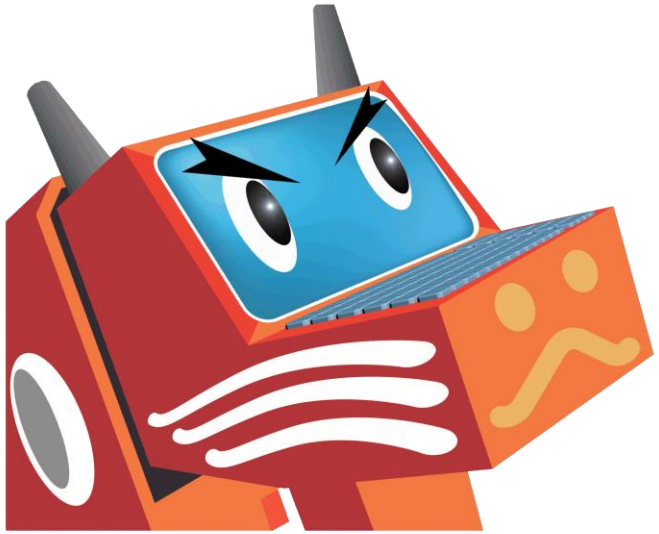
パソコン甲子園2021

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	**切り上げ除算**	基礎	2	☆	☆
2	**トウサ菌トウヒ菌**	基礎	4	★	☆
3	**ソートなぞなぞ**	基礎	4	★	★
4	**災害対応シミュレーション**	グラフ	6	★☆	★★
5	**祭りのくじ引き**	組み合わせ（貪欲法）	6	★★	★☆
6	**伝言ゲーム**	組み合わせ（数え上げ）	8	★★☆	★★
7	**配置ゲーム**	組み合わせ（動的計画法）	8	★★	★★☆
8	**星型５角形**	計算幾何学	8	★★☆	★★
9	**移動マスゲーム**	データ構造	10	★★★☆	★★★
10	**通学路**	グラフ	10	★★★	★★☆
11	**モンスター甲子園**	組み合わせ（動的計画法）	10	★★★★	★★★
12	**平方連続部分文字列**	文字列	12	★★★★☆	★★★☆
13	**ツリーキーボード**	データ構造	12	★★★★	★★★★☆

問題１　切り上げ除算　（２点）

問題１　切り上げ除算

概要

- $n \div d$ の結果を切り上げた整数値を求める.
- $(n + d - 1) \div d$の結果を切り捨てると求めることができる.

問題１　切り上げ除算

解法

- 整数の除算が小数点以下の切り捨てであることを利用し，そのまま計算する．
  答え=(n+d-1)/d，ただし/は整数の除算

- 浮動小数点で除算し、天井(ceil)関数$\lceil x \rceil$で切り上げる．
  答え=(int)ceil(n/d) ，ただし/は浮動小数の除算

問題１　切り上げ除算

解答例１

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

int main(){
    int n, d;
    cin >> n >> d;
    cout << (n+d-1)/d << endl;
    return 0;
}
```

解答例２

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

int main(){
    double n,d;
    cin >> n >> d;
    cout << (int) ceil(n/d) << endl;
    return 0;
}
```

問題２　トウサ菌トウヒ菌　　（４点）

問題２　トウサ菌トウヒ菌

概要

- ３つの数a_1, a_2, a_3が与えられる.
- ３つの数が等差数列か等比数列かを判定する.
- 第10項a_{10}を出力する.

問題２　トウサ菌トウヒ菌

解法

- $a_2 - a_1 = a_3 - a_2$であれば等差数列，そうでなければ題意より等比数列

- 等差数列の一般項は$a_n = a_1 + (n-1)d$,　　ただし$d = a_2 - a_1$
- 等比数列の一般項は$a_n = a_1 \times r^{n-1}$,　　ただし$r = a_2/a_1$

問題２　トウサ菌トウヒ菌

解答例

```
#include <iostream>
#include <cmath>
using namespace std;
int main(){
    int a1, a2, a3;
    cin >> a1 >> a2 >> a3;
    int d1=a2-a1;
    int d2=a3-a2;
    double r=a2/a1;
    if(d1==d2)
        cout << a1+9*d1 << endl;
    else
        cout << int(a1*pow(r,9)) << endl;
    return 0;
}
```

問題２　トウサ菌トウヒ菌

別解

- 数列が等差数列の時に$D_1 = 0, D_2 = 1$となり，数列が等比数列の時に$D_1 = 1, D_2 = 0$となるような式D_1, D_2を作ることができれば、一般項を
$$a_n = D_1 \cdot a_1 r^{(n-1)} + D_2 \cdot (a_1 + (n-1)d)$$
と表すことができる.

- 数列が等差数列の時に0になる式$D'_1 = (a_3 - a_2) - (a_2 - a_1)$は、数列が公比$r$の等比数列の時に値$a_1(r-1)^2$を持つので，$D_1 = D'_1 / a_1(r-1)^2$とおく．題意より$r - 1 \neq 0$.
- 数列が等比数列の時に0になる式は$\frac{a_2}{a_1} - \frac{a_3}{a_2}$, となる．題意より$a_1 a_2 \neq 0$なので、$D'_2 = a_2^2 - a_1 a_3$とおく。数列が公差$d$の等差数列の時，$D'_2$は値$d^2$を持つので，$D_2 = D'_2 / d^2$とする．題意より$d \neq 0$.

問題２　トウサ菌トウヒ菌

解答例２

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

int main(){
    double a1,a2,a3;
    cin >> a1 >> a2 >> a3;
    cout << int( (a3-2*a2+a1)/(a2/a1-1)/(a2/a1-1)*pow(a2/a1,9)
          +(a2*a2-a1*a3)/(a2-a1)/(a2-a1)*(a1+9*(a2-a1)) ) << endl;
    return 0;
}
```

問題３　ソートなぞなぞ（４点）

問題３　ソートなぞなぞ

概要

- アルファベット順に文字がソートされた文字列が一つ与えられる.
- 与えられた文字列の文字を並び替えて作ることのできる，すべての文字列の数を数える.

問題３　ソートなぞなぞ

解法

- 同じものを含む順列を計算する.
- 文字列の長さをn, 文字aの数をn_a, 文字bの数をn_b , …, 文字zの数をn_zとすると,

$$\text{答え} = \frac{n!}{n_a! \cdot n_b! \cdot \cdots \cdot n_z!}$$

- 数え方は様々
  1. 与えられる文字列がソートされていることを利用して数える
  2. 文字をASCIIコードと考え，文字を添え字にした配列を利用して数える
  3. 辞書順で次の順列を生成する関数を利用して数える

問題３　ソートなぞなぞ

解答例１

```
int main(){
    string s;
    cin >> s;
    int c=1,d=2;                          // c: 計算する順列の数, d: 連続する文字の数
    for(int i=1; i<s.size(); ++i){ // 2文字目からスタート
        if(s[i-1]!=s[i]) d=1;             // 前と文字が異なっていたらdをリセット
        c *= i+1;                         // n!の計算
        c /= d;                           // それぞれの文字の階乗を逆数で計算
        d++;                              // 文字が続くとしてdを1増やす
    }
    cout << c << endl;
    return 0;
}
```

問題３　ソートなぞなぞ

解答例２

```
int cnt[256];

int main(){
  int res = 1;
  string str; cin >> str;

  for ( int i = 0; i < str.size(); i++ ){
    res *= (i+1);
    cnt[str[i]]++;
  }

  for ( int i = 0; i < 256; i++ ){
    if ( cnt[i] == 0 ) continue;
    for ( int j = 2; j <= cnt[i]; j++ ) res /= j;
  }

  cout << res << endl;
  return 0;
}
```

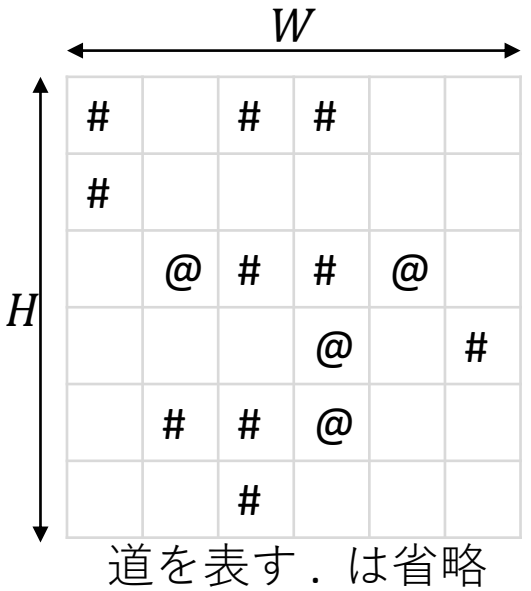
問題３　ソートなぞなぞ

解答例３

```
int main(){
    string s;
    cin >> s;
    int c=1;
    while(next_permutation(s.begin(),s.end())) c++;
    cout << c << endl;
    return 0;
}
```

問題４　災害対応シミュレーション（６点）

問題４　災害対応シミュレーション



道を表す．は省略

概要

H行W列のマスに，壁(#)，道(.)，障害物(@)がある．
建物の中に取り残されたN人の位置が与えられる．
N台のロボットが同じ基地局から発進し，それぞれ一人ずつ
救助するようにしたい．

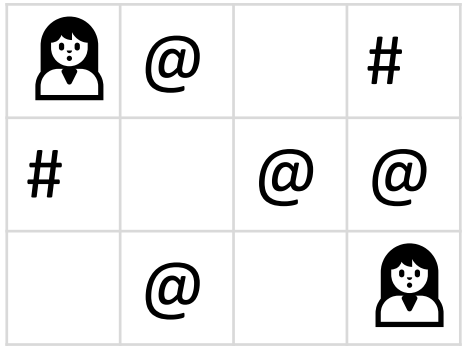
ロボットは建物の外や壁のマスに進むことはできない．
障害物のマスには進めるが，ロボットが１ダメージ受ける．
すべての人を救助できる基地局の位置のうち，ロボットが受けるダメージの最大値が最小になるような基地局の位置を求める．

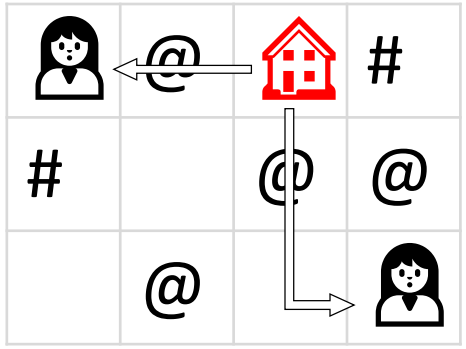
$2 \leqq H, W \leqq 300, 1 \leqq N \leqq 300.$

問題４　災害対応シミュレーション

入出力例

入力
```
3 4 2
.@.#
#.@@
.@..
1 1
3 4
```



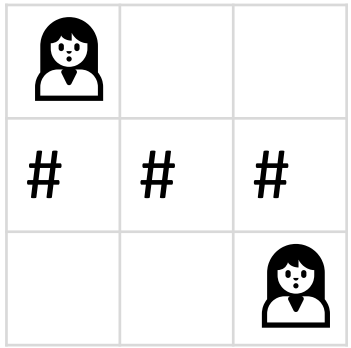


$(1,3)$や$(2,2)$に基地局を配置すると、2台のロボットのダメージの最大値が最小になる.

行番号がより小さいものを出力する.

⇨ 出力
1 3

入力
```
3 3 2
...
###
...
1 1
3 3
```



全員を救助できる基地局の位置は無い

⇨ 出力
-1 -1

問題４　災害対応シミュレーション

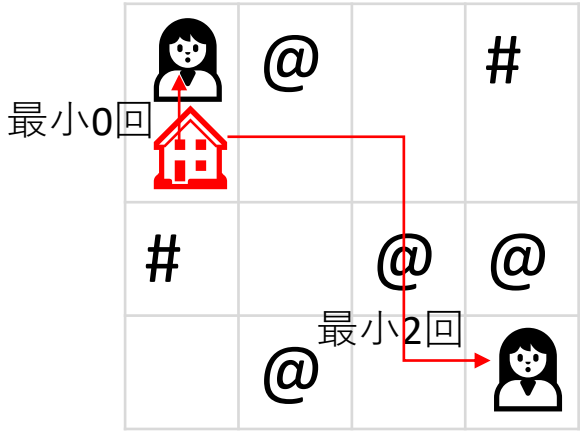
誤解法

基地局を設置可能なすべての位置(道)から、N人全員へ到達するときに経由しなければならない障害物の個数の最小値をBFSで求める.

見つかったダメージの最大値がより小さければ、答えを更新していく.

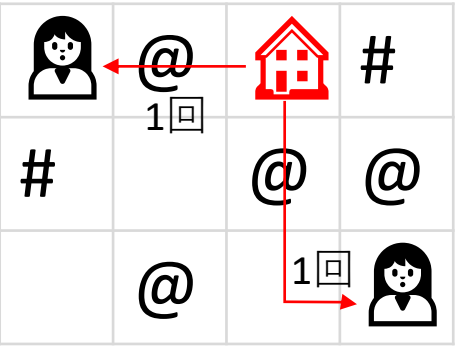
$O(H^2W^2)$　**時間制限**

(1,1)に基地局を設置



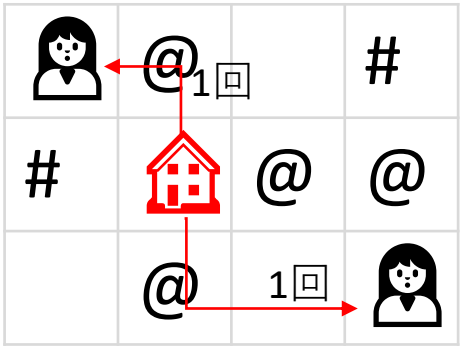
最大ダメージ２

(1,3)に基地局を設置



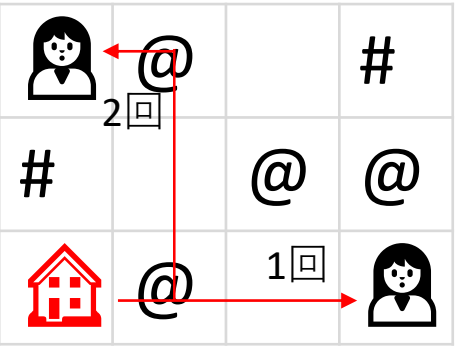
最大ダメージ１

(2,2)に基地局を設置



最大ダメージ１

(3,1)に基地局を設置



最大ダメージ２

問題４　災害対応シミュレーション

解法

N人について、各人の位置から、各マスへ何個の障害物を経由して到達できるかをBFS（幅優先探索）で求めておく.

コストが0または1の01BFS

経由しなければならない障害物の数がより大きいマスがあれば，マスの値を更新する.

全て終わった後，一番小さい値を持つマスが基地局の候補.

$O(NHW)$　　正解

(1,1)の人から各マスへ到達するために経由しなければならない最小の障害物の数を求める

問題５　祭りのくじ引き（６点）

問題５　祭りのくじ引き

概要

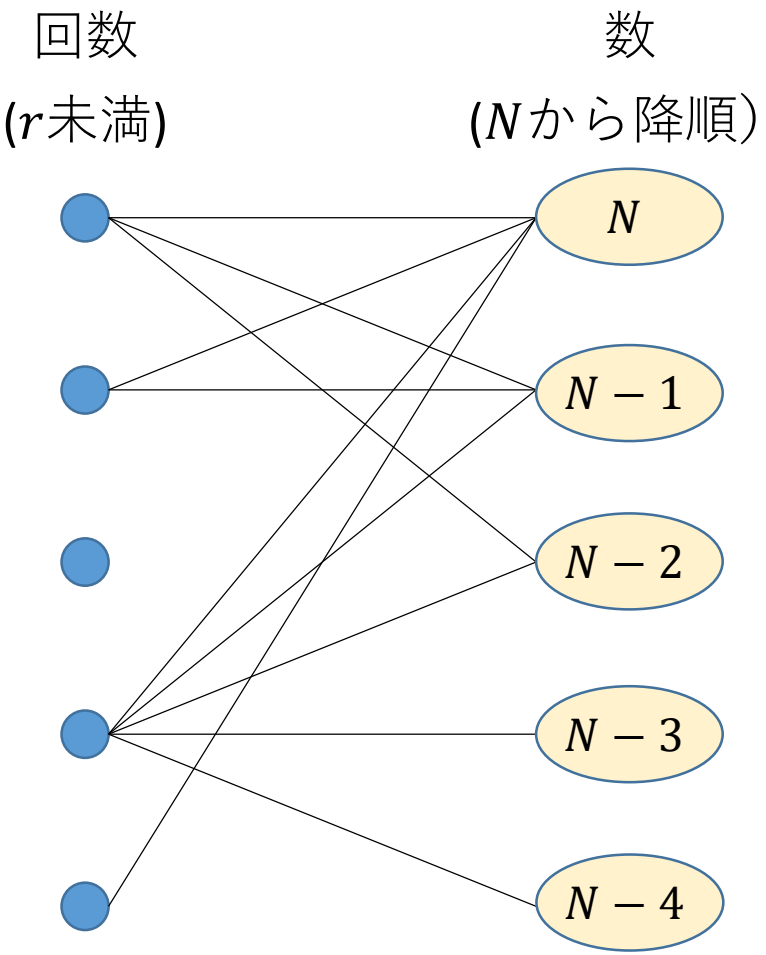
- 1からNまでの数が書かれたくじが１枚ずつ，はずれくじが複数枚ある．
- N人の参加者のそれぞれが，数が書かれたくじを引くまでくじを引く．
- くじを引いた回数とくじに書かれた数の積の昇順に順位が決まる．
- コンドウさんがr回，他の参加者がr_i回くじを引いたとき，コンドウさんがなり得る最高の順位を求めよ．
- $2 \leqq N \leqq 100$，$1 \leqq r \leqq 200$，$1 \leqq r_i \leqq 200$

問題５　祭りのくじ引き

考察

- コンドウさんの順位が最も高くなるのは、コンドウさんが1を引いたとき。このとき、回数と数の積がr未満になる参加者の数を最小になるような引き方を考えればよい.
- 回数r以上の参加者の積は必ずr以上.
- そのため，積がr以上になる回数r未満の参加者数が最も多くなるようなくじの引き方を考えればよい.
- r未満の回数と引いた数を頂点にして二部グラフ(右図)を作れば，最大マッチングで解ける．$O(N^2)$

  ⇒ 制約が小さいので，これでも解ける.
- 想定解法は貪欲解法．$O(N \log N)$



問題５　祭りのくじ引き

貪欲解法のための考察

- 回数が少ない人が大きな数を引けば，回数と数の積がr以上になる可能性が高まる.
- したがって、回数が少ない人ほど，より大きな数を引くようなくじの引き方が有望.
- ただし，回数がとても少ない人が大きな数を引いても積がr未満になることもある.
- そのような人が小さな数を引けば，より回数の多い他の人が大きな数を引くことで積がr以上の人数を増やせる可能性がある.

N=7, r=21

回数r未満の参加者の回数が、昇順に3, 3, 4, 5とする.

回数	3	3	4	5
数	7	6	5	4
積	21	**18**	**20**	**20**

積がr未満の参加者は３人

回数	3	3	4	5
数	7	**4**	6	5
積	21	**12**	24	25

積がr未満の参加者は１人⇒最小

問題５　祭りのくじ引き

貪欲解法のための考察

先ほどの考察を詳細化する.

- 回数r未満の参加者の回数を昇順に並べた列Sを考える．MをSの長さとする.
- 列Sの i番目の要素に以下のような数x_iを割り当てる.
  - $x_0 = N$とする.
  - $S[i] * x_i \geq r$なら$x_{i+1} = x_i - 1$，$S[i] * x_i < r$なら$x_{i+1} = x_i$　$(i < M)$
- x_iは参加者iが引く数の候補を表す.
- $S[i] * x_i \geq r$なら，参加者iがx_iを引けばiの積はr以上．⇒ iはx_iを引くとする.
- $S[i] * x_i < r$なら，参加者iが引ける最大の数x_iを引いてもiの積はr未満.
  ⇒ iはx_M以下の数yを引くとする.
  $x_i \geq x_M \geq y$ なので，yを引いても参加者iの積はr未満.
- この引き方でコンドウさんの順位が最高になることは，数学的帰納法で証明できる.

問題５　祭りのくじ引き

解法

1. r より小さい r_i だけを持つ列を昇順にソートする(この列をS とする).
2. cとxの初期値をそれぞれ0とNにする.
3. 列Sの先頭から順に以下を行う(i は列Sのインデックス).
   - $S[i] * x \geq r$ ならxの値を１減らす.
   - $S[i] * x < r$ ならcの値を１増やす.
4. $c + 1$ が求める順位.

- ステップ1がO($N \log N$) かかる.
- ステップ3がO(N) かかる.
- 全体としてO($N \log N$) かかる.

問題５　祭りのくじ引き

解答例

```
// r未満の回数が昇順に並んだ列を作る
vector<int> S;
for ( int i = 1; i < N; i++ ){
  cin >> t;
  if (t < r) S.push_back(t);
}
int M = S.size();
sort(S.begin(), S.end());
```

```
  // 順位を求める
  int c = 0, x = N;
  for ( int i = 0; i < M; i++ ){
    if (S[i]*x < r) c++;
    else x--;
}
cout << c+1 << endl;
```

問題６　伝言ゲーム　　（８点）

問題６　伝言ゲーム

概要

- N人から選ばれた５人が横一列に並ぶ.
- 隣り合う人が共通の言語を使えれば伝言を正しく伝えられる.
- 左端から右端の人まで伝言を伝えられれば伝言ゲームが成功.
- 伝言ゲームが成功する参加者の並び方の数を計算せよ.

- $5 \leqq N \leqq 200$

問題６　伝言ゲーム

誤解法

- N人から5人を選んだ並べ方をすべて試す.
  O(N^5)　**時間制限**

```
long long res = 0;
for ( int a = 0; a < N; a++ ){
    for ( int b = 0; b < N; b++ ){
        if ( a == b ) continue;
        for ( int c = 0; c < N; c++ ){
            if ( a == c || b == c ) continue;
            for ( int d = 0; d < N; d++ ){
                if ( a == d || b == d || c == d ) continue;
                for ( int e = 0; e < N; e++ ){
                    if ( a == e || b == e || c == e || d == e  ) continue;
                    if ( M[a][b] && M[b][c] && M[c][d] && M[d][e] ) res++;
```

問題６　伝言ゲーム

解法

- 集合を用いた効率的な数え上げ.
- 前処理で共通の言語が使えるかどうかを表す隣接行列を作る.
  - O(KN^2)で構築　（Kは言語の数）
  - 指定された二人が話せるかをO(1)で調べることができるようになる

問題６　伝言ゲーム

解法

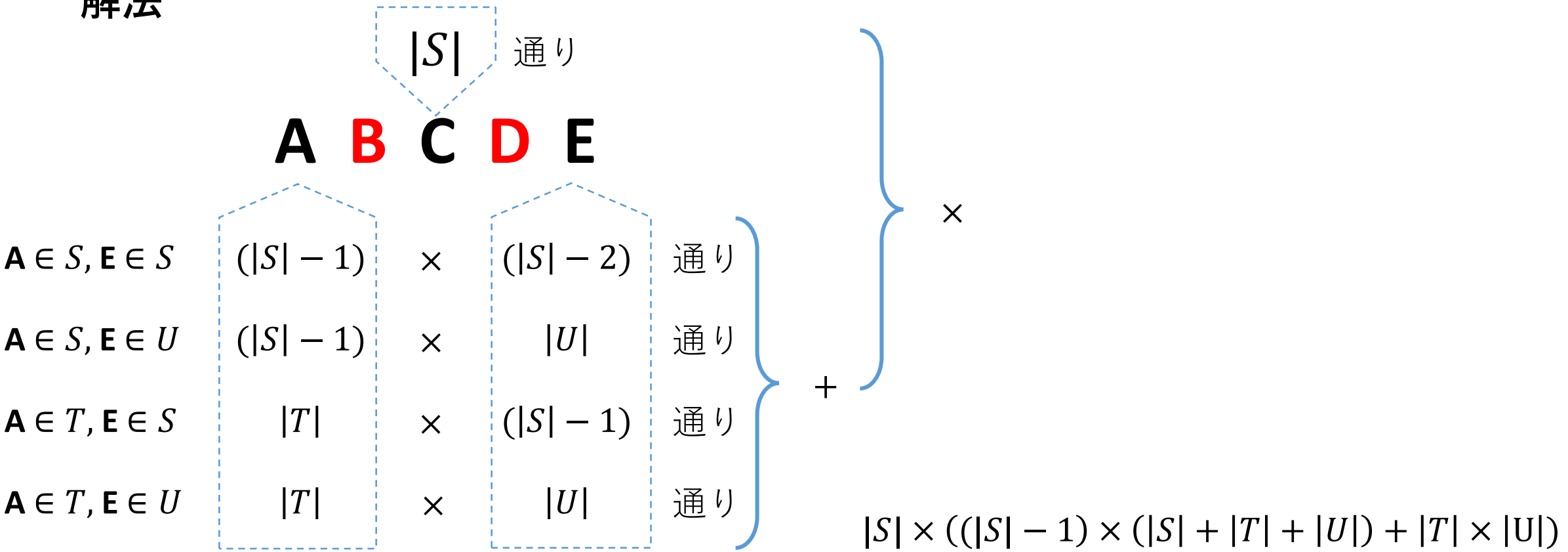
- 選ばれた５人を前からA, B, C, D, Eとする.
- BとDを決め打ちし，以下の集合の要素数を求めておく．　O(N^3)
  BともDとも話せる人の集合　　S
  Bとは話せるがDとは話せない人の集合　　T
  Dとは話せるがBとは話せない人の集合　　U
- これらの集合を使い，BとDを固定したときの通り数を求める.
- BとDを決め打ちしO(N^2)

A　B　C　D　E

問題６　伝言ゲーム

BともDとも話せる人の集合　S
Bとは話せるがDとは話せない人の集合　T
Dとは話せるがBとは話せない人の集合　U

解法



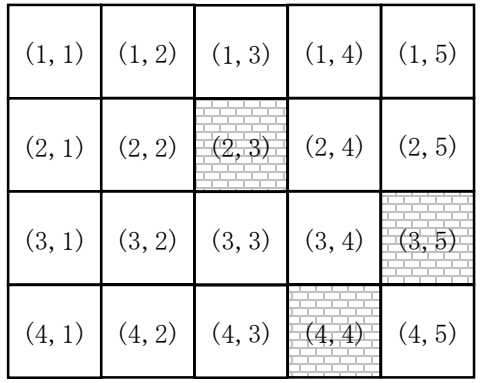
$$|S| \times ((|S| - 1) \times (|S| + |T| + |U|) + |T| \times |\mathrm{U}|)$$

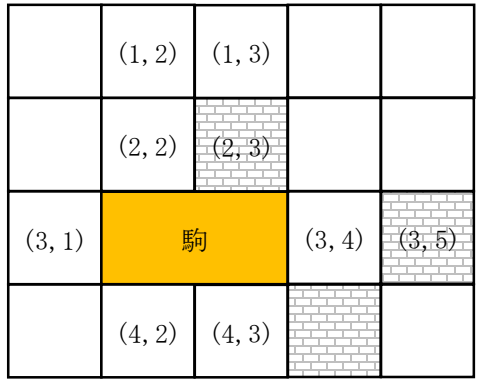
問題７　配置ゲーム　　（８点）

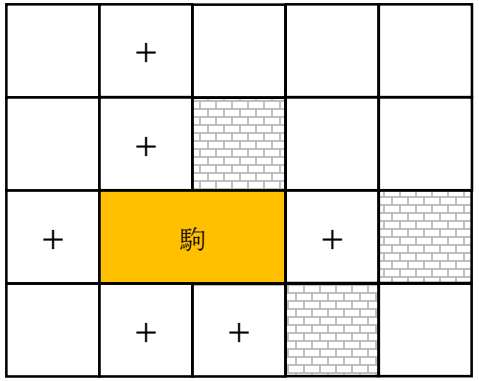
問題７　配置ゲーム

概要

- 縦にH個、横にW個の区画の盤面に，縦にR個、横にC個分の区画を使う長方形の駒を一つ配置する．
- 障害物がある区画を使うような駒の置き方はできない．
- 駒に向かって、縦方向または横方向にまっすぐ移動したとき，障害物に出会わずに駒に到達できる区画の数を求めよ．
- $H, W, R, C \leqq 1{,}000$







問題７　配置ゲーム

誤解法

- 障害物に出会わずに駒に到達できる区画をそれぞれの方向へ線形探索しながら，駒を置く区画を$(H-R+1)\times(W-C+1)$箇所調べる．

  O$((HW)^2)$　**時間制限**

- 障害物に出会わずに駒に到達できる区画の数を，累積和によってO(1)で求めることができるようにしておき，駒を置く区画を$(H-R+1)\times(W-C+1)$箇所調べる．ただし，駒を置けるかの判定をO(RC)で行うと・・・

  O$(HWRC)$　**時間制限**

問題７　配置ゲーム

解法

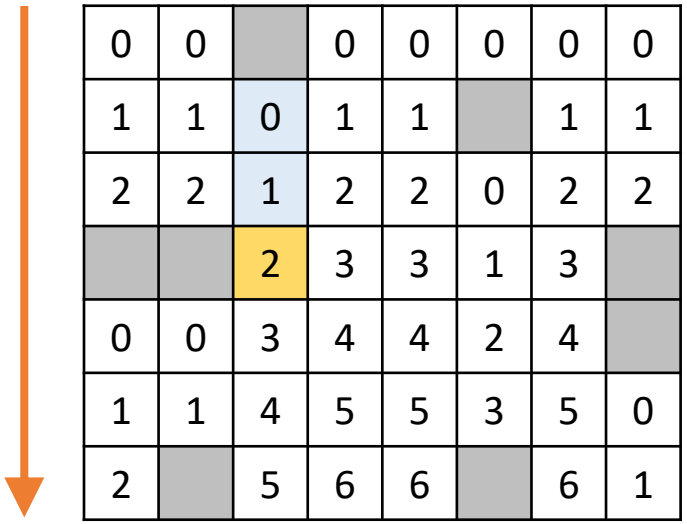
- 駒に到達できる区画の数を，累積和によってO(1)で求めることができるようにしておく．
- さらに，行（列）の区間について，駒に到達できる区画の数を，累積和によってO(1)で求めることができるようにしておく．
- 駒を置く区画を$(H-R+1)\times(W-C+1)$箇所調べる．
- 駒を置けるかの判定も求めておいた累積和で行う．

O(HW)　**正解**

問題７　配置ゲーム

解法

- 障害物に出会わずに駒に到達できる区画の数を，累積和によってO(1)で求めることができるようにしておく（４方向）



0	0		0	0	0	0	0
1	1	0	1	1		1	1
2	2	1	2	2	0	2	2
		2	3	3	1	3	
0	0	3	4	4	2	4	
1	1	4	5	5	3	5	0
2		5	6	6		6	1

$fromNorth[i][j]$

$fromNorth[i][j]$: 区画$[i][j]$にある駒に北側から到達できる区画の数

$fromSouth[i][j]$: 区画$[i][j]$にある駒に南側から到達できる区画の数

$fromEast[i][j]$: 区画$[i][j]$にある駒に東側から到達できる区画の数

$fromWest[i][j]$: 区画$[i][j]$にある駒に西側から到達できる区画の数

区画$[i][j]$に障害物がない場合

$$fromNorth[i][j] = fromNorth[i-1][j] + 1$$

問題７　配置ゲーム

解法

- $fromNorth[i][j]$に対する累積和によって，行の区間について駒に到達できる区画の数をO(1)で求めることができるようにしておく（４方向）

0	0	0	0	0	0	0	0
1	2	2	3	4	4	5	6
2	4	5	7	9	9	11	13
0	0	2	5	8	9	12	12
0	0	3	7	11	13	17	17
1	2	6	11	16	19	24	24
2	2	7	13	19	19	25	26

$fromNorthAcc[i][j]$

$$fromNorthAcc[i][j] = fromNorthAcc[i][j-1] + fromNorth[i][j]$$

行iの区間$[j, j+C)$に渡る，駒に到達できる区画の数
$fromNorthAcc[i][j+C-1] - fromNorthAcc[i][j-1]$

問題７　配置ゲーム

解法

- 駒をおけるかどうかは，$fromSouth[i][j]$を用いて効率よく判定できる．
  $fromSouth[i][j] + 1 < R$なら駒は置けない．

2	2		6	6	0	6	2
1	1	5	5	5		5	1
0	0	4	4	4	3	4	0
		3	3	3	2	3	
2	1	2	2	2	1	2	
1	0	1	1	1	0	1	1
0		0	0	0		0	0

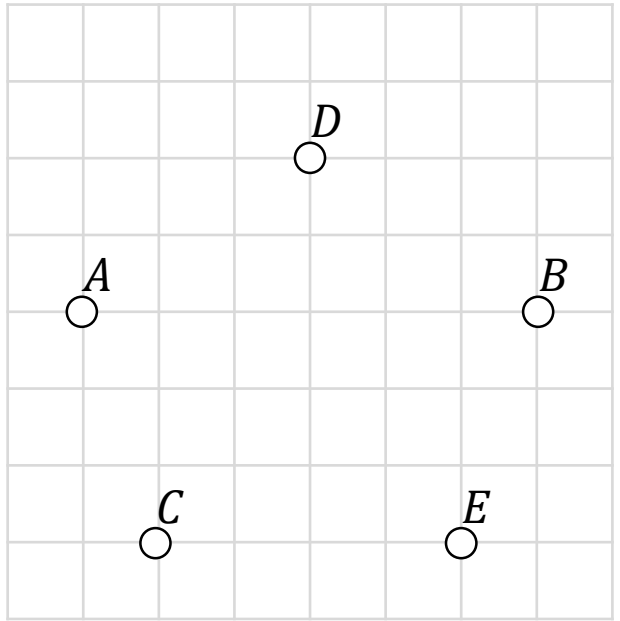
$fromSouth[i][j]$

```
for ( int i = 1; i <= H + 1 - R; i++ ){
  int low = 0; // 列幅Cの区間で行幅が足りない列の数を保持
  for ( int j = 1; j <= C; j++ ) if ( fromSouth[i][j] + 1 < R ) low++;
  for ( int j = 1; j <= W + 1 - C; j++ ){
    if ( low == 0 ) res = max(res, getScore(i, j));
    if (fromSouth[i][j] + 1     < R ) low--;
    if (fromSouth[i][j + C] + 1 < R ) low++;
  }
}
```

$O(HW)$

問題８　星型５角形　　（８点）

問題８　星型５角形



概要

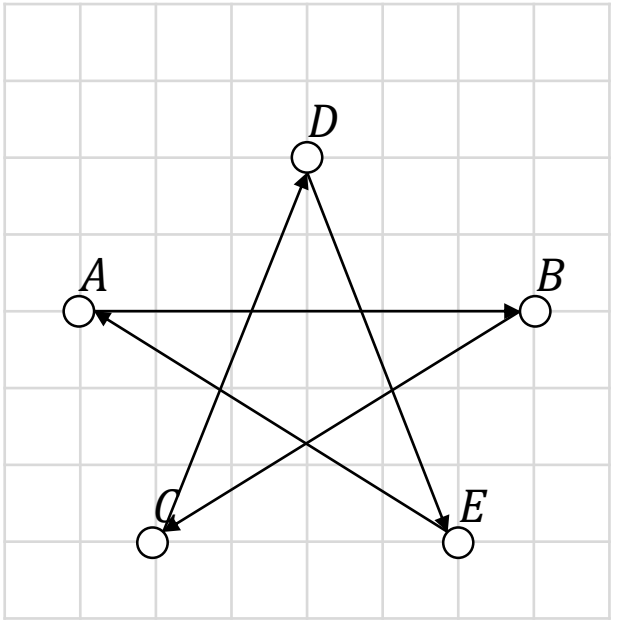
５点A, B, C, D, Eの座標x_i, y_iが整数で与えられる.
座標に重複は無く、どの３点も１直線上に無い.
$A \to B \to C \to D \to E \to A$の順につないだ直線が、
星形５角形になっているか判定する.

$0 \leqq x_i,\ y_i \leqq 100.$
ある点が0.00001動いたとしても答えに影響はない入力しか与えられない.

問題８　星型５角形

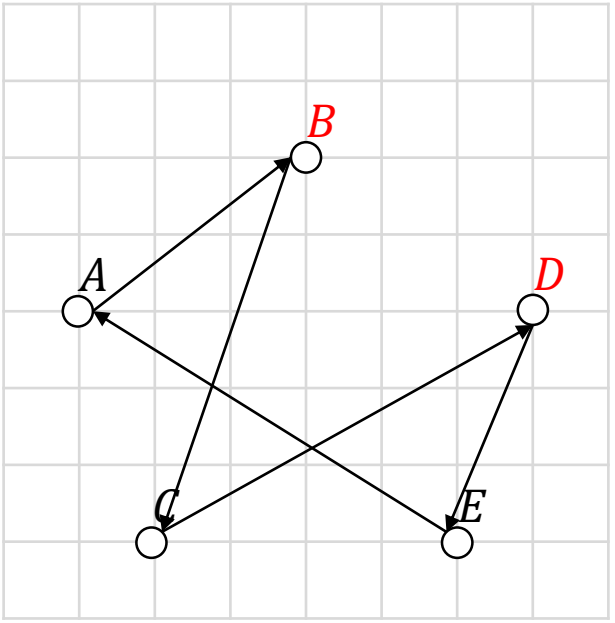
入出力例

入力
1 4
7 4
2 1
4 6
6 1



出力
1

入力
1 4
4 6
2 1
7 4
6 1



出力
0

問題８　星型５角形

解法

以下の線分ペアが交点を持つか調べる.

ABとCD

BCとDE

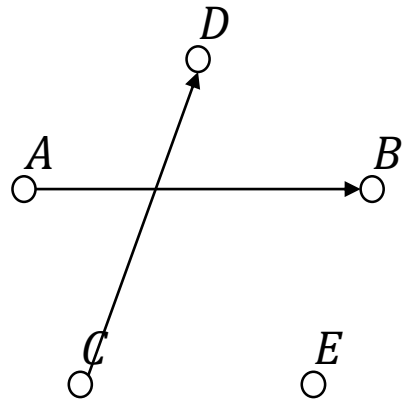
CDとEA

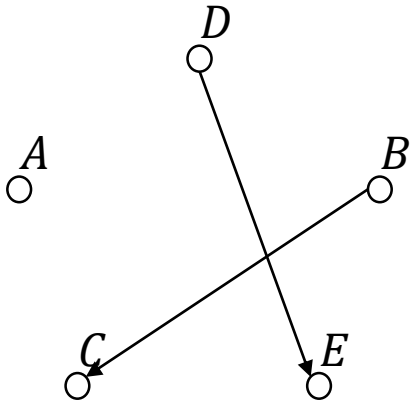
DEとAB

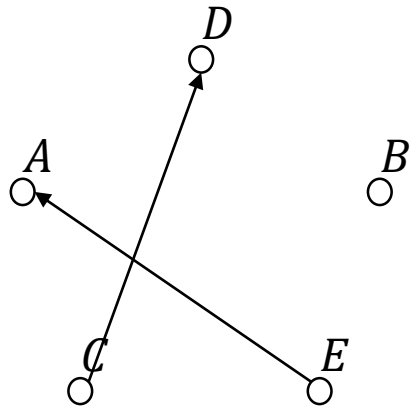
EAとBC

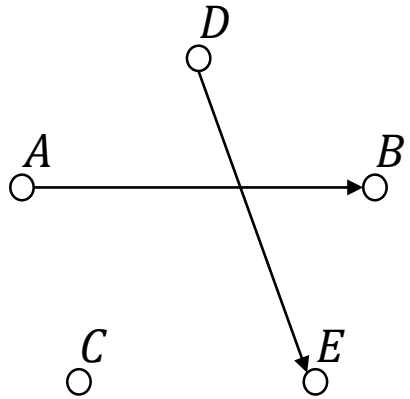
5点とも端点以外で交われば星形５角形.

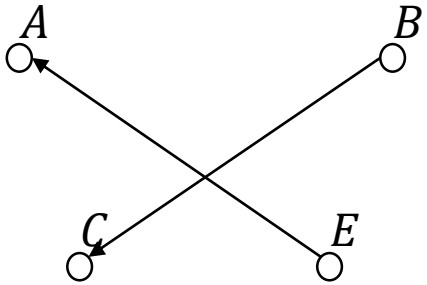
(ただし、他の線分ペアの選び方でも可能)







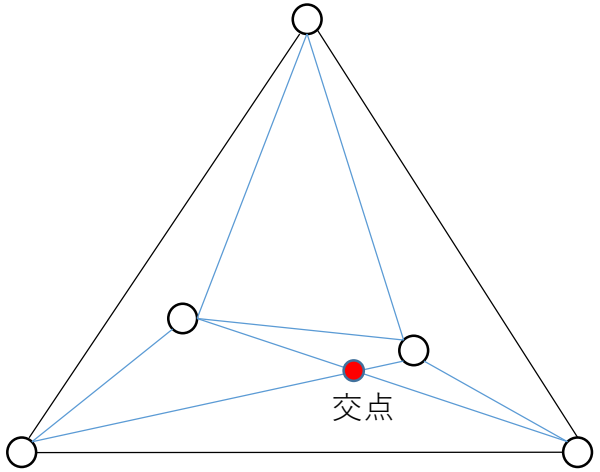




問題８　星型５角形

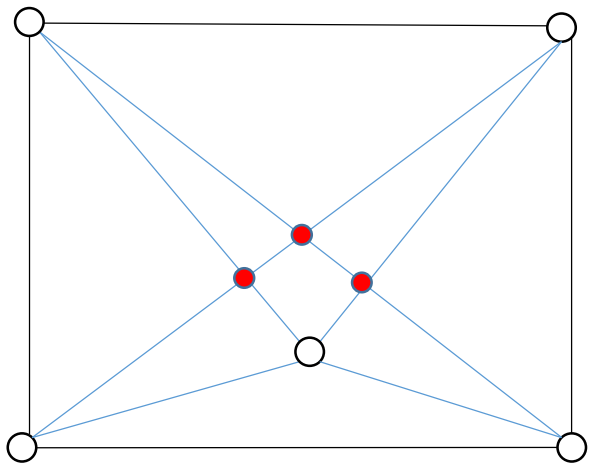
証明（概略）

５点の接続関係をすべて考え、それぞれの場合に交点がいくつになるか考える.
そのために、点A, B, C, D, Eで作られる凸包を考えて場合分けを単純化する.
３点が１直線上に無いので、凸包としてあり得るのは３角形、４角形、５角形の場合.



凸包が３角形の場合

すべての点同士を結んだ線分ペアの組み合わせから
端点以外の交点は１点しか作れない

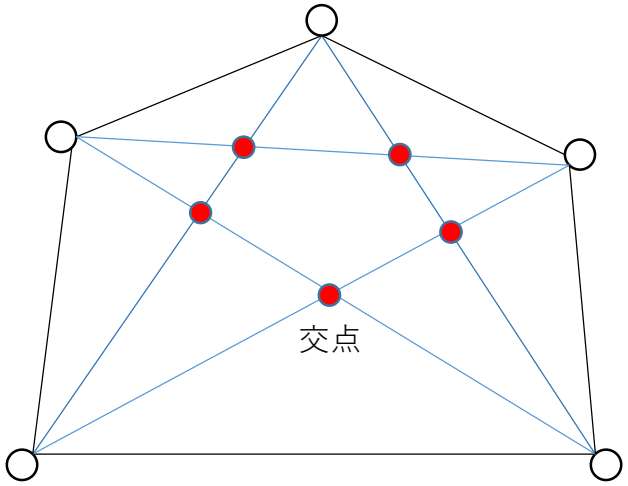


凸包が４角形の場合

すべての点同士を結んだ線分ペアの組み合わせから
端点以外の交点は３点しか作れない

問題８　星型５角形

証明（概略）



凸包が５角形の場合.

すべての点同士を結んだ線分ペアの組み合わせから端点以外の交点が５点作れる

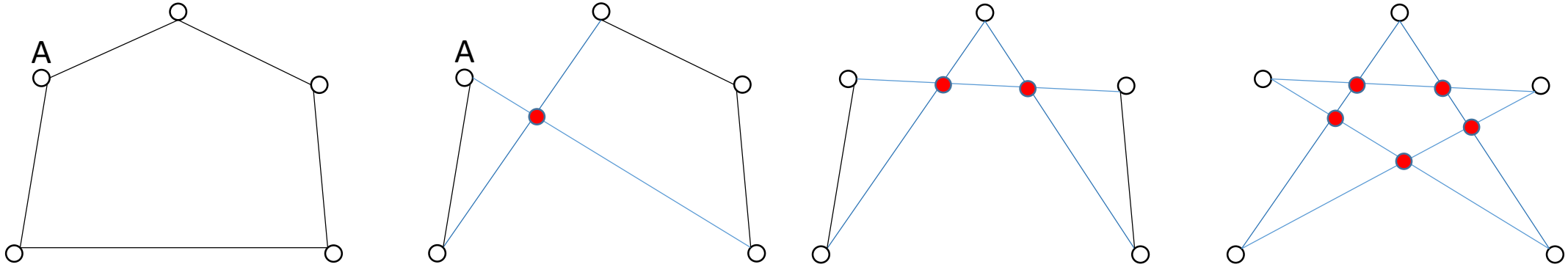
ただし、入力で与えられるのは順番に結ぶ５点なので、このうち５本の線分から構成されて、各点を１度ずつ訪問するもののうち、線分で閉じた形状(分岐せずに元の点に戻るもの)をすべて考える

問題８　星型５角形

証明（概略）

$A \to B \to C \to D \to E \to A$のように、各点を１度ずつ、閉じた形で５点を訪問する方法は、回転(５通り)や逆順(２通り)を除くと５!÷ ５ ÷ ２ ＝ １２通り考えられる.

すべて網羅すると、接続関係が同じものは以下の４通りになる.



凸包の辺を１本でも使うと、端点でしか交わらない線分ペアが発生する.
すべての対角線を使う場合だけ、すべての対角線が端点以外で交わり、
交点が５つできて星形５角形になる.

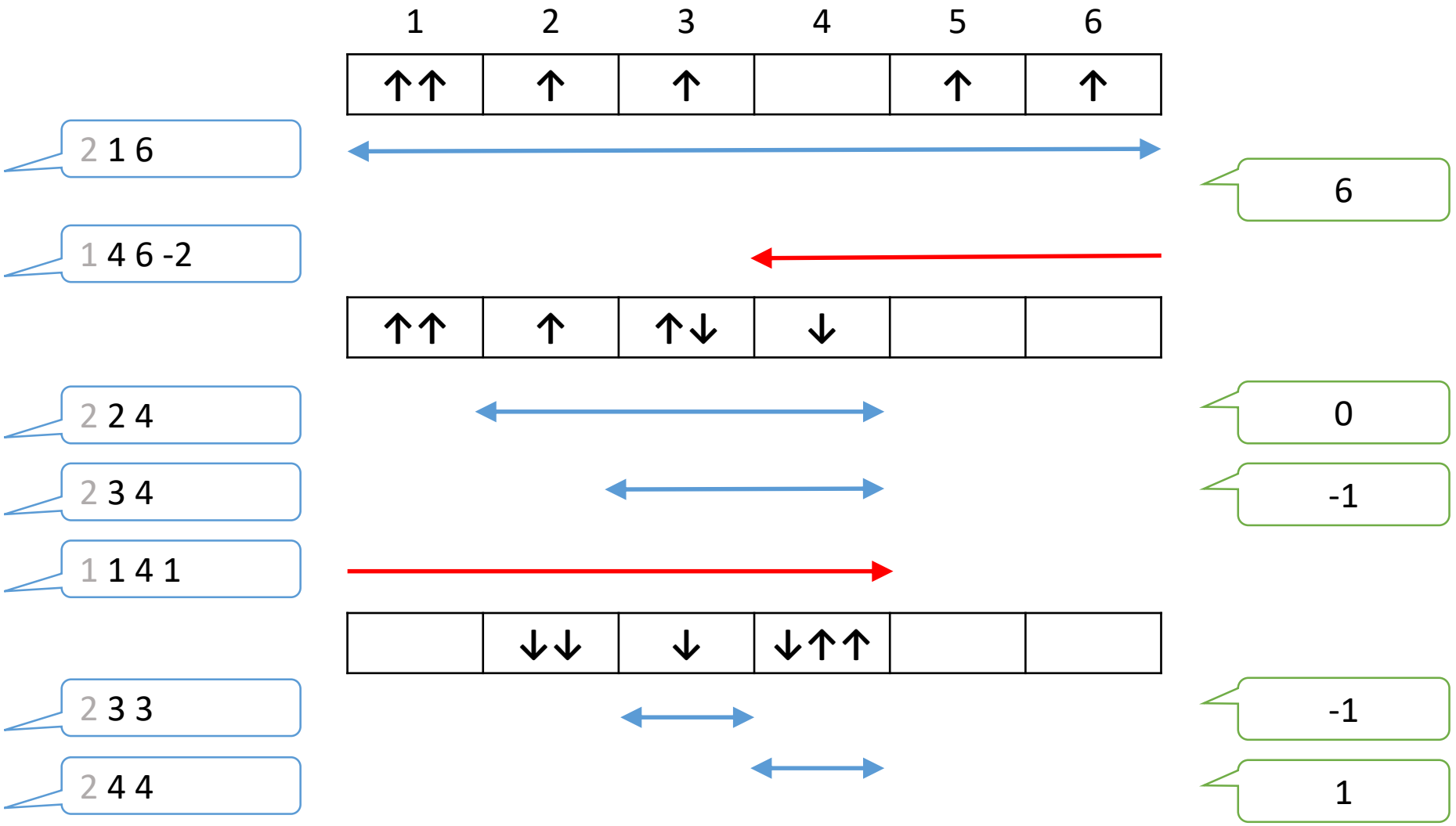
問題９　移動マスゲーム　（１０点）

問題９　移動マスゲーム

概要

- 1からNまでの番号が付いたN個の区画にM人の生徒がいる.
- 区画は１列に並んでおり，どの区画もM人収容できる.
- 最初，M人の生徒はどこかの区画にいて，全員が北を向いている.
- 次の指示に答えなさい
  - 整数a, b, cに対して，a番目からb番目までの区画にいる生徒が全員，cの値に応じて移動し，向きを180度変える．cが正なら東へ，負なら西へ移動する．ただし，指定された範囲$[a, b]$を超えてはならない.
  - 整数d, eに対して，d番目からe番目までの区画にいる生徒のうち，北を向いている生徒の人数から南を向いている生徒の人数を引いた値を報告する.
- $N \leqq$ 1,000,000,000,　$M \leqq$ 100,000

問題９　移動マスゲーム



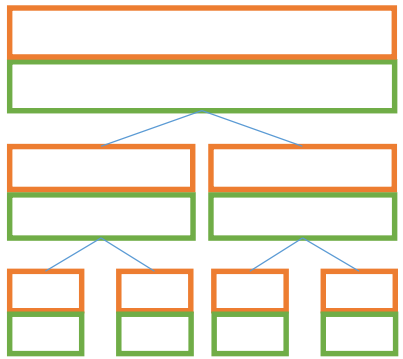
問題９　移動マスゲーム

考察

- 区画の数Nは大きいが，列に並んでいるM人の生徒の順番は変わらない.
- サイズMのセグメント木で管理できる.
- 並びの番号（インデックス）ではなく，区画の番号（座標）に基づいて，区間を探索する.
- 区間に対する更新があるので，**遅延伝播セグメント木**が必要

問題９　移動マスゲーム

解法



遅延評価付き
セグメント木

$lazy\ [k]$

k

l:	c:	z:	r:
移動後の，左端の生徒がいる区画の番号	どれだけ動くか（累積）	反転する回数（偶奇）	移動後の，右端の生徒がいる区画の番号

$value\ [k]$

k

l:	d:	r:
左端の生徒がいる区画の番号	北向きの生徒の数－南向きの生徒の数	右端の生徒がいる区画の番号

セグメント木を辿るときに必要

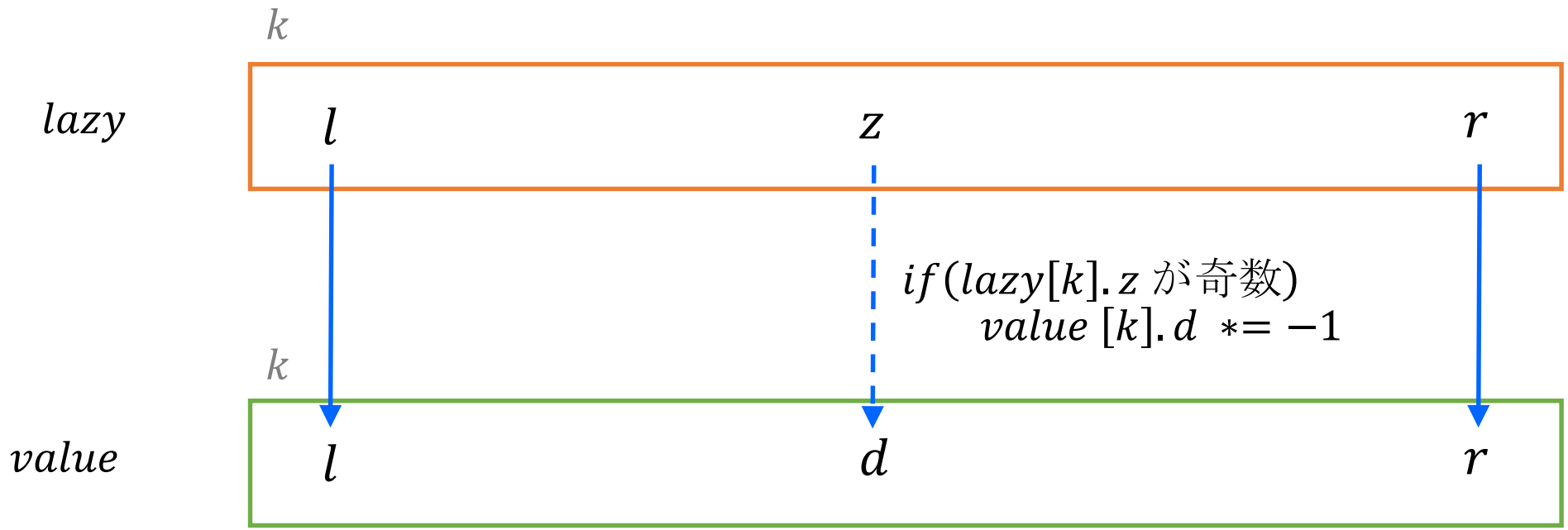
問題９　移動マスゲーム

セグメント木のノードの訪問

```
int query(int a, int b, int k, int l, int r){
  lazy_evaluate(k, l, r);
  if ( value[k].r < a || b < value[k].l ) return 0;
  if ( a <= value[k].l && value[k].r <= b ) return value[k].d;
  else{
    int vl = query(a, b, k*2 + 1, l, (l + r)/2);
    int vr = query(a, b, k*2 + 2, (l + r)/2, r);
    return vl + vr;
  }
}
```

問題９　移動マスゲーム

伝播手順１：値の更新

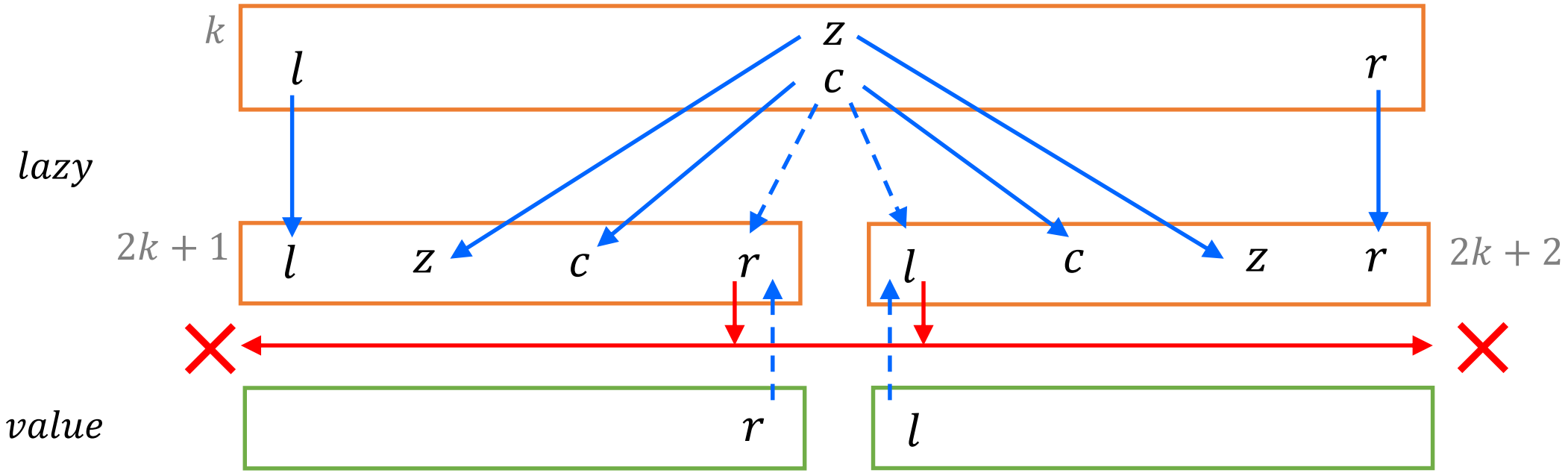


$value\ [k].l = lazy[k].l$

$value\ [k].r = lazy[k].r$

問題９　移動マスゲーム

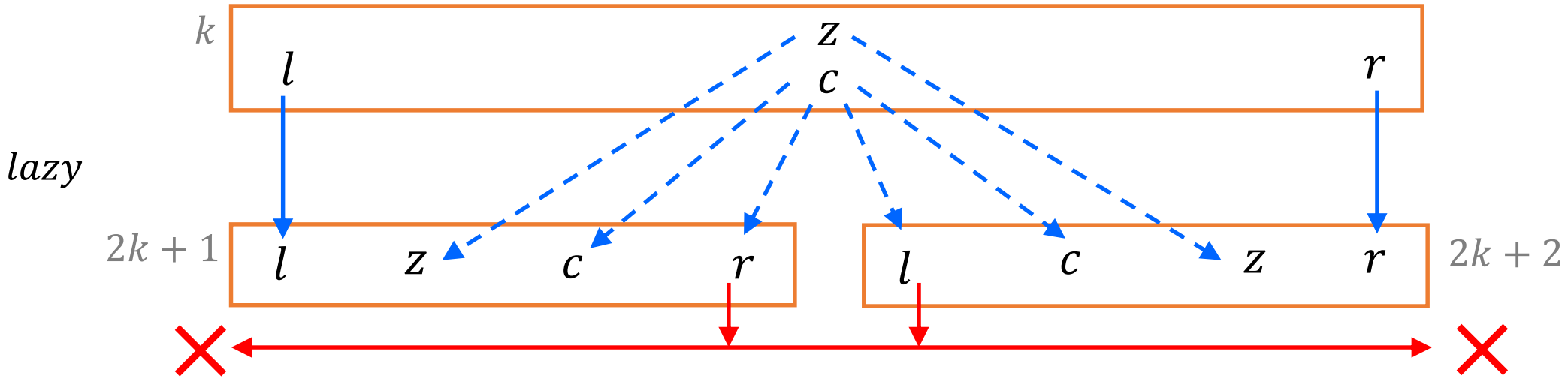
伝播手順２－１：新規



$lazy\,[2k+1].r = value[2k+1].r + lazy[k].c$　　$lazy\,[2k+2].l = value[2k+2].l + lazy[k].c$

問題９　移動マスゲーム

伝播手順２－２：更新



$lazy\,[2k+1].r \mathrel{+}= lazy[k].c$

$lazy\,[2k+1].c \mathrel{+}= lazy[k].c$

$lazy\,[2k+1].z \mathrel{+}= lazy[k].z$

$lazy\,[2k+2].l \mathrel{+}= lazy[k].c$

$lazy\,[2k+2].c \mathrel{+}= lazy[k].c$

$lazy\,[2k+2].z \mathrel{+}= lazy[k].z$

問題１０　通学路　（１０点）

問題１０　通学路

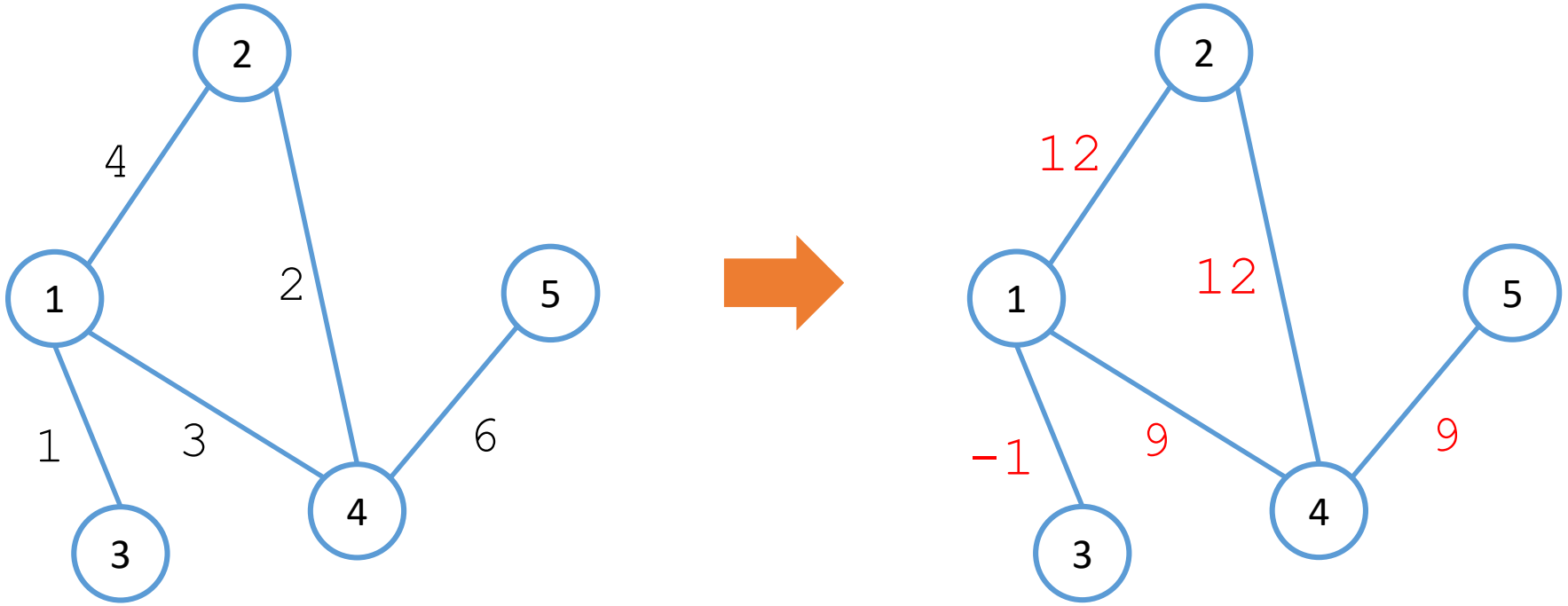
概要

- 頂点数N，辺数Mの重み付き無向グラフが与えられる．グラフは連結.
- 各辺について，その辺を使う場合の，頂点１から頂点Nへの最短経路の長さを求めよ.
- ただし，以下の条件を満たす必要がある.
  - 経路に含まれる頂点はちょうど１回通る
  - 条件を満たす経路の中で距離が最短
- $2 \leqq N \leqq 20,\quad M \leqq N(N-1)/2$，$1 \leqq d_i \leqq 1{,}000$

問題１０　通学路

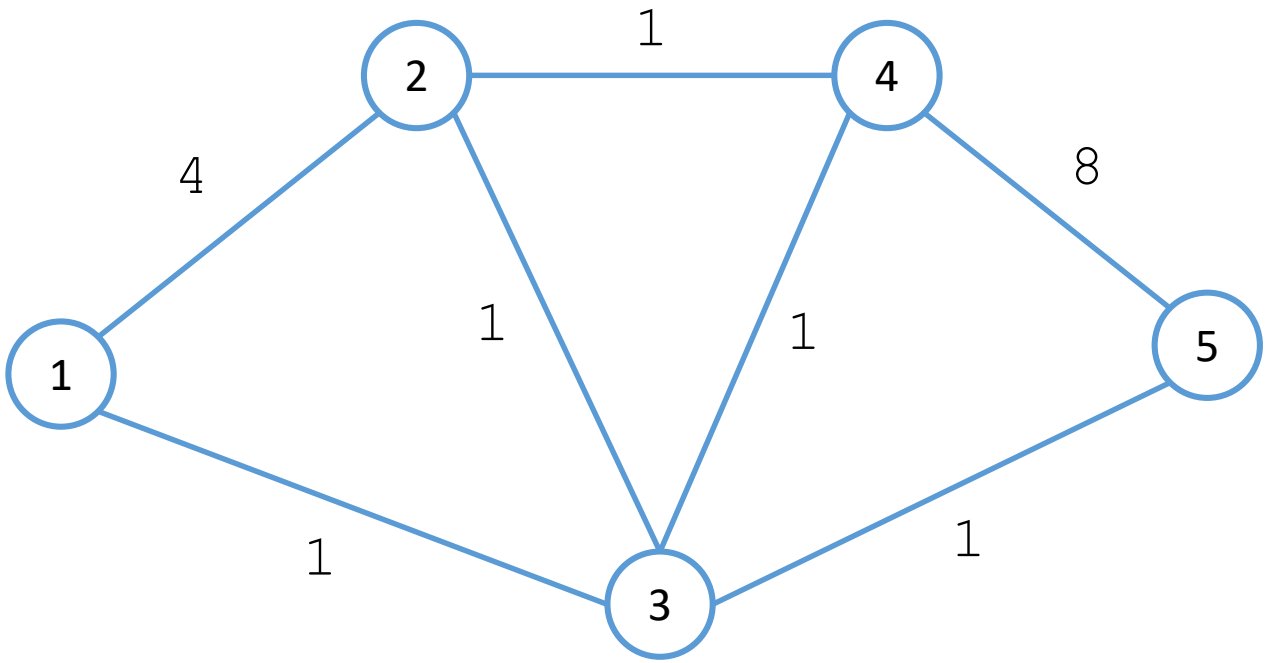
入出力例1

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



問題１０　通学路

コーナーケース



問題１０　通学路

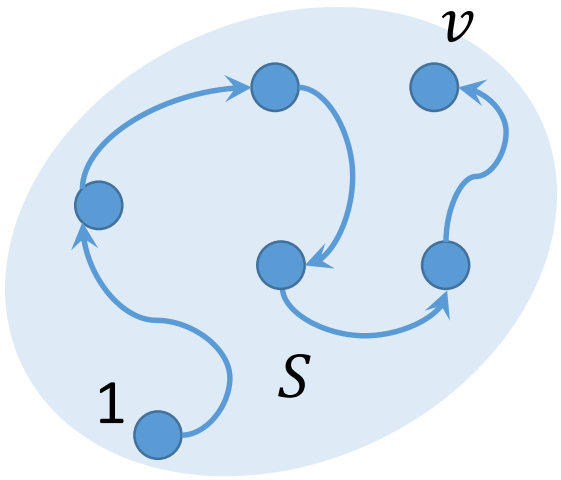
解法

- 動的計画法 (Dynamic Programming) .
- 空間O($N^2 2^N$), 時間O($(N^2+M)2^N$)のbitDP

問題１０　通学路

解法

- $\mathrm{dp1p}[v][S]$ = 頂点1を始点，vを終点とするパスのうち，Sに含まれる頂点をちょうど１度づつ使うものの最小コスト（存在しなければINF）
- $O(N^2 2^N)$のbitDP

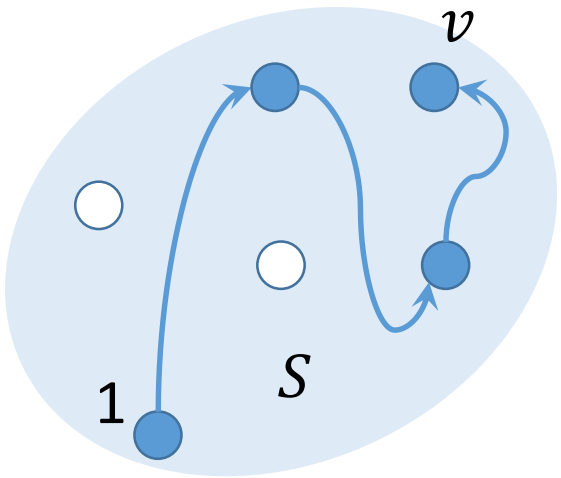



```
dp1p[0][(1 << 0)] = 0;
for( int s = 0; s < (1<<N); s++ ){
  for ( int u = 0; u < N; u++ ){
    if ( (s & (1 << u)) == 0 ) continue;
    for ( int v = 0; v < N; v++ ){
      chmin(dp1p[v][s|(1 << v)], dp1p[u][s] + G[u][v]);
    }
  }
}
```

問題１０　通学路

解法

- $\mathrm{dp1}[v][S]$ =すべての $S' \subset \mathrm{S},\ v \in S'$のうち、 $\mathrm{dp1p}[v][S']$の最小値
  - Sに含まれる頂点を0または1回だけ用いて作るパスのうち v で終わる最短のもの
- v を固定してdp1pから求めることができる

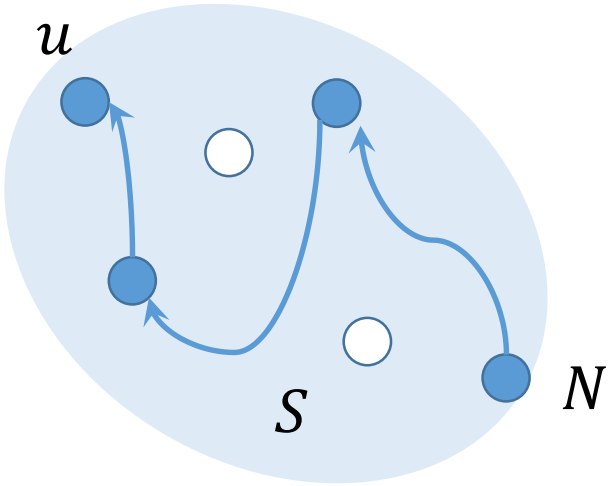



```
for ( int v = 0; v < N; v++ ){
  for ( int s = 0; s < (1<<N); s++ ){
    for ( int t = 0; t < N; t++ ){
      chmin(dp1[v][s|(1<<t)], dp1p[v][s]);
    }
  }
}
```

問題１０　通学路

解法

- 同様にして、N を始点、uを終点とするパスのうち S 内の頂点のみを使うパスの最小コストについても $\mathrm{dp}N[u][S]$として求めておく.



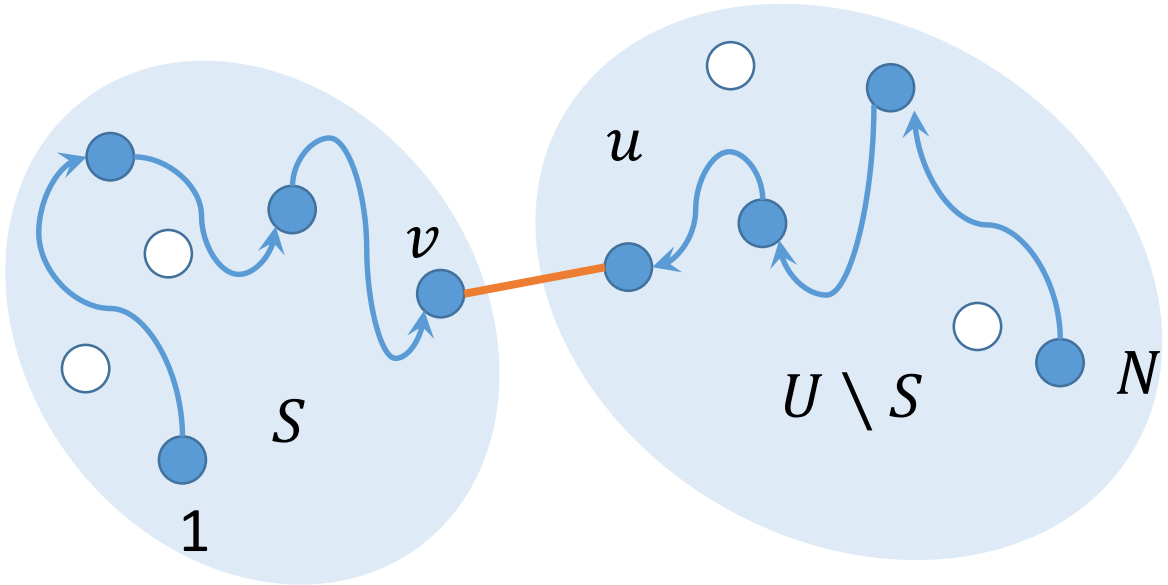
問題１０　通学路

解法

- u_i, v_iを接続するi番目の辺e_iについて答えを計算
- 全ての状態Sについて：

$$cost(e_i) + \min(dp1[u_i][S] + dpN[v_i][U \setminus S])$$

- O($M2^N$)



問題１１　モンスター甲子園　　（１０点）

問題１１　モンスター甲子園

概要

- あなたは３つのステータスがそれぞれ$a, \mathrm{s}, \mathrm{w}$であるモンスターを1体持っている.
- ステータスがa_i, s_i, w_iであるN体のモンスターと順に戦う.
- ２つ以上のステータスで，厳密に上回ると勝利.
- 各対戦の前にコイン１枚を使って自分のモンスターのステータスのうちひとつだけ大きさを１増やせる．その代わりに他の２つの数値のうちどちらかを１減らす必要がある（負にはできない）.
- モンスターの情報と使用できるコインの枚数Cが与えられるので，最大の勝利回数を求めよ.
- $N \leq 50, C \leq 5000,$各ステータス≤ 100

問題１１　モンスター甲子園

誤解法

- $dp[i][c][a][s][w]$: i回試合をして，c枚のコインを使用して，ステータスが$a, \mathrm{s}, \mathrm{w}$のときの最大の勝利回数とした，動的計画法

時間制限 or メモリ制限

問題１１　モンスター甲子園

考察１

- $sum = a + s + w$は一定なので，wの状態は持たなくてよい

$$w = sum - (a + s)$$

問題１１　モンスター甲子園

誤解法

- $dp[i][c][a][s]$:　i回試合をして，c枚のコインを使用して，ステータスが$a, \mathrm{s}, \mathrm{w} = \big(sum - (a+s)\big)$のときの最大の勝利回数，とした動的計画法

時間制限 or メモリ制限

問題１１　モンスター甲子園

解法

- $dp[i][j][a][s]$: i回試合をして，j勝して，ステータスが$a, \mathrm{s}, \mathrm{w} = \left(sum\ -(a+s)\right)$のときの最小のコインの枚数
  - 全ての試合を終えて、$dp[N][j][a][s]$がC以下である最大のjが答え

- O(N^2S^2)の動的計画法（S：ステータスの最大値）　**正解**
- O(N^2S^2)の01BFS　**正解**

問題１１　モンスター甲子園

動的計画法

- $dp[j][a][s]$:　j勝して，ステータスが$a, \mathrm{s}, \mathrm{w} = \left(sum\ -(a+s)\right)$のときの最小のコインの枚数
  動的計画法により，N回試合をして更新する（dpテーブルの$[i]$の次元は削減）

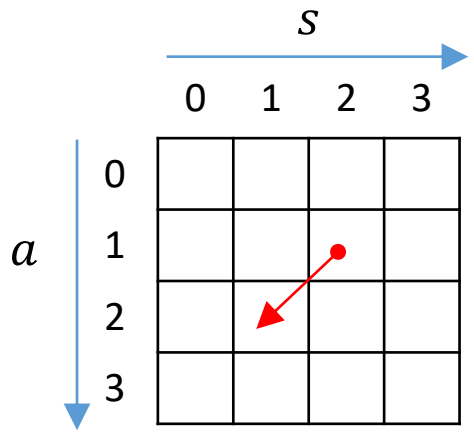
```
dp[0][a0][s0] = 0;
for ( int i = 1; i <= N; i++ ){          // i回目からi+1回目の試合直前の状態を決定する
    for ( int j = (i-1); j >= 0; j--){   // ※逆から回す
```

Part 1
可能なステータスの変更を行いdp[j][a][s]を更新
a, s, wから２つ選ぶ６つの組み合わせ

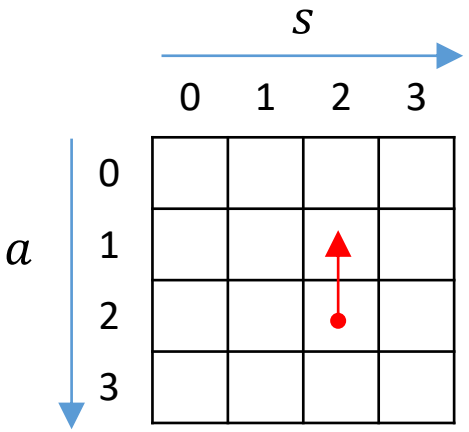
Part 2
勝てる状態についてdp[**j+1**][a][s]を更新

問題１１　モンスター甲子園

動的計画法 Part 1 (6パタンのうちの２つの例)




```
// a <- s
for(int a = 0; a <= sum; a++){
    for(int s = 0; s <= sum; s++ ){
        int w = sum - (a + s);
        if ( valid(a, s, w) && valid(a+1, s-1, w  ) )
            chmin(dp[j][a+1][s-1], dp[j][a][s] + 1);
        …
```



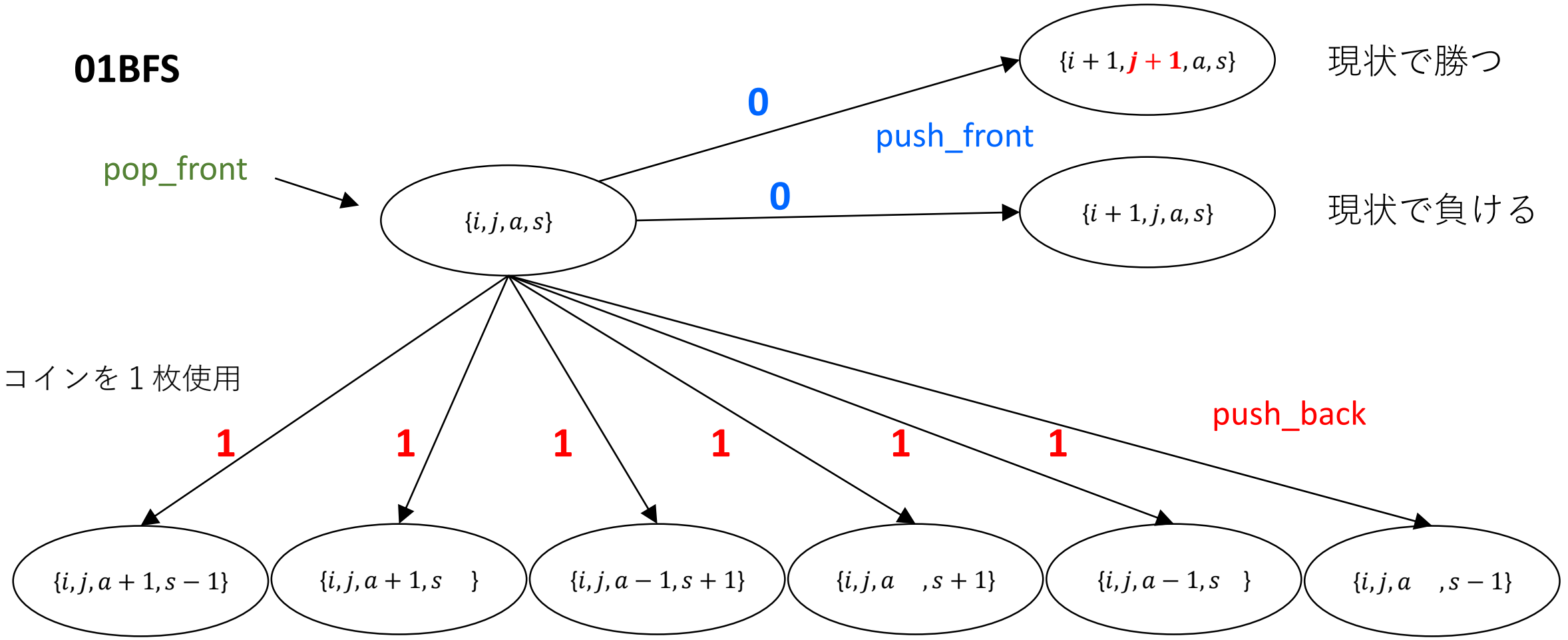

```
// w <- a
for ( int a = sum; a >= 0; a--) {
    for(int s = 0; s <= sum; s++ ){
        int w = sum - (a + s);
        if ( valid(a, s, w) && valid(a-1, s  , w+1) )
            chmin(dp[j][a-1][s  ], dp[j][a][s] + 1);
        …
```

問題１１　モンスター甲子園

動的計画法 Part 2

```
for(int a = 0; a <= sum; a++ ){
    for(int s = 0; s <= sum; s++ ){
        int w = sum - (a + s);
        if ( valid(a, s, w) && win(A, S, W, a, s, w) ) { // i番目の敵に勝てる？
            chmin(dp[j+1][a][s], dp[j][a][s]);
        }
    }
}
```

問題１１　モンスター甲子園



問題１２　平方連続部分文字列　（１２点）

問題12　平方連続部分文字列

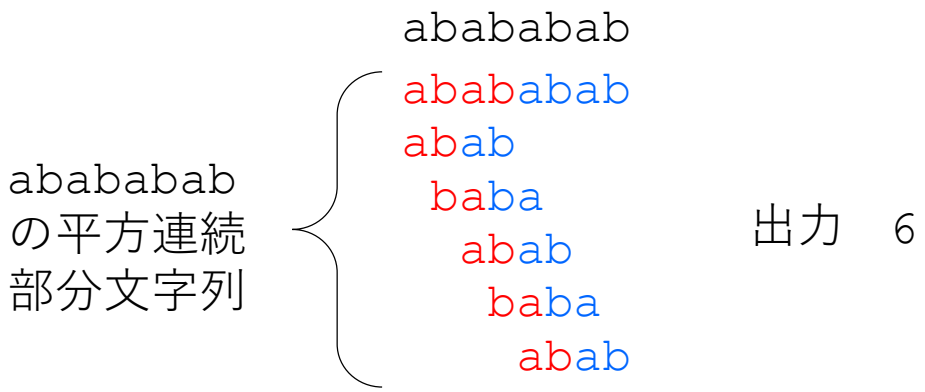
概要

- 文字列strに現れる部分文字列で，同じ部分文字列が 2 回続けて現れるもの（平方連続部分文字列：square factor）をすべて見つけ，その数を求める．
- 3 ≦ strの長さ ≦ 100,000

問題12　平方連続部分文字列

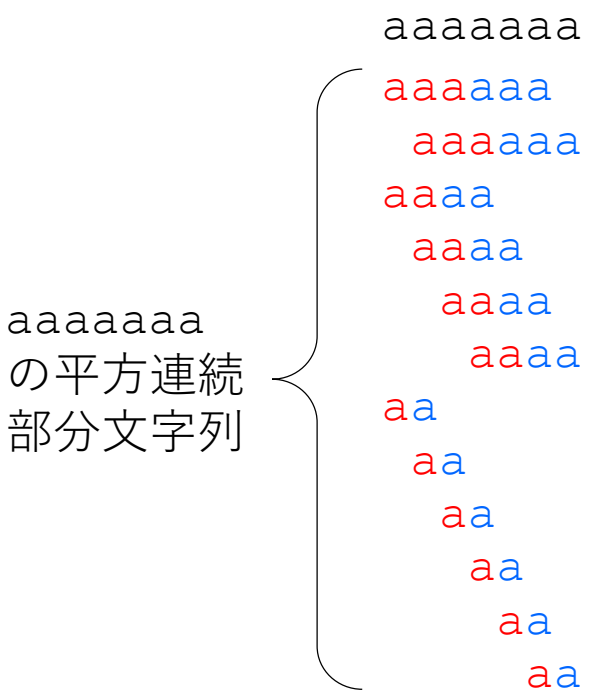
入出力例

入力　abababab



出力　6

入力　aaaaaaa



出力　12

問題12　平方連続部分文字列

解法

- strの長さの上限は100,000なので，愚直に文字を比べていくとO(N^3)で時間切れ．
- ローリングハッシュで解くと、O(N^2)で時間切れ．
- 想定解法は，分割統治法を使ったもの．　O($N \log N$)

分割統治法による解法

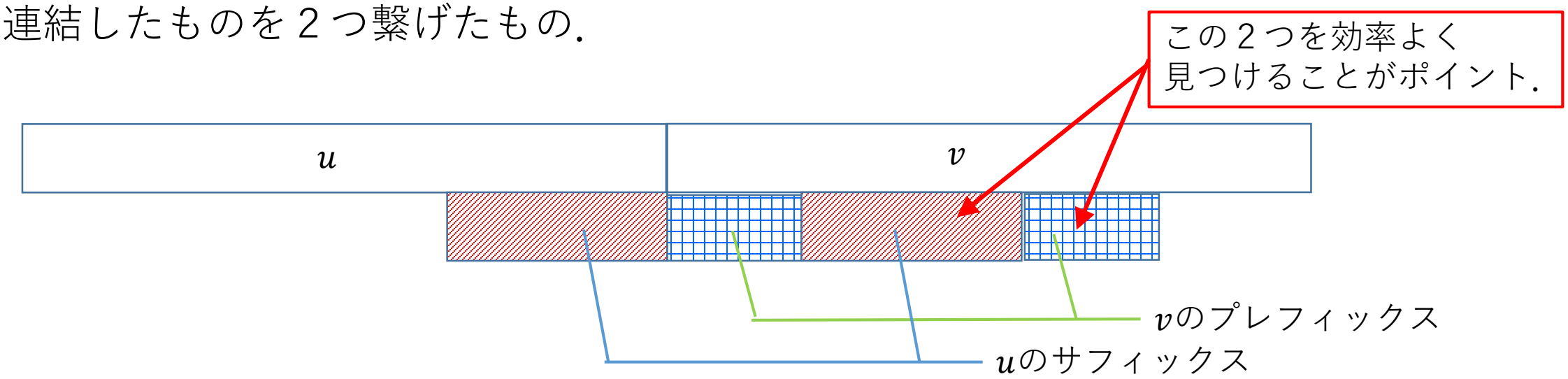
1. 文字列strを真ん中で分割．
2. 左半分と右半分それぞれの中に現れる平方連続部分文字列の個数を，
   分割統治法を使って再帰的に求める．
3. 左半分と右半分、両方にまたがる平方連続部分文字列の個数を求める．
4. 平方連続部分文字列の総数を出力する．

ステップ３の処理がポイント．

問題12　平方連続部分文字列

解法

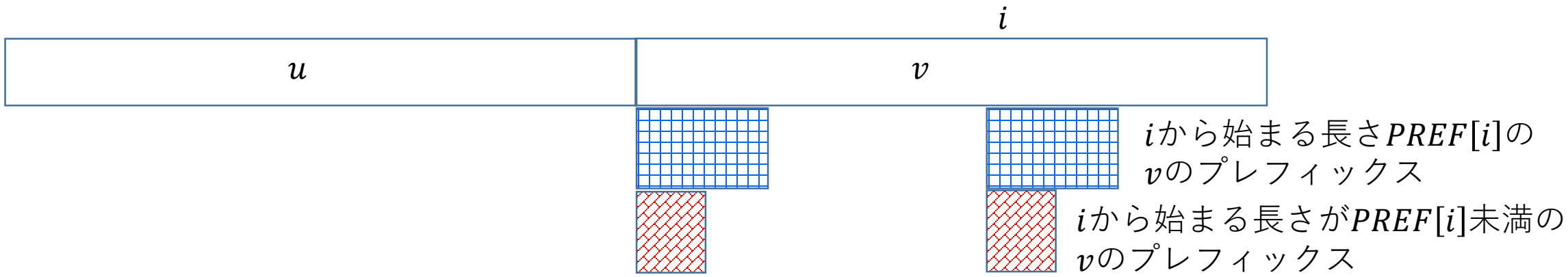
- 文字列wを真ん中で分けたとき，左半分をu，右半分をvとする．
- uとvにまたがる平方連続部分文字列を同じ文字列に分割する切れ目がある場所は，(1) uの中，(2) vの中，(3) uとvの切れ目のいずれか．
- (1)は，wを反転したものを考えると(2)と同様に処理できる．
- (2) に当てはまる平方連続部分文字列は，uのサフィックスとvのプレフィックスを連結したものを２つ繋げたもの．



問題12　平方連続部分文字列

解法

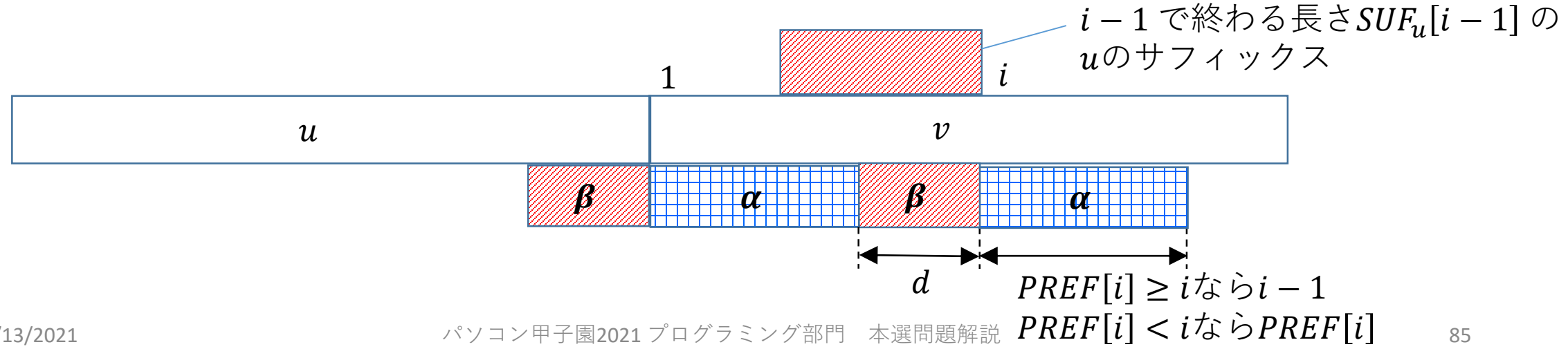
- vの位置iから始まる，vの最長のプレフィックスの長さを，各位置iについて記憶する．この列を$PREF$とする．
- vの位置iで終わる，uの最長のサフィックスの長さを，各位置iについて記憶する．この列をSUF_uとする．
- iから始まる長さ$PREF[i]$のvのプレフィックスのプレフィックスも，iから始まるvのプレフィックス（SUF_uについても同様のことが言える）．



問題12　平方連続部分文字列

解法

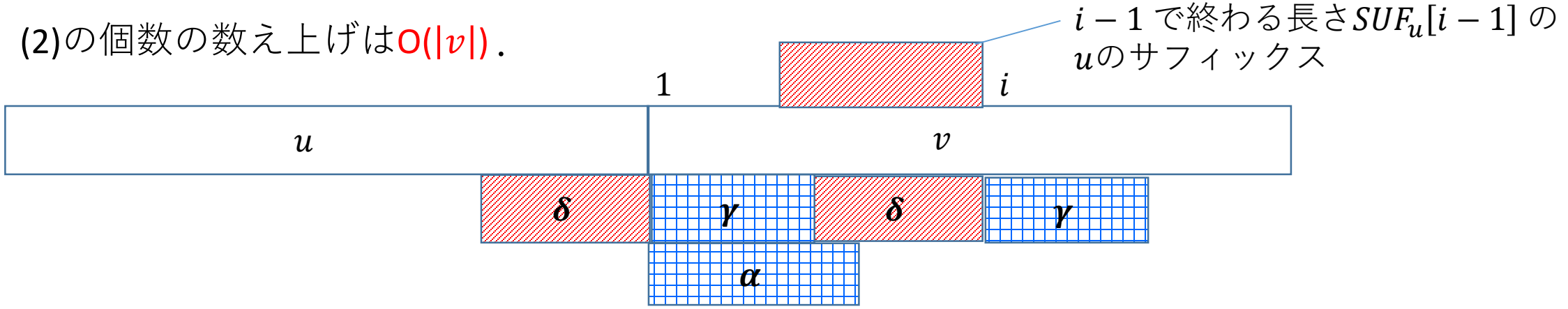
- vの各位置iについて，$PREF[i]$と$SUF_u[i-1]$を使って，切れ目がvの中にある平方連続部分文字列の個数を求める（ただし，vの先頭位置を1として，$i>1$から考える）．
- $PREF[i] \geq i$なら，iから始まる長さ$i-1$のvのプレフィックスをαとする．
- $PREF[i] < i$なら，iから始まる長さ$PREF[i]$のvのプレフィックスをαとする．
- $SUF_u[i-1]$が図の長さd以上なら，$i-1$で終わる長さdのuのサフィックスをβとする．
- このとき，長さ$2(d+|\alpha|)$の平方連続部分文字列$\beta\alpha\beta\alpha$が得られる（$|\alpha|$はαの長さ）．



問題12　平方連続部分文字列

解法

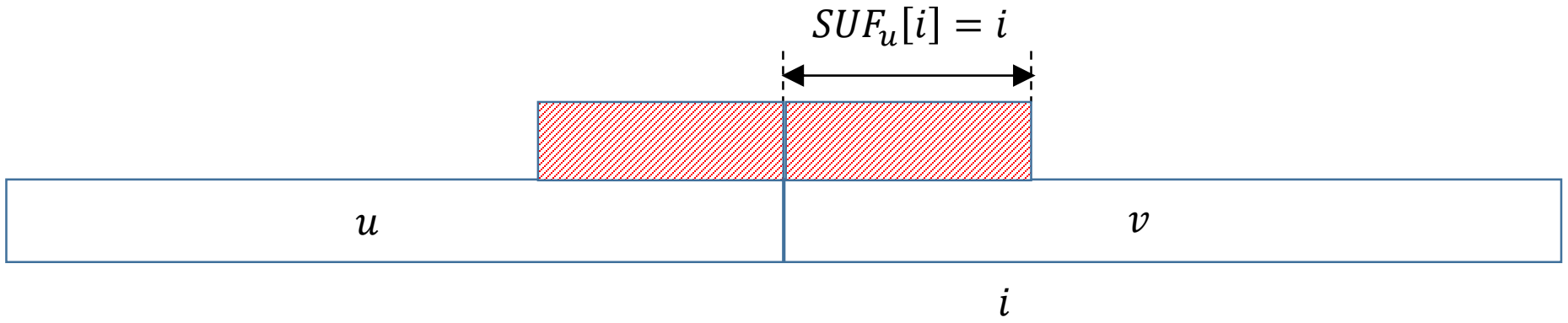
- αのプレフィックス（γとする）も，vのプレフィックス．
- 図の２つのγに挟まれた文字列の長さが$SUF_u[i-1]$以下なら，その文字列はuのサフィックス（δとする）．
- δの長さが$SUF_u[i-1]$ になるまでγを縮めても，平方連続部分文字列$\delta\gamma\delta\gamma$が得られる．
- $SUF_u[i-1]-d$が，位置iを考えたときに得られる(2)の平方連続部分文字列の個数（ただし、$SUF_u[i-1]=i-1$のときは，δの長さを$SUF_u[i-1]$にするとγが空文字列になってしまうので，(2)の個数は$SUF_u[i-1]-d-1$）．
- (2)の個数の数え上げはO(|v|)．



問題12　平方連続部分文字列

解法

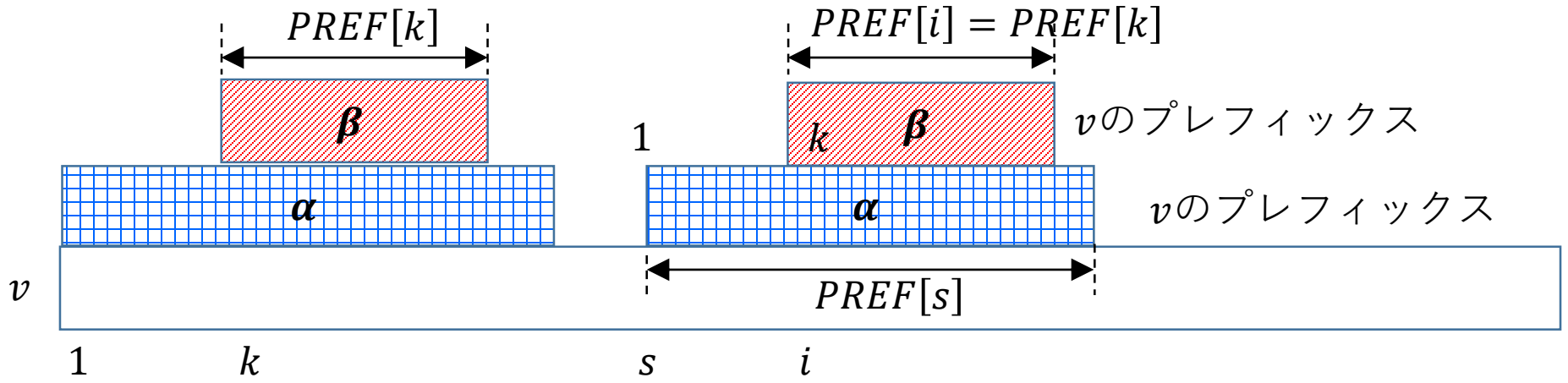
- 平方連続部分文字列の切れ目が，(3) uとvの切れ目のときは，$SUF_u[i] = i$であるiの個数が，(3)に当てはまる平方連続部分文字列の個数.
- 個数の数え上げは，SUF_uを計算するときに行える（すべて数えるのにO($|v|$)）.



問題12　平方連続部分文字列

解法

- $PREF$はO($|v|$)で求まる（$|v|$はvの長さ）.
- $PREF[i]$を求めるときに，$i > j$である$PREF[j]$はすべて得られているとする.
- $i > s$で$s + PREF[s]$が最大となるとき，$k = i - s + 1$とすると、$PREF[k] + k \leq PREF[s]$（図の文字列βが文字列αの末尾まで達しない）なら，$PREF[i] = PREF[k]$となる.
- sを覚えておけば同じ文字を調べる必要がないので，$PREF$はO($|v|$) で求まる.
- SUF_uも，分割前の文字列wを反転した文字列に対して，$PREF$と同様にして求まる.



問題１３　ツリーキーボード　（１２点）

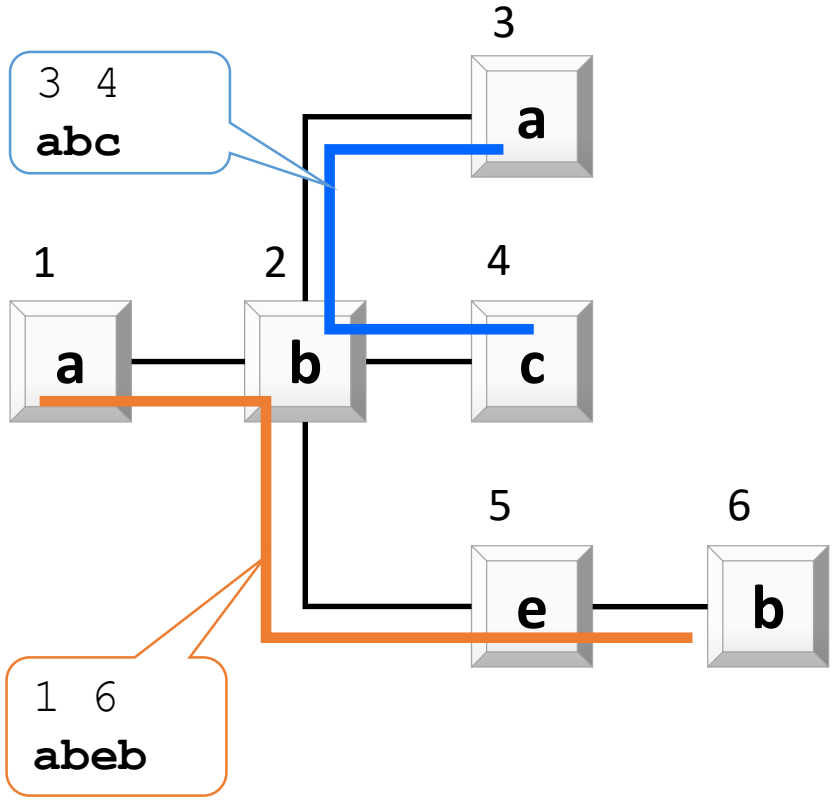
問題１３　ツリーキーボード

概要

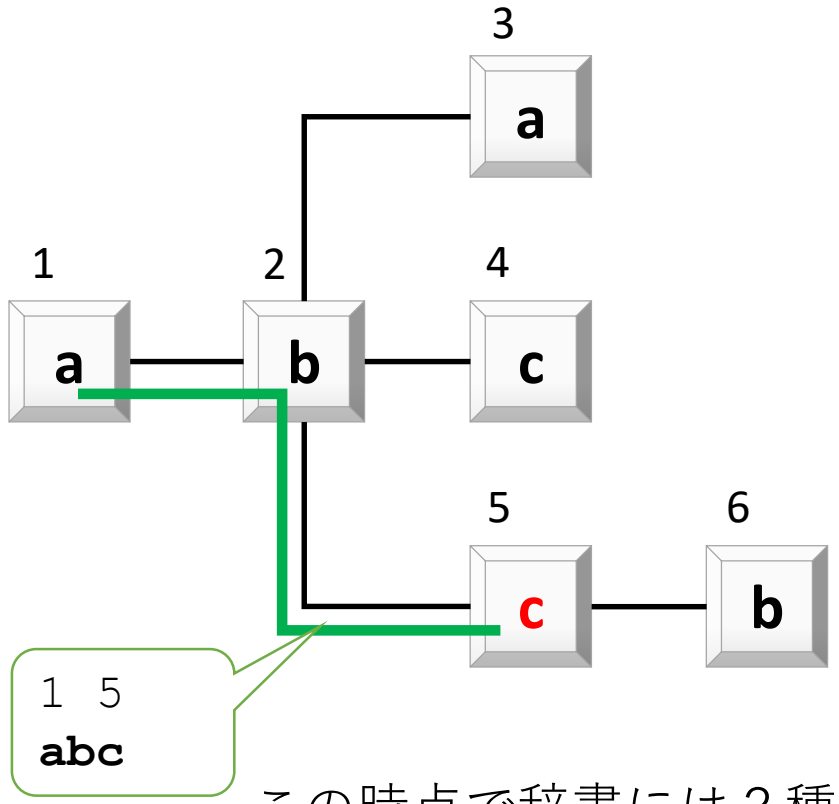
- N頂点の木の形をしたキーボードが与えられる．
- 各キー（ノード）には，１つの文字が割り当てられており，設定によっていつでも変更できる．
- ２つのキーs, tを押すと，sからtへのパスに含まれるキー（文字）を順番に繋げた文字列が辞書に書き込まれる．
- 以下の２種類の操作を含むQ個のクエリに対処せよ：
  1. ２つのキーを入力して，文字列を辞書に書き込む．この直後の辞書のサイズを報告する
  2. 指定されたキーの文字を変更する
- $N \leq 100{,}000, Q \leq 100{,}000$

問題１３　ツリーキーボード

入出力例



キー5をcに変更



この時点で辞書には２種類の文字列が格納されている

問題１３　ツリーキーボード

考察と方針

- 文字列そのものを辞書に格納することはできない
- ハッシュ値を格納する

- 大きい木に対するクエリ処理（使えそうなテクニック）
  - LCA
  - HL分解
  - オイラーツアー
  - 遅延伝播セグメント木

問題１３　ツリーキーボード

前処理

- 根付き木を生成
  - 根からの深さ$depth[v]$を計算
  - LCAに必要な要素を計算
  - オイラーツアーを行いノードの列を生成
- 根から（葉に向かって）vへ辿ったときに得られる文字列のハッシュ値を計算：ord
- 葉から（根に向かって）vへ辿ったときに得られる文字列のハッシュ値を計算：rev

問題１３　ツリーキーボード

文字列の入力処理

- ord とrevから，指定された区間から得られる文字列のハッシュ値$hash$を求める
- $hash$をsetに挿入し，その時点のサイズを出力する

キーの変更の処理

- キーを変更で影響があるのはord, revのみ
- vが変更されたときに影響がある頂点は，vを根とする部分木に含まれる頂点のみ
- オイラーツアーで得られた列に対して，区間addと１点取得のセグメント木２つで管理

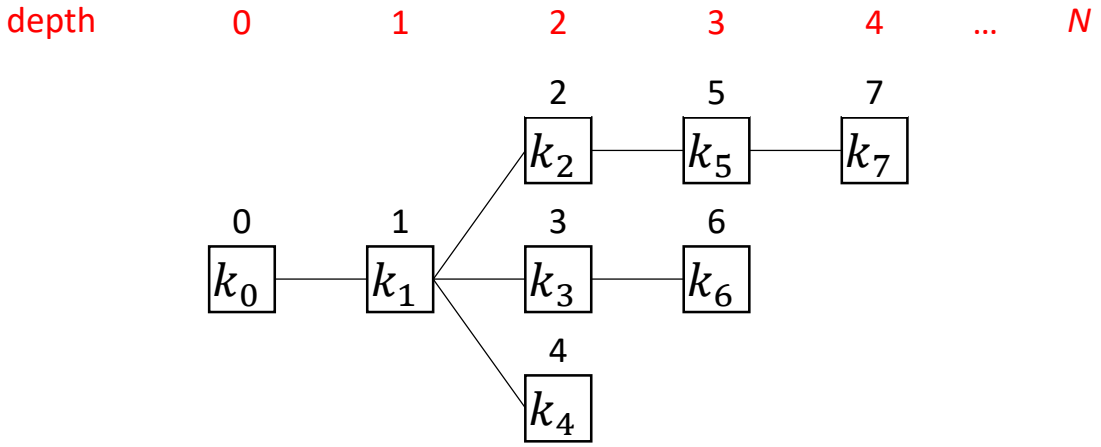
$$ord \rightarrow ordSeg$$

$$rev \rightarrow revSeg$$

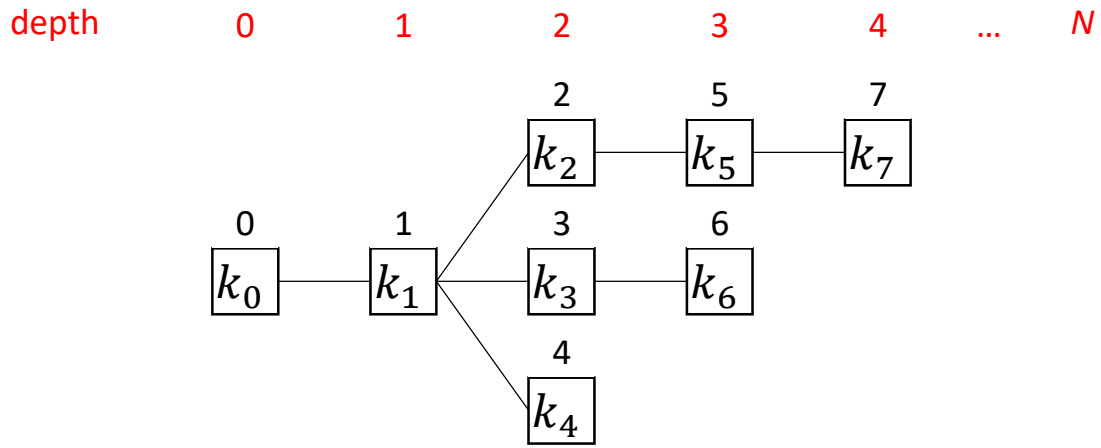
変更時の差分をそれぞれaddする（遅延評価が必要）

問題１３　ツリーキーボード

各ノードのハッシュ値の計算



$$ord[0] = k_0\beta^0$$
$$ord[1] = k_0\beta^0 + k_1\beta^1$$
$$ord[2] = k_0\beta^0 + k_1\beta^1 + k_2\beta^2$$
$$ord[3] = k_0\beta^0 + k_1\beta^1 + k_3\beta^2$$
$$ord[4] = k_0\beta^0 + k_1\beta^1 + k_4\beta^2$$
$$ord[5] = k_0\beta^0 + k_1\beta^1 + k_2\beta^2 + k_5\beta^3$$
$$ord[6] = k_0\beta^0 + k_1\beta^1 + k_3\beta^2 + k_6\beta^3$$
$$ord[7] = k_0\beta^0 + k_1\beta^1 + k_2\beta^2 + k_5\beta^3 + k_7\beta^4$$



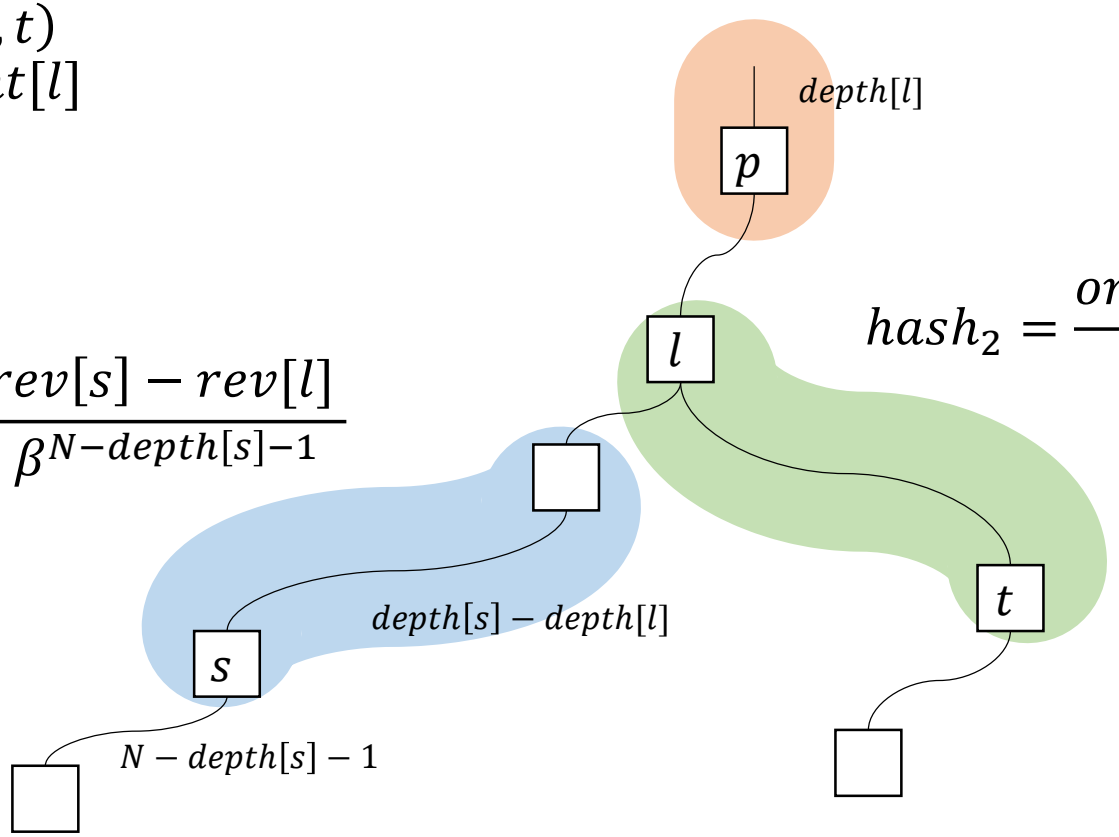
$$rev[0] = k_0\beta^7$$
$$rev[1] = k_0\beta^7 + k_1\beta^6$$
$$rev[2] = k_0\beta^7 + k_1\beta^6 + k_2\beta^5$$
$$rev[3] = k_0\beta^7 + k_1\beta^6 + k_3\beta^5$$
$$rev[4] = k_0\beta^7 + k_1\beta^6 + k_4\beta^5$$
$$rev[5] = k_0\beta^7 + k_1\beta^6 + k_2\beta^5 + k_5\beta^4$$
$$rev[6] = k_0\beta^7 + k_1\beta^6 + k_3\beta^5 + k_6\beta^4$$
$$rev[7] = k_0\beta^7 + k_1\beta^6 + k_2\beta^5 + k_5\beta^4 + k_7\beta^3$$

```
N - depth[v] - 1
```

問題１３　ツリーキーボード

指定文字列(s,t)に対するハッシュ値の計算

$l: LCA(s,t)$
$p: parent[l]$



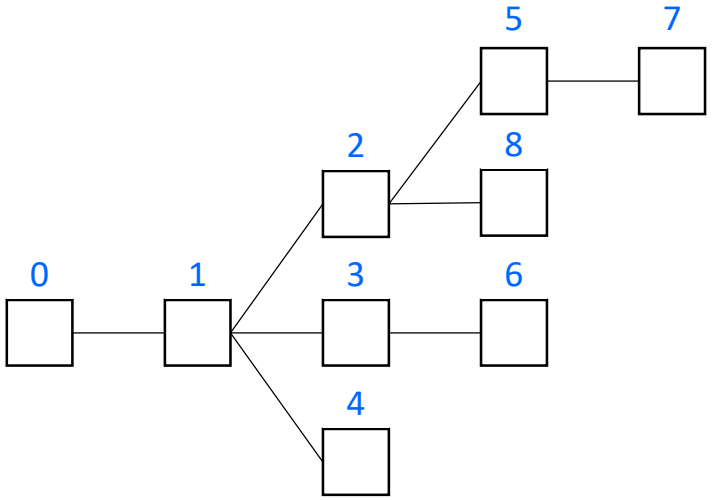
$$hash_1 = \frac{rev[s] - rev[l]}{\beta^{N-depth[s]-1}}$$

$$hash_2 = \frac{ord[t] - ord[p]}{\beta^{depth[l]}}$$

$$hash = hash_1 + hash_2 \times \beta^{depth[s]-depth[l]}$$

問題１３　ツリーキーボード

文字の変更: オイラーツアー



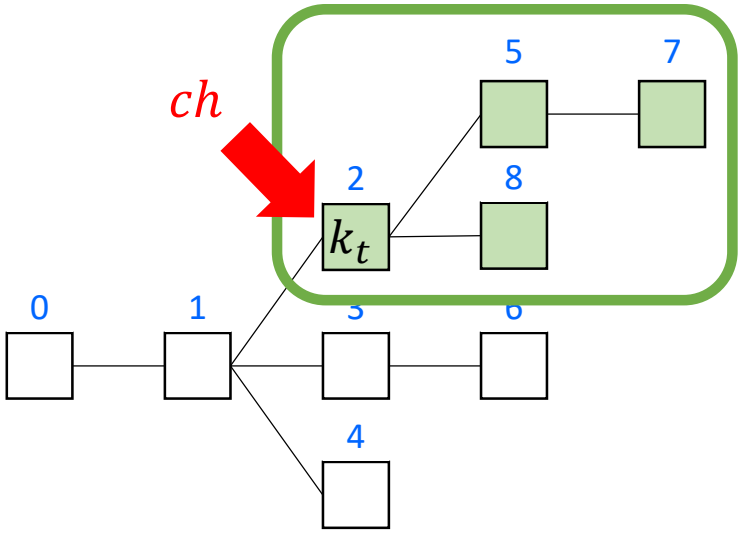
	0	1	2	3	4	5	6	7	8	9
vs	0	1	2	5	7	8	3	6	4	

	0	1	2	3	4	5	6	7	8	9
ls	0	1	2	6	8	3	7	4	5	
rs	9	9	6	8	9	5	8	5	6	

```
void euler_tour(int u, int p){
  ls[u] = vs.size();
  vs.push_back(u);
  for ( int v : G[u] ){
    if (v != p ) euler_tour(v, u);
  }
  rs[u] = vs.size();
}
```

問題１３　ツリーキーボード

文字の変更: 文字の変更



ch

	0	1	2	3	4	5	6	7	8	9
ls	0	1	2	6	8	3	7	4	5	
rs	9	9	6	8	9	5	8	5	6	

$$ordSeg.add(ls[t], rs[t], \beta^{depth[t]} \times (ch - k_t))$$
$$revSeg.add\left(ls[t], rs[t], \beta^{N-1-depth[t]} \times (ch - k_t)\right)$$

	0	1	2	3	4	5	6	7	8	9
vs	0	1	2	5	7	8	3	6	4	