パソコン甲子園2025

全国高等学校パソコンコンクール
プログラミング部門 本選問題解説

THE UNIVERSITY OF AIZU

渡部 有隆　　鈴木 大郎　　奥山 祐市　　西舘 陽平　　松本 和也

問題セット

1 整数部分

2 三分割の不思議

3 薬の効果

4 平均値とK

5 ヒバラ海の2つの島

6 クッキー詰め職人のこだわり

7 ﾊｲﾊﾟｰｷｭｰﾌﾞﾃﾞｰﾀﾍﾞｰｽｼｽﾃﾑ

8 ヒバラ海に響く鐘の音

9 スパイ

10 クッキー詰め職人のこだわり

11 べこべこの攻略

12 荷物管理

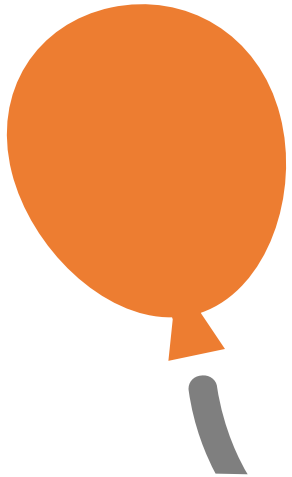
13 通学路のアクセス事情

問題セット

#	タイトル	分野	得点	難易度	
				思考	実装
1	**整数部分**	基礎	2	☆	★
2	**三分割の不思議**	基礎	2	★	☆
3	**薬の効果**	基礎	3	★	★
4	**平均値とK**	基礎	3	★☆	☆
5	**ヒバラ海の２つの島**	整列、幾何	4	★☆	★☆
6	**クッキー詰め職人のこだわり**	数え上げ（動的計画法）	6	★★	★☆
7	**ﾊｲﾊﾟｰｷｭｰﾌﾞﾃﾞｰﾀﾍﾞｰｽｼｽﾃﾑ**	累積和、包除原理	10	★★☆	★★★
8	**ヒバラ海に響く鐘の音**	計算幾何、二分探索	10	★★	★★☆
9	**スパイ**	グラフ	10	★★★	★★☆
10	**クッキー詰め職人のこだわり**	数え上げ（動的計画法、累積和、二分探索）	10	★★★	★★★
11	**べこべこの攻略**	数え上げ、組み合わせ数学	12	★★★☆	★★
12	**荷物管理**	シミュレーション	14	★★★☆	★★★☆
13	**通学路のアクセス事情**	データ構造	14	★★★★	★★★★

問題１　整数部分　（２点）

正答数:　32チーム



問題１　整数部分　(1/4)

概要

$N(1 \leq N \leq 300)$が与えられたとき

$\sum_{i=1}^{N} \lfloor \sqrt{i} \rfloor$ を求める

方針

- ループで解ける
  - math.hやcmathで使用できるfloor()関数やsqrt()関数を使う
  - $\sqrt{x}$ の計算を、2乗してxを超えない最大の整数を求める。
  - 値が正のとき、 floor関数はintへのキャスト(int)yとして計算できる
- Nに関する一般項を求めることでも解ける

問題１　整数部分　(2/4)

解答例１: floor/sqrtとループを使って解く

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n, res=0;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        res += floor(sqrt(i));
    }
    cout << res << '¥n’;
    return 0;
}
```

解答例２: 整数の計算とループを使って解く

```
#include<iostream>
using namespace std;
int main(){
    int n,res=0,sq=0;
    cin >> n;
    for (int k = 1 ; k <= n ; k++){
        if (k>=(sq+1)*(sq+1)) sq++;
        res += sq;
    }
    cout << res << "¥n";
    return 0;
}
```

問題１　整数部分　(3/4)

解法２：Nに対する一般項を求める

$a_k = \lfloor\sqrt{k}\rfloor$を考える。この数列は

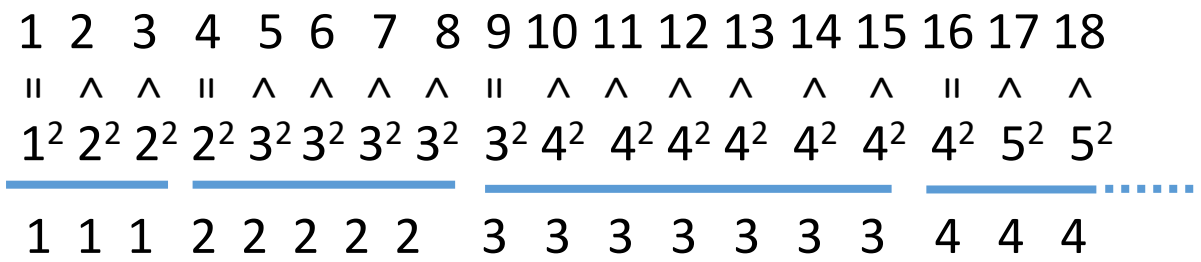
$1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, \cdots, i-1, i, i, i, \cdots, i, i+1, \cdots$

となる。

$\lfloor\sqrt{k}\rfloor = j$となる整数$k$は、$j^2 \le k \le (j+1)^2 - 1$となり、その個数は$C(j) = (j+1)^2 - j^2 = 2j + 1$個となる。また、$a_k = j$となる区間の$a_k$の和は$j \cdot C(j) = j(2j+1)$となる。さらに、$k$は$r\ (0 \le r \le 2j)$を用いて$k = j^2 + r \cdots$①と書くことができる。

ここで、与えられた正の整数をNとし、$M = \lfloor\sqrt{N}\rfloor$とする$a_n = i\ \ (i = 1, 2, \cdots, M-1)$となる列は、列をすべて総和に含めるため、それぞれ$i \cdot N(i)$として計算できる。

$a_n = M$となる列は、その要素数は$N(i)$個以下である。$a_n = j$となる列は、$k = j^2, j^2+1, j^2+2, \cdots, N$となる部分のみ加算を行う。　①より、この長さは、$r + 1 = N - M^2 + 1$となる。



問題１　整数部分　(4/4)

解法２続き

よって、

$$\sum_{k=1}^{N}\left\lfloor\sqrt{k}\right\rfloor=\sum_{j=0}^{M-1}j\cdot C(j)+M(N-M^2+1)$$

ここで

$$\sum_{j=1}^{M-1}j\cdot C(j)=\sum_{j=1}^{M-1}j\cdot(2j+1)=2\sum_{j=1}^{M-1}j^2+\sum_{j=1}^{M-1}j$$

$$=2\frac{(M-1)M(2M-1)}{6}+\frac{M(M-1)}{2}$$

$$=\frac{(M-1)M(4M+1)}{6}$$

ゆえに、

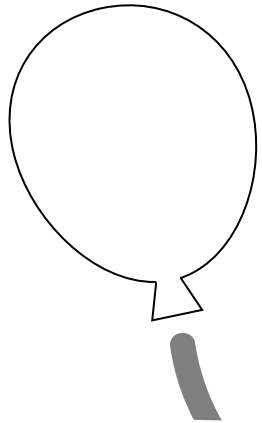
$$\sum_{k=1}^{N}\left\lfloor\sqrt{k}\right\rfloor=\frac{(M-1)M(4M+1)}{6}+M(N-M^2+1)$$

解法２による解答例

```
#include<iostream>
#include<cmath>
int main(){
  int n,m;
  std::cin >> n;
  m = (int)sqrt(n);
  std::cout << m*(n-m*m+1)+(m-1)*m*(4*m+1)/6
            << std::endl;
  return 0;
}
```

問題２　三分割の不思議（２点）

正答数:　　32チーム



問題２　三分割の不思議　(1/2)

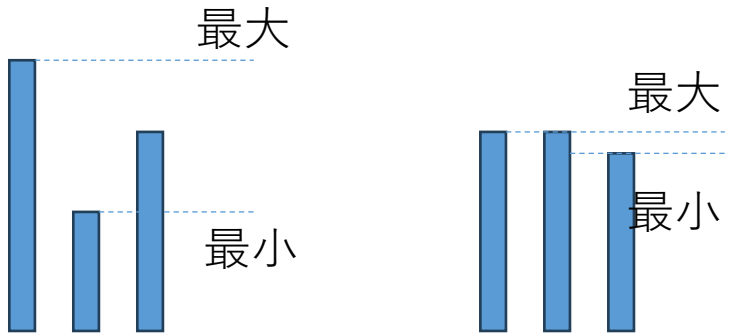
概要

Wが与えらえる。非負整数の組(a, b, c)を、$a + b + c = W$が成り立つように定める。そのようなすべての組で$\max(a, b, c) - \min(a, b, c)$の最小値を求める。

方針

最大値と最小値がなるべく同じになるように
a,b,cを定める

→a,b,cを均等に分ける



解法

Wを3で割った余りがないなら、最小値0
そうでなければ最小値1

解答例

```
#include<iostream>
using namespace std;

int main(){
  int w;
  cin >> w;
  cout << (w%3!=0) << endl;
  return 0;
}
```

問題２　三分割の不思議　(2/2)

解説：なぜ3で割った余りで答えが決まるのか？

最大・最小値とa, b, cは、

$$\min(a,b,c) \leq a,b,c \leq \max(a,b,c)$$

の関係にある。また、

$$\min(a,b,c) \leq \max(a,b,c)$$

より、

$$\max(a,b,c) - \min(a,b,c) \geq 0$$

ここで$S = \max(a,b,c) - \min(a,b,c) = 0$とすると、$\max(a,b,c) = \min(a,b,c)$となり、これを$t$とおく。

$$\min(a,b,c) \leq a,b,c \leq \max(a,b,c)$$

$$t \leq a,b,c \leq t$$

$$\therefore a = b = c = t$$

このとき、$W = 3t$となる。
つまり、$W = 3t$のとき、$a = b = c = t$とすれば、$S = 0$となる。

ここまでで、$W = 3t$の時のSの値を求めることができた。Wは整数となるため、残る$W = 3t + 1$のときと、$W = 3t + 2$の時の値を求めれば、すべて整数WについてSを求めたことになる。

$W = 3t + 1$ としたとき、 $a = t + 1,\ b = t,\ c = t$とすることができる。このとき、

$$S = \max(a,b,c) - \min(a,b,c) = t + 1 - t = 1$$

$W = 3t + 2$ としたとき、$a = t + 1,\ b = t + 1,\ c = t$とすることができる。このとき、

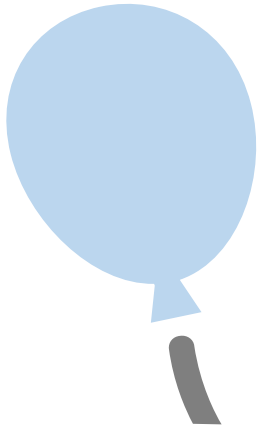
$$S = \max(a,b,c) - \min(a,b,c) = t + 1 - t = 1$$

よって

$$S = \begin{cases} 0 & W \bmod 3 = 0 \\ 1 & \text{それ以外のとき} \end{cases}$$

問題３　薬の効果　（３点）

正答数:　32チーム



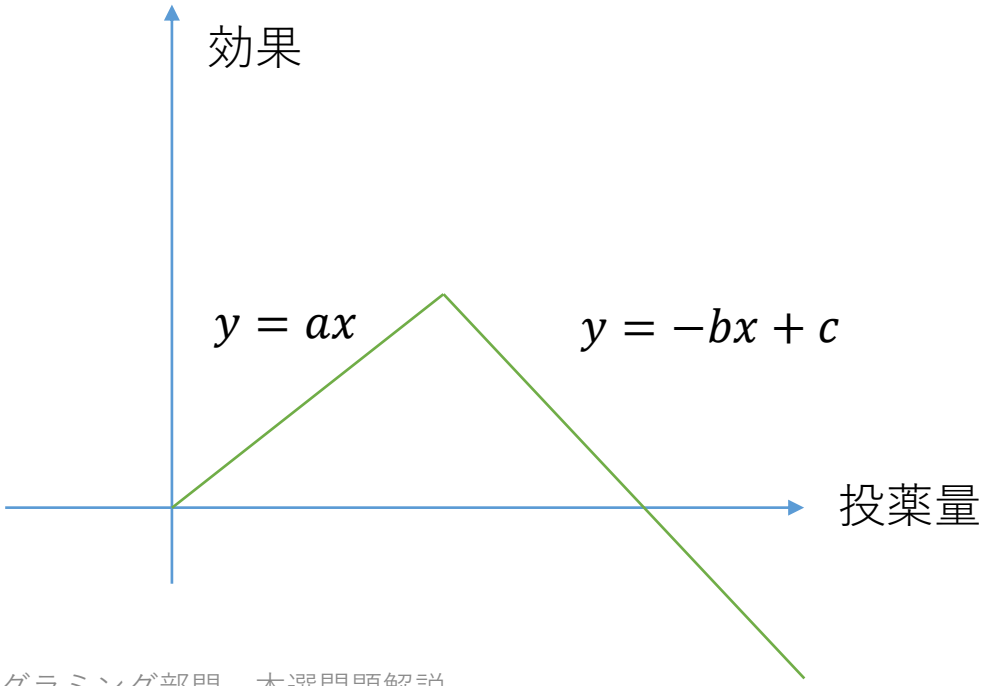
問題３　薬の効果

概要

- 薬の投薬量とその効果を表すグラフの情報と、投薬量のリストが与えられる。
- 薬の効果は、xを投薬量、yを効果とすると、図のように$y = ax$と$y = -bx + c$で表される。
- リストの投薬量から得られる最大の効果と、その効果が得られる投薬番号をすべて求めよ。

制約

- $2 \leq$ 投薬量の数$N \leq 10{,}000$
- $1 \leq$ 投薬量$d_i \leq 50{,}000$
- $1 \leq a \leq 100$
- $1 \leq b \leq 100$
- $1 \leq c \leq 50{,}000$



問題３　薬の効果

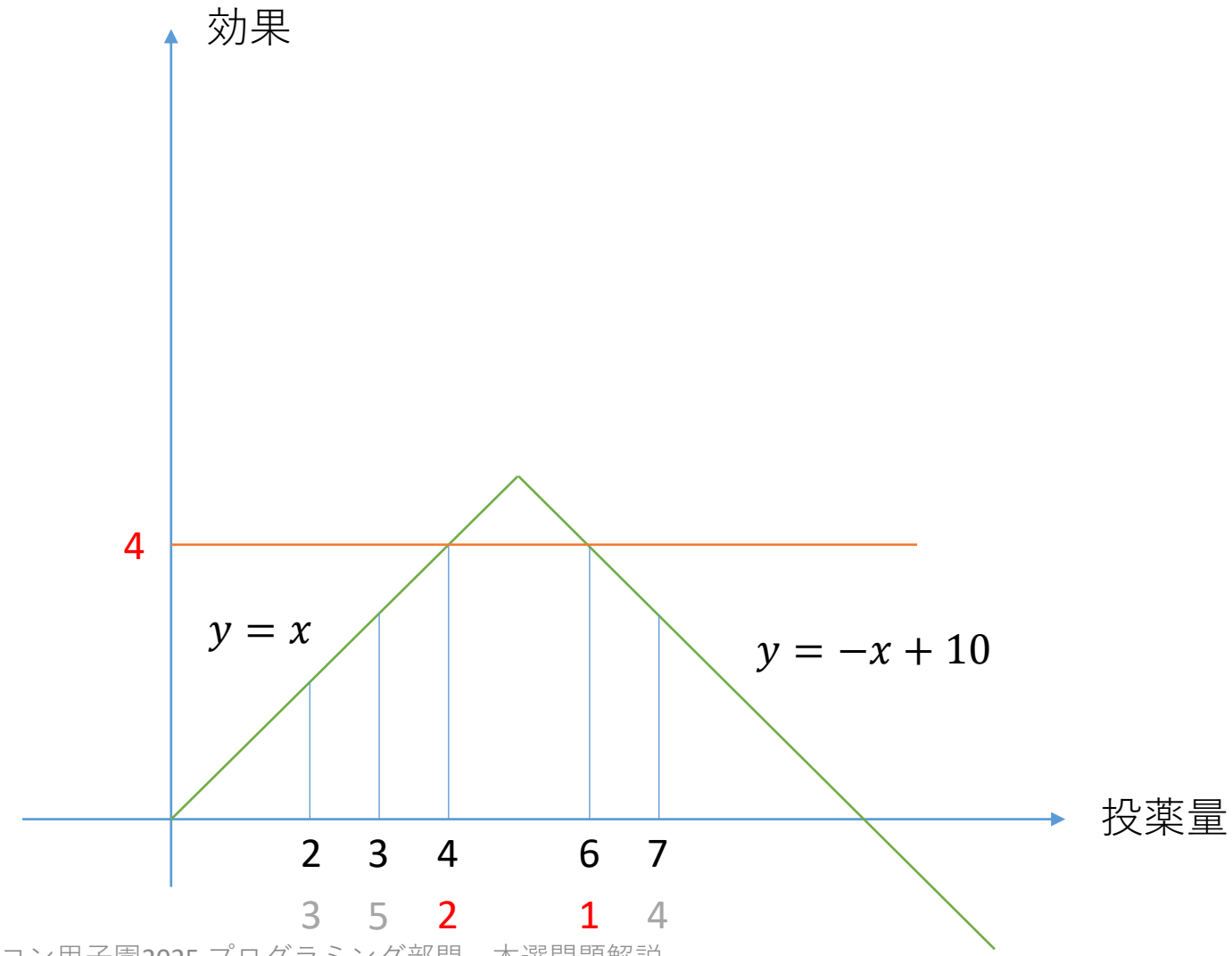
解法

入力例２

```
5
1  1  10
6  4  2  7  3
```

出力例２

```
4
1
2
```



問題３　薬の効果

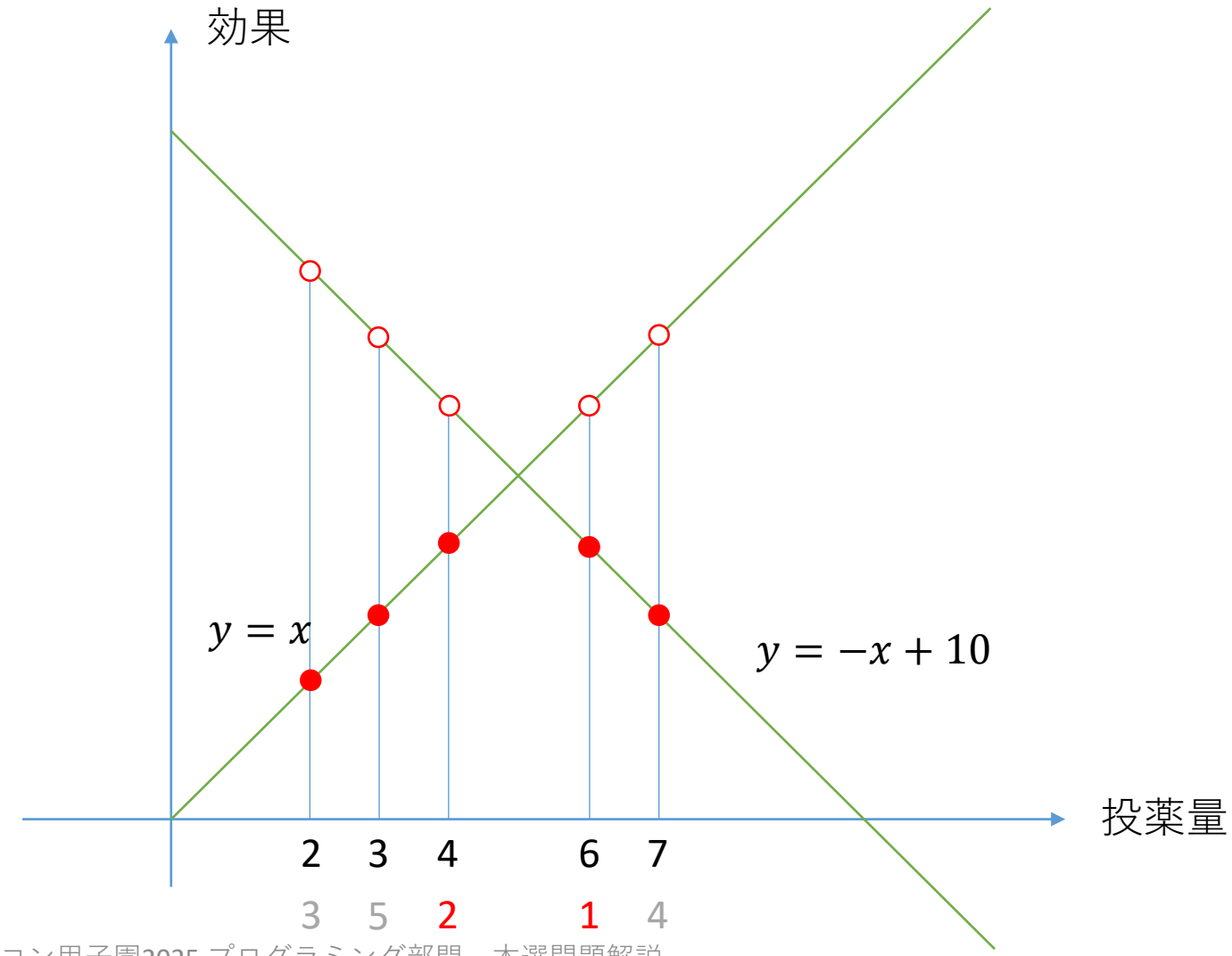
解法

入力例２

```
5
1  1  10
6  4  2  7  3
```

出力例２

```
4
1
2
```



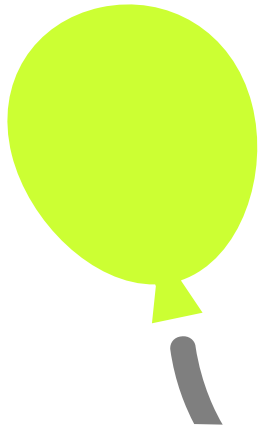
問題３　薬の効果

解答例（C++）

```
int effect(int a, int b, int c, int x){
  return min(a * x, -b * x + c);
}

int main(){
  int N, a, b, c, d[10000];
  cin >> N >> a >> b >> c;
  for ( int i = 0; i < N; i++ ) cin >> d[i];

  int maxv = -5000000;

  for ( int i = 0; i < N; i++ )
    maxv = max(maxv, effect(a, b, c, d[i]));

  cout << maxv << endl;
  for ( int i = 0; i < N; i++ )
    if ( effect(a, b, c, d[i]) == maxv) cout << i+1 << endl;

  return 0;
}
```

問題４　　平均値とK（３点）

正答数:　　31チーム



問題４　平均値とK

概要

- 入力として、４つの整数N, L, H, Kが与えられる。
- 各要素a_iが$L \leq a_i \leq H$を満たす整数からなる、長さNの数列$(a_1, a_2, \cdots, a_N)$を考える。
- この条件を満たすすべての数列Aの中で、「Aの平均値と与えられたKの差の絶対値」として取りうる値の最小値を求めたい。
- すなわち、$\left|\frac{a_1+a_2+\cdots+a_N}{N} - K\right| = \left|\frac{\sum_{i=1}^{N} a_i}{N} - K\right|$の最小値を求めたい。
  - なお、取りうる値の最小値は整数であることが証明できる。

問題４　平均値とK

解法

- $\left|\frac{a_1+a_2+\cdots+a_N}{N}-K\right|=\frac{|a_1+a_2+\cdots+a_N-NK|}{N}$である。
- 数列Aの総和$S=a_1+a_2+\cdots+a_N$は$NL\le S\le NH$を満たす任意の値を取ることができる。
- 差の絶対値として取りうる値の最小値はK,L,Hの大小関係を用いて、次の３つのケースのいずれかになる。
- $K<L$の場合
  - $NK<S$なので、できる限り小さいSを取ればよい。すなわち、$a_i=L$と取ればよく、
    求める最小値は$\frac{|NL-NK|}{N}=L-K$である。
- $H<K$の場合
  - $S<NK$なので、できる限り大きいSを取ればよい。すなわち、$a_i=H$と取ればよく、
    求める最小値は$\frac{|NH-NK|}{N}=K-H$である。
- $L\le K\le H$の場合
  - $a_i=K$と取れば$S=NK$を満たすようにできるので、求める最小値は$\frac{|NK-NK|}{N}=0$である。

問題４　平均値とK

解法（別解）

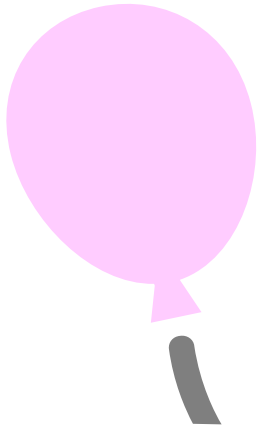
- C++では、

  abs(clamp(K, L, H) - K)

  と書くだけで答えが求まる。
- clamp(val, low, high) は値を範囲内に収める関数であり、val, low, highの大小関係に応じた戻り値を返す。
  - val の値が low より小さければ low を返す。
  - val の値が high より大きければ high を返す。
  - そうでなければ val を返す。
- よって、clamp(K, L, H) の戻り値とKの差の絶対値を取ることで、前述の3ケースの場合分けと同じ結果が得られ、答えが求まる。

問題５　ヒバラ海の２つの島（４点）

正答数:　　30チーム



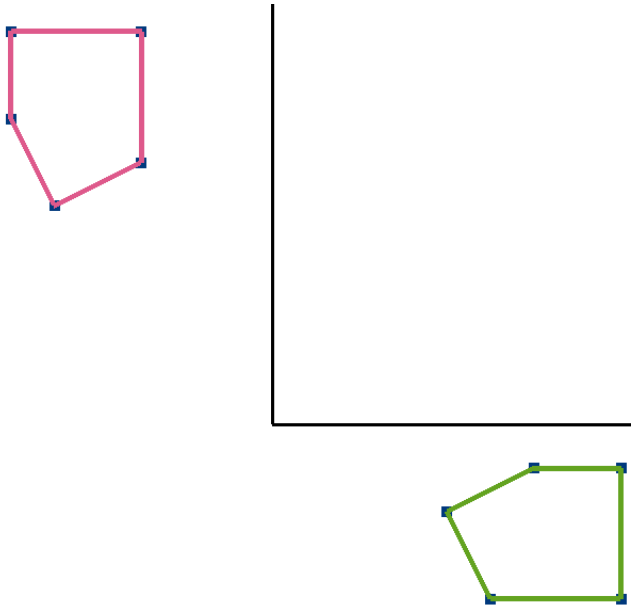
問題５　ヒバラ海の２つの島

概要

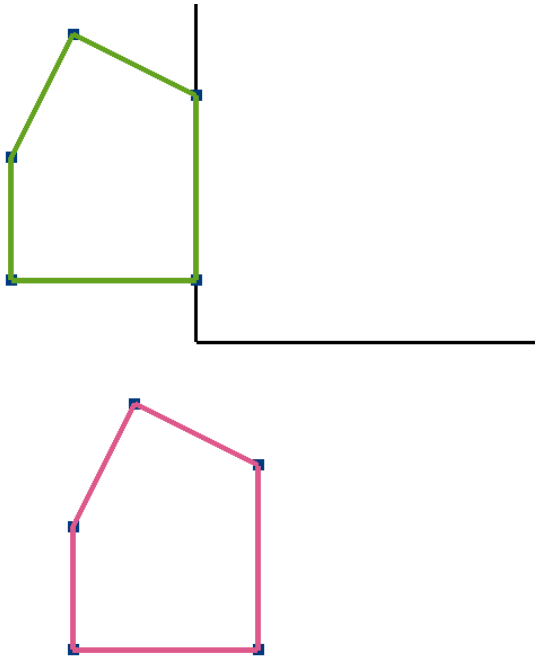
- １つ目の島を構成する頂点の数Nと、その島を構成する頂点のx, y座標が、１つ目の島の領域の重心の周りに反時計回りに整数で与えられる
- ２つ目の島を構成する頂点の数Mと、その島を構成する頂点のx, y座標が、２つ目の島の領域の重心の周りに反時計回りに整数で与えられる
- ２つの島はどちらも凸多角形（すべての内角が１８０°未満の多角形）である。
- ２つの島の形状が一致するかを判定する
  - 島の形状を変えずにx, y座標の平行移動を行ったときに、２つの島を構成するすべての点の座標が互いに一致するときに、島の形状が一致すると考える
- $1 \leq N, M \leq 200{,}000$
- 座標(x, y)の範囲は$-10^9 \leq x_i, y_i \leq 10^9$

問題５　ヒバラ海の２つの島

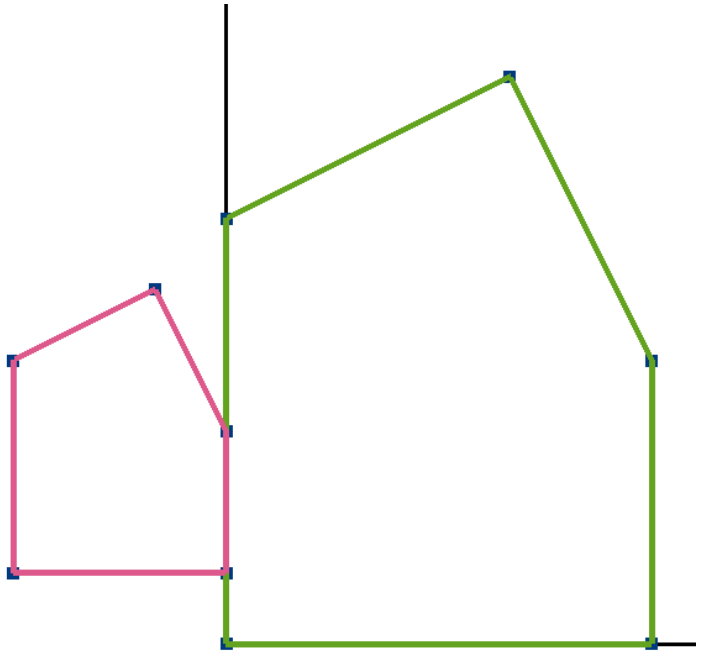
入出力例



回転＆鏡像＆平行移動で一致
→　No



平行移動で一致
→　Yes

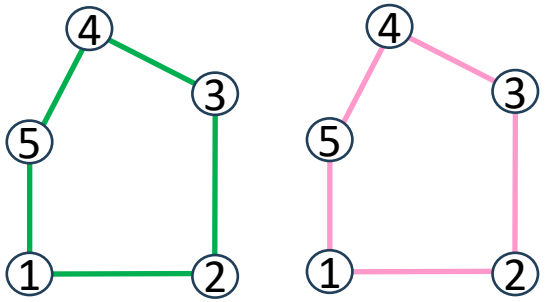


拡縮＆平行移動で一致
→　No

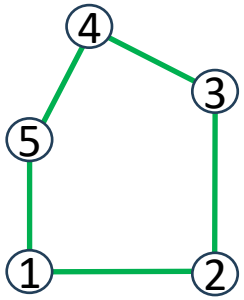
問題５　ヒバラ海の２つの島

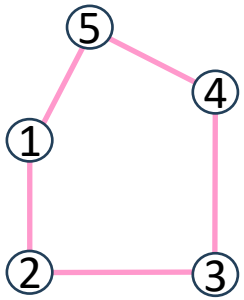
考察

- １つ目の島と２つ目の島の頂点番号が同じ順番（左下隅から始まる、等）で与えられるとは限らないことに注意する



平行移動で一致
→　Yes





平行移動で一致
→　Yes

問題５　ヒバラ海の２つの島

考察

- $N \leq 2 \times 10^5$、 $M \leq 2 \times 10^5$なので、 $\mathrm{O}(NM)$では時間切れ。
- 基準点の対応を正しく取る必要がある
- 基準点を原点とした座標に平行移動し、基準点から頂点の座標を比較していけば$\mathrm{O}(N)$

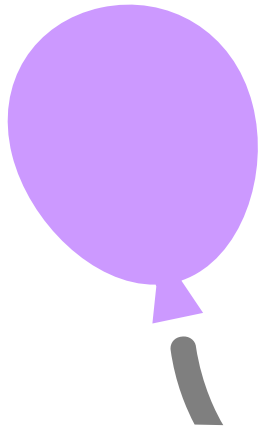
問題５　ヒバラ海の２つの島

解法

- $N \neq M$ならただちに「No」。
- $N = M$なら以下のようにする
  - １つ目の島を構成する頂点の座標を整列(x座標が小さい順で、 x座標が同じ場合はy 座標が小さい順)。整列後の頂点のi番目の座標を(a'_i, b'_i)とする。
  - ２つ目の島を構成する頂点の座標も同様に整列。整列後の頂点のi番目の座標を(c'_i, d'_i)とする。
  - $i = 2$ 番目の頂点からN番目の頂点までのどれかで　$a'_i - a'_1 \neq c'_i - c'_1$または$b'_i - b'_1 \neq d'_i - d'_1$ なら「No」
  - 最後まで「No」でなければ「Yes」

問題６　クッキー詰め職人のこだわり（６点）

正答数:　　23チーム



問題６　クッキー詰め職人のこだわり

概要

- N個のクッキーがベルトコンベアで運ばれてくる。クッキーの大きさはc_iで、すべて異なる。
- 運ばれてきたすべてのクッキーを順番に袋に詰める。
- 袋はいくつ使ってもよいが、新しい袋にクッキーを詰め始めたら、それまでにクッキーを詰めた袋にはそれ以上クッキーを詰められない。
- １つの袋にクッキーを詰めるときは、以下の条件を満たすようにしなければならない。
- 条件：１つの袋に入れる最初のクッキーが最も小さく、最後に入れるクッキーが最も大きくなるように袋詰めする。
- このようにしてクッキーを袋に詰めるとき、条件を満たすような袋詰めの方法の総数を$998{,}244{,}353$で割った余りを求める。

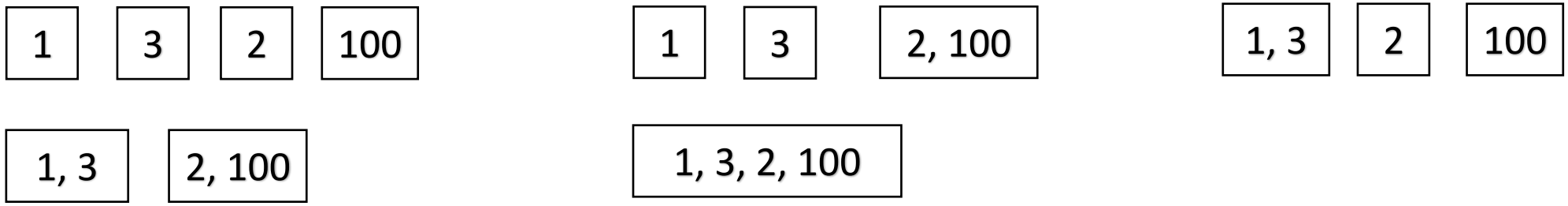
制約：$1 \leq N \leq 2{,}000,\ 1 \leq c_i \leq 1{,}000{,}000$

問題６　クッキー詰め職人のこだわり

入出力例

$N = 4$、$c_1 = 1$、$c_2 = 3$、$c_3 = 2$、$c_4 = 100$

- 条件を満たすような袋詰めの方法は、以下の５つ（四角形が袋を表す）。

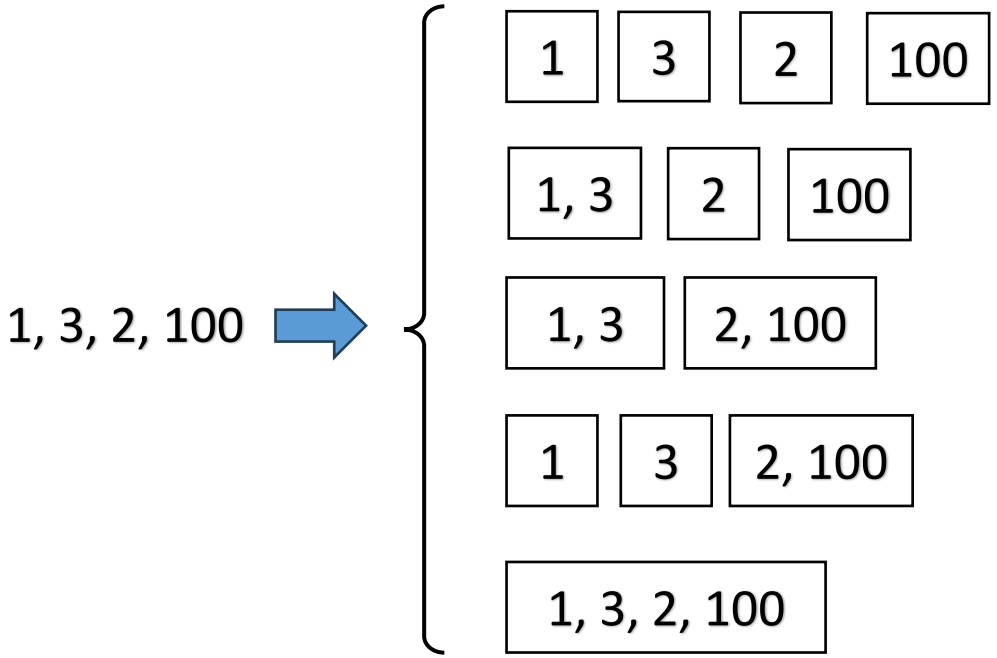


- 以下の３つは条件を満たさない。

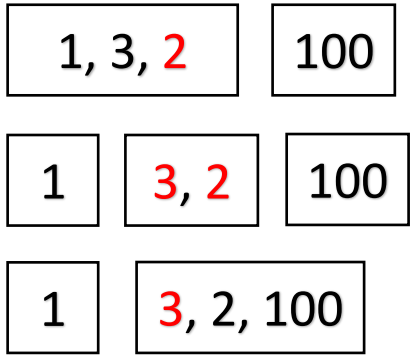
問題６　クッキー詰め職人のこだわり

考察

この問題は、与えられた数列を、先頭が最小で末尾が最大という条件を満たす部分列に分割する方法を求める問題。



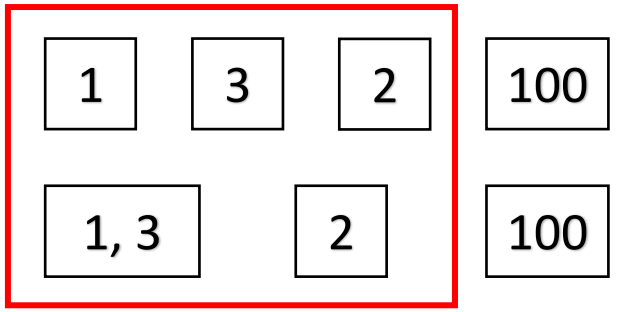
以下の３つは条件を満たさない。



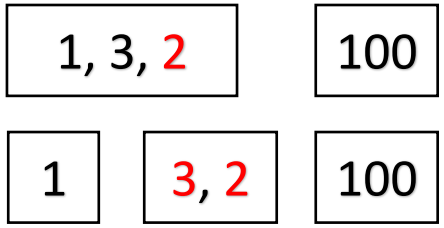
問題６　クッキー詰め職人のこだわり

考察

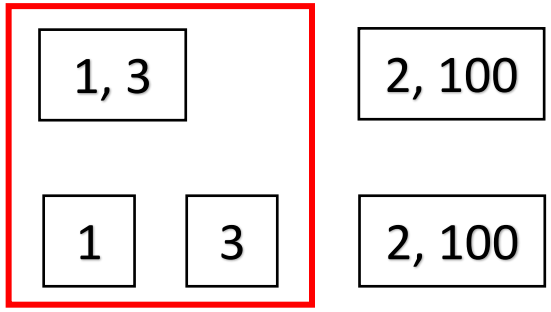
- ５つの分割のうち最初の２つは、数列の先頭の長さ３の部分列を分割した後に、残りの１要素からなる部分列を追加したもの。



以下の２つは条件を満たさない。



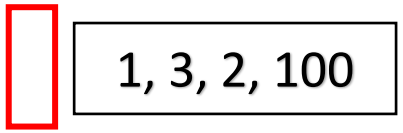
- 次の２つの分割は、数列の先頭の長さ２の部分列を分割した後に、残りの２要素からなる部分列を追加したもの。



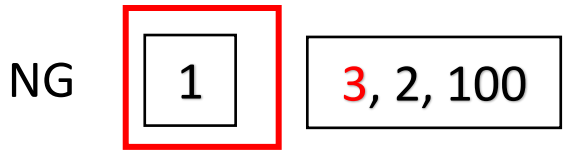
問題６　クッキー詰め職人のこだわり

考察

- 最後の分割は、数列の先頭の長さ０の部分列を分割した後に、残りの４要素からなる部分列を追加したもの。



- 数列の先頭の長さ１の部分列を分割した後に、残りの３要素からなる部分列を追加したものは条件を満たさない。



- 数列の分割の総数は、先頭の長さ０、長さ２、長さ４の部分列を分割する方法の総数１、２，２の和（１＋２＋２＝５）。

問題６　クッキー詰め職人のこだわり

解法

- 数列の先頭の長さi $(1 \leq i \leq N)$の部分列の分割方法の総数がわかれば、それらを使って長さNの数列を分割する方法の総数がわかる。
- これは１次元の動的計画法。数列の先頭の長さi の部分列の分割方法の総数を保存する表dpを用意する（ dp[0] = 1 とする）。
- 以下では、与えられた長さNの数列をA[] = { A[0], A[1], …, A[$N-1$] } とする。
  また、A[]の 部分列 { A[i], A[$i+1$], …, A[j] } をA[i; j]で表す。
- iを0から$N-1$まで増やしながら、以下のようにしてdp[[$i+1$]を計算する。
  - dp[$i+1$]の初期値を0とする。
  - jをiから0まで動かし、部分列A[j; i] が条件を満たすとき、以下を実行する。

    dp[$i+1$] = (dp[$i+1$] + dp[j]) % 998,244,353;
- dp[N]が求める答。

問題６　クッキー詰め職人のこだわり

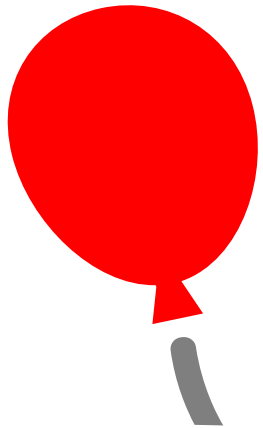
実装

```
vector<long long> dp(N+1);   dp[0] = 1;
for (int i=0; i<N; i++)  {        // dp[1]からdp[N]までを順に求める
    int minv = MaxVal + 1;    // minvをA[]の要素が取り得る最大値MaxValより大きな数で初期化
    dp[i+1] = 0;
    for (int j=i; j>=0; j--)  {    // A[j; i]の先頭が最小という条件判断の効率化のためにjをiから0まで動かす
      // A[j]からA[i]までの部分列A[j; i]が条件を満たせば、dp[i + 1]にdp[j]を加える
      if (A[j] > A[i]) break;   // 末尾A[i]が最大でなければ、任意のk ≤ jについてA[k; i]は条件を満たさない
      if (minv > A[j])   {      // A[j]がA[j; i]の最小値ならA[j; i]は条件を満たすので、dp[i + 1]にdp[j]を加える
          dp[i+1] = (dp[i+1] + dp[j]) % 998244353;
          minv = A[j];          // A[j; i]の最小値minvを更新
      }
    }
}
cout << dp[N] << endl;
```

計算量は$O(N^2)$

問題７　ﾊｲﾊﾟｰｷｭｰﾌﾞﾃﾞｰﾀﾍﾞｰｽｼｽﾃﾑ（10点）

正答数:　　10チーム



問題７　ハイパーキューブデータベースシステム

概要

- N個の属性値$(x_1, x_2, \ldots, x_N)$を持つキャラクターがいる。
- i番目の属性値x_iは1以上v_i以下の整数値を取る。
- 全ての属性値の組み合わせ$(x_1, x_2, \ldots, x_N)(1 \leq x_i \leq v_i)$について、その組み合わせを持つキャラクターの数$c_{x_1,x_2,\ldots,x_N}$が与えらえる。
- 以下のクエリをQ個処理せよ:
  - 各属性値iについて属性iの値がa_i以上b_i以下であるキャラクターの総数を出力せよ

制約

- $1 \leq N \leq 8$
- $1 \leq Q \leq 2 \times 10^5$
- $1 \leq v_i \leq 2 \times 10^5$
- $\left(\prod_{i=1}^{N} v_i = v_1 \times v_2 \times \cdots \times v_N\right) \leq 2 \times 10^5$
- $1 \leq c_{x_1,x_2,\ldots,x_N} \leq 10^9$
- $1 \leq a_i \leq b_i \leq v_i$

問題７　ハイパーキューブデータベースシステム

入出力例

入力例2	出力例2
2 2 3 1 2 3 4 5 6 4 1 2 2 3 2 2 1 1 1 2 1 3 2 2 1 3	16 4 21 15

$[a_1, b_1] = [1,2], [a_2, b_2] = [2,3]$

属性1＼属性2	1	2	3
1	1	2	3
2	4	5	6

→ 16

$[a_1, b_1] = [1,2], [a_2, b_2] = [1,3]$

属性1＼属性2	1	2	3
1	1	2	3
2	4	5	6

→ 21

$[a_1, b_1] = [2,2], [a_2, b_2] = [1,1]$

属性1＼属性2	1	2	3
1	1	2	3
2	4	5	6

→ 4

$[a_1, b_1] = [2,2], [a_2, b_2] = [1,3]$

属性1＼属性2	1	2	3
1	1	2	3
2	4	5	6

→ 15

問題７　ハイパーキューブデータベースシステム

解法

- $\prod v_i, Q \leq 2 \times 10^5$という制約から、条件を満たすキャラクターの数を愚直に求めていては時間制限に間に合わない

- $N = 1$なら？　　→　一次元累積和で解ける
- $N = 2$なら？　　→　二次元累積和で解ける

　　⋮

- $1 \leq N \leq 8$なら？　→　N次元累積和を実装しよう

問題７　ハイパーキューブデータベースシステム

実装方針

1. N次元配列をNに依存しないよう一次元配列で表現する
2. 包除原理を用いて領域和を高速に求める

問題７　ハイパーキューブデータベースシステム

実装

1. N次元配列をNに依存しないよう一次元配列で表現する

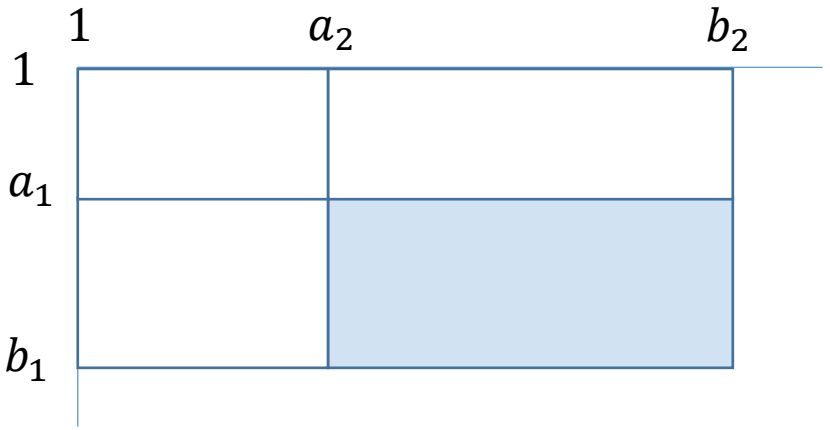
- $c[(x_1, x_2, \ldots, x_N)]$を$c[x_1][x_2][x_3] \ldots [x_N]$と表現としようとすると$N$の値に応じて実装を分ける必要がある
- $(x_1, x_2, \ldots, x_N) = x_1 \times \prod_{i=2}^{N} v_i + x_2 \times \prod_{i=3}^{N} v_i + \cdots + x_N$というようにひとつの整数値として表すと一次元配列で表現することができる

問題７　ハイパーキューブデータベースシステム

実装

2. 包除原理を用いて領域和を高速に求める

- I_iを区間（$[l,r]$や$[l,r)$など）を表すものとする
- $sum(I_1, I_2, \ldots I_N) = (x_i \in I_i)$を満たすすべての整数の組$(x_1, x_2, \ldots, x_N)$についての$c_{x_1,x_2,\ldots,x_N}$の和とする
- 包除原理より、$sum([a_1,b_1],[a_2,b_2]) = sum([1,b_1],[1,b_2]) - sum([1,a_1),[1,b_2]) - sum([1,b_1],[1,a_2)) + sum([1,a_1),[1,a_2))$である



問題７　ハイパーキューブデータベースシステム

実装

2.　包除原理を用いて領域和を高速に求める

- 一般化すると、求める値は

$$sum([a_1, b_1], [a_2, b_2], \dots, [a_N, b_N]) =$$
$$+sum([1, b_1], [1, b_2], \dots, [1, b_N])$$
$$-sum([1, a_1), [1, b_2], \dots, [1, b_N])$$
$$-sum([1, b_1], [1, a_2), \dots, [1, b_N])$$
$$+sum([1, a_1), [1, a_2), \dots, [1, b_N])$$
$$-\dots$$

となる

- $[1, a_i)$が偶数個現れるならば加算を、奇数個現れるならば減算をする

問題７　ハイパーキューブデータベースシステム

実装

- 事前にすべての整数の組$(x_1, x_2, \ldots, x_N)$について$sum([1, x_1], [1, x_2], \ldots, [1, x_N])$を包除原理を用いて求めておく$O(2^N N \prod v_i)$
- 包除原理を用いて各クエリに答える$O(2^N N)$

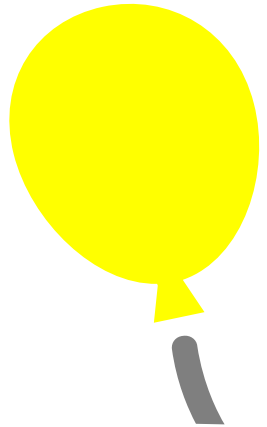
問題７　ハイパーキューブデータベースシステム

計算量

- 構築
  - $O(2^N N \prod v_i)$
  - 工夫すると$O(N \prod v_i)$
- クエリあたり
  - $O(2^N N)$
- 全体
  - $O\left(2^N N (Q + \prod v_i)\right)$

問題８　ヒバラ海に響く鐘の音（１０点）

正答数:　　9チーム



問題８　ヒバラ海に響く鐘の音

概要

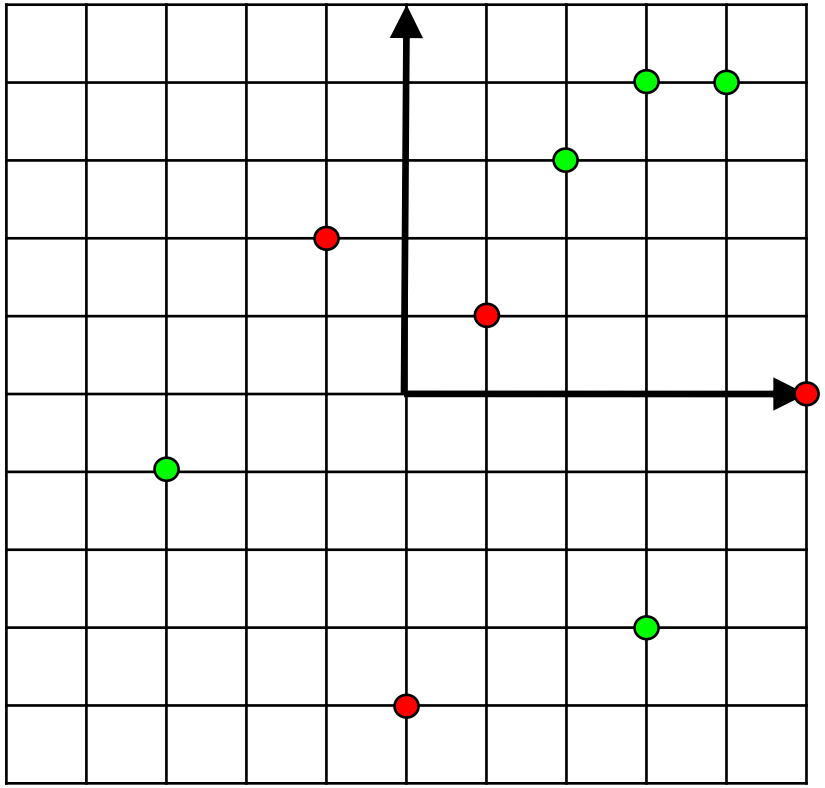
- イヅア組合に属するN個の鐘つき台の座標とイワシロ組合に属するM個の鐘つき台の座標が与えられる
  - 座標に重複は無い
- 鐘には不思議な性質があり、鐘の位置から見て東西と南北の同じ長さの範囲に鐘の音が届く
  - 鐘の位置を(x_1, y_1)とし鐘の音の大きさをDとすると、$|x - x_1| \leq D$かつ$|y - y_1| \leq D$を満たすような位置(x, y)に鐘の音が届く
- イヅア組合とイワシロ組合それぞれで、相手の鐘の音が聞こえる鐘つき台が少なくとも１つずつあるような、最小の鐘の音の大きさを求める
- $1 \leq N, M \leq 200{,}000$
- 座標(x, y)の範囲は$-10^9 \leq x_i, y_i \leq 10^9$
- 時間制限８秒

$|x - x_1| \leq D$かつ$|y - y_1| \leq D$　→　(x_1, y_1)を中心とした縦横$2D$の正方形の範囲内

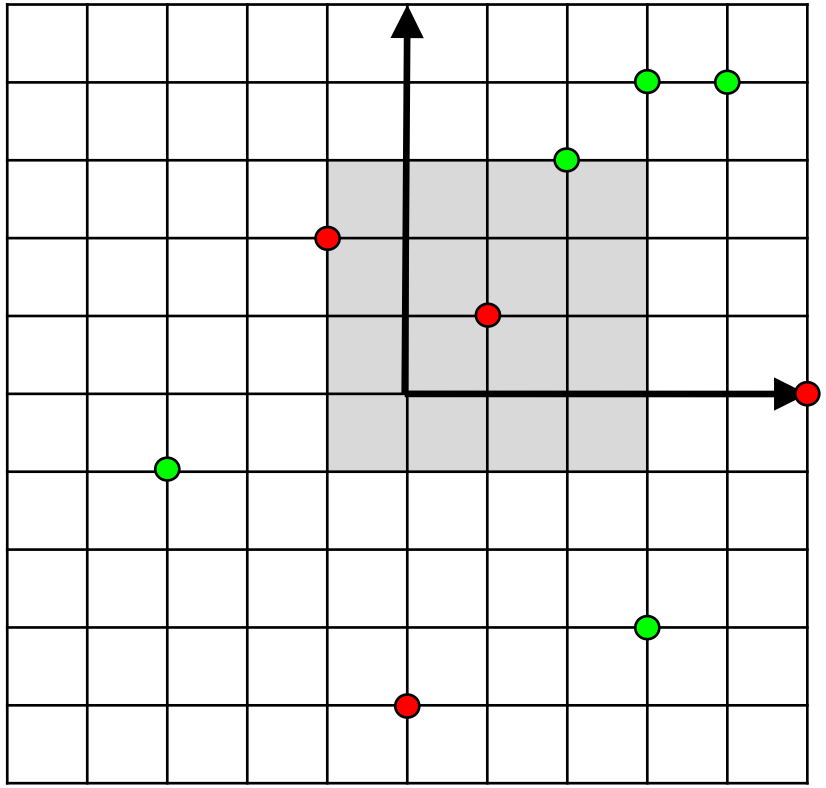
問題８　ヒバラ海に響く鐘の音

入出力例

- イヅア組合（赤）
- イワシロ組合（緑）



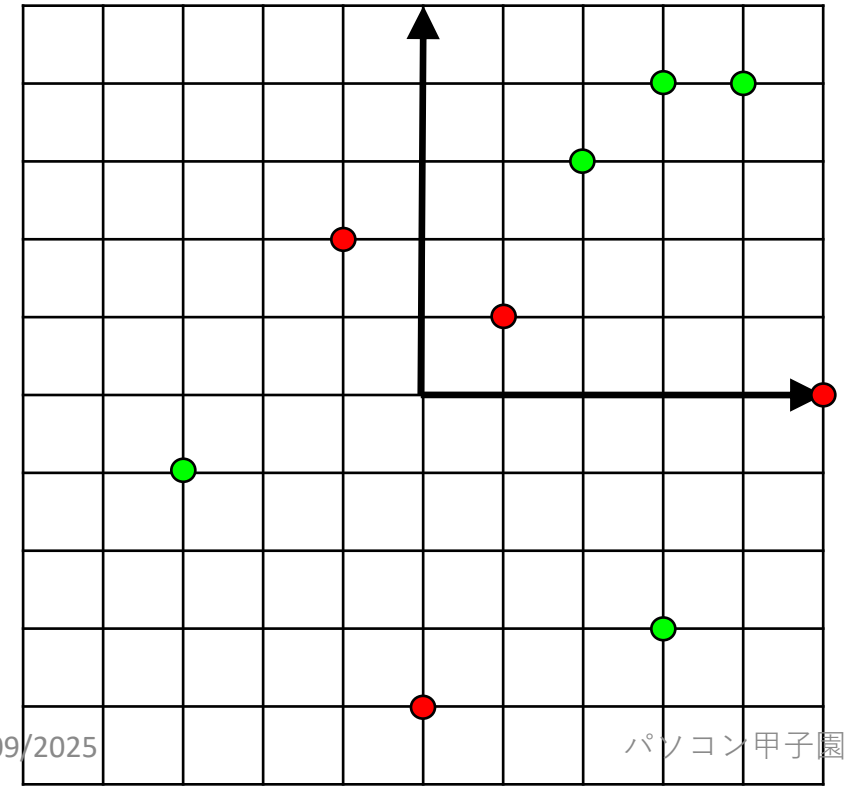
正方形内に赤と緑の点がどちらも存在するようにしたい
→　$D = 2$が最小



問題８　ヒバラ海に響く鐘の音

考察

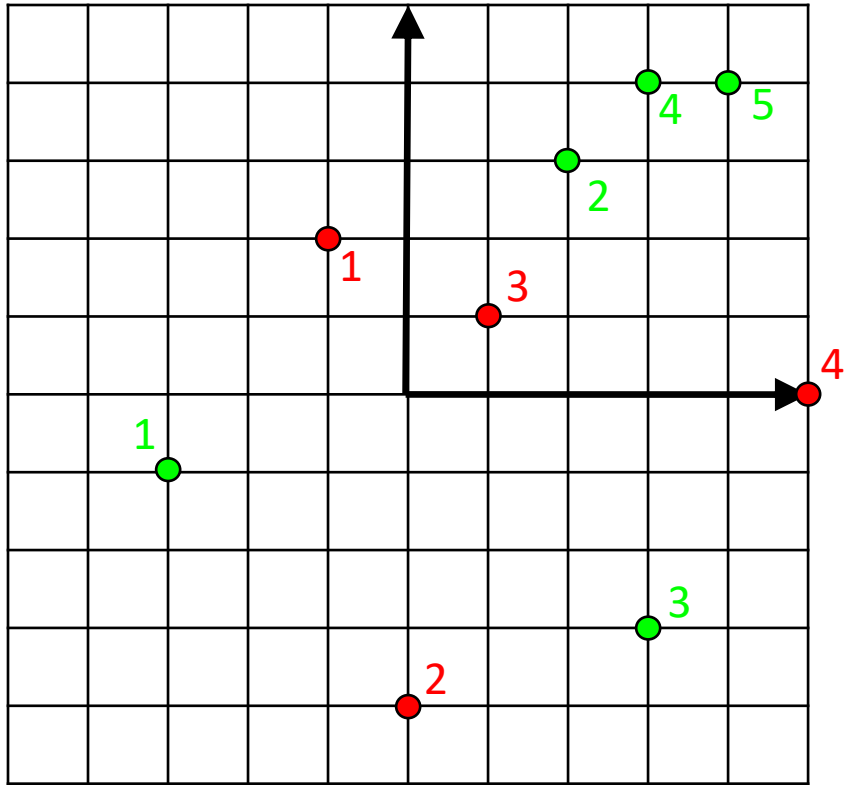
- 点に所属グループがあり、距離の測り方がユークリッド距離ではない場合の最近点対の問題
- あるD の値に対して、条件を満たす（範囲内に赤と緑の点がどちらも存在する）正方形が存在するかは以下のように求める。



問題８　ヒバラ海に響く鐘の音

解法

- 赤い点、緑の点、それぞれのグループ内で、頂点の座標を整列しておく(x 座標が小さい順で、x座標が同じ場合はy 座標が小さい順)



問題８　ヒバラ海に響く鐘の音

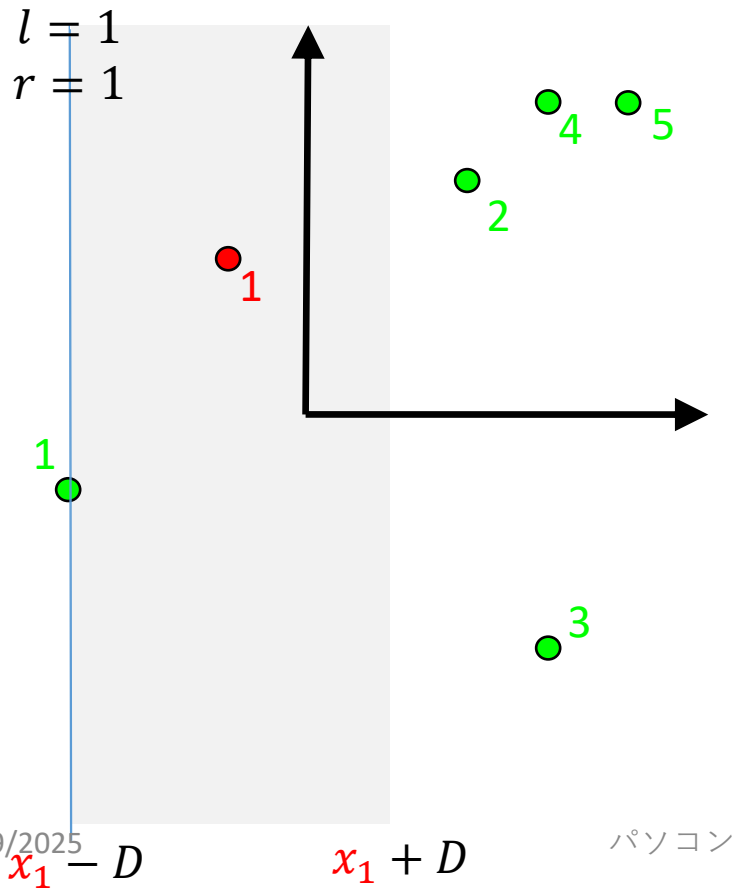
解法

- D の値に対して２分探索をする
- あるD の値に対して、条件を満たす（範囲内に赤と緑の点がどちらも存在する）正方形が存在するかを判定する

問題８　ヒバラ海に響く鐘の音

解法

- あるDの値に対して、条件を満たす（範囲内に赤と緑の点がどちらも存在する）正方形が存在するか？

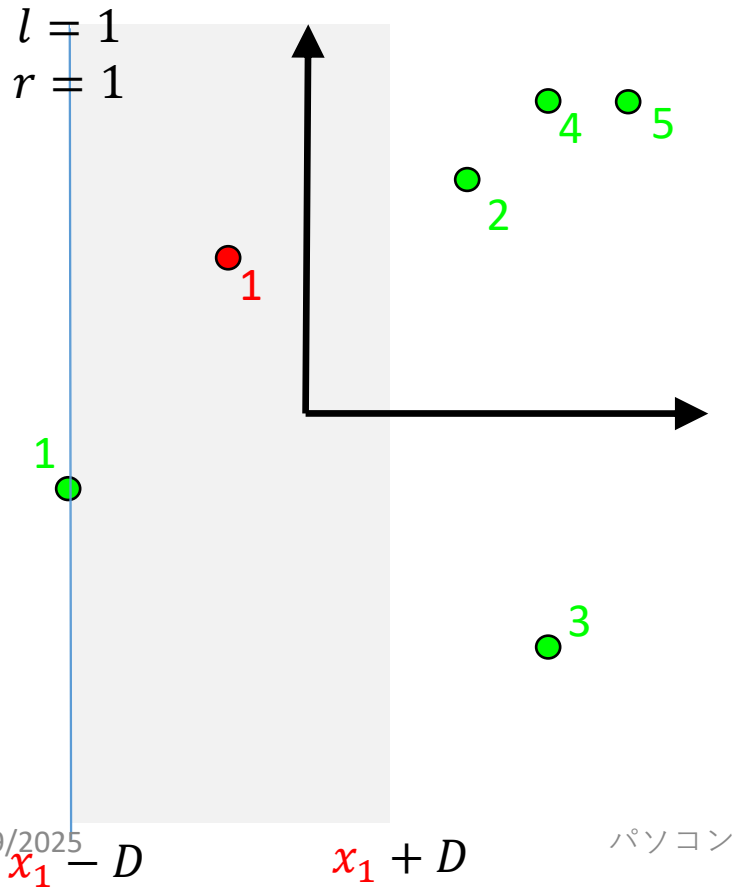


- 赤い点１のx座標$\pm D$の範囲にある緑の点を左から順に探す
- 緑の点を走査するためのインデクス$l = 1, r = 1, l \leq r$を用意する

問題８　ヒバラ海に響く鐘の音

解法

- あるDの値に対して、条件を満たす（範囲内に赤と緑の点がどちらも存在する）正方形が存在するか？

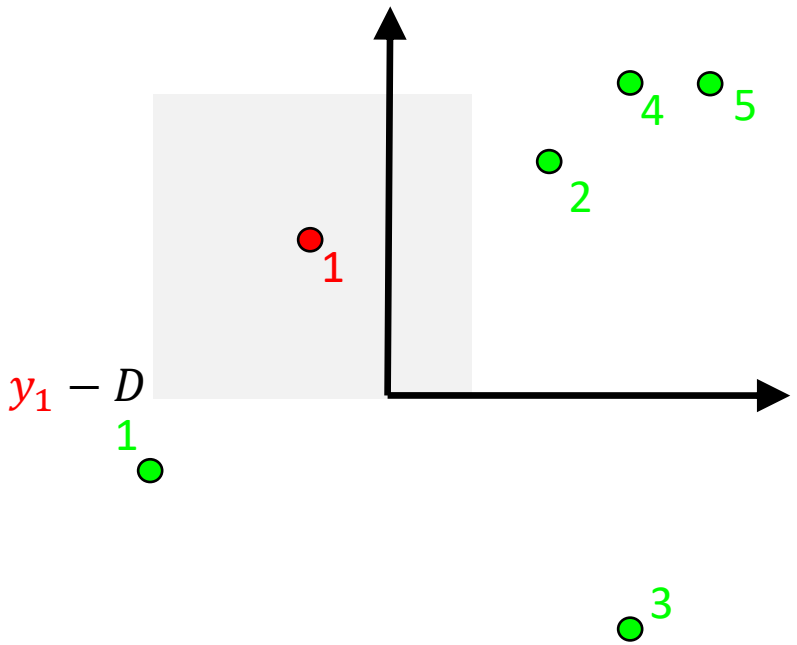


- 赤い点１のx座標$\pm D$の範囲にある緑の点を左から順に探す
- 緑の点を走査するためのインデクス$l = 1, r = 1, l \leq r$を用意する
- 緑の点（候補）を、y座標の昇順で管理するデータ構造を用意する
  - $r = 1$番目の緑の点のx座標が$x_1 + D$以下なので、候補としてデータ構造登録する→$r = 2$とする
  - $r = 2$番目の緑の点のx座標が$x_1 + D$より大きいのでrはそのまま
  - $l = 1$番目の緑の点のx座標が$x_1 - D$以上なのでlはそのまま

問題８　ヒバラ海に響く鐘の音

解法

- あるDの値に対して、条件を満たす（範囲内に赤と緑の点がどちらも存在する）正方形が存在するか？

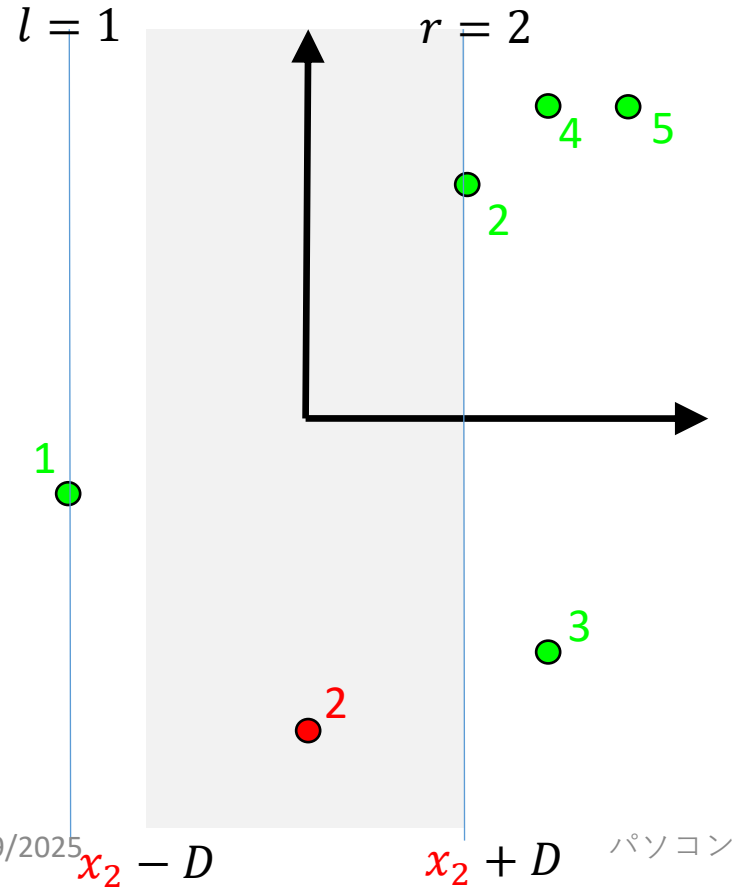


- 赤い点１のx座標$\pm D$の範囲にある緑の点を左から順に探す
- 緑の点を走査するためのインデクス$l = 1, r = 1, l \leq r$を用意する
- 緑の点（候補）を、y座標の昇順で管理するデータ構造を用意する
  - $r = 1$番目の緑の点のx座標が$x_1 + D$以下なので、候補としてデータ構造登録する→$r = 2$とする
  - $r = 2$番目の緑の点のx座標が$x_1 + D$より大きいのでrはそのまま
  - $l = 1$番目の緑の点のx座標が$x_1 - D$以上なのでlはそのまま
- データ構造の中で$y_1 - D$以上となる最初の候補を探索する
- 赤い点１については、そのような候補は無い

問題８　ヒバラ海に響く鐘の音

解法

- あるDの値に対して、条件を満たす（範囲内に赤と緑の点がどちらも存在する）正方形が存在するか？

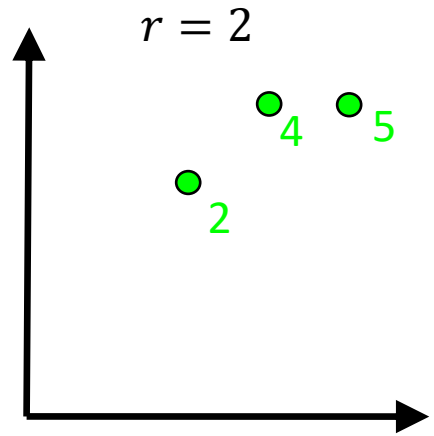


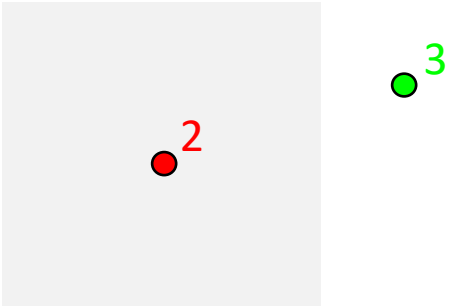
- 赤い点２のx座標$\pm D$の範囲にある緑の点を左から順に探す
- 現在$l=1, r=2$
  - $r=2$番目の緑の点のx座標がx_2+D以下なので、候補としてデータ構造登録する→$r=3$とする
  - $r=3$番目の緑の点のx座標がx_2+Dより大きいのでrはそのまま
  - $l=1$番目の緑の点のx座標がx_2-D未満なので、データ構造から緑の点１を消す→$l=2$とする

問題８　ヒバラ海に響く鐘の音

解法

- あるDの値に対して、条件を満たす（範囲内に赤と緑の点がどちらも存在する）正方形が存在するか？



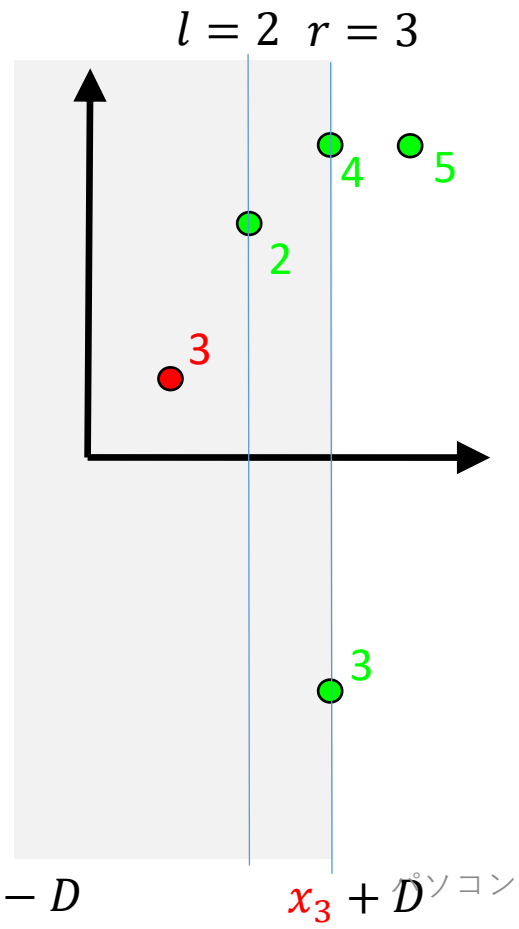


- 赤い点２のx座標$\pm D$の範囲にある緑の点を左から順に探す
- 現在$l=1, r=2$
  - $r=2$番目の緑の点のx座標がx_2+D以下なので、候補としてデータ構造登録する→$r=3$とする
  - $r=3$番目の緑の点のx座標がx_2+Dより大きいのでrはそのまま
  - $l=1$番目の緑の点のx座標がx_2-D未満なので、データ構造から緑の点１を消す→$l=2$とする
- データ構造の中でy_2-D以上となる最初の候補を探索する
- 緑の点２のy座標が見つかるが、y_2+Dより大きい

問題８　ヒバラ海に響く鐘の音

解法

- あるDの値に対して、条件を満たす（範囲内に赤と緑の点がどちらも存在する）正方形が存在するか？

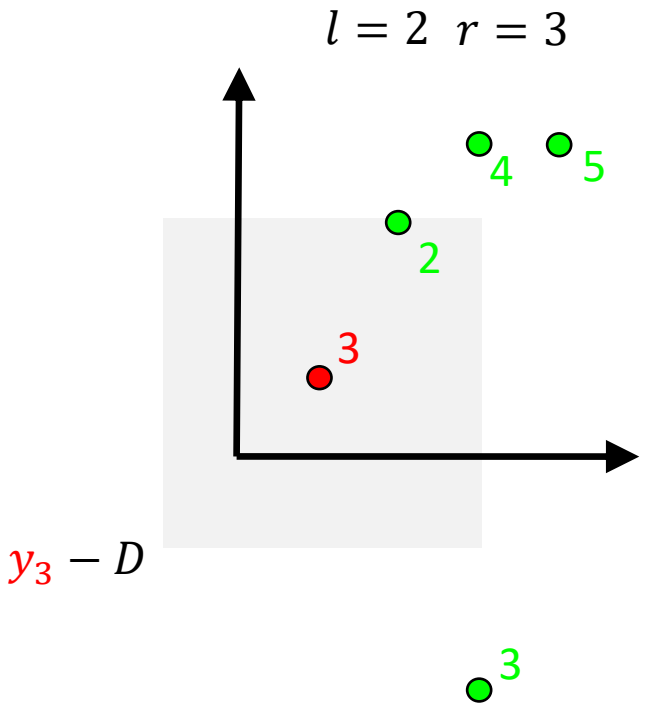


- 赤い点３のx座標$\pm D$の範囲にある緑の点を左から順に探す
- 現在$l=2, r=3$
  - $r=3$番目の緑の点のx座標がx_3+D以下なので、候補としてデータ構造登録する→$r=4$とする
  - $r=4$番目の緑の点のx座標がx_2+D以下なので、候補としてデータ構造登録する→$r=5$とする
  - $l=2$番目の緑の点のx座標がx_2-D以上なので、lはそのまま

問題８　ヒバラ海に響く鐘の音

解法

- あるDの値に対して、条件を満たす（範囲内に赤と緑の点がどちらも存在する）正方形が存在するか？

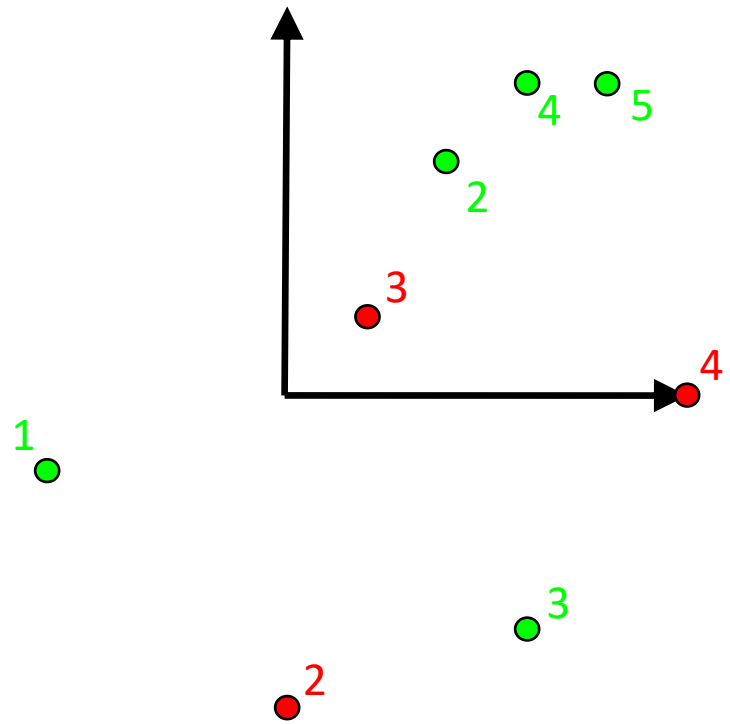


- 赤い点３のx座標$\pm D$の範囲にある緑の点を左から順に探す
- 現在$l = 2, r = 3$
  - $r = 3$番目の緑の点のx座標が$x_3 + D$以下なので、候補としてデータ構造登録する→$r = 4$とする
  - $r = 4$番目の緑の点のx座標が$x_2 + D$以下なので、候補としてデータ構造登録する→$r = 5$とする
  - $l = 2$番目の緑の点のx座標が$x_2 - D$以上なので、lはそのまま
- データ構造の中で$y_3 - D$以上となる最初の候補を探索する
- 緑の点２のy座標が見つかり、その値は$y_3 + D$以下である
- 赤３と緑２のペアは条件を満たす

問題８　ヒバラ海に響く鐘の音

解法

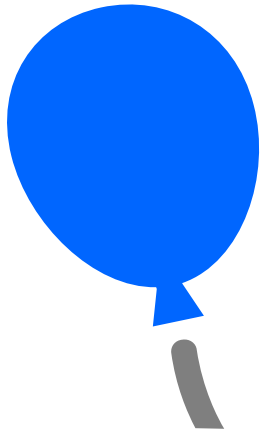
- あるDの値に対して、条件を満たす（範囲内に赤と緑の点がどちらも存在する）正方形が存在するか？



- 赤い点それぞれについて、x座標$\pm D$の範囲にある緑の点をSliding Windowで右方向に走査する
- 緑の点は、それぞれ高々一回ずつしか登録＆削除されない
- 現在の赤い点の座標$y_i - D$をmultisetで探すと、$y_i - D$以上を満たす点が得られる(同じy座標の点が複数あり得るので、削除の際のことも考えるとmultisetが便利)
- その点が$y_i + D$を満たしていれば正方形内にある
- 計算量$O((N+M)\log(M)\log(x_{max} - x_{min}))$

問題９　スパイ　（１０点）

正答数:　　３チーム



問題９　スパイ

概要

- N個の都市があり、そのうちのいくつかの都市には重要施設がある。
- 重要施設には、重要度の低い方から順に1からLまでの番号が振られている。
- 重要施設iがある都市をv_iで表す。
- N個の都市の間は、M本の一方通行の道路で直接結ばれている。各道路には、距離が定められている。
- 最初、スパイは都市v_1にある重要施設1に潜入している。
- スパイは以下の方針に従って移動している。
  - v_1からアジトに最短距離で移動する。
  - アジトへの最短経路の途中にある重要施設に潜入する。
  - 重要度の低い施設から順番に潜入する。
- スパイが最後に潜入する重要施設の番号をfで表すと、スパイはv_1から逃走した後、重要施設$2, 3, \cdots, f$がある都市$v_2, v_3, \cdots, v_f$を順番に訪れてからアジトに移動することになる。
- スパイはv_1からアジトに最短距離で移動するので、v_1からアジトに移動する途中で$v_2, v_3, \cdots, v_f$以外の都市を経由することもあり得る。
- $f = 1$から$f = L$までをそれぞれ仮定したとき、それぞれの仮定についてアジトが存在する可能性のある都市がいくつあるのかを求めたい。

問題９　解法

解法

- 都市を頂点、道路を重み付き有向辺とする、N点M辺のグラフGとして問題を考える。
- 都市iから都市jを直接結ぶ道路(i, j)の距離を$w(i, j)$と表し、都市iから都市jまで最短経路で移動したときの距離（最短距離）を$dist(i, j)$と表す。
- 問題を言い換えると、$f = 1$から$f = L$までのそれぞれについて、
  $\sum_{i=1}^{f-1} dist(v_i, v_{i+1}) + dist(v_f, x) = dist(v_1, x)$ を満たすxの数を求めればよい。
- 全都市間の最短距離を求める必要はなく、v_1から他の都市への最短距離がわかれば、以降に示す手順により解くことができる。

問題９　解法

解法

- v_1から他の都市への最短距離をダイクストラ法で求める。
- $dist(v_1, i) + w(i, y) = dist(v_1, y)$を満たす、有向辺$(i, y)$ から成る有向非巡回グラフ（DAG: Directed Acyclic Graph）G'を構築する。
  - $dist(v_1, i) + w(i, y) > dist(v_1, y)$となる場合、道路$(i, y)$を通るとアジトへ最短距離で行けないので除外でき、$G'$はDAGになる。
- G'において、前述の条件を満たす頂点xの個数が求めるxの数である。この数は次ページに示す動的計画法を用いて求めることができる。

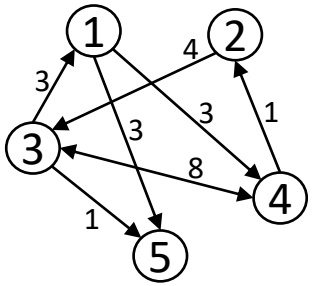
問題９　解法

解法

- $dp[i]$ =「G'における点v_1から点iへの経路の中で、$v_1, v_2, \cdots, v_k$を順にたどることのできるようなkの最大値」と定義する。
- dpは、$dp[i] = \begin{cases} 1 & \text{if } i = v_1 \\ -1 & \text{otherwise} \end{cases}$ と初期化し、 v_1からの最短距離が短い都市順（トポロジカル順序）で以下のように更新すればよい。
  - $dp[i] = \max\left(dp[i], \max_{\forall (j,i) \in E'} \left(dp[j] + \begin{cases} 1 & \text{if } i = v_{dp[j]+1} \\ 0 & \text{otherwise} \end{cases}\right)\right)$
    - E'はG'に含まれる辺の集合。
- このようにdpを更新したとき、$z \leq dp[j]$である頂点jは求める頂点xとしての条件を満たす。
- よって、$f = z$についての答えは$1 \leq j \leq N, z \leq dp[j]$を満たす$j$の個数である。
  - $z = dp[j]$であるjの個数を$counter[z]$に数え上げる。そして、この$counter[z]$について$z = L$から$z = 1$へzが大きい順に累積和を取れば、$f = 1$から$f = L$までのそれぞれの答えが得られる。

問題９　解法

例

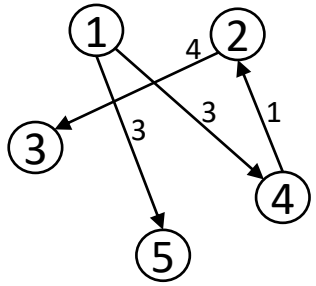


i	1	2	3	4	5
v_i	1	4	2	3	5

- v_1から他の都市への最短距離を求める。

i	1	2	3	4	5
$dist(v_1, i)$	0	4	8	3	3

- DAGを構築する。



- v_1からの最短距離が短い都市順にdpを更新する。
  - この例の場合、1, 4, 5, 2, 3の順
    （最短距離が同じ場合は、どれが先でもよい）

dp

$j \diagdown i$	1	2	3	4	5
初期状態	1	-1	-1	-1	-1
1	1	-1	-1	2	1
4	1	3	-1	2	1
5	1	3	-1	2	1
2	1	3	4	2	1
3	1	3	4	2	1

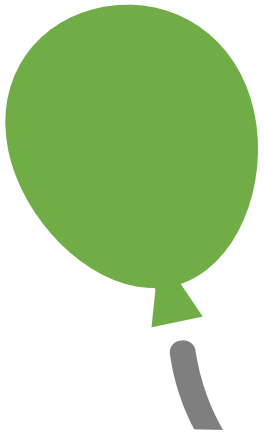
- 答えを求めるために、$z = dp[j]$であるjの個数を数え、$z = L$から$z = 1$へzが大きい順に累積和を取る。

z	1	2	3	4	5
$counter[z]$	2	1	1	1	0

f	1	2	3	4	5
$answer[f]$	5	3	2	1	0

問題１０　クッキー詰め職人のこだわり（１０点）

正答数:　　11チーム



問題１０　クッキー詰め職人のこだわり

概要

- N個のクッキーがベルトコンベアで運ばれてくる。クッキーの大きさはc_iで、すべて異なる。
- 運ばれてきたすべてのクッキーを順番に袋に詰める。
- 袋はいくつ使ってもよいが、新しい袋にクッキーを詰め始めたら、それまでにクッキーを詰めた袋にはそれ以上クッキーを詰められない。
- １つの袋にクッキーを詰めるときは、以下の条件を満たすようにしなければならない。
- 条件：１つの袋に入れる最初のクッキーが最も小さく、最後に入れるクッキーが最も大きくなるように袋詰めする。
- このようにしてクッキーを袋に詰めるとき、条件を満たすような袋詰めの方法の総数を$998{,}244{,}353$で割った余りを求める。

制約：$1 \leq N \leq 200{,}000,\ 1 \leq c_i \leq 1{,}000{,}000{,}000$　問題６より制約が大きくなっている！

問題１０　クッキー詰め職人のこだわり

考察

- 以下の説明では、問題６の解説と同じ記号と記法を使う。列x[]の長さがnのとき、そのインデックスは0から$n-1$までとする。 x[i]からx[j]までのx[]の部分列をx[i:j]で表す。
- 問題６の想定解法の計算量は$O(N^2)$ ⇒　Nの上限が増えたので、時間制限になる場合あり。
- 問題６では、部分列A[0:i]の分割の総数を求めるとき、以下の処理を行った。

  for (j=i; j>=0; j--) { if (部分列A[j:i]が条件を満たす) dp[i+1] += dp[j]; }

- この処理は$O(N)$かかる。これを$O(\log N)$で行うことで、プログラム全体の計算量を$O(N \log N)$に高速化する。
- そのために、累積和と単調スタック(monotonic stack)を使う。

問題１０　クッキー詰め職人のこだわり

考察

- 部分列A$[j;i]$が条件を満たすには、 A$[j;i]$の末尾が最大でなければならない。
- $k<i$かつA$[k]$>A$[i]$とすると、$j \leq k$を満たすすべてのjについてA$[j;i]$の末尾は最大ではない。

5, 15, 10, 12 → 部分列12と10,12の末尾は最大
部分列15,10,12と5,15,10,12の末尾は最大ではない

- 各i $(0 \leq i < N)$について、 $0 \leq k < i$かつA$[k]$>A$[i]$を満たす最大のインデックスkを配列biggerのi番目に保存する。
- iについて、そのようなインデックスがないときは、bigger$[i]$に-1を保存する。

A[] = { 5, 15, 10, 12 } → bigger[] = { -1, -1, 1, 1 }

問題１０　クッキー詰め職人のこだわり

考察

- 部分列A[$j; i$]が条件を満たすには、A[$j; i$]の先頭が最小でなければならない。

A[$0; i$] = { 8, 5, 15, 10, 9, 12 } ⇒ 末尾が12のときに、先頭が最小になる部分列は、5, 15, 10, 9, 12と9, 12と12のみ。

- 各jについて、先頭が最小になる列A[$j; i$]の先頭A[j]をインデックスjの順番に並べると、単調増加列が得られる。

A[$0; i$] = { 8, 5, 15, 10, 9, 12 } ⇒ 5, 9, 12

- この単調増加列中の各要素をそのインデックスで置き換えた配列をsmallerとする。

A[$0; i$] = { 8, 5, 15, 10, 9, 12 } ⇒ smaller[] = { 1, 4, 5 }

問題１０　クッキー詰め職人のこだわり

考察

- smaller[] = { $j_1, j_2, \ldots, j_n$ } の各要素に累積和を対応付ける。
- j_k に対応付ける累積和は、dp[j_1] + dp[j_2] + … + dp[j_k]

A[0; i] = { 8, 5, 15, 10, 9, 12 } ⇒ smaller[] = { 1, 4, 5 }

dp[1]　dp[1]+dp[4]　dp[1]+dp[4]+dp[5]

- smallerの各要素に対応付ける累積和を保存する配列sumを用意する。ただし、sum[0] = 0 として、smaller[i] に sum[$i + 1$] を対応付ける。

smaller[] = { 1, 4, 5 } ⇒ sum[] = { 0, dp[1], dp[1]+dp[4], dp[1]+dp[4]+dp[5] }

問題１０　クッキー詰め職人のこだわり

考察

- ここまでは部分列の先頭が最小であることと、末尾が最大であることを（それぞれsmallerとbiggerを使って）別々に扱った。
- ここからは、両方を組み合わせて、累積和を使って高速にdp[i+1]を求める。

A[0; i] = { 8, 5, 15, 10, 9, 12 } → bigger[0; i] = { -1, 0, -1, 2, 2, 2 }　　smaller[] = { 1, 4, 5 }
sum[] = { 0, dp[1], dp[1]+dp[4], dp[1]+dp[4]+dp[5] }

- bigger[i]=2なので、A[4; i]とA[5; i]は条件を満たすが、 A[1; i]は条件を満たさない。

A[4; i] = { 9, 12 }　　A[5; i] = { 12 }　　A[1; i] = { 5, 15, 10, 9, 12 }

- 以上より、dp[6] = dp[4] + dp[5] = sum[3] – sum[1] 。← 累積和でdp[] が求まる！
- sum[3]はsum[]の末尾の要素。sum[1]のインデックス1はbigger[i]（この場合は2）以上のsmaller[]の最小の要素（この場合は4）のインデックス（smaller[1] = 4）。
  smaller[j] に sum[$j + 1$] を対応付けていることに注意。

問題１０　クッキー詰め職人のこだわり

考察

- dp[] の計算で使う累積和のインデックスの計算（bigger[i] 以上のsmaller[]の最小の要素のインデックスを求める）は、smallerが単調増加列なので二分探索で求まる。
- 累積和によるdp[]の計算は$O(\log N)$でできる。
- bigger[]の計算は、dp[] の計算より前にできる（スタックを使って $O(N)$ でできる）。
- 後は、各iについてsmaller[]とsum[]の更新が高々$O(\log N)$でできればよい。
- smaller[]とsum[]の更新は、 iを0から$N-1$まで動かす間に、全部で$O(N)$でできる。したがって、各iについて実質的に$O(1)$でできる。

問題１０　クッキー詰め職人のこだわり

解法

- smaller[]はスタックとして管理する。sum[]はsmaller[]と対応付けられているので、必然的にsum[]もスタックになる（smaller[]は単調増加列なので単調スタック）。
- iについて、スタックsmaller[]の先頭vとA[i]を比較し、$v >$ A[i]ならスタックsmaller[]の先頭をポップする。これをスタックの先頭がA[i]より小さくなるまで続ける。その後、iをスタックsmaller[]にプッシュする。
- スタックsmaller[]の先頭をポップするときは、スタックsum[]の先頭もポップする（smaller[]とsum[]の要素が対応付いているため）。スタックsmaller[]にiをプッシュするときは、スタックsum[]にもiと対応付けられた値dp[v_1]+dp[v_2]+…+dp[v_n]+dp[i]をプッシュする（iをプッシュする前のsmaller[]を{ $v_1, v_2, \ldots, v_n$ }とする）。
- smallerの処理は、各iについて iをスタックsmaller[]にプッシュする処理と、それらのポップと比較だけなので、 iを0から$N-1$まで動かす間に全部で$O(N)$かかる。Sumについても同様。

問題１０　クッキー詰め職人のこだわり

実行例

A[] = { 8, 5, 15, 10, 9, 12 }　bigger[] = { -1, 0, -1, 2, 2, 2 }

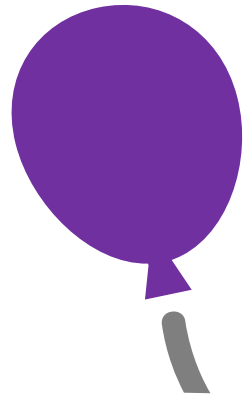
smallerとsumの赤字で示した要素が、スタックからポップされる要素

	初期値	i=0	i=1	i=2	i=3
A[i]		8	5	15	10
smaller	{ }	{ 0 }	{ 1 }	{ 1, 2 }	{ 1, 3 }
sum	{ 0 }	{ 0, dp[0] }	{ 0, dp[1] }	{ 0, dp[1], dp[1]+dp[2] }	{ 0, dp[1], dp[1]+dp[3] }
		8=A[0] > A[1]=5		15=A[2] > A[3]=10	
dp[$i+1$]	1	sum[1]−sum[0]	sum[1]−sum[0]	sum[2]−sum[0]	sum[2] −sum[1]

	i=4	i=5
A[i]	9	12
smaller	{ 1, 4 }	{ 1, 4, 5 }
sum	{ 0, dp[1], dp[1]+dp[4] }	{ 0, dp[1], dp[1]+dp[4], dp[1]+dp[4]+dp[5] }
	10=A[3] > A[4]=9	
dp[$i+1$]	sum[2] −sum[1]	sum[3] −sum[1]

問題１１　べこべこの攻略（１０点）

正答数:　　9 チーム



問題１１　べこべこの攻略

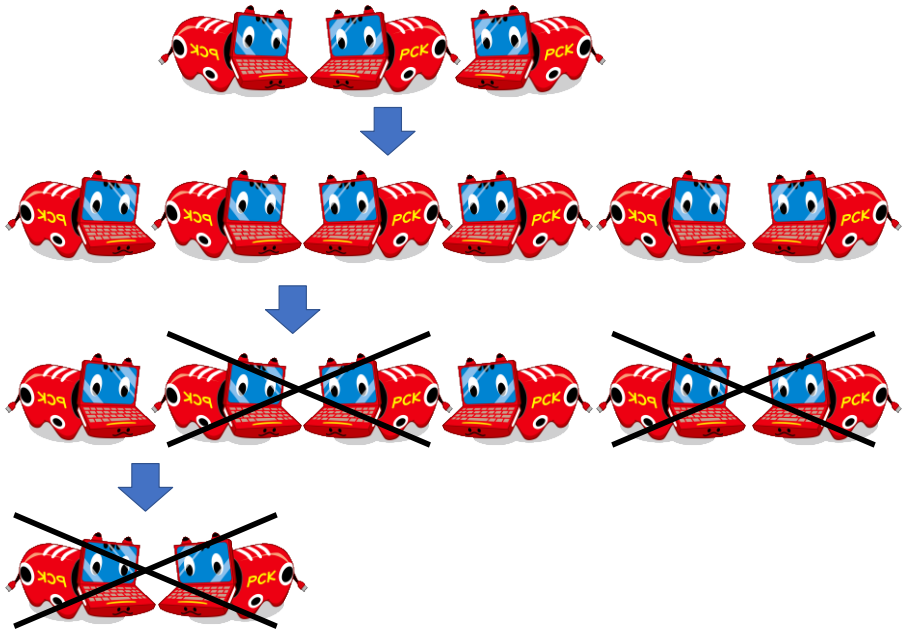
概要

- 左(L)または右(R)に向いた赤べこをN個１列に並べる。
- あらかじめ M 個の赤べこが並んでいる。その左に K 個、右に$N-M-K$個の赤べこを並べる。
- 隣り合っている２個の赤べこが向かい合うとき、それらを消すことができる。
- 赤べこを次々と消していくことで、すべての赤べこを消すことができるような、赤べこの並べ方の総数を$998{,}244{,}353$で割った余りを求めよ。

制約：$1 \leq N \leq 500{,}000,\ 1 \leq M \leq N,\ 0 \leq K \leq N-M$

（ただし、Nは偶数）

$N=6,\ M=3,\ K=1$で、
文字列RLLが与えられたとき



この例では、すべての赤べこを消すことのできる並べ方は１つだけ。

問題１１　べこべこの攻略

考察

- RとLをそれぞれ開きカッコと閉じカッコとみなすと、この問題は、カッコの対応がとれた文字列の総数を求める問題の変形版になる。
- N個のカッコからなる、カッコの対応がとれた文字列の総数は、 $H = N/2$ とすると

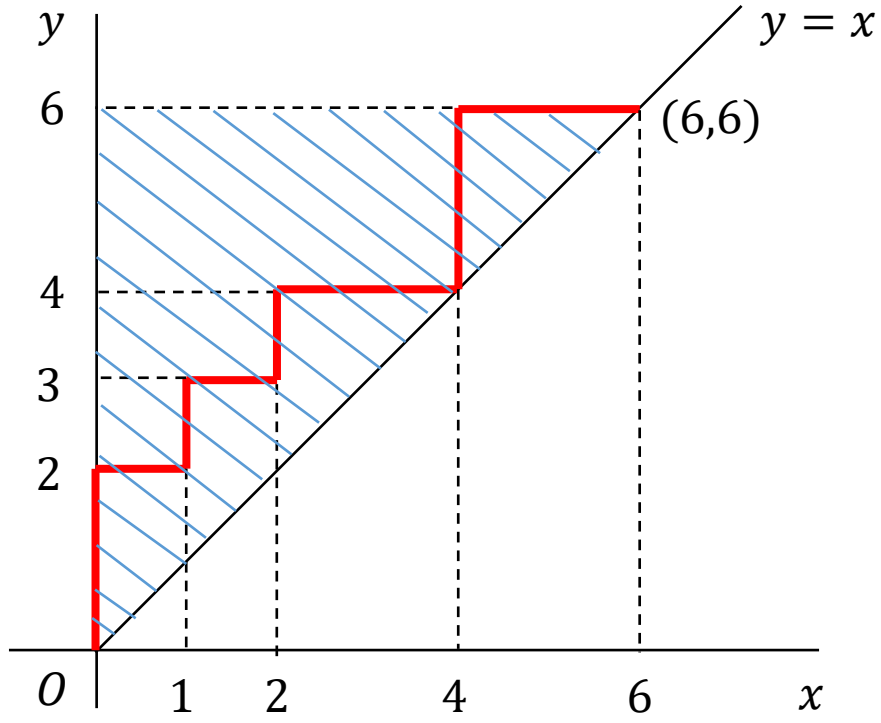
  $\frac{N!}{(H+1)!H!}$ で表される。これをカタラン数という。
- しかし、この問題では先頭から$K+1$個目から$K+\mathrm{M}$個目まではあらかじめ決まっているので、カタラン数の公式をそのまま使うことはできない。
- カタラン数の求め方に沿って、解法を考察する必要がある。
- まず、RとLの対応が取れた文字列の総数を表すカタラン数の求め方を復習し、それを踏まえて問題の解法を考察する。

問題１１　べこべこの攻略

考察：カタラン数の求め方の復習 (1)

- 文字RとLからなる文字列を先頭から順に読むことを、座標平面上の移動に対応させる。
- 原点Oから出発して、Rを読んだらy軸の正の方向（以下、北）に1進み、Lを読んだら x 軸の正の方向（以下、東）に1進む。
- RとLの対応がとれた長さ$N(=2H)$の文字列が与えられたときは、原点Oから出発して、$y \geq x$の領域（図の斜線の領域）を通って進んでいき、すべての文字を読み終えたとき点(H, H)に到達する。
- これ以降は北または東へ進む経路だけを考える。単に経路と言ったときは、西（x軸の負の方向）や南（y軸の負の方向）には進まないものとする。

$N = 12$ $(H = 12)$として、文字列 RRLRLRLLRRLL を読んだときの移動経路

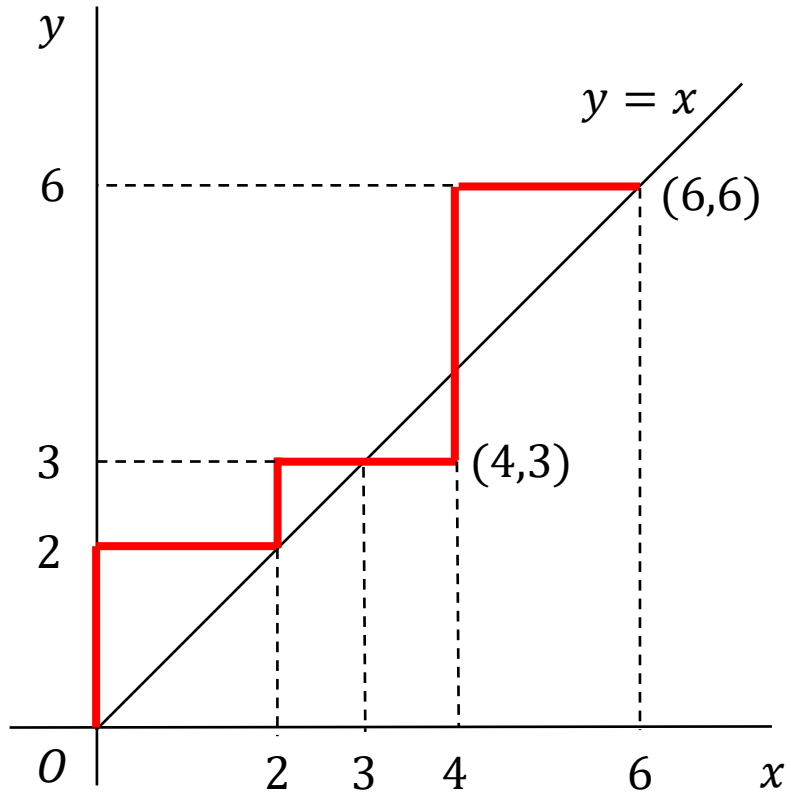


問題１１　べこべこの攻略

考察：カタラン数の求め方の復習 (2)

- 原点Oから出発し、$y < x$を満たす点(x, y)を少なくとも１つ通って点(H, H)に到達する経路の集合をS_{Bad}とする。
- S_{Bad}に属する経路は、RとLをH個ずつ持つが、RとLの対応はとれていない文字列を表す。
- 原点Oから出発して$y \geq x$の領域だけを通って進み、点(H, H)に到達する経路の数は、O から(H, H) までの経路の数から$|S_{Bad}|$を引いた値（Xが集合のとき、$|X|$はXの要素の個数を表す）。
- ΠをS_{Bad}に属する経路とする。Πが最初に$y < x$を満たす点(x, y)を通るとき、その点は$(v + 1, v)$と書ける$(0 \leq v \leq H - 1)$。$y = x$上の点(v, v)から東へ1進むことで 点$(v + 1, v)$ へ進む（右図参照）。

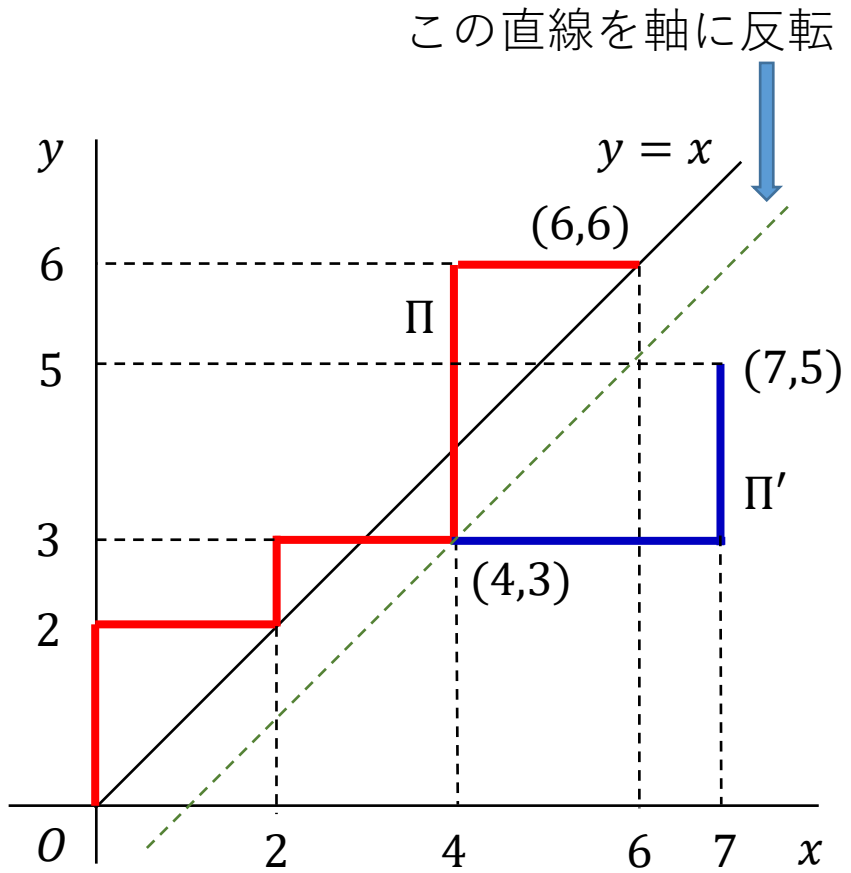
S_{Bad}に属する経路の例
$(N = 12,\ H = 6,\ v = 3)$



問題１１　べこべこの攻略

考察：カタラン数の求め方の復習 (3)

- 経路Πの$(v+1, v)$から(H, H)までを、$(v+1, v)$を通り直線$y = x$と平行な直線を軸に反転した経路をΠ′とする（経路Π′ を経路Πの反転と呼ぶ）。
- Π′は、$(v+1, v)$まではΠと同じように進み、それ以降はΠでは東へ進んでいたときは北へ進み、 Πでは北へ進んでいたときは東へ進む。
- Πでは、$(v+1, v)$から(H, H)に達するまでに東に$H-v-1$、北に$H-v$進む。$(v+1, v)$以降を反転すると東に$H-v$、北に$H-v-1$進むので、経路Π′はvに関わらず点$(H+1, H-1)$に到達する（右図参照）。

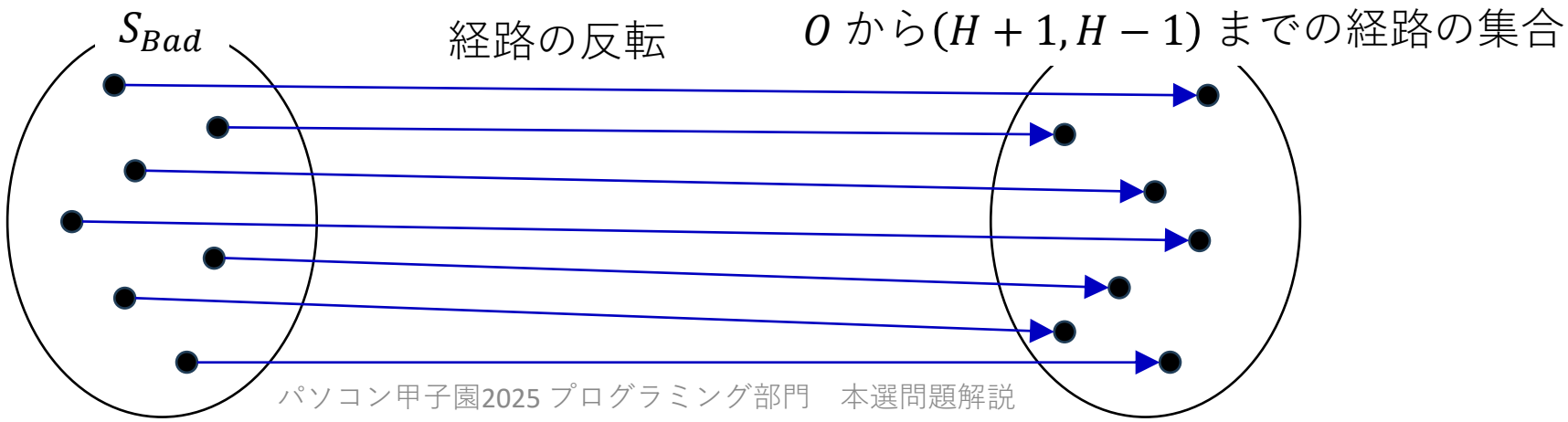


vの値に関わらず点$(7,5)$に達する。

問題１１　べこべこの攻略

考察：カタラン数の求め方の復習 (4)

- 集合S_{Bad}と、原点O から$(H+1, H-1)$ までの経路の集合の間には以下の関係が成り立つ。
  - S_{Bad}に属する２つの経路Π_1, Π_2が異なる$(\Pi_1 \neq \Pi_2)$なら、それらを反転した経路Π'_1, Π'_2も異なる経路になる（$\Pi'_1 \neq \Pi'_2$）。
  - 原点O から$(H+1, H-1)$ までのどの経路も、S_{Bad}に属する経路の反転である。
- このようなとき、集合S_{Bad} と原点O から$(H+1, H-1)$ までの経路の集合の間には１対１対応または全単射が存在すると言い、これら２つの集合に属する経路の数は等しい。



問題１１　べこべこの攻略

考察：カタラン数の求め方の復習 (5)

- 以上より、集合S_{Bad}に属する経路の数$|S_{Bad}|$ は、原点O から$(H+1, H-1)$ までの経路の数に等しいので、原点Oから出発し、$y \geq x$の領域を通って点(H, H)に到達する経路の数は以下のように求まる。

(原点Oからスタートし、 $y \geq x$の領域を通って進んでいき点(H, H)に到達する経路の数)
$=$ (Oから(H, H)までの経路の数) $-$ $|S_{Bad}|$
$=$ (Oから(H, H)までの経路の数) $-$ (Oから$(H+1, H-1)$までの経路の数)
$= {}_N C_H - {}_N C_{H+1}$

$$= \frac{N!}{H!\,H!} - \frac{N!}{(H+1)!\,(H-1)!} = \left(1 - \frac{H}{H+1}\right)\frac{N!}{H!\,H!} = \frac{N!}{(H+1)!\,H!}$$

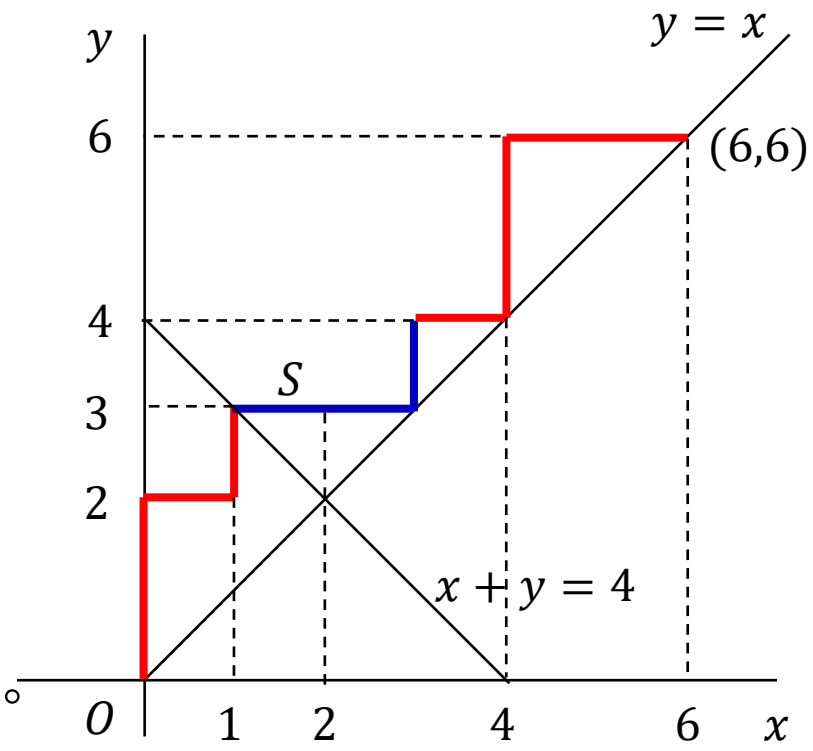
※$x \geq 0, y \geq 0$のとき、原点Oから東または北だけに進んで、東にx、北にy進んだ点(x, y)に達する経路の総数は ${}_{x+y}C_x$（数学Aの典型問題）。
${}_n C_k$ は、n個のものの中からk個を選ぶ方法の数（数学Aの教科書や参考書を参照）。

問題１１　べこべこの攻略

考察：問題の解法 (1)

- この問題では、長さNの文字列の先頭から$K+1$個目から$K+M$個目までの文字列Sがあらかじめ決まっている。
- Sから対応するRとLのペアを取り除いても、求める答は 変わらない（ たとえばSがL**RL**LRでもLLRでも答は同じ）。
- これ以降はSがL…LR…Rの形だけを考える（LとRの数はそれぞれ０個以上）。S中のLの数をa、Rの数をbとする。
- 先頭からK個のRまたはLを読むと、原点Oから出発して$x+y=K$上の点に達する。
- 原点Oから出発して$x+y=K$上の点を通り、そこから東へa、北へb進んだ後に点(H,H)に到達する経路（右図参照）で、 $y \geq x$ の領域だけを通るものの総数がこの問題の答え。

S =LLR、$N=12$、$H=6$、$M=3$、$K=4$、$a=2$、$b=1$のときに、文字列RRLR**LLR**LRRLLを読んだときの経路



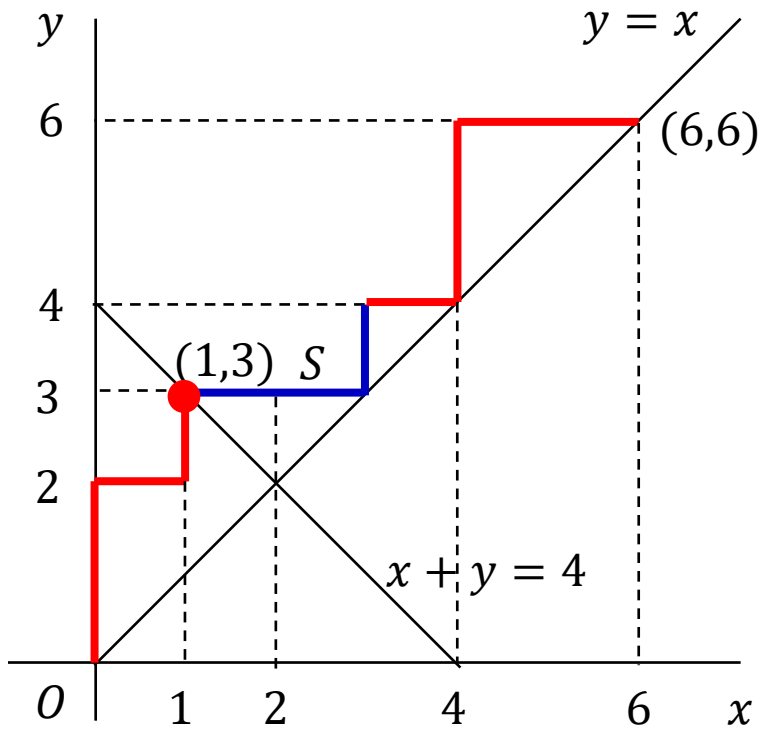
問題１１　べこべこの攻略

考察：問題の解法 (2)

- $x+y=K$上の点からaだけ東へ進んでも、$y<x$の領域に入らないようにしなければならない。
- xとyの範囲はともに、$0 \le x, y \le H$。また、Kの範囲は$0 \le K \le N-a-b$。
- したがって、可能な$x+y=K$上の点のx座標は

  $\max(0, K-H+b), \max(0, K-H+b)+1, \ldots, \left\lfloor \frac{K-a}{2} \right\rfloor$だけ。
- これらの点の１つ$(c, K-c)$を固定する（ただし、$\max(0, K-H+b) \le c \le \left\lfloor \frac{K-a}{2} \right\rfloor$）。原点$O$から$(c, K-c)$を通った後、東へ$a$、北へ$b$進み、最後に$(H, H)$に達するような、$y \ge x$の領域だけを通る経路の総数を求める 。

S =LLR、$N=12$、$H=6$、$M=3$、$K=4$、$a=2$、$b=1$のときに、$c=1$としたときの経路の例



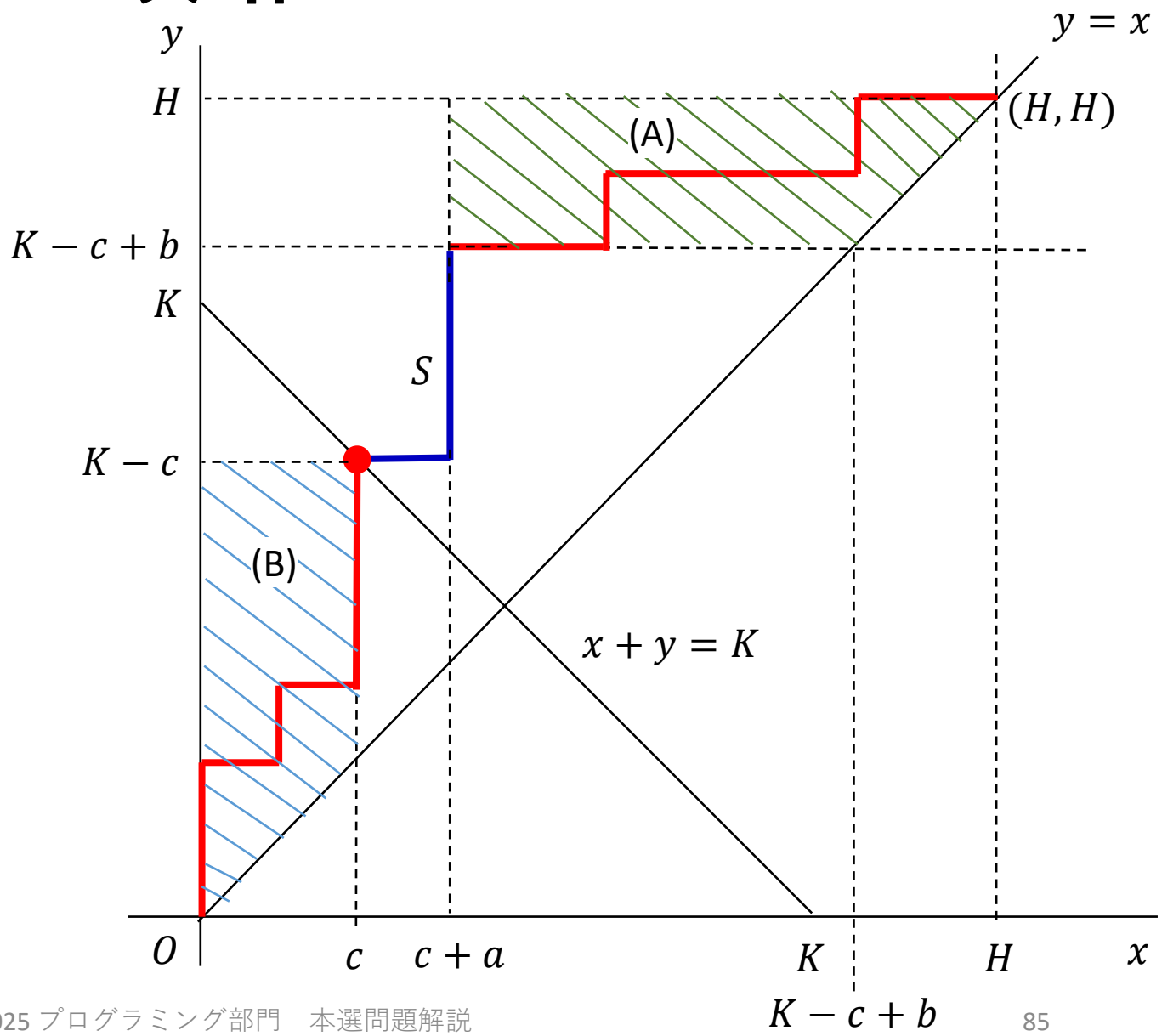
問題１１　べこべこの攻略

考察：問題の解法 (3)

原点Oから$(c, K-c)$を通った後、
東へa、北へb進み、最後に(H, H)に
達するような、$y \geq x$の領域だけを
通る経路の総数は、

領域(A)を通る経路の総数と、
領域(B)を通る経路の総数との積。

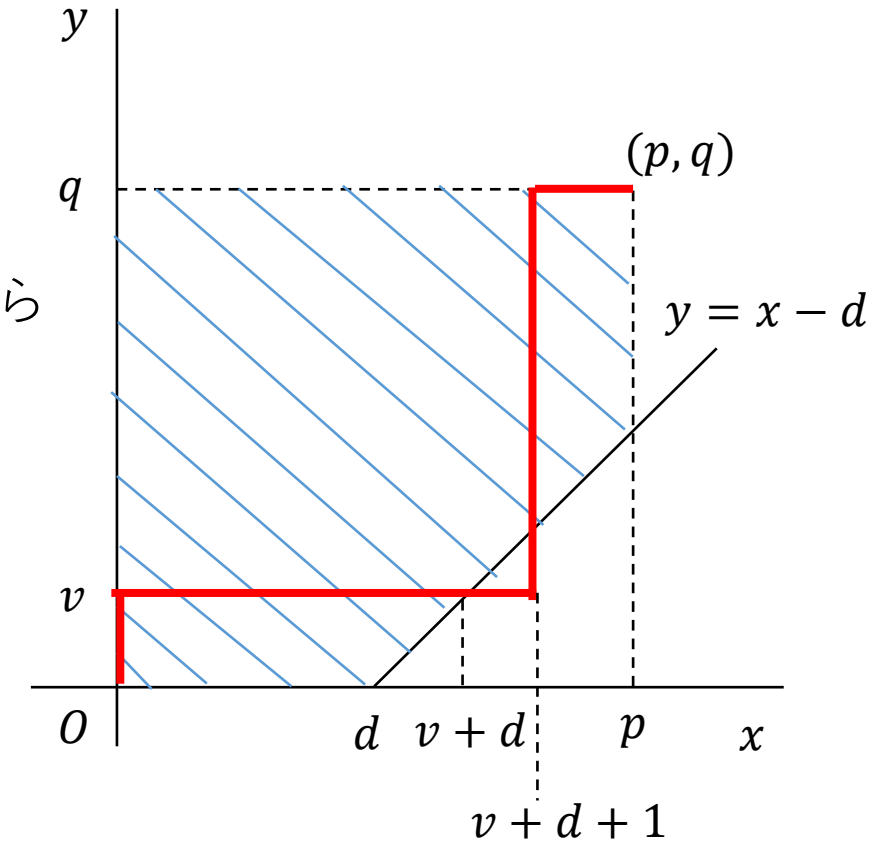
領域(A)、(B)を通る経路の総数は、
カタラン数の求め方を一般化する
ことで、求めることができる。



問題１１　べこべこの攻略

考察：問題の解法 (4)

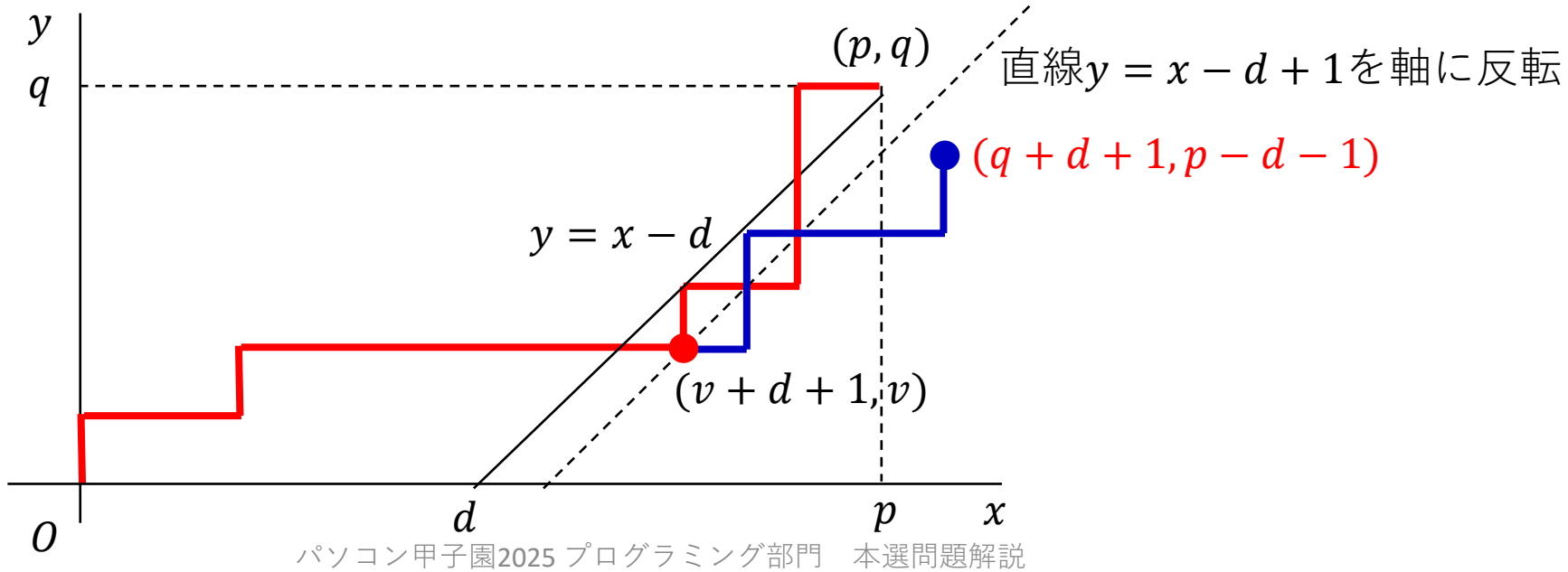
- 原点Oから$y \geq x - d$の領域（図の斜線の領域）だけを通って(p, q)に達する経路の総数を求める$(0 \leq d \leq p$、$p - d \leq q)$。
- 原点Oから、$y < x - d$を満たす点(x, y)を少なくとも１つ通って点(p, q)に達する経路の集合をS_{Bad}とする。
- 求めたい経路の総数は、Oから(p, q)に達する経路の総数から$|S_{Bad}|$を引いた値。
- S_{Bad}に属する経路Πが最初に$y < x - d$を満たす点(x, y)を通るとき、その点は$(v + d + 1, v)$と書ける$(0 \leq v < p - d)$。$y = x - d$上の$(v + d, v)$から東へ1 進むことで$(v + d + 1, v)$へ進む。



問題１１　べこべこの攻略

考察：問題の解法 (5)

- 経路Πの$(v+d+1, v)$から(p, q)までの範囲を、$(v+d+1, v)$を通り直線$y = x - d$と平行な直線を軸に反転した経路をΠ′とする。
- Πでは、$(v+d+1, v)$から(p, q)に達するまでに東に$p-v-d-1$、北に$q-v$進む。$(v+d+1, v)$以降を反転すると東に$q-v$、北に$p-v-d-1$進むので、経路Π′はvに関わらず点$(q+d+1, p-d-1)$に到達する



問題１１　べこべこの攻略

考察：問題の解法 (6)

- 経路の集合S_{Bad}と、原点Oから$(q+d+1, p-d-1)$までの経路の集合の間には1対1対応が存在する（証明はカタラン数の求め方と同様）。
- 以上より、原点Oから出発し、$y \geq x-d$の領域だけを通って点(p,q)に達する経路の総数は以下のように求まる。

(原点Oから出発し、$y \geq x-d$の領域だけを通って点(p,q)に達する経路の総数)
$=$ (Oから(p,q)までの経路の数) $- S_{Bad}$
$=$ (Oから(p,q)までの経路の数) $-$ (Oから$(q+d+1, p-d-1)$までの経路の数)
$= {}_{p+q}C_{p} - {}_{p+q}C_{q+d+1}$
$= \dfrac{(p+q)!}{p!\,q!} - \dfrac{(p+q)!}{(q+d+1)!\,(p-d-1)!}$

- 領域(A)を通る経路の総数は$p = H-a-c$、$q = H-K-b+c$、$d = K-a+b-2c$で、領域(B)を通る経路の総数は$p = c$、$q = K-c$、$d = 0$で得られる。

問題１１　べこべこの攻略

考察：問題の解法 (6)

- 原点Oから$(c, K-c)$を通った後、東へa、北へb進み、最後に(H, H)に達するような、$y \geq x$の領域だけを通る経路の総数を$P^{c}_{H,K,a,b}$とすると、$P^{c}_{H,K,a,b}$は以下の式で表せる。

$$\frac{(K-2c+1)(K-2c+b-a+1)}{(K-c+1)(H-c-a+1)} \cdot \frac{K!}{c!\,(K-c)!} \cdot \frac{(N-K-a-b)!}{(H-c-a)!\,(H-K+c-b\,)!}$$

- 以上より、この問題の答は以下の式で表せる。

$$P^{\max(0,K-H+b)}_{H,K,a,b} + P^{\max(0,K-H+b)+1}_{H,K,a,b} + \cdots + P^{\left\lfloor \frac{K-a}{2} \right\rfloor}_{H,K,a,b}$$

問題１１　べこべこの攻略

実装

- 答はとても大きな数になるので、$998{,}244{,}353$で割った余りを出力する（以下では、$P = 998{,}244{,}353$とする）。
- 導かれた答の式から、以下の値の計算が必要。
  - 0からNまでの階乗をPで割った余り
  - 0からNまでの$\mathrm{mod}\ P$での逆元（整数xの$\mathrm{mod}\ P$での逆元yは、$(x \times y)\%\ P = 1$を満たす整数）
  - 0からNまでの階乗の$\mathrm{mod}\ P$での逆元
- iの階乗をPで割った余りをfac[i]、iの$\mathrm{mod}\ P$での逆元をinv[i]、iの階乗の$\mathrm{mod}\ P$での逆元をfinv[i]とすると、以下のコードで計算量$O(N)$で計算できる。

```
fac[0] = fac[1] = inv[1] = finv[0] = finv[1] = 1;
for  (int i=2; i<=N; i++)  {
   fac[i] = fac[i-1] * i % P;   inv[i] = P – inv[P % i] * (P / i) % P;   finv[i] = finv[i-1] * inv[i] % P;
}
```

問題１１　べこべこの攻略

実装と計算量

- fac[i]、inv[i]、finv[i]を使って、以下の式を計算する。

$$P_{H,K,a,b}^{\max(0,K-H+b)} + P_{H,K,a,b}^{\max(0,K-H+b)+1} + \cdots + P_{H,K,a,b}^{\left\lfloor \frac{K-a}{2} \right\rfloor}$$

+は、加算の結果を$998{,}244{,}35$で割った余りを求める演算

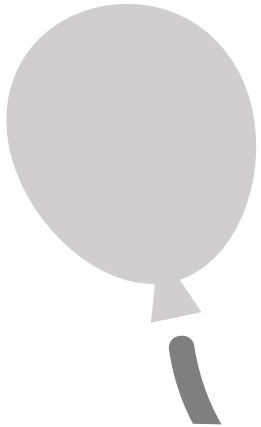
ただし、

$$P_{H,K,a,b}^{c} = \frac{(K-2c+1)(K-2c+b-a+1)}{(K-c+1)(H-c-a+1)} \cdot \frac{K!}{c!\,(K-c)!} \cdot \frac{(N-K-a-b)!}{(H-c-a)!\,(H-K+c-b\,)!}$$

- $P_{H,K,a,b}^{c}$は、iの階乗にfac[i]、 iの逆数にinv[i]、 iの階乗の逆数にfinv[i]を使い、$998{,}244{,}353$を法とした乗算で計算する（xとyのPを法とした乗算は $(x \times y)\%\, P$）。
- 上の式の計算量は$O(N)$、 fac[]、inv[]、finv[] の計算量も$O(N)$なので、この問題の計算量は $O(N)$。

問題１２　荷物管理　　（１４点）

正答数:　　５チーム



問題１２　荷物管理

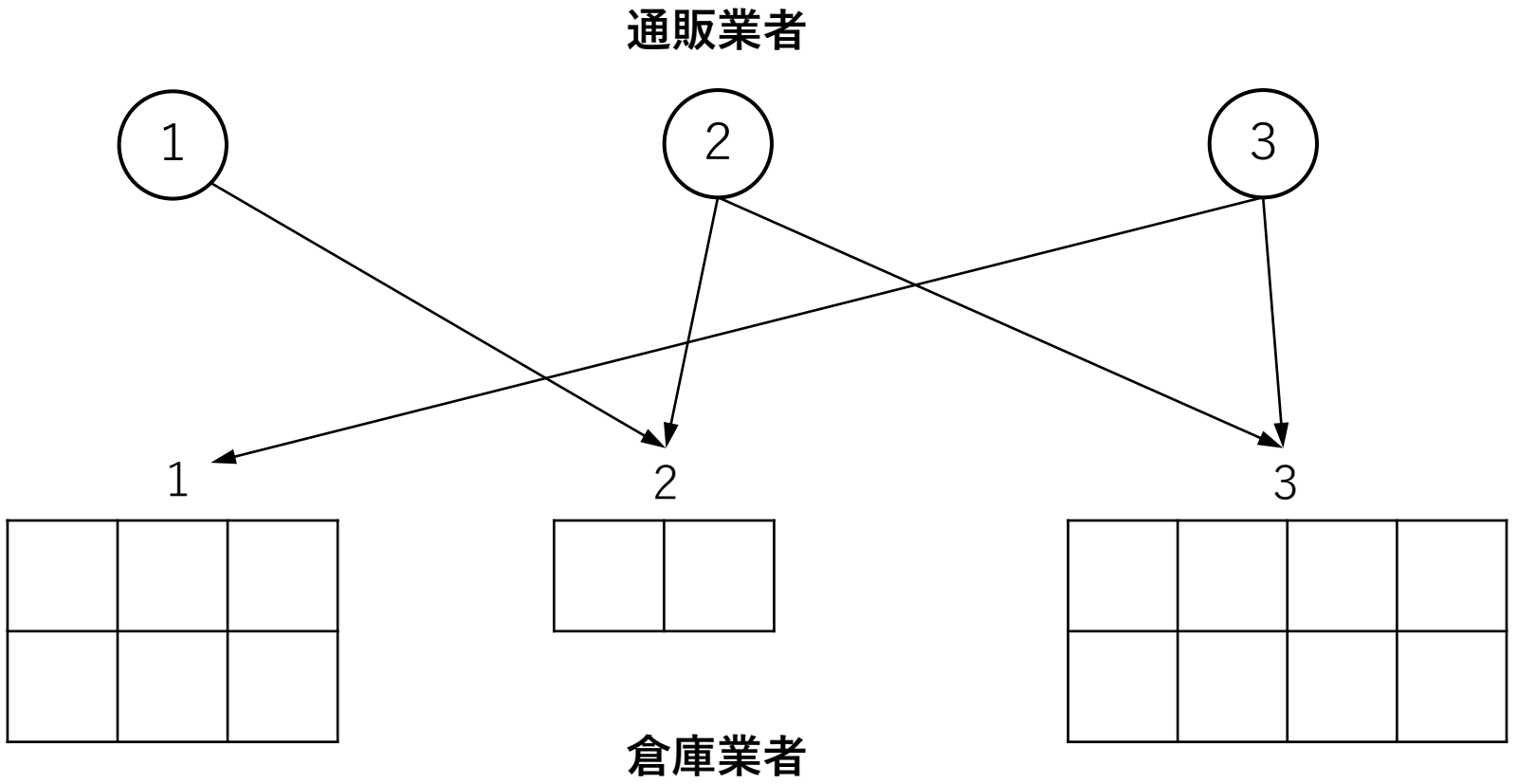
概要

- 通販業者がN社と、倉庫業者がM社ある
- 各通販業者は、いくつかの倉庫業者と荷物の保管に関する契約を交わしている
- i番目の倉庫業者が倉庫に保管できる荷物の数a_iが決まっており、その数までは荷物を受け取ることができるが、制限を超える場合には超えた分の荷物は受け取ることはできない
- i日目に、いずれか１社の通販業者が、契約を交わしたすべての倉庫業者にd_i個の荷物を送る
- １日目からQ日目までの各日について、その日に荷物を送った通販業者から倉庫業者が受け取った荷物の合計を求める
- $1 \leq N, M \leq 200{,}000$、 $1 \leq Q \leq 200{,}000$
- $1 \leq a_i \leq 10^9$、$1 \leq d_i \leq 10^9$
- ただし、i番目の通販業者が契約を交わしている倉庫業者の数$k_i (1 \leq k_i \leq M)$とすると、$k_1 + k_2 + \cdots + k_N \leq 200{,}000$をみたす
- 時間制限３秒

問題１２　荷物管理

入出力例

入力例	出力例
3 3 **6 2 8** **1 2** **2 2 3** **2 1 3** 4 1 1 2 2 3 4 1 1	1 3 8 0



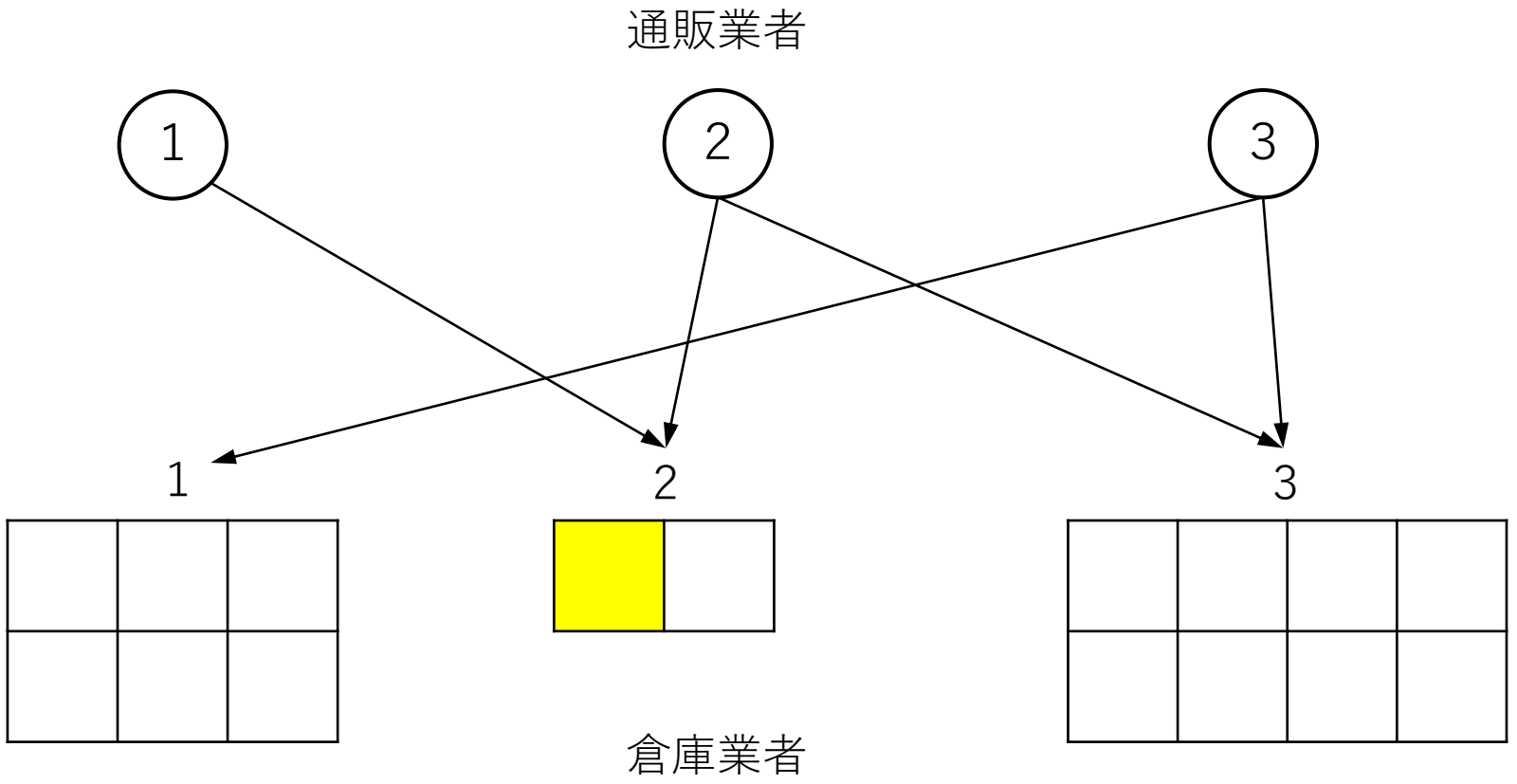
問題１２　荷物管理

入出力例

入力例	出力例
3 3 6 2 8 1 2 2 2 3 2 1 3 4 **1 1** 2 2 3 4 1 1	**1** 3 8 0

出力：１

１日目：通販業者１が１個



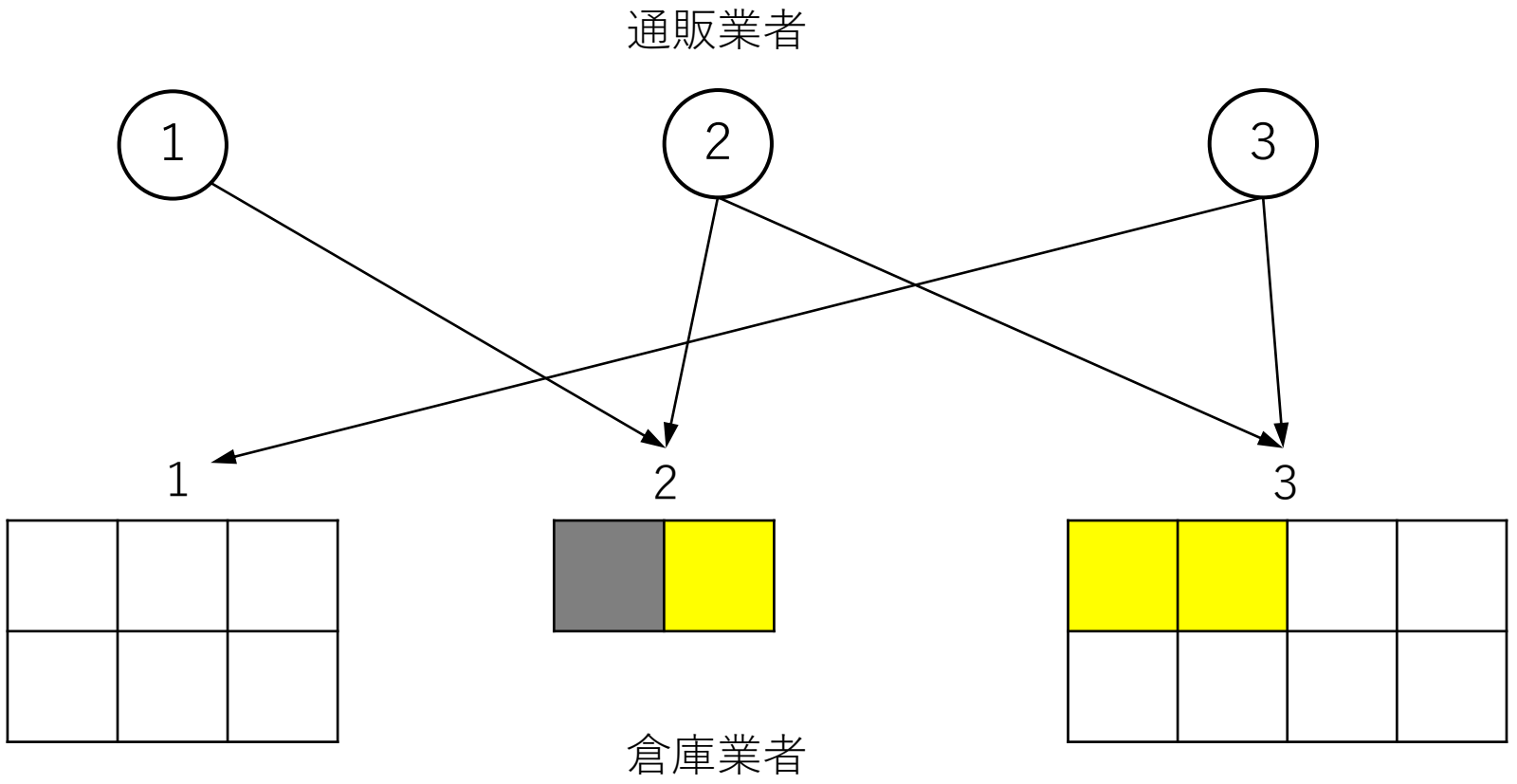
問題１２　荷物管理

入出力例

入力例	出力例
3 3 6 2 8 1 2 2 2 3 2 1 3 4 1 1 **2 2** 3 4 1 1	1 **3** 8 0

出力：３

２日目：通販業者２が２個



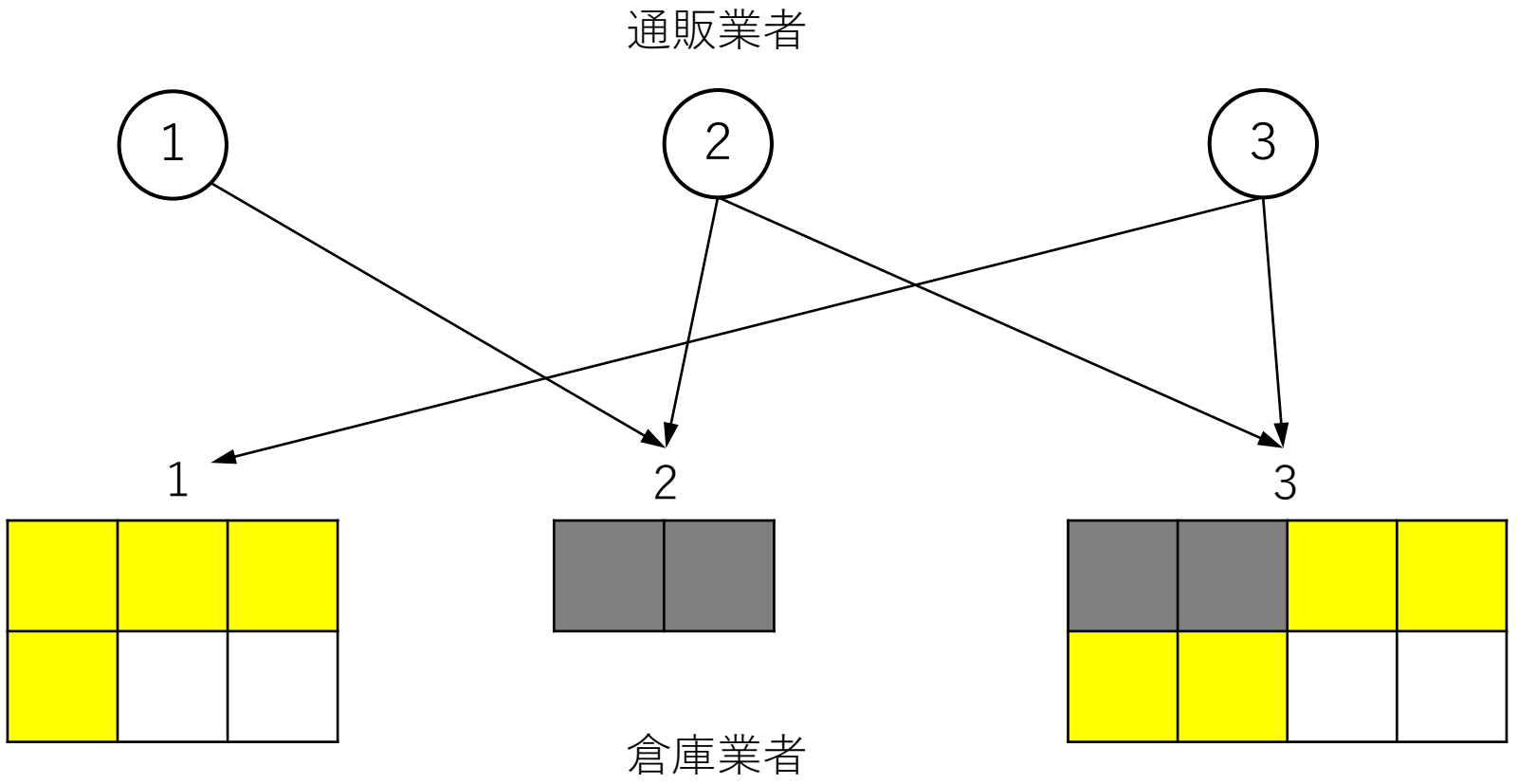
問題１２　荷物管理

入出力例

入力例	出力例
3 3 6 2 8 1 2 2 2 3 2 1 3 4 1 1 2 2 **3 4** 1 1	1 3 **8** 0

出力：８

３日目：通販業者３が４個



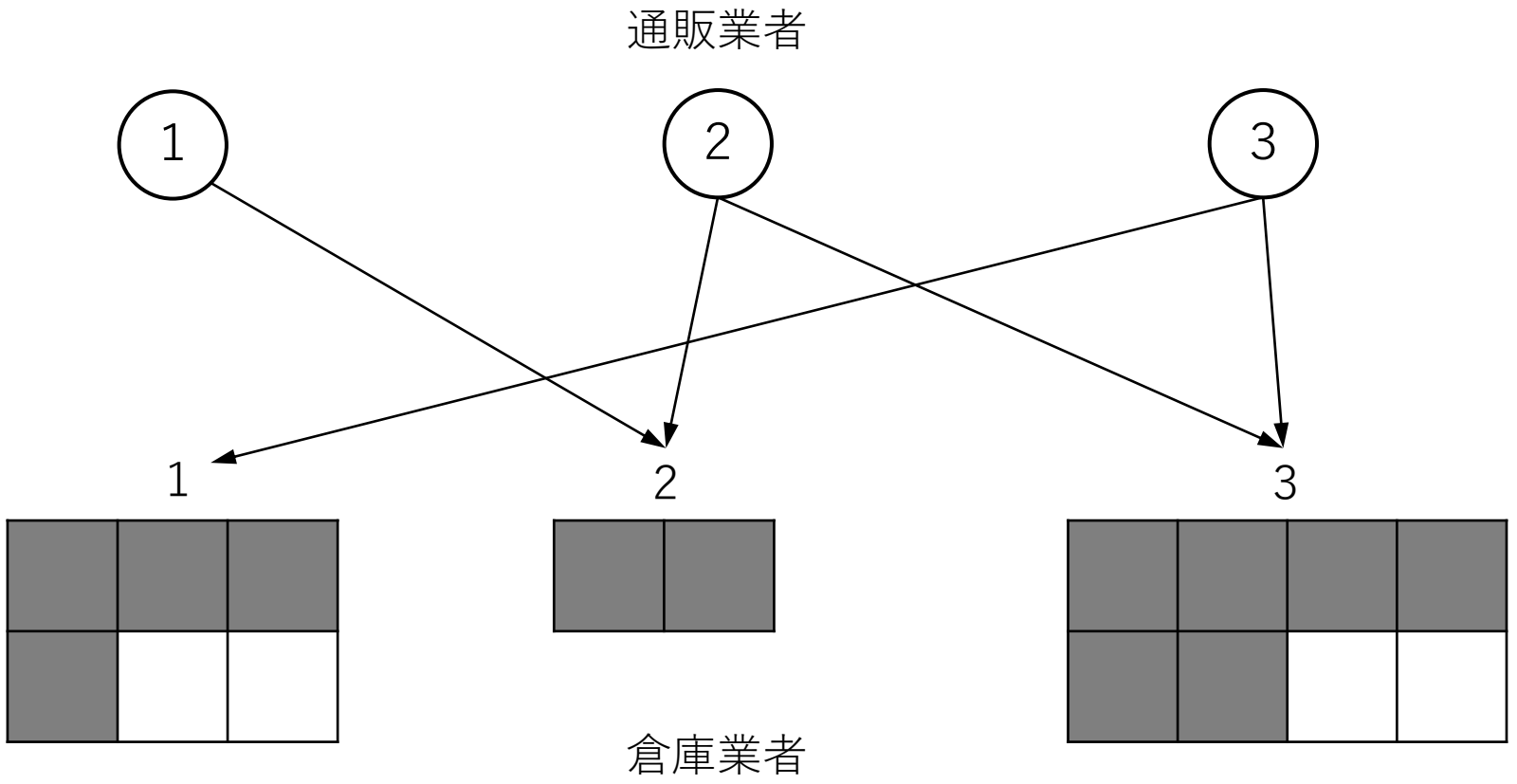
問題１２　荷物管理

入出力例

入力例	出力例
3 3 6 2 8 1 2 2 2 3 2 1 3 4 1 1 2 2 3 4 **1 1**	1 3 8 **0**

出力：０

４日目：通販業者１が１個



問題１２　荷物管理

考察

- $1 \leq N, M \leq 200{,}000$なので、各通販業者は$M$個の倉庫業者と契約できそうだが、$i$番目の通販業者が契約を交わしている倉庫業者の数を$k_i$すると、$k_1 + k_2 + \cdots + k_N \leq 200{,}000$をみたす
- 愚直にシミュレーションすると最悪$\mathrm{O}(NM)$程度になりえる

問題１２　荷物管理

解法

A）各倉庫業者iごとに、いつ総受け取り量が容量a_iを超えるかを二分探索で求めておく

- それより前の配送では、常に丸々d受け取れる
- ある日xに端数が出た場合、その分だけ受け取る、または、ある日xにちょうど満杯になる（そのような日は高々１日だけ）
- その日xの以後は荷物を全く受け取れなくなる

B）通販業者iごとに、q日目の時点ですべて受け取れる契約倉庫の数$f_{i,q}$を管理（端数があれば、それも求めておく）

C）q日目に通販業者t_qがd_q個の荷物を送る → $d_q f_{t_q,q} + r_{t_q,q}$を出力

問題１２　荷物管理

解法

A）各倉庫業者iごとに、いつ総受け取り量が容量a_iを超えるかを二分探索で求めておく

- 各通販業者jごとに、k回目の配送までに何個の荷物を送ったか累積和$sumD[j][k]$を求めておく
- 各倉庫業者iにおいて、x日目の配送で倉庫の容量が超えるかどうかは倉庫iが契約しているすべての通販業者について$sumD$を加算する
- $S =$ 倉庫業者iと契約している通販業者の集合とする
- このとき、各通販業者jが配送する回数をq_jとすると、計算量$O\left(\sum_{s \in S} \log q_s\right) \leq O(\deg(i) \log Q)$程度
- $E = k_1 + k_2 + \cdots + k_N$とすると、全倉庫について合計で$O(E\ (\log Q)^2)$.

問題１２　荷物管理

解法

B）通販業者iごとに、q日目の時点ですべて受け取れる契約倉庫の数$f_{i,q}$を管理（端数があれば、それも求めておく）

- 各倉庫iについて、丸々d受け取れる最後の日lは二分探索で求めてある
- $f_{i,1}$に+1しておく、 $f_{i,l+1}$に-1しておく(imos法)
- 全倉庫に対して処理を終えたら、累積和を計算する(imos法)

問題１２　荷物管理

計算量

A）各倉庫業者iごとに、いつ総受け取り量が容量a_iを超えるかを二分探索で求めておく

- $\mathrm{O}(E\ (\log Q)^2)$

B）通販業者iごとに、q番目のクエリの時点ですべて受け取れる契約倉庫の数$f_{i,q}$を管理

- $\mathrm{O}(E\ \log Q)$

C）q日目に通販業者t_qがd_q個の荷物を送る

- $\mathrm{O}(\mathrm{Q})$

- 総計算量 $\mathrm{O}(E\ (\log Q)^2 + Q)$

問題13　通学路のアクセス事情（14点）

正答数:　　0 チーム

問題13　通学路のアクセス事情

概要

- N頂点の木が与えられる。各頂点iには整数のパラメータa_iが定められている。
- 木のi番目の辺は頂点p_iと頂点$i+1$を結ぶ無向辺であり、パラメータc_iが定められている。
- この木に関するQ個のクエリを処理せよ。
- i番目のクエリではj, xが与えられるのでc_jを xに変更する。この変更は恒久的である。その後、以下の問題の答えを出力せよ。
  - 頂点$1, 2, .., N$にそれぞれ$\mathrm{a}_1, \mathrm{a}_2, \ldots, \mathrm{a}_\mathrm{N}$ 人がいる。配列Dを$d_i \leftarrow c_i$として初期化する。あなたは0回以上以下の操作を繰り返す。
    1. 頂点vであって人が一人以上いるものを選ぶ
    2. vと接続する辺iであって$d_i > 0$であるものを一つ選ぶ
    3. vにいる人を一人辺iと接続している頂点に移動させる
    4. $d_i \leftarrow d_i - 1$とする
  - 頂点1に存在する人数を最大化し、その人数を出力せよ。

制約

- $2 \leq N \leq 200000,\ 1 \leq Q \leq 200000,\ 1 \leq a_i, c_i, x \leq 1000000000$

問題13　通学路のアクセス事情

考察

- 頂点1へ離れる方向へ人を移動させるメリットはまったくないので、グラフは有向グラフであり(子から親に向けて有向辺が張られている)、辺の向きの方向にのみ人が移動すると考えてよい。
  - ➢ 頂点1を根とした根付き木として考えることができる

愚直解

- クエリを答える際は、この根付き木に関する動的計画法を適用することで$O(N)$で計算することができる
  - ➢ 全体で$O(NQ)$のため**時間制限**

問題13　通学路のアクセス事情

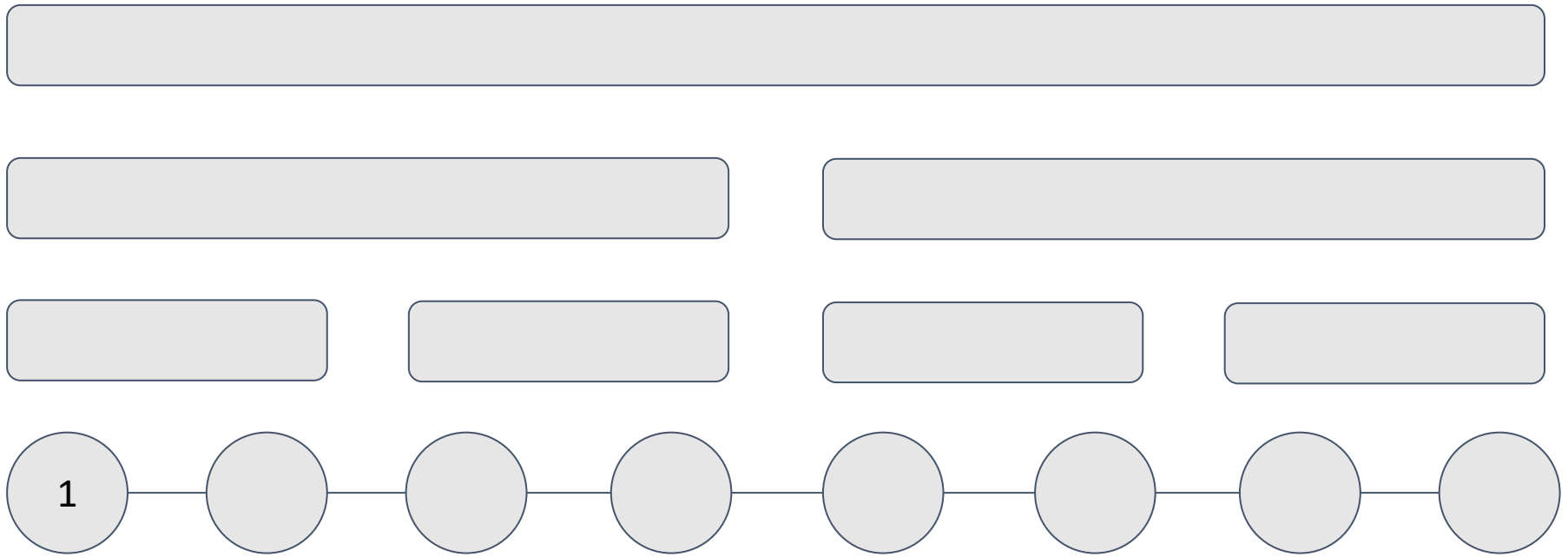
解法

- 重軽分解（Heavy-Light Decomposition、HL分解）で得られたHeavy Pathをセグメント木で管理する

問題13　通学路のアクセス事情

パスグラフ上のセグメント木

- 各ノードについて、そのノードに対応している頂点のみを考慮したとき、一番左の頂点に何人集めることができるか？という問題を考える。



問題13　通学路のアクセス事情

セグメント木：　ノードに持たせるデータ

- $head$: そのノードが管理している区間の左端の頂点番号
- $tail$: そのノードが管理している区間の右端の頂点番号
- ans: そのノードが管理している頂点に関して人の移動を行い、 $head$に集めることができる最大の人数
- $remain$: ansを達成するように人の移動を実際に行ったあと、頂点$tail$に人をいくらでも追加できるとしたら、 ansをどれだけ増やせるか?

問題13　通学路のアクセス事情

セグメント木：　葉ノードの計算

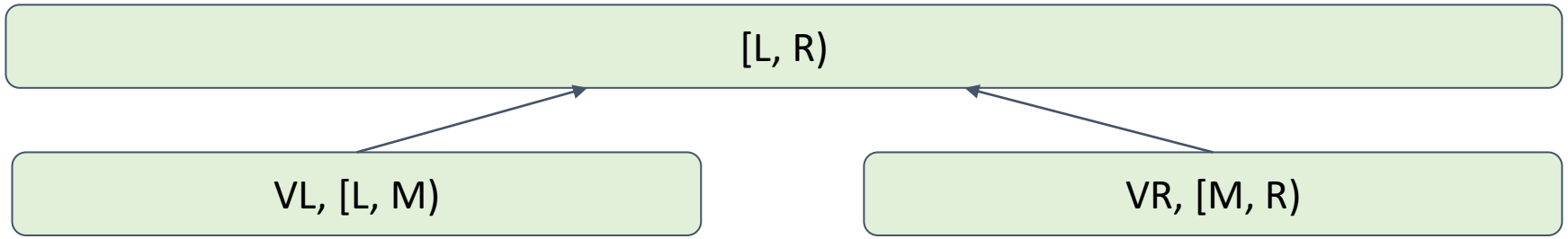
セグメント木の葉は一つの頂点に対応した解を持っている。このノードに関するデータの計算方法は以下の通り：

- $head = i$
- $tail = i$
- $ans = a_i$　　# はじめ頂点iにa_i人いて、(一頂点のみだから)人の移動が発生しないため
- $remain = \infty$　　# 頂点$tail = i$に人を追加すると、追加した分だけansが増えるから

問題13　通学路のアクセス事情

セグメント木：　ノードのマージ

区間[L, M)、区間[M, R)に対応するノードVL, VRの値が計算済だと仮定して、区間[L, R)に対応するノードの値を計算する

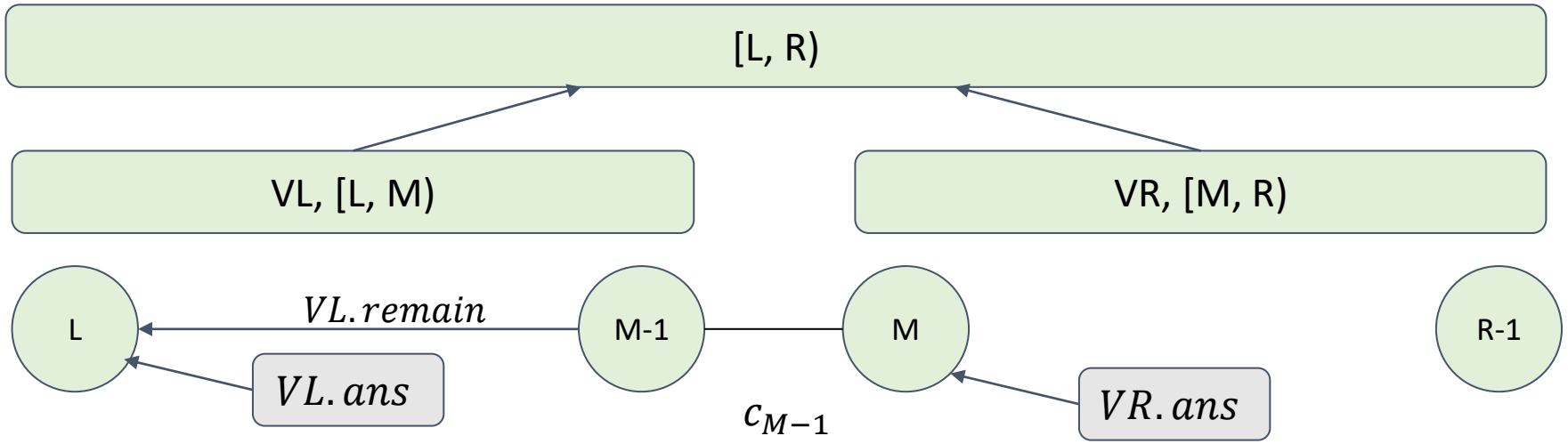


- $head = VL.head$
- $tail = VR.tail$

問題13　通学路のアクセス事情

セグメント木：　ノードのマージ

区間[L, M)、区間[M, R)に対応するノードVL, VRの値が計算済だと仮定して、区間[L, R)に対応するノードの値を計算する

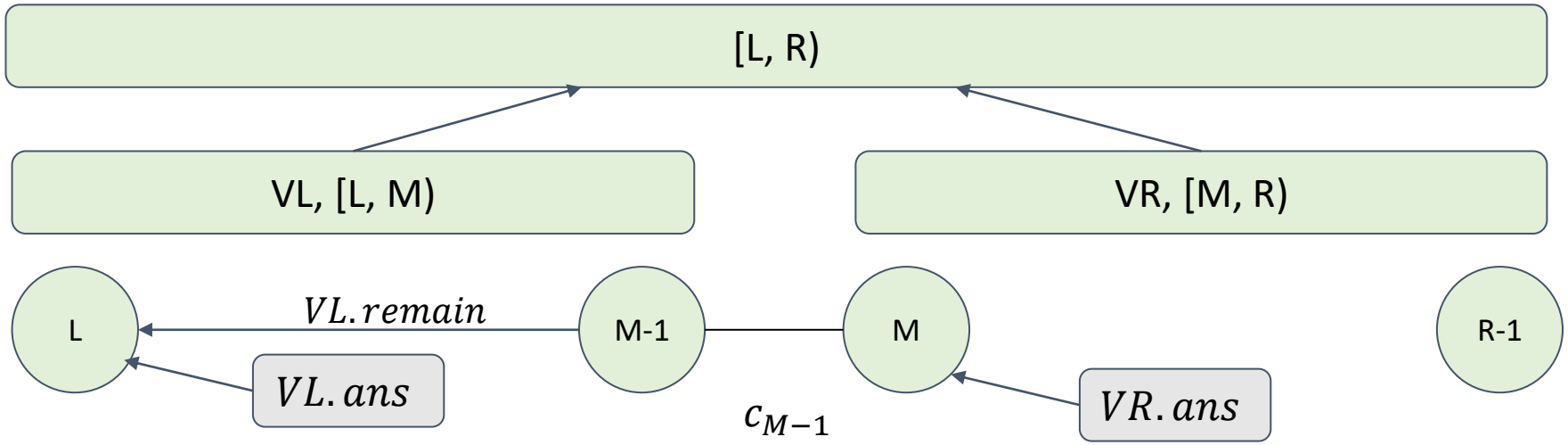


- 頂点Mに現在$VR.ans$人いる。ここで頂点$M-1$に移すことのできる人数は$\min(VR.ans, c_{M-1})$人である。よって、頂点Mから頂点Lに移動させることができる人数は$\min(\min(VR.ans, c_{M-1}), VL.remain)$人である。この人数を$mv$とする

問題13　通学路のアクセス事情

セグメント木：　ノードのマージ

区間[L, M)、区間[M, R)に対応するノードVL, VRの値が計算済だと仮定して、区間[L, R)に対応するノードの値を計算する

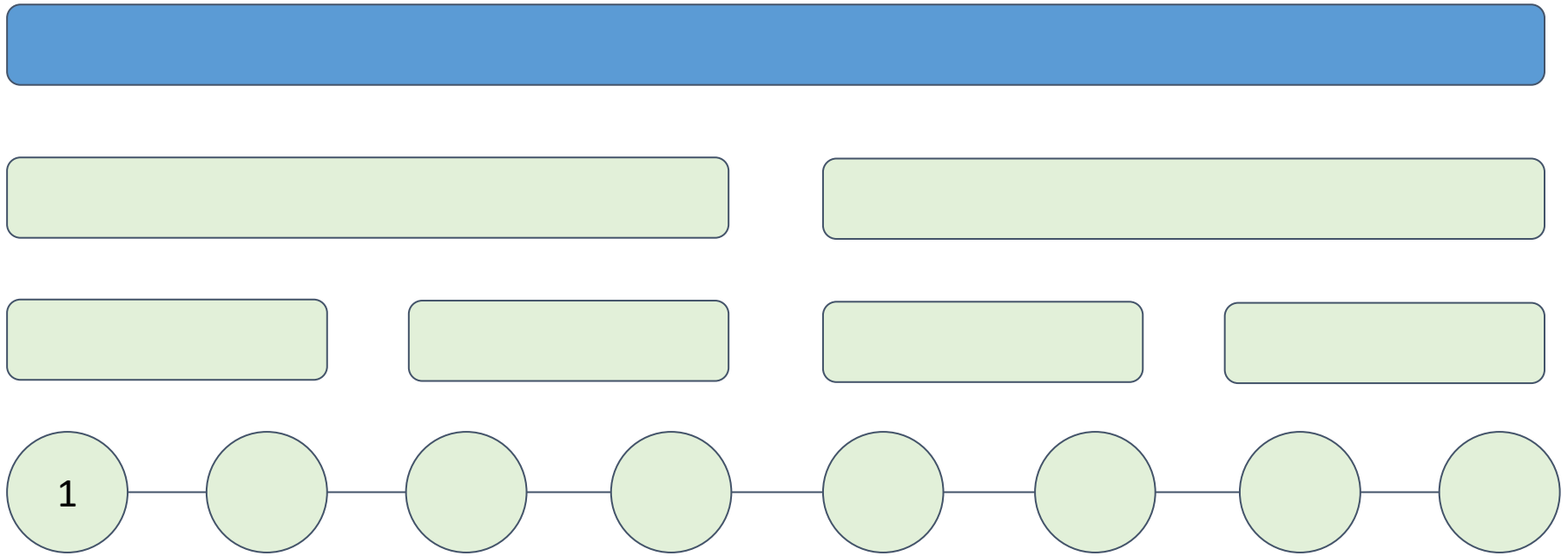


- $ans \leftarrow VL.ans + mv$
- $remain \leftarrow \min(VL.remain - mv, c_{M-1} - mv, VR.remain)$
  - mv人通る辺はその分耐久度が減少する
  - VR側の辺は新たに人が移動することが無いのでそのまま

問題13　通学路のアクセス事情

セグメント木：　解の計算

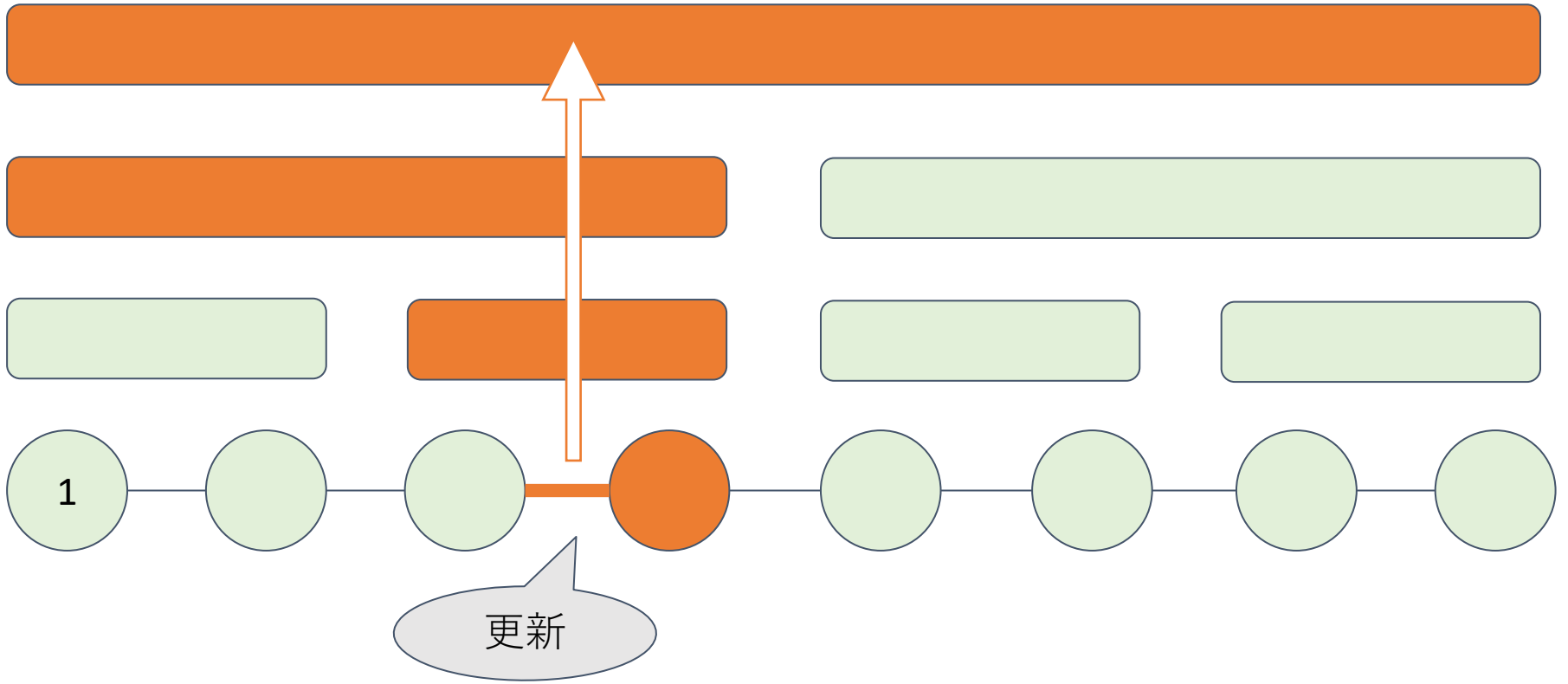
根ノードを参照



問題13　通学路のアクセス事情

セグメント木：　c_iの更新

更新された辺の前または後に対応するノードと、その祖先を再計算



問題13　通学路のアクセス事情

パスグラフ上のセグメント木：計算量

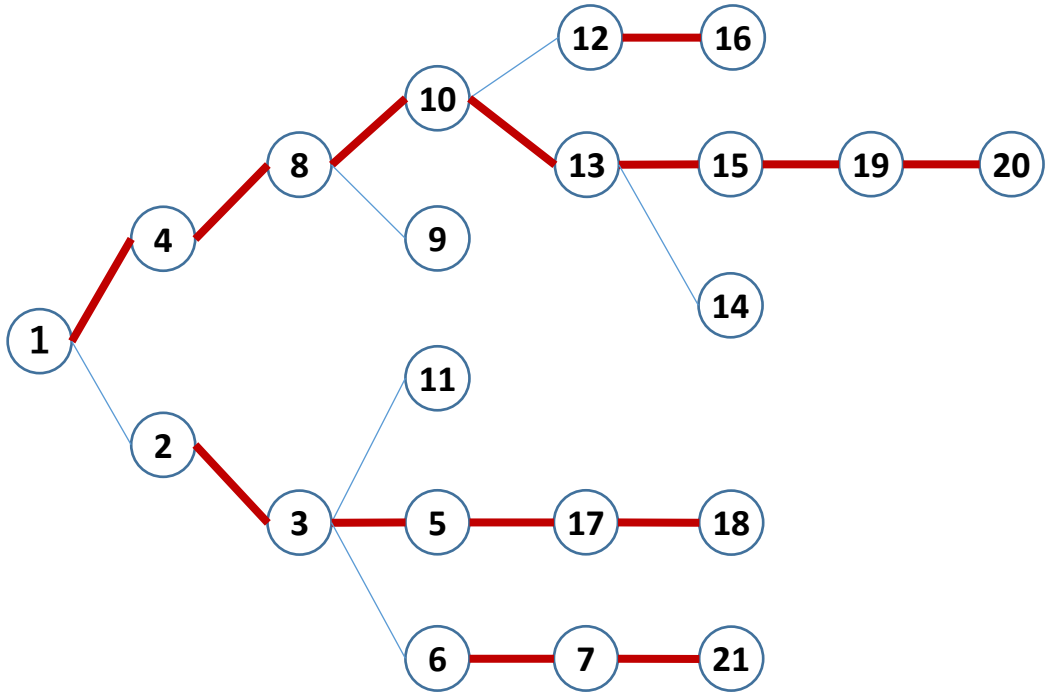
- 葉ノードの計算・ノードのマージ：$O(1)$
- 解の計算：$O(1)$
- 辺の更新：$O(\log N)$
- 計：$O(N\ +\ Qlog\ N)$

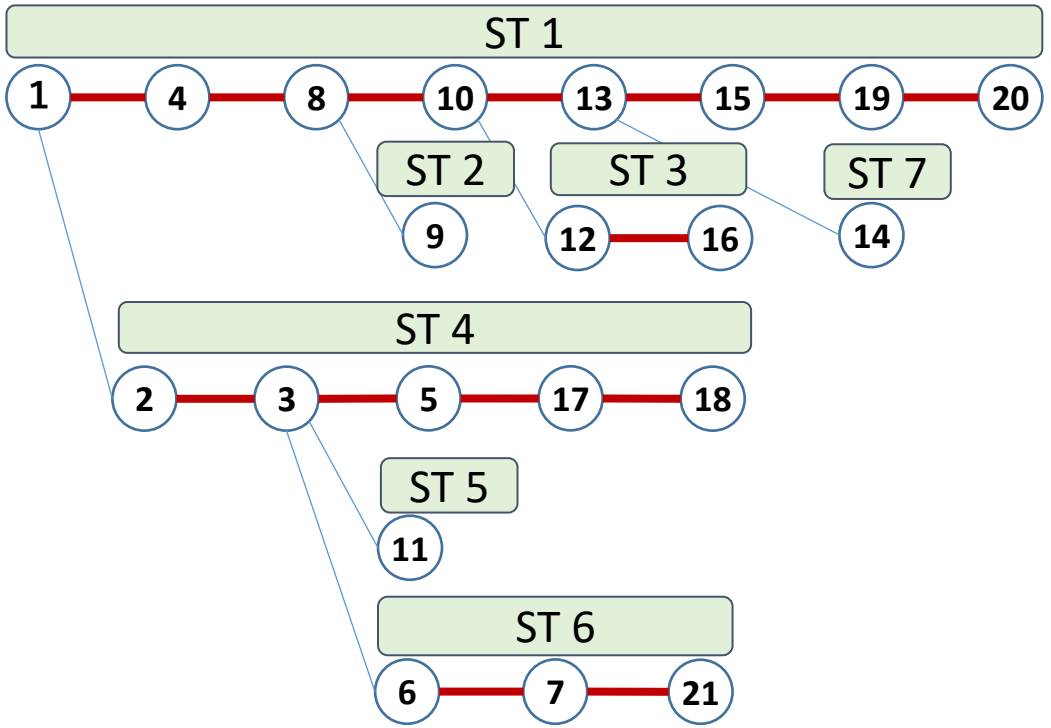
問題13　通学路のアクセス事情

HL分解

- Heavy Pathそれぞれについてセグメント木で解を管理。
- 答えは頂点1があるHeavy Pathのセグ木の根ノードとする

➢ 他のHeavy Pathにある頂点からの寄与を計算する必要がある





問題13　通学路のアクセス事情

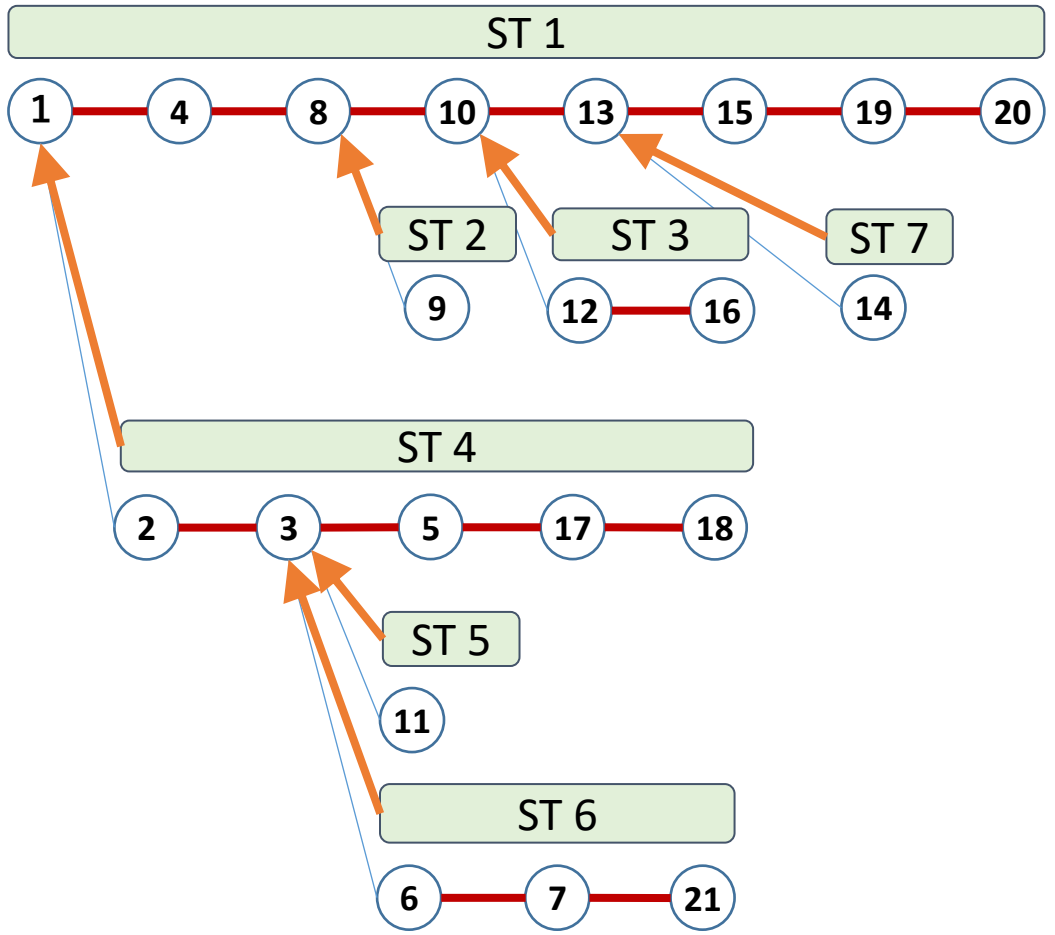
HL分解：他のHeavy Pathにある頂点からの寄与を計算

他のセグメント木（ST）の寄与だけ、あらかじめaに加算しておく方法で解決:

- min(c_{11}, ST 3の解)をa_{10}に加算
- min(c_8, ST 2の解)をa_8に加算
- min(c_{10}, ST 5の解)をa_3に加算
- min(c_4, ST 6の解)をa_3に加算
- min(c_1, ST 4の解)をa_1に加算
- min(c_{13}, ST 7の解)をa_{13}に加算

✓ この前処理は$O(N)$で計算してもよい

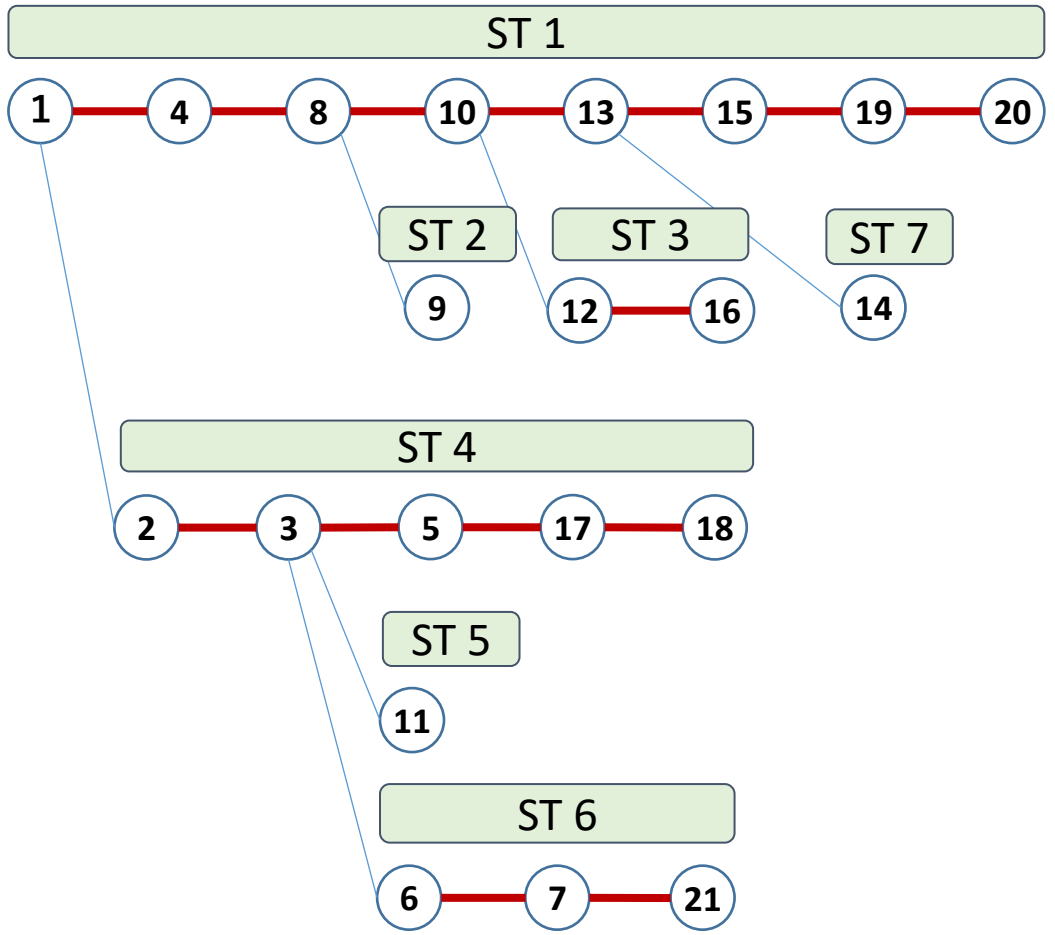
➢ ST 1の解が木全体の解と一致する



問題13　通学路のアクセス事情

HL分解：辺の更新

1. 変更されるc_iが影響する可能性のある寄与を全部打ち消す
2. c_iを変更する
3. 打ち消した寄与を元に戻す



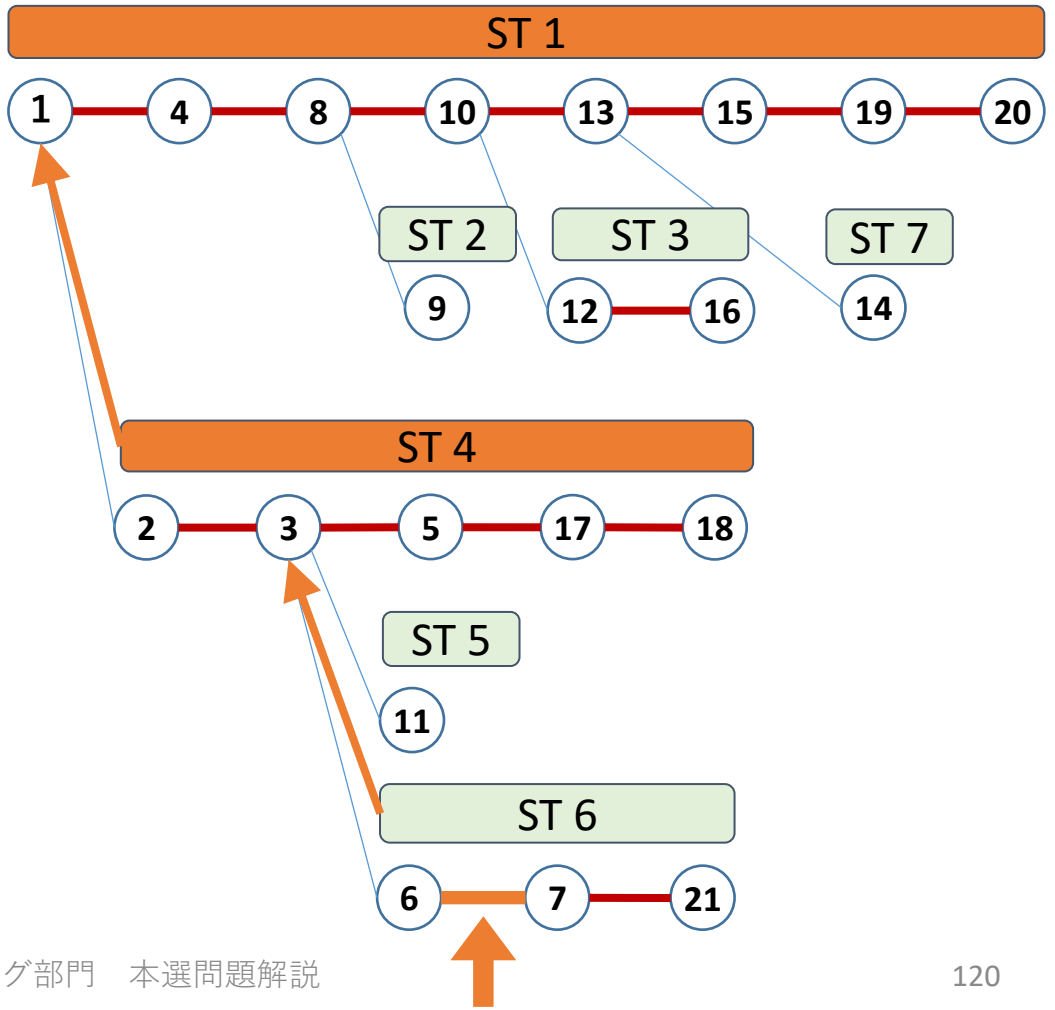
問題13　通学路のアクセス事情

HL分解：辺の更新: 1: 変更される辺が影響しうる寄与を消す

頂点1から変更される辺に向かって寄与を打ち消していく。

例: c_6 (頂点6-7間)の変更

1. min(c_1 , ST4の解)をa_1に減算する
2. ST1の頂点1に対応するノードと、その祖先を再計算(step1をST1に反映)
3. min(c_5 , ST6の解)をa_3に減算する
4. ST4の頂点3に対応するノードと、その祖先を再計算(step3をST4に反映)



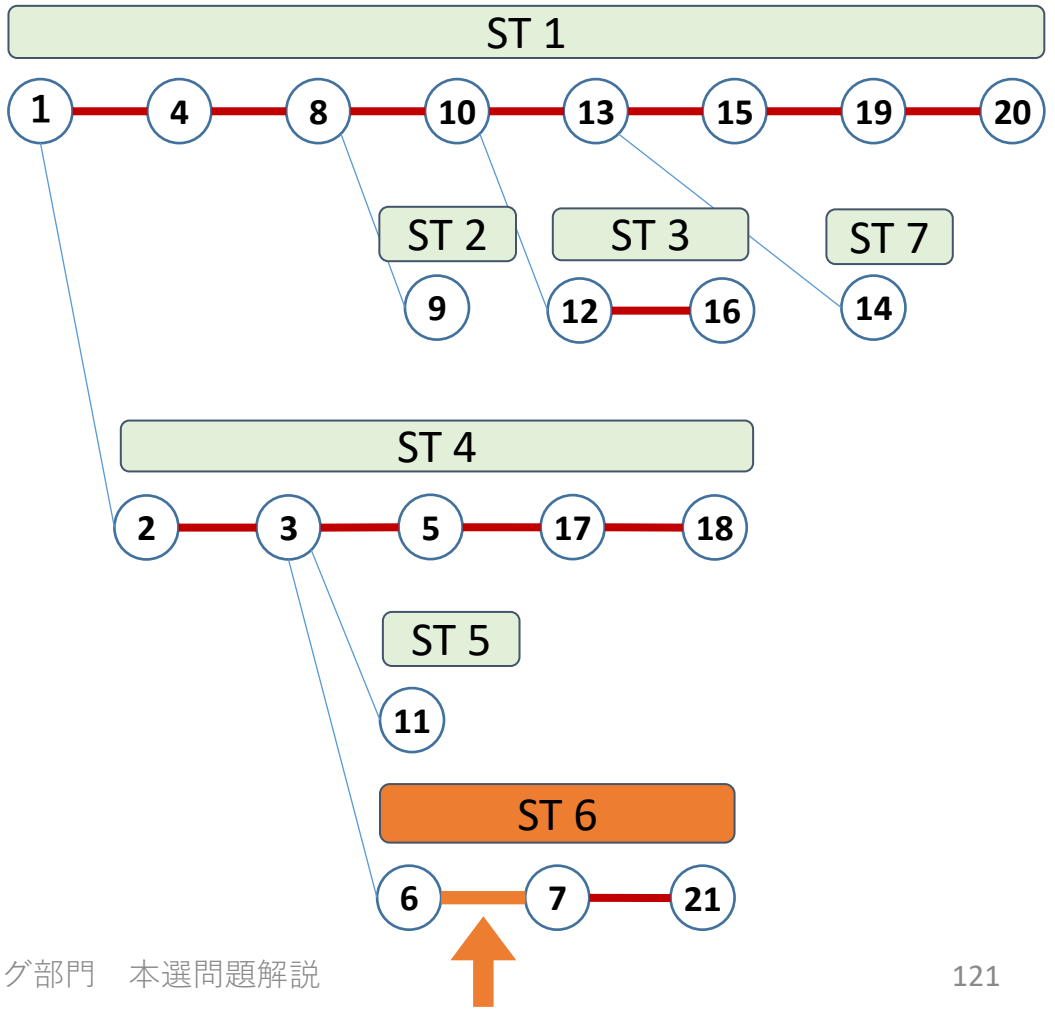
問題13　通学路のアクセス事情

HL分解：辺の更新: 2: c_iを変更

c_iを変更して、対応するSTのノードを再計算

例: c_6 (頂点6-7間)の変更

- ST6の頂点7とその祖先ノードを再計算



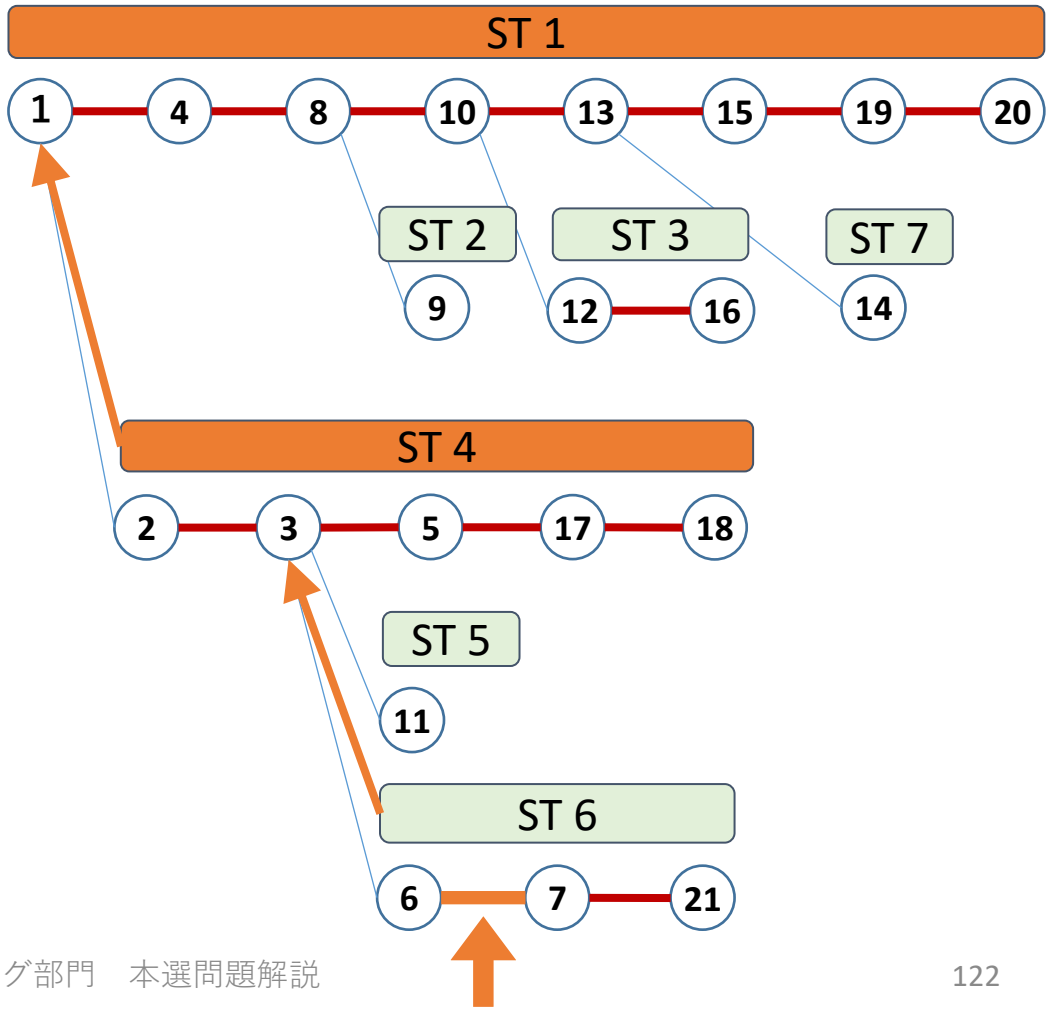
問題13　通学路のアクセス事情

HL分解：辺の更新: 3: 寄与を戻す

寄与を消したときとは逆順で、根へ上る方向で計算。

例: c_6 (頂点6-7間)の変更

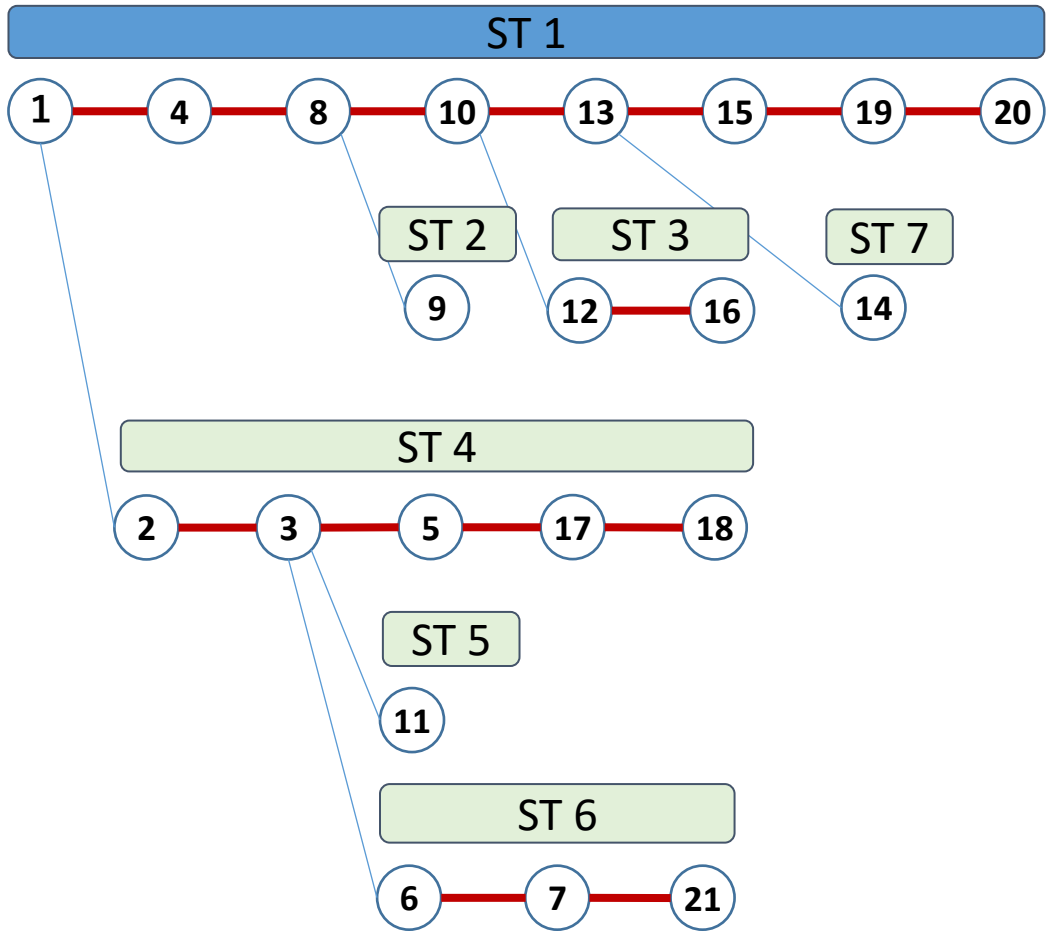
- min(c_5, ST6の解)をa_3に加算
- ST4の頂点3とその祖先を再計算(反映)
- min(c_1, ST4の解)をa_1に加算
- ST1の頂点1とその祖先を再計算(反映)



問題13　通学路のアクセス事情

HL分解：解の計算

頂点1に対応するHeavy Pathのセグメント木の根に解がある



問題13　通学路のアクセス事情

計算量

- 任意の頂点vについて、パス$1-v$間にあるHeavy Pathの種類数は$O(\log N)$　(証明略)
- セグメント木全体の頂点数はN頂点→ノード数の合計も$O(N)$
- 辺更新の際に行うセグメント木の操作の回数はHeavy Pathの種類数のみによるので、$O(N + Qlog^2N)$

ご視聴ありがとうございました。
また来年2026で!!