パソコン甲子園2026

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

AIZU
THE UNIVERSITY OF AIZU

渡部 有隆　　鈴木 大郎　　奥山 祐市　　西舘 陽平　　松本 和也



講評

- 全てのチームが１問以上に正解　○
- 全ての問題が解かれた　？
- 全ての問題を解いたチームがいない　？

※凍結時の順位表

2026/11/07 17:15　時点の結果

#	ID	ユーザ	所属	得点	正	誤	F01	F02	F03	F04	F05	F06	F07	F08	F09	F10	F11	F12	F13
1	P06	炙りビン	灘高等学校	64	10	2	1	1	1	1	1	1	1	1		1 -1	1 -1		
2	P05	Luster	東大寺学園高等学校	42	8	1	1	1	1	1	1	1	1	1 -1		-1	-1		-3
2	P02	Au79	聖光学院高等学校	42	8	1	1	1	1	1	1 -1	1	1	1	-6				
4	P04	Ad Astra	開成高等学校	42	8	2	1	1	1	1	1	1 -2	1	1					
5	P08	higuma	北海道札幌南高等学校	38	7	2	1	1	1	-2	1	1	1	1 -2					
6	P03	Sabotage	筑波大学附属駒場高等学校	36	7	1	1	1	1	1	1	-12	1 -1	1			-1		
7	P16	競プロerのひとり言	茨城県立並木中等教育学校	36	7	3	1	1	1	1	1		1	1 -3					
8	P01	Quavallers	早稲田高等学校	32	7	5	1	1	1	1 -1	1	1 -4	1			-3			
8	P11	快進の一撃	北海道札幌南高等学校	32	7	5	1	1 -1	1	1	1	1 -4	1			-3			
10	P30	エスパー同好会	大分県立大分上野丘高等学校	26	6	0	1	1	1	1	1	-3	1						
10	P24	zsmd342	愛知県立瑞陵高等学校	26	6	0	1	1	1	1	1	-4	1						
12	P10	5/11	明石工業高等専門学校	26	6	2	1	1	1	1	1 -2	-2		1					
13	P23	RISEI	豊田工業高等専門学校	26	6	3	1 -1	1	1	1 -2	1	-6	1	-4		-1			
14	P15	TwinStream	茨城工業高等専門学校	22	6	0	1	1	1	1	1	1							
15	P21	uranium271	静岡県立静岡高等学校	22	5	1	1	1	1		1 -1	-1	1		-3				
16	P27	三月の月髄は我が手に	兵庫県立大学附属高等学校	16	5	0	1	1	1	1	1	-5							
17	P19	電算機部	長岡工業高等専門学校	16	4	1	1	1 -1	1			-3	1						
17	P17	NullKaiSEr	埼玉県立熊谷高等学校	16	5	1	1	1	1	1	1 -1								
17	P26	サラダP教	大阪教育大学附属高等学校天王寺校舎	16	5	1	1	1	1	1 -1	1	-2							
20	P31	魔法少女M	N高等学校	16	4	3	1	1	1					1 -3		-1			
20	P12	MechaNIKAl	宮城県仙台二華高等学校	16	5	3	1	1 -1	1	1	1 -2								
22	P18	MexPistols	武蔵高等学校中学校	16	5	5	1	1	1	1 -2	1 -3	-6	-4						
22	P22	疲労困憊ルエラー	滝高等学校	16	5	5	1 -1	1 -3	1	1 -1	1	-3							
24	P20	ハロー・ワールド	富山高等専門学校	12	4	5	1 -1	1 -1	1	-1	1 -3	-1							
25	P14	電脳部破壊人	福島県立郡山北工業高等学校	12	4	7	1 -1	1 -2	1 -1		1 -3	-7	-1						
26	P09	CyanAce	静岡県立浜松工業高等学校	10	3	1	1		1		1 -1								
26	P25	Leo	洛南高等学校	10	3	1	1	-2	1		1 -1								
28	P28	ググガガ年金	和歌山県立紀北工業高等学校	10	4	8	1	1 -4	1	1 -4	-3								
29	P07	abcd	広島県立広島国泰寺高等学校	4	2	0	1		1										-1
29	P29	Python信者	徳島県立城ノ内中等教育学校	4	2	0	1		1					-4					
31	P13	絶頂のエゴイスト	福島県立平工業高等学校	1	1	2	1 -2	-9	-5										

問題セット



1 天秤

2 平方差の行方

3 正多角形の重なり

4 マンションの照明

5 合計点からわかること

6 等脚台形



7 支払うことができない金額

8 回文を作ろう

9 機能の選定

10 つながる三人組

11 データセンター増強計画

12 データ伝送

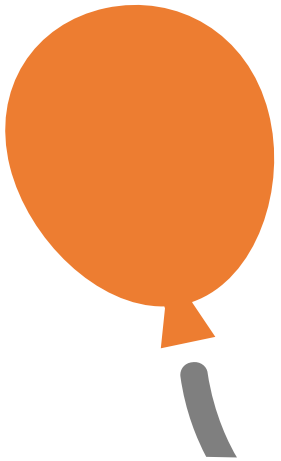
13 植物園の栽培計画

問題セット2026

#	タイトル	分野	得点	難易度 思考	難易度 実装
1	天秤		1	★	☆
2	平方差の行方		2	★☆	★☆
3	正多角形の重なり		3	★★	★☆
4	マンションの照明		4	★☆	★★
5	合計点からわかること		6	★★	★★
6	等脚台形		6	★★★	★★
7	支払うことができない金額		10	★★☆	★★☆
8	回文を作ろう		10	★★★	★★☆
9	機能の選定		10	★★★☆	★★☆
10	つながる三人組		11	★★★☆	★★★
11	データセンター増強計画		11	★★★☆	★★★
12	データ伝送		13	★★★★	★★★☆
13	植物園の栽培計画		13	★★★★	★★★★

問題１　天秤　　（１点）

正答数:　　31 チーム



問題１　天秤

概要

- 重さが $a \leq b \leq c \leq d$ の4個のおもりがある。
- 天秤の左右に2個ずつ載せる。
- 左右の合計の差の絶対値を最小にせよ。

制約

$$1 \leq a \leq b \leq c \leq d \leq 100$$

問題１　天秤

考察：左右への分け方は3通り

（左右を入れ替えても差は変わらない）

a ＋ b	対	c ＋ d
a ＋ c	対	b ＋ d
a ＋ d	対	b ＋ c

最小の差を与えるのは (a + d) と (b + c) の組合せ。

問題１　天秤

考察：答えを直接計算できる

3通りの差を D_1, D_2, D_3 とする（$a \le b \le c \le d$）。

$D_1 = (c + d) - (a + b), \quad D_2 = (b + d) - (a + c)$

$D_1 - D_2 = 2(c - b) \ge 0$　よって　$D_1 \ge D_2$

$x = d - c \ge 0,\ y = b - a \ge 0$ とおくと、

$D_2 = x + y, \quad D_3 = |(a + d) - (b + c)| = |x - y|$

$x + y \ge |x - y|$ より、$D_1 \ge D_2 \ge D_3$

答え $= |(a + d) - (b + c)|$

問題１　天秤

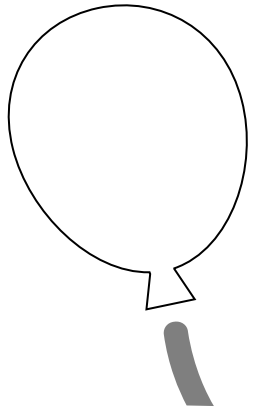
実装例

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int a, b, c, d;
    cin >> a >> b >> c >> d;
    cout << abs((a + d) - (b + c)) << '¥n';
    return 0;
}
```

問題２　平方差の行方　（２点）

正答数:　　26 チーム



問題２　　平方差の行方

概要

- 10進表記でd 桁の整数Nを$a_d a_{d-1} \cdots a_2 a_1$とする。
- Nが1桁の整数になるまで以下の操作を繰り返す
  - $N \coloneqq \left|a_d^2 - a_{d-1}^2\right| + \left|a_{d-1}^2 + a_{d-2}^2\right| + \cdots + \left|a_3^2 - a_2^2\right| + \left|a_2^2 - a_1^2\right|$
- N が1桁の整数になるまで操作の回数を出力する。
- N が1桁にならない場合は-1を出力する。

制約

$1 \leq N \leq 10^{18}$

問題２　　平方差の行方

解法

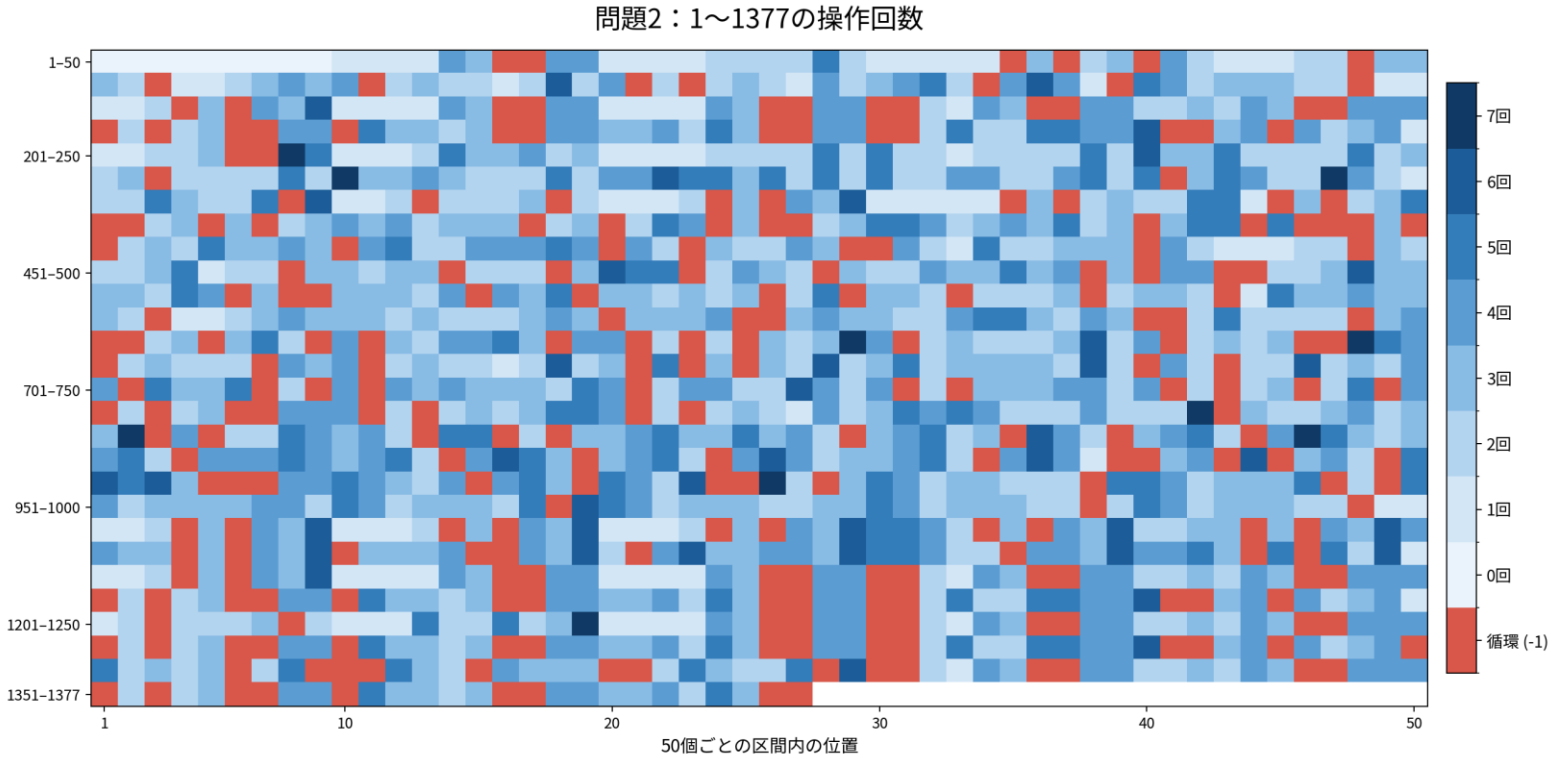
- 「**操作**」**の計算方法**→以降、これを$f(N)$とする
  - Nを10で割った余りが、最下位の桁になる。
  - Nを10で割る→ Nの余りを求める を繰り返すと各桁の値が求まる
  - 求めた桁の数とその前の数の2乗差の絶対値を計算し、
    答えに加算する。
- **ループ解法**
  - 操作を100回～10000回行って1桁になっていない
    →正しいとは言い切れない
  - しかし、解析すると8回で判定できることが分かる
- **循環検出法**
  - 訪れた数字を記憶し、同じものが現れれば循環
  - $n_1 = f(n_1); n_2 = f\big(f(n_2)\big);$ としてn1==n2となれば循環
    (Floydの循環検出)

問題２　　平方差の行方

ループ解法の着眼点

$1 \leq N \leq 10^{18}$　　1回の操作後の最大値は$f(9090909090909090) = 81 \times 17 = 1377$

1~1377までの数であれば、全列挙して性質を調べることができる

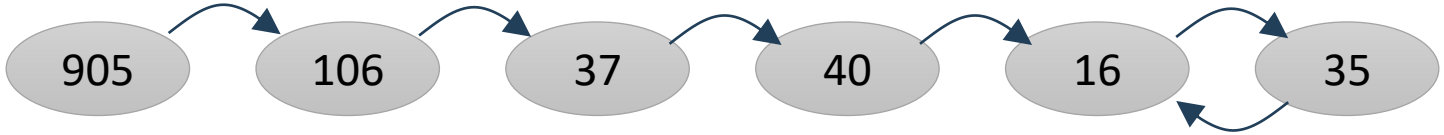


$1 \leq N \leq 1377$の操作の最大値は、
循環を除くと7

$1 \leq N \leq 10^{18}$の操作の最大値は、
循環を除くと８

８回操作して
Nが1桁にならななければ
循環していると判断できる

問題２　平方差の行方



循環とその検出

操作の性質

- 同じ整数に対して操作を行うと、必ず同じ整数が得られる。
- したがって、同じ整数が2回現れた場合、それ以降は同じ整数の並びが繰り返される。これを**循環**と呼ぶ。

循環が必ず検出できる理由

- 1回の操作で、整数は必ず1377以下になる。
- 1桁にならずに操作を繰り返すと、取り得る整数は10以上1377以下の1368種類に限られる。
- したがって、操作後の整数を1369個並べると、必ず同じ整数が2回以上現れる。
- 同じ整数からは必ず同じ整数に進むため、その後は同じ変化を繰り返す。

結論：1桁になるか、循環するかのいずれかである。

問題２　平方差の行方

解法：共通部分

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

long long next(long long n) {
    long long s = 0;
    int a = n % 10;
    n /= 10;
    while (n) {
        int b = n % 10;
        s += abs(a * a - b * b);
        a = b;
        n /= 10;
    }
    return s;
}
```

set版

```
int main() {
    long long n;
    cin >> n;

    set<long long> st;
    int ans = 0;

    while (n >= 10) {
        if (st.count(n)) {
            cout << -1 << '¥n';
            return 0;
        }
        st.insert(n);
        n = next(n);
        ans++;
    }

    cout << ans << '¥n';
    return 0;
}
```

Floydの循環検出版

```
int main() {
    long long n;
    cin >> n;

    long long slow = n, fast = n;
    int ans = 0;

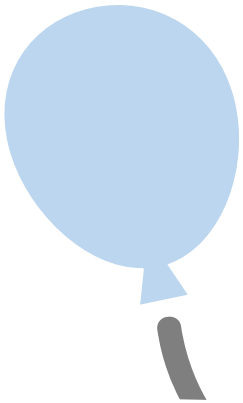
    while (slow >= 10) {
        slow = next(slow);
        fast = next(next(fast));
        ans++;

        if (slow == fast && slow >= 10) {
            cout << -1 << '¥n';
            return 0;
        }
    }

    cout << ans << '¥n';
    return 0;
}
```

問題３　正多角形の重なり（３点）

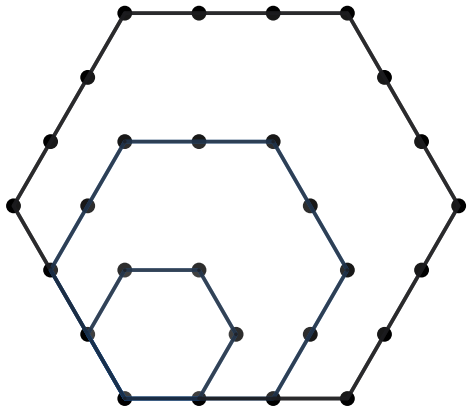
正答数:　　30チーム



問題３　正多角形の重なり

概要

- 同じ長さの棒を使い、1辺がそれぞれ $1, 2, \cdots, K$本の棒からなる K 個の正N 角形を作る。
- これらの正多角形を、1つの頂点と、その頂点につながる2辺が重なるように配置する。
- 棒と棒の継ぎ目が同じ位置に重なる場合は、1つとして数える。
- 継ぎ目の総数を求めよ。



$N = 6, K = 3$のときの図

制約

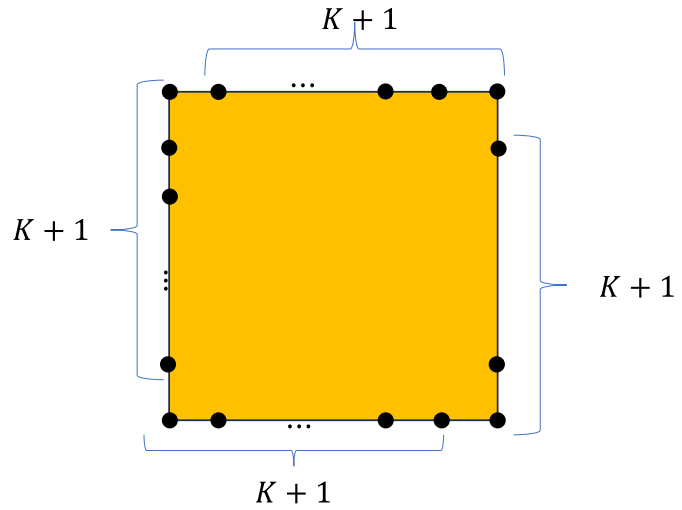
$3 \leq N \leq 50, \quad 1 \leq K \leq 5000$

問題３　正多角形の重なり

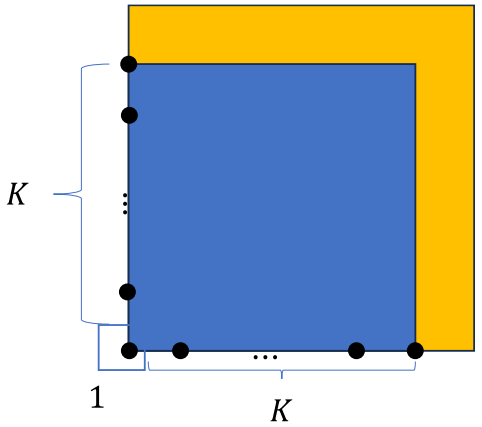
Kを一つ増やした時にどれだけ継ぎ目が増えるかを考える。

=>

｛1辺$K+1$のN角形の継ぎ目の数｝ー｛1辺Kと$K+1$のN角形で重なる継ぎ目の数｝



$K+1$本でN角形を作った時の
継ぎ目の数：$N(K+1)$



重なる継ぎ目の数：$2K+1$

K→K+1としたときに増える継ぎ目の数-> $N(K+1)-(2K+1)$

問題３　正多角形の重なり

解法１：

$a_1 = N$

$a_{i+1} = a_i + \{N(i+1) - (2i+1)\}$

として、 iを1から順にループ

```
#include <stdio.h>

int main(void) {
    long long N, K;
    scanf("%lld %lld", &N, &K);
    long long ans = N;
    for (long long i = 2; i <= K; i++) {
        ans += (N - 2) * i + 1;
    }
    printf("%lld¥n", ans);
    return 0;
}
```

解法２：

a_iの一般項を使う

$$a_i = a_1 + \sum_{i=1}^{K-1} \{N(i+1) - (2i+1)\}$$

$$= \frac{(N-2)K(K+1)}{2} + K + 1$$

```
#include <stdio.h>

int main(void) {
    long long N, K;
    scanf("%lld %lld", &N, &K);
    long long ans = (N - 2) * K * (K + 1) / 2 + K + 1;
    printf("%lld¥n", ans);
    return 0;
}
```

問題４　マンションの照明（４点）

正答数:　　20 チーム

問題４　マンションの照明

概要

- Akabekoマンションの階段の電灯は、１階からN階までの各階に設置されたスイッチを使って、状態を点灯(ON)または消灯(OFF)にすることができる。
- 奇数個のスイッチが同時に切り替わったときは照明の状態が変わる。
- 偶数個のスイッチが同時に切り替わったときは照明の状態は変わらない。
- スイッチは故障することがある。ある時刻にb階のスイッチが故障すると、b階ではそれ以降スイッチを切り替えられない。スイッチ切り替えとスイッチ故障は同時には起こらない。
- 時刻iに起こるイベント$event_i$が与えられる$(1 \leq i \leq E)$。各イベントは、スイッチ切り替えかスイッチ故障のいずれか。
- 時刻0での照明の状態Sが与えられたとき、照明の状態がどう変わるかをシミュレーション。スイッチ切り替えイベント発生直後の照明の状態とスイッチが切り替えられた最も下の階と上の階を出力。

制約

$1 \leq N \leq 1{,}000$，$1 \leq E \leq 1{,}000$，$1 \leq b \leq N$，S: ONかOFFのいずれか

問題４　マンションの照明

スイッチ切り替えイベントの入力は、スイッチ切り替え状況を表す文字列から得られる文字列。

スイッチ切り替え状況を表す文字列

スイッチ切り替え状況を表す文字列は○とxからなる文字列。以下の意味を持つ。

- １文字が１つの階のスイッチの切り替えに関する情報を表す。
- ○はスイッチが切り替わったことを、xは切り替わっていないことを表す。
- スイッチ切り替え状況を表す文字列には、スイッチが故障した階の情報は含まれない。
- したがって、スイッチ切り替え状況を表す文字列の長さは、故障していないスイッチの個数に等しい。
- 左からj番目の文字は、スイッチを切り替えることのできる階のうち、下からj番目の階に対応。

$N = 8$で、１，３，４，７階のスイッチが故障しているとき：

ｏｘｏｏ ⇒ ２，６，８階のスイッチが切り替わった ⇒ 照明の状態が切り替わる

２ ５ ６ ８

スイッチ切り替えイベントの入力形式

スイッチ切り替えイベントの入力では、スイッチ切り替え状況を表す文字列に現れる連続した○やxを、数字列とそれに続く○またはxで表してもよい。

ooooooooox を **10ox**、**5o1oo3ox**、**o8o1o1x** などのように表してよい。

問題４　マンションの照明

入出力例

`10 3 ON`	$N = 10,\ E = 3,\ S =$ `ON`
`2 2x2xox2o2x`	時刻１で５，７，８階のスイッチが切り替わった ⟶ `OFF 5 8` を出力
`1 3`	時刻２で３階のスイッチが故障した
`2 2x2xoxo2x`	時刻３で６，８階のスイッチが切り替わった ⟶ `OFF 6 8` を出力

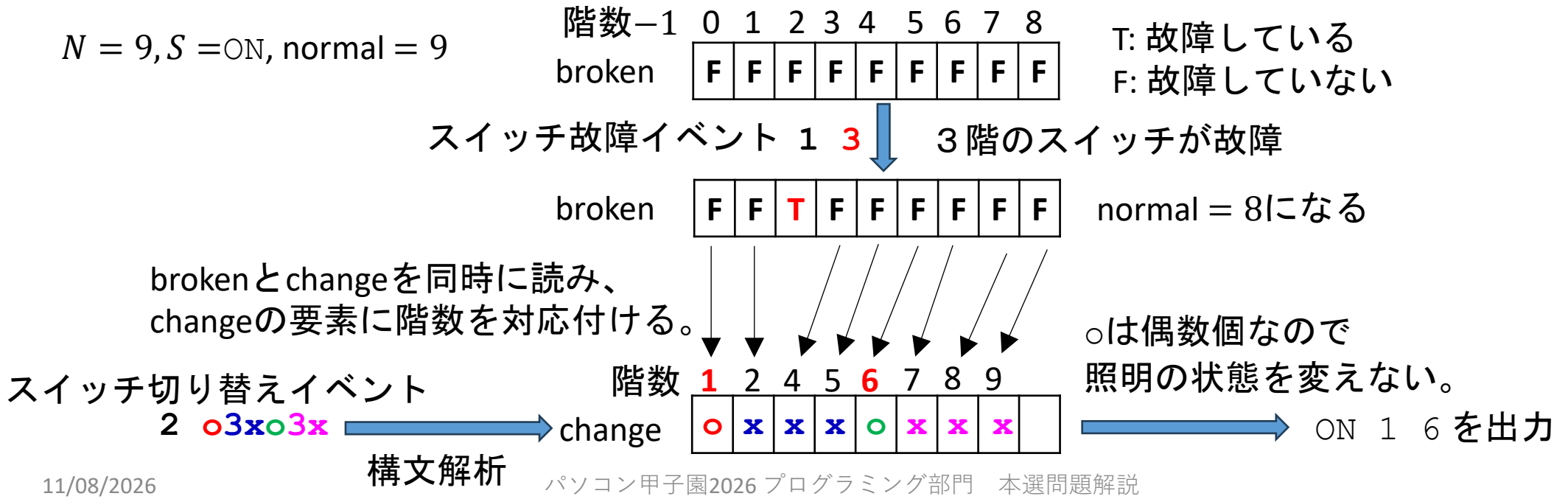
時刻１
- 入力は`2x2xox2o2x`なので、スイッチ切り替え状況を表す文字列は`xxxxoxooxx`。
- どの階のスイッチも故障していない ⇒ スイッチ ｘｘｘｘｏｘｏｏｘｘ
  階数 １２３４５６７８９１０

時刻３
- 入力は`2x2xoxo2x`なので、スイッチ切り替え状況を表す文字列は`xxxxoxoxx`。
- 時刻２で３階のスイッチが故障したので、時刻３の文字列には３階の情報は含まれない。
  ⇒ スイッチ ｘｘｘｘｏｘｏｘｘ
  階数 １２４５６７８９１０

問題４　マンションの照明

やるべきこと

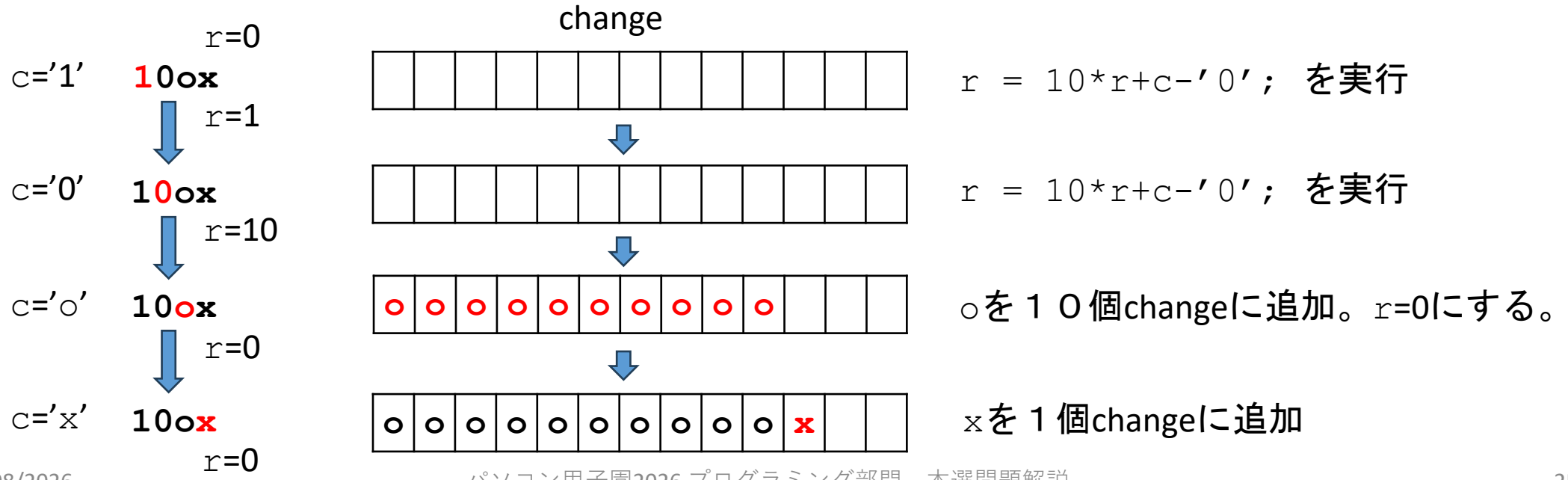
- スイッチ故障イベントで指定された、スイッチが故障した階を配列brokenに保存する。brokenはスイッチ切り替えイベントが入力されたときに、どの階のスイッチが切り替わったかを知るために利用する。さらに、故障していないスイッチの個数（スイッチ切り替え状況を表す文字列の長さ）normalを１減らす。
- スイッチ切り替えイベントの入力文字列を構文解析して、oとxだけから成るスイッチ切り替え状況を表す文字の配列changeに変換する。



問題４　マンションの照明

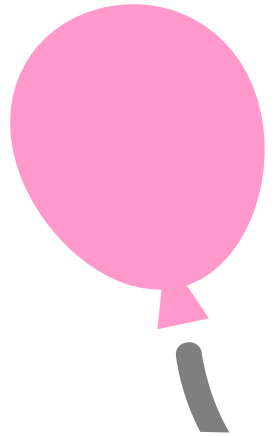
スイッチ切り替えイベントで入力された文字列の構文解析

- 文字の繰り返し回数を表す数をrとする。rの初期値は0。
- 文字列を順に読む。いま読んでいる文字をcで表す。cが数字か、それ以外の文字かで場合分け。
- cが数字のとき：　`r = 10*r+c-'0';` を実行する。
- それ以外のとき：rが0でなければ、配列changeに文字cをr個追加する。その後、rを0に初期化。
  rが0なら、配列changeに文字cを１個追加する。



問題５合計点からわかること（６点）

正答数: 25 チーム



問題５　合計点からわかること

概要

- あるアンケートには N 個の項目があり、項目 i の点数は a_{i}。
- 回答者は、各項目について「当てはまる」かどうかを選ぶ。
- 記録されるのは、選んだ項目の点数の合計だけ。
- 指定された項目Kの回答を合計点から必ず判定できるか？（Yes/No）。

制約

$1 \leq N \leq 20$、$1 \leq K \leq N$、$0 \leq a_{\mathrm{i}} \leq 10^6$

問題５　合計点からわかること

具体例：同じ合計点なら判定できない

$N = 3$、$K = 1$、点数 $[7, 2, 5]$

回答A：項目1だけ選ぶ → 7 点

回答B：項目2と3を選ぶ → 2 + 5 = 7 点

合計点 7 から項目1の回答を区別できない → No

問題５　合計点からわかること

具体例：必ず区別できる場合

$N = 3$、$K = 1$、点数 $[7, 5, 3]$

項目1を選ばない：$\{0, 3, 5, 8\}$

項目1を選ぶ：$\{7, 10, 12, 15\}$

共通する合計点がない → Yes

問題５　合計点からわかること

考察：部分集合和の集合を作る

- 項目K以外から作れる合計点の集合をSとする。
- Kを選ばない場合の合計点は $s \in S$。
- Kを選ぶ場合の合計点は $t + a_k$（$t \in S$）。
- 同じ合計点が作れれば No。

No ⇔ ある $t \in S$ について $t + a_k \in S$

例：$K = 1$、点数 $[7,2,5] \rightarrow S = \{0, 2, 5, 7\}$、$0 + 7 = 7 \rightarrow$ No

問題５　合計点からわかること

解法：set による判定

- K以外の項目の部分集合和をすべて列挙する。
- 得られた合計点をsetに格納する。
- 各tについて $t + a_{\mathrm{k}}$ が set に含まれるか調べる。
- 見つかれば No、なければ Yes。

時間 $O(N \cdot 2^{\mathrm{N}})$ 程度（set の操作を含む）／ 空間 $O(2^{\mathrm{N}})$

別解：合計点をソートし、二分探索など。

問題５　合計点からわかること

実装例 C++

```
int N, K; cin >> N >> K;
vector<long long> a(N);
for (auto &x : a) cin >> x;

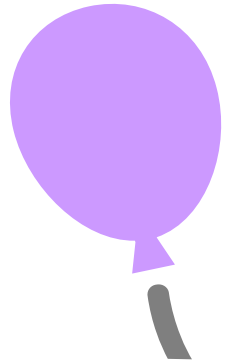
long long target = a[K - 1];
vector<long long> sums = {0};

for (int i = 0; i < N; ++i) {
    if (i == K - 1) continue;
    int m = sums.size();
    for (int j = 0; j < m; ++j)
        sums.push_back(sums[j] + a[i]);
}

set<long long> st(sums.begin(), sums.end());
for (long long x : sums)
    if (st.count(x + target)) { cout << "No¥n"; return 0; }
cout << "Yes¥n";
```

問題６　等脚台形　（６点）

正答数:　　８チーム



問題６　等脚台形

概要

- N本の棒から４本を選び、各棒を１辺として等脚台形を作る
- 作ることのできる等脚台形の面積の最大値を求める
- ただし、等脚台形を作れない場合は0を出力する

問題文：「*等脚台形とは、少なくとも１組の対辺が互いに平行で、平行な２辺のそれぞれについて、その両端の内角が互いに等しい四角形である*」

※ 長方形や正方形も等脚台形に含める

※ 平行四辺形は含まれないが、同じ４辺で長方形ができるので、含まれると考えても解法に影響はない

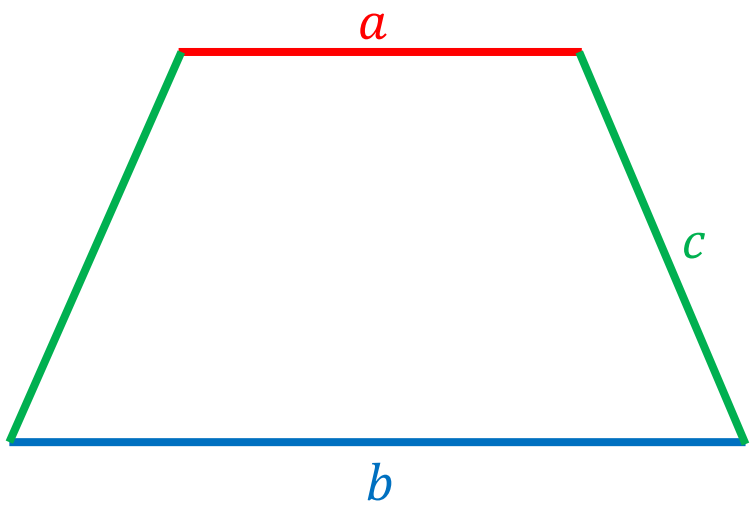
制約

$4 \leq N \leq 200{,}000$
$1 \leq a_i \leq 100{,}000$
すべてのケースのNの合計$\leq 200{,}000$

問題６　等脚台形

等脚台形の面積

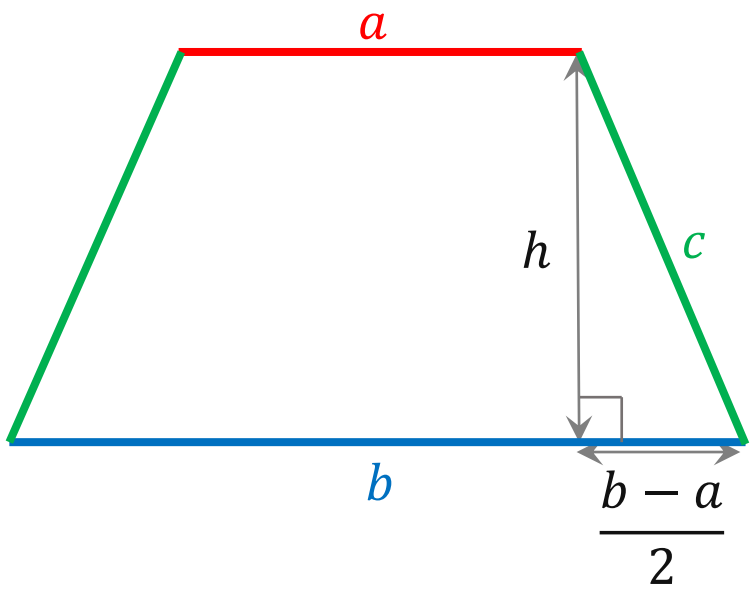
上底の長さをa,
下底の長さをb,
脚の長さcとする。



問題６　等脚台形

等脚台形の面積

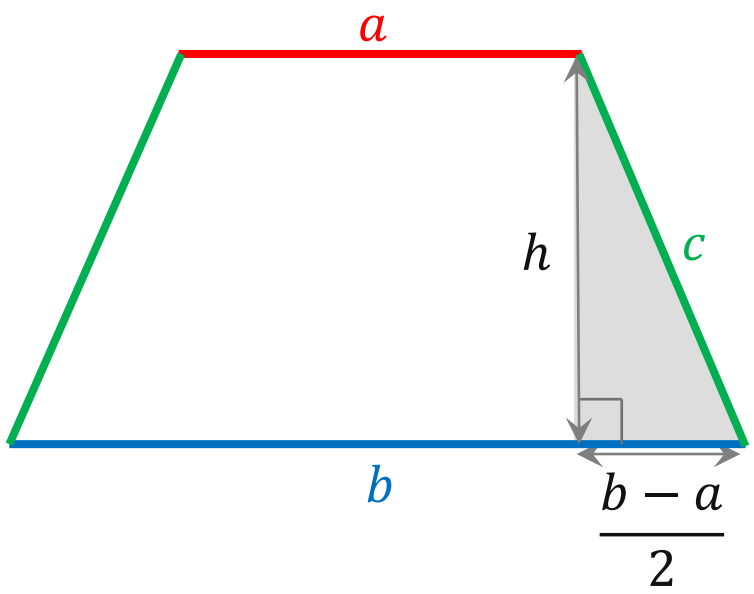
上底の長さをa,
下底の長さをb,
脚の長さcとする。



問題６　等脚台形

等脚台形の面積

上底の長さをa,
下底の長さをb,
脚の長さcとする。



$h^2 + \left(\frac{b-a}{2}\right)^2 = c^2$ より

$$h = \sqrt{c^2 - \frac{(b-a)^2}{4}}$$

台形の面積の公式より：$S = \frac{a+b}{2}h$

ただし、面積が正の実数になるには

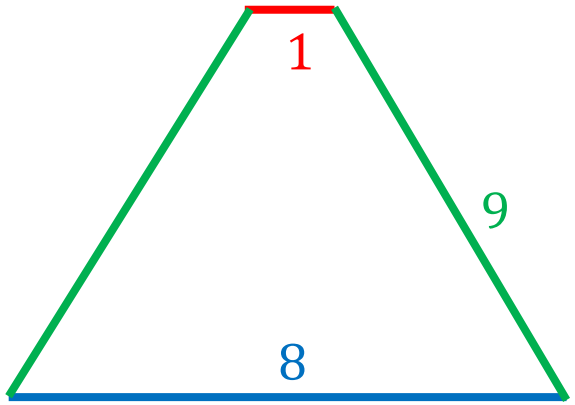
$|b-a| < 2c$

が必要

問題６　等脚台形

入力例１の１ケース目

５１１９９８



$$h = \sqrt{9^2 - \frac{(8-1)^2}{4}} = \sqrt{68.75}$$

$$S = \frac{1+8}{2}\sqrt{68.75} \approx 37.31203889$$

入力例１の２ケース目

４１２３４

等脚台形は作れない

問題６　等脚台形

時間制限になる解法

- 同じ長さの２本を脚にするすべての選び方に対して、
  残りの棒２本の選び方すべてを試す

  時間計算量$\mathrm{O}(N^4)$

- 同じ長さの２本のうち最も長いものを脚にする選び方に対して、
  残りの棒２本の選び方すべてを試す

  時間計算量$\mathrm{O}(N^2)$

Nが最大200,000なので、調べる候補を絞る必要がある

問題６　等脚台形

誤解法例（誤った貪欲法）

①　脚の選択

同じ長さの棒を２本選ぶとき、選べる長さの最大値をcとする
長さcの棒を２本選び、脚として使う

②　上底下底の選択

脚に使う２本を除き、残った棒のうち一番長い２本を使う

最大面積にならないことがある

問題６　等脚台形

誤解法例（誤った貪欲法）

- 棒の長さが3, 3, 4, 5, 10の場合

① 脚に長さ3の棒を2本を選ぶ
残った棒の長さは4, 5, 10

② 長さ5と10の棒を上底と下底にする

$$S = \frac{5+10}{2}\frac{\sqrt{3^2-(10-5)^2/4}}{2}$$
$$= \frac{15}{4}\sqrt{9-6.25} \approx 6.21867$$

問題６　等脚台形

誤解法例（誤った貪欲法）

- 棒の長さが3, 3, 4, 5, 10の場合

① 脚に長さ3の棒を2本を選ぶ
残った棒の長さは4, 5, 10

② 長さ5と10の棒を上底と下底にする

$$S = \frac{5+10}{2}\frac{\sqrt{3^2-(10-5)^2/4}}{2}$$
$$= \frac{15}{4}\sqrt{9-6.25} \approx 6.21867$$

② 長さ4と5の棒を上底と下底にする

$$S = \frac{4+5}{2}\frac{\sqrt{3^2-(5-4)^2/4}}{2}$$
$$= \frac{9}{4}\sqrt{9-0.25} \approx 6.65559$$

こちらの方が大きい

問題６　等脚台形

貪欲法で候補を絞る

①　脚の選択

同じ長さの棒を２本選ぶとき、選べる長さの最大値をcとする
長さcの棒を２本選び、脚として使う

②　上底下底の選択

脚に使う２本を除き、残った棒を長さの昇順に並べる（ソート後の長さをl_iとする）
この順番で隣り合う２本l_iとl_{i+1}を、すべて上底下底の候補として調べる

各候補の面積を計算し、最大値を答える

全体：　時間$O(N\log N)$、空間$O(N)$

ソートに$O(N\log N)$

問題６　等脚台形

実装例 C++

```
long double solve(vector<long long> a) {
    sort(a.begin(), a.end());
    int k = -1;
    for (int i = (int)a.size() - 1; i > 0; --i)
        if (a[i] == a[i-1]) { k = i; break; }
    if (k == -1) return 0;
    long long c = a[k];
    a.erase(a.begin() + k - 1, a.begin() + k + 1);
    long double ans = 0;
    for (int i = 1; i < (int)a.size(); ++i) {
        long long x = a[i-1], y = a[i], d = y - x;
        if (d >= 2 * c) continue;
        long long D = 4 * c * c - d * d;
        long double area = (x + y) * sqrtl((long double)D) / 4;
        ans = max(ans, area);
    }
    return ans;
}
```

実装上の注意

- $|b-a| \geq 2c$は候補から除く
- 平方根の中身は、まず64ビット整数で求めて、結果が正のときだけ実数型で面積を計算する

問題６　等脚台形

① 脚の選択 の正当性

同じ長さで２本ある棒の、最大長さをmとする
脚以外の２辺を底辺と呼ぶ

底辺に長さmの棒を使う本数で場合分けする

I. 底辺に長さmの棒を１本も使わない場合

底辺をそのままに、脚に長さmの棒2本を使えば、c^2が最大になるので、Sが最大になる

II. aとbの両方に長さm棒を使う場合

底辺と脚の役割を入れ替えれば、I. と同じことが言える

III. 底辺のうち１本に長さmの棒を使う場合

$$S = \frac{a+b}{2}\sqrt{c^2 - \frac{(b-a)^2}{4}}$$

上底の長さをa, 下底の長さ$b = m$とする
このとき脚に使う棒の長さを$c < m$と書くと

$$S_1 = \frac{a+m}{2}\sqrt{c^2 - \frac{(m-a)^2}{4}}$$

一方、未使用のmが１本あるので、下底に使っていたmと未使用のmを脚に使い、下底に脚に使っていた棒cを使うと

$$S_2 = \frac{a+c}{2}\sqrt{m^2 - \frac{(c-a)^2}{4}}$$

$$S_2^2 - S_1^2 = \frac{1}{16}\left[(m-c)\left\{(m+c)\left(2a^2+m^2+c^2\right)+8acm\right\}\right] > 0$$

が言えるので、２本のmを脚に使う方が良い

問題６　等脚台形

② 上底下底の選択 の正当性

脚に最大の長さであるmの棒を２本使う
残った棒を昇順に並べ、その長さを$l_1, l_2, \cdots, l_k$とする。

長さ順で隣接する棒l_iとl_{i+1}を使う正当性を示せばよい

脚を長さmの棒で固定したので

$$S = \frac{a+b}{2}\sqrt{m^2 - \frac{(b-a)^2}{4}}$$

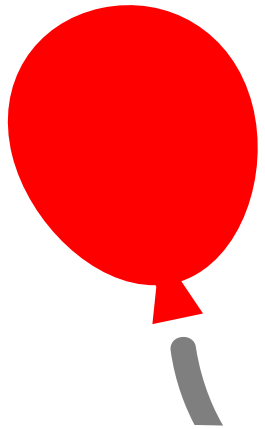
$a+b$部分がより大きく
$b-a$部分がより小さく
なるような選択肢を考える

底辺としてl_iと$l_j (i<j)$を選んだとする．このとき，$l_i \le l_j$である
この２本が隣り合っていなければ、$l_i \le l_{j-1} \le l_j$なので、l_iをl_{j-1}に置き換えることで
$l_i + l_j \le l_{j-1} + l_j$ かつ $l_j - l_i \ge l_j - l_{j-1}$ にできる

したがって、隣り合う２本の組をすべて調べればよい

問題７　支払うことができない金額（10点）

正答数:　　14 チーム



問題７　支払うことができない金額

概要

- 価値Aの硬貨と価値Bの硬貨を、それぞれ何枚でも使える。
- P, Qを0以上の整数として、 $AP + BQ$の金額を支払える。
- 支払うことができない正の整数で表される金額のうち、K番目に小さい金額を出力するプログラムを作成せよ。
- 制約
  - $1 \leq A, B \leq 10^5, 1 \leq K \leq 5 \times 10^9$
  - AとBは互いに素（最大公約数が1）
- 支払うことができない金額が、K個未満のときは-1を出力する。

問題７　支払うことができない金額

- 入力例１: $A = 4, B = 7, K = 6$
  - 支払うことができない金額（小さい順）: 1, 2, 3, 5, 6, **9**, …
  - 答え（6番目に小さい金額）: **9**
- 入力例２: $A = 3, B = 5, K = 3$
  - 支払うことができない金額（小さい順）: 1, 2, **4**, …
  - 答え（3番目に小さい金額）: **4**
- 入力例３: $A = 99991, B = 99989, K = 4998900060$
  - 答え（4998900060番目に小さい金額）: **9997800119**
- 入力例４: $A = 1, B = 37, K = 1$
  - 答え: **-1**（すべての金額が支払える）

問題７　支払うことができない金額

考察: 同じ余りに注目する

- $M = A + B$とする。
- 金額$x = AP + BQ$を支払えるなら、
  $x + M = A(P + 1) + B(Q + 1)$
  も、必ず支払える。

- 例: $A = 4, B = 7, M = 11$
  - 余り1:　**1**, **12**, **23**, **34**, **45**, …
  - 余り6:　**6**, **17**, **28**, **39**, **50**, …
- **赤文字の金額は支払えない。**
- **青文字の金額は支払える。**

（色の意味は、これ以降同じ）

① $\gcd(A, B) = 1$なので、 $\gcd(A, M) = \gcd(A, M - A) = \gcd(A, B) = 1$
- gcd: 最大公約数

② $0, A, 2A, \cdots, (M - 1)A$を$M$で割った余りは、$0, 1, 2, \cdots, M - 1$を重複なくすべて取る。

③ Aの倍数の金額は、価値Aの硬貨だけで支払える。したがって、②より、どの余りについても、その余りになる支払える金額が必ず存在する。

問題７　支払うことができない金額

解法: 余りごとの最小金額を考える

- dist[r] =「Mで割った余りがrの、支払える最小の0以上の整数」
  - $0 \leq r < M$
- 例: $A = 4, B = 7, M = 11$

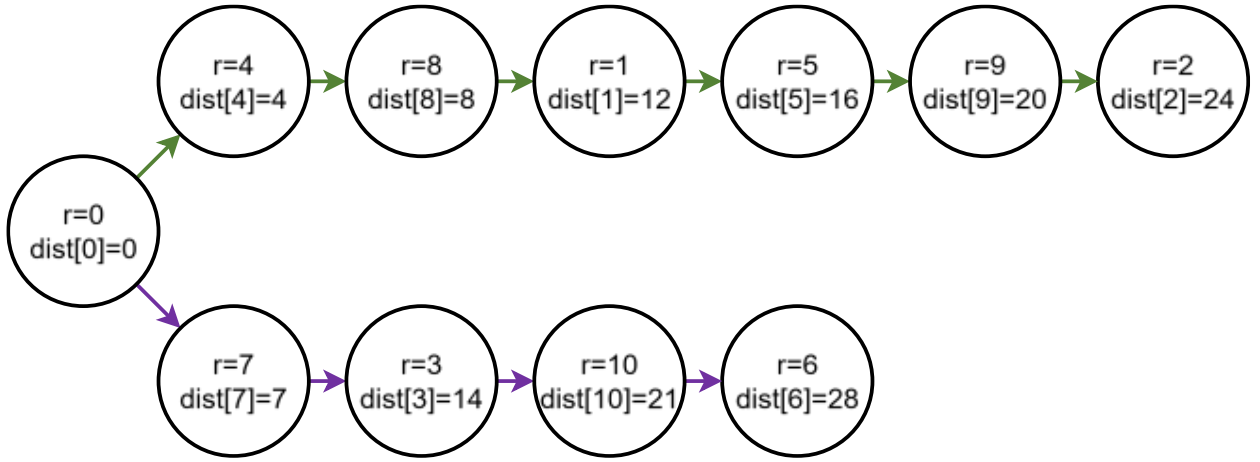
余りr	dist[r]	同じ余りを持つ金額
1	12	**1**, **12**, **23**, **34**, **45**,…
6	28	**6**, **17**, **28**, **39**, **50**, …
7	7	**7**, **18**, **29**, **40**, **51**, …

- dist[r]以上の同じ余りrを持つ金額は、すべて支払える。

問題７　支払うことができない金額

解法: ダイクストラ法で、dist[r] を求める　例: $A = 4, B = 7, M = 11$

- 余りを頂点とする、辺が追加されていくグラフについて、0を始点とする単一始点最短経路問題を解く。
- 頂点: $0, 1, \cdots, M-1$（M点）
- 1本の辺 = 硬貨を1枚追加
  - $r \to (r + A)\%M$（**価値Aの硬貨を追加**）
  - $r \to (r + B)\%M$（**価値Bの硬貨を追加**）
- dist[r] = 頂点0から頂点rへの最短距離



※追加後に dist が更新できた辺だけを描画

①余り0の頂点から開始: dist[0] = 0、(dist[0], 0) を優先度付きキュー（pq）に追加

②pqから、pqに含まれる中で距離が最小の頂点を取る

③A, Bの2本の辺を追加。distをより小さい値に更新できたら、(新しい距離, 頂点) をpqに追加

④pqが空になるまで、②, ③を繰り返す。

問題７　支払うことができない金額

解法: x以下で支払える金額の個数を数える

- d = dist[r] とすると、余りrの金額で支払えるものは
  $d, d+M, d+2M, d+3M, \cdots$
- 余りrの支払える金額の個数
  - $d > x$のとき、0 個
  - $d \leq x$のとき、$\left\lfloor \frac{x-d}{M} \right\rfloor + 1$ 個

- 例: $A = 4, B = 7,\ M = 11, x = 15$

余りr	d=dist[r]	$d, d+M,$ $d+2M, \cdots$	x以下の個数
1	12	**12**, 23, 34, ⋯	1
4	4	**4**, **15**, 26, ⋯	2
6	28	28, 39, 50, ⋯	0

- num_can_pay(x) = x以下で**支払える**金額の個数（金額0を含む）
- count_cannot(x) = x以下で**支払えない**金額の個数 = x − (num_can_pay(x) − 1)

問題７　支払うことができない金額

解法: x以下で支払える金額の個数を数える（例で確認）
$A = 4, B = 7, M = 11, x = 15$

- $\mathrm{dist}[r] \leq x$であるのは、$r = 0, 1, 3, 4, 7, 8$のとき
- $\mathrm{num_can_pay}(x) = \left\lfloor \frac{15-\mathrm{dist}[0]}{11} \right\rfloor + \left\lfloor \frac{15-\mathrm{dist}[1]}{11} \right\rfloor + \left\lfloor \frac{15-\mathrm{dist}[3]}{11} \right\rfloor + \left\lfloor \frac{15-\mathrm{dist}[4]}{11} \right\rfloor + \left\lfloor \frac{15-\mathrm{dist}[7]}{11} \right\rfloor + \left\lfloor \frac{15-\mathrm{dist}[8]}{11} \right\rfloor + 6$

$$= \left\lfloor \frac{15-0}{11} \right\rfloor + \left\lfloor \frac{15-12}{11} \right\rfloor + \left\lfloor \frac{15-14}{11} \right\rfloor + \left\lfloor \frac{15-4}{11} \right\rfloor + \left\lfloor \frac{15-7}{11} \right\rfloor + \left\lfloor \frac{15-8}{11} \right\rfloor + 6 = 1+0+0+1+0+0+6 = 8$$

  - $0, 4, 7, 8, 11, 12, 14, 15$の8個が**支払える**
- $\mathrm{count_cannot}(x) = 15 - (8 - 1) = 8$
  - $1, 2, 3, 5, 6, 9, 10, 13$の8個が**支払えない**

- num_can_pay(x) = x以下で**支払える**金額の個数（金額0を含む）
- count_cannot(x) = x以下で**支払えない**金額の個数 = x − (num_can_pay(x) − 1)

問題７　支払うことができない金額

解法: 二分探索で答えを求める

- 支払えない金額の最大値（max_cannot_pay）= distの最大値$-M$
- 支払えない金額の総数 = count_cannot(max_cannot_pay)
- count_cannot(max_cannot_pay) $< K$のとき:
  - -1 を出力する。
- count_cannot(max_cannot_pay) $\geq K$のとき:
  - count_cannot(x)は単調増加であるので、[0, max_cannot_pay]の区間において、count_cannot(y) $\geq K$となる最小のy（答え）を二分探索で求めればよい。

※ $\gcd(A, B) = 1$なのでフロベニウスの硬貨交換問題であるため、支払えない金額の最大値は$AB - A - B$、支払えない金額の総数は$\frac{(A-1)(B-1)}{2}$で求められる。

問題７　支払うことができない金額

時間計算量

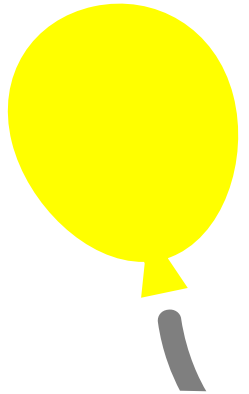
- ダイクストラ法で dist を求める: $O(M \log M)$
- count_cannot(y) $\geq K$となる最小のyを二分探索で求める: $O(M \log(AB))$
  - $M = A + B$

空間計算量

- $O(M)$

問題８ 回文を作ろう （１０点）

正答数: ９チーム



問題８　回文を作ろう

概要

- 長さ N の文字列Sと、正整数列L, R が与えられる。
- 先頭からa文字、末尾からb文字を削除する。$a, b \geq 0$、$a + b < N$。
- 残った文字列が回文なら得点。
- 得点は $L[a] + R[b]$。
- すべての回文について合計し総得点を求めよ。

制約

$1 \leq N \leq 200{,}000$ ／ 英小文字 ／ $1 \leq L[i], R[i] \leq 1{,}000$

問題８　回文を作ろう

具体例：削除位置と回文

S ＝aaba　　L ＝[1,2,3,4]　　R ＝[4,3,2,1]



$a = 1$、$b = 2$ → 残る文字列は「a」

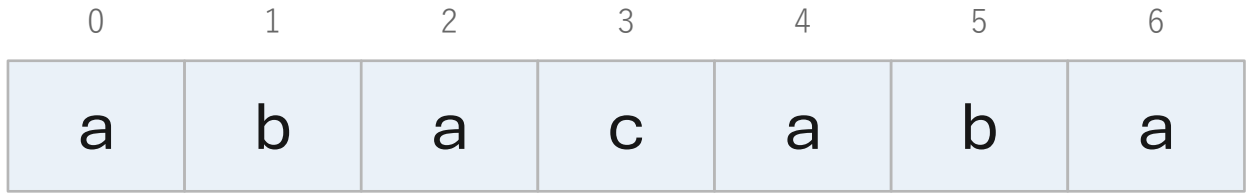
得点 $= L[1] + R[2] = 2 + 2 = 4$

回文区間 $[l, r]$ に対し、$a = l$、$b = N - 1 - r$

問題８　回文を作ろう

Manacher法：左右対称性を利用　（※概要の紹介のみ）

例：S ＝abacaba



中心 3 の回文 $[0,6]$ が分かっている。

位置 1 と位置 5 は中心 3 に関して対称。

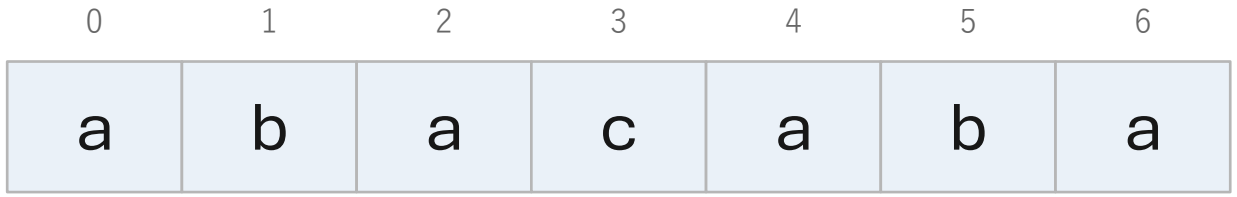
位置 1 の「aba」[0, 2] → 位置 5 の「aba」[4, 6] に再利用。

左右対称な回文の情報を再利用し、全中心の最大半径を $O(N)$ で求める。

問題８　回文を作ろう

考察：中心の半径から区間加算へ

S ＝ abacaba、中心 $c = 3$、最大半径 $h = 4$（奇数長）



回文：c、aca、bacab、abacaba

開始位置：$3, 2, 1, 0 \rightarrow [0,3]$ の各位置に $+1$

終了位置：$3, 4, 5, 6 \rightarrow [3,6]$ の各位置に $+1$

奇数長：開始 $[c-h+1, c]$、終了 $[c, c+h-1]$

問題８　回文を作ろう

差分配列と累積和：区間加算をまとめる

例：[0,3] と [2,5] に +1

位置	0	1	2	3	4	5	6
[0,3]	1	1	1	1	0	0	0
[2,5]	0	0	1	1	1	1	0
差分 D	1	0	1	0	−1	0	−1
累積和 C	1	1	2	2	1	1	0

区間 $[l, r]$ に +1：$D[l]$ += 1、$D[r+1]$ −= 1

問題８　回文を作ろう

得点の集計

- Manacher法で各中心の最大半径を求める。
- 開始位置・終了位置に対する区間加算を差分配列に記録。
- 累積和で、各位置が回文の端になる回数を復元する。

合計 $=$ $\boldsymbol{\Sigma}$ (開始回数$[\boldsymbol{l}]$ $\times$ $\boldsymbol{L}[\boldsymbol{l}]$) $+$ $\boldsymbol{\Sigma}$ (終了回数$[\boldsymbol{r}]$ $\times$ $\boldsymbol{R}[\boldsymbol{N}-\boldsymbol{1}-\boldsymbol{r}]$)

計算量

時間計算量 $O(N)$ ／ 空間計算量 $O(N)$

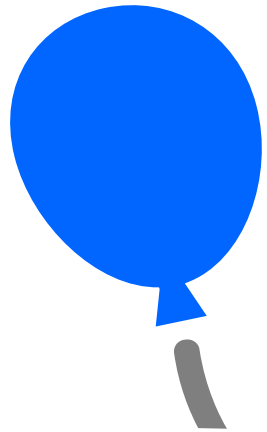
問題８　回文を作ろう

別解

- ローリングハッシュ ＋ 二分探索。
- 時間計算量 $O(N \log N)$

問題９　機能の選定（１０点）

正答数:　　0 チーム



問題９　機能の選定

概要

- 開発中のシステムへの搭載を検討中のN個の機能と、守るべきM個のルールがある。
- ルールは２つの要求から成り、１つ以上の要求に従わなければならない。
- 要求は「機能iを搭載する」（$+i$で表す）か「機能iを搭載しない」（$-i$で表す）のいずれか（$1 \leq i \leq N$）。
- まず、M個のルールすべてを守れるか判定する。守れなければ「Impossible」を出力。
- 経営陣からQ個の提案を受け取る。
- 提案も２つの要求から成るが、両方の要求に従わなければならない点がルールと異なる。
- すべてのルールを守れるならば、各提案について、その提案を採用できるかを判定し、採用できれば「Yes」、できなければ「No」を出力。
- ある提案を採用できるのは、すべてのルールを守ったうえで、その提案の２つの要求に従うことができるとき。

制約

$1 \leq N \leq 20{,}000,\ \ 1 \leq M \leq 20{,}000,\ \ \ 1 \leq Q \leq 100{,}000$

問題９　機能の選定

問題９の論理による解釈

- ルール「+2 -3」を「命題２、または、命題３の否定」という命題で表す（この命題を節と呼ぶ）。このルールを守れるなら、「命題２、または、命題３の否定」は真とする。
- すべてのルールを守ることを、各ルールが表す節を「かつ」で繋げた命題で表す（節を「かつ」で繋げた命題を連言標準形と呼ぶ）。すべてのルールが守れるなら、対応する連言標準形は真にできる。
- 提案「+2 -3」を「命題２、かつ、命題３の否定」という命題で表す。提案を採用できるなら、すべてのルールを守ることを表す命題と提案が表す命題を「かつ」で繋げた命題は真にできる。

「かつ」、「または」、「否定」を含む命題の真偽を表す真理値表（Tは真、Fは偽を表す）

AかつB（$A \wedge B$と書く）

A	B	$A \wedge B$
T	T	T
T	F	F
F	T	F
F	F	F

AとBが真のときだけ$A \wedge B$は真

AまたはB（$A \vee B$と書く）

A	B	$A \vee B$
T	T	T
T	F	T
F	T	T
F	F	F

AとBが偽のとき以外は$A \vee B$は真

Aの否定（$\overline{A}$と書く）

A	$\overline{A}$
T	F
F	T

Aが偽のときだけ$\overline{A}$は真

問題９　機能の選定

入出力例１

3 3	$N = 3,\ M = 3$
+1 +2	ルール $1 \vee 2$（命題１または命題２）
-1 +3	ルール $\overline{1} \vee 3$（命題１の否定または命題３）
-2 -3	ルール $\overline{2} \vee \overline{3}$（命題２の否定または命題３の否定）
3	$Q = 3$
+1 +3	提案 $1 \wedge 3$（命題１かつ命題３）
-1 -3	提案 $\overline{1} \wedge \overline{3}$（命題１の否定かつ命題３の否定）
-1 -2	提案 $\overline{1} \wedge \overline{2}$（命題１の否定かつ命題２の否定）

1	2	3	$(1 \vee 2) \wedge (\overline{1} \vee 3) \wedge (\overline{2} \vee \overline{3})$
T	T	T	F
T	T	F	F
T	F	T	T
T	F	F	F
F	T	T	F
F	T	F	T
F	F	T	F
F	F	F	F

- すべてのルールを守れるということは、命題$(1 \vee 2) \wedge (\overline{1} \vee 3) \wedge (\overline{2} \vee \overline{3})$を真にできること。
- 右上の真理値表から(a)１と３が真、２が偽 (b)２が真、１と３が偽のとき、この命題は真。
  したがって、すべてのルールを守れる。
- 表の３行目より、１つ目の提案は採用できる。
- 表の６行目より、２つ目の提案は採用できる。
- 表の７，８行目より、３つ目の提案は採用できない。

問題９　機能の選定

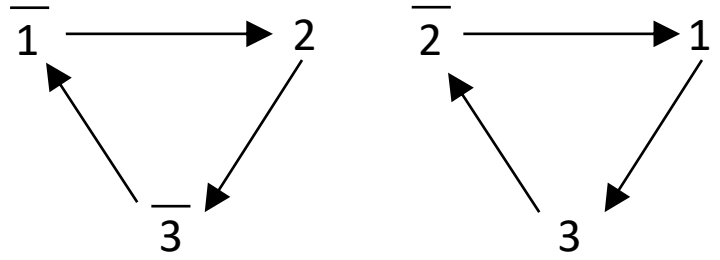
グラフによる解法

- 真理値表を使って解くと、$O(2^N)$かかる（各命題の真偽をすべて考えなければならない）。
- 「すべてのルールを守る」という命題をグラフで表すと、効率よく解くことが可能。
- 命題「AならばB」（$A \Longrightarrow B$と書く）の真偽は、以下の真理値表のとおり。
- $A \vee B$と$\overline{A} \Longrightarrow B$と$\overline{B} \Longrightarrow A$の真偽は同じなので、$A \vee B$の代わりに $\overline{A} \Longrightarrow B$や$\overline{B} \Longrightarrow A$ を考えてもよい。
- 連言標準形の各節$A \vee B$と真偽が同じ命題 $\overline{A} \Longrightarrow B$と $\overline{B} \Longrightarrow A$に対応する辺 $\overline{A} \longrightarrow B$ と $\overline{B} \longrightarrow A$ から成る有向グラフを含意グラフという。
- 含意グラフに辺 $A \longrightarrow B$ があれば、$A \Longrightarrow B$は真（逆も言える）。
- 三段論法（$A \Longrightarrow B$と$B \Longrightarrow C$が真なら$A \Longrightarrow C$ は真）より、AからBへ到達可能なら$A \Longrightarrow B$は真（逆も言える）。

A	B	$A \Longrightarrow B$
T	T	T
T	F	F
F	T	T
F	F	T

Aが真、Bが偽のとき以外は、$A \Longrightarrow B$は真

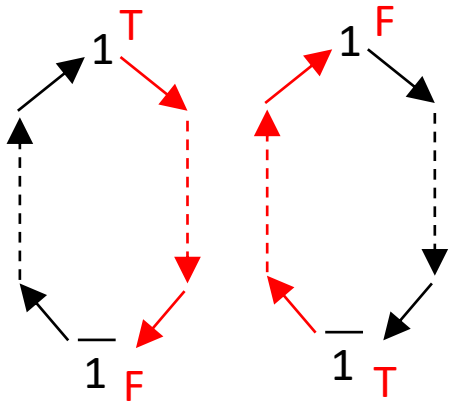
命題$(1 \vee 2) \wedge (\overline{1} \vee 3) \wedge (\overline{2} \vee \overline{3})$の含意グラフ



問題９　機能の選定

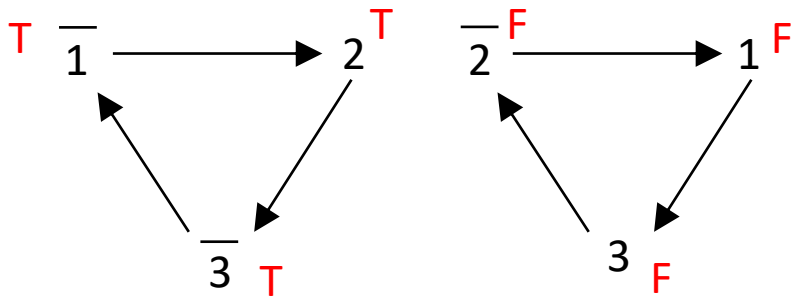
グラフによる解法（続き）

- 含意グラフの頂点に真か偽を割り当てる。
- 有向辺A⟶Bがあれば命題A ⟹ Bは真なので、Aに真、Bに偽を割り当てることはできない。
- 頂点AからBに到達可能なら、命題A ⟹ B は真なので、Aに真、Bに偽を割り当てることはできない。
- Aに真を割り当てたら、$\overline{A}$に偽を割り当てなければならない。
- 任意の頂点xについて、xと$\overline{x}$が同じ強連結成分に含まれれば命題は偽（左下の図を参照）。
  なければ真にできる割り当てがある。



どちらの割り当ても許されない。

命題$(1 \vee 2) \wedge (\overline{1} \vee 3) \wedge (\overline{2} \vee \overline{3})$の含意グラフ



1と$\overline{1}$、2と$\overline{2}$、3と$\overline{3}$のどれも、同じ強連結成分に含まれることはないので、この命題は真。

問題９　機能の選定

問題前半の解法

- すべてのルールを守れるかどうかは、含意グラフを作り、強連結成分分解をすることで解ける。
- 1からNまでのそれぞれの数xについて、$+x$と$-x$が同じ強連結成分(SCC)に属するかどうかチェック。
- 計算量：$O(N+M)$
- すべてのルールを守れるか判定する問題を論理により解釈した問題（「すべてのルールを守る」を表す命題が真か判定する）は2-SATと呼ばれ、上で示したような効率の良い解法が存在する。

問題後半の解法

- すべてのルールを守ったうえで、提案を構成する２つの要求の両方に従うことができるか判定する問題には、$O(N+M)$ や$O(N\log M)$のような効率的な解法は見つかっていない。
- ３つの解法を考える。
  1. 各提案について、すべてのルールと提案から新たに含意グラフを作って強連結成分分解を行う。
  2. 既存の含意グラフで、頂点間の到達可能性を調べる。
  3. 既存の含意グラフのSCCを頂点とする非循環有効グラフ(DAG)で、SCC間の到達可能性を調べる。

問題９　機能の選定

解法１（各提案について、強連結成分分解を行う）

- 命題 x の真偽は、命題 $x \vee x$ の真偽と同じ。
- 提案「+1 +3」に対応する命題 $1 \wedge 3$ の真偽は $(1 \vee 1) \wedge (3 \vee 3)$ の真偽と同じ。
- 入出力例１で提案「+1 +3」を採用できるか判定する問題は、
  命題 $(1 \vee 2) \wedge (\overline{1} \vee 3) \wedge (\overline{2} \vee \overline{3}) \wedge (1 \vee 1) \wedge (3 \vee 3)$ が真かどうかを判定する問題（2-SAT）になる。
- 各提案について、すべてのルールにその提案を加えた含意グラフを作り、強連結成分分解をする。
- 提案ごとに毎回含意グラフを作って強連結成分分解をするので、計算量は $O(Q(N+M))$。
- この問題の制約の上限（$N = M = 20{,}000, Q = 100{,}000$）だと40億 ⇒ 時間制限（２秒）で時間切れ

問題９　機能の選定

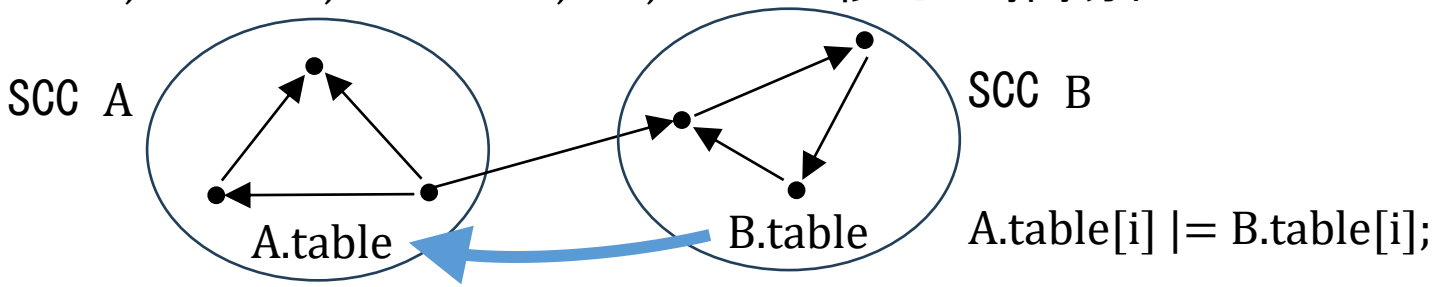
解法２（既存の含意グラフで、到達可能性を調べる）

- 提案を表す命題 $x \wedge y$ が偽になるのは、以下の３つのどれかが成り立つとき。
  - $x \Longrightarrow \overline{x}$ が真のとき（xを真にすると$\overline{x}$は偽になり、$x \Longrightarrow \overline{x}$ が偽になってしまう）
  - $y \Longrightarrow \overline{y}$ が真のとき（yを真にすると$\overline{y}$は偽になり、$y \Longrightarrow \overline{y}$ が偽になってしまう）
  - $x \Longrightarrow \overline{y}$ が真のとき（yを真にすると$\overline{y}$は偽になる。xも真にすると、$x \Longrightarrow \overline{y}$ が偽になってしまう）
- 既存の含意グラフで、xから$\overline{x}$、yから$\overline{y}$、xから$\overline{y}$ のいずれかが到達可能かどうかを調べる。
- 提案ごとに到達可能性を調べると、計算量は$O(Q(N+M))$ ⇒ 時間切れ
- 含意グラフを作った直後に全頂点間の到達可能性を調べる。全ての提案の計算量は$O(Q)$。
- 全頂点間の到達可能性判定
  - Warshall法　$O(N^3)$ ⇒ $20{,}000^3 = 8{,}000{,}000{,}000{,}000$　時間切れ
  - bitsetを使ったWarshall法　$O(N^3/64)$ ⇒ $20{,}000^3/64 = 125{,}000{,}000{,}000$　時間切れ
  - 各頂点についてDFS $O(N(N+M))$ ⇒ $20{,}000(20{,}000+20{,}000) = 800{,}000{,}000$　２秒だと時間切れ

問題９　機能の選定

解法３（既存の含意グラフのSCCを頂点とするDAGで、SCC間の到達可能性を調べる）

- グラフを強連結成分分解すると、SCCを頂点とするグラフには閉路がない（このグラフをDAGと言う）。
- DAGの各頂点である含意グラフのSCC(Aとする)に、そのSCCから各SCCへの到達可能性を表すテーブル(A.tableとする)を持たせる。テーブルの各要素(C++だとbool型)が各SCCの到達可能性を表す。
- 含意グラフのSCCをトポロジカルソートの逆順に訪れ、各SCCが持つテーブルを完成させる。
- 一般に、DAGの頂点をトポロジカルソートの逆順に訪れると、いま訪問している頂点から到達可能な頂点はすべて訪問済みであることが保証される。
- 含意グラフから作られるDAGの頂点であるSCC(Aとする)をトポロジカルソートの逆順で訪問するとき、A⟶B となるSCC B のテーブルは完成している。C++なら、各SCC iについて A.table[i] |= B.table[i]; を実行すれば、 Bを経由したときのAから各SCCへの到達可能性を得られる。
- 必要なメモリは$O(N^2)$ ⇒$20{,}000 \times 20{,}000 = 400{,}000{,}000$ =約400MB　ジャッジサーバだと確保できる
- 計算量は $O(NM)$ ⇒$20{,}000 \times 20{,}000 = 400{,}000{,}000$　２秒だと時間切れ



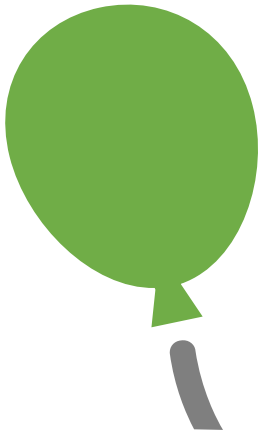
問題９　機能の選定

解法３の別解

- DAGの各頂点(含意グラフのSCC)に持たせるテーブルを、bool型配列ではなくBitsetにする。
- 各要素は１ビットで表現されるので、必要なメモリはbool型配列のときの1/8になる。
- SCC AからSCC Bへの辺 A⟶B があるとき、Bを経由したときのAから各SCCへの到達可能性は A.table |= B.table; で得られる。これは、bool型配列のときの1/64の時間で実行できる。
- 計算量は $O(NM/64)$ ⇒$20{,}000 \times 20{,}000/64 = 6{,}250{,}000$　制限時間内に終了できる

問題１０　つながる三人組（１１点）

正答数:　　1 チーム

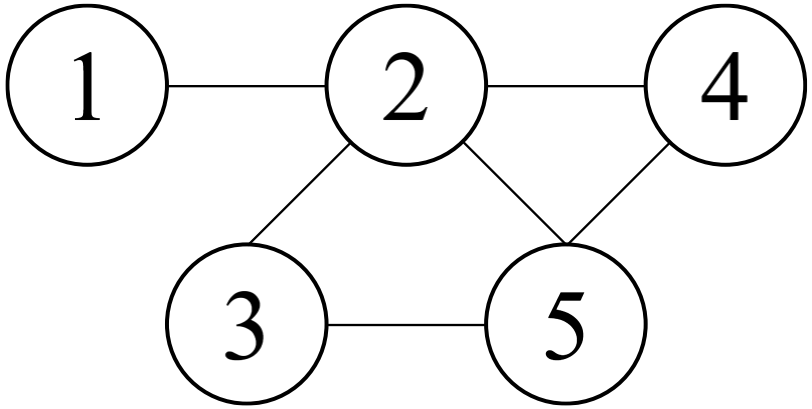


問題１０　つながる三人組

概要

- N人の利用者がいるSNSの運用が開始された。
- 利用者は、他の利用者と、お互いの間につながりを登録することと、お互いの間のつながりを削除することができる。
- 異なる３人の利用者a, b, cからなる組について、
  - aとbの間、bとcの間、cとaの間

  のすべてにつながりが登録されているとき、
  この組を「つながる三人組」と呼ぶ。
- 開始時にはどの利用者の間にもつながりは登録されていない。
- つながりの登録・削除がQ回行わえる。各操作の直後のつながる三人組の数を出力するプログラムを作成せよ。
- 制約: $2 \leq N \leq 2 \times 10^5, 1 \leq Q \leq 2 \times 10^5$。登録操作の形式は「$1\ u\ v$」であり、削除操作の形式は「$2\ u\ v$」である$(1 \leq u < v \leq N)$。
- 時間制限: 6秒



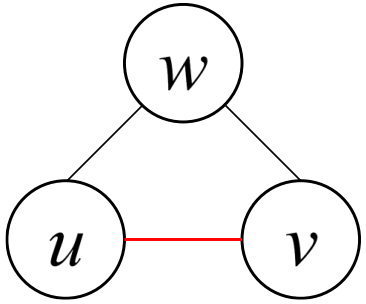
この場合、つながる三人組の数は$(2, 3, 5)$と$(2, 4, 5)$の２組

問題１０　つながる三人組

考察: つながる三角形の数の増減

- 利用者を**頂点**、つながりを**辺**とする**グラフ**として考える。
- つながる三人組を、**三角形**と以後表記する。
- 辺$u-v$の追加・削除で増減する三角形は、その辺を含むものだけ。
  - 増減する三角形における第３の頂点wは、uとvの両方とつながっている。
- $\Delta=$ uとvの共通の隣接頂点wの数

操作	三角形の数の変化
辺$u-v$を追加	Δ個増加
辺$u-v$を削除	Δ個減少



- 他の辺の状態は変わらないので、Δを数えられればよい。

問題１０　つながる三人組

考察: 共通の隣接頂点を数えるには

- 素朴な数え方としては、uの隣接頂点wをすべて調べ、辺$v-w$の有無を確認する方法がある
- この方法は、次数の大きい頂点への操作が続くと、１回に$O(Q)$の計算量が必要であり、全体で$O(Q^2)$になり得る→時間制限
  - ビット演算を利用するなどの定数倍の高速化をすれば時間制限を超えない可能性があるが、すべての頂点同士のつながりを保持する必要があるのでメモリ制限を超えてしまう
- なので、**確認する頂点の数を、どの操作でも$O(\sqrt{Q})$に抑えたい**

問題１０　つながる三人組

解法: 前処理として、登場する辺の集合を作り、頂点の順序を固定する

- すべての操作クエリを読み、一度でも登場する辺を集めた集合Eを作る。
  - $M = |E| \leq Q$
  - 操作直後のグラフの辺の集合は、常にEの部分集合
- deg[v] = Eの辺をすべて張ったグラフでの頂点vの次数
  - この次数は前処理で求め、以降の操作中は変わらない。
- $u \prec v \Leftrightarrow (\deg[u], u) < (\deg[v], v)$
  - 次数が小さい頂点を前に置く。同じ次数なら頂点番号が小さい方を前に置く。
- Eの各辺を、順序が前の頂点から後の頂点へ向きつける（有向辺とする）。
- 各頂点uについて、 uから出る有向辺の終点（$= u$より順序が後ろの隣接頂点）をすべて格納した隣接リストout[u]を作る。

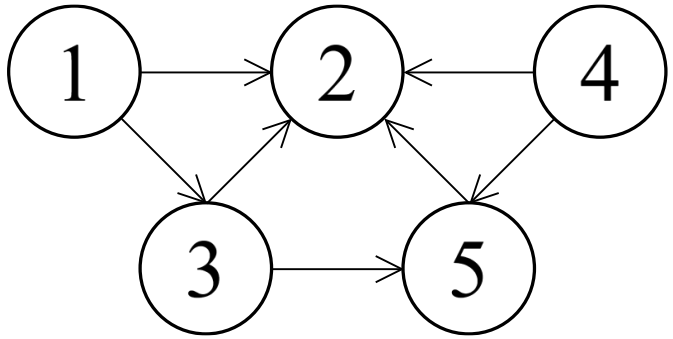
問題１０　つながる三人組

前処理の例: 入力例１の場合の向きつけ

- $E = \{1-2, 2-3, 1-3, 2-4, 2-5, 3-5, 4-5\}, M = 7$

頂点（順序順）	1	4	3	5	2
deg	2	2	3	3	4
out	{2, 3}	{2, 5}	{2, 5}	{2}	{}

- $1 \prec 4 \prec 3 \prec 5 \prec 2$
- (deg, 番号)の順序で向きを決める。



問題１０　つながる三人組

向きつけの性質: 出次数は小さい

- ある頂点uから出る辺の数（出次数）を$k = |\mathrm{out}[u]|$とする。

根拠	分かること
出ていく辺はuの隣接辺の一部	$\deg[u] \geq k$
行き先xの順序はuの後	$\deg[x] \geq \deg[u] \geq k$
行き先がk個ある	次数がk以上の頂点が少なくともk個

- $k^2 \leq$ 次数の合計 $= 2M$。よって、$k \leq \sqrt{2M}$
- $M \leq 200{,}000$なので、$\mathrm{out}[u]$の要素数はどの頂点でも高々632

問題１０　つながる三人組

解法: 共通の隣接頂点の位置により３通りに分ける

- $u \prec v$と並んでおり、 wがu, vの共通の隣接頂点であるとき、u, v, wの順序により３通りに分ける。

ケース	順序	数え方
A	$u \prec v \prec w$	out[u] を走査
B	$u \prec w \prec v$	out[u] を走査
C	$w \prec u \prec v$	common_in に保持（次スライドで説明）

- ケースAとBでは、wはout[u]に含まれるので、辺$v - w$の有無をその場で 確認する。
  - out[u]の要素数は高々632であるので、その場で確認しても計算量はそれほど多くない
- ケースCでは、wはout[u]に含まれないので、別の処理が必要。

問題１０　つながる三人組

common_in の定義と三角形の数の増減

- common_in$[\{x, y\}]$ は、xとyの両方より順序が前の、現在のグラフに含まれる共通の隣接頂点の数
  - この説明における「現在」は、各操作を行う直前という意味
- つまり、 $\mathrm{common_in}[\{x, y\}] = \left|\{w\colon w \prec x, w \prec y, \text{辺} w - x \text{と} w - y \text{が現在存在}\}\right|$
  - 辺$x - y$自体の有無にはよらない
- uとvの共通の隣接頂点wの数（三角形の数の増減）

  $\Delta = \mathrm{common_in}[\{u, v\}] + \mathrm{out}[u]$の走査で見つけた数

問題１０　つながる三人組

common_in の更新: 変化するのはどのペアか

- $w \prec x$である辺$w - x$の状態が変わるとき、以下の３つの条件をすべて満たせば、$\mathrm{common_in}[\{x, y\}]$の数が変わる
  - $x \neq y$である
  - $\mathrm{out}[w]$にyが含まれる（$w \prec y$）
  - 辺$w - y$が現在のグラフに含まれる
- 追加操作のとき: $\mathrm{common_in}[\{x, y\}] \mathrel{+}= 1$
- 削除操作のとき: $\mathrm{common_in}[\{x, y\}] \mathrel{-}= 1$
- また、これらの３条件をすべて満たし、かつ現在のグラフに辺$x - y$が含まれるならば、$\Delta \mathrel{+}= 1$
  - 削除操作でも増やす

問題１０　つながる三人組

解法: 辺$w-x$に関する１回の操作の処理手順

- 最初の操作の前に、$\mathrm{ans}=0$と初期化する

手順	追加	削除
①$w \prec x$となるように、必要ならswap(w,x)する	両操作共通	
②$\Delta=\mathrm{common_in}[\{w,x\}]$	両操作共通	
③out[w] に含まれる頂点を走査	それぞれの条件を満たすとき、Δを増やしcommon_inを増やす	それぞれの条件を満たすとき、Δを増やしcommon_inを減らす
④答えを更新	$\mathrm{ans}\mathrel{+}=\Delta$、現在のグラフに辺$w-x$を追加	$\mathrm{ans}\mathrel{-}=\Delta$、現在のグラフから辺$w-x$を削除
⑤答えを出力	両操作共通	

問題１０　つながる三人組

「③ out[*w*] に含まれる頂点を走査」の実装例

```
for (int y : out[w]) {
    if (x == y) continue;
    // currentEdges: 現在のグラフに含まれる辺を保持
    if (currentEdges.count({w, y}) == 0) continue;

    if (currentEdges.count({x, y}) >= 1) delta++;    // delta: Δ
    if (type == 1) common_in[{x, y}]++;    // 追加の場合
    else common_in[{x, y}]--;                    // 削除の場合
}
```

問題１０　つながる三人組

入力例１の場合

操作（追加: $+$, 削除: $-$）	$\{w, x\}$の順序	走査の回数	common_in の増減	Δ	答え
$+1$	$1 \prec 2$	2	なし	0	0
$+2$	$3 \prec 2$	2	なし	0	0
$+1$	$1 \prec 3$	2	$\text{common_in}[\{2,3\}] \mathrel{+}= 1$	1	1
$+2$	$4 \prec 2$	2	なし	0	1
$+2$	$5 \prec 2$	1	なし	0	1
$+3$	$3 \prec 5$	2	$\text{common_in}[\{2,5\}] \mathrel{+}= 1$	1	2
$+4$	$4 \prec 5$	2	$\text{common_in}[\{2,5\}] \mathrel{+}= 1$	1	3
-1	$1 \prec 3$	2	$\text{common_in}[\{2,3\}] \mathrel{-}= 1$	1	2

問題１０　つながる三人組

時間計算量

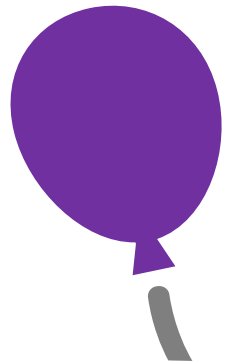
- 前処理（Eの構築、次数の計算、辺の向きつけ）: $O(N+Q)$
- １回の操作（$\mathrm{out}[u]$の高々$\sqrt{2M}$頂点を調べる）: $O(\sqrt{Q})$
  - $M = |E| \leq Q$
- 全体: $O(N + Q\sqrt{Q})$

空間計算量

- 全体: $O(N+Q)$

問題１１　データセンター増強計画（１２点）

正答数:　　1 チーム



問題１１　データセンター増強計画

概要

- 1号棟からN号棟までのN棟のデータセンターの処理能力を増強する。
- i号棟の処理能力の増加量をp_iに設定する。
- 増強工事の作業は何回かに分けて行う。１回の作業は作業１、作業２のいずれか。

  作業１：１つの棟を選び、その棟の処理能力をV増やす。

  作業２：番号が連続したM個の棟を選び、それらの棟の処理能力を１ずつ増やす。
- VとMは工事前も工事中も変わらない。
- 各棟の増加量が設定した値と等しくなるように増強するまでの作業回数の最小値を求める。
- 各棟の増加量が設定した値と等しくなるように増強することができなければ、-1を出力。

制約

$1 \leq N \leq 200{,}000,\ \ 1 \leq V \leq 1{,}000{,}000{,}000,\ \ \ 1 \leq M \leq N,\ \ \ 0 \leq p_i \leq 1{,}000{,}000{,}000$

問題１１　データセンター増強計画

入出力例１

```
3 1 2
1 2 3
```

$N = 3,\ V = 1,\ M = 2$
$p_1 = 1,\ p_2 = 2,\ p_3 = 3$

$\longrightarrow$ 4を出力

入出力例２

```
5 3 2
3 2 1 5 4
```

$N = 5,\ V = 3,\ M = 2$
$p_1 = 3,\ p_2 = 2,\ p_3 = 1,\ p_4 = 5,\ p_5 = 4$

$\longrightarrow$ -1を出力

入出力例３

```
3 2 3
10 10 10
```

$N = 3,\ V = 2,\ M = 3$
$p_1 = 10,\ p_2 = 10,\ p_3 = 10$

$\longrightarrow$ 10を出力

問題１１　データセンター増強計画

考え方

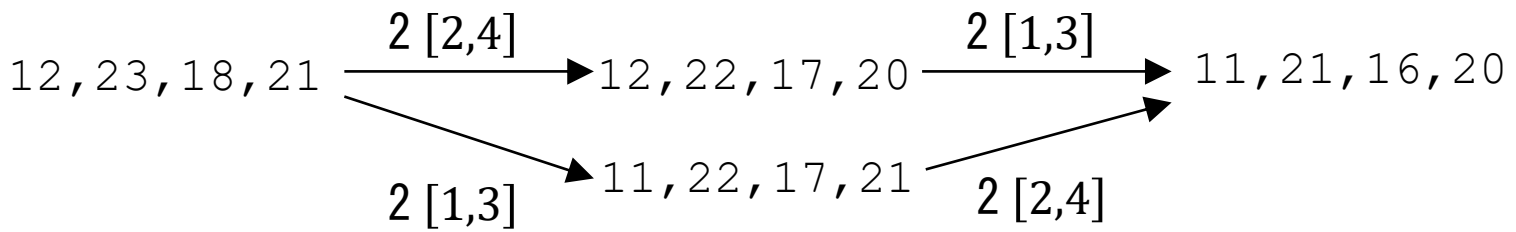
- 問題を「以下の２つの操作を使って、数列$p_1, p_2, \ldots, p_N$の各要素をすべて0にするのに必要な最小の操作回数を求める」と読み替える。
  - 操作１：要素iをV減らす。
  - 操作２：区間$[i, i+M-1]$ $(1 \leq i \leq N-M+1)$の各要素を１ずつ減らす。
- 要素が負になったら、どのような操作を行っても0にはできないことに注意。

どのような順番で操作を行うか？

- 連続する２つの操作を入れ替えても、得られる結果は変わらない。

例：$N=4, V=M=3$で、操作２を２回行ったとき

2 [2,4]は、区間[2,4]に操作２を行ったことを示す。



数列への操作列の中の操作の順番を自由に入れ替えても、同じ結果が得られる。

問題１１　データセンター増強計画

考え方（続き）

- 数列の各要素を0にするためのポイントは、数列の各要素pから$p \bmod V$を取り除いてVの倍数にすること。
- Vの倍数になれば、操作１、操作２のどちらを使っても0にすることができるので、操作回数が最小になるような操作を選べばよい。
- 数列の全要素を0にする操作の列があれば、その中の操作の順番をどのように入れ替えても0にできる。
- したがって、全要素を0にできる操作列があるなら、はじめに全要素をVの倍数にする操作を行うように操作の順番を入れ替えても、全要素を0にできる。
- 以下の２ステップで数列を操作して、全要素を0にできるかどうかを判定する。
  1. 全要素をVの倍数にするための操作を行う。全要素をVの倍数にできなければ終了。
  2. 全要素をVの倍数にできたら、操作の数が最小になるように操作１，２を使って全要素を0にする。ステップ１と２で得られた操作回数の和を出力して終了。

問題１１　データセンター増強計画

数列の要素をVの倍数にする方法

- 要素に操作１を行っても、要素をVで割った余りは変わらない。
  $V = 5$、要素が12のとき、操作１を行う回数が１回でも２回でも、余りは2。
- したがって、要素をVの倍数にするには、操作２を行う必要がある。
- 数列に対して左から右へ向かって順番に、操作２を0回以上行う。
- まず、区間$[1, M]$について要素１がVの倍数になるまで操作２を行う。次に$[2, M+1]$について要素２がVの倍数になるまで操作２を行う。これを$[N-M+1, N]$まで行う。

$N = 4, V = 5,\ M = 3$のとき

$$12,23,18,21 \xrightarrow{2\ [1,3] \times 2} 10,21,16,21 \xrightarrow{2\ [2,4]} 10,20,15,20$$

$2\ [1,3] \times 2$は、区間$[1,3]$に操作２を
２回行うことを示す。

問題１１　データセンター増強計画

数列の要素をVの倍数にする方法（続き）

- ある区間に操作２を行うとき、区間中の全要素を減らすと$O(NM)$かかる。
- 尺取り法を使えば、$O(N)$でできる。

$N = 4, V = 5,\ M = 3$のとき

$$12,23,18,21 \xrightarrow{2\ [1,3]\times 2} 10,21,16,21 \xrightarrow{2\ [2,4]} 10,20,15,20$$

尺取り法による処理

$$12,23,18,21 \longrightarrow 10,23,18,21 \longrightarrow 10,20,18,21 \longrightarrow 10,20,15,21 \longrightarrow 10,20,15,20$$

dec[1]=2
sub=0+dec[1]=2

$23 - \text{sub} - 1 = 20$
dec[2]=1
sub=2+dec[2]=3

$18 - \text{sub} - 0 = 15$
dec[3]=0
sub=3+dec[3]-dec[1]=1

$21 - \text{sub} - 0 = 20$
dec[4]=0
sub=1+dec[4]-dec[2]=0

問題１１　データセンター増強計画

全要素を最小の操作数で0にする方法

- ここまでで、全要素はVの倍数になっている。
- 全要素をVで割る。得られた数列に対して、以下の操作を行う。
  - 操作１’：要素iを１減らす（コスト1）
  - 操作２’：区間$[i, i+M-1]$（$1 \leq i \leq N-M+1$）の各要素を１ずつ減らす（コストV）
- コスト最小になる操作列のコストを求めれば、それがVの倍数から成る数列の全要素を0にするのに必要な操作１，２の最小の操作回数。
- 区間$[i, i+M-1]$に操作２’を行うとコストはV、同じ区間の各要素に１回ずつ操作１’を行うとコストはM。
- $V \geq M$と$V < M$で場合分けする。
- $V \geq M$のときは、操作１’だけを行う方がコストが小さい（$V = M$のときは同じ）。
- $V < M$のときは、可能な限り操作２’を行う方がコストが小さい。

問題１１　データセンター増強計画

$1(2) \times 3$は、要素2に操作１を３回行ったことを示す。

$V \geq M$の場合

$N = 3, V = 5,\ M = 3$のとき

操作１’を行うとき　$3,2,3 \xrightarrow[\text{コスト}\ 1 \times 3 = 3]{1(1) \times 3} 0,2,3 \xrightarrow[\text{コスト}\ 1 \times 2 = 2]{1(2) \times 2} 0,0,3 \xrightarrow[\text{コスト}\ 1 \times 3 = 3]{1(3) \times 3} 0,0,0$　コストは８

操作２’を行うとき　$3,2,3 \xrightarrow[\text{コスト}\ 5 \times 2 = 10]{2\ [1,3] \times 2} 1,0,1 \xrightarrow[\text{コスト}\ 1 \times 1 = 1]{1(1)} 0,0,1 \xrightarrow[\text{コスト}\ 1 \times 1 = 1]{1(3)} 0,0,0$　コストは12

- 操作１’だけを行って全要素を0にするほうがコストが小さい（$V = M$のときはどのような操作列でも同じコストなので、操作１’だけを行うことにしてよい）。
- 数列が$a_1, a_2, \ldots, a_N$のとき、操作１’だけを行って全要素を0にするコストは$a_1 + a_2 + \cdots + a_N$。
- これがVの倍数から成る数列の全要素を0にするのに必要な操作１，２の最小の操作回数。

問題１１　データセンター増強計画

$V < M$の場合

$N = 3, V = 1,\ M = 3$のとき

操作１’を行うとき　$3,2,3 \xrightarrow[\text{コスト}\ 1\times 3 = 3]{1(1)\times 3} 0,2,3 \xrightarrow[\text{コスト}\ 1\times 2 = 2]{1(2)\times 2} 0,0,3 \xrightarrow[\text{コスト}\ 1\times 3 = 3]{1(3)\times 3} 0,0,0$　コストは８

操作２’を行うとき　$3,2,3 \xrightarrow[\text{コスト}\ 1\times 2 = 2]{2\,[1,3]\times 2} 1,0,1 \xrightarrow[\text{コスト}\ 1\times 1 = 1]{1(1)} 0,0,1 \xrightarrow[\text{コスト}\ 1\times 1 = 1]{1(3)} 0,0,0$　コストは４

- 可能な限り操作２’を行うほうがコストが小さい。
- 数列を左から右へ向かって順番に、操作２’を0回以上行う。
- まず、区間$[1, M]$についてその区間の最小の要素が0になるまで操作２’を行う。次に$[2, M+1]$について同様に操作２’を行う。これを$[N-M+1, N]$まで行う。
- 0でないすべての要素に、全要素が0になるまで操作１’を行う。
- 区間$[i, i+M-1]$の要素の最小値を効率よく求めるには、遅延セグメント木などを使えばよい。

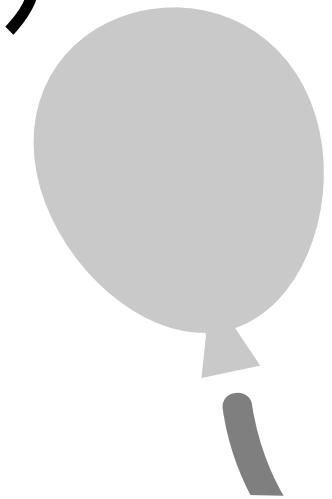
問題１１　データセンター増強計画

計算量

- 全要素をVの倍数にする処理：尺取り法を使えば$O(N)$。
- 全要素がVの倍数の数列の全要素を0にするための最小の操作回数を求める処理：
  - $V \geq M$のとき：$O(N)$
  - $V < M$のとき：遅延セグメント木などを使って、尺取り法も使えば$O(N \log N)$。
- 以上より、計算量は$O(N \log N)$。

問題１２ データ伝送（１３点）

正答数: 0 チーム



問題１２　データ伝送

概要

- 辺の重みがc_iであるN頂点の木と、整数Mが与えられる。
- 以下の条件を満たすように、木の上を移動する。
  - 訪問した頂点から、番号が厳密に増加するM 個の頂点を順に選べる。
- 移動距離の合計を最小化せよ。

制約

$1 \leq N \leq 200{,}000$、$1 \leq M \leq 20$、$1 \leq c_i \leq 1{,}000{,}000$

問題１２　データ伝送

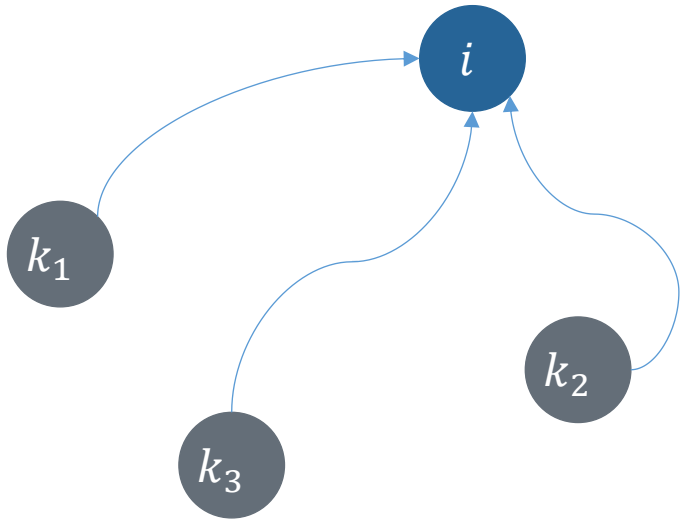
素朴解：前の頂点 k をすべて試す

dp[i][j]：頂点 i で終わる長さ j の増加部分列を実現する最小コスト

初期値　**dp[i][1] = 0**

遷移　dp[i][j] = min { dp[k][j−1] + dist(k, i) }
k < i

各 (i,j) について、すべての $k < i$ を調べる。



各 k < i から頂点 i への遷移を考える

計算量

$\Omega(N^2M) \rightarrow \quad N \leq 200{,}000$ では間に合わない。

問題１２　データ伝送

高速化

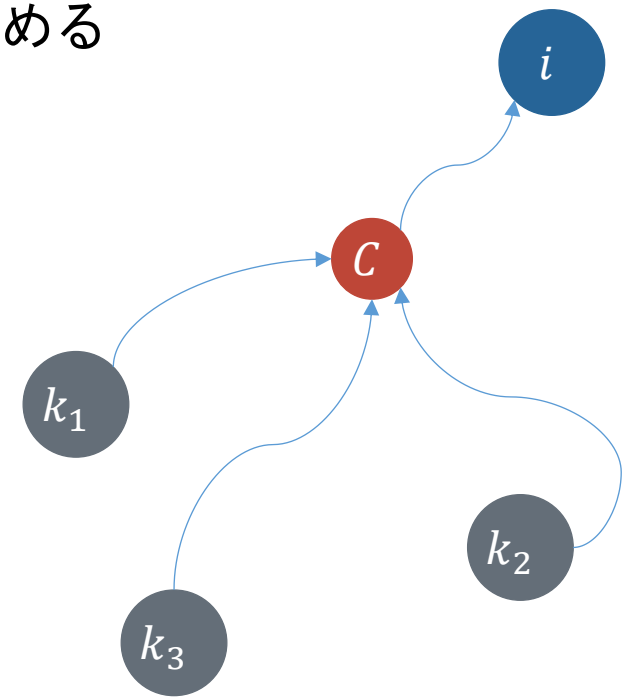
ナイーブ解では、過去の頂点kをすべて調べる　→　経由点cに情報を集める

経由点 c を用いて、候補のコストをまとめられないか？

dp[k][j-1] + dist(k,c) + dist(c,i)

各 c に過去の k について dp[k][j-1] + dist(k,c) の最小値を保存

ただし、経由点を少数に絞り、最短距離を正しく求める工夫が必要



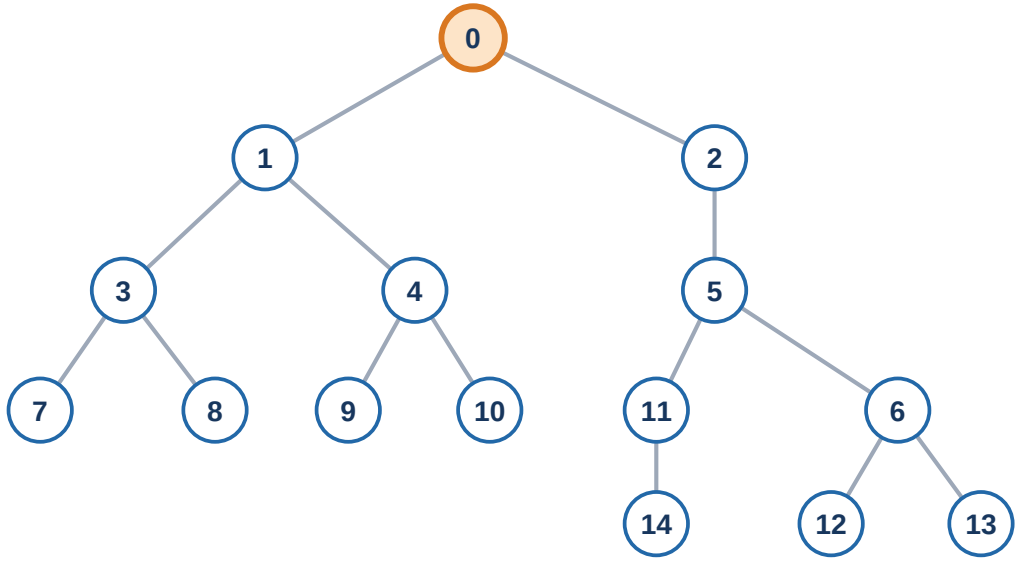
問題１２　データ伝送

重心分解

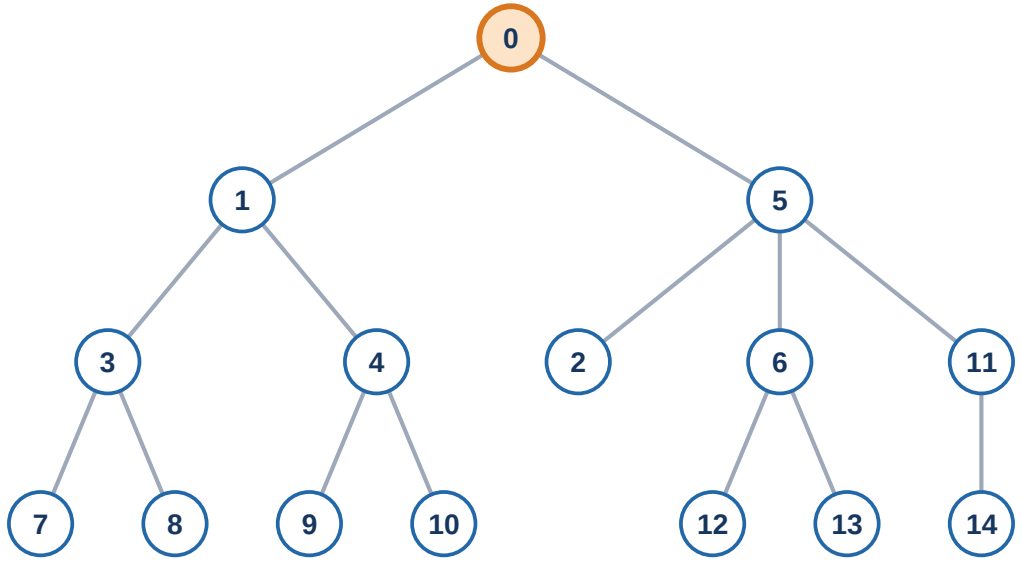
重心：取り除いた後の各連結成分の大きさが、元の半分以下になる頂点。

① 重心を選ぶ　② 重心を除いて分割　③ 各部分木でも繰り返す

元の木（15頂点）



重心分解木（※）



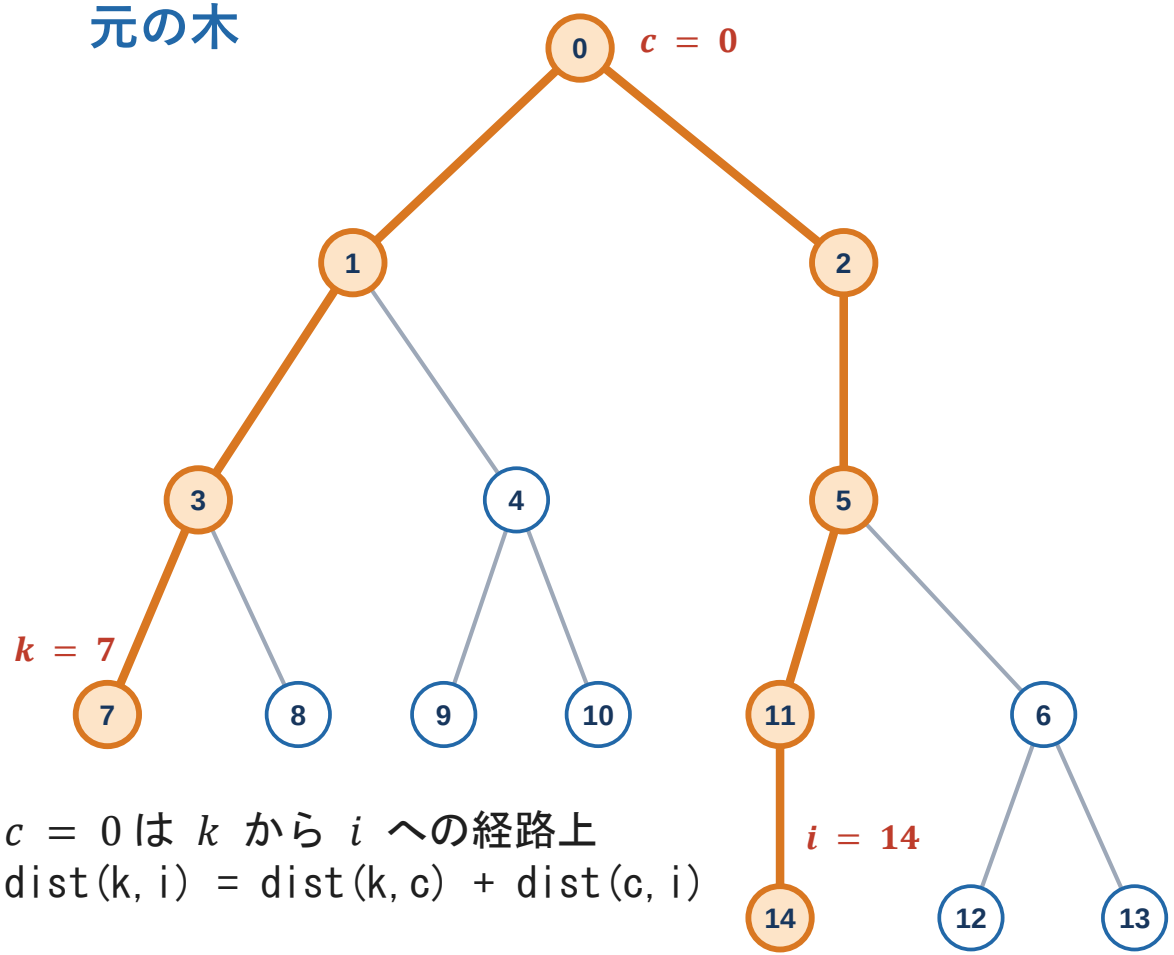
高さは $O(\log N)$

各頂点から根までの祖先も $O(\log N)$ 個。

問題１２　データ伝送

例：　$k = 7$、$i = 14$、経由点 $c = 0$

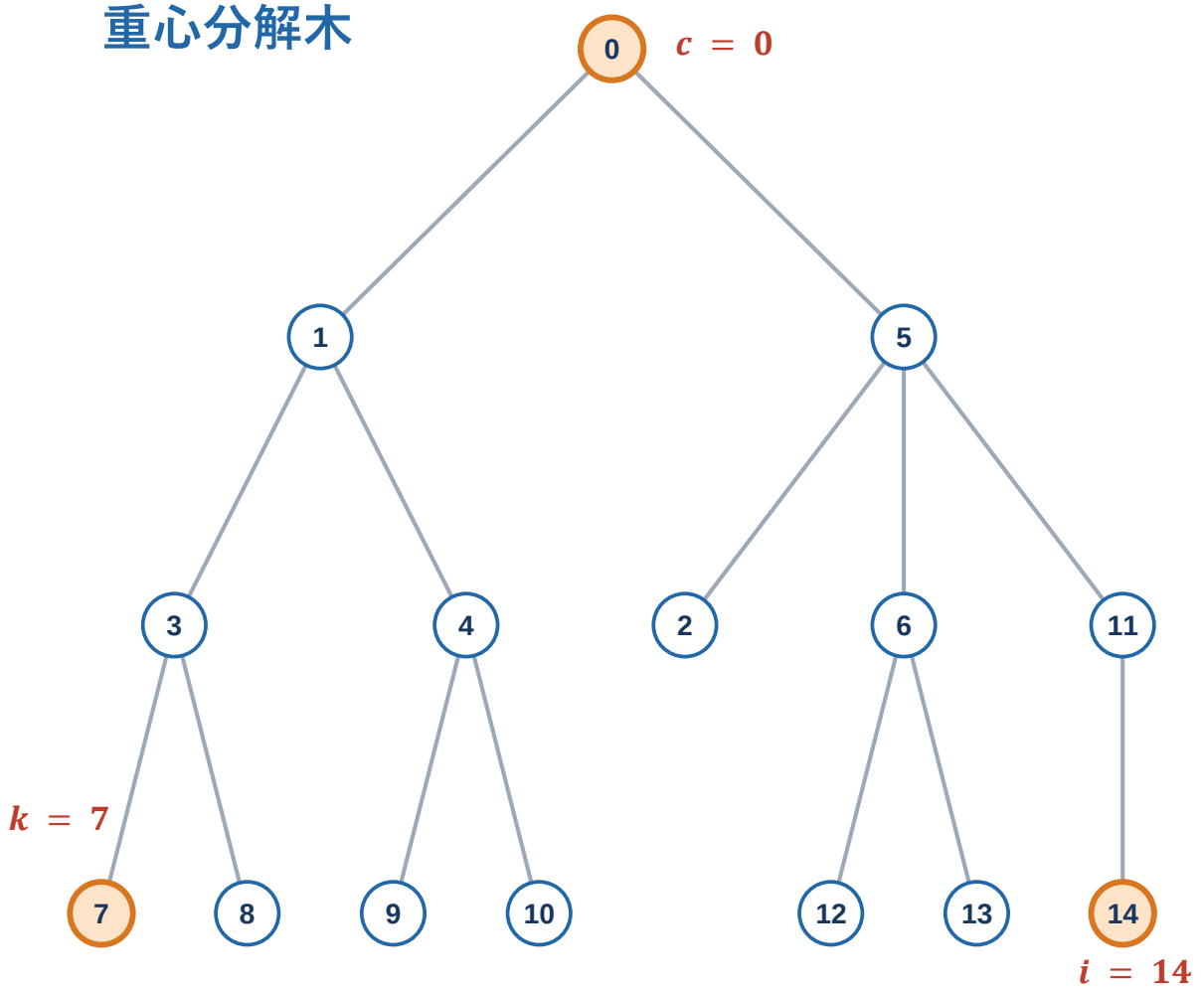
元の木



$c = 0$ は k から i への経路上

dist(k, i) = dist(k, c) + dist(c, i)

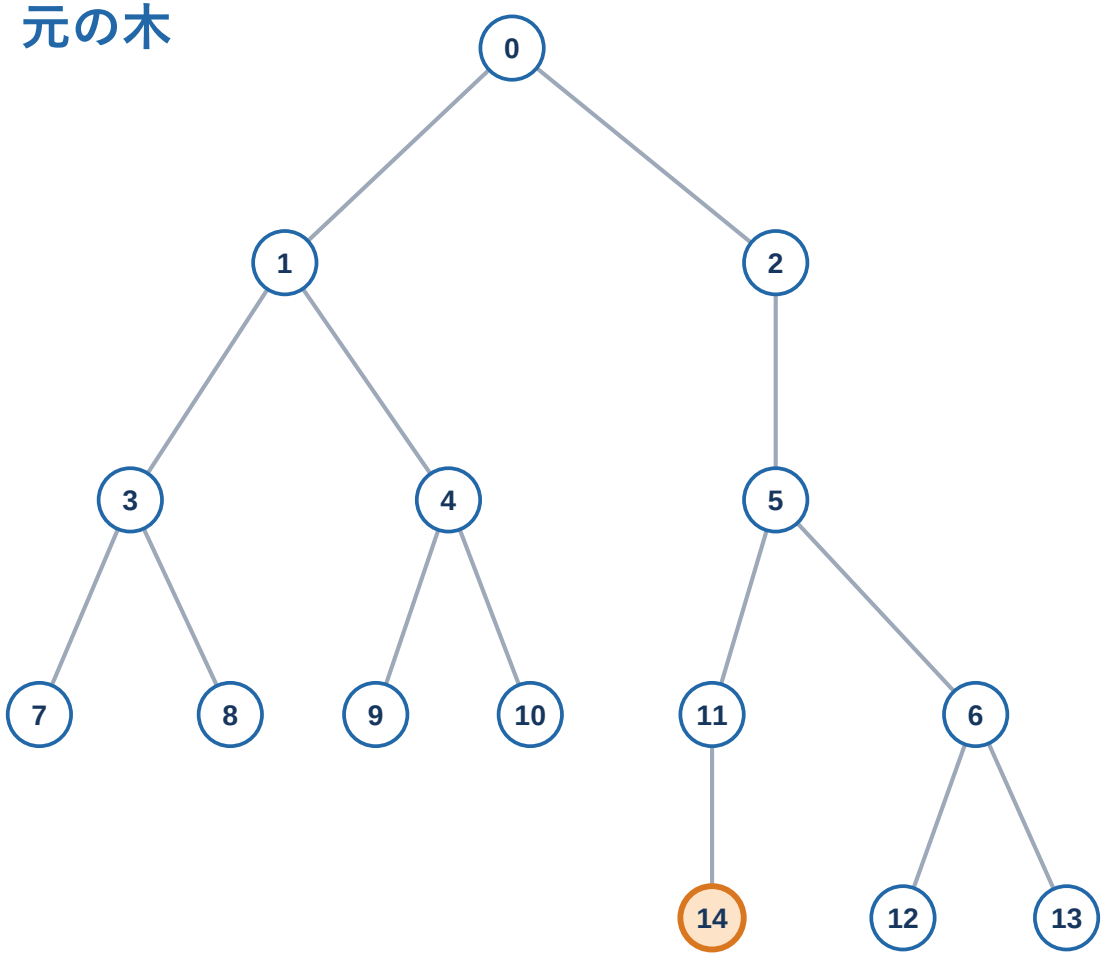
重心分解木



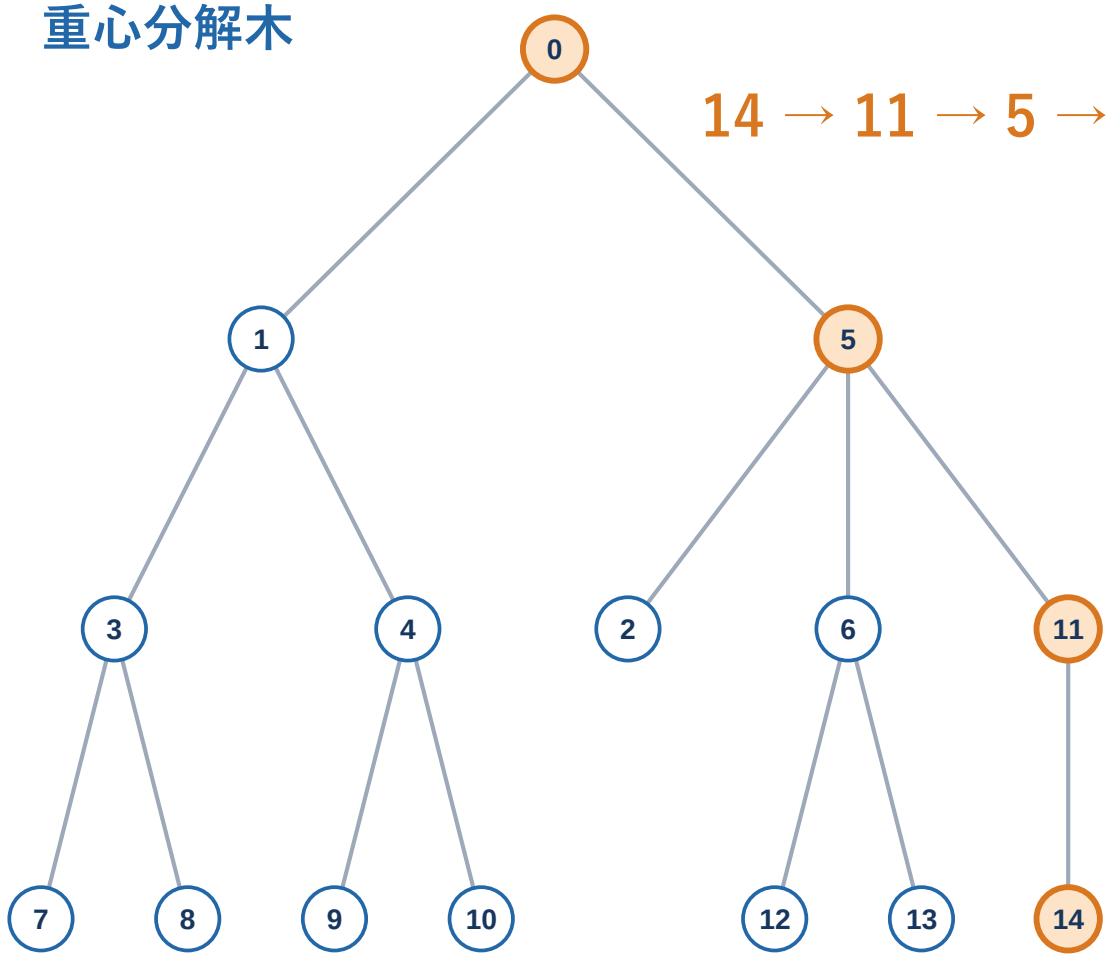
問題１２　データ伝送

頂点 $i = 14$ から重心分解木の根までをたどり、通ったノードを経由点とした情報を参照する

元の木



重心分解木



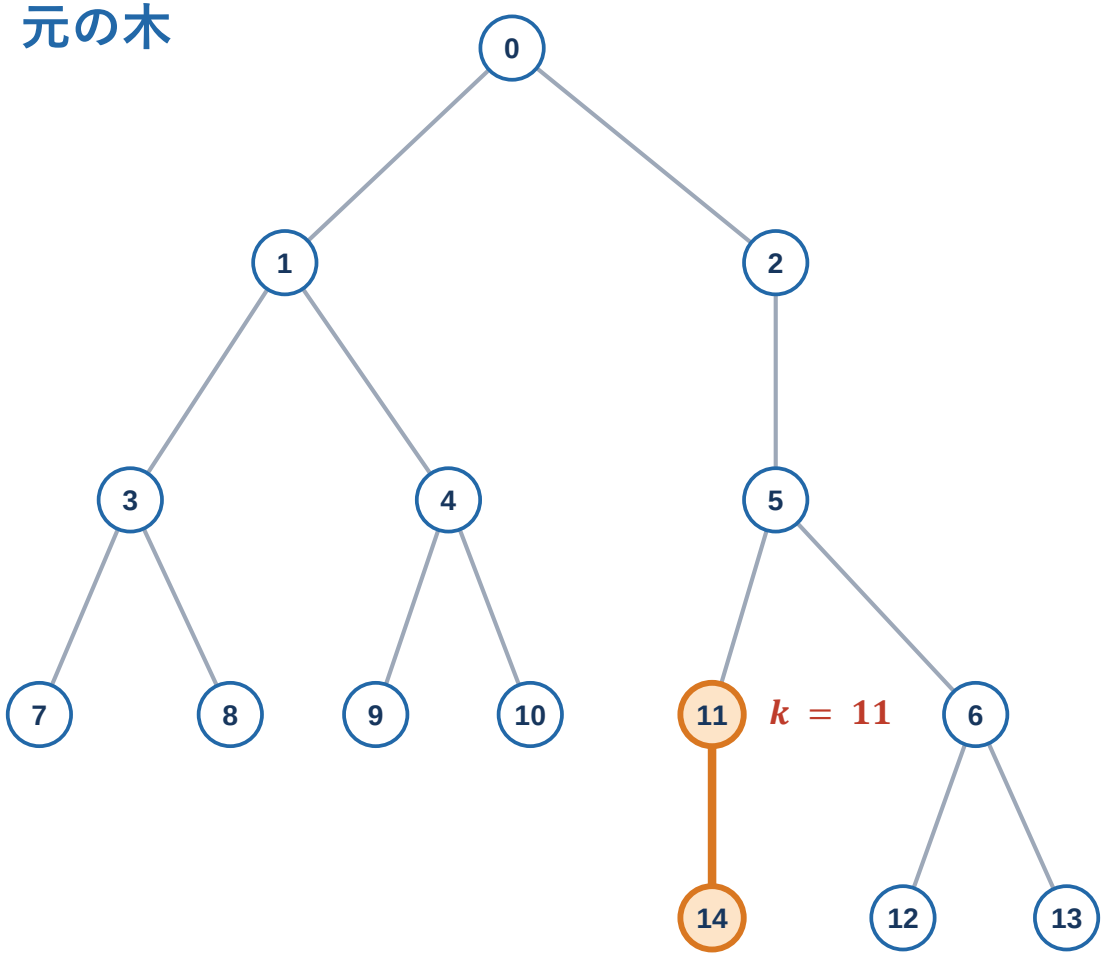
14 自身から根 0 までの4頂点を調べる。

問題１２　データ伝送

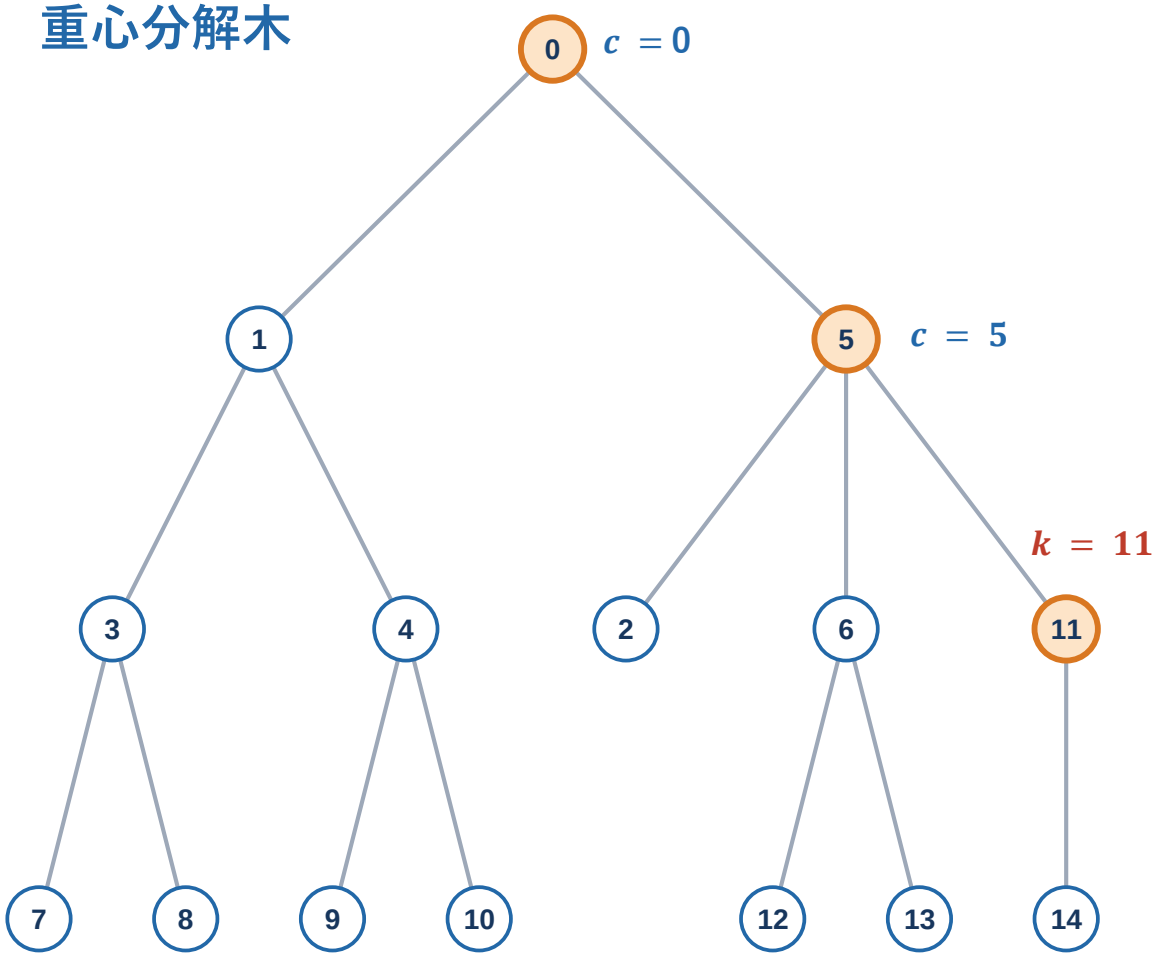
ep[c][j] = min { dp[k][j] + dist(k,c) }
（k ＜ i、c は 重心分解木のkの祖先）

ep[c][j]：経由点 c に集めた過去の最小コスト

元の木



重心分解木



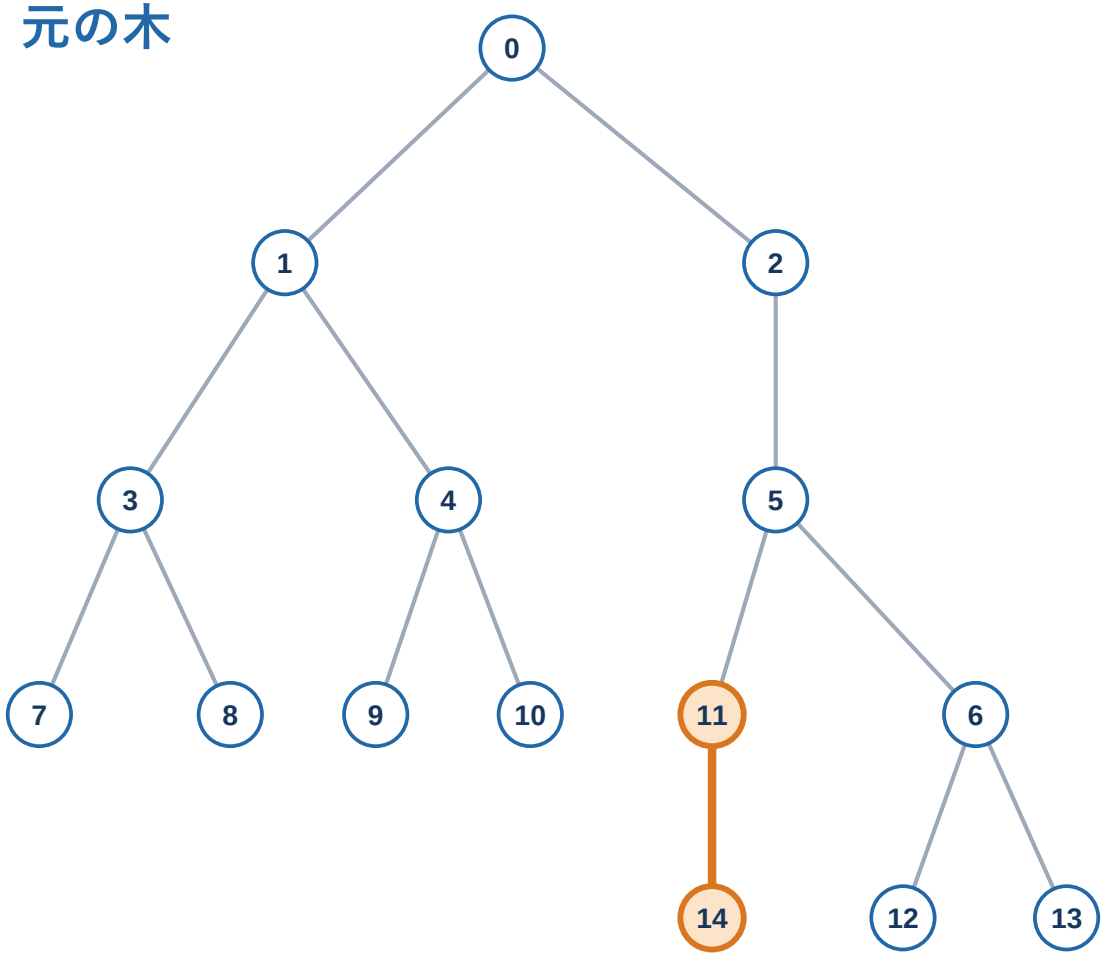
問題１２　データ伝送

各経由点 c について「過去の最小コスト + c から14への距離」を計算

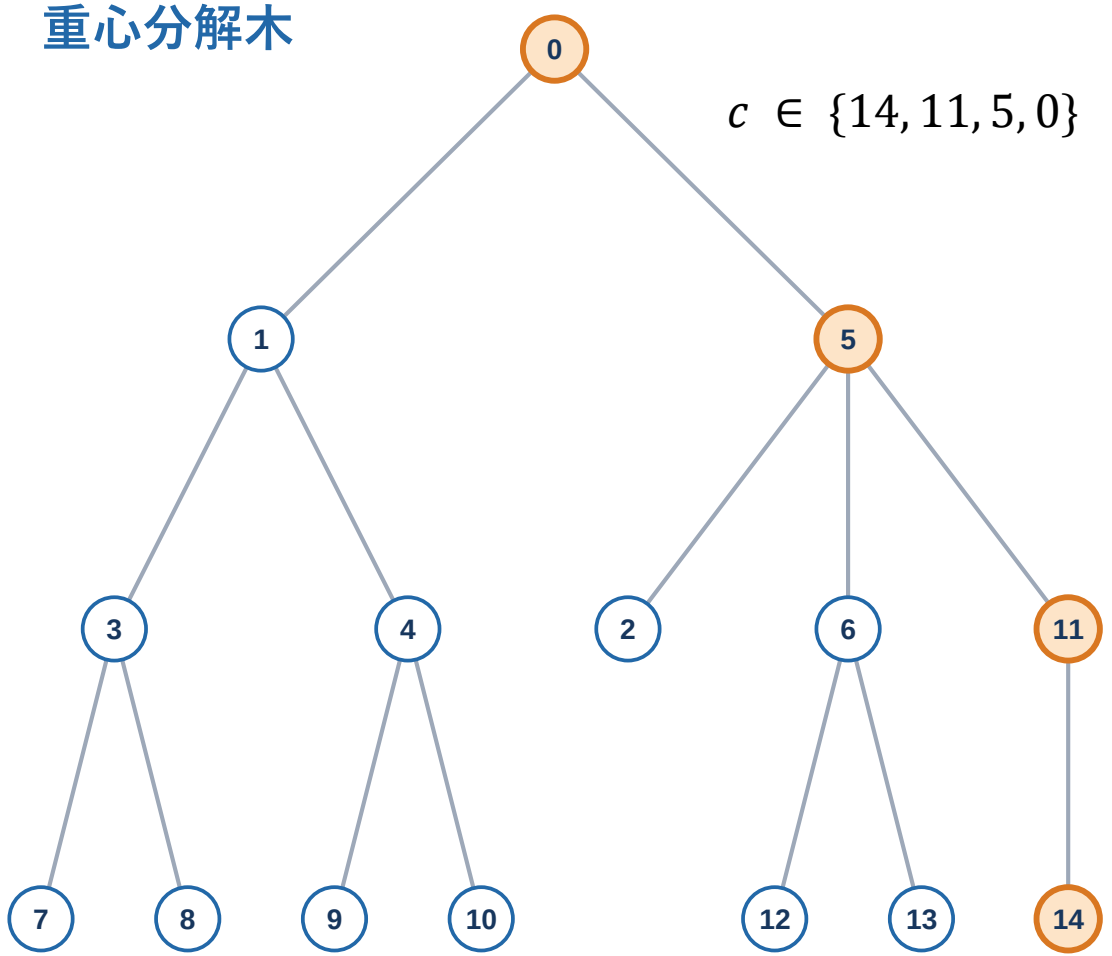
現在の頂点 $i = 14$ の dp[14][j] を計算する

dp[14][j] = min { ep[c][j-1] + dist(14,c) }

元の木



重心分解木



$c \in \{14, 11, 5, 0\}$

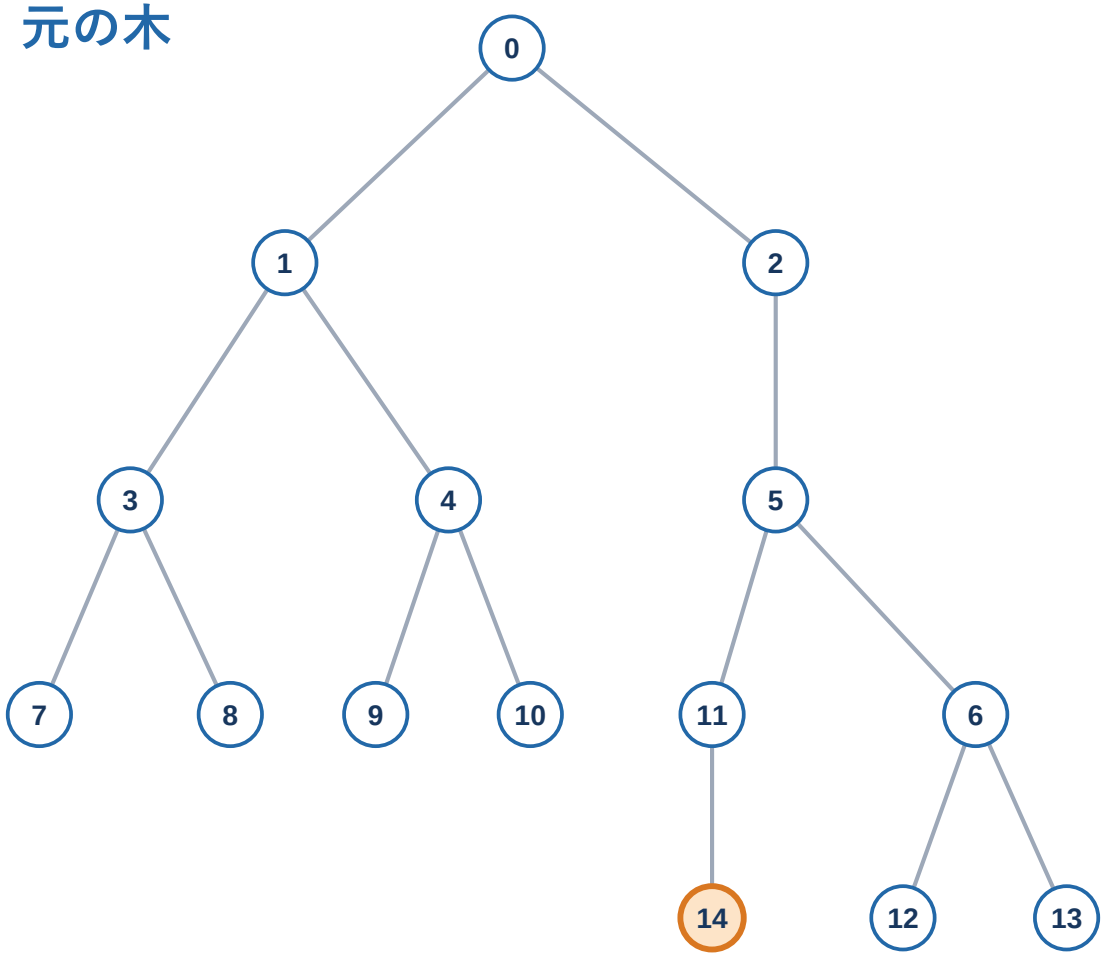
問題１２　データ伝送

dp の計算後、現在の頂点 $i = 14$ を ep に登録する

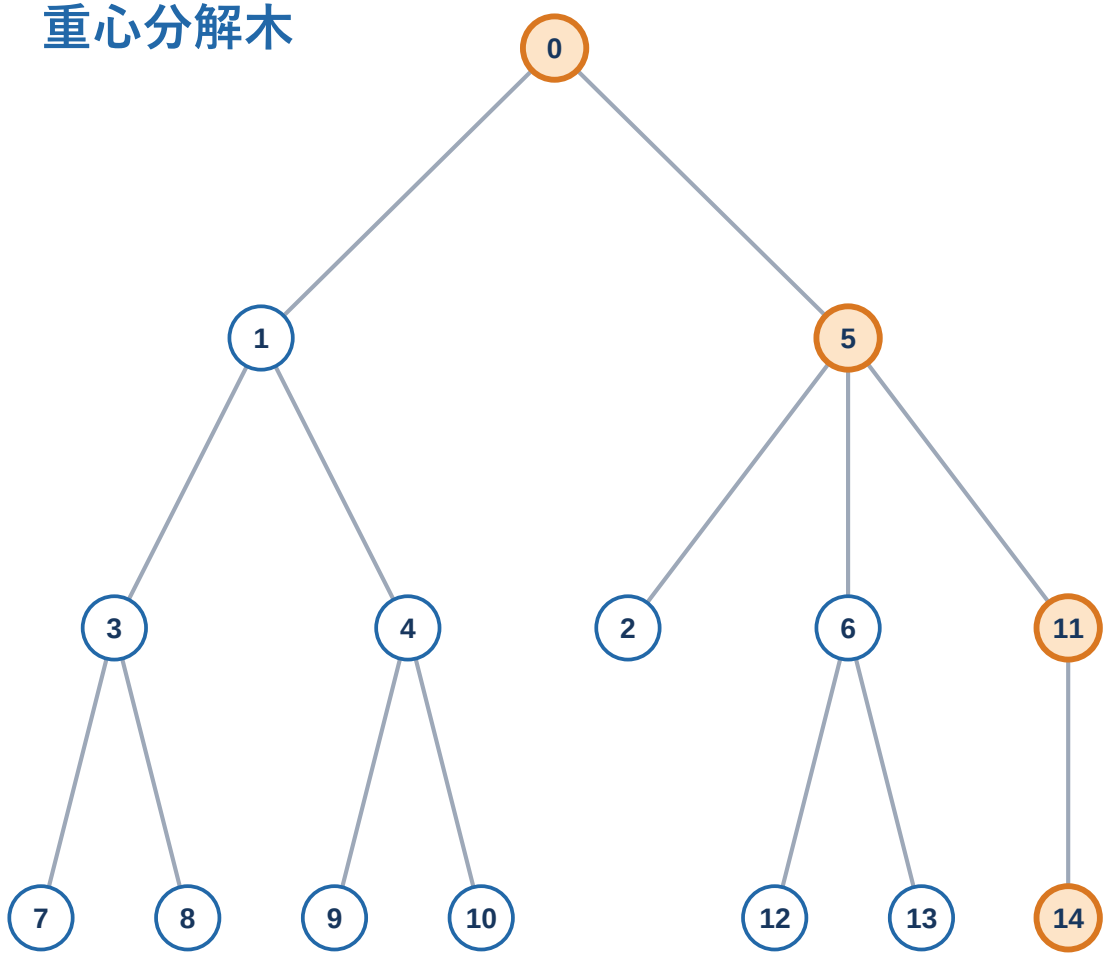
先に dp を計算 → その後 ep を更新（$k < i$ を守る）

```
ep[c][j] ← min { ep[c][j], dp[14][j] + dist(14,c) }
```

元の木



重心分解木



問題１２　データ伝送

実装：各 i で dp を計算してから ep を更新する

```
for (int i = 1; i <= N; i++) {
    vector<ll> dp(M + 1, INF);
    dp[1] = 0;
    for (int j = 2; j <= M; j++) {
        for (auto [c, d] : anc[i]) {
            if (ep[c][j-1] != INF)
                dp[j] = min(dp[j], ep[c][j-1] + dist(i,c));
        }
    }
    ans = min(ans, dp[M]);
    for (auto [c, d] : anc[i])
        for (int j = 1; j <= M; j++)
            if (dp[j] != INF)
                ep[c][j] = min(ep[c][j], dp[j] + dist(i,c));
}
```

問題１２　データ伝送

距離の計算：dist(i,c) は重心分解の各段階で求めておく

- 各重心 c から、現在の連結成分を探索する。
- 木では経路が一意なので、BFSまたはDFSでその距離を計算できる。

anc[i] に (c, dist(i,c)) を保存する。

計算量

- 重心分解と距離前計算：$O(N \log N)$
- dp と ep の更新：$O(NM \log N)$
- 全体：$O(NM \log N)$

問題１３　植物園の栽培計画　（１３点）

正答数:　　0 チーム



問題１３　植物園の栽培計画

概要

- N個の植物とQ個の栽培計画があるとき、各栽培計画に適している植物の数を求める
- 各植物には、好む環境の型t_i、給水指数の基準値w_i 、日照指数の基準値s_i 、基準値からのずれに対する耐性値c_iがある
- 各栽培計画には、給水指数p_iと日照指数r_iが定められている
- 植物がある栽培計画に適しているのは、基準値からのずれがその植物の耐性値以下であり、かつ、その植物の型に応じた次の条件を満たす場合
  - A型：給水指数と日照指数が、どちらもその植物の基準値以上である
  - B型：給水指数がその植物の基準値以下であり、日照指数がその植物の基準値以上である
  - C型：給水指数と日照指数が、どちらもその植物の基準値以下である
  - D型：給水指数がその植物の基準値以上であり、日照指数がその植物の基準値以下である
- 基準値からのずれとは、「給水指数の差の絶対値」と「日照指数の差の絶対値」の合計である
- ここで、「給水指数の差の絶対値」とは、栽培計画の給水指数と、その植物の給水指数の基準値との差の絶対値を表す。「日照指数の差の絶対値」も同様に、栽培計画の日照指数と、その植物の日照指数の基準値との差の絶対値を表す

問題１３　植物園の栽培計画

制約

植物の数　　$1 \leq N \leq 200{,}000,$
栽培計画の数　$1 \leq Q \leq 200{,}000$
植物の型　　t_i：A，B，C，Dのいずれか
植物の給水指数の基準値、日照指数の基準値、基準値からのずれ
$0 \leq w_i, s_i, c_i \leq 1{,}000{,}000{,}000$
栽培計画の給水指数、日照指数
$0 \leq p_i, r_i \leq 1{,}000{,}000{,}000$

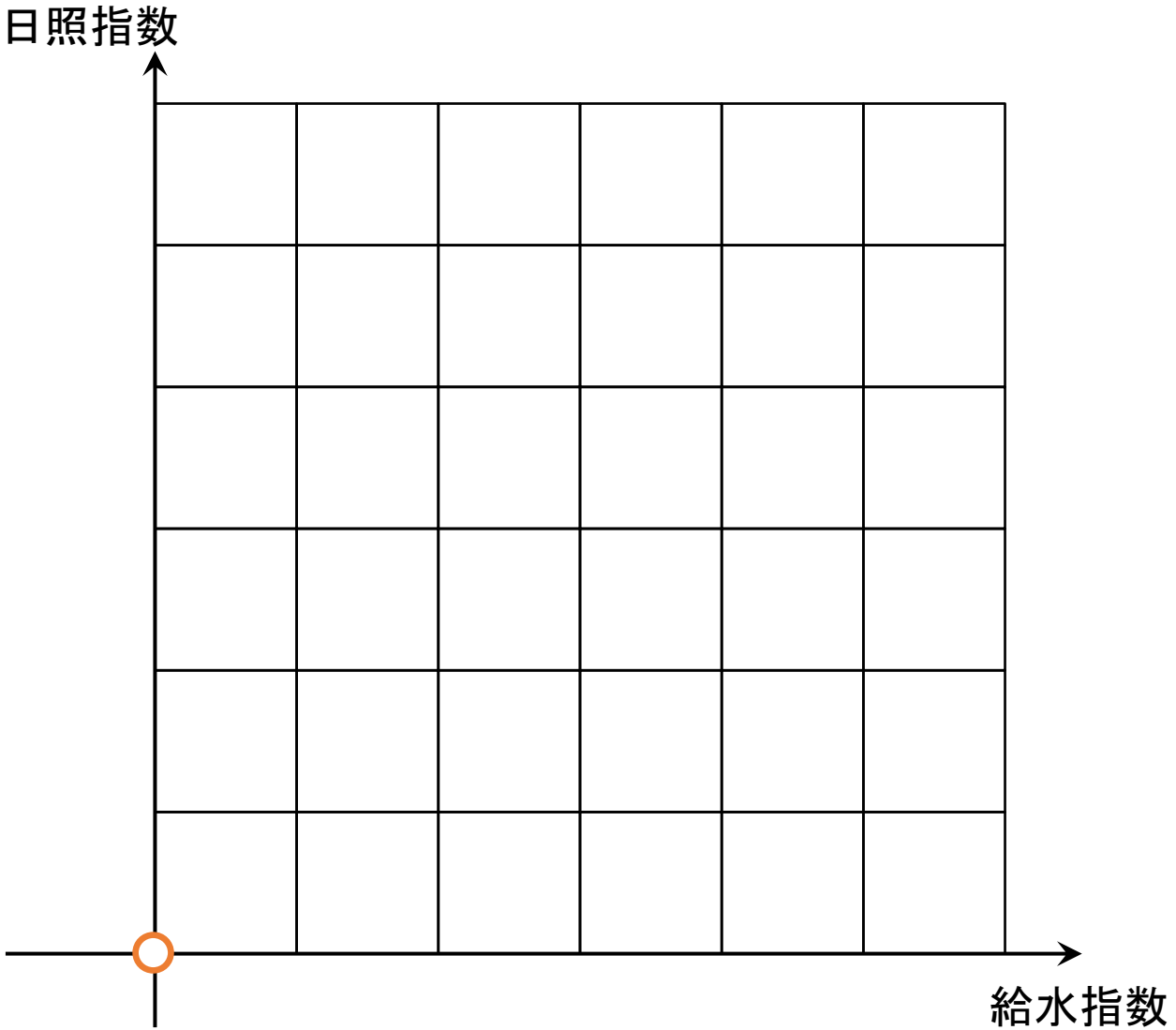
問題１３　植物園の栽培計画

入出力例

```
4 5
A 0 0 4
B 3 0 3
C 3 3 2
D 0 3 3
1 1
2 1
3 1
1 2
4 4
```

- A型：給水指数と日照指数が、どちらもその植物の基準値以上である。

(給水指数の基準値,
日照指数の基準値) ＝ $(0, 0)$



問題１３　植物園の栽培計画

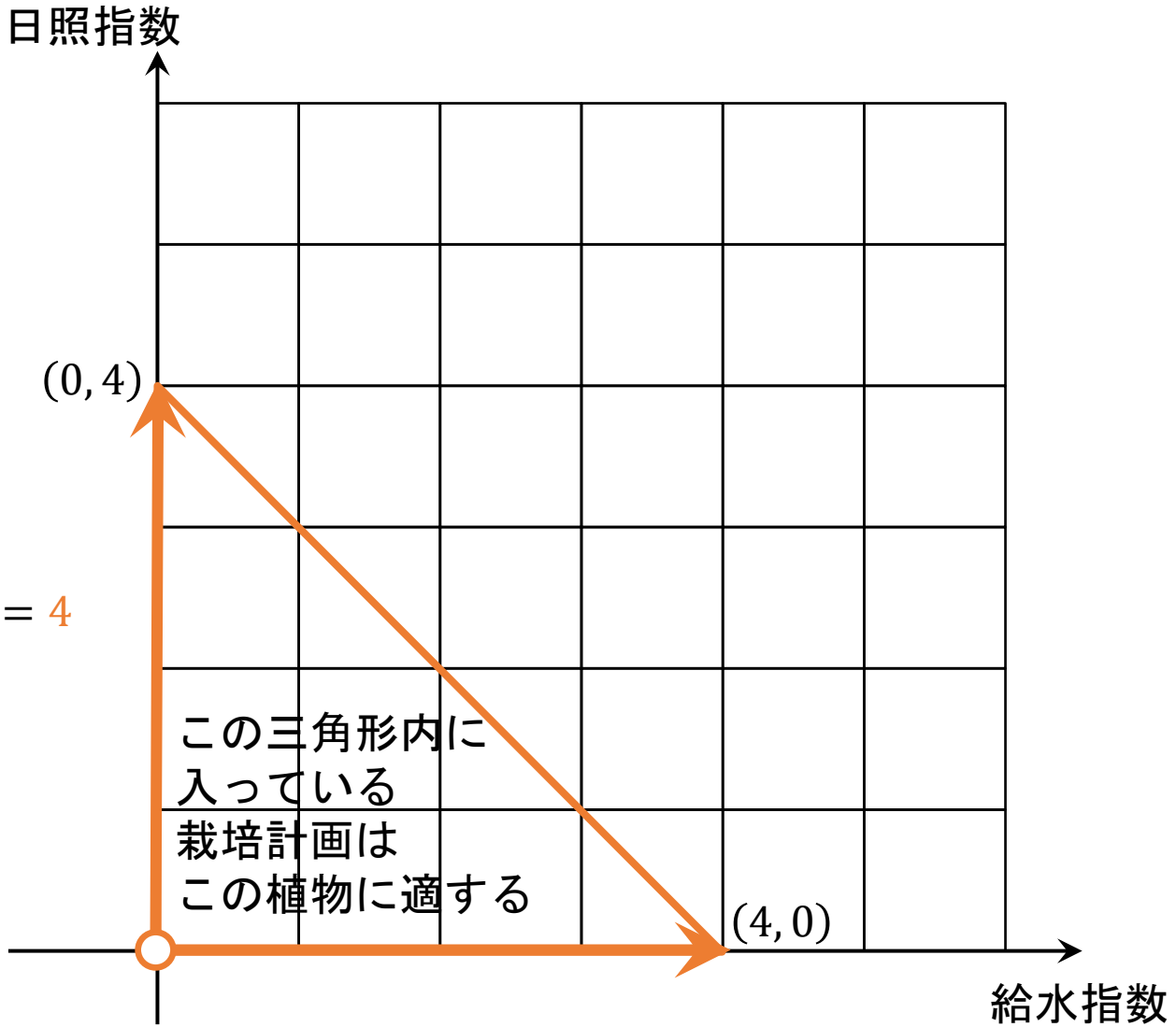
入出力例

```
4 5
A 0 0 4
B 3 0 3
C 3 3 2
D 0 3 3
1 1
2 1
3 1
1 2
4 4
```

- A型：給水指数と日照指数が、どちらもその植物の基準値以上である。

(給水指数の基準値,
日照指数の基準値) ＝ $(0, 0)$

基準値からのずれに対する耐性値 ＝ 4



問題１３　植物園の栽培計画

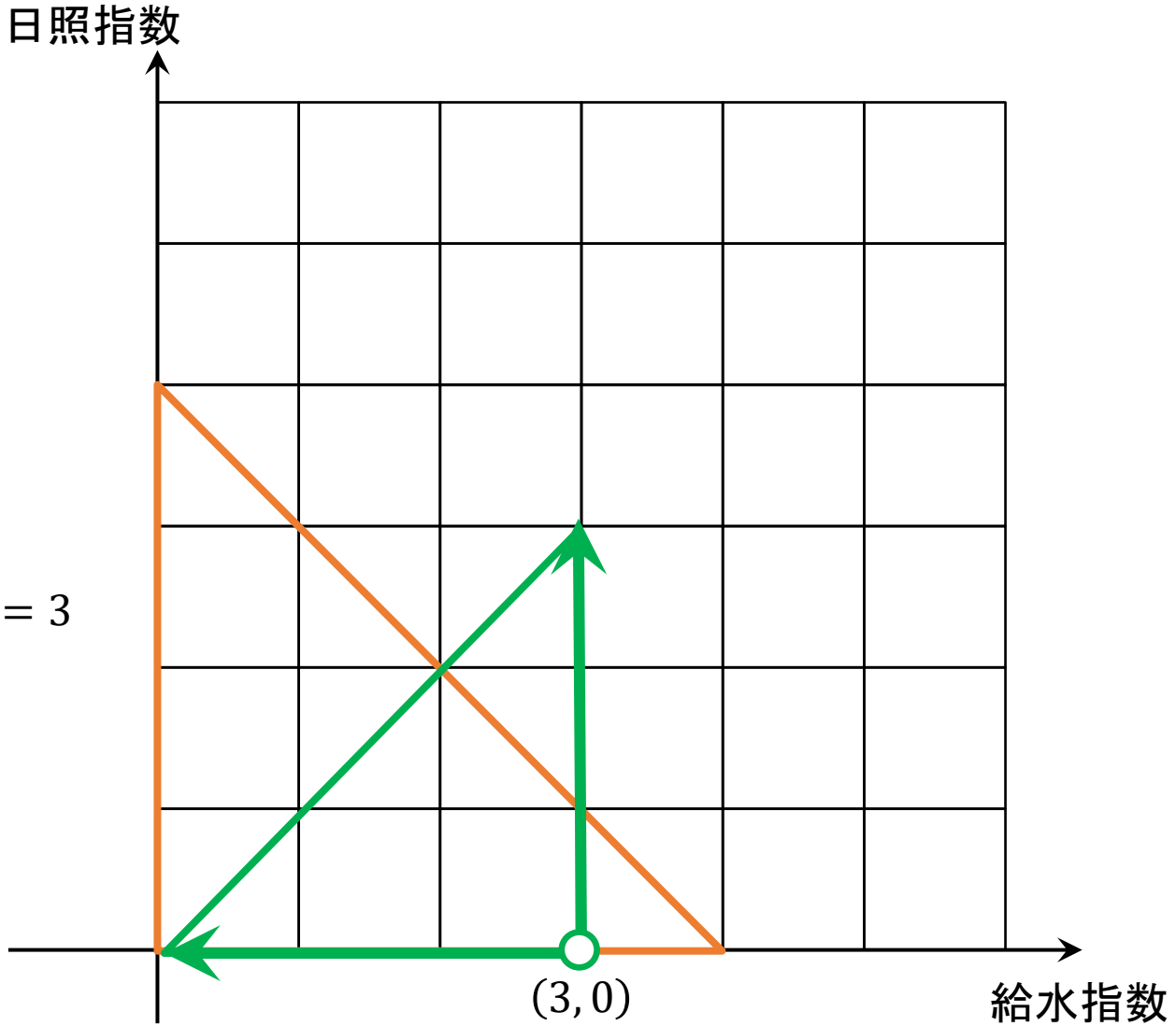
入出力例

```
４５
Ａ００４
Ｂ３０３
Ｃ３３２
Ｄ０３３
１１
２１
３１
１２
４４
```

- B型：給水指数がその植物の基準値以下であり、日照指数がその植物の基準値以上である。

(給水指数の基準値, 日照指数の基準値) ＝ $(3, 0)$

基準値からのずれに対する耐性値 ＝ 3



問題１３　植物園の栽培計画

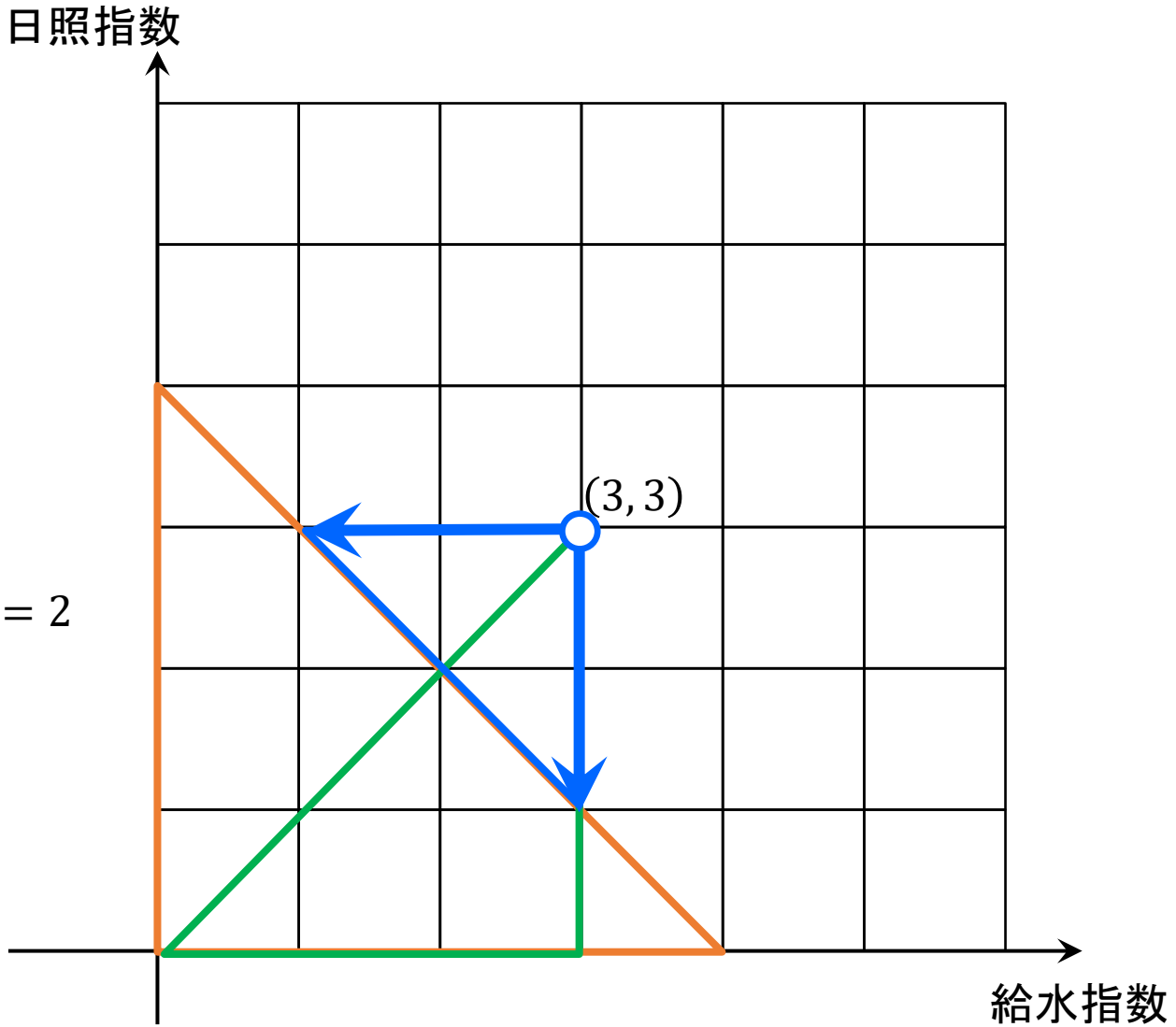
入出力例

```
４５
Ａ００４
Ｂ３０３
Ｃ３３２
Ｄ０３３
１１
２１
３１
１２
４４
```

- C型：給水指数と日照指数が、どちらもその植物の基準値以下である。

(給水指数の基準値,
日照指数の基準値) ＝ (3, 3)

基準値からのずれに対する耐性値 = 2



問題１３　植物園の栽培計画

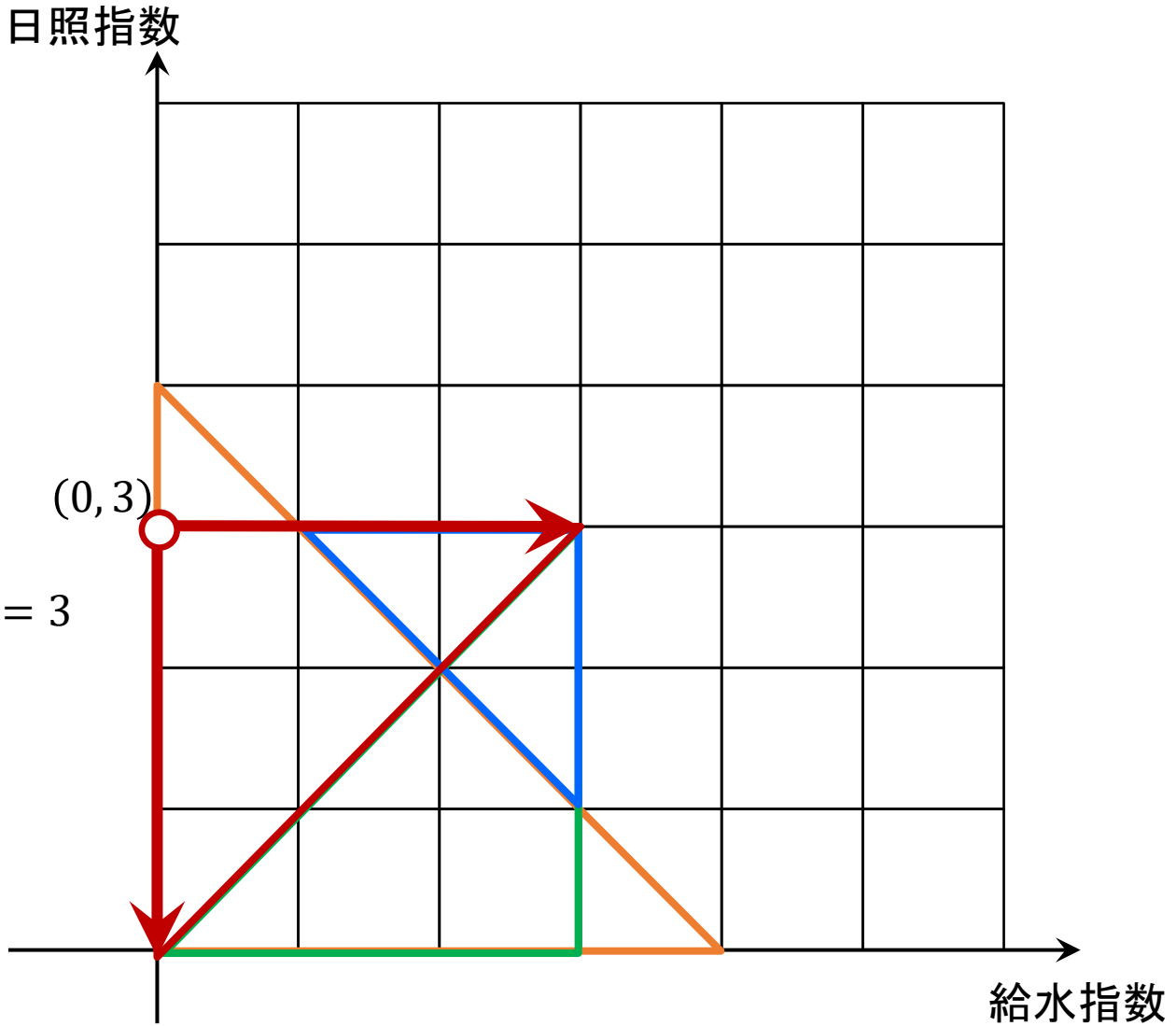
入出力例

```
4 5
A 0 0 4
B 3 0 3
C 3 3 2
D 0 3 3
1 1
2 1
3 1
1 2
4 4
```

- D型：給水指数がその植物の基準値以上であり、日照指数がその植物の基準値以下である

(給水指数の基準値,
日照指数の基準値) ＝ $(0, 3)$

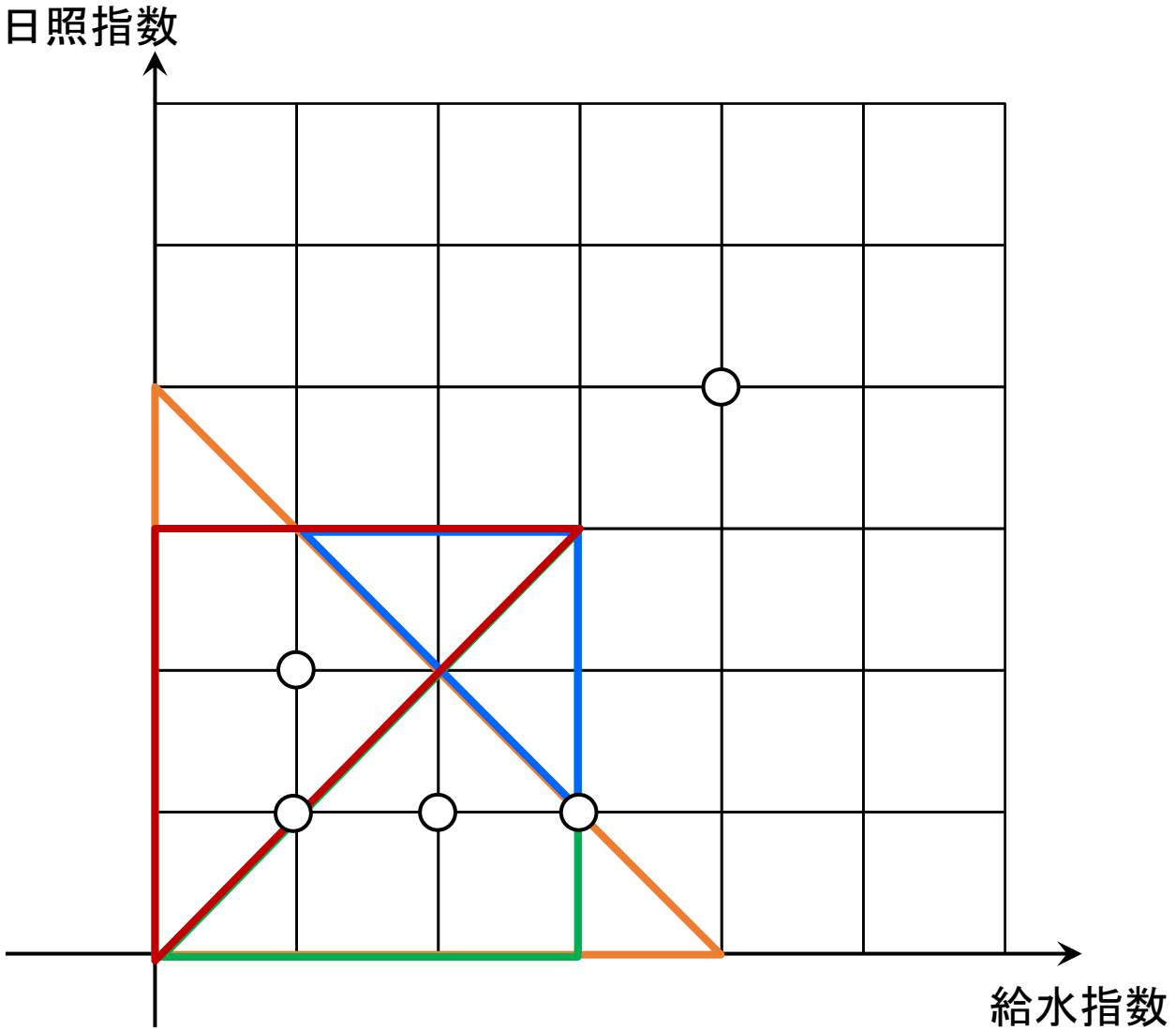
基準値からのずれに対する耐性値 ＝ 3



問題１３　植物園の栽培計画

入出力例

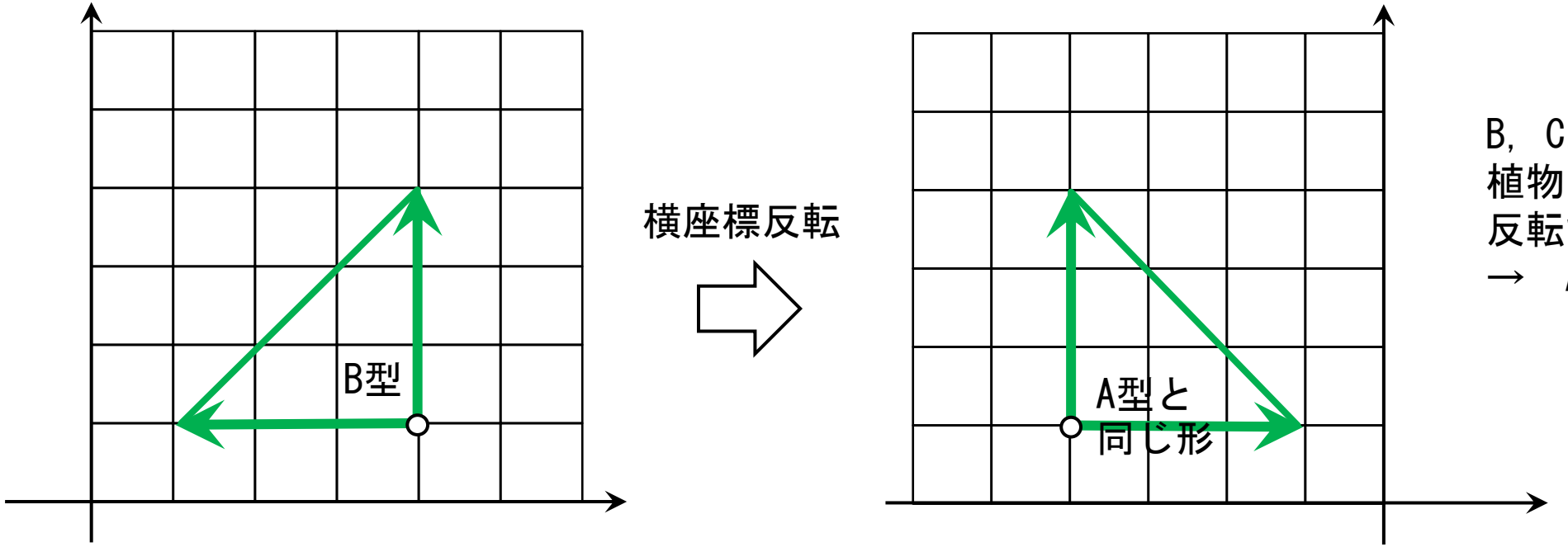
4 5
A 0 0 4
B 3 0 3
C 3 3 2
D 0 3 3
1 1　→　3
2 1　→　2
3 1　→　3
1 2　→　2
4 4　→　0



問題１３　植物園の栽培計画

植物の型

- A型：給水指数と日照指数が、どちらもその植物の基準値以上である。
- B型：給水指数がその植物の基準値以下であり、日照指数がその植物の基準値以上である。
- C型：給水指数と日照指数が、どちらもその植物の基準値以下である。
- D型：給水指数がその植物の基準値以上であり、日照指数がその植物の基準値以下である。



B，C，D型はそれぞれの型の植物だけでまとめて座標を反転する
→　A型として解ける

問題１３　植物園の栽培計画

A型の処理の仕方を考える

植物の給水指数の基準値をa_i, 日照指数の基準値をb_i, 基準値からのずれに対する耐性値をc_iとする
栽培計画の給水指数をu_j, 日照指数をv_jとする

① 基準値との大小関係

給水指数が基準値以上である → $u_j \geq a_i$
日照指数が基準値以上である → $v_j \geq b_i$

② 基準値からのずれ（耐性値）の条件

基準値からのずれがその植物の耐性値以下である

$\rightarrow \quad |u_j - a_i| + |v_j - b_i| \leq c_i$

$\Leftrightarrow \quad (u_j - a_i) + (v_j - b_i) \leq c_i$

$\Leftrightarrow \quad u_j + v_j \leq a_i + b_i + c_i$

①を満たす植物について
$u_j - a_i \geq 0, \quad v_j - b_i \geq 0$

問題１３　植物園の栽培計画

例としてA型の植物３個を考える

植物

給水指数 基準値　日照指数 基準値　基準値からのずれに対する耐性値

植物１：$(a_1, b_1, c_1) = (0, 0, 4)$

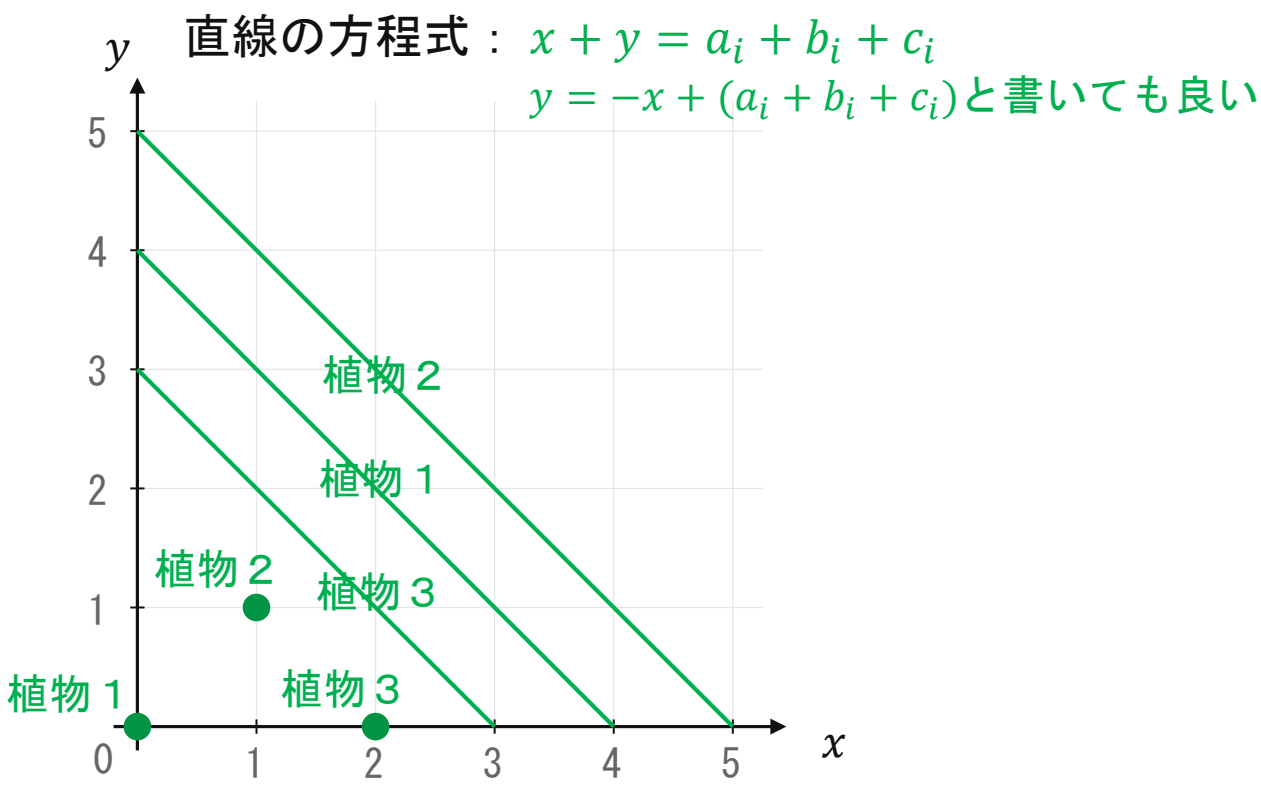
$a_1 + b_1 + c_1 = 4$

植物２：$(a_2, b_2, c_2) = (1, 1, 3)$

$a_2 + b_2 + c_2 = 5$

植物３：$(a_3, b_3, c_3) = (2, 0, 1)$

$a_3 + b_3 + c_3 = 3$



問題１３　植物園の栽培計画

例として栽培計画３個を考える

計画

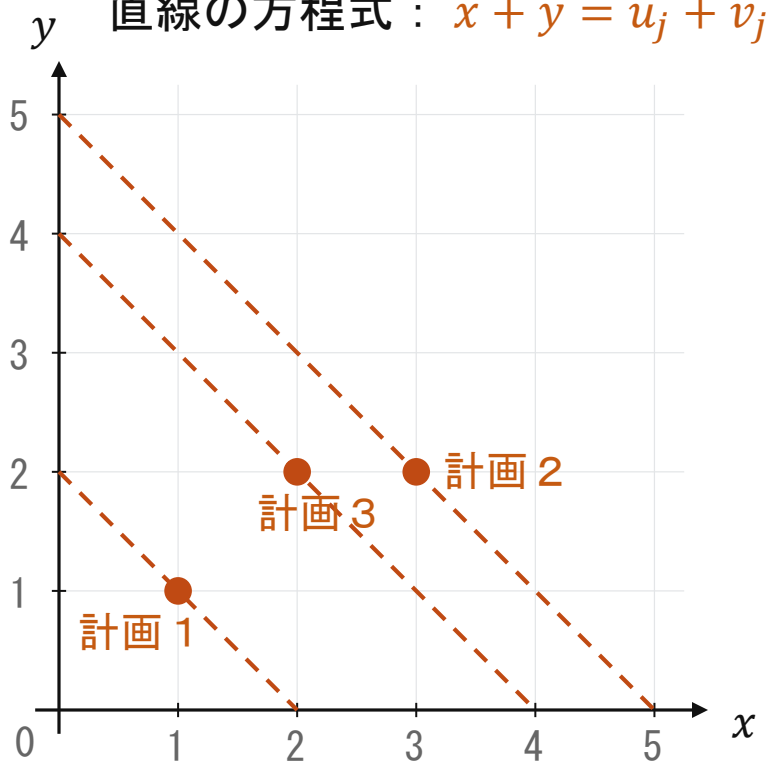
給水指数　日照指数

計画１：$(u_1, v_1) = (1, 1)$
$u_1 + v_1 = 2$

計画２：$(u_2, v_2) = (3, 2)$
$u_2 + v_2 = 5$

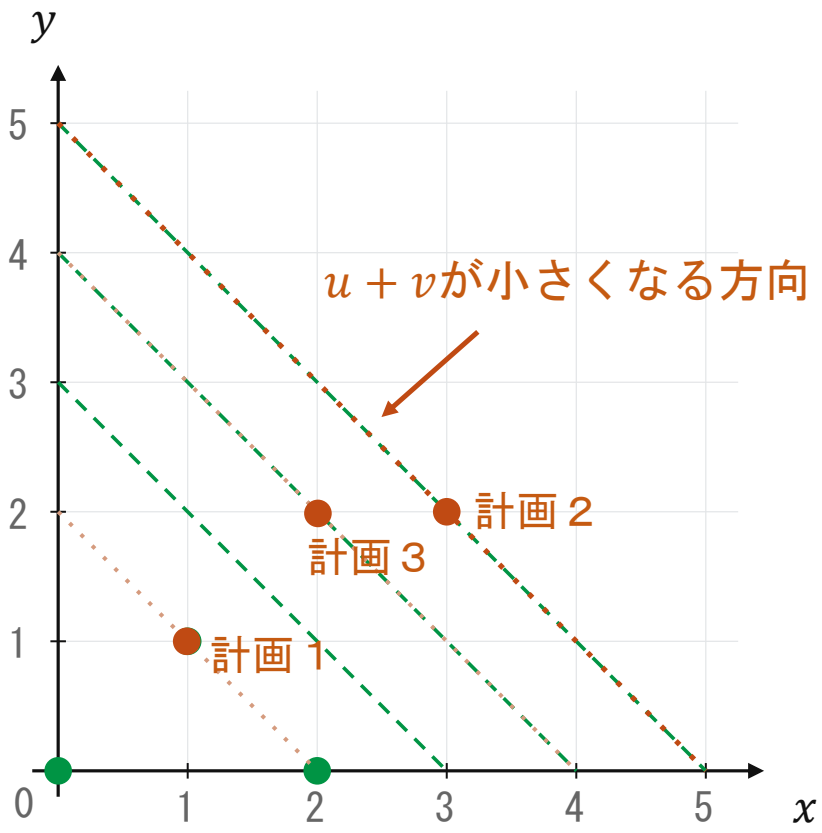
計画２：$(u_3, v_3) = (2, 2)$
$u_3 + v_3 = 4$

直線の方程式：$x + y = u_j + v_j$



問題１３　植物園の栽培計画

制約　$u_j + v_j \leq a_i + b_i + c_i$



$u+v$が小さくなると

$u_j + v_j \leq a_i + b_i + c_i$ を満たす植物が単調に増える

$u+v$の降順で計画を処理する

この例では計画２，計画３，計画１の順
植物も$a_i + b_i + c_i$の降順に並べておく

各計画を満たす植物の数を調べる前に、
$u_j + v_j \leq a_i + b_i + c_i$を満たす植物を
チェックリストに追加し、
その中で$u_j \geq a_i$と$v_j \geq b_i$を満たす植物の数を求める

問題１３　植物園の栽培計画

降順処理の進み方

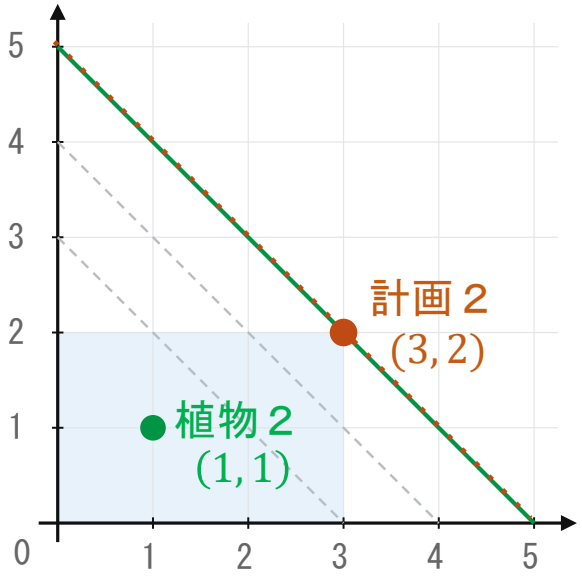
薄い青：　$u_j \geq a_i$ ，$v_j \geq b_i$ を満たす範囲

1. $u_2 + v_2 = 5$の時点

$a_2 + b_2 + c_2 = 5$は

$u_2 + v_2 \leq a_2 + b_2 + c_2$をみたす

→　植物２をチェックリストに追加



植物２は $3 \geq 1$ ，$2 \geq 1$

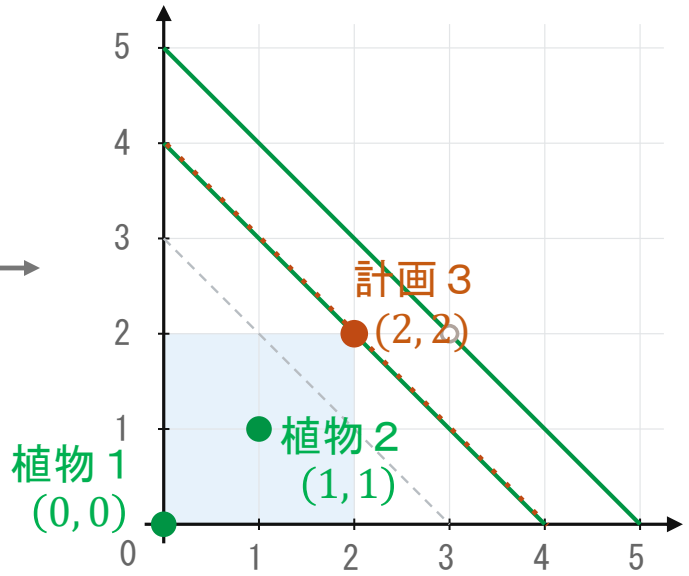
答え：1

2. $u_3 + v_3 = 4$の時点

$a_1 + b_1 + c_1 = 4$は

$u_3 + v_3 \leq a_1 + b_1 + c_1$をみたす

→　植物１をチェックリストに追加



植物１は $2 \geq 0$ ，$2 \geq 0$

植物２は $2 \geq 1$ ，$2 \geq 1$

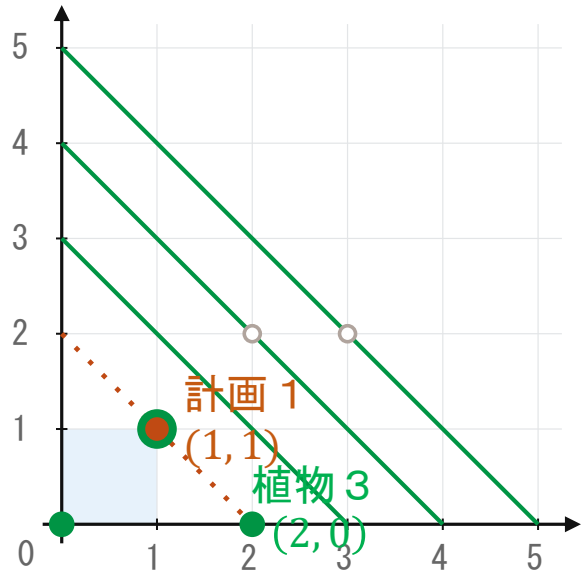
答え：2

3. $u_1 + v_1 = 2$の時点

$a_3 + b_3 + c_3 = 3$は

$u_3 + v_3 \leq a_1 + b_1 + c_1$をみたす

→　植物３をチェックリストに追加



植物１は $1 \geq 0$ ，$1 \geq 0$

植物２は $1 \geq 1$ ，$1 \geq 1$

植物３は $1 \leq 2$ ，$1 \geq 0$

答え：2

問題１３　植物園の栽培計画

候補の植物が、$u_j \geq a_i$, $v_j \geq b_i$ をみたすかを効率よく求めたい
→　２次元Fenwick Tree（別名 Binary Indexed Tree, BIT)を使う

- $add(a,b)$：点(a,b)をFenwick Treeに追加する　$\mathrm{O}(\log^2 N)$
- $sum(x,y)$：$a_i \leq x$かつ$b_i \leq y$である点の個数を求める $\mathrm{O}(\log^2 N)$

座標の値は10^9までで、反転後は負になることもある
→　座標圧縮を使う

問題１３　植物園の栽培計画

座標圧縮

座標を小さい順の通し番号に置き換える

植物１：(40, 15)
植物２：(150, 80)
植物３：(900, 15)

⇩

植物１：(1, 1)
植物２：(2, 2)
植物３：(3, 1)

横座標：

元の値	40	150	900
圧縮後	1	2	3

縦座標：

元の値	15	80
圧縮後	1	2

問題１３　植物園の栽培計画

Fenwick Treeへの問い合わせは、必ずしも存在する番号が使われるとは限らない

栽培計画の座標 $(u_j, v_j) = (200, 90)$ の問い合わせ

$a_i \leq 200$ の範囲

元の値	40	150	900
圧縮後	1	2	3

$b_i \leq 90$ の範囲

元の値	15	80
圧縮後	1	2

$a_i \leq 200$ の範囲にあるのは、40, 150 の２種類なので、問い合わせに使うのは 2
$b_i \leq 90$ の範囲にあるのは、15, 80 の２種類なので、問い合わせに使うのは 2

Fenwick Treeには $(2, 2)$ で問い合わせる

例えば、$u_j = 20$ なら、該当する基準値がないので横の添字は0とする
このような圧縮後の番号は upper_bound等、2分探索で求められる

問題１３　植物園の栽培計画

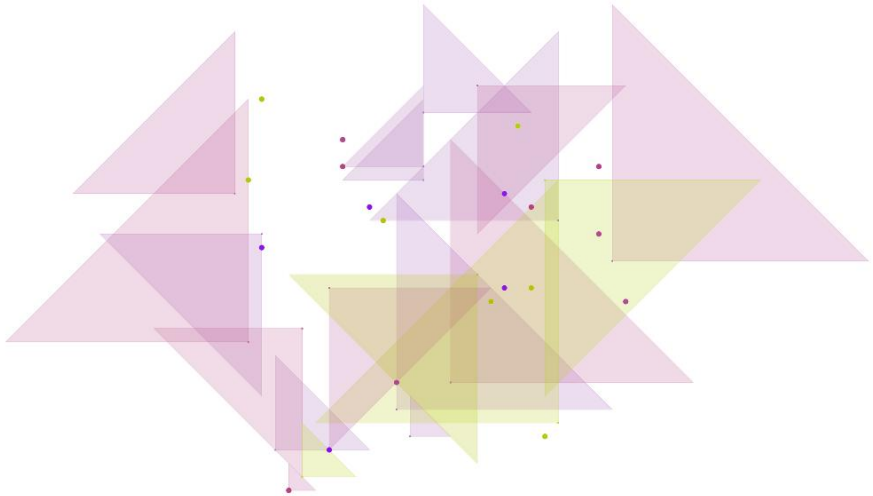
計算量

- 植物・計画のソート　$O(N \log N + Q \log Q)$
- 座標圧縮・BITの準備　$O(N \log^2 N)$
- 植物の追加（全体でN回）　$O(N \log^2 N)$
- 計画の問い合わせ（全体で$4Q$回）　$O(Q \log^2 N)$

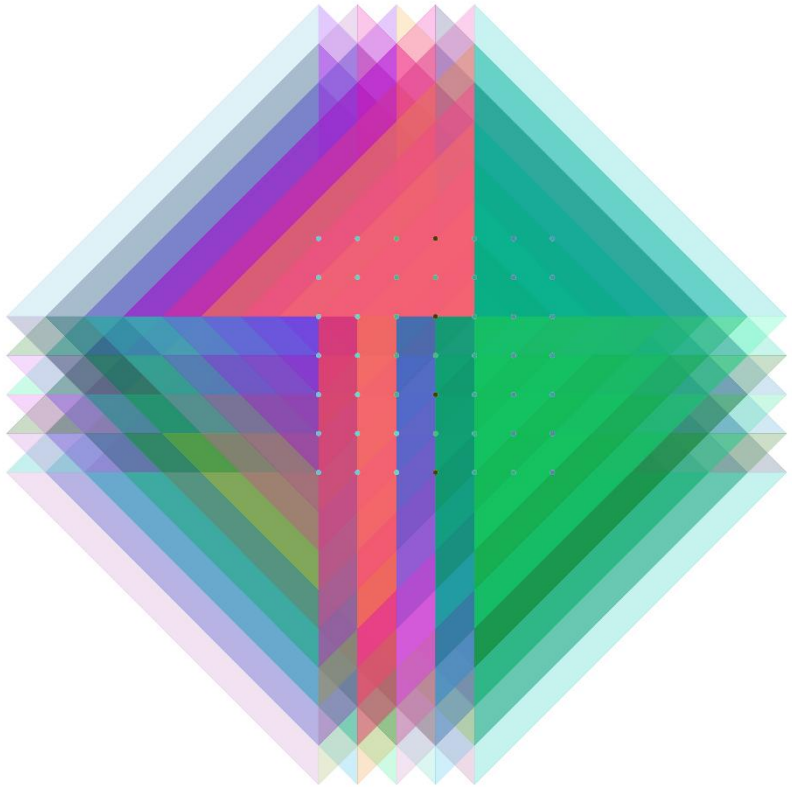
問題１３　植物園の栽培計画

ジャッジデータ例

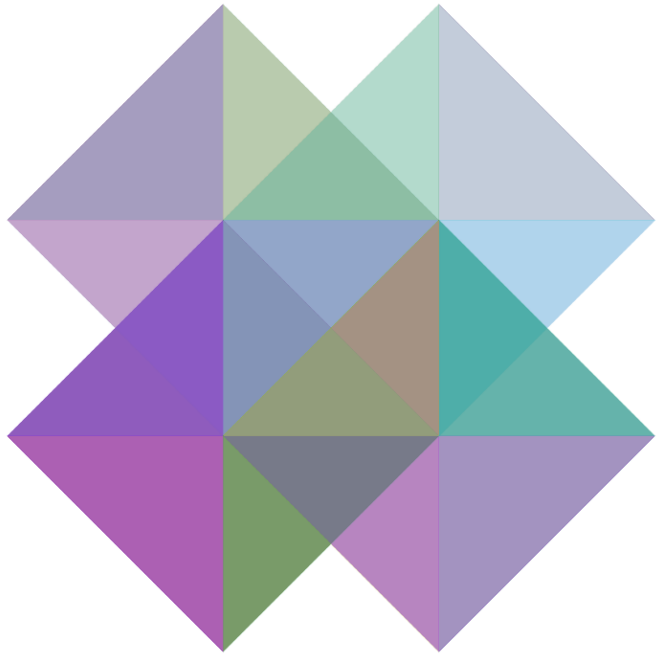
01_small_02.in



01_small_03.in



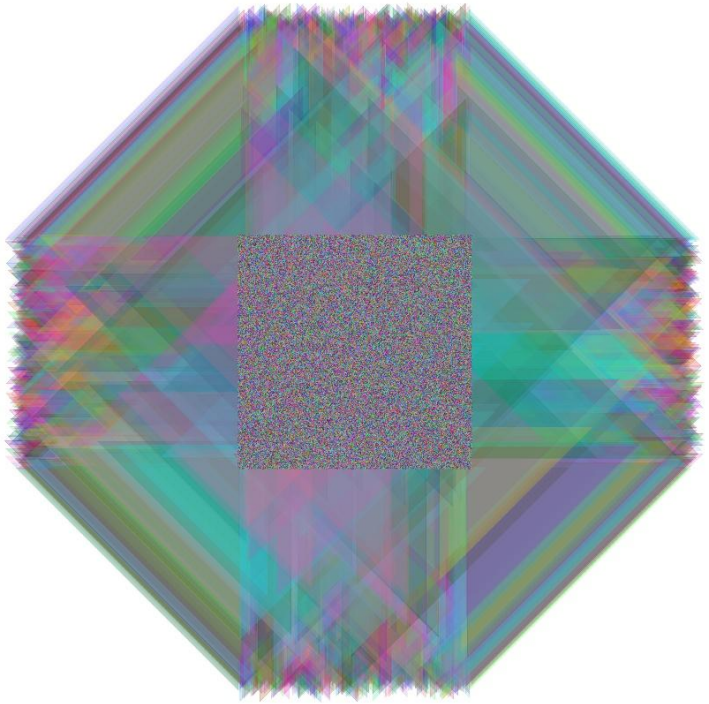
03_general_02.in



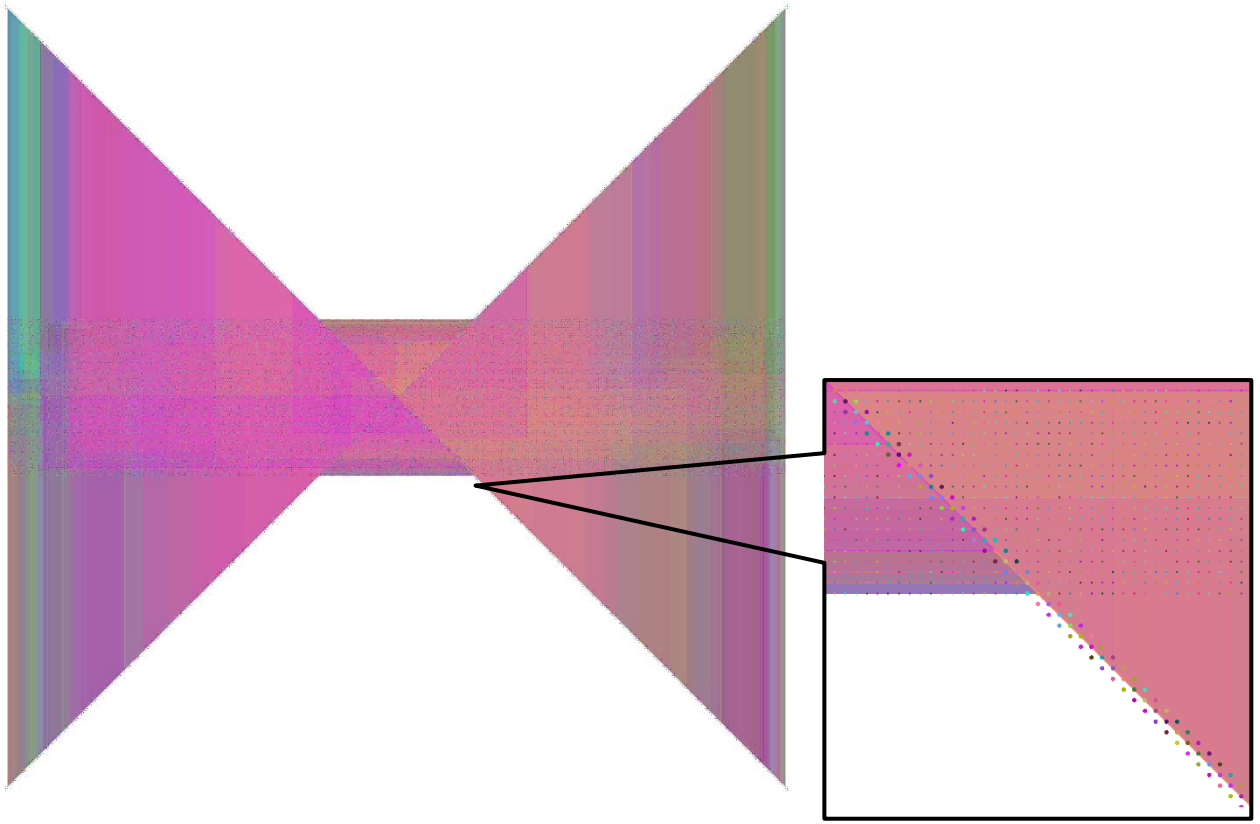
問題１３　植物園の栽培計画

ジャッジデータ例

05_large_01.in



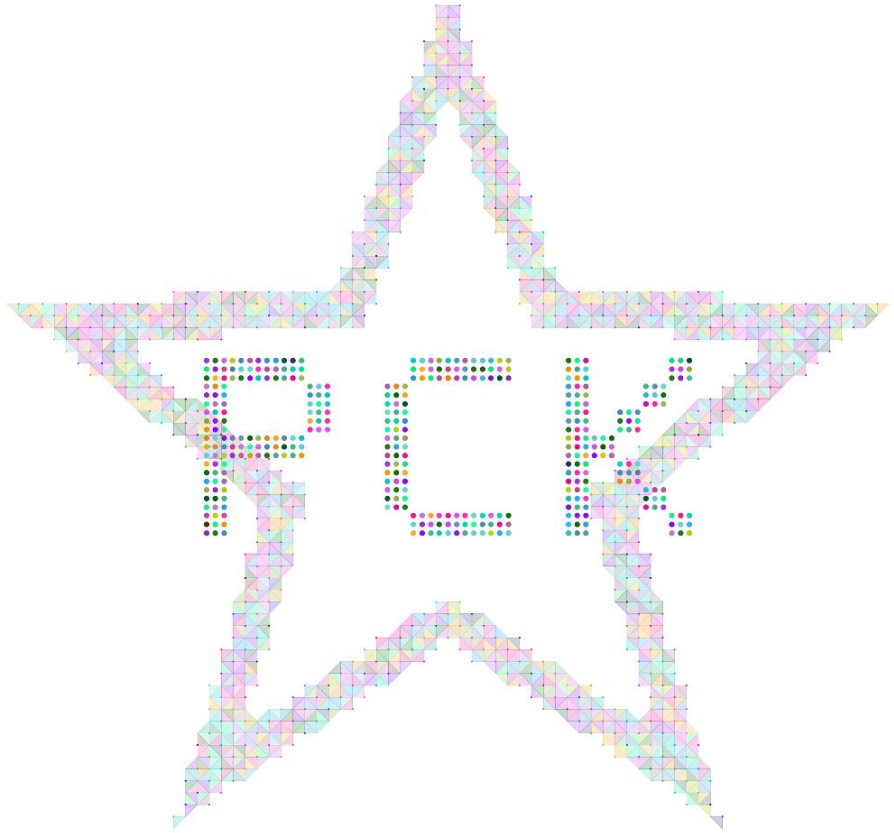
08_corner_critical_01.in



問題１３　植物園の栽培計画

ジャッジデータ例

10_beautiful_01.in



10_beautiful_02.in

