










に・ぜろ・に・さん パソコン甲子園2023

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



問題セット

-  1 穴あきワッフル
-  2 サバを読む
-  3 村を整える
-  4 ロールケーキ
-  5 9つの数字
-  6 PCK君のひそかな楽しみ

-  7 順位の予測
-  8 条坊制都市
-  9 連環迷路
-  10 投網
-  11 串団子
-  12 星座を探して
-  13 点の削除

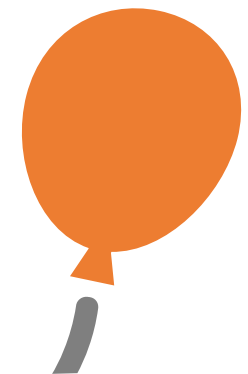
問題セット

#	タイトル	分野	得点	難易度	
				思考	実装
1	穴あきワッフル	基礎	2	☆	☆
2	サバを読む	基礎	3	★	☆
3	村を整える	整列・シミュレーション	4	★☆	★
4	ロールケーキ	貪欲法	5	★★	★☆
5	9つの数字	組み合わせ（全探索）	6	★★	★★☆
6	PCK君のひそかな楽しみ	数え上げ	6	★★★	★★☆
7	順位の予測	シミュレーション	9	★★★★	★★★★
8	条坊制都市	探索	9	★★★★	★★★★
9	連環迷路	計算幾何&グラフ	10	★★★★	★★★★☆
10	投網	データ構造	10	★★★★☆	★★★★☆
11	串だんご	数え上げ（動的計画法）	10	★★★★☆	★★★★
12	星座をさがして	計算幾何・文字列アルゴリズム	13	★★★★★	★★★★★☆
13	点の削除	探索	13	★★★★★☆	★★★★★

問題 1 穴あきワッフル (2点)

正答数: 30チーム

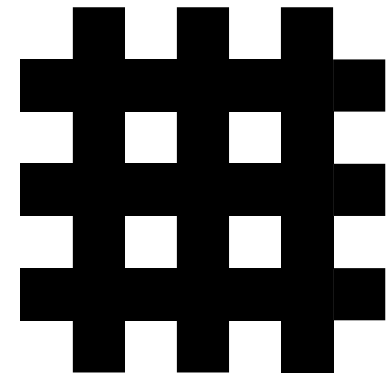
最初の正解: 5分45秒



問題 1 穴あきワッフル

概要

- 穴あきワッフルは、生地が格子状になっている。生地が縦に N 本、横に N 本あるワッフルを、レベル N の穴あきワッフルと呼ぶ。
- 線の太さは1 cmであり、同方向の線は1 cm間隔で並んでいる。
- 入力として穴あきワッフルのレベル N ($1 \leq N \leq 100$)が与えられる。
- 1 cm \times 1 cmの領域を1文字とし、生地には「#」、穴には「.」を使い、レベル N の穴あきワッフルを出力する。



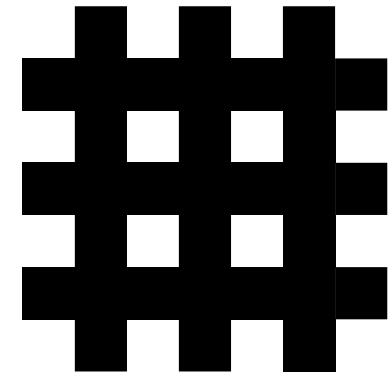
問題 1 穴あきワッフル

解法

- レベル N の穴あきワッフルの線の長さ M は、 $2N + 1$ である。
- 上から i 番目、左から j 番目 ($1 \leq i, j \leq M$)の文字は、 i と j の両方が奇数の場合は'.' (白色) である。それ以外の場合は'#' (黒色) である。

解答例 (C++)

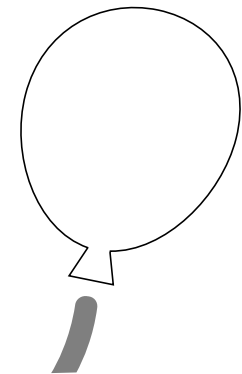
```
int N;
cin >> N;
int M = (2*N)+1;
for (int i = 1; i <= M; i++) {
    for (int j = 1; j <= M; j++) {
        if ((i%2 == 1) && (j%2 == 1)) {
            cout << '.';
        } else {
            cout << '#';
        }
    }
    cout << endl;
}
```



問題 2 サバを読む (3点)

正答数: 29チーム

最初の正解: 10分31秒



問題 2 サバを読む

概要

- 数値 X ($1 \leq X \leq 1,000,000,000$) が10進数で与えられる。
- 数値 X を10進表記で読み取ったときの値 $>$ 数値 X を k 進表記で読み取ったときの値であればYes, そうでなければNoを出力。

テストケース

30029

No

8264

Yes

10進数値を k 進数で読み取るときの値

4	2	1	7	6	2	1
*	*	*	*	*	*	*
k^6	k^5	k^4	k^3	k^2	k^1	k^0
$4k^6$	$+2k^5$	$+k^4$	$+7k^3$	$+6k^2$	$+2k^1$	$+k^0$

問題 2 サバを読む

解法

- 10進数の値を k 進数で読み取るとき、 k をいくつにすべきか
 - $k > 10$ であれば、2桁以上のとき読み取る値が大きくなる。
 - $k < 10$ であれば、2桁以上のとき読み取る値が小さくなる。
 - 1桁の数値は値が変わらない。
- $k < 10$ の k 進数に変換できるか？
 - 10進数の値を示す数字に「9」が含まれていなければ、9進数として読み取ることができる。



読み取った数値が2桁以上で、9が含まれていなければYes.

問題 2 サバを読む

解答例

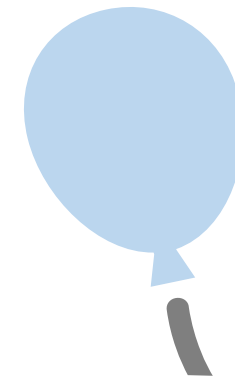
```
#include <stdio.h>
#include <string.h>
int main(void)
{
    char str[10];
    scanf("%s", str);
    printf("%s¥n", (strlen(str)==1 || strchr(str, (int)'9')) ? "No" : "Yes");
    return 0;
}
```

長さ
文字にマッチすると
そのアドレスが返る→true
マッチしなければNULL→false

問題 3 村を整える (4点)

正答数: 29チーム

最初の正解: 17分52秒



問題 3 村を整える

概要

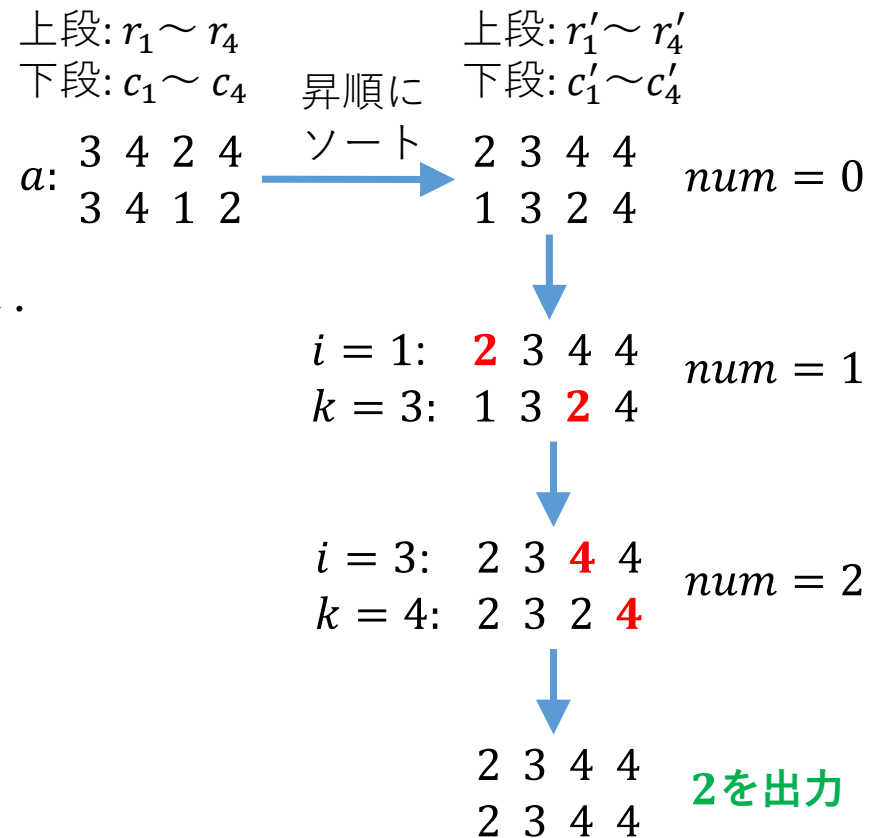
- $N(1 \leq N \leq 1,000)$ 区画の畑がある. i 番目の畑に植えるべき作物の数 r_i ($1 \leq r_i \leq 1,000$)が与えられる. この数だけ植えると, 村が整うと言う.
- すべての畑に適当な数の作物を植えてしまった. i 番目の畑に c_i ($1 \leq c_i \leq 1,000$)本の作物を植えてしまったので, r_i 本にして村を整えたい.
- 畑の作物の数を変更するときは以下のやり方にしたがわなければならない.
畑 i の作物の数を変更したいとする. 畑 i の作物の数 (x 本) よりも作物の数が多い畑 k を選ぶ. 畑 k の作物の数が y 本するとき, 畑 i に $y - x$ 本の作物を加えて畑 i の作物の数を y 本にする.
- このやり方を何度か行って村が整うようにできるならその最小回数を, できないなら -1 を出力する.

問題 3 村を整える

解法 1 (シミュレーション)

```

a ← (r1, c1) ... (rN, cN)
num ← 0
a ← sort(a) // a = (r'1, c'1) ... (r'N, c'N) と置く.
for i ← 1 to N:
  if r'i < c'i:
    -1 を出力して終了
  if r'i ≠ c'i:
    if (i ≤ k ≤ N と r'i = c'k を満たす k がある):
      num ← num + 1
      c'i ← c'k // 解法には不要
    else:
      -1 を出力して終了
num を出力して終了
  
```



問題 3 村を整える

解法 1 の正しさ

- -1が出力される時、村を整えられないことが言えればよい。
- 以下の条件のいずれかが成り立つと村を整えられないことがわかる。
 - $r'_i < c'_i$ を満たす*i*がある。
 - $r'_i = c'_k$ を満たす*k*がないような*i*がある。
- 解法1で、ある*i*で-1が出力されるのは、以下のいずれかの場合。
 1. $r'_i < c'_i$ のとき、
 2. $r'_i = c'_k$ と $i \leq k \leq N$ を満たす*k*がないとき。
- *a*は昇順にソート済みで、 $1 \leq j \leq i - 1$ を満たす*j*について $c'_j \leq r'_j$ が成り立つので、上の2の場合には c'_1 から c'_{i-1} にも r'_i と等しいものがないことがわかる。

以上より、解法1は正しい。

問題 3 村を整える

解答例 計算量: $O(N^2)$

```
void solve(vector<pair<int,int>> a) {
    int num=0;
    sort(a.begin(), a.end());
    for (int i = 0; i < N; i++ ) {
        if (a[i].first < a[i].second){ cout << -1 << endl; return; } //  $r_i < c_i$ 
        if (a[i].first != a[i].second) { //  $r_i \neq c_i$ 
            int j;
            for (j=i; j<N; j++)
                if (a[i].first == a[j].second) break; //  $r_i = c_j$ 
            if (j>=N) { cout << -1 << endl; return; }
            num++;
        }
    }
    cout << num;
}
```

問題 3 村を整える

解法 2 (条件のチェック)

- 「解法1の正しさ」で述べた条件が成り立つと村を整えられない。
- この条件の否定と同値な条件Aは以下のとおり。
すべての i について、 $r_i \geq c_i$ 、かつ、 $r_i = c_k$ を満たす k がある。
- 条件Aが成り立つ。 \Leftrightarrow 以下の条件Bが $i = 1$ で成り立つ。
 $i \leq j$ を満たすすべての j について、 $r'_j \geq c'_j$ 、かつ、 $r'_j = c'_k$ と $i \leq k \leq N$ を満たす k がある。
- このとき、解法1のforループの各繰り返しで条件Bが成り立ち、その結果、村が整えられることがわかる。
- 以上より、入力 $r_1, \dots, r_N, c_1, \dots, c_N$ が条件Aを満たすかチェックするだけで、村が整えられるかがわかる。

問題 3 村を整える

解答例 計算量: バケツを使うと $O(N)$, setを使うと $O(N \log N)$

```
void solve(vector<int> r, vector<int> c) {
    int num=0;
    vector<bool> b(1000+1);
    for (int i = 0; i < N; i++) { b[c[i]]=true; } // バケツを使う
    for (int i = 0; i < N; i++) {
        if (!b[r[i]] || r[i] < c[i]) { //  $r_i = c_k$ を満たす $k$ がないか、 $r_i < c_i$ なら整えられない
            cout << -1 << endl; return;
        }
        if (r[i] != c[i]) num++;
    }
    cout << num;
}
```

問題 4 ロールケーキ (5点)

正答数: 29チーム

最初の正解: 17分58秒



問題4 ロールケーキ

概要

- 文字 o , x を含む文字列 S ($2 \leq \text{len}(S) \leq 100,000$)が与えられる.
- S を下記の条件で文字列を切り分けてピースを作り, ピースの数を最大にする
 - ピースは2文字以上
 - それぞれのピースは同じ長さ
 - それぞれのピースに含まれる o の数は同じ

問題4 ロールケーキ

解法

- 2文字のピースを考える
 - 3文字のピースを作った場合、2文字のピースを作ったときと同じかそれ以下の数になる
 - 4文字以上のときも同様
- oの数が0～2のときについて、それぞれ先頭から条件に合う文字列を切り分ける。
 - 先頭から切り出すと最良になることは、帰納法で証明できる（スケジューリング問題）。
- oの数が0～2のそれぞれについて切り分け、ピースが多いものが答え

ピースの文字数の考察

3文字で切り分けたとき

oが3つ	oが1 or 2つ	oがなし
ooo	OOX XXO OXO XOX XOO OXX	XXX
oo を取り出せる	OX または XO を取り出せる	XX を取り出せる



2文字で切り分けることが最良

問題 4 ロールケーキ

解答例

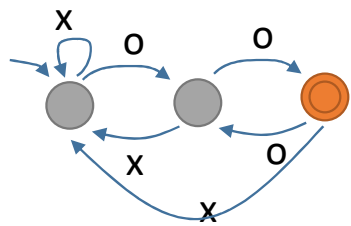
```
#include <stdio.h>
#define MAX(a, b) ((a) > (b) ? (a) : (b))
int main(void) {
    char in[100001];
    int sum_oo=0, sum_xx=0, sum_ox=0, i;
    scanf("%s", in);
    for(i=0; in[i]!='\0'; ++i)
        if(in[i]=='x' && in[i+1]=='x') {++sum_oo; ++i;}
    for(i=0; in[i]!='\0'; ++i)
        if(in[i]=='o' && in[i+1]=='o') {++sum_xx; ++i;}
    for(i=0; in[i]!='\0' ; ++i)
        if(in[i]=='o' && in[i+1]=='x'
           || in[i]=='x' && in[i+1]=='o') {++sum_ox; ++i;}
    printf("%d\n", MAX(MAX(sum_oo, sum_xx), sum_ox));
    return 0;
}
```

問題4 ロールケーキ

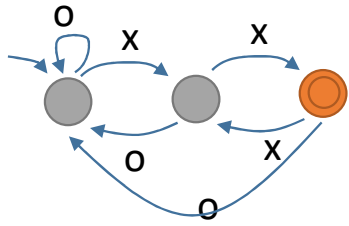
解答例

```
int fsm_oo[3][2]={{1,0},{2,0},{1,0}};
int fsm_xx[3][2]={{0,1},{0,2},{0,1}};
int fsm_ox[4][2]={{1,2},{1,3},{3,2},{1,2}};
int st_oo=0, st_xx=0, st_ox=0;
int cn_oo=0, cn_xx=0, cn_ox=0;
char c; int in;
```

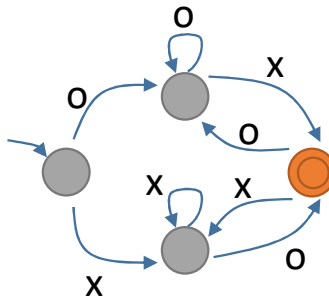
oo, xx, ox/xo を受理する有限オートマトンを作る



ooを受理



xxを受理



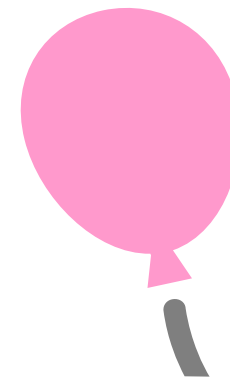
oxとxoを受理

```
while(1){
    scanf("%c",&c);
    if(c=='¥n') break;
    in = c=='o' ? 1 : 0;
    st_oo=fsm_oo[st_oo][in];
    st_xx=fsm_xx[st_xx][in];
    st_ox=fsm_ox[st_ox][in];
    if(st_oo==2) ++cn_oo;
    if(st_xx==2) ++cn_xx;
    if(st_ox==3) ++cn_ox;
}
printf("%d¥n",MAX(MAX(cn_oo,cn_xx),
                    cn_ox));
```

問題 5 9つの数字（6点）

正答数: 19チーム

最初の正解: 36分34秒



問題 5 9つの数字

7	8	9
4	5	6
1	2	3

概要

- 家の入口に特殊な電子錠がついている。
 - この電子錠には1から9までの数字が、右の図のように縦3行、横3列に配置されている。数字は操作開始前に自由に並び替えることができる。
 - この電子錠には数字を指し示すカーソルが1つあり、カーソルは常にどれか1つの数字を指し示している。入力開始時には、カーソルが右下の数字を指している。
- 電子錠を開錠するための暗証番号として、1から9までの数字からなる数字列が設定されている。以下の操作で、暗証番号を左から順番に入力していくことで解錠することができる：
 - 隣接する上下左右のどれかの数字へカーソルを移動する。 (1回の操作)
 - カーソルが指し示す数字を入力する。 (1回の操作)
- 暗証番号を表す整数 N ($1 \leq N < 10^{10^5}$)が与えられたとき、暗証番号を入力するために必要な操作回数の最小値を求めよ。

問題 5 9つの数字

解法

- 電子錠において、 (a, b) に配置した数字から (c, d) に配置した数字へのカーソル移動に要する操作回数は、数字間のマンハッタン距離。

$$|a - c| + |b - d|$$

- 例えば、右図の配置方法において、 $(0, 1)$ に配置された7から $(2, 0)$ に配置された6への移動に要する操作回数は、 $|0 - 2| + |1 - 0| = 3$
- 9つの数字の配置方法は全部で $9! = 362,880$ 通りあり、最小の操作回数を求めるためには、全ての配置方法について操作回数を調べる必要がある。
- 暗証番号は最大で 10^5 桁なので、左から順番に数字間の距離を足していく素朴な解法では、全ての配置方法について調べると**時間制限**

9	7	5
4	8	2
6	1	3

問題 5 9つの数字

解法

- 全探索する前に、与えられた暗証番号を表す数字の列について、数字 i から数字 j ($1 \leq i, j \leq 9$)へ移動する回数を二次元配列に記録しておく。
 - 例えば、暗証番号が74174747の場合、 $c[1][7]=1, c[4][1]=1, c[4][7]=2, c[7][4]=3$ と記録。
 - これら4つ以外の回数は0
- その回数に数字間のマンハッタン距離を掛けた積を全ての数字の組み合わせについて計算し、操作回数に加算する。
- カーソルを右下から最初の数字へ移動するための操作回数を別に加算する。
- 暗証番号の桁数分だけ数字の入力操作が必要。
- 数字の種類を m 、暗証番号の桁数を n で表すと、計算量は $O(m! m^2 + n)$

問題 5 9つの数字

解答例 (C++)

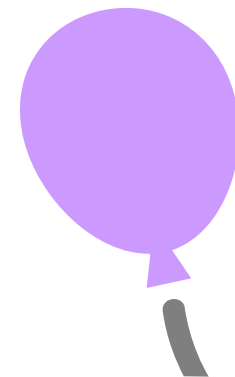
```
string N; cin >> N;
vector<pair<int,int>> p;
vector<vector<int>> c(9+1, vector<int>(9+1, 0));
for (size_t i = 0; i < N.length()-1; i++)
    c[N[i]-'0'][N[i+1]-'0']++;
for (int i = 0; i < 3; i++)
    for (int j = 0; j < 3; j++)
        p.push_back(make_pair(i, j));
int ans = 99999999;
do {
    int ops = N.length();
    ops += dist(p[N[0]-'0'-1],
                make_pair(3-1, 3-1));
    for (int i = 1; i <= 9; i++)
        for (int j = 1; j <= 9; j++)
            ops += c[i][j] * dist(p[i-1], p[j-1]);
    ans = min(ans, ops);
} while (next_permutation(p.begin(), p.end()));
```

```
int dist(pair<int,int> a,
         pair<int,int> b) {
    return abs(a.first-b.first)
        + abs(a.second-b.second);
}
```

問題 6 PCK君のひそかな楽しみ (6点)

正答数: 12チーム

最初の正解: 33分39秒



問題 6 PCK君のひそかな楽しみ

概要

- 長さ N ($3 \leq N \leq 2 \times 10^5$) の文字列 S と PCK君のラッキーナンバー L ($1 \leq L \leq 10^{15}$) が与えられる。
- S の中に現れる, 以下の条件を満たす部分文字列の個数を求める。
 - P,C,Kがこの順番に L 個以上現れる (ただし、P,C,Kは連続していなくても良い)。

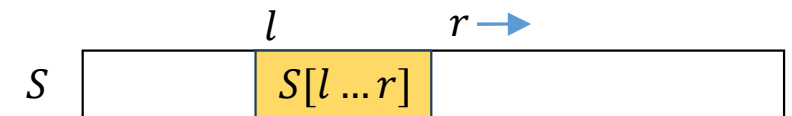
テストケース

$L = 2$ PCCKA
PCCKA PCCKA
 } }
 } }

問題 6 PCK君のひそかな楽しみ

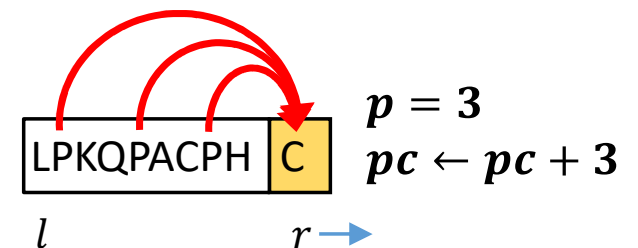
解法 (尺取り法)

- 求める部分文字列の個数を num とする.
- 文字列 S の l から $r - 1$ までの範囲の文字列 $S[l \dots r]$ の中の PCK の個数を調べる ($0 \leq l < r \leq N$).
- $S[l \dots r]$ の中の P, C, K, PC, PCK, CK の個数を変数 p, c, k, pc, pck, ck にそれぞれ保存する.
- 部分文字列 $S[l \dots r]$ の中の PCK が L 個未満なら, 部分文字列の右端を伸ばす (r を増やす). 右端に追加された文字 x に応じて, 右の表のように変数を更新する.



x	p	c	k	pc	pck	ck
P	+1					
C		+1		+ p		
K			+1		+ pc	+ c

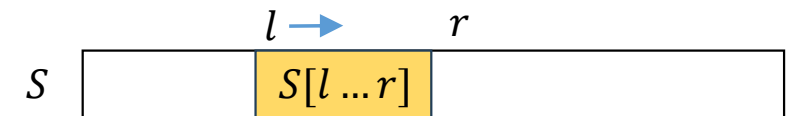
追加された x が C のとき



問題 6 PCK君のひそかな楽しみ

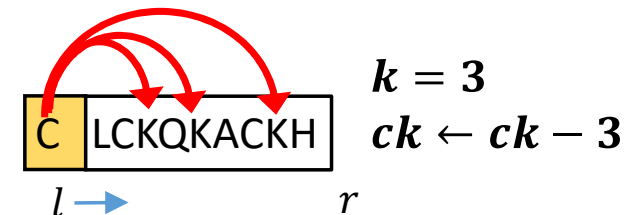
解法 (続き)

- 部分文字列 $S[l \dots r]$ 中のPCKが L 個以上になったら、 num に $N - r + 1$ を加え、部分文字列の左端を縮める (l を増やす)。左端から削除された文字 y に応じて、右の表のように変数を更新する。
- 左端を削除しても $S[l \dots r]$ 中のPCKが L 個以上なら、 num に $N - r + 1$ を加え、再び左端を縮める。
- 左端を削除したとき $S[l \dots r]$ 中のPCKが L 個未満になったら、右端を伸ばす。



y	p	c	k	pc	pck	ck
P	-1			$-c$	$-ck$	
C		-1				$-k$
K			-1			

削除された y が C のとき



問題 6 PCK君のひそかな楽しみ

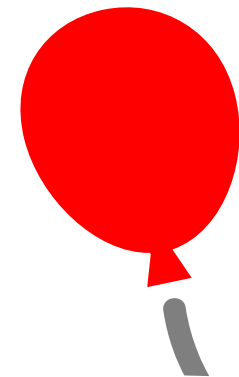
解答例 計算量: $O(N)$

```
void solve(string S) {
    long long int num, p, c, k, pc, pck, ck; num = p = c = k = pc = pck = ck = 0;
    for (int l = 0, r = 0; l < N; l++) {
        while (r < N && pck < L) {
            if (S[r] == 'P') { p += 1; }
            if (S[r] == 'C') { c += 1; pc += p; }
            if (S[r] == 'K') { k += 1; ck += c; pck += pc; }
            r++;
        }
        if (pck >= L) { num += N - r + 1; }
        if (S[l] == 'P') { p -= 1; pc -= c; pck -= ck; }
        if (S[l] == 'C') { c -= 1; ck -= k; }
        if (S[l] == 'K') { k -= 1; }
    }
    cout << num;
}
```


問題 7 順位の予測 (9点)

正答数: 2チーム

最初の正解: 2時間49分53秒



問題 7 順位の予測

概要

- M チームが参加し、 N 問出題されるプログラミングコンテストの順位を予測せよ。参加チームは以下の基準で順位付けされる：
 - 正答した問題の得点の合計が大きい方が良い成績となる
 - 得点の合計が等しい場合は、その得点の合計に到達した時刻が早い方が良い成績となる
- このコンテストでは、特殊なスコアボードを採用しており、以下の情報が開示される。
 - 問題 i の得点は S_i である
 - 問題 i に正答したチームの数は K_i である
 - 問題 i を j 番目に正答したチームは $P_{i,j}$ である
- コンテスト終了時点でのスコアボードの情報から、各チームについて、取りうる順位の最小値と最大値を求めよ。

問題 7 順位の予測

制約

- $1 \leq N \leq 200,000$
- $1 \leq M \leq 300$
- $K_1 + K_2 + \dots + K_N \leq 200,000$
- 各チームがいずれかの問題を正答した時刻は全て異なる

問題 7 順位の予測

解法

- チーム i の総得点を $score_i$ とする。
- チーム i の順位は「 $score_i < score_j$ を満たすチーム j のチーム数」+「 $score_i = score_j$ を満たすチーム j のうち、チーム i よりも早く全ての問題を解いたチーム数」+1。
- 前者は変えようがないため、後者を最小化or最大化することを考える。
- まず、スコアの順にチームに仮のランクを付与する。このとき、スコアが唯一のチームとスコアが0であるチームの順位が確定する。
- $score_i$ が等しいチームの **グループ** でランクを更新する。

Rank	Score
1	100
2	90
2	90
2	90
5	85
6	80
6	80
8	0
8	0
8	0

確定

確定

問題 7 順位の予測

最小値（最も良い順位）

- チーム i が解ける全ての問題を解いた時点で $score_i$ に達していないチームの数を最大化すればよい。
 - チーム i が解いていない問題を解いたチームと、チーム i が解いていた問題でチーム i よりも遅く解いたチームの和集合のサイズとなる（これを $behind_{sum}$ とする）。

チーム数 $m = 5$

p_1 t_1 t_2

p_2 t_3

t_1, t_2, t_3 には勝てる

t_5 に注目

p_3 t_4 t_5 t_3 t_1

t_4 には勝てない

p_4 t_2 t_4 t_5

$$5 - 3 = 2$$

問題 7 順位の予測

最大値 (最も悪い順位)

- チーム i が解けるすべての問題を解き切る前にできるだけ多くの参加者が解き切るケースを考える。
- チーム i が j 番目の問題を x_j 番目に解いたとき、 $m - \lceil (k_j - x_j) \rceil$ の最小値 (これを $behind_{min}$ とする)

チーム数	$m = 5$	p_1	t_1	t_2					
t_4 に注目		p_2	t_3						
		p_3	t_4	t_5	t_3	t_1	後ろに 3 チーム		
		p_4	t_2	t_4	t_5		後ろに 1 チーム	$5 - 1 = 4$	

チーム i が解いた問題について、後ろに何人いるか? の最小値

問題 7 順位の予測

実装の方針

- スコアが同じチームグループごとに、各チームの最小・最大の順位を決定
 - 以下のデータを作る
 - グループに含まれるチームが解いた問題セット
 - 問題セットの各問題について、グループに含まれるチームが解いた順番（関係のないチームを排除）
 - グループ内の各チーム*i*について、順位を決定していく

$$\blacktriangleright \minRank_i = baseRank + m - behind_{sum}$$

$$\blacktriangleright \maxRank_i = baseRank + (m - behind_{min})$$

問題 7 順位の予測

計算量

- $O(M \sum K_i + M^2)$

問題 8 条坊制都市 (9点)

正答数: 7チーム

最初の正解: 1時間5分47秒



問題 8 条坊制都市

概要

- H 行 W 列の行列 A が与えられる。行列の各要素は非負の整数である
- A から i 行目、 j 列目を抜き出した行列を考える。この行列に出現しない最小の非負整数(Minimum EXcluded value = MEX)を出力する
- $i = 1, 2, 3, \dots, H$ 、 $j = 1, 2, 3, \dots, W$ についてこれを行う
- $2 \leq H, W \leq 1500$

問題 8 条坊制都市

テストケース

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}$$

1行目1列目を消すと2が残る -> 0

1行目2列目を消すと1が残る -> 0

2行目1列目を消すと1が残る -> 0

2行目2列目を消すと0が残る -> 1

問題 8 条坊制都市

考察

「 A の MEX 」 \geq 「1行・1列消した後の MEX 」 が成り立つ

- 明らかに増えることはない
- どのような時に消した後の MEX が元々の MEX より小さくなる？

問題 8 条坊制都市

考察

例えば左の行列だと、 A の MEX は3

3行目3列目を消すと2が全部消えて MEX が2になる

1行目2列目を消すと0が全部消えて MEX が0になる

$$A = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \\ 2 & 4 & 2 \end{pmatrix}$$

A の MEX 未満の値 v に対して、 A から v を全て消すような消し方が存在する

⇒

そのような消し方をした時の MEX は v 以下になる

問題 8 条坊制都市

解法

$$A = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \\ 2 & 4 & 2 \end{pmatrix}$$

観察から、各値 v について「 v を全部消すような消し方は存在するか？存在するならどのような消し方か？」を考えることにする。

行・列の消し方は HW 通りあり、考慮すべき v は $O(HW)$ 個あるので、すべての消し方を愚直に調べようとすると $\Omega(H^2W^2)$ かかってしまう。(時間制限に間に合わせることは困難)

実は、ほとんどの消し方は考慮しなくて良い

問題 8 条坊制都市

解法

A のMEX未満の数について、それぞれの座標に存在するかをあらかじめ並べておく

$$A = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \\ 2 & 4 & 2 \end{pmatrix}$$

0 . . . (1, 1) (2, 2)

1 . . . (1, 2) (2, 1)

2 . . . (1, 3) (2, 3) (3, 1) (3, 3)

問題 8 条坊制都市

解法

$$A = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \\ 2 & 4 & 2 \end{pmatrix}$$

0 . . . (1, 1) (2, 2)

1 . . . (1, 2) (2, 1)

2 . . . (1, 3) (2, 3) (3, 1) (3, 3)

「並べた座標のうち1つ目の座標が消えない消し方において v が存在するすべての座標を消すことはできない」が成り立つ(1つ目の座標が消えていないため)

1つ目の座標を消すような消し方は $H + W - 1$ 通りしか無い

問題 8 条坊制都市

解法

$$A = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \\ 2 & 4 & 2 \end{pmatrix}$$

0 . . . (1, 1) (2, 2)

1 . . . (1, 2) (2, 1)

2 . . . (1, 3) (2, 3) (3, 1) (3, 3)

例えば2なら、

1行目を消して列は自由に選んで消す or

3列目を消して行は自由に選んで消す とすると、

またこのような消し方に限り、1つ目の座標(1, 3)が消える。これ以外の消し方は考慮しなくて良い

問題 8 条坊制都市

解法

消し方を $H + W - 1$ 個まで絞ることができた。
さらに消し方の候補を絞ることができる。

$$A = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \\ 2 & 4 & 2 \end{pmatrix} \begin{matrix} (\overline{1, 3}) \underline{(2, 3)} (3, 1) (3, 3) & \text{(1行目を消した時)} \\ (\overline{1, 3}) (\overline{2, 3}) \underline{(3, 1)} (\overline{3, 3}) & \text{(3列目を消した時)} \end{matrix}$$

実際に行(列)を消してみても、残ったうち一つ目の座標の列(行)を消すような方法が候補となる

⇒その消し方ですべて消えるか確かめれば良い

考えるべき消し方は各要素につき高々2通り

問題 8 条坊制都市

解法

- 考慮すべき座標は $O(HW)$ 個
- それぞれの値で消し方を高々2回試す
 - ✓ それぞれの座標は高々2回しか消えない
- うまく実装をすると $O(HW)$ でこの問題を解ける

問題 8 条坊制都市

解法

実装上の注意

1. 元々の MEX が小さい場合
 - 元々の MEX より大きい値の座標を全て除去しても、解はその値では無く、元々の MEX 以下
2. ある行(列)を消すだけで、ある値の座標を全て除去できる場合
 - その行(列)を消すと、列(行)の消し方に関わらず MEX が値以下になる
3. 同じ消し方で複数の値の座標が全て除去できる場合
 - 消せる値の中で最小のみを考慮する

$$\begin{array}{c} \left(\begin{array}{cc|c} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 4 & 2 & 0 \end{array} \right) \end{array}$$

問題 8 条坊制都市

実装の方針

以下の値を計算する

- i 行目を削除することで、 MEX を $rows[i]$ 以下にできる。
- j 列目を削除することで、 MEX を $cols[j]$ 以下にできる。
- i 行目 j 列目を削除することで、 MEX を $poss[i][j]$ 以下にできる。
- $\min(poss[i][j], \min(rows[i], cols[j]))$ が i 行 j 列目を削除したときの MEX

問題 9 連環迷路 (10点)

正答数: 2チーム

最初の正解: 2時間39分42秒



問題 9 連環迷路

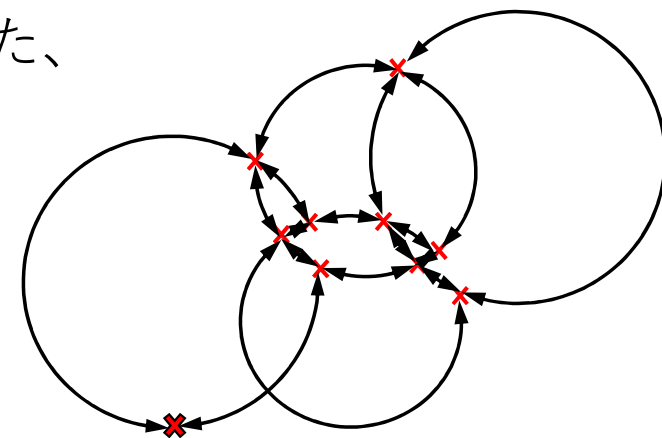
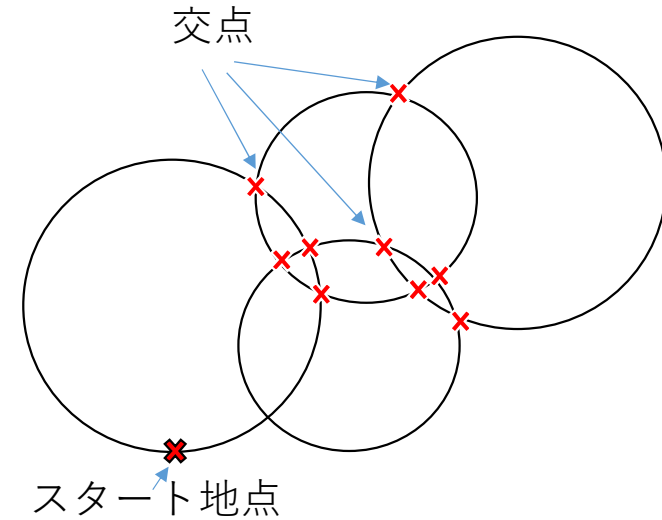
概要

- 座標平面上に円が N ($1 \leq N \leq 100$)個与えられる.
- 各円の中心の x 座標と y 座標 x_i, y_i ($-10,000 \leq x_i, y_i \leq 10,000$)と半径 r_i ($1 \leq r_i \leq 10,000$)は整数.
- スタート地点(原点)を通る円が1つ以上ある.
- 現在たどっている円 A と別の円 B が共有点を持てば、円 A の周上でその共有点まで移動することで円 B の周上に移ることができる.
- どの円の周上にも、スタート地点から到達できる.
- 連環迷路のゴール地点は、 N 個の円の周上のうち、スタート地点からの周上に沿った最短の移動距離が最も大きい点である.

問題 9 連環迷路

解法

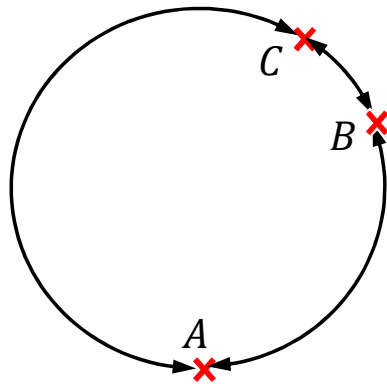
- 交点を列挙する
- スタート地点と、すべての交点を節点とした、グラフに変換する



問題 9 連環迷路

解法

- ダイクストラ法などで、スタート地点からすべての節点への最短距離を求める。
- ゴール地点候補は、各円内で隣り合う節点の間にある点のどこか。



ある円周上で隣り合う点 A と点 B について、以下がわかっている

- ✓ スタート地点から点 A までの最短距離 D_A
- ✓ スタート地点から点 B までの最短距離 D_B
- ✓ 円周上に沿った点 A と点 B の間の距離 D_{AB}

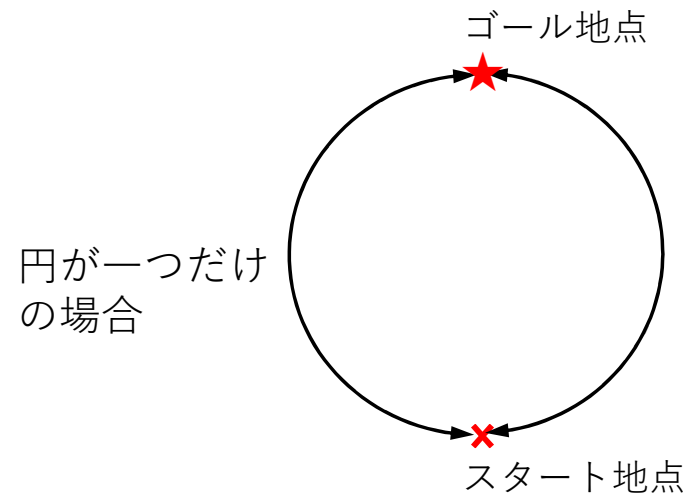
点 A と点 B の間にあるスタート地点から最も遠い点までの距離 L は、 $\frac{1}{2}(D_A + D_B + D_{AB})$ で求まる。

点 B と点 C の間、点 C と点 A の間についても、同様に計算する。
すべての円について同様に計算し、最も大きい値が答え。

問題 9 連環迷路

解法

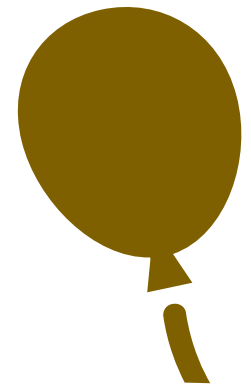
- 計算量 $O(N^3)$
 - 交点列挙 $O(N^2)$
 - グラフへ変換 $O(N^3)$
 - ダイクストラ $O(N^2 \log N)$
 - ゴール地点列挙 $O(N^3)$
- コーナーケースにも注意する.



問題 10 投網 (10点)

正答数: 3チーム

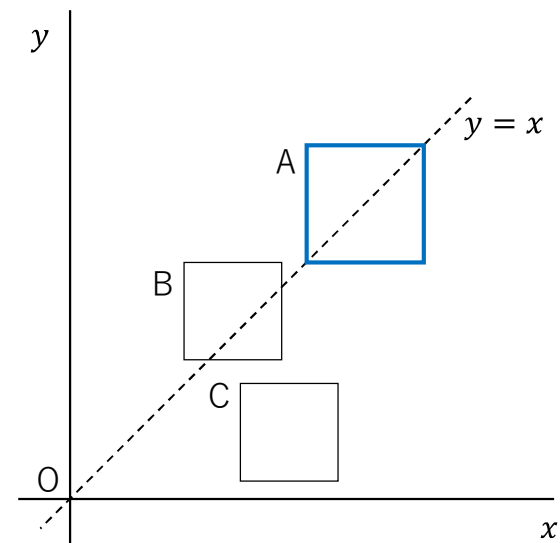
最初の正解: 1時間58分38秒



問題 1 0 投網

概要

- N 群の魚群が座標平面上にいる
 - i 番目の魚群は座標 (x_i, y_i) に存在して、 f_i 匹の魚で構成されている
- 直線 $y = x$ に対角線がかかるかつ、全ての角が格子点に一致するように正方形の網を投げる
- (網の周上or内部にいる魚の数 - 網の周の長さ)の値は最大でいくつになるか？
- $1 \leq N \leq 200,000$
- $0 \leq x_i, y_i \leq 200,000$
- $1 \leq f_i \leq 1,000,000,000$

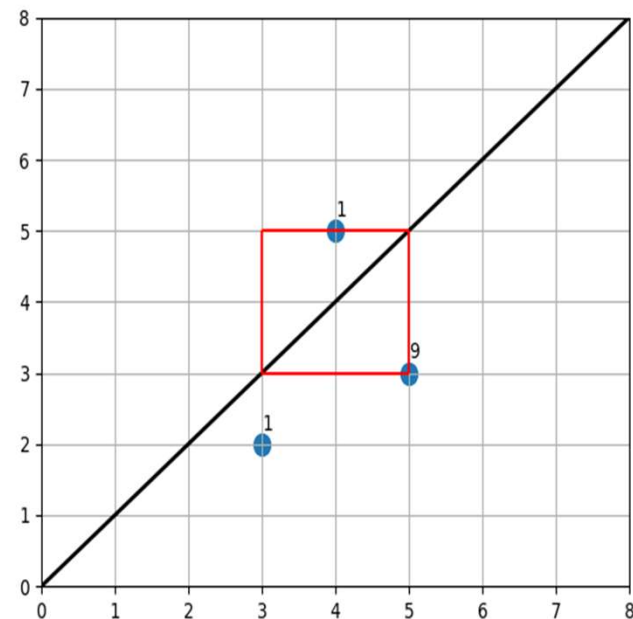


問題 1 0 投網

テストケース

(3, 3)を左下角、(5, 5)を右上角となるように網を投げると、 $10 - 8 = 2$ が利益となる。

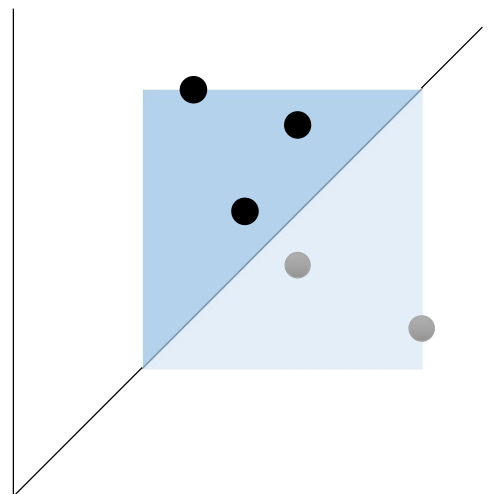
利益を3以上にする網の投げ方は存在しない



問題 1 0 投網

解法

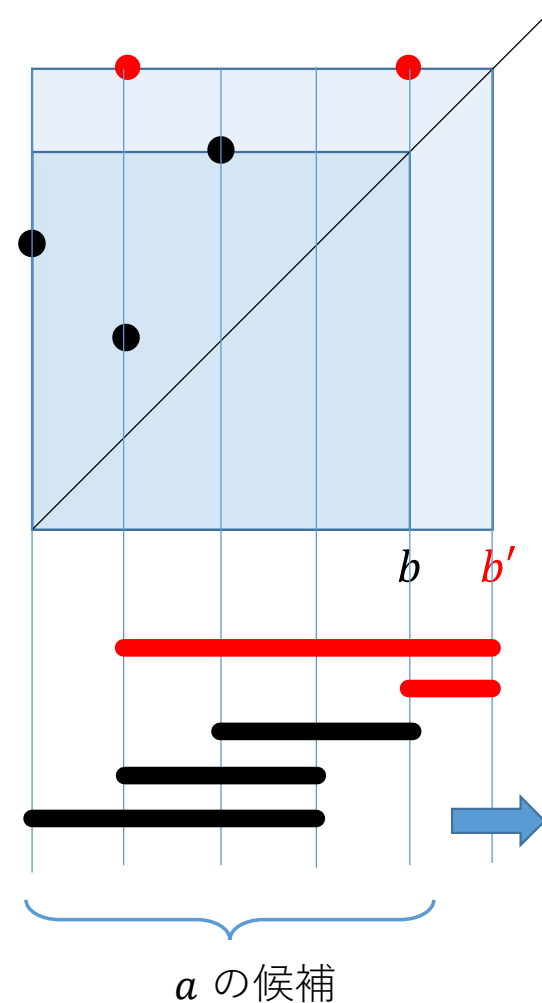
- 簡単のため、直線 $y = x$ の下側にある点については $y = x$ について対称な点へと移して考える。
- 投網の領域 $(a, a), (a, b), (b, b), (b, a)$ は区間 $[a, b]$ で表すものとする。
 - ✓ 点 (x, y) に存在する魚が捕まえられるための条件は、区間 $[x, y]$ が区間 $[a, b]$ に含まれていることと言い換えることができる。
- この変換により、この問題は直線上の区間のクエリに言い換えられる。



問題 1 0 投網

解法

- 投網 $[a, b]$ のうち、 b を昇順に固定して全探索する。
- 「 b を固定したときの、各 a を選択したとしたときに得られる(漁獲高 - 網のコスト)」の列を管理する。このとき、以下の3つのクエリを処理する。
 1. 区間 $[x_i, y_i]$ が $b = y_i$ を満たすような任意の i について $[0, x_i]$ に p_i を加算
 - b が大きくなったことによって、うまい a を選択すると地点 i の魚を得ることができるようになった
 2. 区間 $[0, b)$ に -4 を加算する
 - b が(直前と比較して)1 増えているため、網の枠のコストはそれぞれ4 増やす操作をしている。
 3. $[0, b)$ の max を得る
 - 1, 2 によって b を固定したときの各 a に対する(漁獲高 - 網のコスト)を計算しているため、その最大値が解の候補となる。



問題 1 0 投網

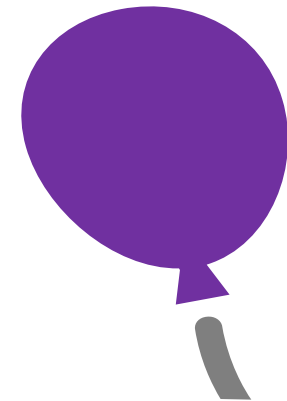
解法

- 区間加算と区間最大値を求めるデータ構造
 - ⇒ 遅延伝搬付きセグメント木(Lazy Propagation Segment Tree)
- 解が -4 となり得ることに注意する。

問題 1 1 串団子 (10点)

正答数: 6チーム

最初の正解: 25分51秒



問題 1 1 串団子

概要

- 数列 A が与えられる
- A から K 個の要素を選び、自由に並び替えたものを B とする
- ありうる B 全てについて、「最大要素の添字の和」の和を求める
- 結果を998244353で割ったあまりを出力する

制約

- $1 \leq N \leq 5000$
- $1 \leq K \leq N$
- $1 \leq a_i \leq 1,000,000,000$

問題 1 1 串団子

テストケース

$$A = (1, 1, 2, 2), K = 2$$

B としてあり得る列は $(1, 1)(1, 2)(2, 1)(2, 2)$ の4通り

$(1, 1)$. . . 最大値は1で、1である添字は1と2 $\rightarrow 1 + 2 = 3$

$(1, 2)$. . . 最大値は2で、2である添字は2 $\rightarrow 2$

$(2, 1)$. . . 最大値は2で、2である添字は1 $\rightarrow 1$

$(2, 2)$. . . 最大値は2で、2である添字は1と2 $\rightarrow 1 + 2 = 3$

$$\text{合計は } 3 + 2 + 1 + 3 = 9$$

問題 1 1 串団子

解法（概要）

- 「 B の最大要素」、「 B の最大要素以外の値の列」を固定した時の寄与を考える
- 状態をうまくまとめて、動的計画法を適用する
- 組み合わせ計算が必要になるので、パスカルの三角形等を利用して前計算しておく

問題 1 1 串団子

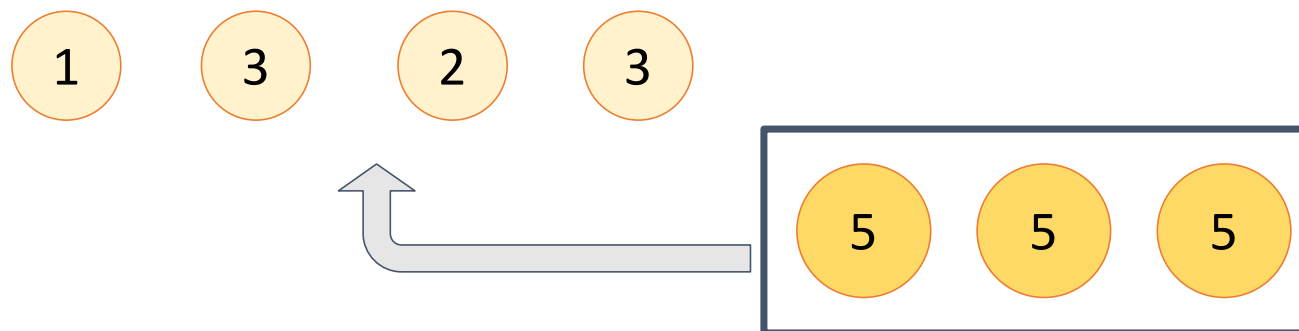
解法 (詳細)

B の最大要素 \max 、 \max の個数 i を固定する (場合の数を求める)

- B の残り($K - i$)要素は、必ず \max より真に小さい値である必要がある
- この($K - i$)要素の内訳とその並べ方を固定した時の寄与を求める

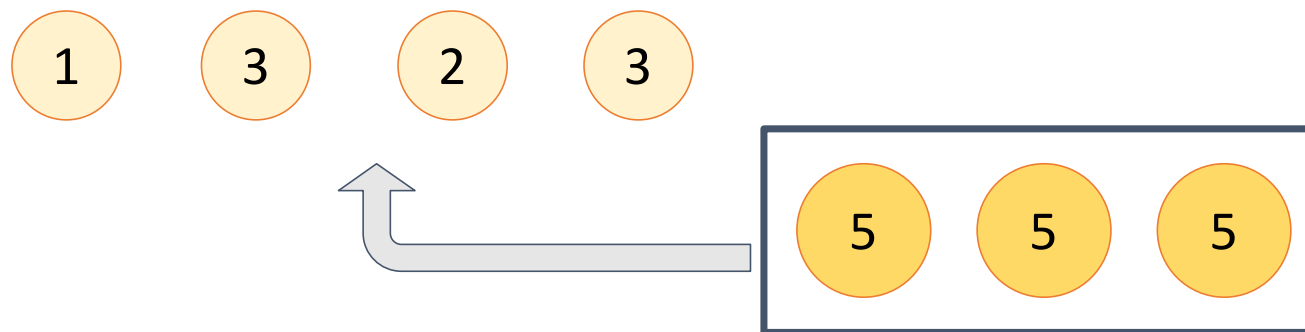
固定した($K - i$)要素の列に \max を i 個まとめて挿入してできる列の \max の添字の和を求めたい

- $K = 7$
- $i = 3$
- $\max = 5$



問題 1 1 串団子

解法 (詳細)

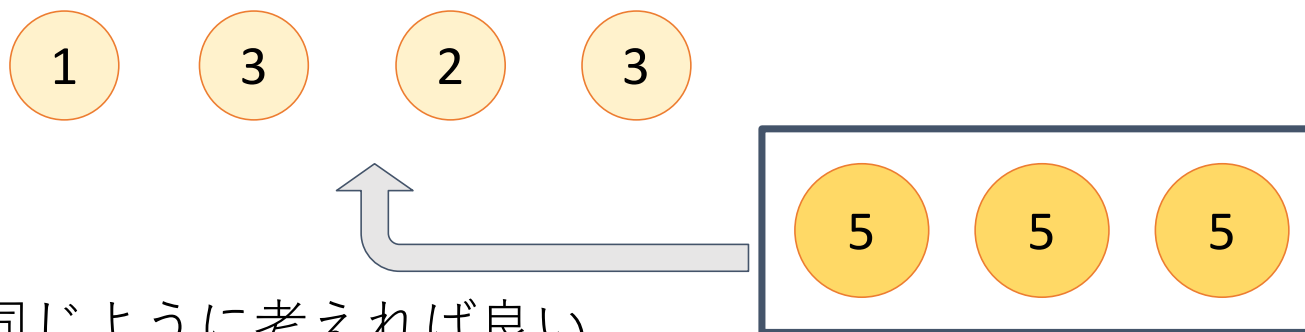


j 番目にmaxが挿入されるような挿入の仕方の数は？

- j 番目に 1個max を挿入して、残りの $(i - 1)$ 個は $(K - 1)$ 項の中から自由に割り当てて良い
- $\Rightarrow {}_{K-1}C_{i-1}$ 通り

問題 1 1 串団子

解法 (詳細)

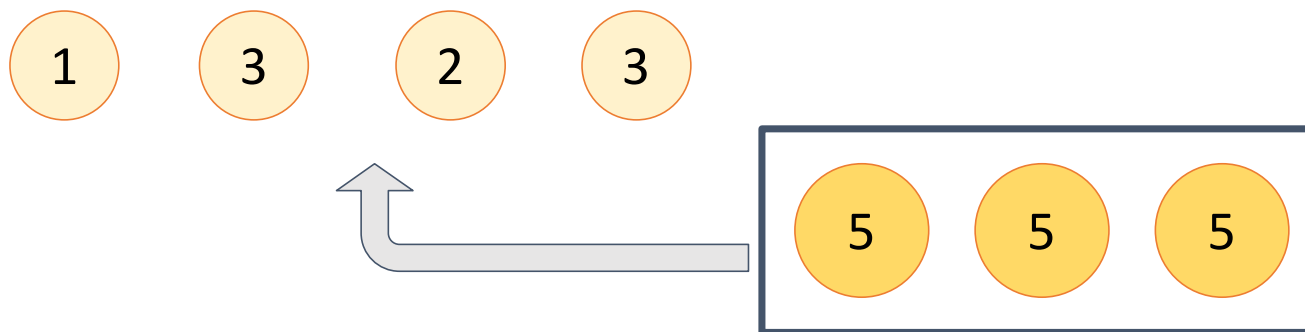


$j = 1, 2, 3, \dots, K$ について同じように考えれば良い

$$\begin{array}{rcl}
 1 & 2 & 3 & 4 & 5 & 6 & \dots & K & & {}_{K-1}C_{i-1} \times 1 & & \\
 1 & 2 & 3 & 4 & 5 & 6 & \dots & K & & {}_{K-1}C_{i-1} \times 2 & & \\
 1 & 2 & 3 & 4 & 5 & 6 & \dots & K & & {}_{K-1}C_{i-1} \times 3 & & \\
 & & & & & & & & & \dots & \Rightarrow & \text{総和} & \\
 & & & & & & & & & & & & = (1 + 2 + \dots + K) \times {}_{K-1}C_{i-1} & \\
 & & & & & & & & & & & & = \frac{K(K+1)}{2} \times {}_{K-1}C_{i-1} &
 \end{array}$$

問題 1 1 串団子

解法 (詳細)



- \max 以外の $(K - i)$ 要素の内訳とその並べ方の数が分かれば「最大要素」と「最大要素の個数」を固定した時の寄与がわかる。
- 動的計画法を用いてこれを求める

問題 1 1 串団子

解法 (詳細)

$dp[i][j]$:= 小さい方から i 種類目までの値を利用して、長さ j の数列を作る通り数とする。

$dp[i][j]$ が既知の状態、 $(i + 1)$ 番目に小さい値を k 個挿入すると

$$dp[i + 1][j + k] = dp[i + 1][j + k] + dp[i][j] \times {}_{j+k}C_k$$

問題 1 1 串団子

解法 (詳細)

$j + k$ 個の箱が一行に並んでいる。 k 個の箱を選んで新しい要素を入れる。残り j 個には元々決まっていた列を「順番を変更せず」入れる

元々決まっていた列の数は $dp[i][j]$ 通りなので、 $dp[i][j] \times {}_{j+k}C_k$

$$j = 4, k = 3, {}_{j+k}C_k$$



問題 1 1 串団子

解法（詳細）

- $dp[0][0] = 1$ が自明であり、ここから遷移を進めていく。
- このDPテーブルは $O(NK)$ で埋めることができる。
- 次に、「最大要素について小さい方から*i*種類目の値」と「最大要素の個数について*j*個」について解を求め、総和を計算する。

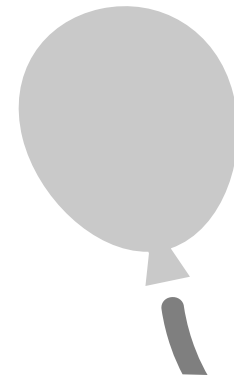
$$sum = sum + dp[i-1][K-j] \times \frac{K(K+1)}{2} \times {}_{K-1}C_{i-1}$$

- この計算は、各値の頻出数を考えると、 $O(N)$ で計算できる。

問題 1 2 星座を探して (13点)

正答数: 0チーム

最初の正解: -



問題 1 2 星座を探して

概要

- 星座が $N(1 \leq N \leq 1,000)$ 個与えられる.
- 各星座は $M(1 \leq M \leq 1,000)$ 個の座標平面上の点 $x_i, y_i(-1,000 \leq x_i, y_i \leq 1,000)$ 点で構成されている.
- 以下の操作を、任意の回数組み合わせることで一致させることができる星座は、同じ種類の星座と考える.
 - 平行移動
 - 原点を中心とした回転
 - 原点を中心として、正の係数を倍率とした拡大・縮小
- それぞれの星座について、同じ種類の星座がいくつあり、何番目の星座が同じ種類の星座かを求める.

問題 1 2 星座を探して

解法

- 各星座の座標を M 倍して重心を求め、原点へ平行移動する.
- 各星座内で、星座を構成する点を偏角でソートしておく.
 - 偏角順で見たときに、次の点との間でなす角を求めた列 T を作る.
- 各星座内で、星座を構成する点の原点からの距離の2乗を求めておく.
 - 全ての点について、距離の2乗を最大公約数で割り算した列 R を作る.
- T と R の要素をペアにして配置した列 $S = \{(T_1, R_1), (T_2, R_2), \dots, (T_M, R_M)\}$ を作る.
 - このとき T の要素は大きな数 (10^8 など) を掛けて、整数に切り上げる.

問題 1 2 星座を探して

解法

- 比較演算子を $T_i < T_j \cup (T_i = T_j \cap R_i < R_j) \rightarrow (T_i, R_i) < (T_j, R_j)$ のように定義する(C++などの標準的なペアの比較演算子).
- $S = \{(T_1, R_1), (T_2, R_2), \dots, (T_M, R_M)\}$ の要素をシフトして回転したときに、辞書順最小になるように並べ替える.
- 列 S をローリングハッシュでハッシュ値に変換する.
- 星座同士のハッシュ値が一致すれば、それらの星座は同一であるとみなす.

問題 1 2 星座を探して

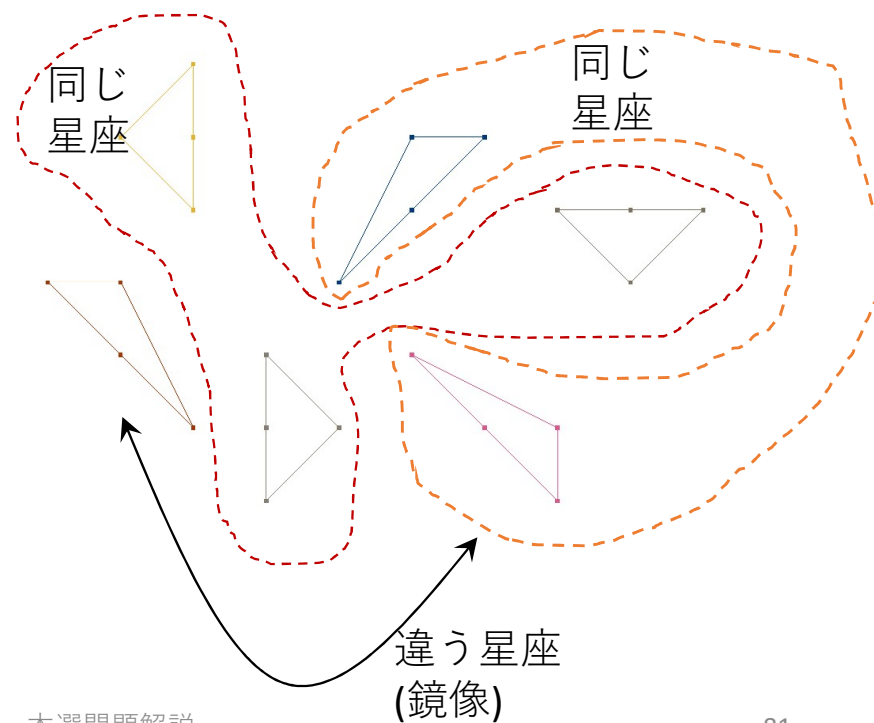
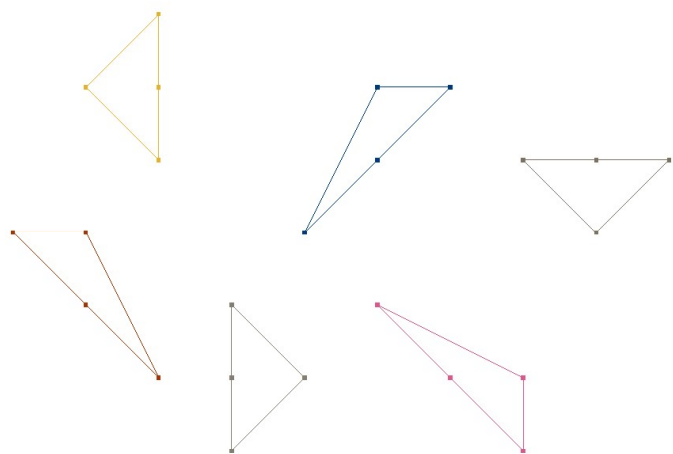
解法

- 計算量 $O(N^2 + NM \log M)$
 - 各星座の偏角ソート $O(M \log M)$
 - 各星座の距離の最小公倍数 $O(M \log |x_i^2 + y_i^2|)$
 - 列 S の辞書順回転 (BoothのLexicographically Minimal String Rotation algorithm) $O(M)$
 - ハッシュ値への変換 $O(M)$
 - 以上を N 回行う
 - ハッシュ値の比較 $O(N^2)$

問題 1 2 星座を探して

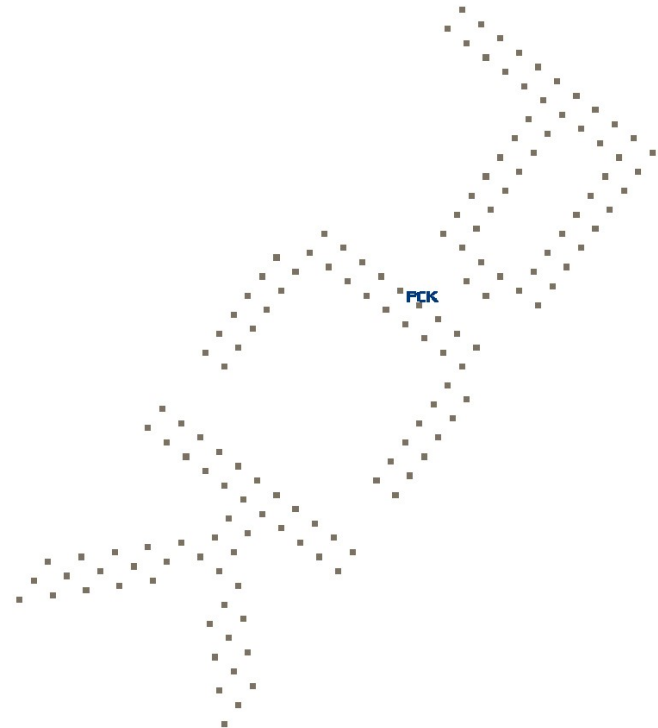
入力データ

01_small_01.in



問題 1 2 星座を探して

10_beautiful_01.in

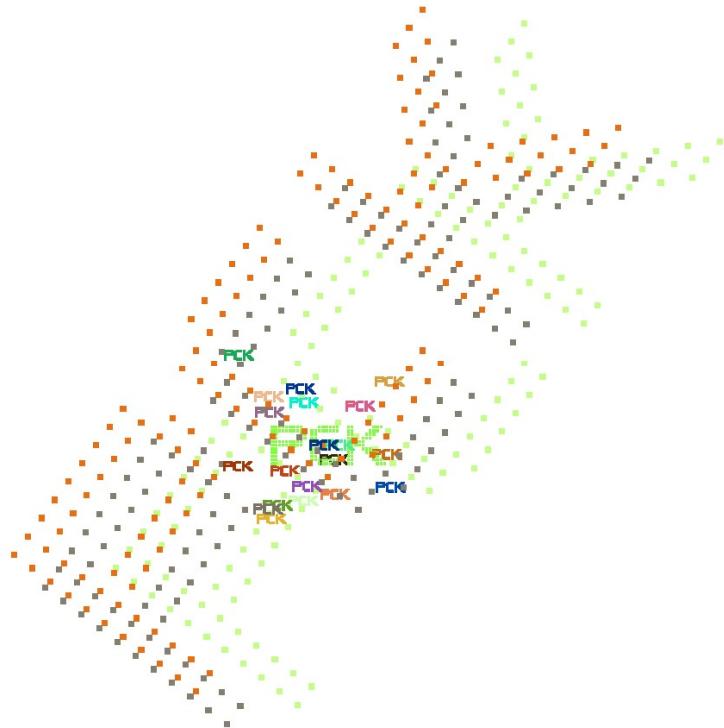


10_beautiful_02.in

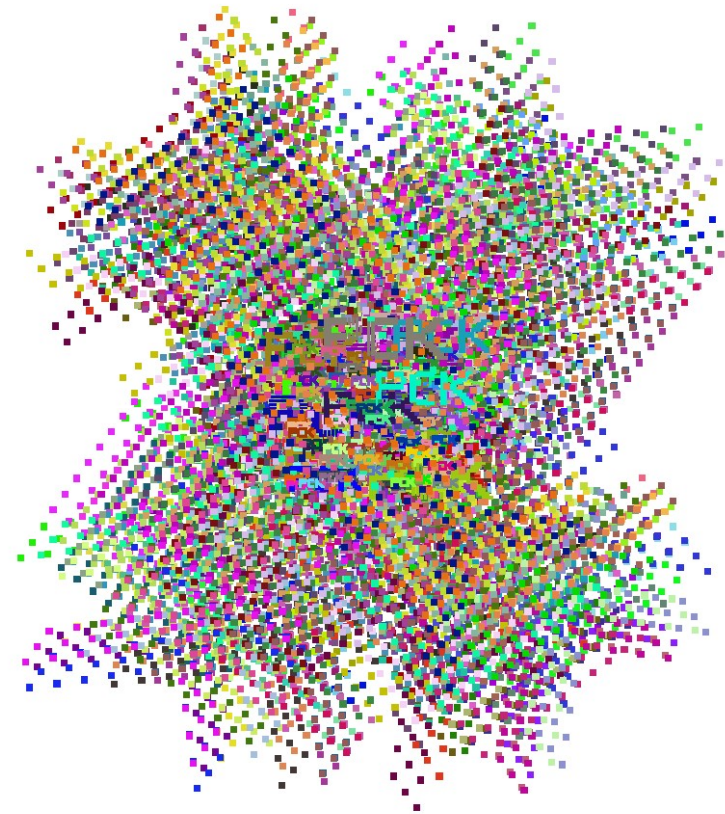


問題 1 2 星座を探して

10_beautiful_03.in



10_beautiful_04.in



問題 1 2 星座を探して

10_beautiful_05.in



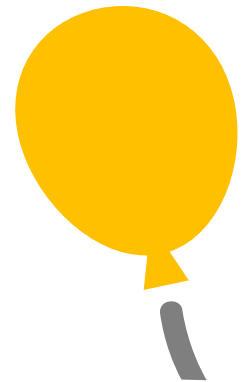
問題 1 3 点の削除 (1 3 点)

正答数:

1チーム

最初の正解:

3時間30分16秒



問題 1 3 点の削除

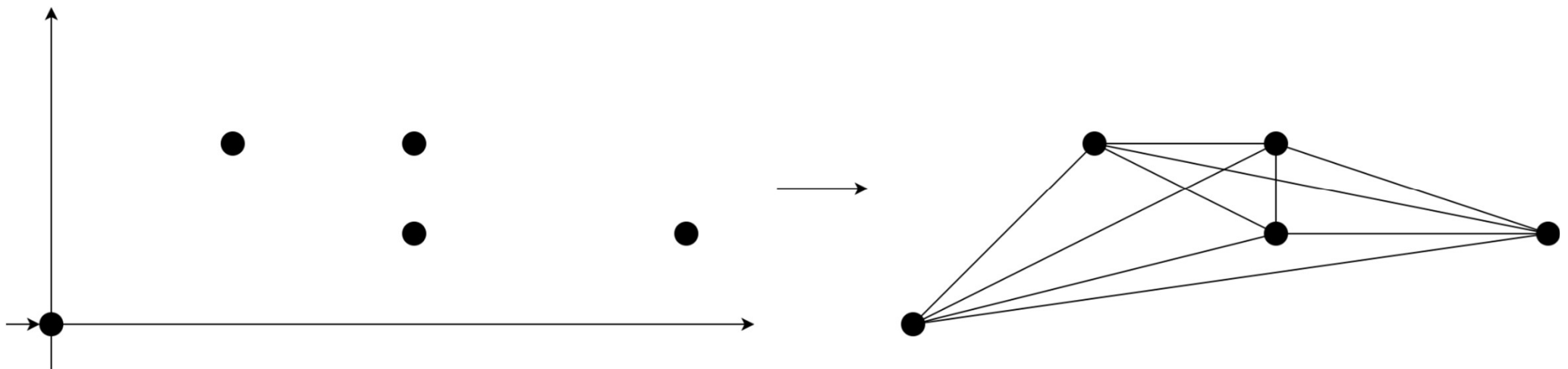
概要

- 座標平面上の N 個の点を与えられる
- 点を K 個削除した時の頂点間距離の二乗の最小値を最大化する
- $3 \leq N \leq 1000$
- $0 \leq K \leq N - 2, K \leq 30$

問題 1 3 点の削除

考察

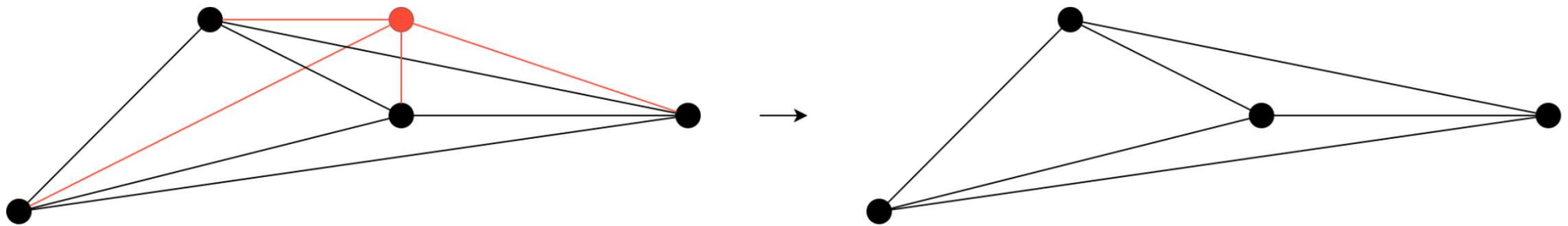
- 各点をグラフの頂点、二点間の距離を辺重みとしたグラフとして考える



問題 1 3 点の削除

考察

- ある頂点を削除すると、その頂点を端点とする辺を削除することができる



問題 1 3 点の削除

解法

- 課題: 辺重みの最小値の最大化
- 全ての辺を重みで昇順ソートした後、二分探索する
 - 判定問題 $f(x)$
 - 重みの小さい方から x 個の辺のみを含むグラフに対して、頂点を最大 K 回削除することによって、全ての辺を削除できるか？
 - (大きさが K 以下の頂点被覆は存在するか？)
- 以下、この判定問題の解法を考える

問題 1 3 点の削除

$O(N^{K+1})$ 解法(TLE)

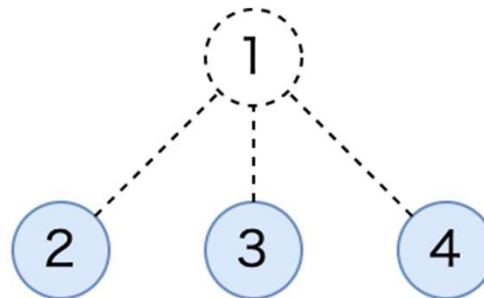
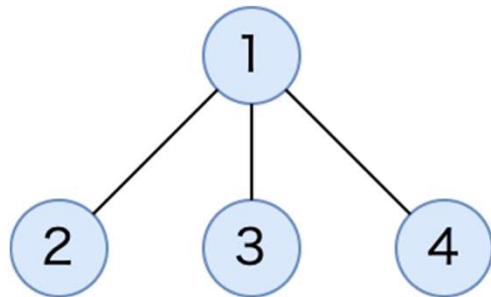
- 各頂点に対して愚直に削除するか削除しないかを割り当てる
- この解法を高速化する

```
bool dfs(int i, int k) {
    if (k < 0) return false;
    if (i == n) {
        return is_vertex_cover();
    }
    status[i] = DELETE;
    if (dfs(i + 1, k - 1)) return true;
    status[i] = KEEP;
    if (dfs(i + 1, k)) return true;
    return false;
}
```

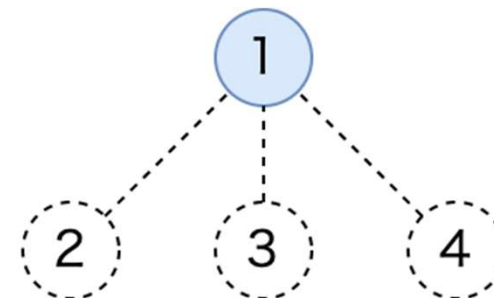
問題 1 3 点の削除

考察

- ある頂点を削除する場合
 - ✓ その頂点のみを削除する
- ある頂点を削除しない場合
 - ✓ 隣接する全ての頂点を削除する必要がある



頂点1を削除する場合

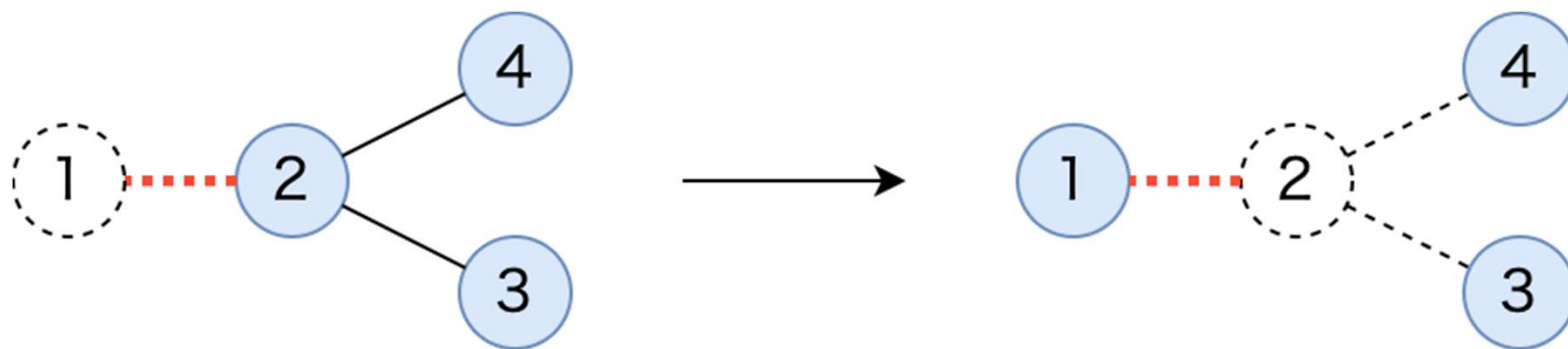


頂点1を削除しない場合

問題 1 3 点の削除

考察

- 次数が1以下の頂点は削除する必要はない
 - 次数が1の場合
 - 隣接する頂点を削除することによって損しない



頂点1の代わりに頂点2を削除しても損しない

問題 1 3 点の削除

解法

- 各頂点を順番に見ていき、削除する・しないを決めていく
 - 次数が1以下 -> 削除しない
 - 次数が2以上 -> 削除する、または削除しない
- 次数が2以上の場合のみ分岐する
- この場合分けによって計算量が改善する

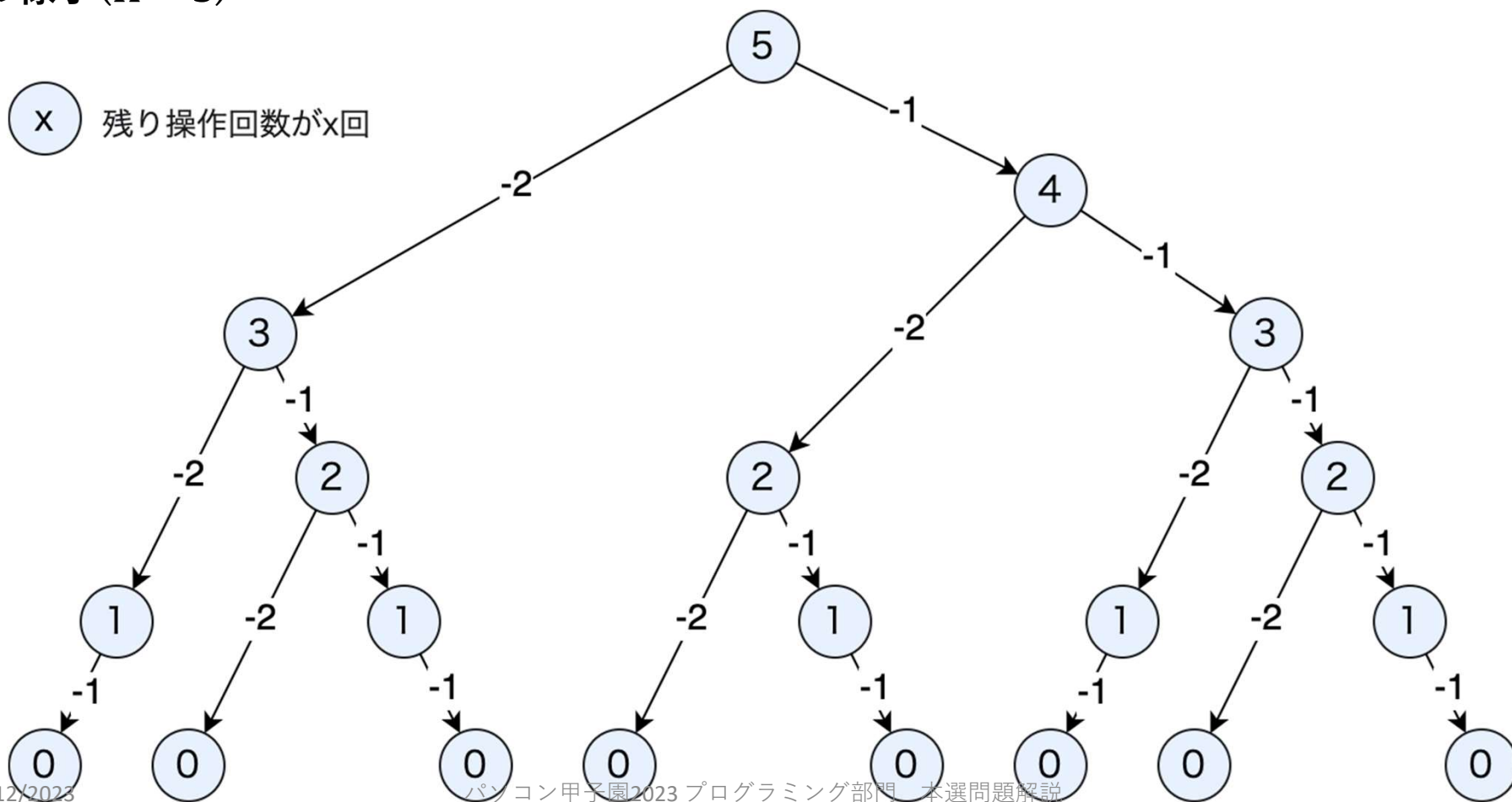
問題 1 3 点の削除

考察

- 分岐するときの残り削除回数 K の減少量に注目する
 - 削除する場合
 - ✓ 1減少する
 - 削除しない場合
 - ✓ 少なくとも2減少する
 - ✓ (次数が2以上かつ隣接する頂点を全て削除する必要があるため)
- 分岐する度に K が1または2以上減少する

問題 1 3 点の削除

分岐の様子(K = 5)



問題 1 3 点の削除

考察

フィボナッチ数列

- $F_0 = 0$
- $F_1 = 1$
- $F_k = F_{k-1} + F_{k-2} (k \geq 2)$

$$F_{30} = 832,040, F_{31} = 1,346,269$$

全体の分岐回数は高々 F_{K+1} 回

判定問題を $O(NK + NF_K)$ で解くことができる

問題 1 3 点の削除

計算量

- 頂点間距離のソート: $O(N^2 \log N)$
- 判定問題一回あたり: $O(NK + NF_K)$
- 全体: $O(N^2 \log N + NF_K \log N)$

ご視聴ありがとうございました。
また来年2024で!!

