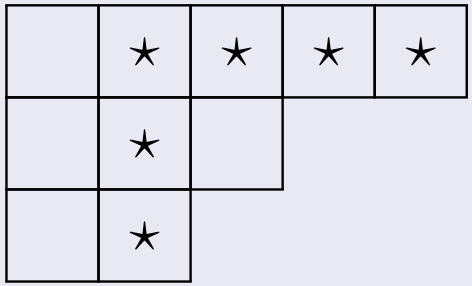
形が $\lambda = (5, 3, 2)$ である標準ヤング盤の例



フック長の公式

フック長 (hook length)

ヤング図形 Y_λ の cell $x = (i, j)$ に対するフック長 $h(x)$ は、
(下にある cell 数) + (右にある cell 数) + 1



フック長の公式 (hook length formula)

λ が n の分割であるとき、形が λ である標準ヤング盤の総数は、以下の式で表される

$$\mathrm{SYT}_\lambda = \frac{n!}{\prod_{x \in Y_\lambda} h_\lambda(x)}$$

フック長の公式

フック長の公式 (hook length formula)

λ が n の分割であるとき、形が λ である標準ヤング盤の総数は、以下の式で表される

$$\mathrm{SYT}_\lambda = \frac{n!}{\prod_{x \in Y_\lambda} h_\lambda(x)}$$

例 (形が $\lambda = (2,1)$ である標準ヤング盤)

$$\mathrm{SYT}_\lambda = \frac{3!}{1 \times 1 \times 3} = 2$$

実際に、以下の 2 種類がある

1	2
3	

,

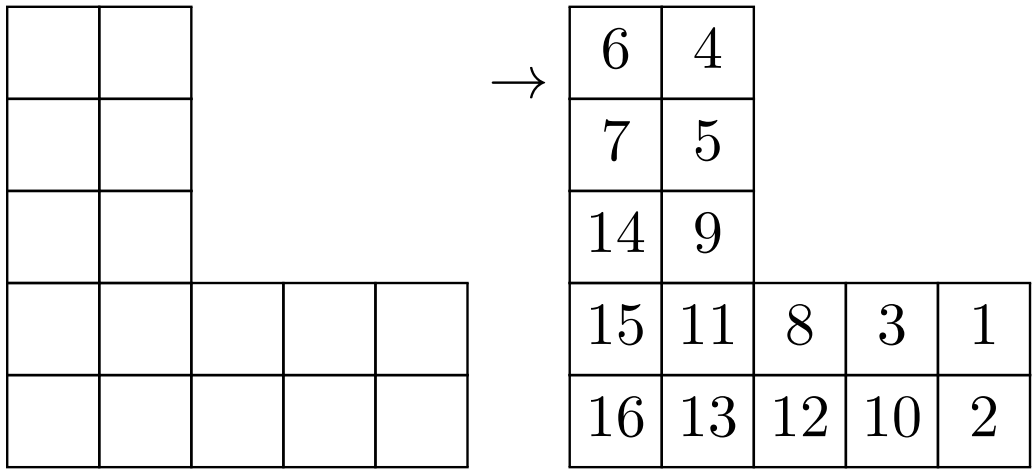
1	3
2	

Gnomon numbering

Gnomon numbering ▸ Link

- $n \times n$ のグリッドの右上 $m \times m$ 領域を除去してできるヤング図形を $L(n, m)$ とする
- 以下のルールにしたがって 1 から $n^2 - m^2$ までの整数を 1 つずつ cell に書く通り数は？
  - 各行は左から右に単調減少
  - 各列は上から下に単調増加

例: $L(5, 3)$



Gnomon numbering

- 転置っぽいことをして整数 x を $n^2 - m^2 + 1 - x$ で置き換えれば、標準ヤング盤になる
- 結局、フック長の公式を適用すればよい

6	4			
7	5			
14	9			
15	11	8	3	1
16	13	12	10	2

$\rightarrow$

16	15	14	7	6
13	11	9	5	4
12	8			
10	3			
2	1			

$\rightarrow$

1	2	3	10	11
4	6	8	12	13
5	9			
7	14			
15	16			

Gnomon numbering

実装例

```
using ll = long long int;
int main() {
    // 元の問題だと N = 10000, M = 5000 で固定
    int N, M; scanf("%d%d", &N, &M);
    int cells = N * N - M * M;

    ll ans = 1, div = 1;
    for(int i=1; i<=cells; i++) (ans *= i) %= MOD;

    for(int i=0; i<N; i++) {
        int C = (i < N - M ? N : N - M);
        for(int j=0; j<C; j++) {
            int R = (j < N - M ? N : N - M);
            int hr = R - 1 - i, hc = C - 1 - j;
            (div *= (hr + hc + 1)) %= MOD;
        }
    }

    (ans *= mod_pow(div, MOD-2)) %= MOD;
    printf("%lld\n", ans);
    return 0;
}
```

類題

練習にどうぞ (以下ネタバレ注意)

- 実は単純な組み合わせになる問題
  - 多重ループ (ABC021) Link
  - 11 (ABC066 / 600 点) Link
- 挿入 DP
  - 文字列 (TDPC) Link
  - Flexible Permutation (CPSCO2019 Session 3) Link
- $\mathbb{F}_2$ 上の線形代数
  - 数列 XOR (codeFlyer 本戦 / 600 点) Link
  - Xor Sum 3 (ABC141 / 600 点) Link
- Hall の結婚定理
  - Exhausted? (ARC076 / 1000 点) Link
  - Construction of a tree (AGC029 / 2200 点) Link
- フック長の公式
  - Maximin Game (KUPC 2019) Link