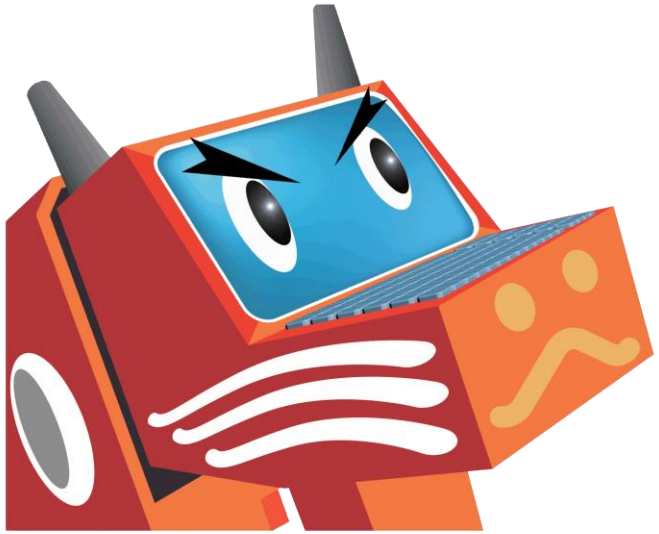
パソコン甲子園2020

に・ぜろ・に・ぜろ

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

#	タイトル	分野	得点	難易度	
				思考	実装
1	ケーキの価格	基礎	2	☆	☆
2	角度の変換	基礎	3	☆	☆
3	ホットケーキ	基礎	4	★	☆
4	最小のMAW	文字列	5	★☆	★
5	スロットマシン	整数	5	★☆	★
6	はんぶんこ	計算幾何学	7	★★☆	★☆
7	旅館の客室番号	整数	7	★☆	★★☆
8	壊れた出口	グラフ	9	★★★	★★☆
9	ロボットアーム	グラフ・数学	10	★★★☆	★★
10	柿の実	計算幾何学	11	★★★	★★★☆
11	高速道路網の再編	グラフ	11	★★★★	★★☆
12	交換と転倒数	データ構造	12	★★★★☆	★★★★
13	山小屋スタンプラリー	木・数え上げ・数学	14	★★★★★	★★★★★

問題１　ケーキの価格

概要

- 軽減税率制度が適用された金額を計算する.
- ケーキの税抜き価格pが与えられたとき，以下の差を計算せよ：
  - 店内で食べる場合　→　10 % の税率が適用される
  - 持ち帰る場合　→　8 %の税率が適用される
- 50 ≦ p ≦ 5,000.　pは50の倍数.

問題１　ケーキの価格

解法

$$\text{差額} = p \times 1.10 - p \times 1.08$$

$$= p \times (1.10 - 1.08)$$

$$= p \times 0.02$$

$$= \frac{p}{50}$$

pは 50の倍数　→　p を 50 で割った値を出力する.

整数で計算できる.

問題１　ケーキの価格

解答例

```
int main(){
  int x;
  cin >> x;
  cout << x / 50 << endl;
  return 0;
}
```

問題２　角度の変換

概要

- 秒で表した角度dを度分秒に変換して出力せよ.
  - 1度より小さい角度まで表したいときには，分や秒という単位を使う.
  - 1度を分で表すと60分、１分を秒で表すと60秒になる.
  - たとえば，10.52度を秒で表すと37,872秒，度分秒で表すと10度31分12秒になる.

問題２　角度の変換

- 単位の変換

```
1 [度] =   60 [分]
1 [分] =   60 [秒]
```

```
1 [度] = 3600 [秒]
```

入出力例

```
37872                              [秒]

37872 / 3600          = 10 [度]
37872 % 3600 / 60 = 31 [分]
37872 % 60            = 12 [秒]
```

問題２　角度の変換

解答例

```
int d;
cin >> d;
cout << d / 3600 << " " << d % 3600 / 60 << " " << d % 60 << endl;
```

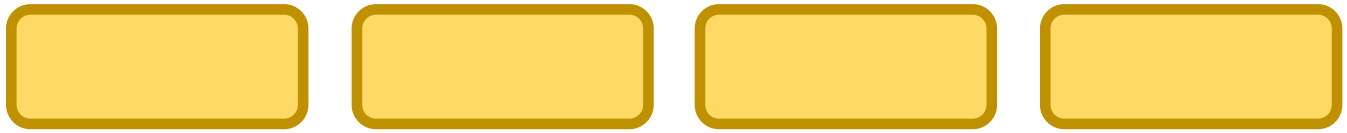
問題３　ホットケーキ

概要

- あなたは１つのフライパンを持っている.
- このフライパンでは，一度にホットケーキを３枚まで，片面だけ焼くことができる.
- ホットケーキは１分で片面が焼きあがる.
- N枚のホットケーキを作るために必要な最短の時間（分）を求めよ.
- $1 \leqq N \leqq 1{,}000$

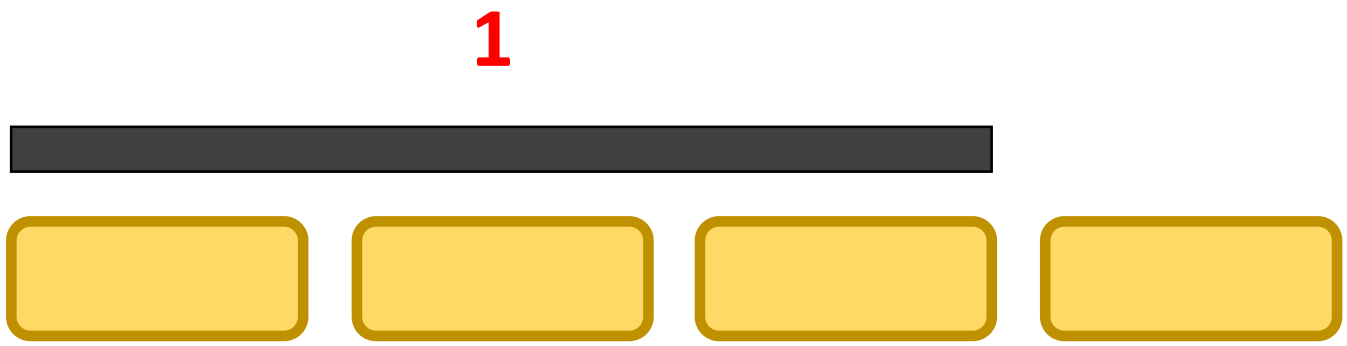
問題３　ホットケーキ

$N = 4$



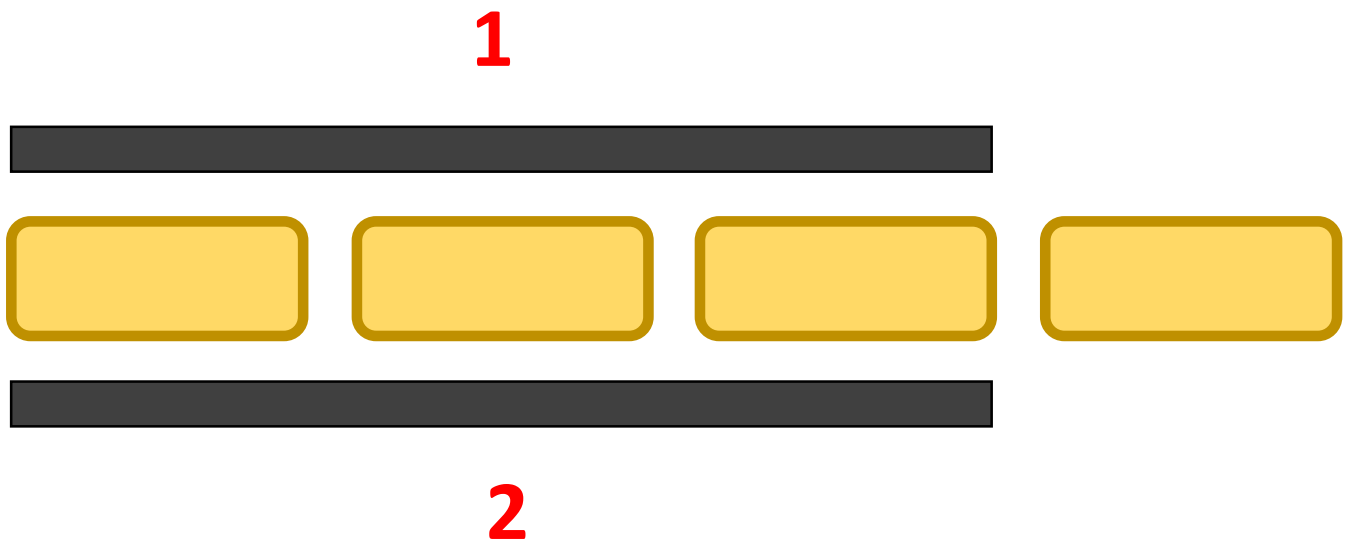
問題３　ホットケーキ

$N = 4$



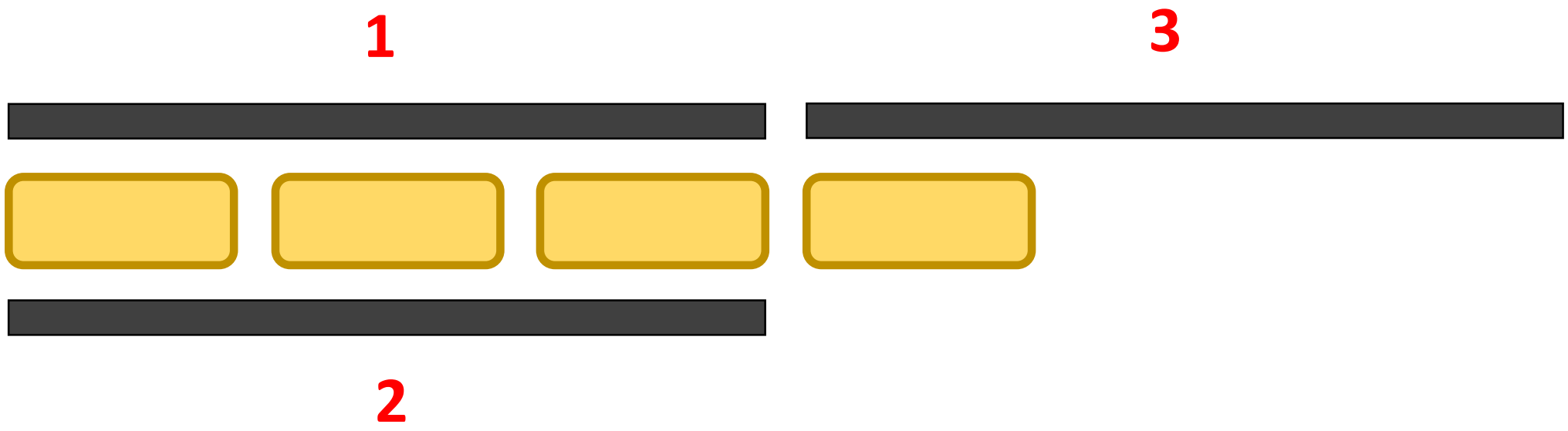
問題３　ホットケーキ

$N = 4$



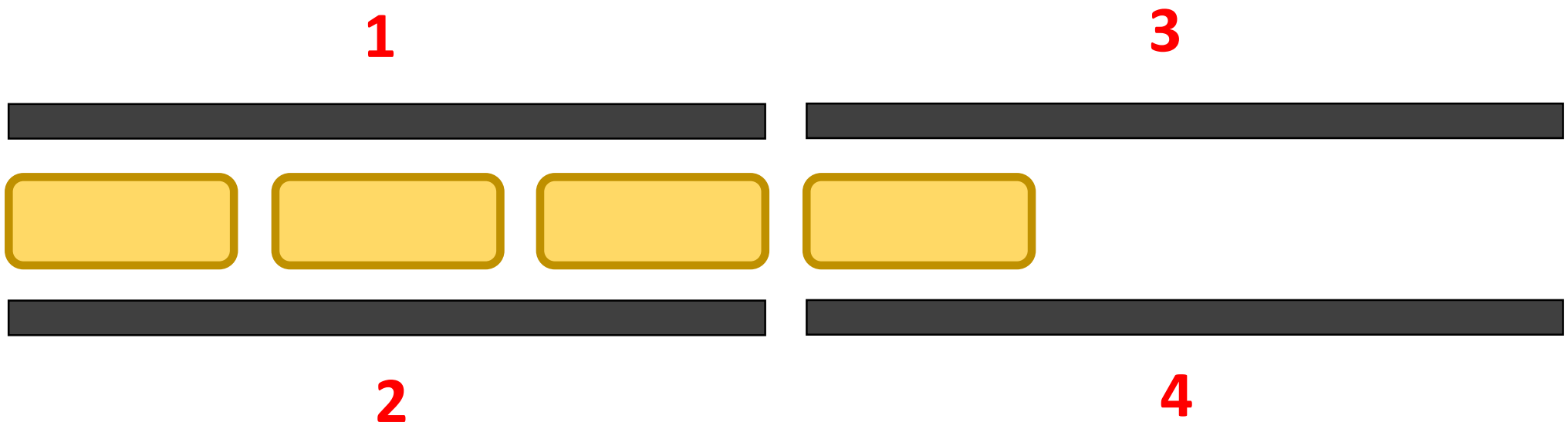
問題３　ホットケーキ

$N = 4$



問題３　ホットケーキ

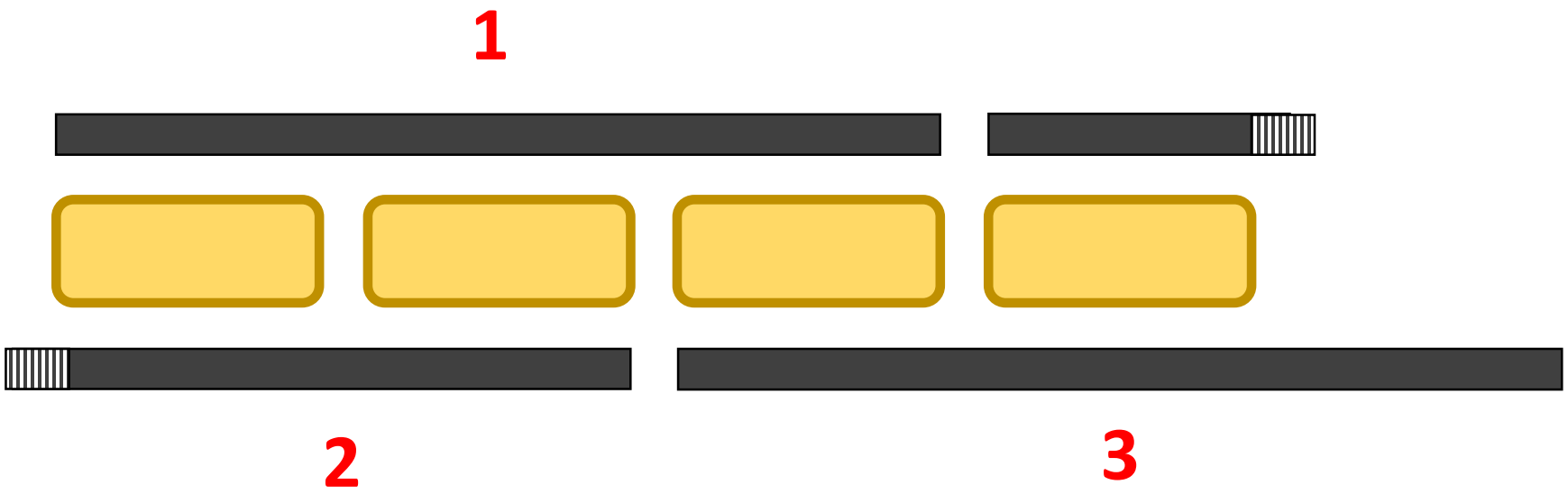
$N = 4$



最適だろうか？

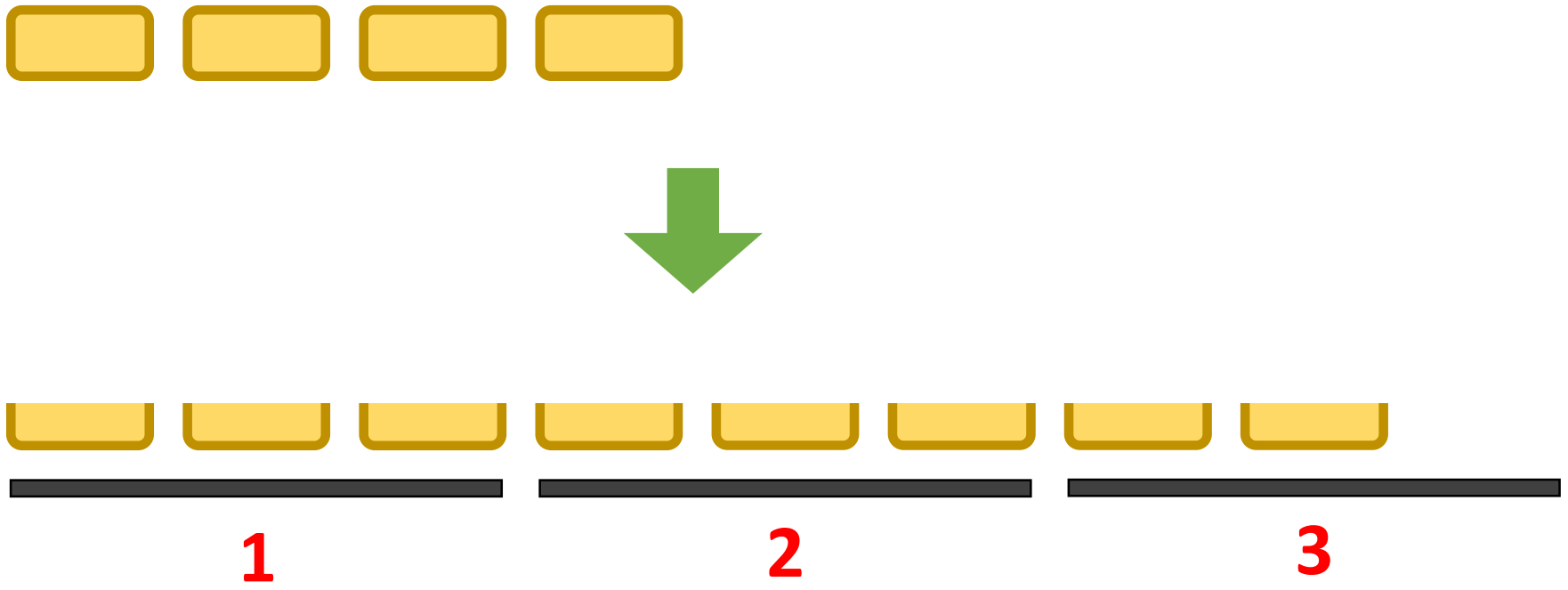
問題３　ホットケーキ

$N = 4$



３回で焼ける

問題３　ホットケーキ



問題３　ホットケーキ

解法（式１）

```
ans = (2 * N) / 3 + ((N % 3 == 0)?0:1)
```

解法（式２）

```
ans = (2 * N + 2) / 3
```

解法（式３）

```
ans = 2 * (N / 3) + N % 3
```

問題３　ホットケーキ

コーナーケース

```
if ( N < 3 ) ans = 2        /* 最低でも２回は必要 */
```

問題３　ホットケーキ

解答例

```
int N;
cin >> N;
cout << max(2, (2*N + 2) / 3) << endl;
```

問題４　最小のMAW

概要

- 長さNの文字列Sから，以下の条件を満たす文字列Tを作成したい．
  - Tの長さは２以上
  - TはSの連続部分文字列ではない
  - Tの連続部分文字列でTでないものはすべて，Sの連続部分文字列である
- 上の条件を満たす文字列のうち辞書式順序で最小の文字列を求めよ．
- $3 \leqq N \leqq 400{,}000$

問題４　最小のMAW

- 長さNの文字列Sから，以下の条件を満たす文字列Tを作成したい．
  - Tの長さは２以上
  - TはSの連続部分文字列ではない
  - Tの連続部分文字列でTでないものはすべて，Sの連続部分文字列である
- 上の条件を満たす文字列のうち辞書式順序で最小の文字列を求めよ．

入力例1

```
4
aabb
```

- S = aabb
- T = aaa, ba, bbb
- aaa が辞書式順序で最小

問題４　最小のMAW

解法

- Sの中で最も小さい文字を探す．これをchとする．
- chが連続する文字列の長さ（ランレングス）の最大値Lを求める．
- chを$L+1$個並べた文字列を出力する．
- O(N).

問題４　最小のMAW

正当性

- そのような文字列をAとすると、Aより辞書順が小さい文字列は
  (1) Aから何文字か削った文字列，または
  (2) Aの一部をそれより辞書順が小さい文字で書き換えたもの

- (1)はSに含まれるのでＮＧ
- (2)は置き換える文字がSの中に存在しないためＮＧ

入力例2

```
4
tree
```

- S = tree
- A = eee

問題４　最小のMAW

解答例

```
int N; cin >> N;
string s; cin >> s;
char ch = 'z';
for ( int i = 0; i < N; i++ ) ch = min(ch, s[i]);
int len = 0;
int ans = 0;
for ( int i = 0; i < N; i++ ){
  if ( s[i] == ch ){
    len++;
    ans = max(ans, len);
  } else {
    len = 0;
  }
}
for ( int i = 0; i < ans + 1; i++ ) cout << ch;
cout << endl;
```

問題４　最小のMAW

おまけ

```
int N; cin >> N;
string s; cin >> s;
char ch = *min_element(s.begin(), s.end());
string cur(1, ch);
while( s.find(cur) != string::npos )  cur += ch;
cout << cur << endl;
```

$O(N^2)$ → **時間制限**

問題５　スロットマシン

概要

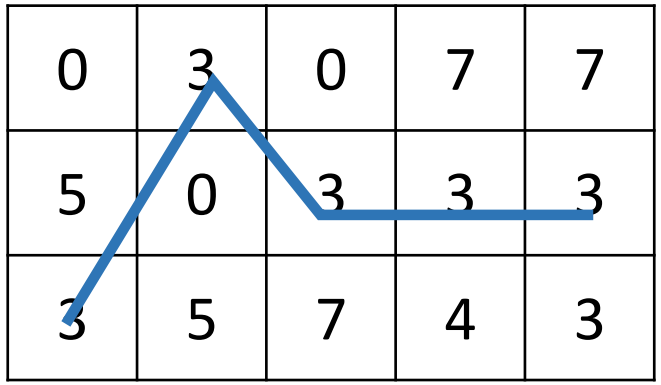
- $W \times H$個のマスからなるスロットマシンに表示されている数字が与えられたとき，得られる得点を求めよ．
- $1 \leqq W \leqq 1{,}000$，$1 \leqq H \leqq 1{,}000$

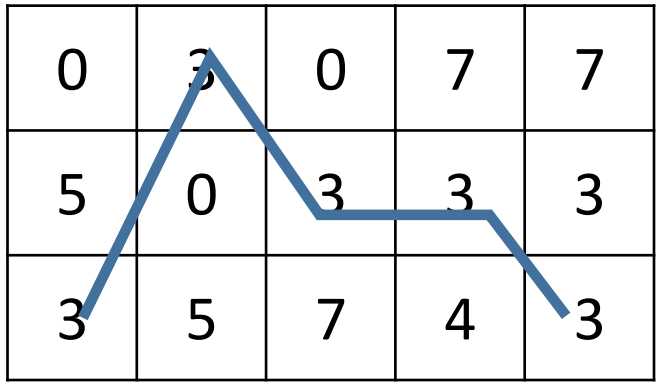
0	3	0	7	7
5	0	3	3	3
3	5	7	4	3

問題５　スロットマシン

入出力例１

0	3	0	7	7
5	0	3	3	3
3	5	7	4	3



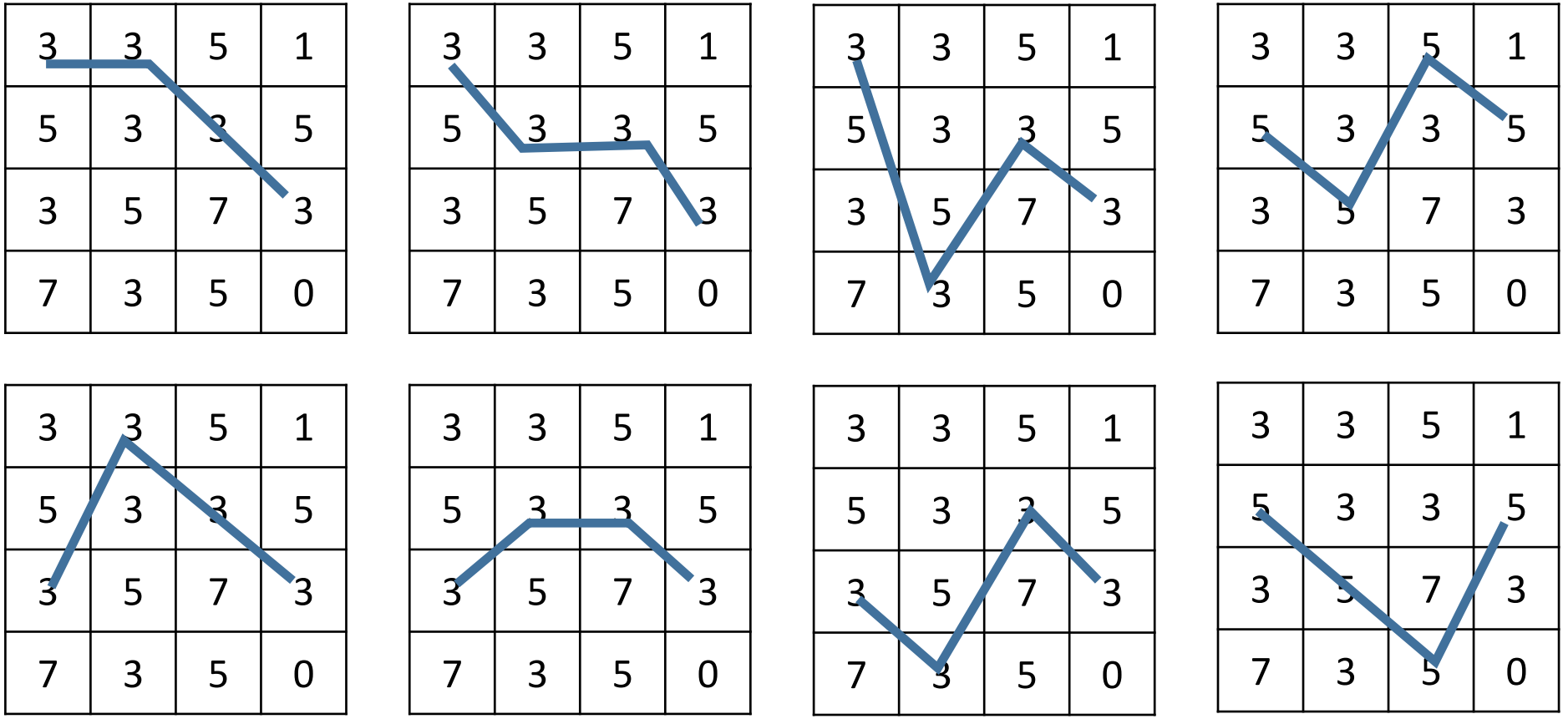


３　＋　３　＝　６

問題５　スロットマシン

入出力例２

3	3	5	1
5	3	3	5
3	5	7	3
7	3	5	0



$3 \times 6 + 5 \times 2 = 28$

問題５　スロットマシン

解法

- 各列について，それぞれの数字が何個あるかを数える.
- 各数字について，その数字と全ての列の個数を掛けあわせた値を答えに加算する.
- ただしmod 998,244,353

	0	1	2	3
	3	3	5	1
	5	3	3	5
	3	5	7	3
	7	3	5	0

0	0	0	0	1	
1	0	0	0	1	
2	0	0	0	0	
3	2	3	1	1	18
4	0	0	0	0	+
5	1	1	2	1	10
6	0	0	0	0	‖
7	1	0	1	0	28

・・・

問題５　スロットマシン

解答例

```
// 各列の数字の個数を数える
for ( int i = 0; i < H; i++ ){
  for ( int j = 0; j < W; j++ ){
    char d; cin >> d;
    cnt[d - '0'][j]++;
  }
}
```

```
// 得点の加算
long long ans = 0;
for ( int i = 0; i <= 9; i++ ){
  long long r = 1;
  for ( int j = 0; j < W; j++ ){
    r = (r * cnt[i][j]) % M;
  }
  r = (r * i) % M;
  ans = (ans + r) % M;
}
```

問題６　はんぶんこ

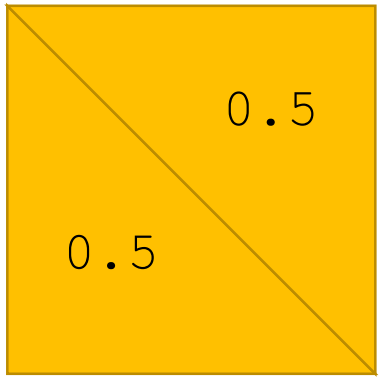
概要

- N個の点で構成される煎餅（凸多角形）が与えられる.
- １本の対角線に沿って煎餅を２つに分割したときの，それらの面積の差の最小値を求めよ.
- $4 \leqq N \leqq 60{,}000$
- $0 \leqq x_i, y_i \leqq 100{,}000{,}000$
- x_i, y_iは整数

問題６　はんぶんこ

入出力例1

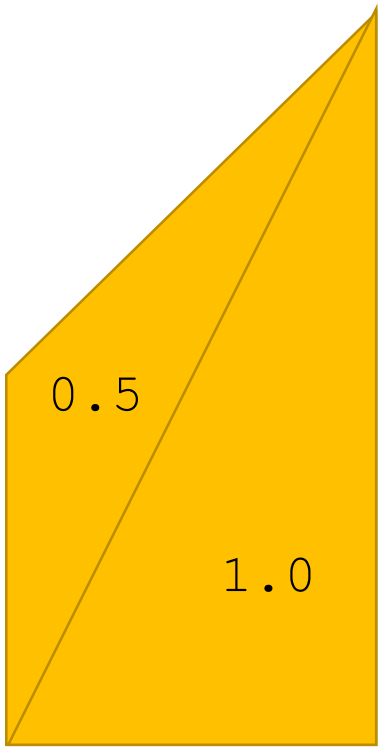
```
4
0 0
1 0
1 1
0 1
```




```
0.0
```

入出力例2

```
4
0 0
1 0
1 2
0 1
```




```
0.5
```

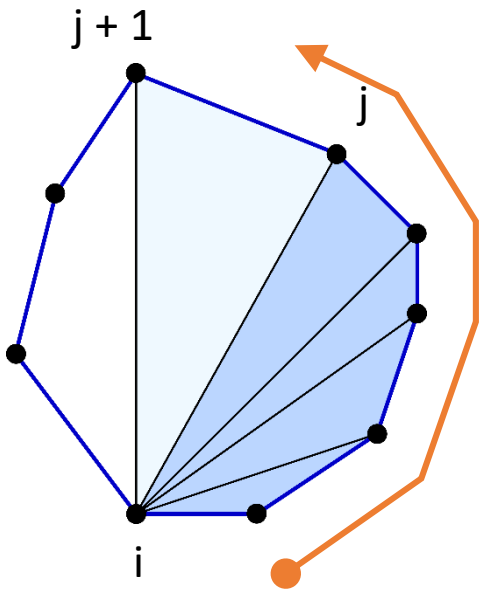
問題６　はんぶんこ

解法１（※時間制限）

- 全ての対角線で分割してみて，最小値を求める．
- N個から対角線の１つの端点p_iを決め打ちして，残りのp_jをO(N)で走査
  - 面積は累積和で求まる

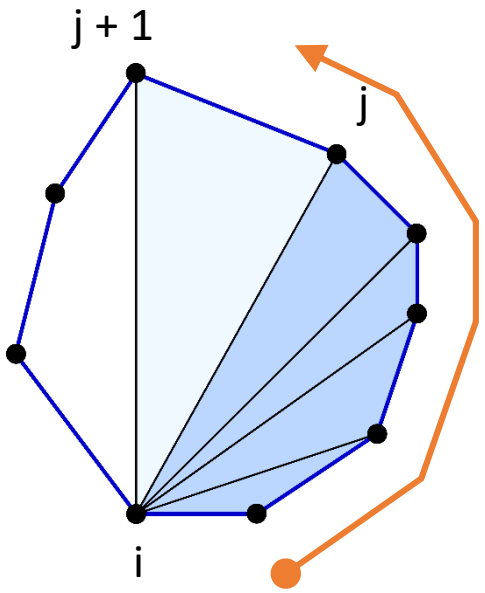
問題６　はんぶんこ

解法１



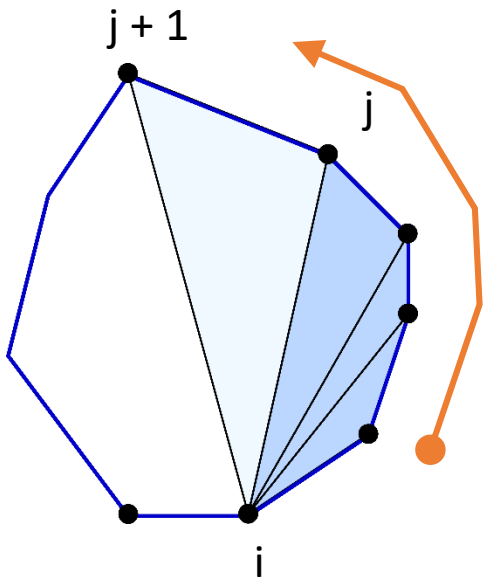
問題６　はんぶんこ

解法１



三角形を追加した場合に
半分を超える

i++
j = i + 1



三角形を追加した場合に
半分を超える

i++
j = i + 1

問題６　はんぶんこ

解法１

- 全体の面積をA，追加する三角形の面積をΔ，現在の面積を累積和Sとする．
- 以下の過程で，$|A - 2(S + \Delta)|$の最小値を更新していく．
  - $\Delta + S$が$\frac{A}{2}$を超えない
    - jをインクリメントしてSにΔを加算
  - $\Delta + S$が$\frac{A}{2}$を超える
    - iをインクリメントしてjをi+1にリセットする．Sも0にリセットする
- O(N^2)

→　**時間制限**

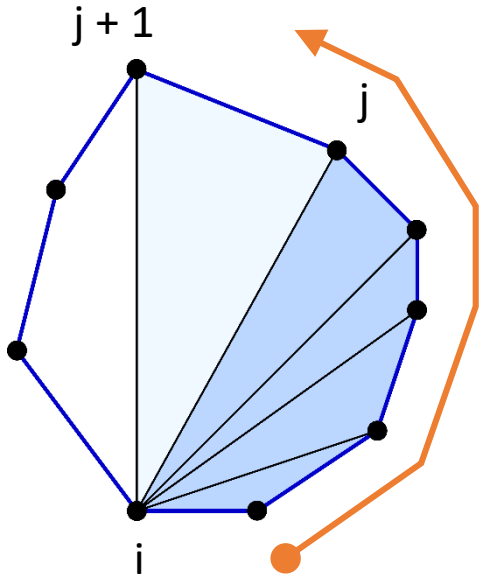
問題６　はんぶんこ

解法２

- 累積和Sの差分を有効利用して高速化する.

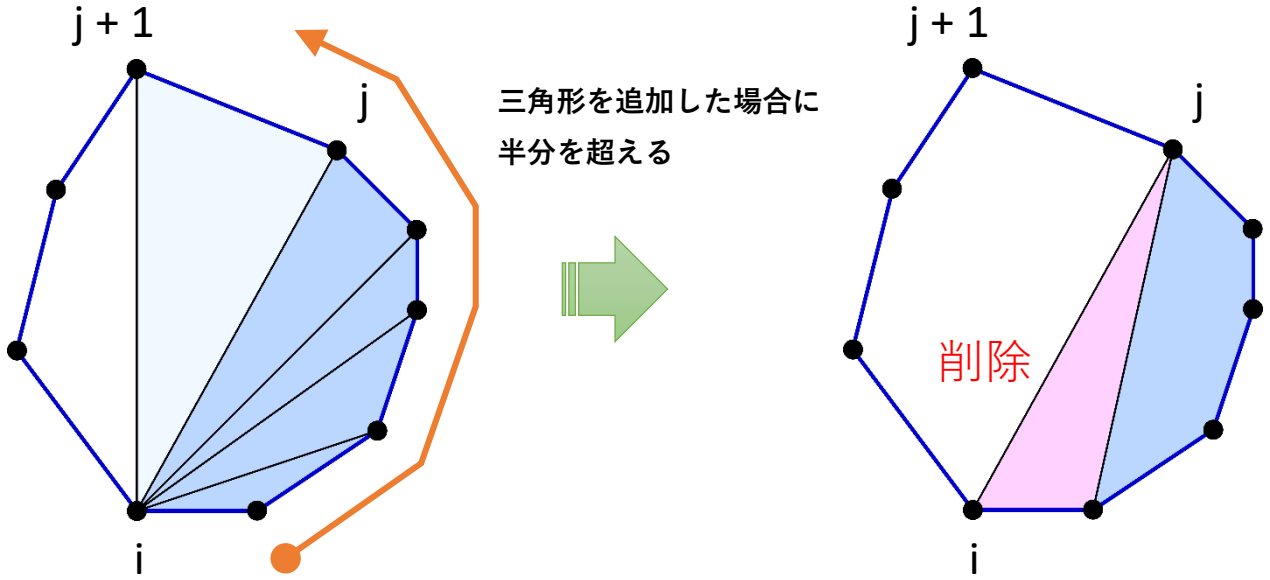
問題６　はんぶんこ

解法２



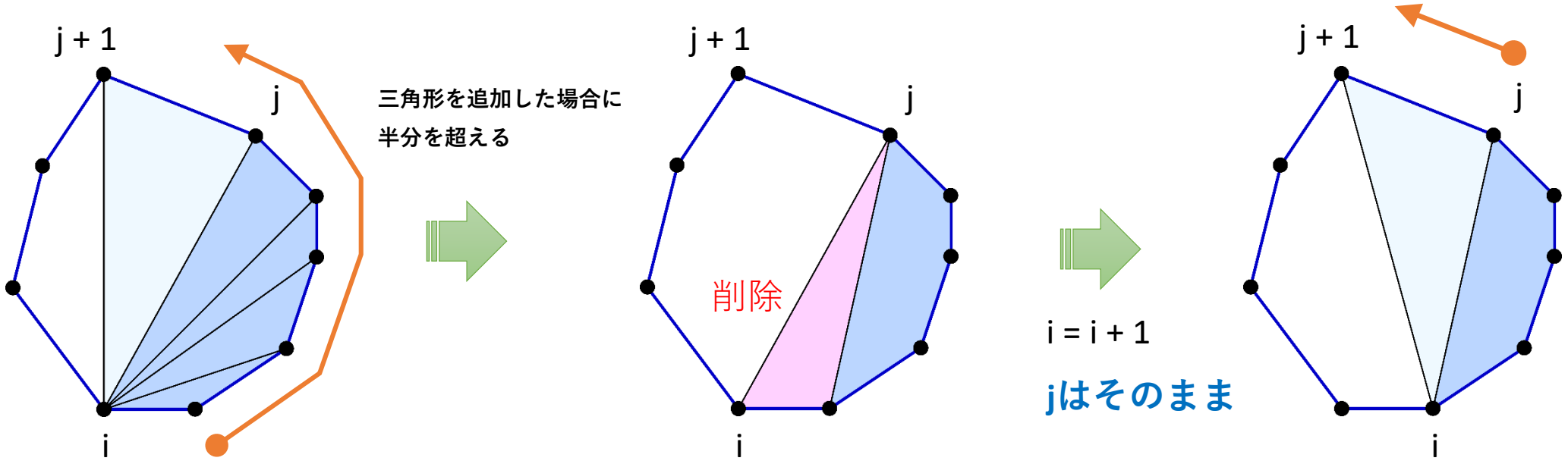
問題６　はんぶんこ

解法２



問題６　はんぶんこ

解法２



問題６　はんぶんこ

解法２

- 全体の面積をA，追加する三角形の面積をΔ，現在の面積を累積和Sとする．
- 以下の過程で，$|A - 2(S + \Delta)|$の最小値を更新していく．
  - $\Delta + S$が$\frac{A}{2}$を超えない
    - jをインクリメントしてSにΔを加算
  - $\Delta + S$が$\frac{A}{2}$を超える
    - Sから三角形(p_i, p_{i+1}, p_j)の面積を引く
    - iをインクリメント（jは変更しない）
    - iが起点に戻っていたら終了
- doubleによる誤差に注意（加算による情報落ちなど）→整数で計算する．
- O(N)

→　**正解！**

問題６　はんぶんこ

解答例

```
int i = 0;
int j = i;
while(1){
  long long dlt = getArea(p[i], p[j], p[(j + 1) % N]);
  long long newSum = sum + dlt;
  ans = min(ans, abs(newSum - (all - newSum)));
  if ( newSum*2 < all  ){
    sum += dlt;
    j = (j + 1) % N;
  } else {
    sum -= getArea(p[i], p[(i + 1) % N], p[j]);
    i = (i + 1) % N;
    if ( i == 0 ) break;
  }
}
```

問題７　旅館の客室番号

概要

- 旅館の客室が番号順に並んでいる.
- 客室番号に４と９の数字は含まれない.
- ２つの客室番号pとqの距離を求めよ.
- １つ隣の部屋に移動する距離を１とする.
- $1 \leqq p < q \leqq 10^{100}$＝10の100乗

問題７　旅館の客室番号

入出力例

```
87327530128536 18076352536182765328716235 6

570627338004098176263624 1
```

組み込みの型（int, long など）は直接使えない

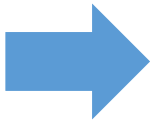
問題７　旅館の客室番号

解法

- ８進数から１０進数への変換
- 多倍長整数

BigIntegerクラス？

K進数
加算
減算
乗算
除算
高速フーリエ変換？



文字列
加算
減算

で十分

問題７　旅館の客室番号

解答例

```
int main(){
  string p, q;
  cin >> p >> q;
  cout << sub(toDecimal(q), toDecimal(p)) << endl;
  return 0;
}
```

問題７　旅館の客室番号

１０進数へ変換

```
string toDecimal(string s){
  string v = "0";
  for (char ch : s){
    if ( ch >= '5' ) ch--;
    v = add(v, v);
    v = add(v, v);
    v = add(v, v);
    v = add(v, string(1, ch));
  }
  return v;
}
```

1 2 3 4 5 6 7 8 9
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
1 2 3 – **4 5 6 7** –

vを８倍する

問題７　旅館の客室番号

加算

```
string add(string a, string b){
  int n = max(a.size(), b.size()) + 1;
  a = zfill(a, n);
  b = zfill(b, n);
  string c = string(n, '0');
  for ( int i = a.size()-1; i >= 1; i-- ){
    int v = (a[i] - '0') + (b[i] - '0') + (c[i] - '0');
    c[i] = ((v % 10) + '0');
    c[i - 1] = ((v / 10) + '0');
  }
  return toNum(c);
}
```

減算

```
string sub(string a, string b){
  int n = max(a.size(), b.size()) + 1;
  a = zfill(a, n);
  b = zfill(b, n);
  string c = string(n, '0');
  int car = 0;
  for ( int i = a.size()-1; i >= 1; i-- ){
    int v = (a[i] - '0') - (b[i] - '0' + car);
    if ( v < 0 ){
      c[i] = (10 + v) + '0';
      car = 1;
    } else {
      c[i] = v + '0';
      car = 0;
    }
  }
  return toNum(c);
}
```

問題８　壊れた出口

概要

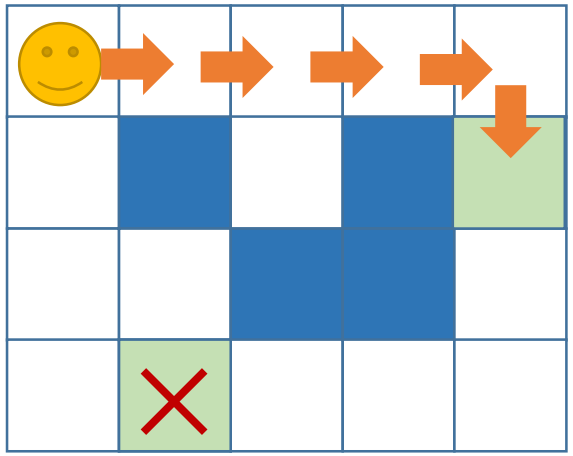
- $N \times M$個のマスからなる迷路が与えられる.
- 各マスは地面（.），出口（E），壁（#）のいずれか.
- ゲームの各プレイで，プレイごとに指定されたマスから出発して出口を目指す.
- 迷路には2つ以上の出口が存在するが，そのうちの一つが壊れている．どの出口が壊れているかはプレイごとに指定される.
- P回プレイを行う．各プレイが終了するまでの最小移動回数を求めよ.
- $2 \leqq N \leqq 1{,}000$，$2 \leqq M \leqq 1{,}000$，$1 \leqq P \leqq 100{,}000$

問題８　壊れた出口

入出力例

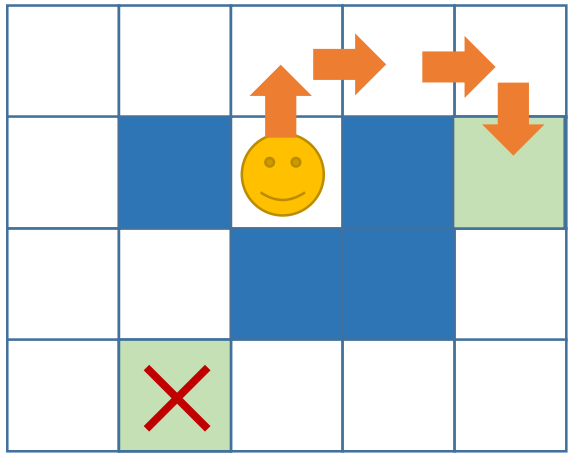
```
4 5
.....
.#.#E
..##.
.E...
3
0 0 3 1
1 2 3 1
1 2 1 4
```

プレイ１



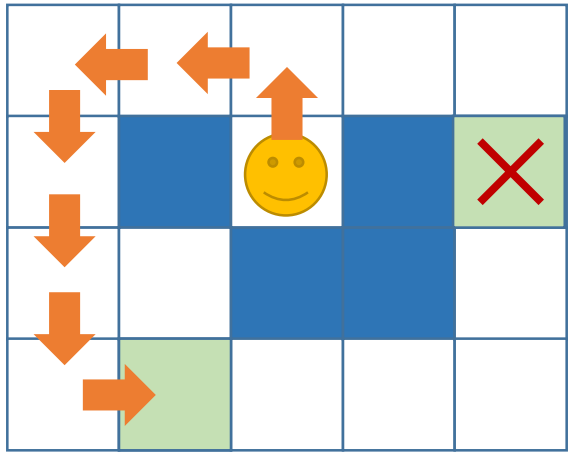
5回

プレイ２



4回

プレイ３



7回

問題８　壊れた出口

素朴なアルゴリズム

- プレイごとに幅優先探索.
- O(NMP)

→　**時間制限**

問題８　壊れた出口

解法

- 出口から幅優先探索を行う．
  - 最初に全ての出口をキューに入れておく
- 各マスについて，最も近い出口への距離と，次に近い出口への距離を計算しておく．
- 各プレイの処理は：
  - 最も近い距離の出口が壊れていなければ，その値を出力する
  - 最も近い距離の出口が壊れていれば，次に近い出口の距離を出力する

- $O(NM + P)$

→　**正解！**

問題９　ロボットアーム

概要

- ロボットアームで，１列に並んだ1からNの番号が割り振られたN個の部品を，それらの番号の小さい順に並び替える.
- ロボットアームは1回の動作で３つの部品を選択し，それらに対して長さ３の巡回置換をかけることができる.
- 部品の列を番号の昇順（恒等置換）にするために必要な，ロボットアームの最小の操作回数を求めよ．ただし、どのように操作を行っても昇順にできない場合は-1を出力せよ。
- $3 \leqq N \leqq 200{,}000$

問題９　ロボットアーム

入出力例

```
6
2 3 4 1 6 5
```

問題９　ロボットアーム

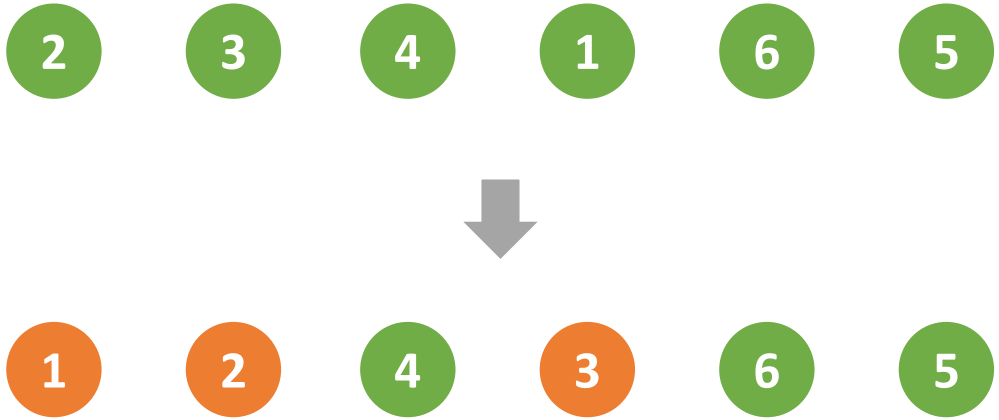
入出力例

```
6
2 3 4 1 6 5
```

問題９　ロボットアーム

入出力例

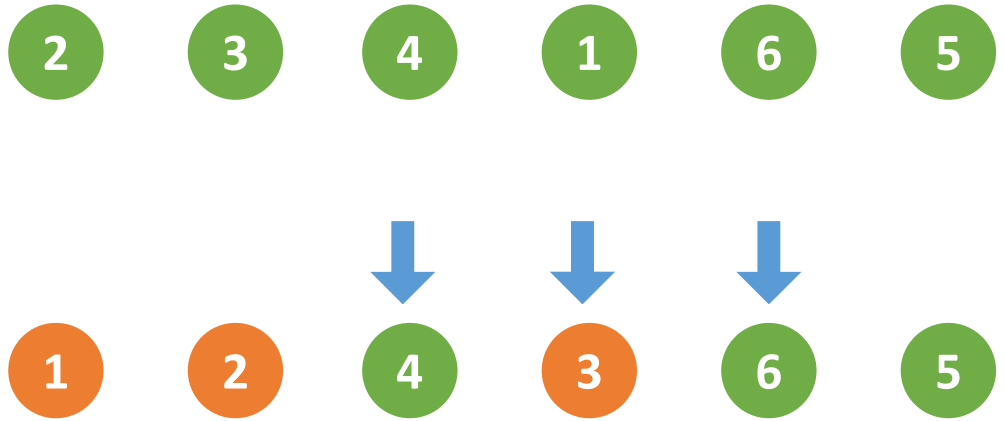
```
6
2 3 4 1 6 5
```



問題９　ロボットアーム

入出力例

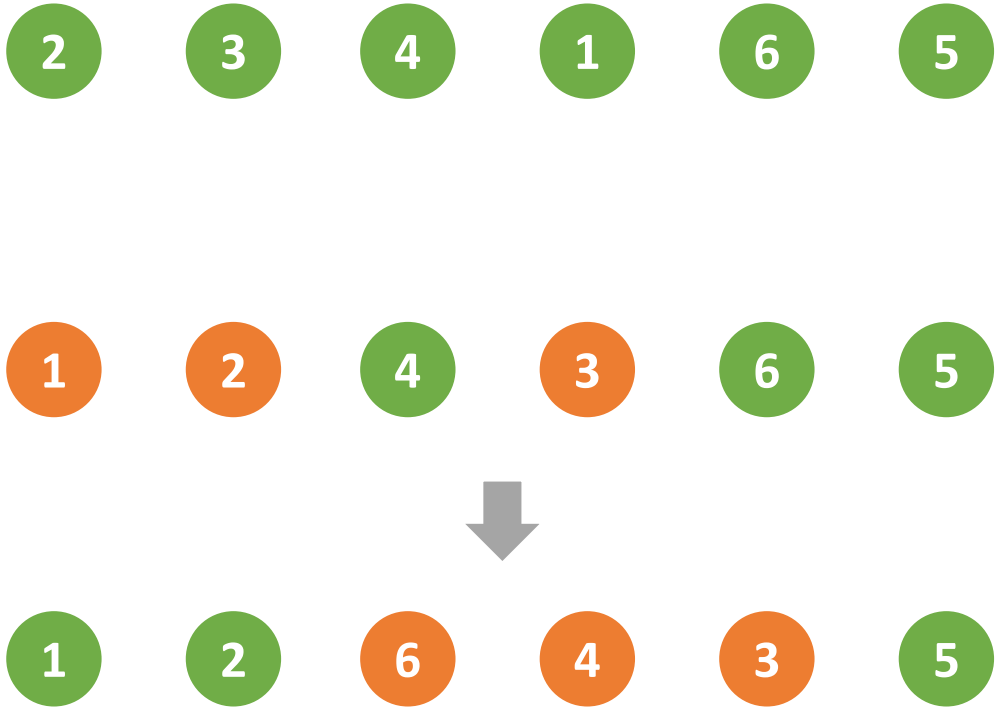
```
6
2 3 4 1 6 5
```



問題９　ロボットアーム

入出力例

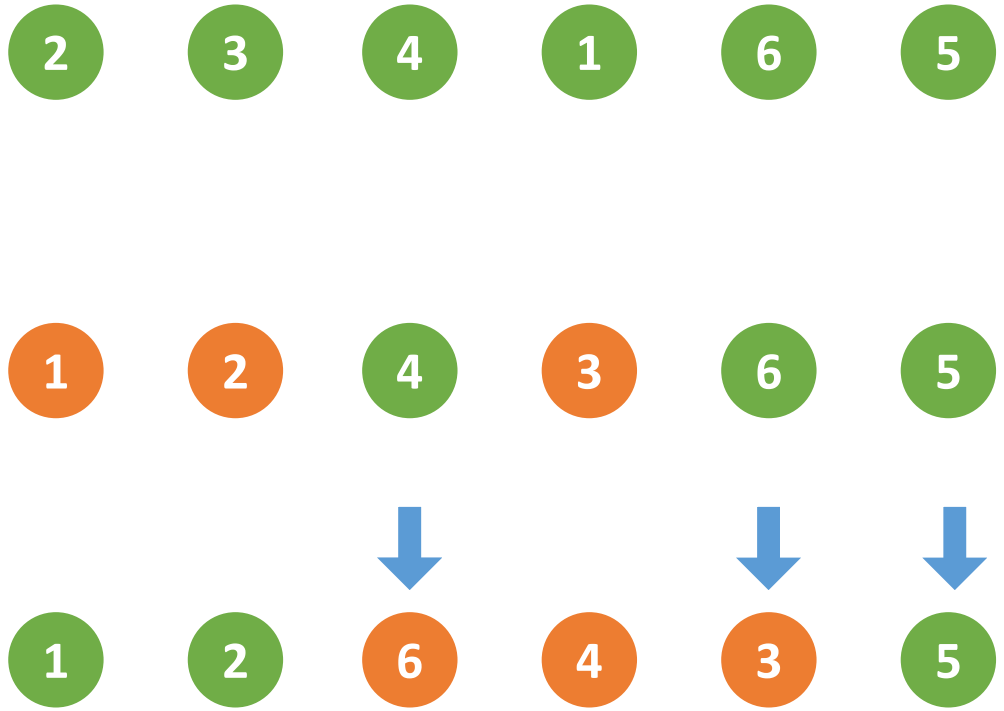
```
6
2 3 4 1 6 5
```



問題９　ロボットアーム

入出力例

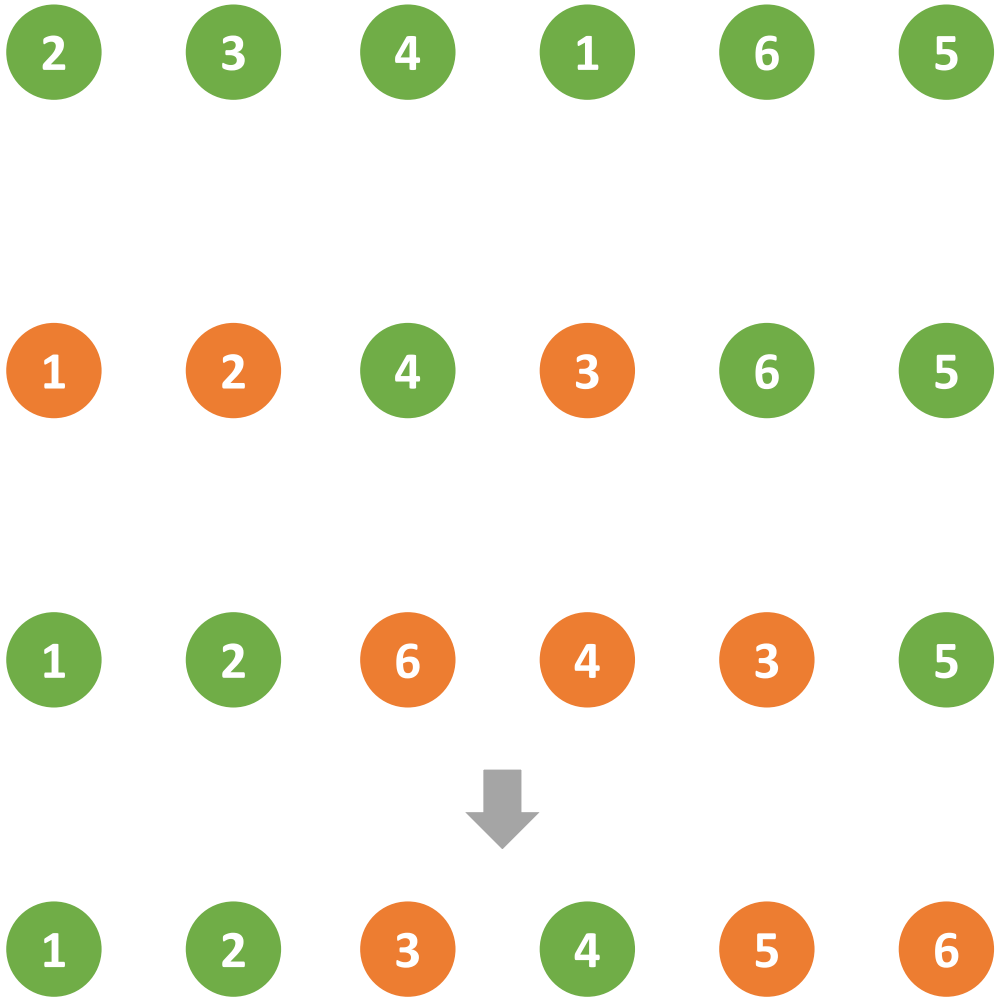
```
6
2 3 4 1 6 5
```



問題９　ロボットアーム

入出力例

```
6
2 3 4 1 6 5
```



問題９　ロボットアーム

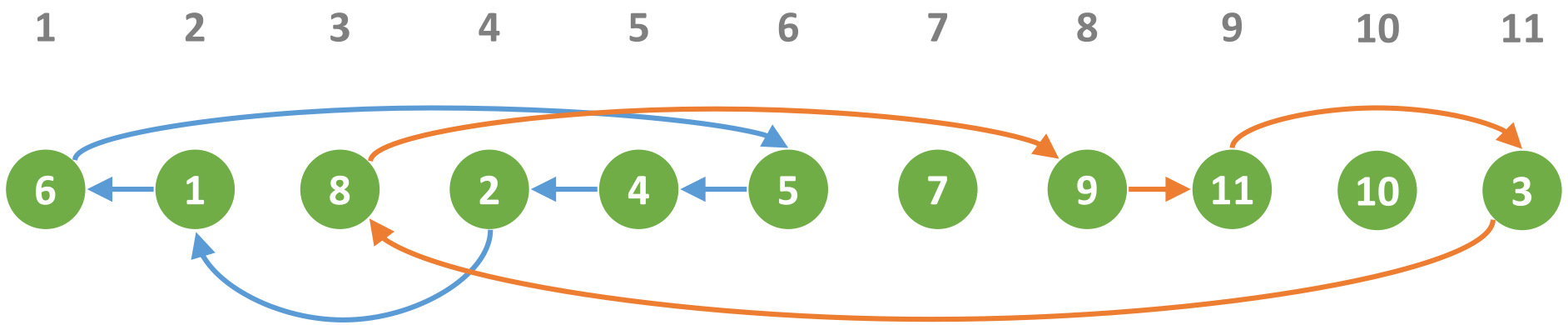
解法

- 与えられた数列をPとする.
- Pを恒等置換にできるかを判定する方法
  - Pの転倒数の偶奇を調べる
  - 偶置換　→　恒等置換にできる
  - 奇置換　→　恒等置換にできない
    - 長さ３の巡回置換では，転倒数を偶数個しか増減できない

問題９　ロボットアーム

解法

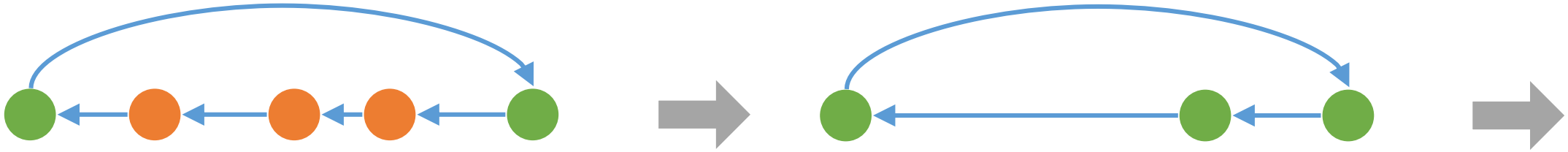
- iからP_iへ辺をはったグラフGを作り，サイクルを検出する．
- サイクルに含まれるノードの数Lが**奇数**の場合と**偶数**の場合を考える．



問題９　ロボットアーム

解法

- サイクルに含まれるノードの数Lが**奇数**の場合
  長さ３の巡回置換1回で２個のノードを削除でき，最後の３つに対して1回の置換をかけて，最小でfloor($\frac{L}{2}$) 回.

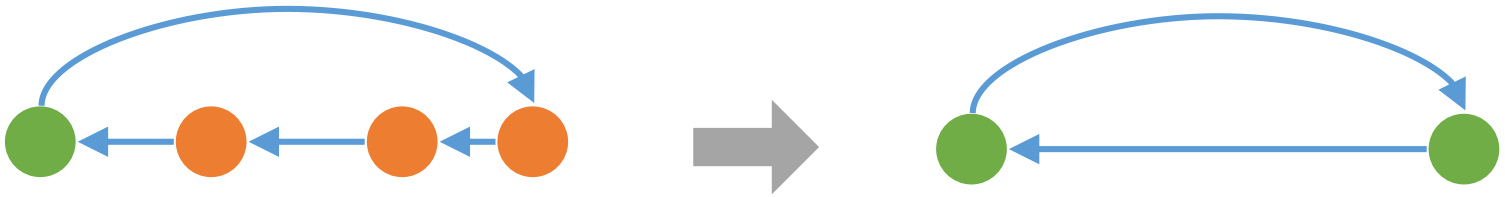


問題９　ロボットアーム

解法

- サイクルに含まれるノードの数Lが**偶数**の場合

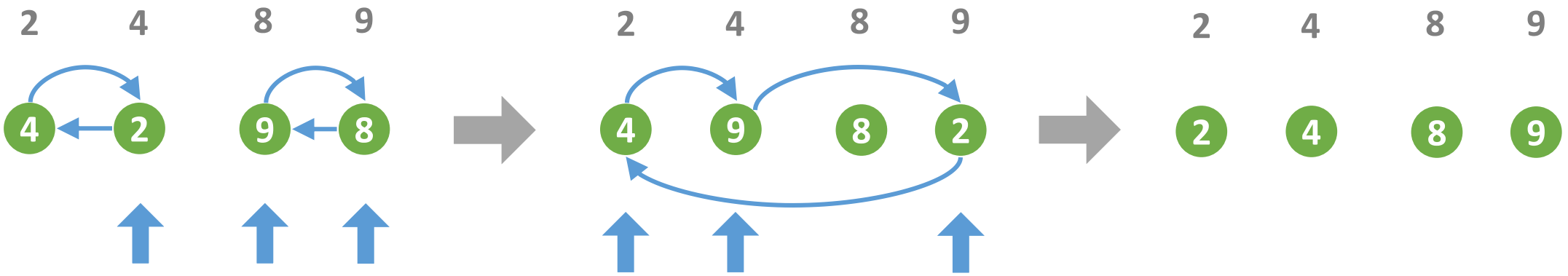
  長さ３の巡回置換1回で2個のノードが減っていくが，２個のノードを含む（転倒数１の）サイクルが残る．



問題９　ロボットアーム

解法

このようなサイクル２つに対しては長さ３の巡回置換２回で整列できるので，Lが偶数の場合もfloor($\frac{L}{2}$)回で数える．

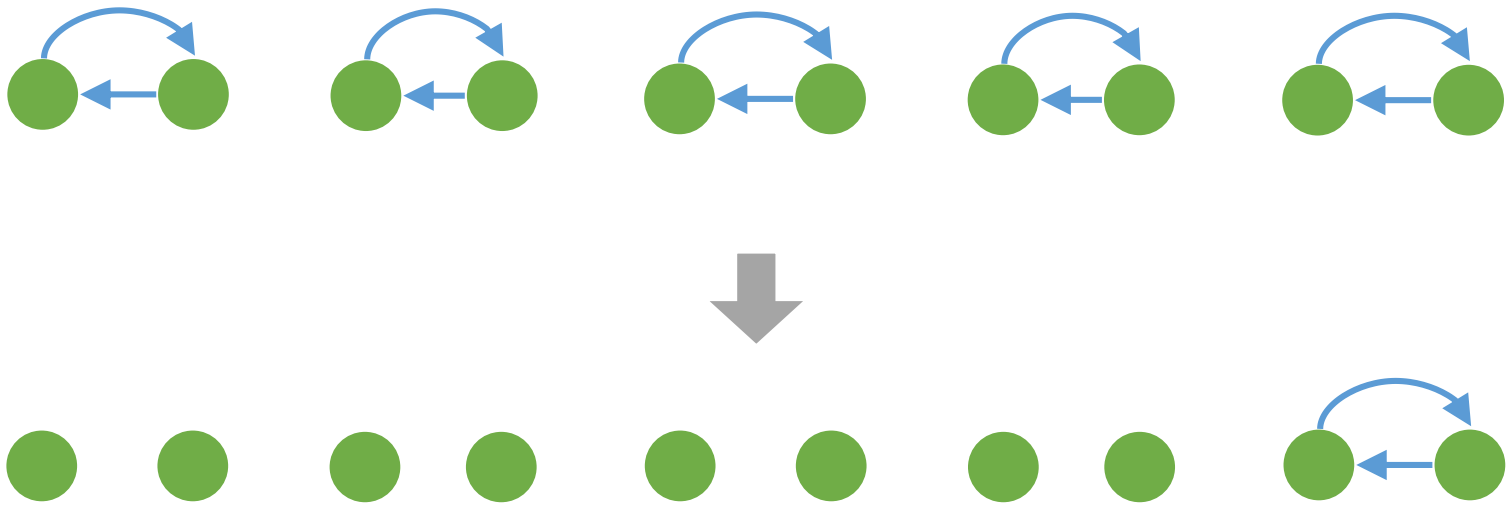


問題９　ロボットアーム

解法

初期状態の転倒数が偶数であれば，漏れなくノード数２のペアを作ることができる．

→　Lが偶数であるサイクルの数が奇数であれば不可能　→　**-1**



問題９　ロボットアーム

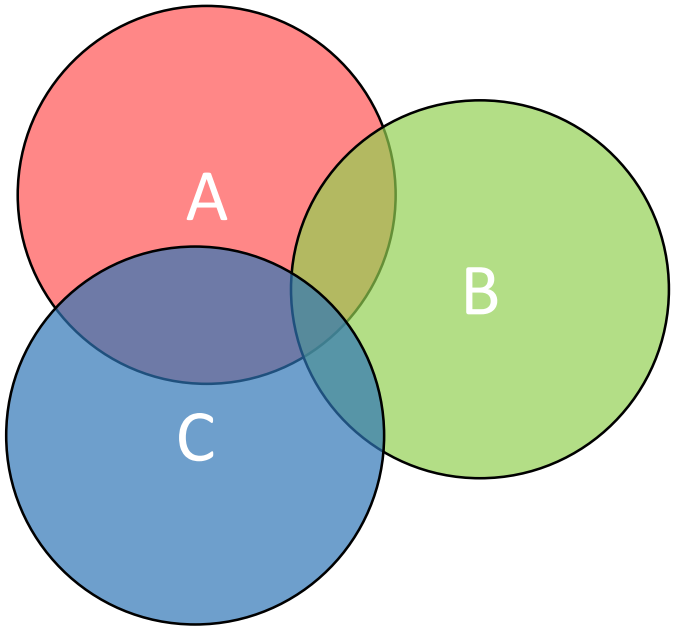
解答例

```
int main(){
  cin >> N;
  for ( int i = 0; i < N; i++ ){
    int p; cin >> p;
    G[p - 1].push_back(i);                    // グラフを作る
  }
  int ans = 0, even = 0;
  for ( int i = 0; i < N; i++ ){
    if ( !vis[i] ){
      int nc = bfs(i);                        // サイクルの中のノード数
      ans += nc / 2;                          // 答えに加算
      if ( nc % 2 == 0 ) even++;              // ノード数が偶数のものを数える
    }
  }
  if ( even % 2 == 0 ) cout << ans << endl;
  else cout << -1 << endl;                    // 偶数の要素をもつコンポーネントが奇数個ある場合NG
}
```

問題１０　柿の実

概要

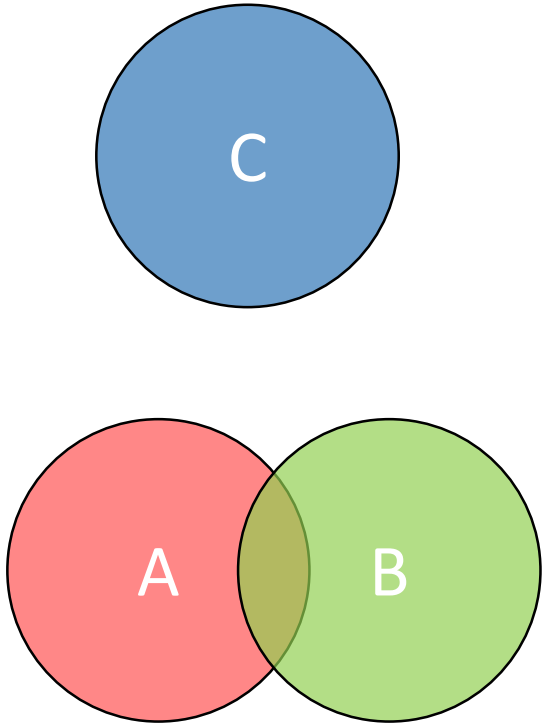
- 円が３つ与えられる.
- ３つの円は半径rが同じで、中心点x_i,y_iは相異なる.
- ３つの円がカバーする面積を求める.
- $1 \leqq r \leqq 500$.
- $0 \leqq x_i, y_i \leqq 500$.
- 誤差がプラスマイナス0.00001を超えてはならない.



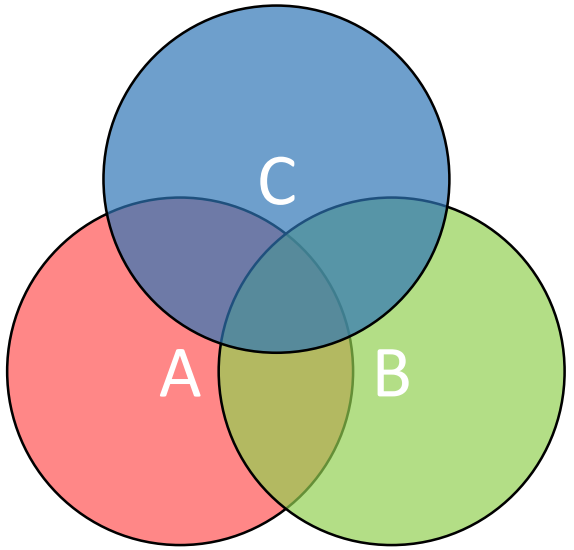
$A \cup B \cup C$を求める.

問題１０　柿の実

入出力例



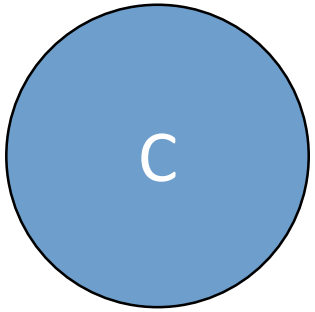
$A + B + C - A \cap B$



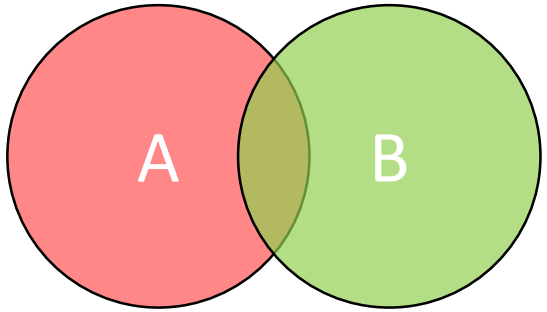
$A + B + C$
$-A \cap B - B \cap C - C \cap A$
$+A \cap B \cap C$

問題１０　柿の実

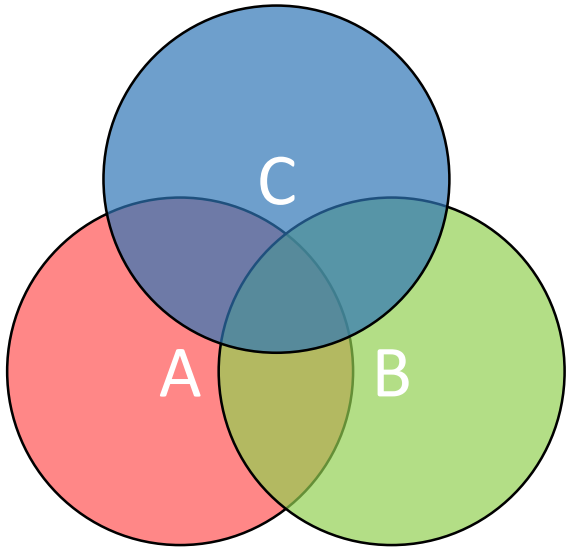
入出力例



$A = B = C = 12.566371$
$A \cap B = 4.913479$



$A + B + C - A \cap B = 32.785633$



$A = B = C = 28.274334$
$A \cap B = 16.500415$
$B \cap C = 10.219895$
$C \cap A = 10.219895$
$A \cap B \cap C = 7.332936$

$A + B + C$
$-A \cap B - B \cap C - C \cap A$
$+A \cap B \cap C = 55.215732$

問題１０　柿の実

解法１

- 面積$A,\ B,\ C,\ A \cap B,\ B \cap C,\ C \cap A,\ A \cap B \cap C$ を求める.
- $A \cup B \cup C = A + B + C - A \cap B - B \cap C - C \cap A + A \cap B \cap C$　（包除原理）
- $A \cap B$などの重なった部分を求めるために場合分けをする.

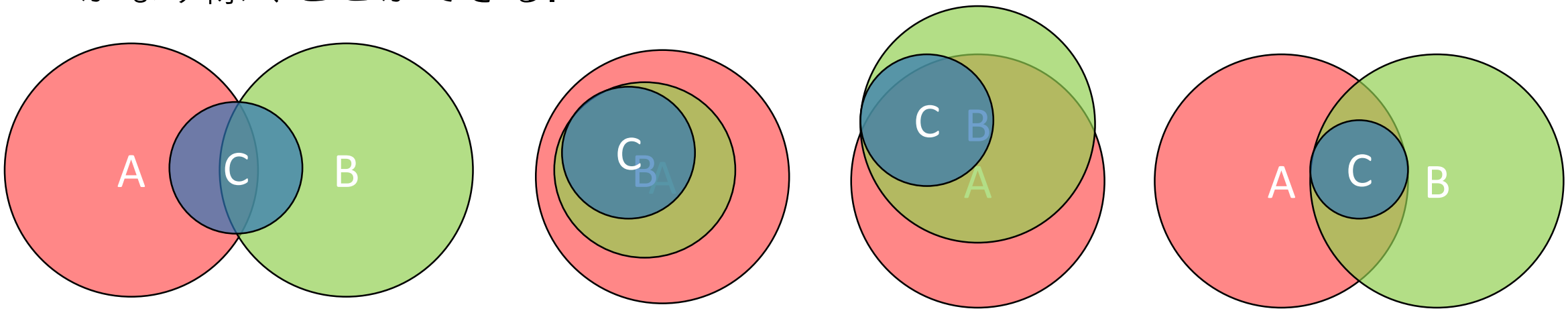
解法２

- 幅$\mathrm{dx} = 10^{-5}$程度の刻み幅でx=-500からx=1000まで走査.
- 各xについて、各円の縦方向の最大値と最小値のうちどれかに含まれる領域を足していく.

問題１０　柿の実

解法１

- ３つの円の半径は同じ＆中心点が同じ座標の円はない.
- ある円が他の円を完全に内包する場合や、二つの円を合わせた中にもう一つが完全に内包されていてかつ交差がある場合など、扱いにくいケースをかなり除くことができる.

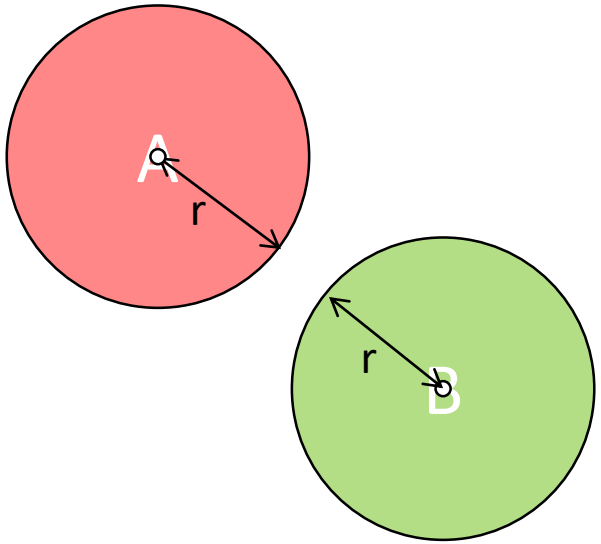


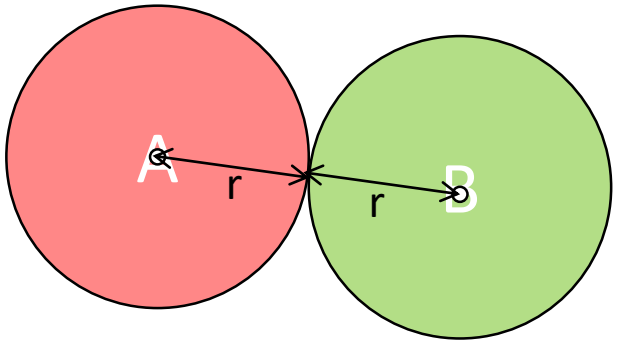
問題１０　柿の実

解法１

- 円の中心同士の距離が2r以上.

  ⇨　$A \cap B = 0.$

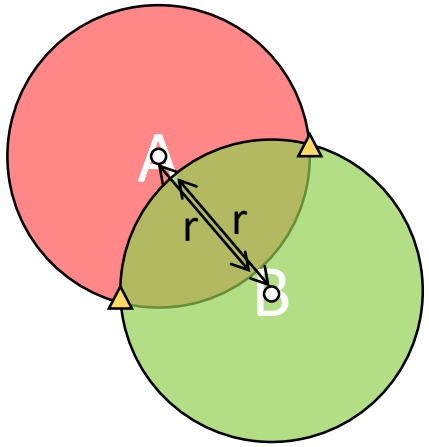




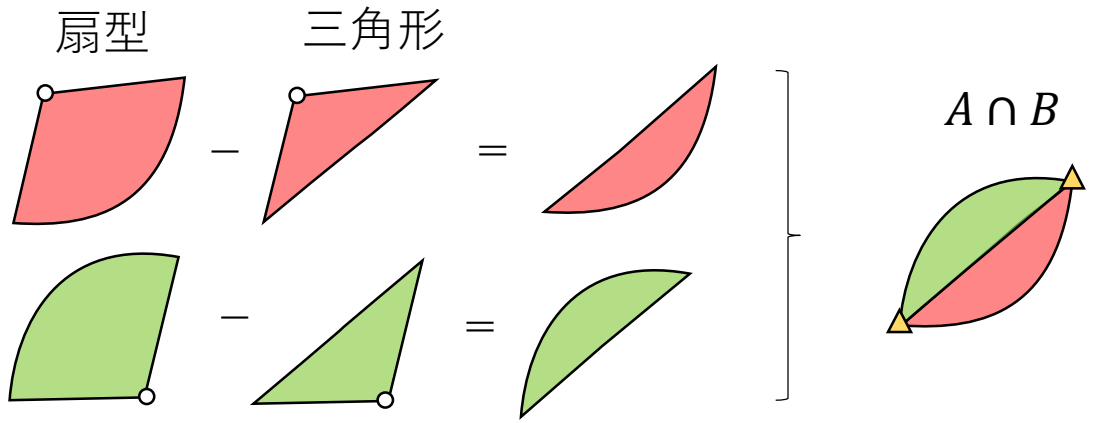
問題１０　柿の実

解法１

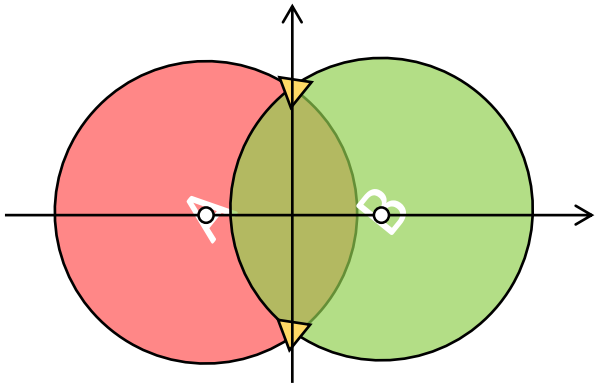
- 円の中心同士の距離が2r未満.
  ⇨　$A \cap B$を求める.



方法１



方法２

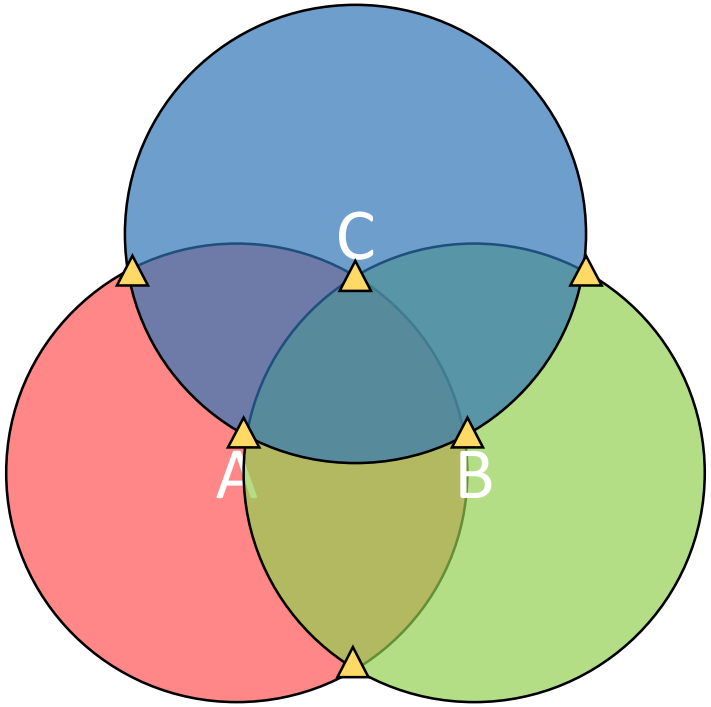


円の中心を結ぶ方向とそれに直交する方向を軸とした座標系に変換すると解析的に求めることができる

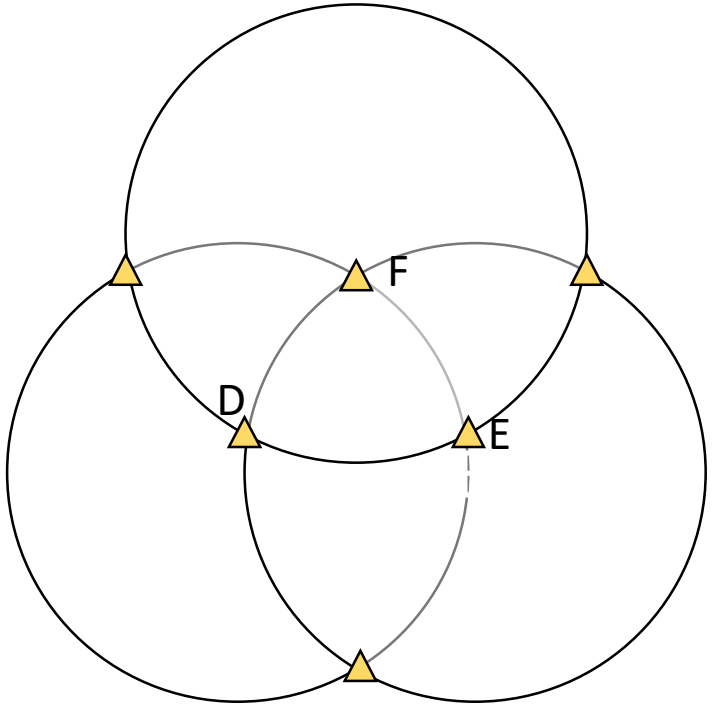
問題１０　柿の実

解法１

- すべての円の中心同士の距離が2r未満.



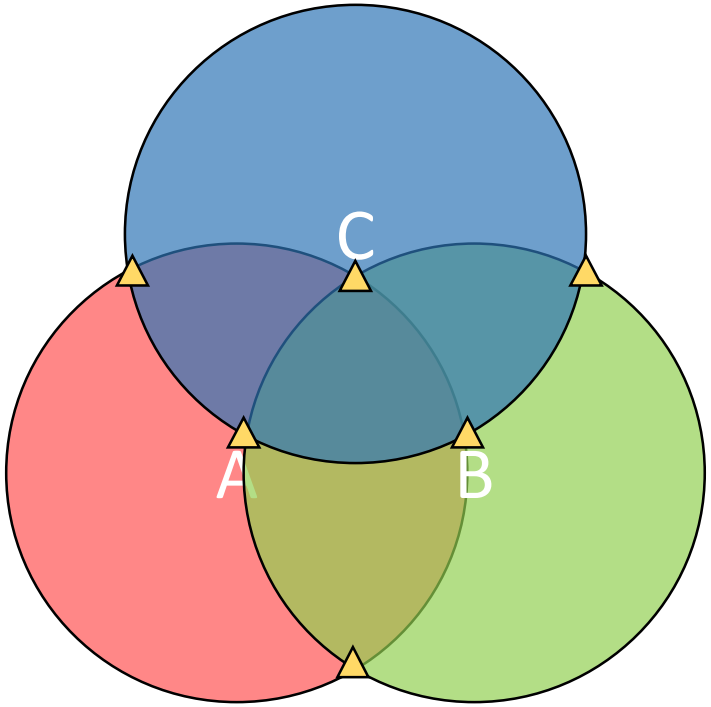
交点のうち、３つの円すべてに含まれる点をD、E、Fとする
(DEFが見つからないような場合は後述).

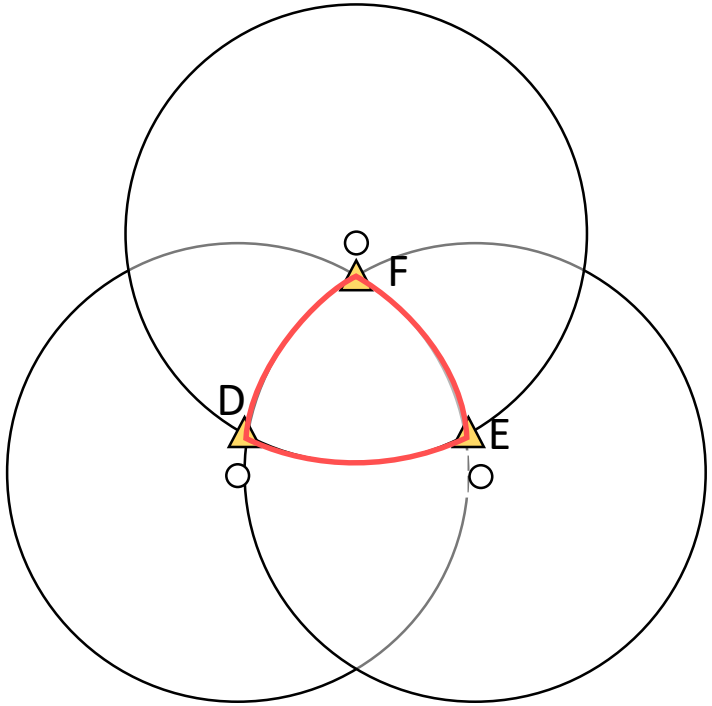


問題１０　柿の実

解法１

$A \cap B \cap C$を求める.

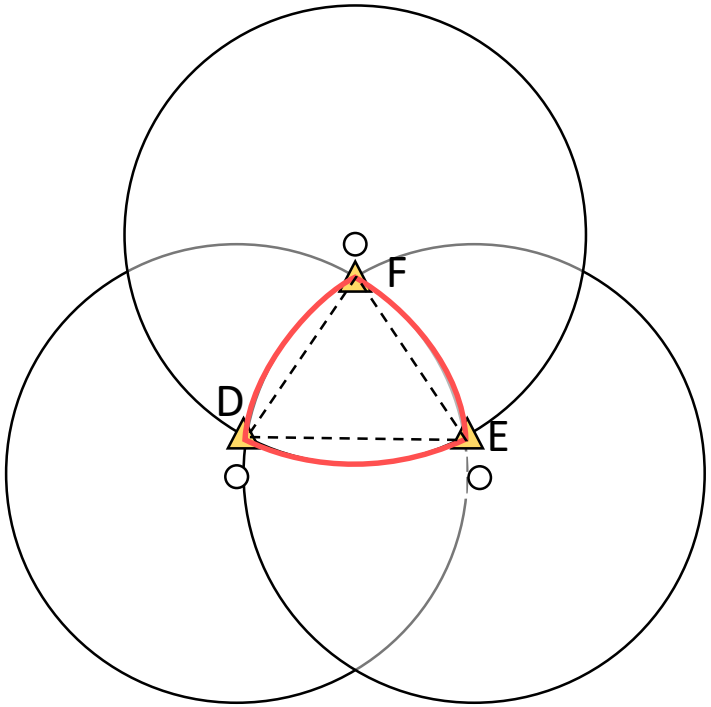




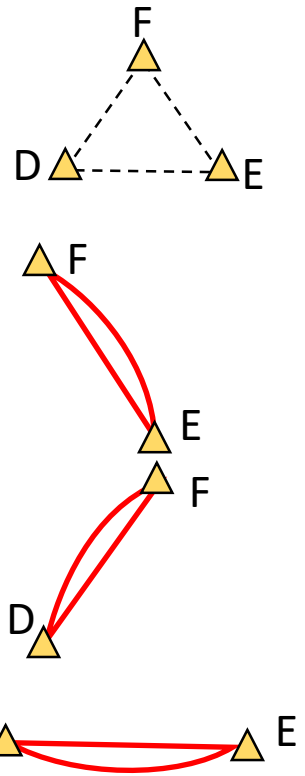
問題１０　柿の実

解法１

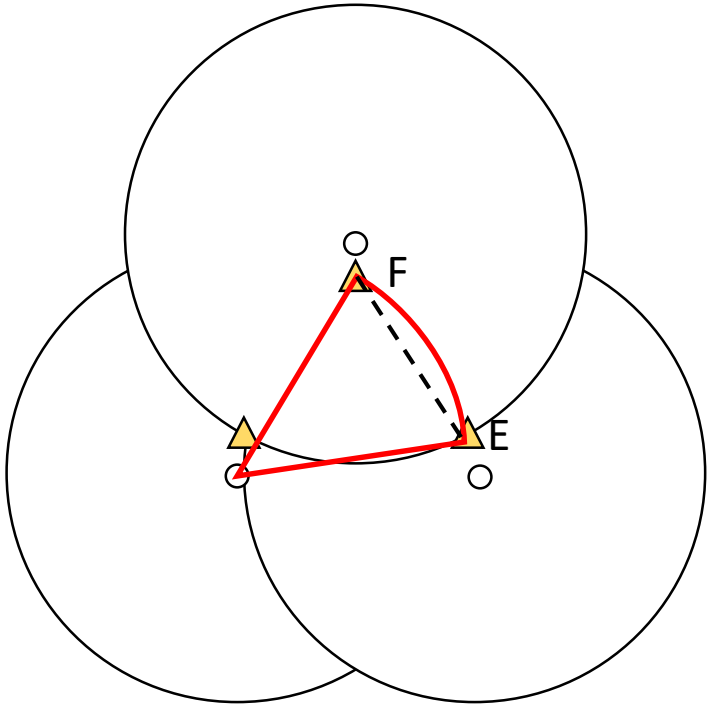
$A \cap B \cap C =$



三角形



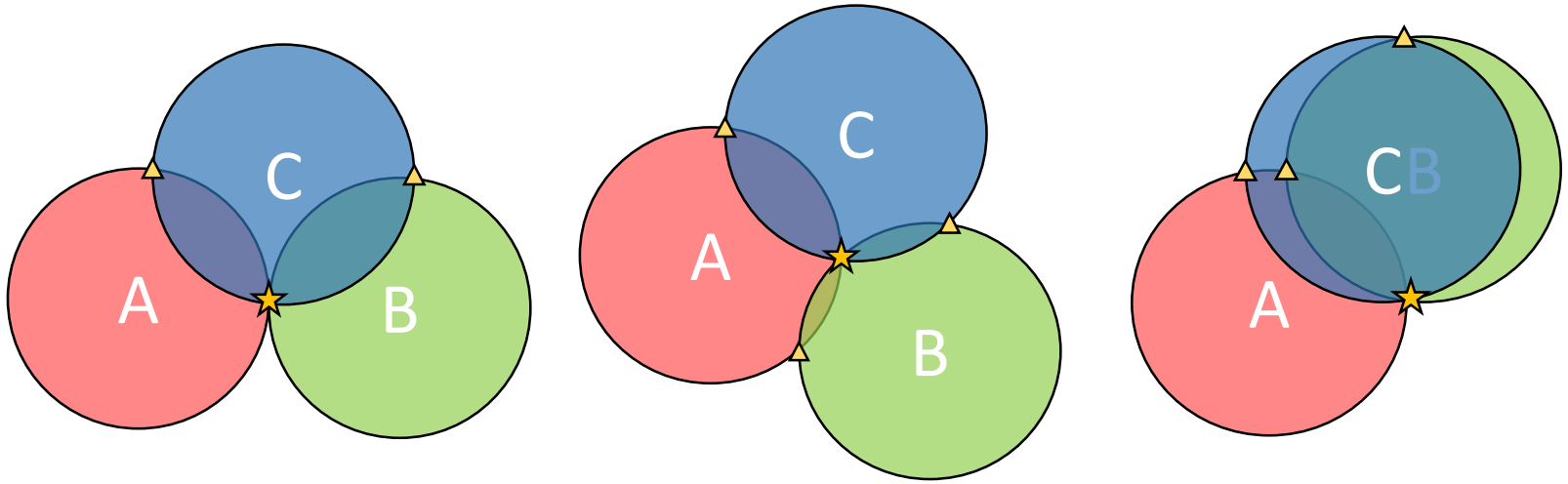
円二つの場合に使った、共通部分を計算する
（扇形から三角形を除く）方法を再利用できる.



問題１０　柿の実

解法１

- DEFが見つからないような場合

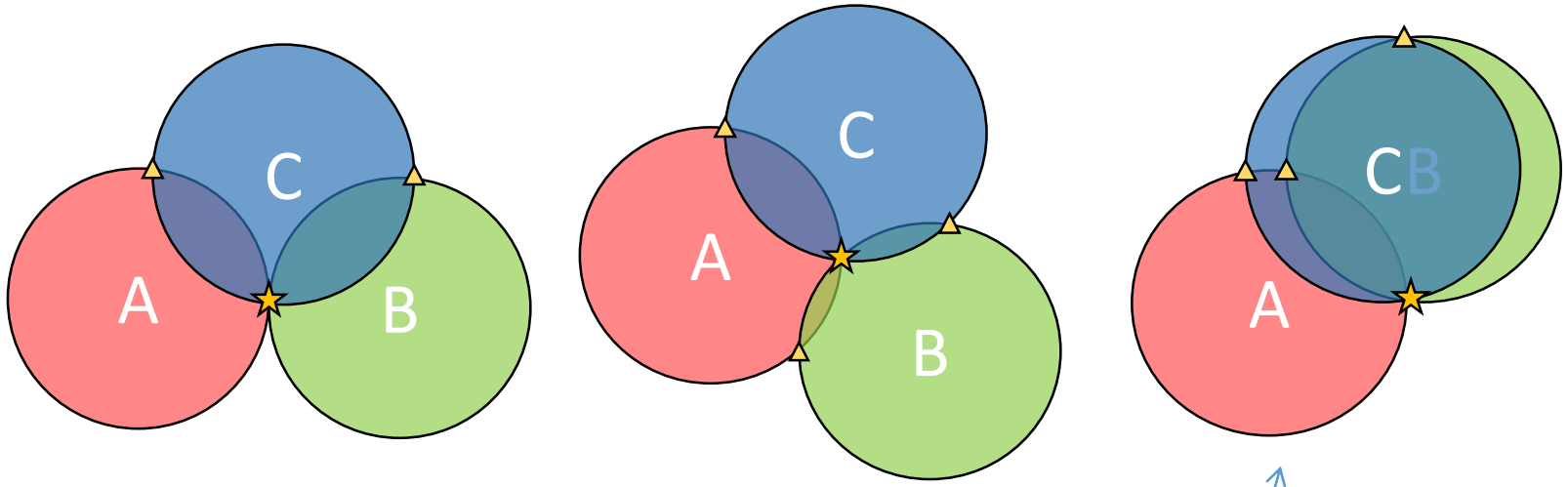


交点が重なる

問題１０　柿の実

解法１

- DEFが見つからないような場合



交点が重なる

⇨　交点座標の重複で検出.　$A \cap B \cap C = 0$とは限らないことに注意.

問題１１　高速道路網の再編

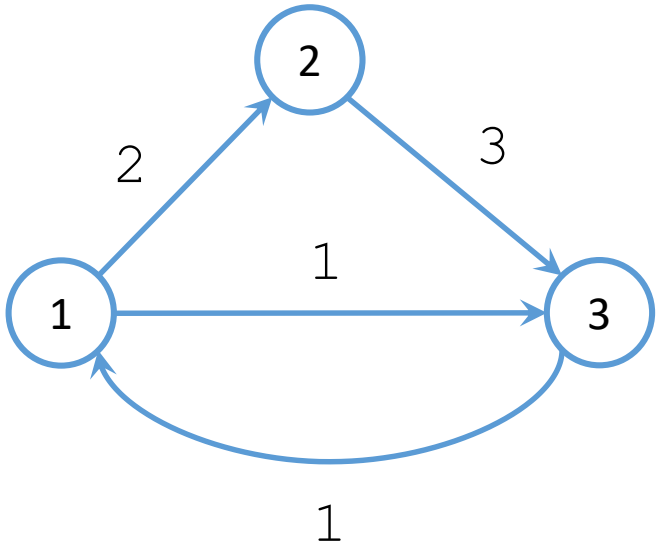
概要

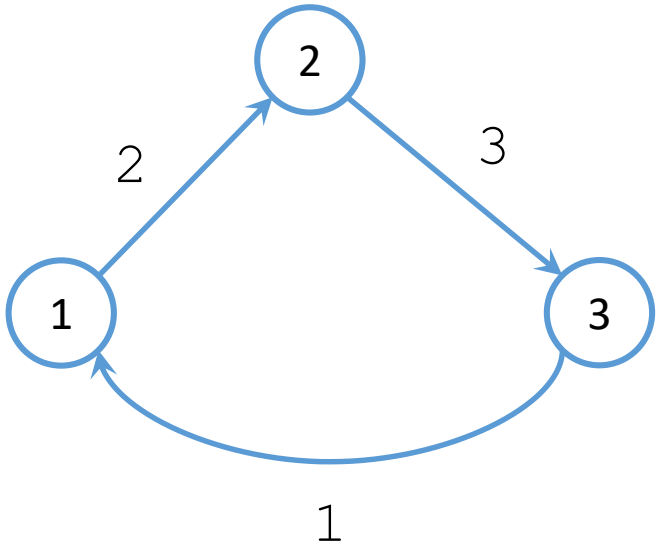
- 高速道路網＝**強連結なグラフ**が与えれる．頂点数N，エッジ数M
  - 1からNの番号が割り当てられたN個の都市（地点）がある．
  - 道路は一方通行で，ある地点とある地点を直接結ぶ道路は多くても２つしかない（２つあるときは互いに逆向き）
  - どの地点からも，他のすべての地点へ必ずたどり着くことができる。
- 高速道路網を維持するには道路ごとにコストc_iがかかる．
- 高速道路網全体の維持コストが最小になるように再編し，その維持コストの最小値を求めよ．
- $2 \leqq N \leqq 13$，$N \leqq M \leqq N(N-1)$，$1 \leqq c_i \leqq 10{,}000$

問題１１　高速道路網の再編

入出力例1

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



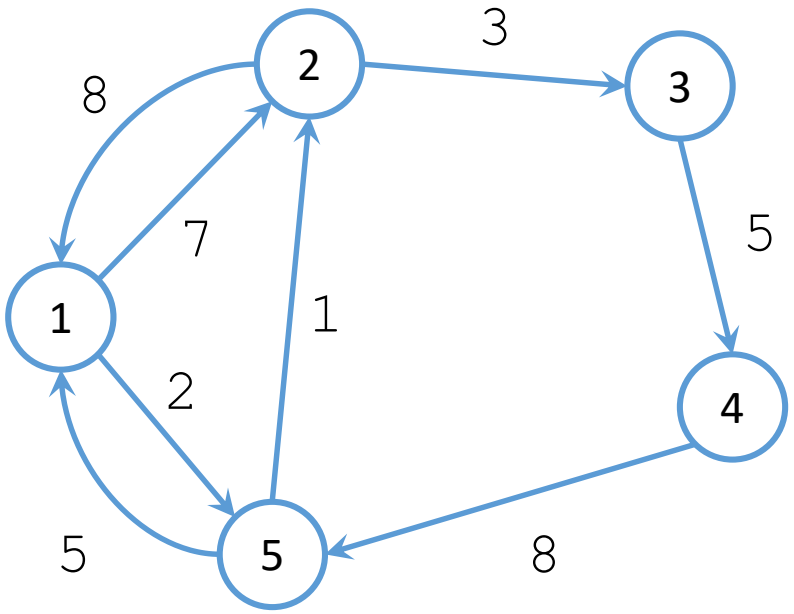


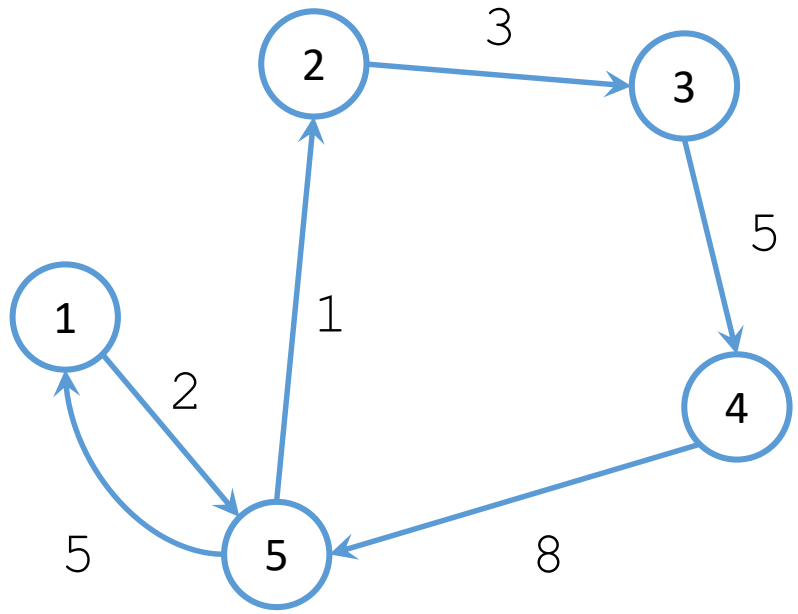

```
6
```

問題１１　高速道路網の再編

入出力例2

```
5 8
1 2 7
1 5 2
2 1 8
2 3 3
3 4 5
4 5 8
5 1 5
5 2 1
```





24

問題１１　高速道路網の再編

解法

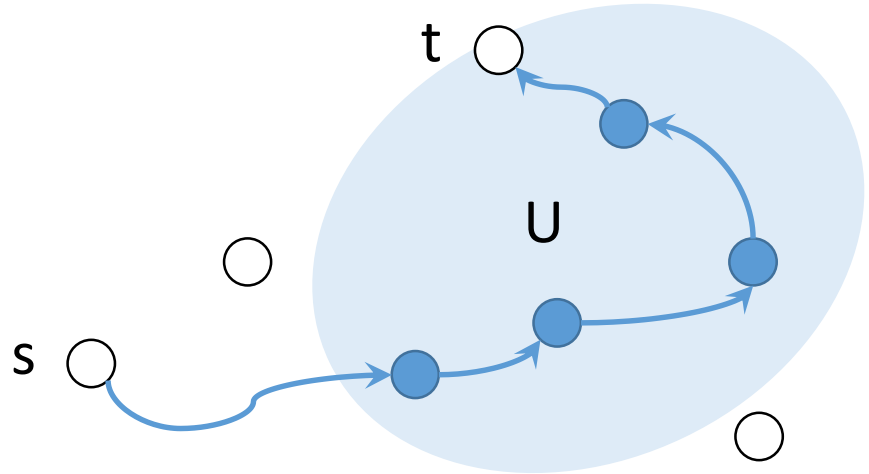
- 動的計画法 (Dynamic Programming) .
  - bitDP-1: ある頂点からある頂点へのパスであって，含まれる頂点の集合がUであるもののコストの最小値を，あらかじめbitDPで求めておく
  - bitDP-2: 頂点1を含む強連結成分の頂点集合Sについて辺のコストの和の最小値をbitDPで求める
- O($N^2 3^N$)

問題１１　高速道路網の再編

bitDP-1

- ある頂点からある頂点へのパスであって，含まれる頂点の集合がUであるもののコストの最小値を，あらかじめ求めておく．

dp1[s][t][U] = 始点sから出発し，最後にtを訪問した，頂点集合Uにおける最小値

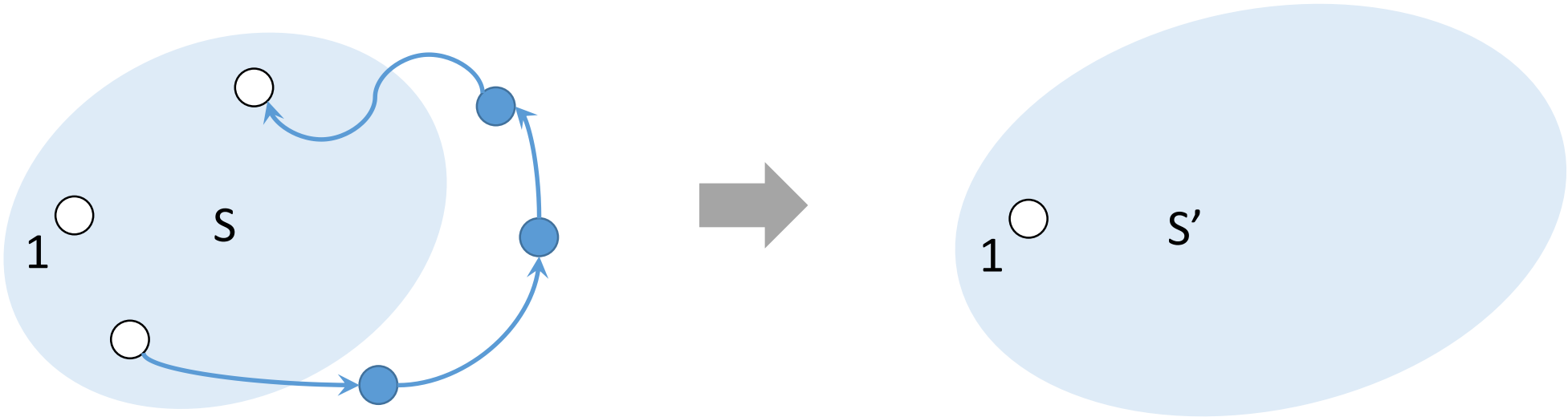


巡回セールスマン問題（TSP）

問題１１　高速道路網の再編

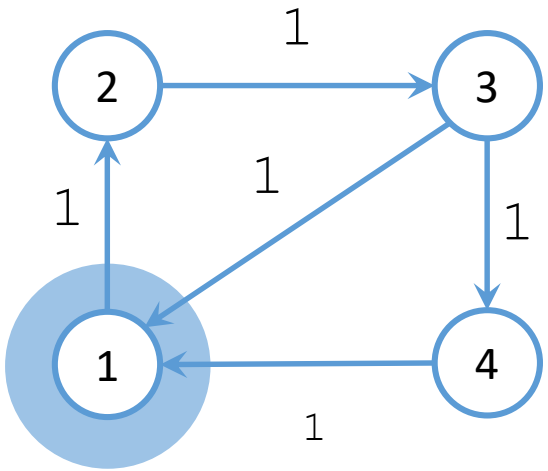
bitDP-2

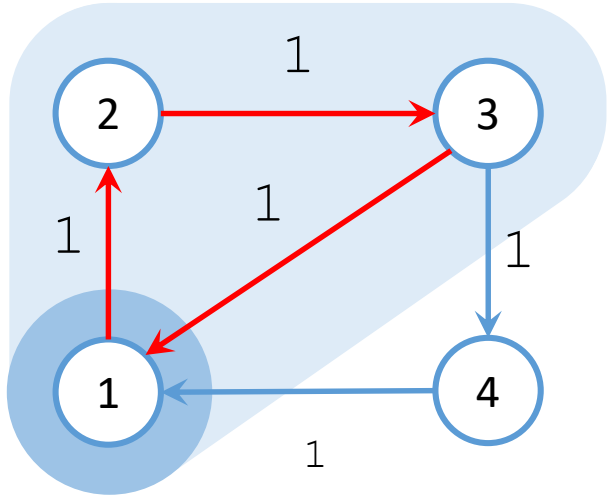
- 頂点1を含む強連結成分の頂点集合を持ってdp.
- dp2[S]=頂点1を含む強連結成分がSであるときの，辺のコストの和の最小値とする.
- 遷移はSの頂点からSの頂点へのパスを追加することで行う(bitDP-1の結果を用いる).

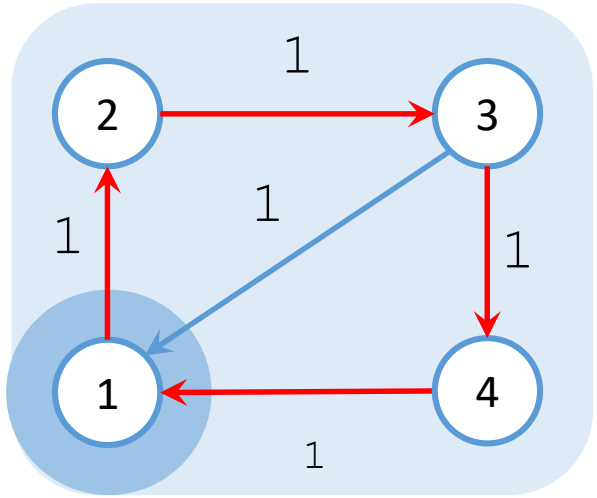


問題１１　高速道路網の再編

1を含むサイクルを列挙すると、(1,2,3),(1,2,3,4)







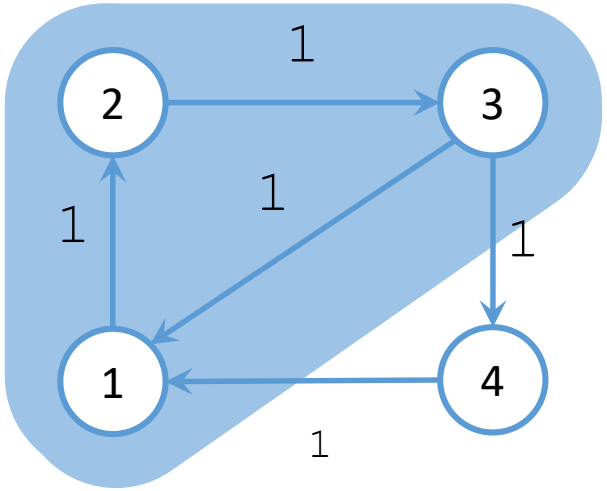
$$dp2[\{1\}] = 0$$

$$dp2[\{1,2,3\}] = \min \begin{cases} dp2[\{1,2,3\}] \\ dp2[\{1\}] + 3 \end{cases}$$

$$dp2[\{1,2,3,4\}] = \min \begin{cases} dp2[\{1,2,3,4\}] \\ dp2[\{1\}] + 4 \end{cases}$$

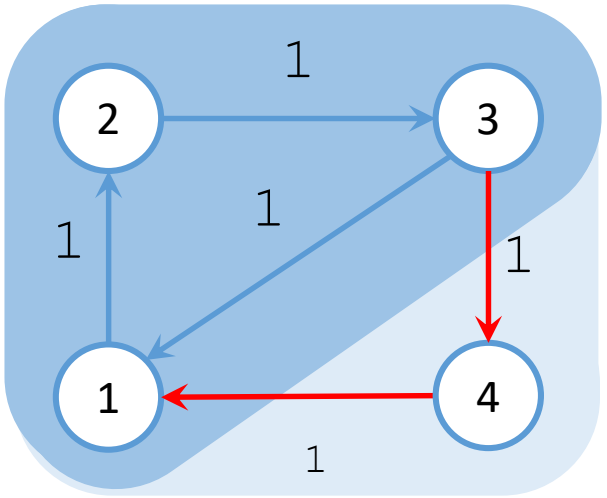
問題１１　高速道路網の再編

{1,2,3}からの遷移を考える



{1,2,3}を縮約したときのサイクル、

つまり{1,2,3}から{1,2,3}へのパスは(3,4,1)の一つ.



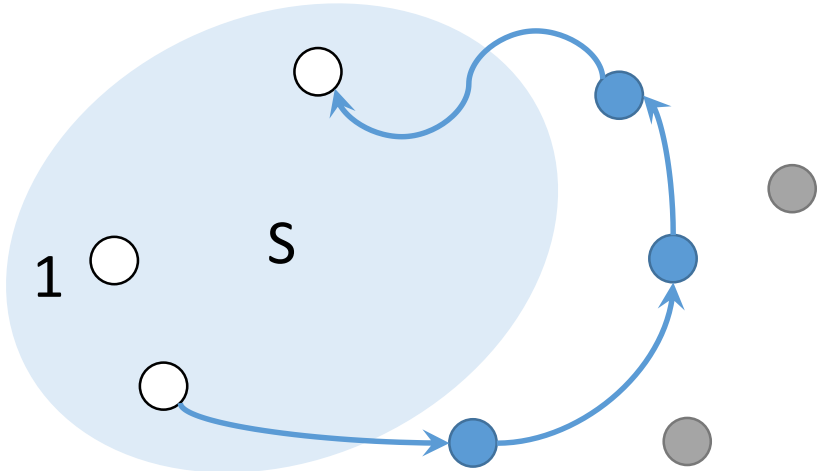
$$dp2[\{1,2,3,4\}] = \min \begin{cases} dp2[\{1,2,3,4\}] \\ dp2[\{1,2,3\}] + 2 \end{cases}$$

問題１１　高速道路網の再編

解答例

```
for ( int s = 0; s < N; s++ ){
   for ( int v : G[s] ) dp1[s][v][0] = W[s][v];
   for ( int e = 0; e < (1 << N); e++ ){
     for ( int u = 0; u < N; u++ ){
       if ( dp1[s][u][e] == INF ) continue;
       for ( int v : G[u] ){
         dp1[s][v][e | (1 << u)] = min(dp1[s][v][e | (1 << u)], dp1[s][u][e] + W[u][v]);
       }
     }
   }
 }
```

問題１１　高速道路網の再編



解答例

```
dp2[1] = 0;

for ( int e = 0; e < (1 << N); e++ ){
  if ( dp2[e] == INF ) continue;
  for ( int f = 0; f < (1 << N); f++ ){
    if ( (e & f) > 0 ) continue;
    for ( int a = 0; a < N; a++ ){
      for ( int b = 0; b < N; b++ ){
        if ( (e & (1 << a))  == 0 || (e & (1 << b)) == 0 ) continue;
        dp2[e + f] = min(dp2[e + f], dp2[e] + dp1[a][b][f]);
      }
    }
  }
}
cout << dp2[(1 << N) -1] << endl;
```

$O(3^N)$

$O(N^2)$

問題１２　交換と転倒数

概要

- 長さNの数列$a_1, a_2, \ldots, a_N$,とM個のクエリが与えられる.
- 各クエリについて，数列中の指定された２つの要素の値を交換し，交換後の数列の転倒数を出力する.
- ２回目以降の各操作は，その直前の操作の結果得られた数列に対して行う.
- $2 \leqq N \leqq 500{,}000$，$0 \leqq a_i \leqq 1{,}000{,}000{,}000$，$1 \leqq M \leqq 30{,}000$
- すべての要素の値は異なる.

問題１２　交換と転倒数

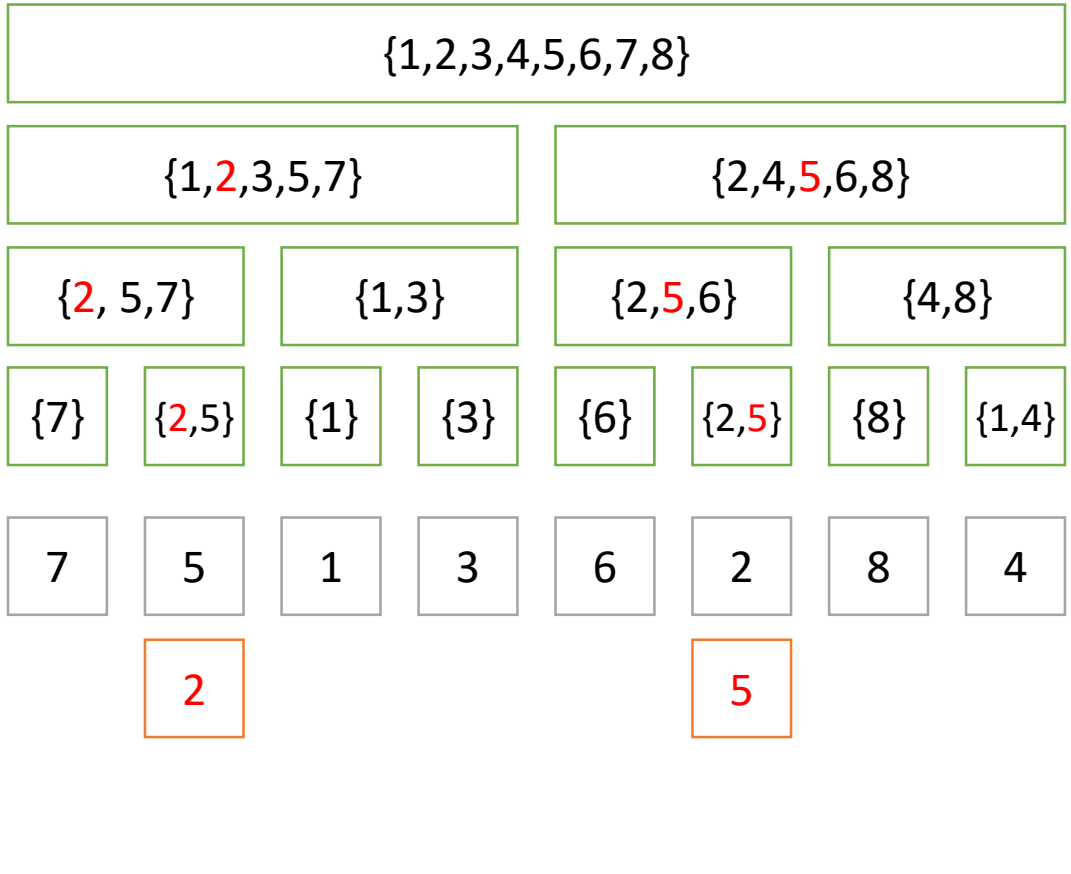
解法

- O($(N+M)\,log(N+M)+Mlog^2(N+M)$)　→　**正解**
- O($(N+M)log^2(N+M)$)　→　**正解**（定数によっては時間制限）
- O($N\,log\,N+M\sqrt{N\log N}$)　→　**正解**（定数・メモリによっては不正解）
- etc.

- O($N\,log\,N+MN$)　→　時間制限
- O($N^2+\ MN$)　→　時間制限
- etc.

問題１２　交換と転倒数

セグメント木に平衡二分木をのせる



- 各ノードに，対応する区間に1度でも現れるような要素の「集合」を持つ.
- t番目までのクエリを処理したときのi番目のノードの集合を$S_{t(i)}$とする.
- 初期状態：$S_{0(i)} = \{\}$
- t番目のクエリで位置kにvを追加する：

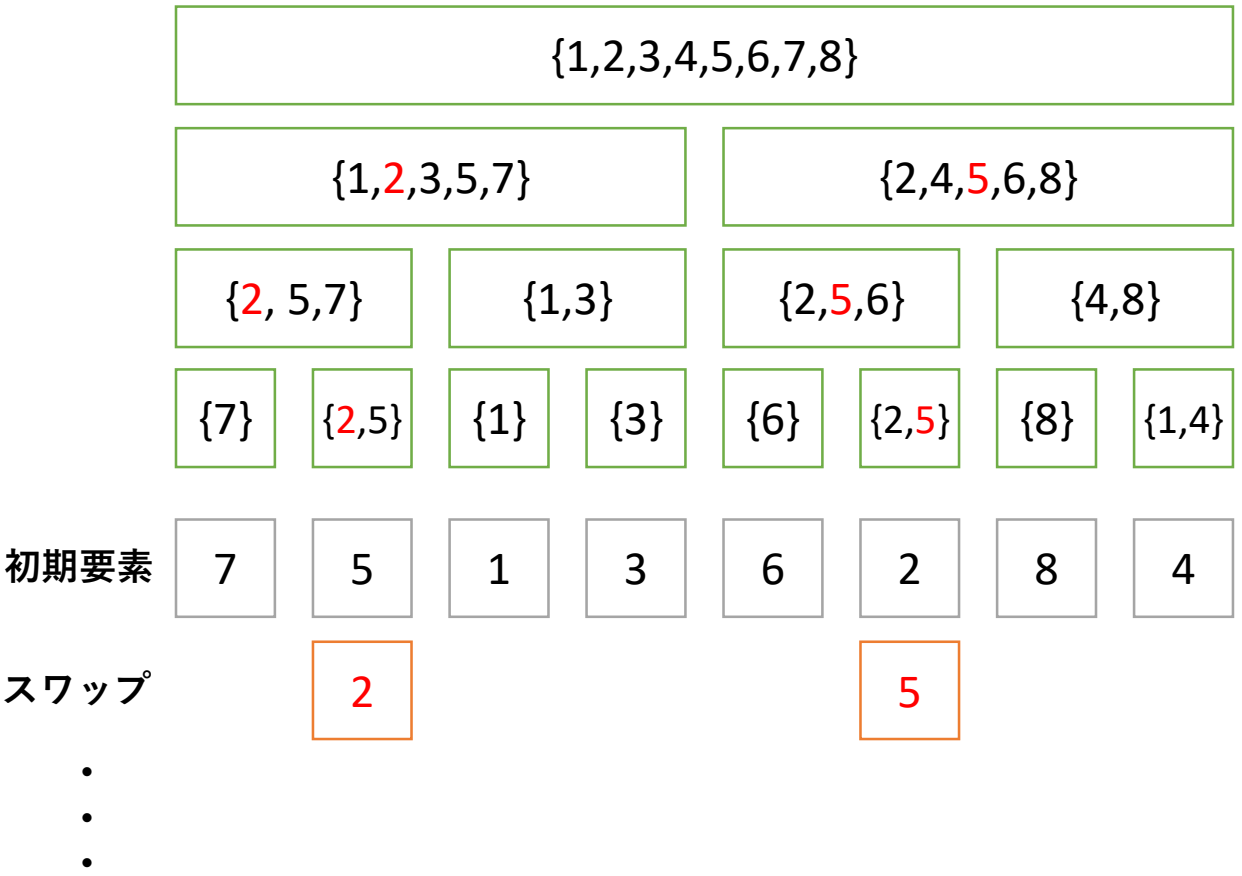
```
k += 1 << L;
while(k){
  S[k].insert(v); // Sはstd::set, 平衡二分木など
  k >>= 1;
}
```

- 時間：$O((N+M)log^2(N+M))$ **定数倍が重い**
- 空間：$O((N+M)logN)$

$$\sum S_{(N+M)(i)} \leqq (N+M)log(N+M)$$

問題１２　交換と転倒数

セグメント木にBIT (Binary Index Tree) をのせる



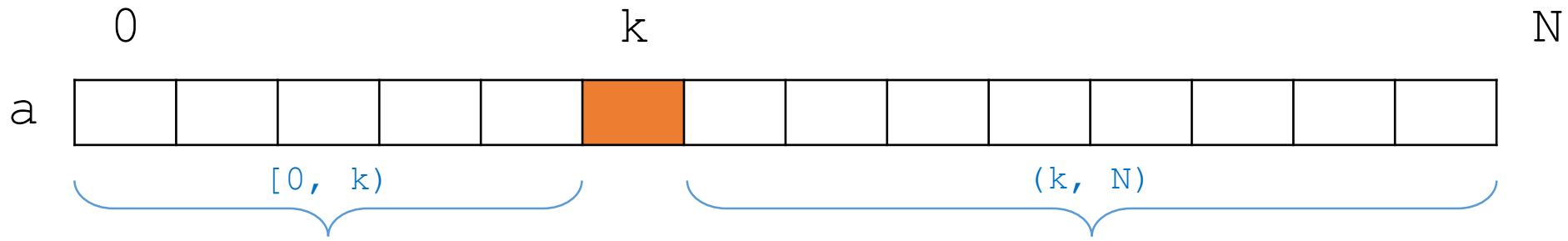
- クエリを先読みすることで，$S_{(N+M)(i)}$をあらかじめ求めておくことができる．
- $S_{(N+M)(i)}$を座標圧縮し，各ノードにサイズ$\left|S_{(N+M)(i)}\right|$のBITを持たせる．これを$B(i)$とする．
- クエリは以下のように書き換えられる．

```
k += 1 << L;
while(k){
  idx = lower_bound(S[k].begin(), S[k].end())と先頭の距離
  // Sはstd::vector
  B[i].add(idx, 1)
  k >>= 1;
}
```

- 時間：$O((N+M)log^2(N+M))$　**定数倍が軽い**
- 空間：$O((N+M)logN)$

問題１２　交換と転倒数

転倒数の計算: 一か所のa_kを変更した場合

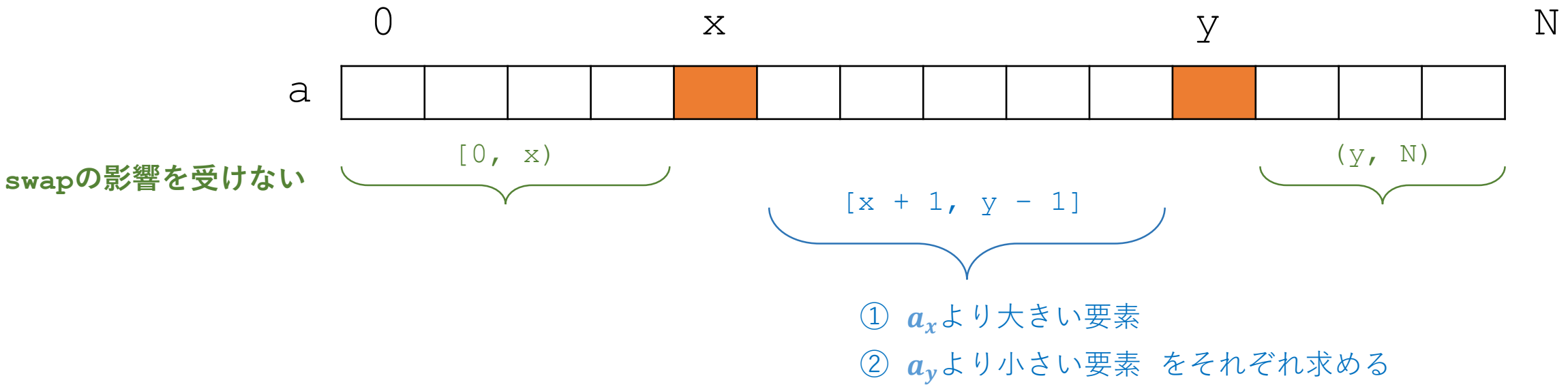


転倒数に寄与する要素　　a_kより大きい要素　　a_kより小さい要素

- 書き換え前の個数を引き，書き換え後の個数を足すことで書き換え後の転倒数を求めることができる.
- 2点swapは（一か所書き換え）×2として処理できる.　　→（定数倍軽くすることが可能）

問題１２　交換と転倒数

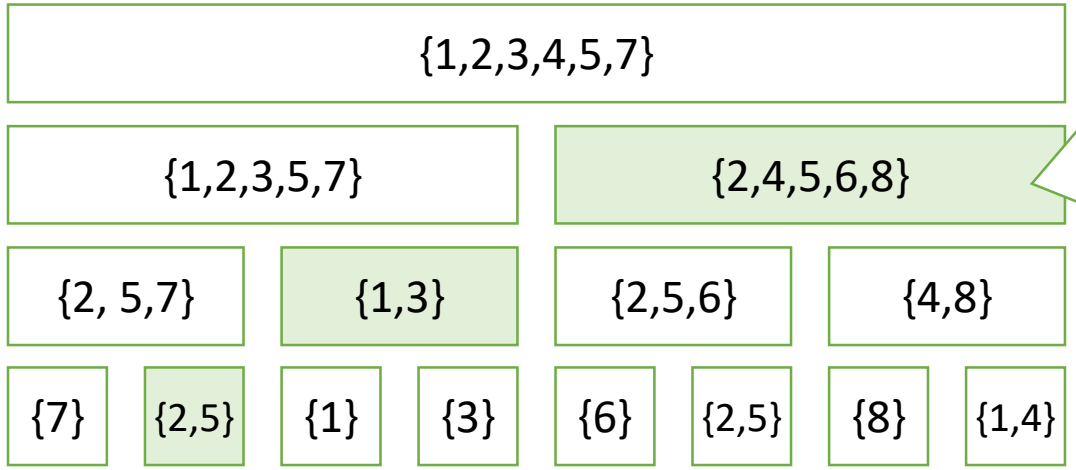
転倒数の計算: 2点swap処理の定数倍高速化

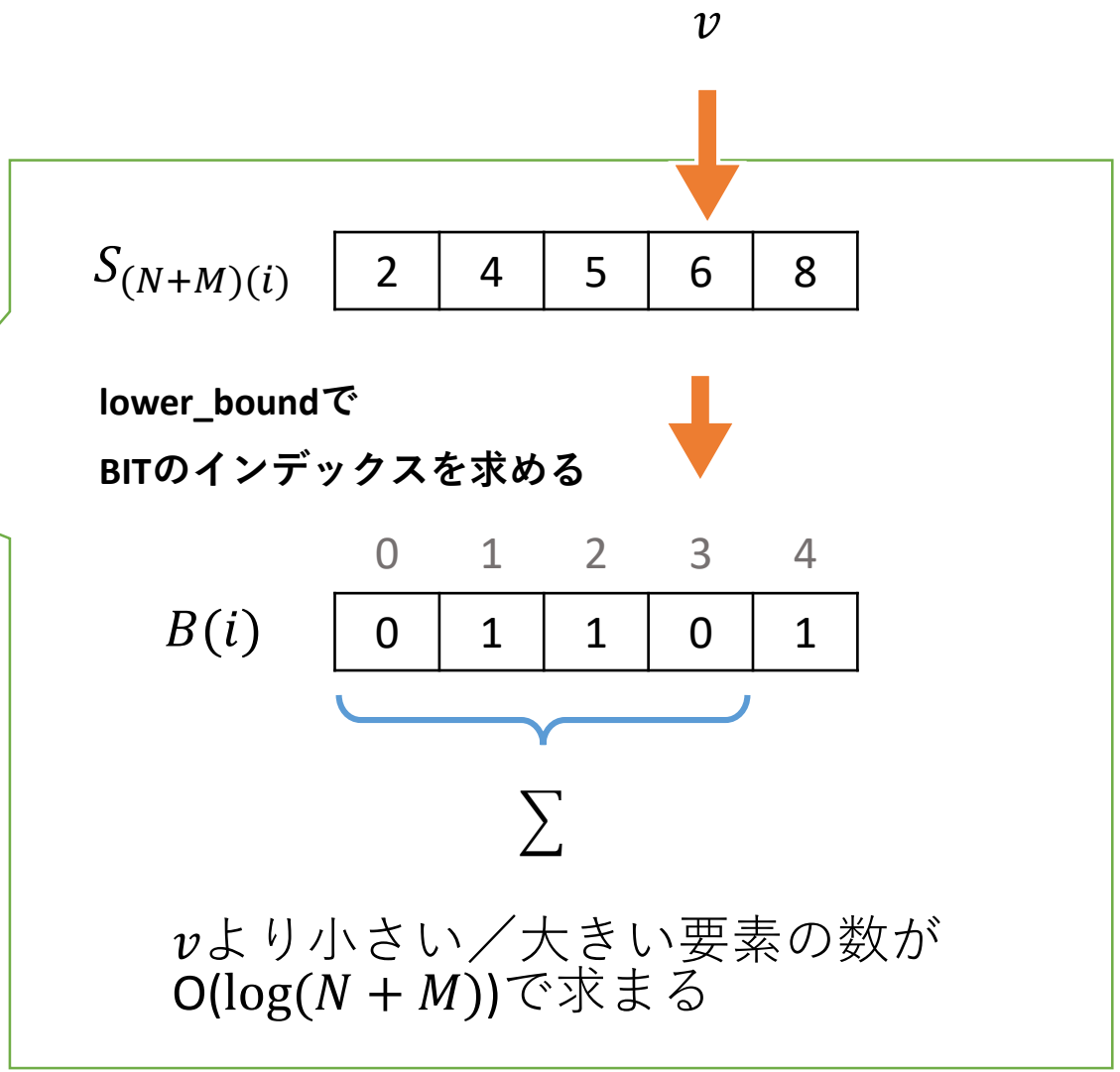


- 「すべての要素が相異なる」という制約から①の個数と②の個数の和は$y - x - 1$に等しいため，片方が求まればもう片方はO(1)で求まる（さらなる定数倍高速化）．

問題１２　交換と転倒数

区間に対する要素の数え上げ





問題１３　山小屋スタンプラリー

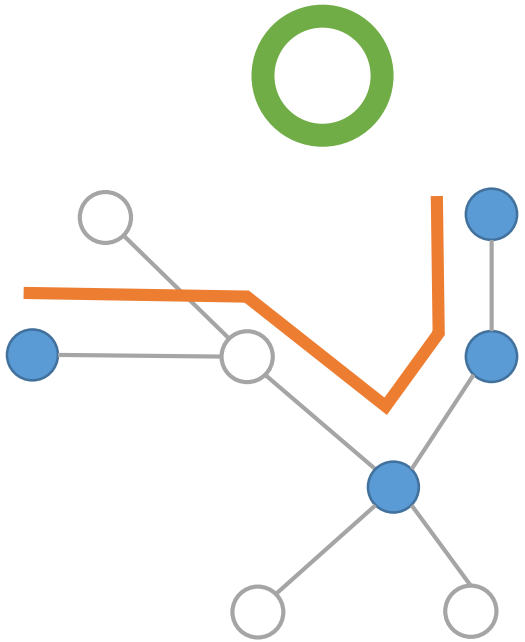
概要

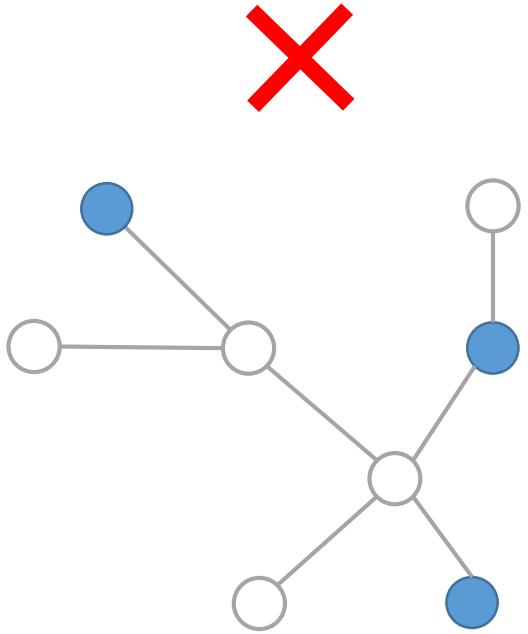
- N棟の山小屋と各山小屋をつなぐ道が与えられる．→　N頂点の木
- すべての山小屋にスタンプを設置して，スタンプラリーをする．
  - まず山小屋を２つ以上選ぶ．その後，選んだ山小屋のどれか１つに移動し，そこからスタンプラリーを開始する．
  - スタンプラリーの間は，道を通って山小屋の間を移動する．移動中に通った山小屋すべてを訪れ，そこでスタンプを押す．
  - 既にスタンプを押した山小屋を訪れると，その時点でスタンプラリーは失敗となり終了する．
  - 選んだ山小屋をすべて訪れた時点で，スタンプラリーが成功する．
- スタンプラリーを成功させるたには，山小屋を巡るルートを正しく選ぶ必要がある．しかし，山小屋の選び方によっては，どのようにルートを選んでも成功できなくなってしまう．
- スタンプラリーを成功させることができるような山小屋の選び方は何通りあるか？選んだ山小屋の数ごとに、山小屋の選び方の数を求めよ．

問題１３　山小屋スタンプラリー

概要

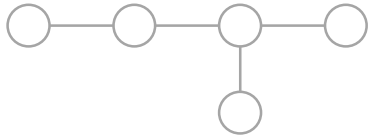
- N頂点の木が与えられる.
- 選んだ全ての頂点が，１つのパス上にあるような頂点の選び方を，選んだ頂点の数ごとに求めよ.



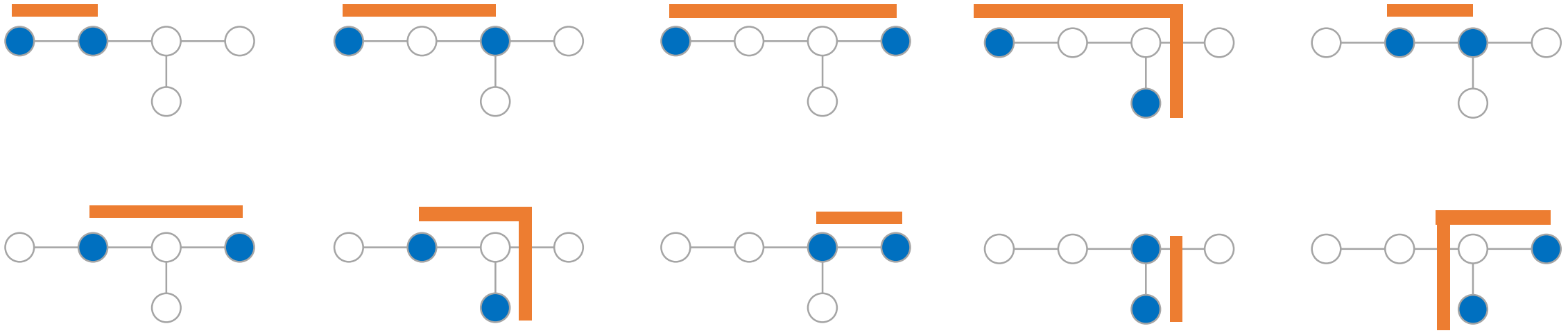


問題１３　山小屋スタンプラリー

入出力例



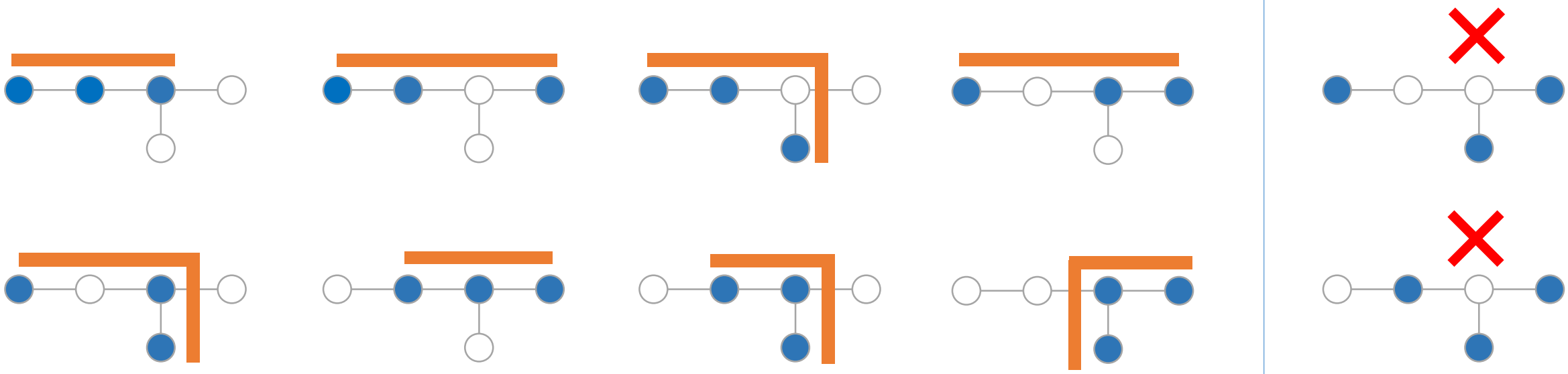
2点選んだ場合：10通り



問題１３　山小屋スタンプラリー

入出力例

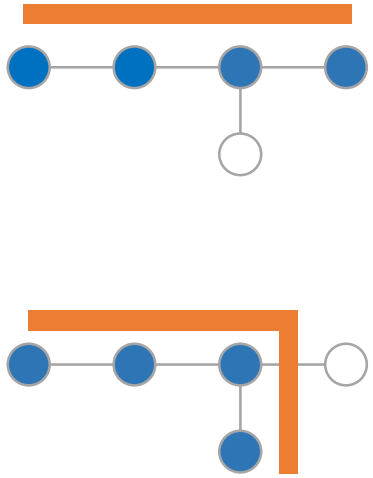
3点選んだ場合：8通り

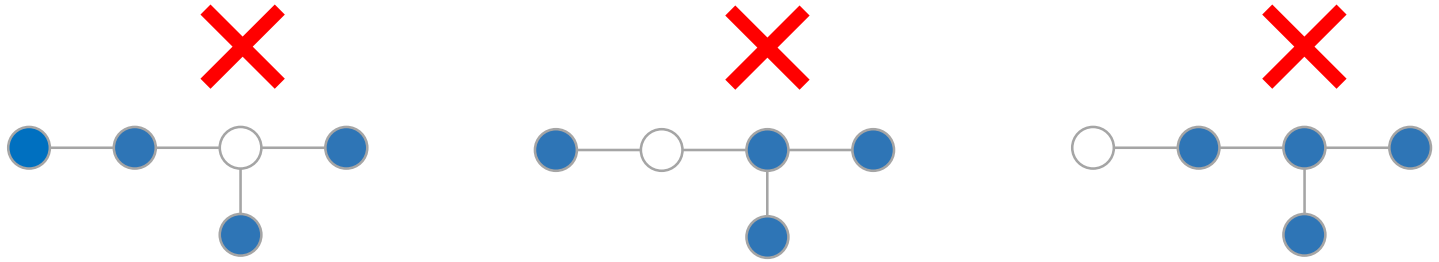


問題１３　山小屋スタンプラリー

入出力例

4点選んだ場合：2通り





問題１３　山小屋スタンプラリー

解法

- 木の重心分解
- 高速フーリエ変換
- O($Nlog^2N$)

問題１３　山小屋スタンプラリー

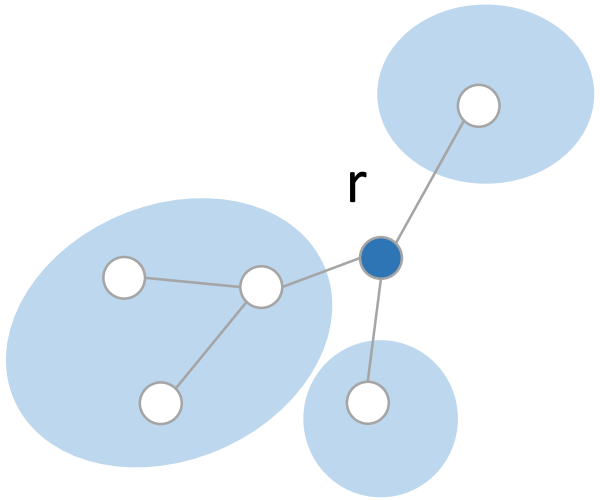
解法

- P_i: 長さがiであるパスの個数
- F_i: 長さが$(N - i + 1)$ のパスの個数（P_iから変換）
- H_i: $(N - i + 2)$個の頂点を選んだ時の，条件を満たす選び方の通り数（F_iを利用）

問題１３　山小屋スタンプラリー

P_i : 長さが i であるパスの個数を求める

頂点 r を通るパスの個数について考える． r の子の数を l とする



r の子 j を根とする部分木の深さテーブル C_{jd} : 深さ d のノード数

r を根とする部分木の深さテーブル A_d : 深さ d のノード数

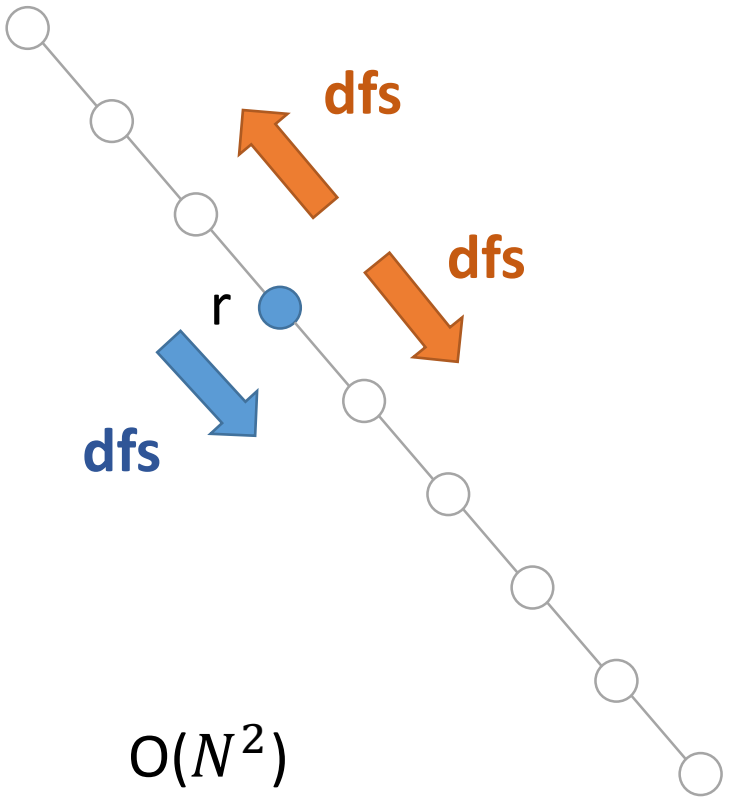
$$P_i = \frac{A * A - \sum_{j=0}^{l-1} C_j * C_j}{2}$$

＊畳み込み

$$(X * Y)_k = \sum_{i=0}^{k} X_i Y_{k-i}$$

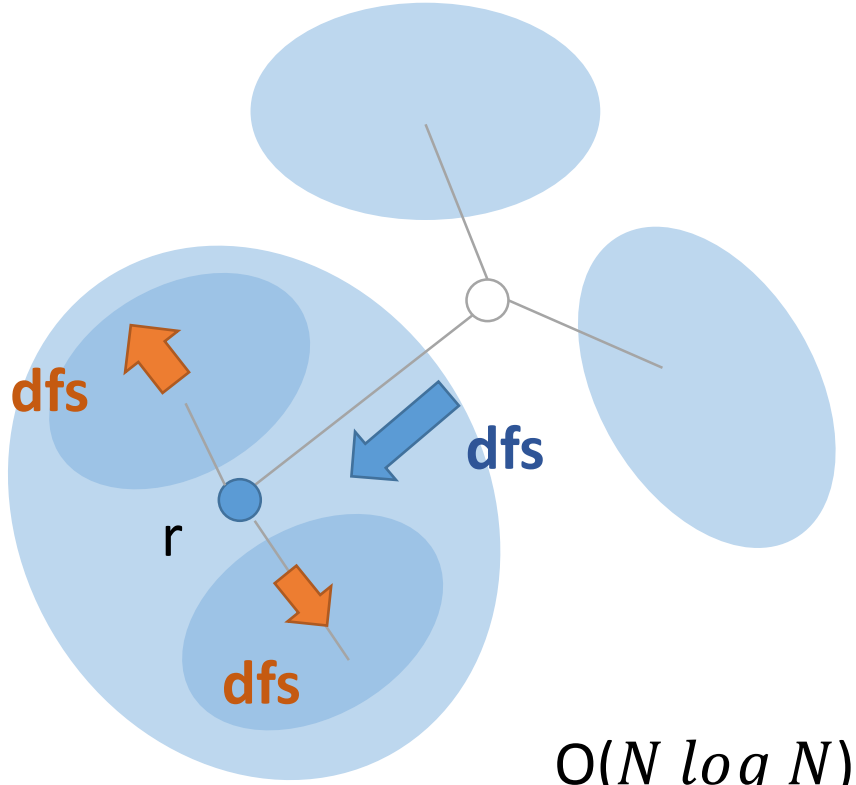
問題１３　山小屋スタンプラリー

コーナーケース



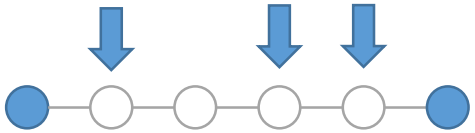
$O(N^2)$

重心分解



$O(N\ log\ N)$

問題１３　山小屋スタンプラリー



H_iを求める

F_i : 長さ$(N-i+1)$のパスの個数

H_i : $(N-i+2)$ 個の頂点を選んだときの条件を満たすものの通り数

$$H_i = \sum_{j=0}^{i} {}_{N-j}\mathrm{C}_{N-i} \times F_j$$

$$H_i = \sum_{j=0}^{i} \frac{(N-j)!}{\big((N-j)-(N-i)\big)!\,(N-i)!} F_j$$

$$(N-i)!\,H_i = \sum_{j=0}^{i} \frac{(N-j)!}{(i-j)!} F_j$$

$$G_i = \frac{1}{i!}$$

$$Y_i = (N-i)!\,H_i$$

$$X_j = (N-j)!\,F_j$$

とおくと

$$Y = X * G \quad \text{（畳み込み）}$$

問題１３　山小屋スタンプラリー

実装

- 長さが i であるパスの個数
  - 愚直に計算　→　O(N^2)　→　**時間制限**
  - 重心分解　→　O($N\ log\ N$)
- 畳み込み
  - 愚直に計算　→　O(N^2)　→　**時間制限**
  - FFT(高速フーリエ変換)　→　O($N\ log\ N$)
  - NTT(数論変換)　→　O($N\ log\ N$)
- 重心分解の階層の中でFFT/NTT　→　O($N\ log^2 N$)