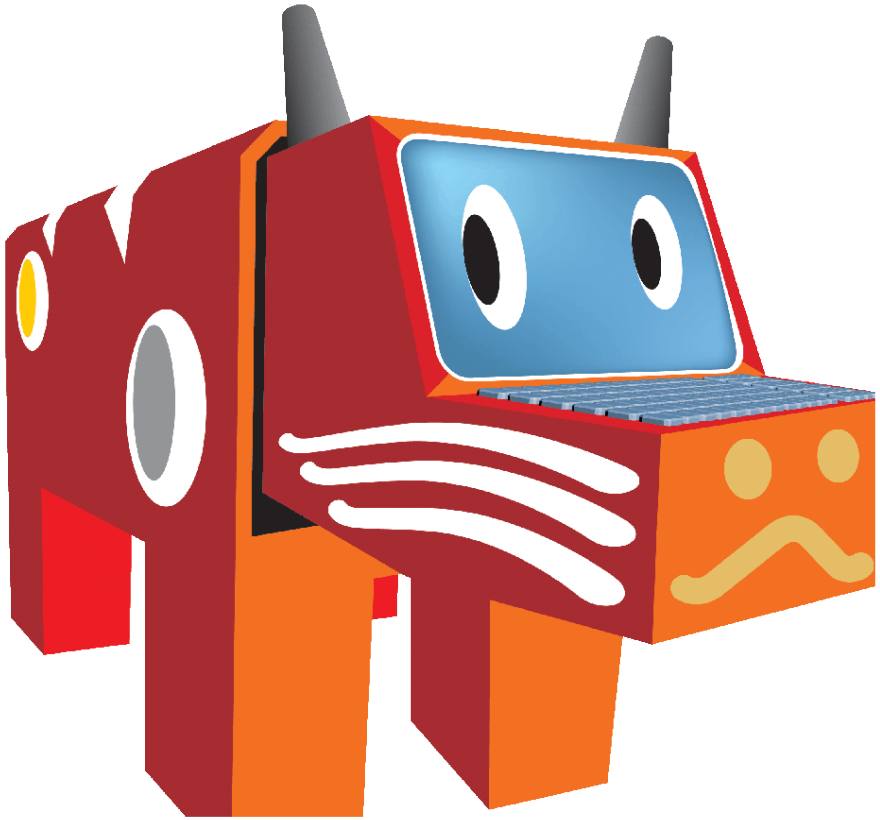
AIZU
THE UNIVERSITY OF AIZU

パソコン甲子園２０１７予選

プログラミング部門　解説



会津大学

概要

AIZU
THE UNIVERSITY OF AIZU

- 参加チーム数６３２チーム

#	タイトル	分野	実装	思考	得点	正答数
1	お年玉	入出力・四則演算	☆		3	５８１
2	買い物	四則演算・条件判定	☆	☆	4	５３３
3	９月Ｘ日	剰余	☆	☆	4	５４０
4	エルの予約	条件判定・ループ処理	★	★	5	３３４
5	電線	最大公約数	☆	★☆	6	２２２
6	トランポリン	貪欲法	★★	★★	１１	７７
7	積み荷の配置	動的計画法	★★☆	★★★	１１	１８
8	ダンジョン	AdHoc	★★	★★★	１１	１９
9	人気のユーザ名	データ構造	★★★	★★★★	１４	9
10	道路網改修	グラフ理論	★★★☆	★★★☆	１４	5
11	ネットワークの課金システム	データ構造	★★★★★	★★★★★	１７	2

問１　お年玉

AIZU
THE UNIVERSITY OF AIZU

問題概要

- Ａ君がもらったお年玉の金額aと、Ｂ君がもらったお年玉の金額bが与えられる.
- ２人のお年玉を足し合わせ、半分ずつに分けると、いくらになるか求める.
- 1000≦a,b≦50000. ただし、aとbどちらの金額も1000の倍数である.

総評・解法

- 提出数　９７７、正答数　５８１.
- 変数宣言、和と商の計算、標準入出力処理が行えるかを問う最も基本的な問題である.
- int型変数二つに値を読み込み、足して割ったものを出力する.
- 多かった間違い
  - (a+b)/2を計算するとき、カッコをつけ忘れている.
  - 解答以外の余計な文字列を出力している.
  - main関数の返り値をvoidで宣言しているのにreturn 0とするなど、基本的な文法ミス.

問１　お年玉

AIZU
THE UNIVERSITY OF AIZU

解答例（C/C++）

```
#include <stdio.h>

main() {
  int a, b;
  scanf("%d %d", &a, &b);
  printf("%d¥n", (a+b)/2);
}
```

問１　お年玉

AIZU
THE UNIVERSITY OF AIZU

解答例（Ｊａｖａ）

```
import java.io.*;
import java.util.*;

class P01 {
    void solve(){
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
        System.out.println((a+b)/2);
    }
    public static void main(String[] a) { new P01().solve(); }
}
```

問２　買い物

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 自分の手持ちの金額mと友達の手持ちの金額f、買いたい本の値段bが与えられる.
- 本を買うために、友達から借りなければならない金額を求める.
- ただし、自分の手持ちの金額と友達の手持ちの金額を合わせても、本が買えない場合は「NA」と出力する.
- 0≦m,f≦10000. 100≦b≦20000.

問２　買い物

AIZU
THE UNIVERSITY OF AIZU

総評・解法

- 提出数　１５７１、正答数　５３３.
- 入出力処理、四則演算に加えて、条件分岐を使ったプログラムを実装できるかが問われている.
- m ≧ bをみたすなら「０」、そうではなくてm + f ≧ bをみたすなら「b−m」、どちらもみたさないなら「NA」と出力する.
- 多かった間違い
  - 以上(≧)ではなくて、より大きい(>)で判定している.

問２　買い物

AIZU
THE UNIVERSITY OF AIZU

解答例（C/C＋＋）

```
#include <stdio.h>

main() {
    int m, f, b;
    scanf("%d %d %d", &m, &f, &b);

    if      ( m     >= b ) printf("0¥n");
    else if ( m + f >= b ) printf("%d¥n", b - m );
    else printf("NA¥n");
}
```

問２　買い物

AIZU
THE UNIVERSITY OF AIZU

解答例（Ｊａｖａ）

```
import java.io.*;
import java.util.*;

class P02 {

    void solve(){
        Scanner sc = new Scanner( System.in );
        int m = sc.nextInt();
        int f = sc.nextInt();
        int b = sc.nextInt();

        if ( m + f >= b ) System.out.println( Math.max( 0, b - m ) );
        else System.out.println("NA");
    }

    public static void main( String[] a ) {new P02().solve(); }
}
```

問３　９月X日

AIZU
THE UNIVERSITY OF AIZU

問題概要

- ２０１７年９月の日にちXが与えられたとき、その日の曜日を出力する.
- 月曜日は「mon」、火曜日は「tue」、水曜日は「wed」、木曜日は「thu」、金曜日は「fri」、土曜日は「sat」、日曜日は「sun」と出力する.
- 1≦X≦30.

問３　９月X日

AIZU
THE UNIVERSITY OF AIZU

総評

- 提出数　１０３５、正答数　５４０.
- 条件分岐や剰余算を使ったプログラムを実装できるかが問われている.
- 多かった間違い
  - 日にちの剰余算の間違い.
  - １日から３０日まですべての条件分岐を書いて、そのうちいくつかの曜日のスペルミス

解法

- 曜日を出力するための文字列配列を用意する.
- ９月１日が「fri」なので、配列の０番目が「thu」.
- 与えられた日にちを7で割った余りで文字列配列を参照し、出力する.

問３　９月X日

AIZU
THE UNIVERSITY OF AIZU

解答例（C/C＋＋）

```
#include <stdio.h>
const char* wd[7] = { "thu", "fri", "sat", "sun", "mon", "tue", "wed" };
main() {
    int d;
    scanf("%d", &d);
    printf("%s¥n", wd[d%7]);
}
```

問３　９月X日

AIZU
THE UNIVERSITY OF AIZU

解答例（Ｊａｖａ）

```
import java.io.*;
import java.util.*;

class P03 {
    static final String[] wd = { "thu", "fri", "sat", "sun", "mon", "tue", "wed" };

    void solve(){
        Scanner sc = new Scanner( System.in );
        int d = sc.nextInt();
        System.out.println( wd[d%7] );
    }

    public static void main( String[] a ) {new P03.solve(); }
}
```

問４　エルの予約

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 新しい予約の開始時刻aと終了時刻b、すでに存在するN個の予約の開始時刻s_iと終了時刻f_iが与えられる.
- 新しい予約が、すでに存在する予約と重複しないか求める.
- ただし、ある予約の終了時刻と、別の予約の開始時刻が同じ場合は重複しないと考えて良い.
- $0 \leqq a < b \leqq 100$, $0 \leqq N \leqq 100$, $0 \leqq s_i < f_i \leqq 100$.

問４　エルの予約

AIZU
THE UNIVERSITY OF AIZU

総評

- 提出数　１８２７、正答数　３３４.
- 条件分岐やループを使ったプログラムを実装できるかが問われている.
- 多かった間違い
  - 重複の条件の間違い.
  - ある予約の終了時刻と、別の予約の開始時刻が同じ場合に重複と判定.

解法

- すでにある予約N個すべてに対して、$b > s_i$ かつ $a < e_i$ となるものが一つでもあるか調べる.

問４　エルの予約

AIZU
THE UNIVERSITY OF AIZU

解答例（C/C＋＋）

```
#include <stdio.h>

main() {
    int i, a, b, n, s, e;

    scanf(“%d %d %d", &a, &b, &n);
    for ( i=0; i<n; ++i ) {
        scanf("%d %d", &s, &e);
        if ( b > s && a < e ) {
            printf("1¥n");
            return 0;
        }
    }

    printf("0¥n");
}
```

問４　エルの予約

AIZU
THE UNIVERSITY OF AIZU

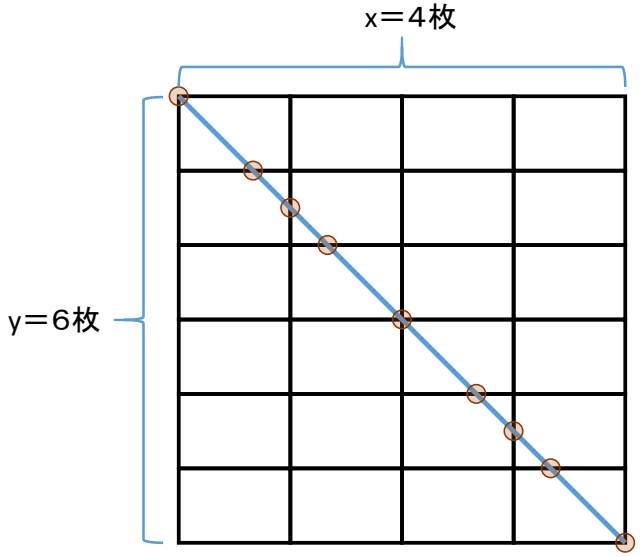
解答例（Ｊａｖａ）

```
import java.io.*;
import java.util.*;

class P04 {

    void solve(){
        Scanner sc = new Scanner( System.in );
        int a = sc.nextInt();
        int b = sc.nextInt();
        int n = sc.nextInt();

        for ( int i=0; i<n; ++i ) {
            int s = sc.nextInt();
            int e = sc.nextInt();
            if ( b > s && a < e ) {
                System.out.println( "1" );
                return;
            }
        }
        System.out.println( "0" );
    }

    public static void main( String[] a ) {new P04.solve(); }
}
```

問５　電線

問題概要

- 横の長さが２ｍ、縦の長さが１ｍの長方形のパネルが、壁一面に敷き詰められている.
- パネルの枚数は、横方向にx 枚、縦方向にy 枚.
- 壁の左上隅から右下隅まで、まっすぐ電線を張るとき、パネルの継ぎ目と電線の交点の数を求める.
- 1≦x,y≦1000.
- 右図の場合、交点の数は９と数える.



図：パネルと電線の交点

問５　電線

AIZU
THE UNIVERSITY OF AIZU

総評

- 提出数８５６、正答数 ２２２.
- 縦方向と横方向の境界が重なる場所（角）で重複した交点を除く方法を考える問題.
- 多かった間違い
  - 重複を除くための計算式の間違い.

解法

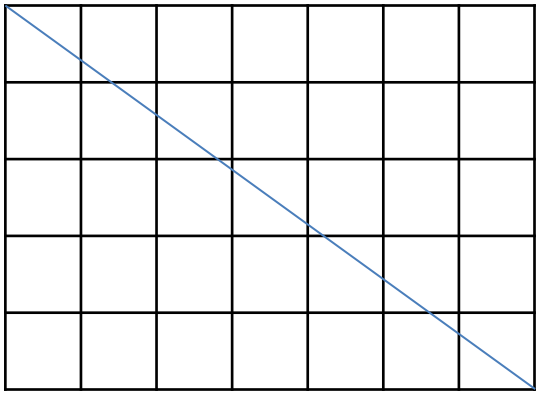
- ｘ + ｙ − (xとyの最大公約数 − 1) を出力.

問５　電線

考え方

仮に、パネルの横の長さ＝縦の長さ＝ １ m、x ≧ y として考える.

x=7,y=5の場合

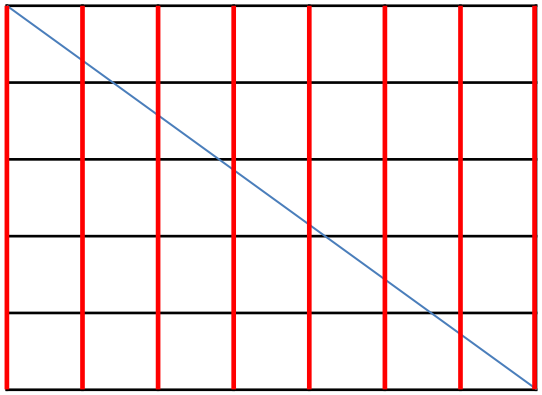


答えは１ ２

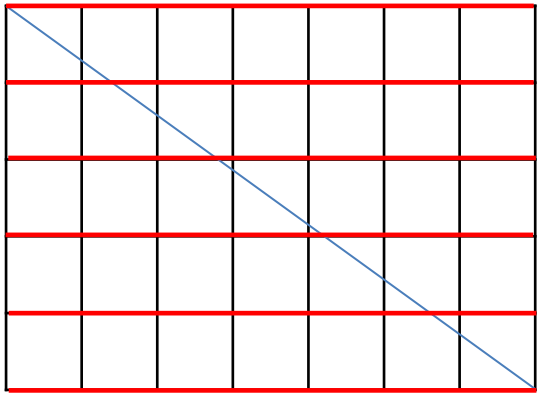
問５　電線

考え方

仮に、パネルの横の長さ＝縦の長さ＝ １ m、x ≧ y として考える.



縦方向の継ぎ目の線と電線は
x+1回交わる(x=7なら8回).



横方向の継ぎ目の線と電線は
y+1回交わる(y=5なら6回)

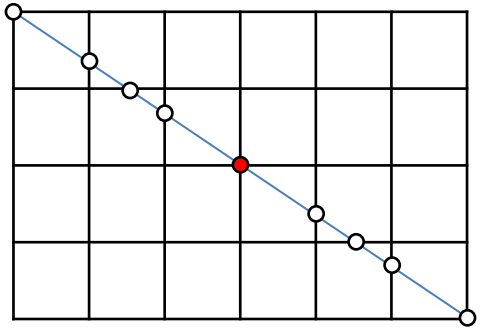
左上隅と右下隅で重複してカウントしているので、
全部合わせて (x+1) + (y+1) − 2 = <u>x + y = 12</u> が答え？

問５　電線

考え方

仮に、パネルの横の長さ＝縦の長さ＝ １ m、x ≧ y として考える.

x=6,y=4の場合



答えは９　≠ x + y = 10

５つ目の交点は縦と横の継ぎ目の分、両方カウントしている.
こういった、途中で発生する重複を除くにはどうすればよいか？

途中で発生する重複の個数は、xとyの最大公約数と同じ.

x + y − (xとyの最大公約数 − 1) が答え.

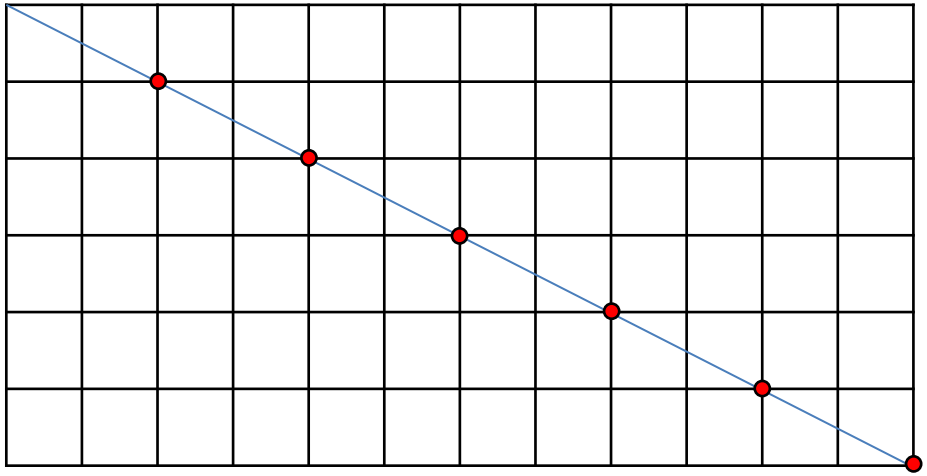
問５　電線

AIZU
THE UNIVERSITY OF AIZU

最大公約数の図形的な解釈

仮に、パネルの横の長さ＝縦の長さ＝ １ m、x ≧ y として考える.

x=12, y=6の場合、最大公約数=6



xが12進む間、x,yが両方とも整数になるのは6回(最大公約数)になる

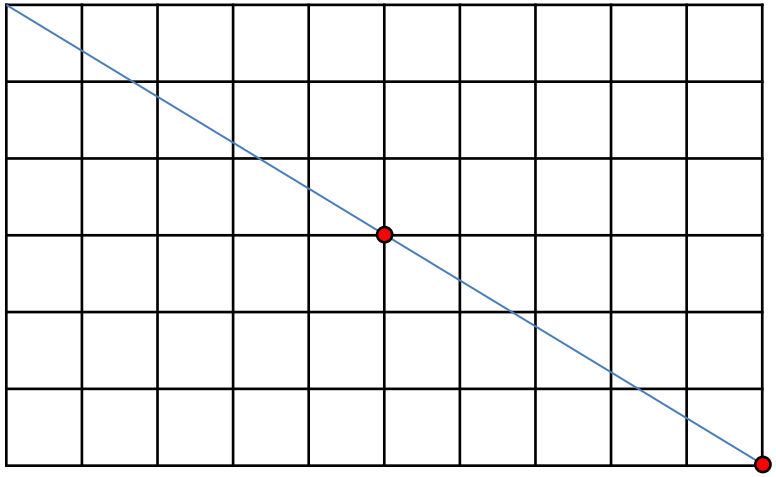
問５　電線

AIZU
THE UNIVERSITY OF AIZU

最大公約数の図形的な解釈

仮に、パネルの横の長さ＝縦の長さ＝ １ m、x ≧ y として考える.

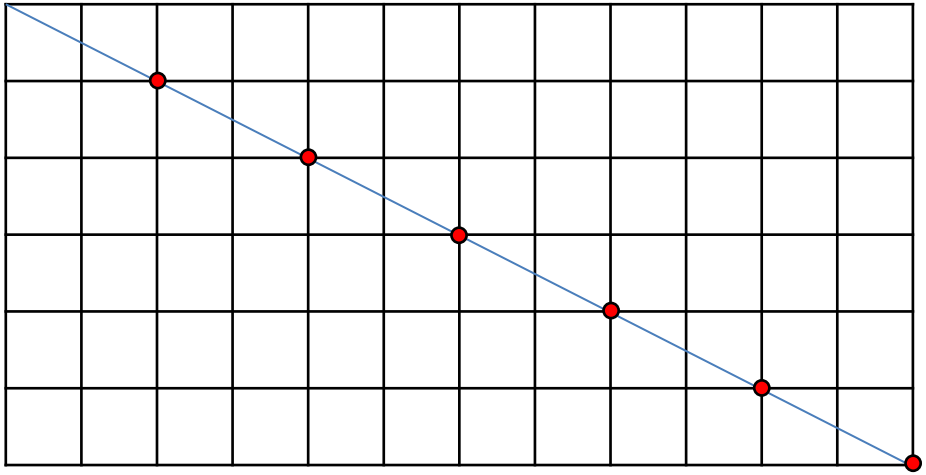
x=10, y=6の場合、最大公約数=2



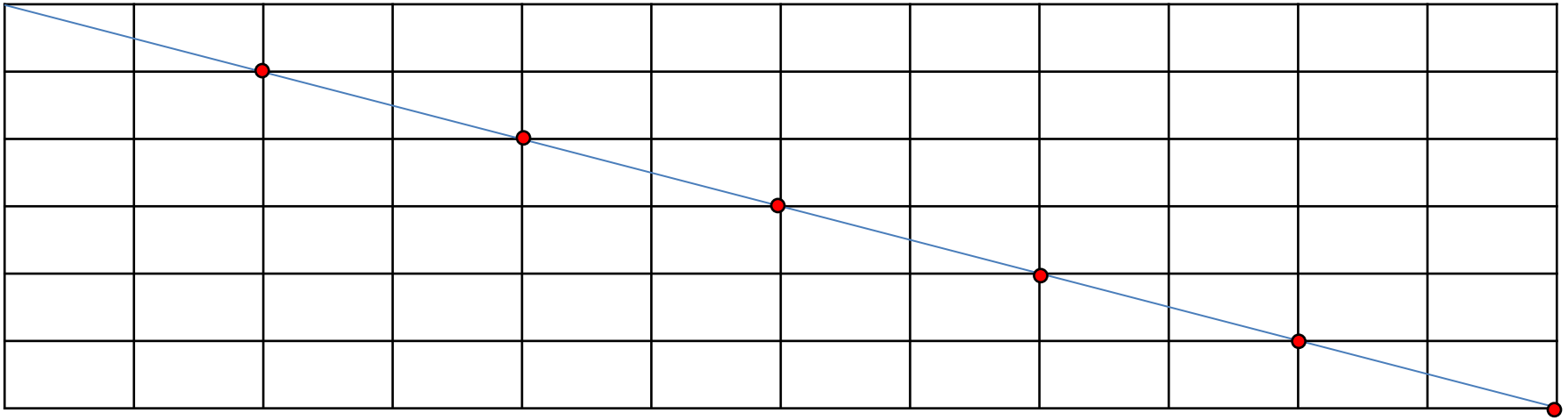
xが10進む間、x,yが両方とも整数になるのは2回(最大公約数)になる

問５　電線

最大公約数の図形的な解釈



パネルを横方向に引き伸ばしても結果は同じ.



問５　電線

AIZU
THE UNIVERSITY OF AIZU

解答例（C/C＋＋）

```
#include <stdio.h>

int gcd( int x, int y ){ return y ? gcd(y, x%y) : x; }

main(){
  int x, y;
  scanf(“%d %d”, &x, &y);
  printf(“%d¥n”, (x+1) + (y+1) - (gcd(x, y) - 1) – 2);
}
```

問５　電線

AIZU
THE UNIVERSITY OF AIZU

解答例（Ｊａｖａ）

```
import java.io.*;
import java.util.*;

class P05 {

    int gcd( int a, int b ) {
        if ( a < b ) return b%a == 0 ? a : gcd( a, b%a );
        return a%b == 0 ? b : gcd( b, a%b );
    }

    void solve(){
        Scanner sc = new Scanner( System.in );
        int x = sc.nextInt();
        int y = sc.nextInt();
        System.out.println( (x+1) + (y+1) - (gcd(x, y) - 1) - 2 );
    }

    public static void main( String[] a ) {new P05().solve(); }
}
```

問６　トランポリン

AIZU
THE UNIVERSITY OF AIZU

問題概要

- １０m間隔で直線上に設置されたトランポリンがN台ある.
- i番目のトランポリンで水平方向に跳ぶことができる最大距離d_iが与えられる.
- 左端のトランポリンから始めて、跳ぶことができる範囲にあるトランポリンに跳び移っていくことを繰り返す.
- 左端から右端のトランポリンまで行って、右端から左端のトランポリンに戻ってくることができるなら「yes」、できないなら「no」と出力する.
- $2 \leqq N \leqq 3 \times 10^5$, $1 \leqq d_i \leqq 10^6$.

問６　トランポリン

AIZU
THE UNIVERSITY OF AIZU

総評

- 提出数 731、正答数 77.
- 間違った判定方法を色々と思いつきやすい問題.
- 多かった間違い
  - 隣のトランポリンに行けないだけでnoと判定.
  - ２重ループで時間制限.
  - 右端まで行ってから左端に戻ってくる（復路）の判定ミス.

問６　トランポリン

AIZU
THE UNIVERSITY OF AIZU

解法

- 0番目から順番に、現在見ているi番目までのトランポリンで行ける最大距離d_{max}を更新していく.
- i番目のトランポリンで、$d_{max} < 10 \times i$になったら、その時点で「no」.
- 右端まで行けたら、トランポリンのランポリンで水平方向に跳ぶことができる最大距離d_iの配列を反転して同じことをする.
- 最後まで行けたら「yes」.

問６　トランポリン

AIZU
THE UNIVERSITY OF AIZU

解答例（C）

```
#include <stdio.h>
#define MAX_N 300000
int n, bounce[MAX_N];

int IsPossibleToReach() {
    int i, d_max = 0;
    for ( i=0; i<n; ++i ) {
          if ( d_max < 10*i ) return 0;
          if ( 10*i + bounce[i] > d_max ) d_max = 10*i + bounce[i];
    }
    return d_max >= 10*n ? 1 : 0;
}

main() {
    int i;  scanf("%d", &n);
    for ( i=0; i<n; ++i ) scanf("%d", &bounce[i]);

    if ( !IsPossibleToReach() ) { printf("no¥n"); return 0; }

    for ( i=0; i<n/2; ++i ) {
          int tmp = bounce[i];
          bounce[i] = bounce[n-1-i];
          bounce[n-1-i] = tmp;
    }

    if ( IsPossibleToReach() ) printf("yes¥n");
    else printf("no¥n");
}
```

問６　トランポリン

AIZU
THE UNIVERSITY OF AIZU

解答例（C++）

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

bool solve(vector<int> v){
  int maxp = 0;
  for ( int i = 0; i < v.size(); i++ ){
    if ( i*10 <= maxp ){
      maxp = max(maxp, v[i] + i*10);
    }
  }
  return 10*(v.size()-1) <= maxp;
}

main(){
  int N; cin >> N;
  vector<int> v, u;
  for ( int i = 0; i < N; i++ ){
    int x; cin >> x; v.push_back(x);
  }
  u = v;
  reverse(u.begin(), u.end());

  cout << (solve(v)&&solve(u)?"yes":"no") << endl;
}
```

問６　トランポリン

AIZU
THE UNIVERSITY OF AIZU

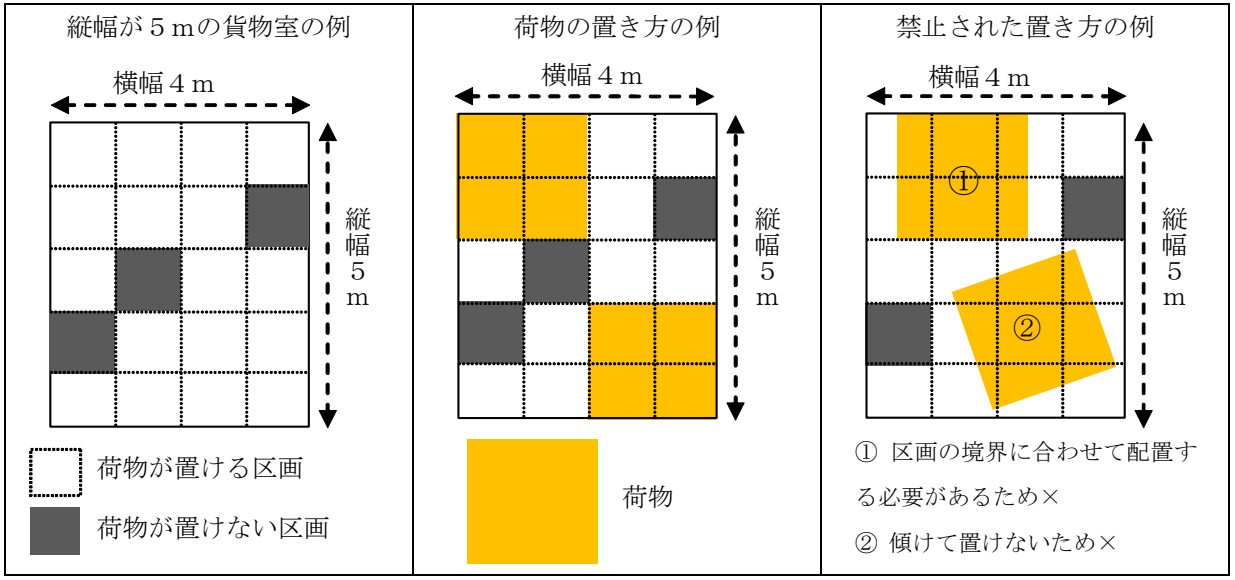
解答例（Ｊａｖａ）

```
import java.io.*;
import java.util.*;

class P06 {
    ArrayList<Integer> m_bounce = new ArrayList<Integer>();

    boolean IsPossibleToReach() {
          int current = 0;
          for ( int i=0; i<m_bounce.size(); ++i ) {
              if ( current < 10*i ) return false;
              current = Math.max( current, 10*i + m_bounce.get( i ) );
          }
          return current >= 10*m_bounce.size() ? true : false;
    }

    void solve(){
          Scanner sc = new Scanner( System.in );
          int n = sc.nextInt();
          for ( int i=0; i<n; ++i ) m_bounce.add( sc.nextInt() );

          if ( !IsPossibleToReach() ) { System.out.println( "no" ); return; }
          Collections.reverse( m_bounce );
          if ( IsPossibleToReach() ) System.out.println( "yes" );
          else System.out.println( "no" );
    }
    public static void main( String[] a ) {new P06().solve(); }
}
```

問題７　積み荷の配置

問題概要

- 横幅が４メートル、縦幅がHメートルの貨物室がある.
- 貨物室は縦横1メートル×１メートルの区画に区切られている.
- 荷物を置けない区画がN個与えられる.
- 貨物室に、縦横の長さが２メートルの荷物を、最大で何個置けるか.
- ただし、荷物は区画の境界に合わせて置く必要がある.
- $2 \leqq H \leqq 10^4$, $0 \leqq N \leqq 4 \times 10^4$.

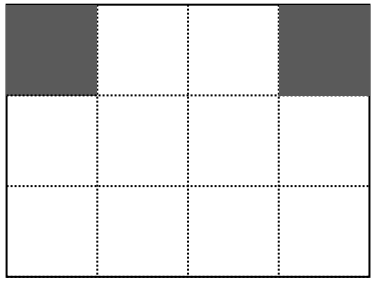


問題７　積み荷の配置

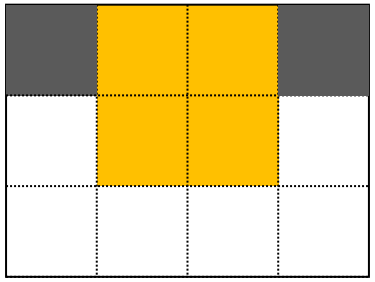
総評

- 提出数 ２８２、正答数 １８.
- 数え上げを行う効率的なアルゴリズムを実装できるかが問われている.
- 多かった間違い
  - 上から順番に配置してみる方法.

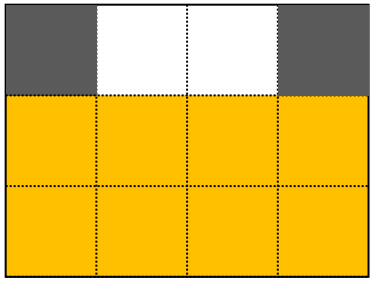
例えばこのような場合



左上から順番に調べていって空いているところから配置すると１つしか置けない



本当は２つ置ける

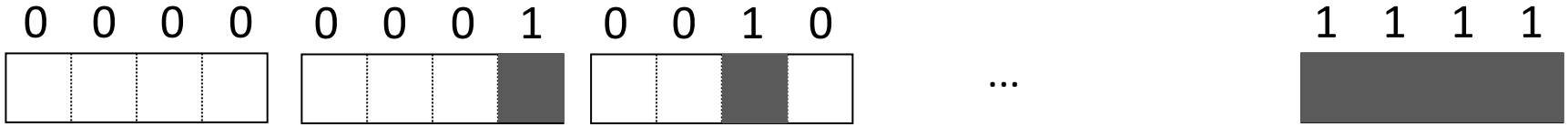


問題７　積み荷の配置

AIZU
THE UNIVERSITY OF AIZU

解法

- 荷物が置けない場所&すでに置いた場所のパターンをインデクスとしたビットDP.
- 一行に４区画なので、荷物が置けない場所は4ビットの整数(16通り)で表すことが出来る.



- 入力で与えられた、「荷物が置けない区画」と「すでに置いた荷物」をまとめて、荷物が置けない場所として、各行で４ビット整数で表すことが出来る.
- 荷物が置けない場所をインデクスとして、その置かれ方でそこまでに置けた荷物の最大値を更新していく.

問題７　積み荷の配置

AIZU
THE UNIVERSITY OF AIZU

解答例（C++）

```
#include <stdio.h>

#define MAX_H 100000
int blocked[MAX_H];
int dp[2][1<<4];

main() {
    int h, n;  scanf("%d %d", &h, &n);
    int x, y;
    for ( int i=0; i<n; ++i ) {
          scanf("%d %d", &x, &y);  blocked[y] |= 1<<x;
    }

    int locate[5] = { 0x0, 0x3, 0x6, 0xc, 0xf };//置き方パターン
    int add[5] = { 0, 1, 1, 1, 2 };//各置き方パターンで置ける個数

    int* crt = dp[0];
    int* next = dp[1];

    for ( int obstacle=0; obstacle<1<<4; ++obstacle ) {
          if ( obstacle&blocked[0] ) crt[obstacle] = -1;
          else crt[obstacle] = 0;
    }
```

問題７　積み荷の配置

AIZU
THE UNIVERSITY OF AIZU

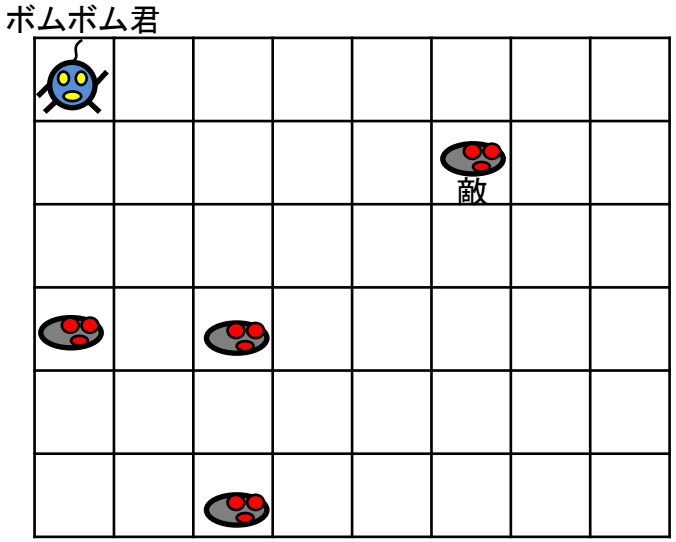
解答例（C++）

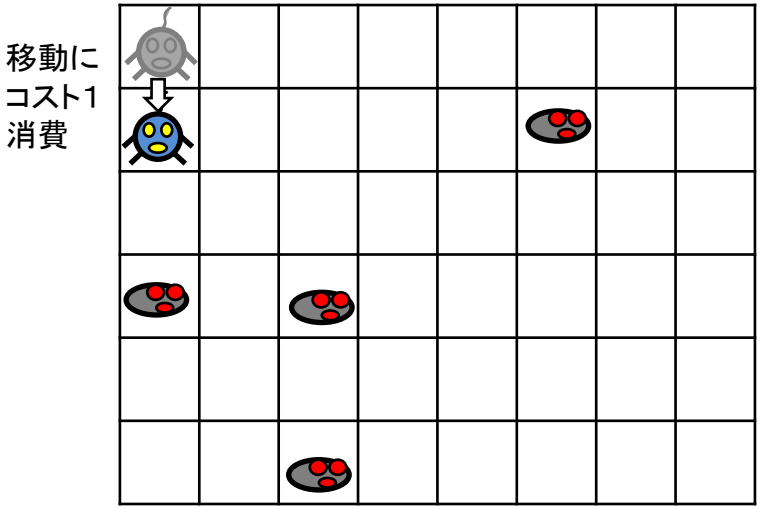
```
    for ( int i=0; i<h-1; ++i ) {
          std::fill( next, next+(1<<4), -1 );
          for ( int obstacle=0; obstacle<1<<4; ++obstacle ) {//既に有るものパターン(current)
              if ( crt[obstacle] == -1 ) continue;
              for ( int k=0; k<5; ++k ) {//置き方パターン(next)

                    int check0 = blocked[i] & locate[k];
                    int check1 = blocked[i+1] & locate[k];

                    if ( !check0 && !check1 && !(obstacle&locate[k])) {
                        int patt = blocked[i+1] | locate[k];
                        next[patt] = std::max( crt[obstacle] + add[k], next[patt] );
                    }
              }
          }
          int* tmp = crt;
          crt = next;
          next = tmp;
    }

    int res = 0;
    for ( int i=0; i<1<<4; ++i ) if ( next[i] > res ) res = next[i];
    printf("%d¥n", res );
}
```

問題８　ダンジョン

問題概要

- マス目状に区切られた長方形の盤面の、横方向のマスの数W(1≦W≦500)と縦方向のマスの数H(1≦H≦500)が与えられる.
- 盤面上に居る敵の位置がN(1≦N≦100,000)個与えられる.
- 初期位置(0, 0)からコスト１を消費して「ボムボム君」を上下左右１マスずつ動かすことができる.





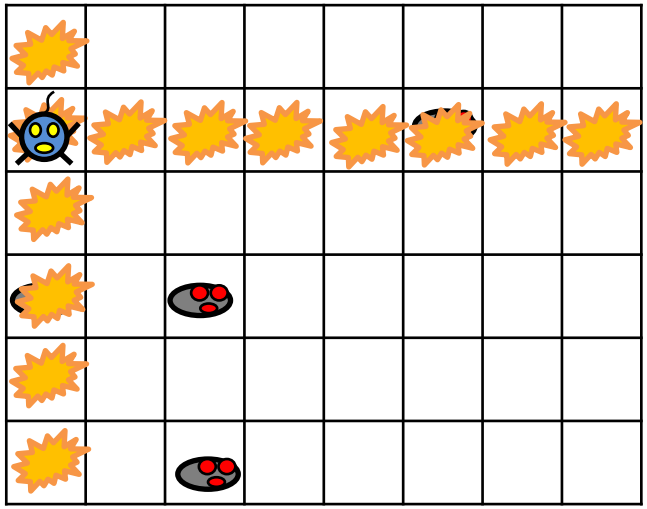
問題８　ダンジョン

AIZU
THE UNIVERSITY OF AIZU

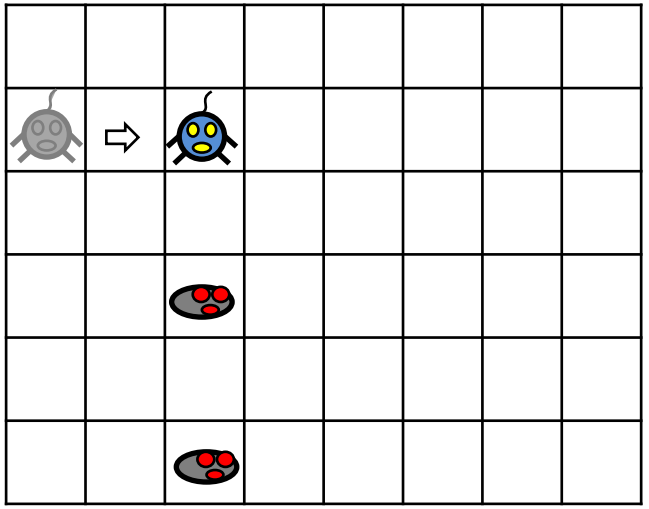
問題概要

- ボムボム君は爆弾を使用し、自分がいるマスと列番号が同じマスにいる敵と、行番号が同じマスにいる敵を一掃することができる.
- ボムボム君が全ての敵を倒すための最小のコスト(移動距離)を出力する.

同じ行と同じ列の敵を一掃！



移動にコスト2消費
ここで爆弾を使えば最小コスト3でゲームクリア



問題８　ダンジョン

AIZU
THE UNIVERSITY OF AIZU

総評

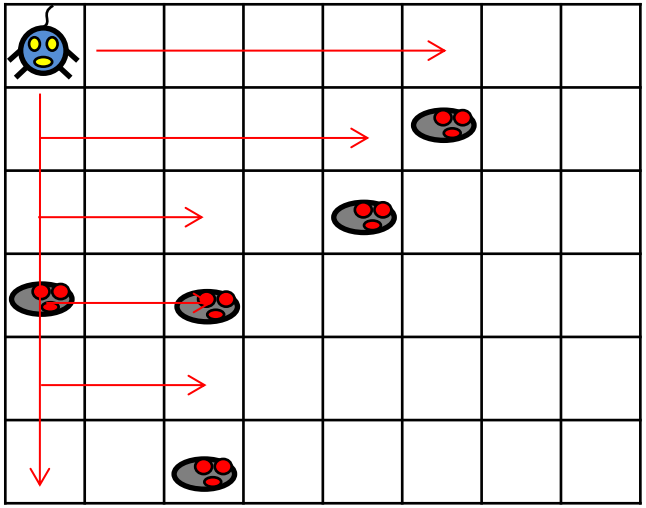
- 提出数 ９４、正答数 １９.
- 間違った判定方法を色々と思いつきやすい問題.
- 多かった間違い
  - 間違った貪欲法.

問題８　ダンジョン

AIZU
THE UNIVERSITY OF AIZU

解法

- 最小移動距離となる経路は、まっすぐ進んでから一度曲がった経路のうちどれか（何度も曲がっても結局一度曲がったものと距離は同じ）．



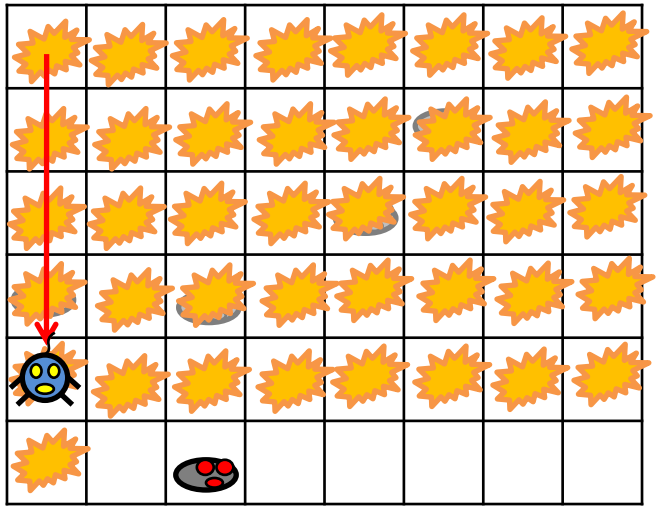
- 曲がってからいくつ進めば、その経路の場合の最小移動距離になるのか？

問題８　ダンジョン

AIZU
THE UNIVERSITY OF AIZU

解法

- 最初にまっすぐ進んだ経路上にあるマスと同じ行、列にいる敵はすでに倒したと考えることが出来る.

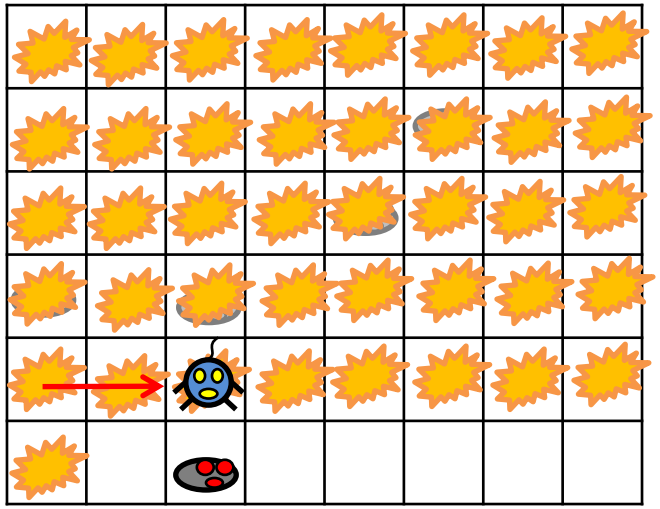


問題８　ダンジョン

AIZU
THE UNIVERSITY OF AIZU

解法

- 最初にまっすぐ進んだ経路上にあるマスと同じ行、列にいる敵はすでに倒したと考えることが出来る.
- 縦に進む場合なら、曲がってから横に進むべき距離は、それより先の縦座標に居る敵の、最大横座標になる.

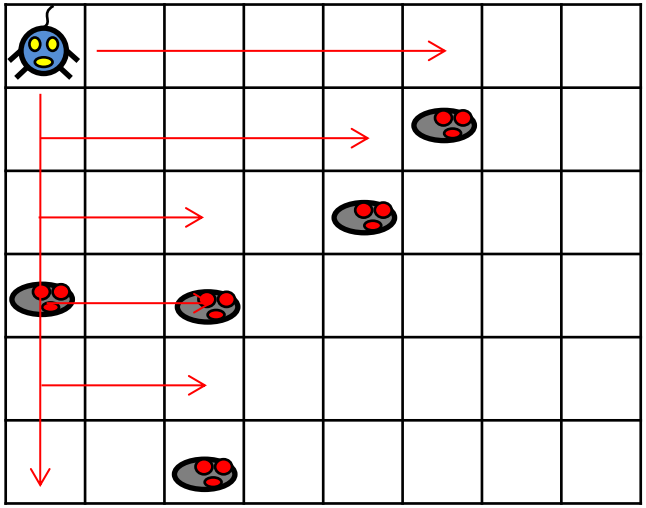


問題８　ダンジョン

AIZU
THE UNIVERSITY OF AIZU

解法

- WとHのどちらか小さい方にまっすぐ進む.
- 例えば、Hの方が小さかった場合、各y座標より下で、最大のx座標を持つ敵のx座標を記録する配列を作る（int maxX[H]）.
- 入力時に、各y座標での最大x座標を求めて置けば、maxXは、O(H)で求められる.
- 計算量 O(N + min(W, H)).
- 面倒であれば、小さい方を選ぶ処理を省いて、O(N+W)やO(N+H)でも良い.



maxX[0] = 5

maxX[1] = 4

maxX[2] = 2

maxX[3] = 2

maxX[4] = 2

maxX[5] = 0

問題８　ダンジョン

AIZU
THE UNIVERSITY OF AIZU

解答例（C++）

```
main() {

    int width, height, num_enemy;
    cin >> width >> height >> num_enemy;

    vector< int > rangeMaxY( width );//iからNまでの間に居る敵のy座標の最大値
    vector< int > maxY( width );//i列目の敵のy座標最大値

    int x, y;
    for ( int i=0; i<num_enemy; ++i ) {
          cin >> x >> y;
          if ( y > maxY[x] ) maxY[x] = y;
    }

    rangeMaxY[width-1] = 0;
    for ( int i=width-2; i>=0; --i ) {
          rangeMaxY[i] = max( maxY[i+1], rangeMaxY[i+1] );
    }

    int min_cost = width;
    for ( int i=0; i<width; ++i ) {
          min_cost = min( i + rangeMaxY[i], min_cost );
    }

    cout << min_cost << endl;
}
```

問題８　ダンジョン

AIZU
THE UNIVERSITY OF AIZU

解答例（Java）

```
class P08 {
    void solve(){
        Scanner sc = new Scanner( System.in );
        int width = sc.nextInt();
        int height = sc.nextInt();
        int num_enemy = sc.nextInt();
        int[] rangeMaxY = new int[width];
        int[] maxY = new int[width];

        for ( int i=0; i<num_enemy; ++i ) {
            int x = sc.nextInt();
            int y = sc.nextInt();
            if ( y > maxY[x] ) maxY[x] = y;
        }

        rangeMaxY[width-1] = 0;
        for ( int i=width-2; i>=0; --i ) {
            rangeMaxY[i] = Math.max( maxY[i+1], rangeMaxY[i+1] );
        }

        int min_cost = width;
        for ( int i=0; i<width; ++i ) {
            min_cost = Math.min( i + rangeMaxY[i], min_cost );
        }
        System.out.println( min_cost );
    }
    public static void main( String[] a ) {new P08().solve(); }
}
```

問題９ 文字列スワップ

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 文字列Sが与えられる.
- 隣り合う2文字をスワップする操作をK回まで行うことができる.
- 辞書順最小の文字列を求めてください.
- 2≦Sの長さ≦200000, 0≦K≦10^9

総評

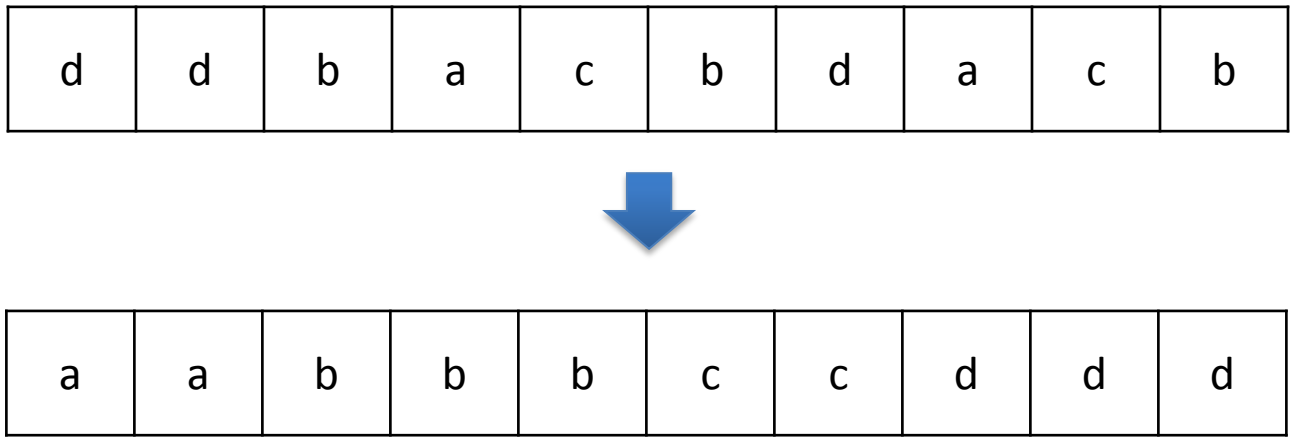
- 提出数 ５５、正答数 ９.
- シミュレーションでは明らかに時間切れになる.
- 辞書順最小にするようなスワップの仕方と、スワップ完了後の文字列の求め方が難しい問題.
- 多かった間違い
  - シミュレーションで時間切れ.

問題９ 文字列スワップ

AIZU
THE UNIVERSITY OF AIZU

解法

- 小さいものから（'a'から）左に貪欲に詰めて得られる文字列が辞書順最小になる （Kを考慮しない場合）.



問題９ 文字列スワップ

AIZU
THE UNIVERSITY OF AIZU

解法

- K を考慮する場合、単に小さいものから移動する方法はうまくいかない．例： K = 1

d	d	b	a	c	b	d	a	c	b

小さいものから移動した場合

d	d	a	b	c	b	d	a	c	b

最適な移動

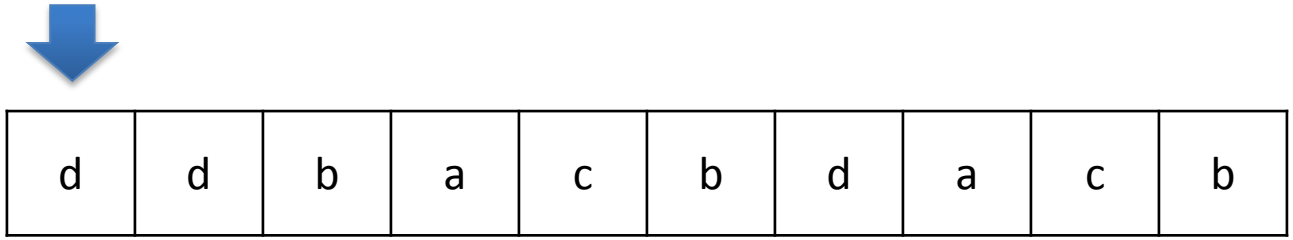
d	b	d	a	c	b	d	a	c	b

問題９ 文字列スワップ

AIZU
THE UNIVERSITY OF AIZU

解法

- 常に最も左側の文字を可能な限り小さくしたい.
- また、可能な限り少ないステップで文字を移動したい.

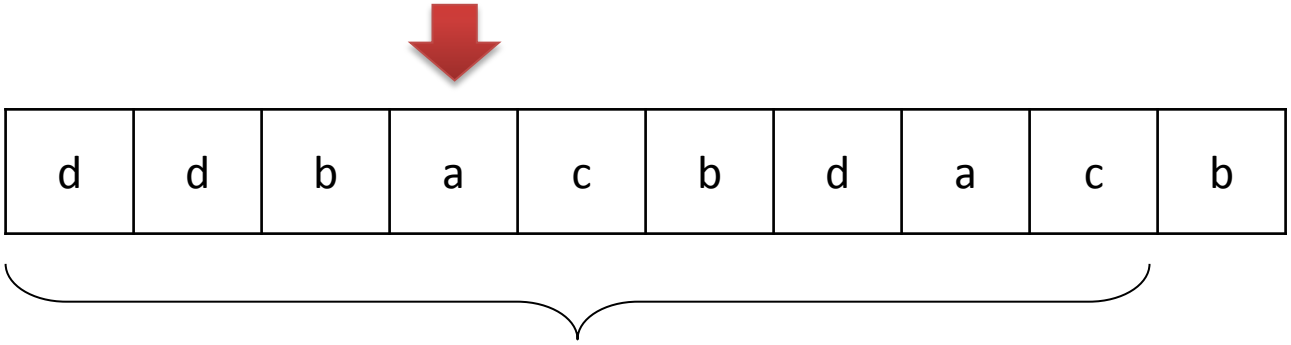


問題９ 文字列スワップ

AIZU
THE UNIVERSITY OF AIZU

解法

- 以下の処理をK>0の間限り繰り返す：

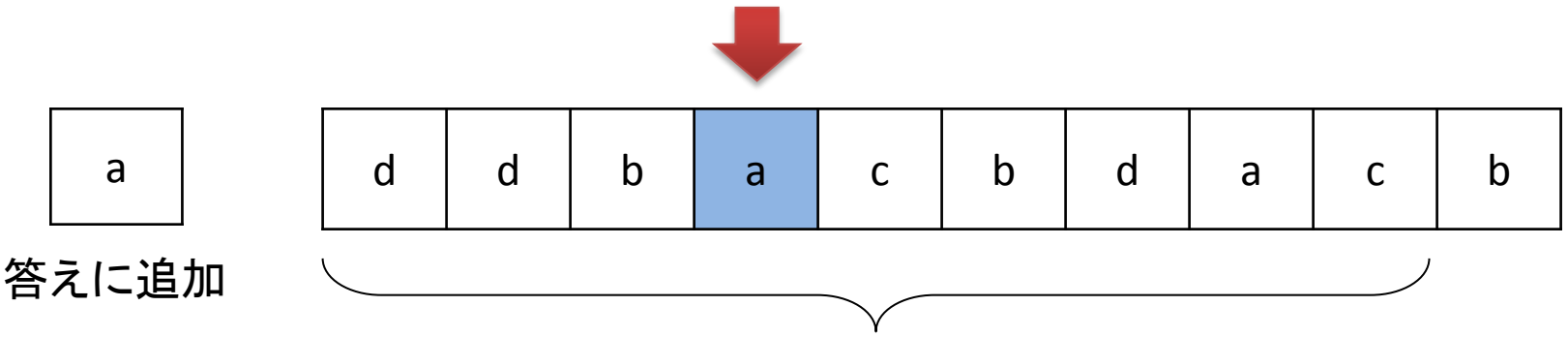


問題９ 文字列スワップ

AIZU
THE UNIVERSITY OF AIZU

解法

- 以下の処理をK>0の間限り繰り返す：



先頭からK文字で一番小さい文字
複数ある場合は最も左にあるものを選ぶ→idxとする

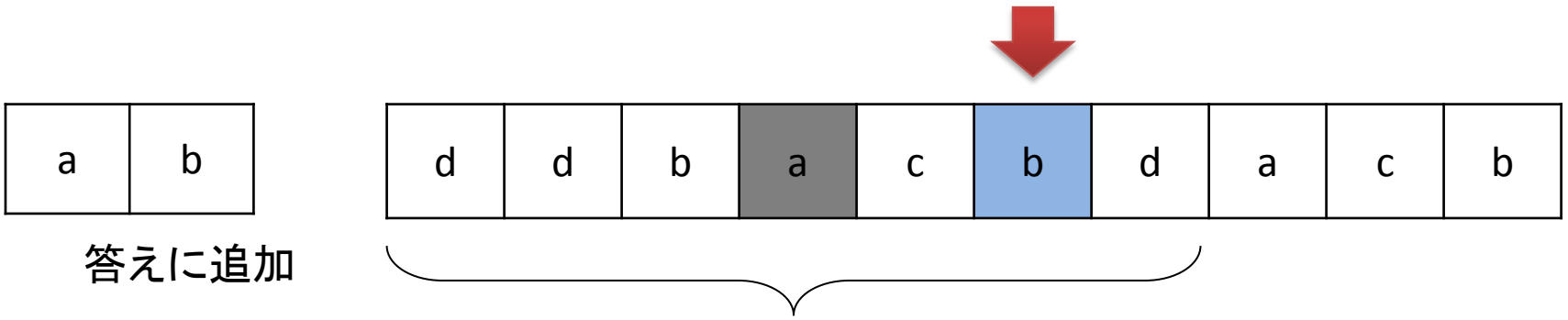
先頭へ移動するためのスワップ数をKから引く
つまり K -= 先頭から(idx-1)文字目までの文字数

問題９ 文字列スワップ

AIZU
THE UNIVERSITY OF AIZU

解法

- 以下の処理をK>0の間限り繰り返す：



先頭からK文字で一番小さい文字
複数ある場合は最も左にあるものを選ぶ→idxとする
※既に選ばれた文字を除く

先頭へ移動するためのスワップ数をKから引く
つまりK -= 先頭から(idx-1)文字目までの文字数
※既に選ばれた文字の分を除く

問題９ 文字列スワップ

AIZU
THE UNIVERSITY OF AIZU

解法

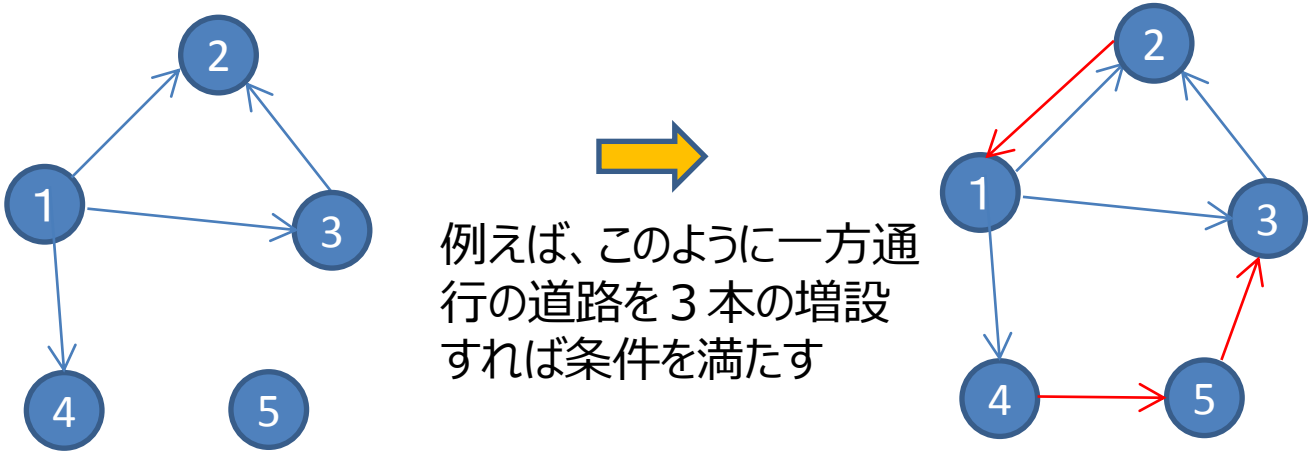
- 先頭からすでに選ばれた文字を除いたK文字の位置tは二分探索で求めることができる.
- 先頭からiまでの位置にありかつ選ばれていない文字の数はRMQやＢＩＴで求めることができる.
- 先頭からt文字目までで最適な文字はＲＭＱで求めることができる.
- ＢＩＴやＲＭＱで未選択文字の数を数えながら二分探索をする処理を|S|回繰り返す、$O(|S| \log|S|^2)$のアルゴリズムとなる.

問題１０　除雪

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 0 からN-1 までの番号が割り当てられたN 個の都市がある.
- ２つの都市を結ぶ一方通行のM本の道路で道路網が形成されている.
- 一方通行の道路をいくつか増設することで、どの都市からでもすべての道路を巡ることができるような道路網にしたい.
- 増設しなければいけない道路の数の最小値を求める.



問題１０　除雪

AIZU
THE UNIVERSITY OF AIZU

総評

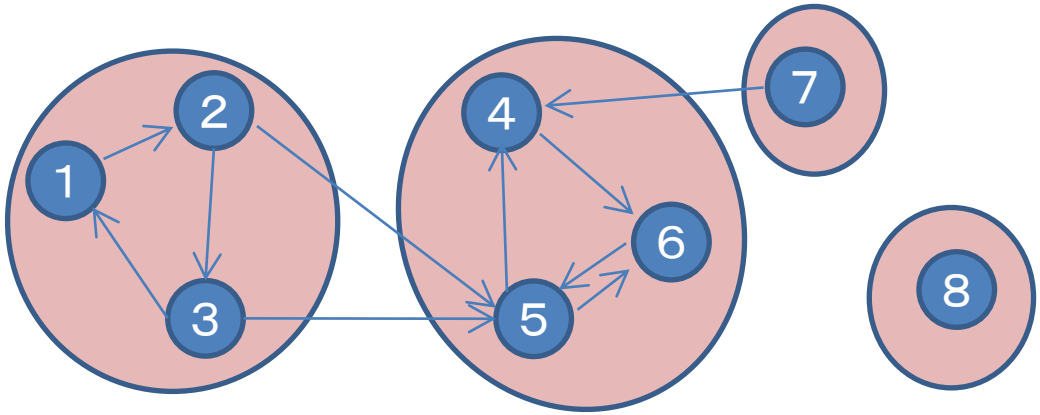
- 提出数　２３、正答数　５.
- グラフの知識を前提とした問題.
- すでに行き来できる都市同士の間では、新しい道路を増設する必要がない.
- これらの都市をグループにしてしまえばよいことには気づきやすいが、その後どうすればよいか、発想が難しい.

問題１０　除雪

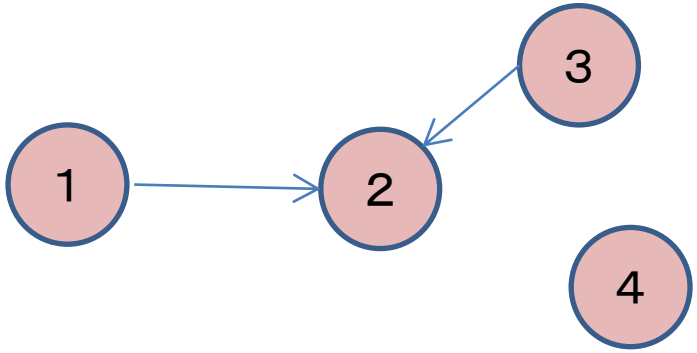
AIZU
THE UNIVERSITY OF AIZU

解法

- 強連結成分分解して、すでに行き来できる都市でグループを作る



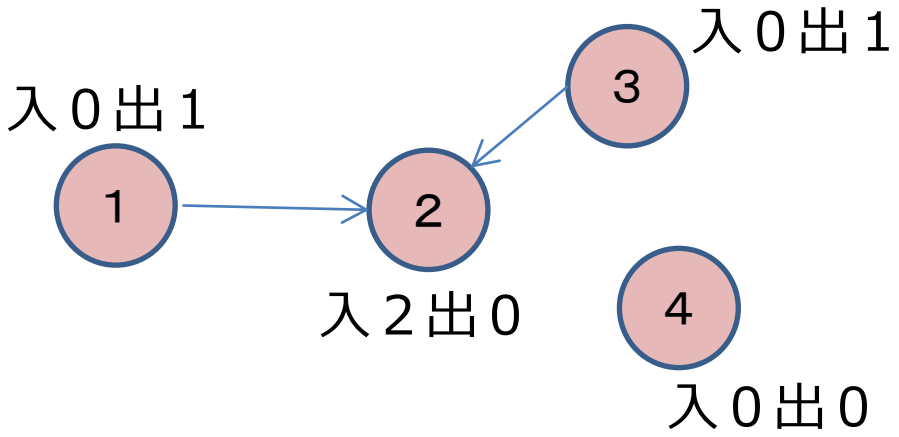
- グループを節点、あるグループの都市から別のグループの都市につながる辺だけを辺としたグラフを作る（このグラフにはループは存在しない）



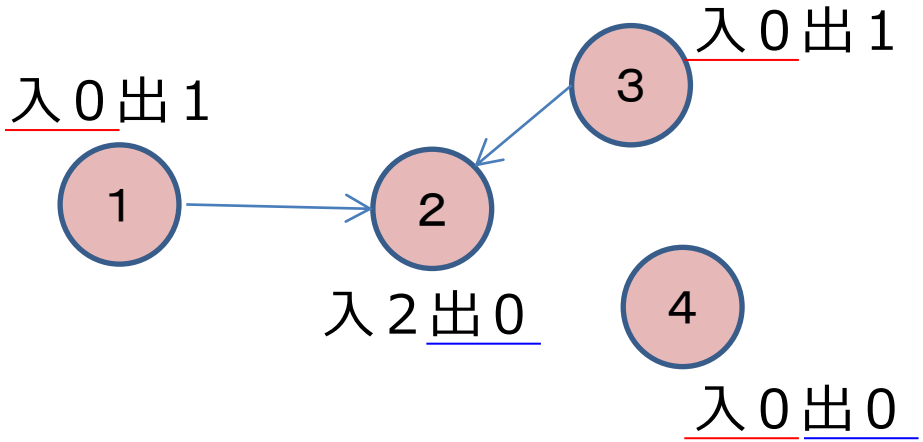
問題１０　除雪

解法

- 各節点の入次数と出次数を計算する



- 入次数０の節点の数と、出次数０の節点の数の大きい方が答え



入次数０の節点の数：３
出次数０の節点の数：２

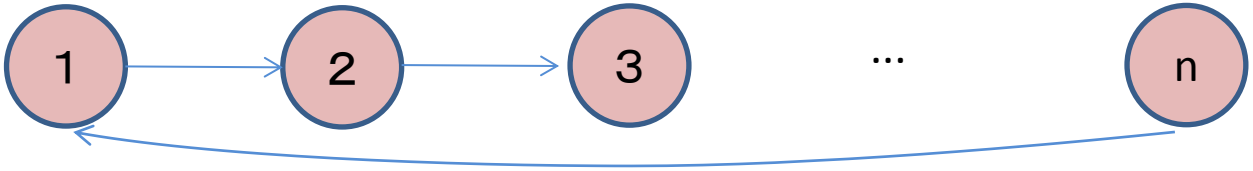
答えは３

問題１０　除雪

- なぜ入次数と出次数が０の節点の数を数えるのか？

節点の数がn個のとき

n本の辺でループを作れば、最小の辺の数で、どの頂点から始めても、
他のどの頂点へもたどり着けるようにできる



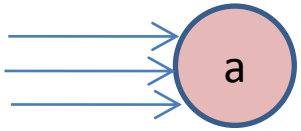
強連結成分分解して、グループのグラフを作れば、あとは
すでに存在する辺を利用して、ループを作るには最低何本の辺が必要か？
という問題と同じ.

問題１０　除雪

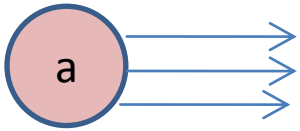
AIZU
THE UNIVERSITY OF AIZU

- なぜ入次数が０の節点と出次数が０の節点の数を数えるのか？

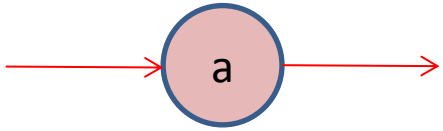
入次数が１より大きくても、出次数が０だとループには参加できない



出次数が１より大きくても、入次数が０だとループには参加できない



すべての節点がこのような形になっていれば、ループに参加しているとみなせる



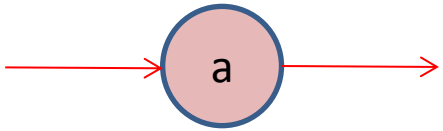
すべての節点の入次数も出次数も１以上になっていれば良い

問題１０　除雪

- なぜ入次数が０の節点と出次数が０の節点の数を数えるのか？

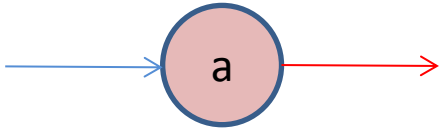
入次数0,出次数0の節点の場合

この節点をループに参加させるには、
入次数を１、出次数を１にする



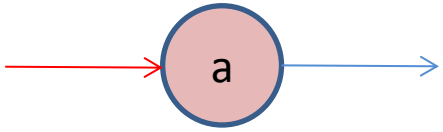
入次数1,出次数0の節点の場合

この節点をループに参加させるには、
出次数を１にする

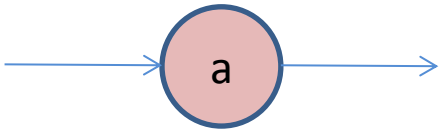


入次数0,出次数1の節点の場合

この節点をループに参加させるには、
入次数を１にする



入次数1,出次数1の節点の場合



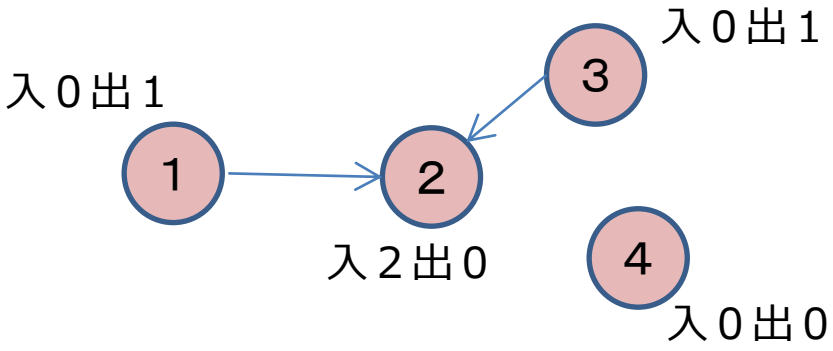
この節点は、すでにループの一部
になっているとみなせる

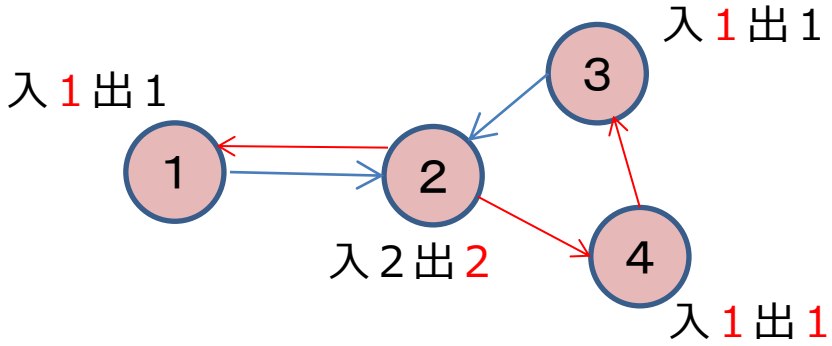
問題１０　除雪

- なぜ入次数が０の節点と出次数が０の節点の数を数えるのか？

１．入次数が０の節点の数　≧　出次数０の節点の数　の場合

出次数が０の節点から入次数が０への辺を張ってループを作れば、入り次数が０の節点をすべてカバーすると、必ず出次数０の節点もループに参加させることが出来る





２．入次数が０の節点の数　＜　出次数０の節点の数　の場合

同様のことがいえる

- 入次数が０の節点と出次数が０の節点の数を数えて、大きい方の数だけ辺を張れば、全体をカバーするループが作れる

問題１０　除雪

AIZU
THE UNIVERSITY OF AIZU

解答例（C++）

```
typedef std::vector<int> vInt;

int nNode, nEdge;
std::vector< vInt > edgeTable;
std::vector< vInt > revEdgeTable;
vInt order;
vInt group;
std::vector<bool> used;

main() {
    scanf("%d %d", &nNode, &nEdge);

    edgeTable.resize( nNode );
    revEdgeTable.resize( nNode );
    used.resize( nNode );
    group.resize( nNode );

    int s, t;
    for ( int i=0; i<nEdge; ++i ) {
          scanf("%d %d", &s, &t);
          edgeTable[s].push_back( t );
          revEdgeTable[t].push_back( s );
    }

    //StrngConctComp()は強連結成分分解を行い groupに各節点のグループ番号を記録して グループの数を返す
    FindNumNewEdges( StrngConctComp() );
}
```

問題１０　除雪

AIZU
THE UNIVERSITY OF AIZU

解答例（C++つづき）

```
void FindNumNewEdges( int nGroup ) {
    if ( nGroup == 1 ) {
          printf("0¥n");
          return;
    }

    vInt inCnt( nGroup );
    vInt outCnt( nGroup );

    for ( int s=0; s<nNode; ++s ) {//始点s
          int grp_s = group[s];//sの属するグループ番号

          for ( int t: edgeTable[s] ) {//終点t
              int grp_t = group[t];//tの属するグループ番号

              if ( grp_s != grp_t ) {
                    ++inCnt[grp_t];
                    ++outCnt[grp_s];
              }
          }
    }

    size_t inZero = std::count( ALL(inCnt), 0 );
    size_t outZero = std::count( ALL(outCnt), 0 );
    printf("%ld¥n", std::max( inZero, outZero ));
}
```

問題１０　除雪

AIZU
THE UNIVERSITY OF AIZU

解答例（Java）

```
int nNode, nEdge;
ArrayList< ArrayList<Integer> > edgeTable = new ArrayList< ArrayList<Integer> >();
ArrayList< ArrayList<Integer> > revEdgeTable= new ArrayList< ArrayList<Integer> >();
ArrayList<Integer> order = new ArrayList<Integer>();
int[] group = null;
boolean[] used = null;

void solve(){
      Scanner sc = new Scanner( System.in );
      nNode = sc.nextInt();
      nEdge = sc.nextInt();
      for ( int i=0; i<nNode; ++i ) {
          edgeTable.add( new ArrayList<Integer>() );
          revEdgeTable.add( new ArrayList<Integer>() );
      }

      group = new int[nNode];
      used = new boolean[nNode];
      for ( int i=0; i<nEdge; ++i ) {
          int s = sc.nextInt();
          int t = sc.nextInt();
          edgeTable.get( s ).add( t );
          revEdgeTable.get( t ).add( s );
      }
      //StrngConctComp()は強連結成分分解を行い groupに各節点のグループ番号を記録して グループの数を返す
      FindNumNewEdges( StrngConctComp() );
}
```

問題１０　除雪

AIZU
THE UNIVERSITY OF AIZU

解答例（Javaつづき）

```
void FindNumNewEdges( int nGroup ) {
        if ( nGroup == 1 ) {
            System.out.println("0");
            return;
        }
        int[] inCnt = new int[nGroup];
        int[] outCnt = new int[nGroup];
        for ( int i=0; i<nGroup; ++i ) {
            inCnt[i] = 0;
            outCnt[i] = 0;
        }
        for ( int s=0; s<nNode; ++s ) {//始点s
            int grp_s = group[s];//sの属するグループ番号
            for ( int t: edgeTable.get(s) ) {//終点t
                  int grp_t = group[t];//tの属するグループ番号
                  if ( grp_s != grp_t ) {
                      ++inCnt[grp_t];
                      ++outCnt[grp_s];
                  }
            }
        }
        int inZero = 0;
        int outZero = 0;
        for ( int i=0; i<nGroup; ++i ) {
            if ( inCnt[i] == 0 ) ++inZero;
            if ( outCnt[i] == 0 ) ++outZero;
        }
        System.out.println( Math.max( inZero, outZero ) ); }
```

問題１１ ネットワークの課金システム

問題概要

- 節点数がNの木が与えられる.
- 各辺は太さを持ち、それが直接繋ぐ節点間の通信料はその太さの値になる。ただし、太さがKで割り切れる場合、その通信料は０になる.
- 以下の種類を含むQ個のクエリに答える.
  - 節点xから出ている全ての辺の太さをa増加させる.
  - 2点間(s, t)の通信料の総和を報告する.
- 2≦N≦200000, 1≦K≦100000, 1≦Q≦10000

総評

- 提出数 ８、正答数 ２.
- NやQが大きいので、かなり工夫が要る.
- ネットワークにループが無い(木である)ことを利用した、高度なデータ構造が必要.

問題１１ ネットワークの課金システム

解法

- 木に対して Heavy-Light Decomposition （HL分解）を行う.
- HL分解を行った後の木をセグメントツリーで管理する.
  - HL分解後のグループ（＝ノード）内の要素からできるリストが”有効な”区間となる
- 各sendクエリは、

  ＋始点から根までの距離

  +終点から根までの距離

  －LCA (Least Common Ancestor)から根までの距離

  から得られる.

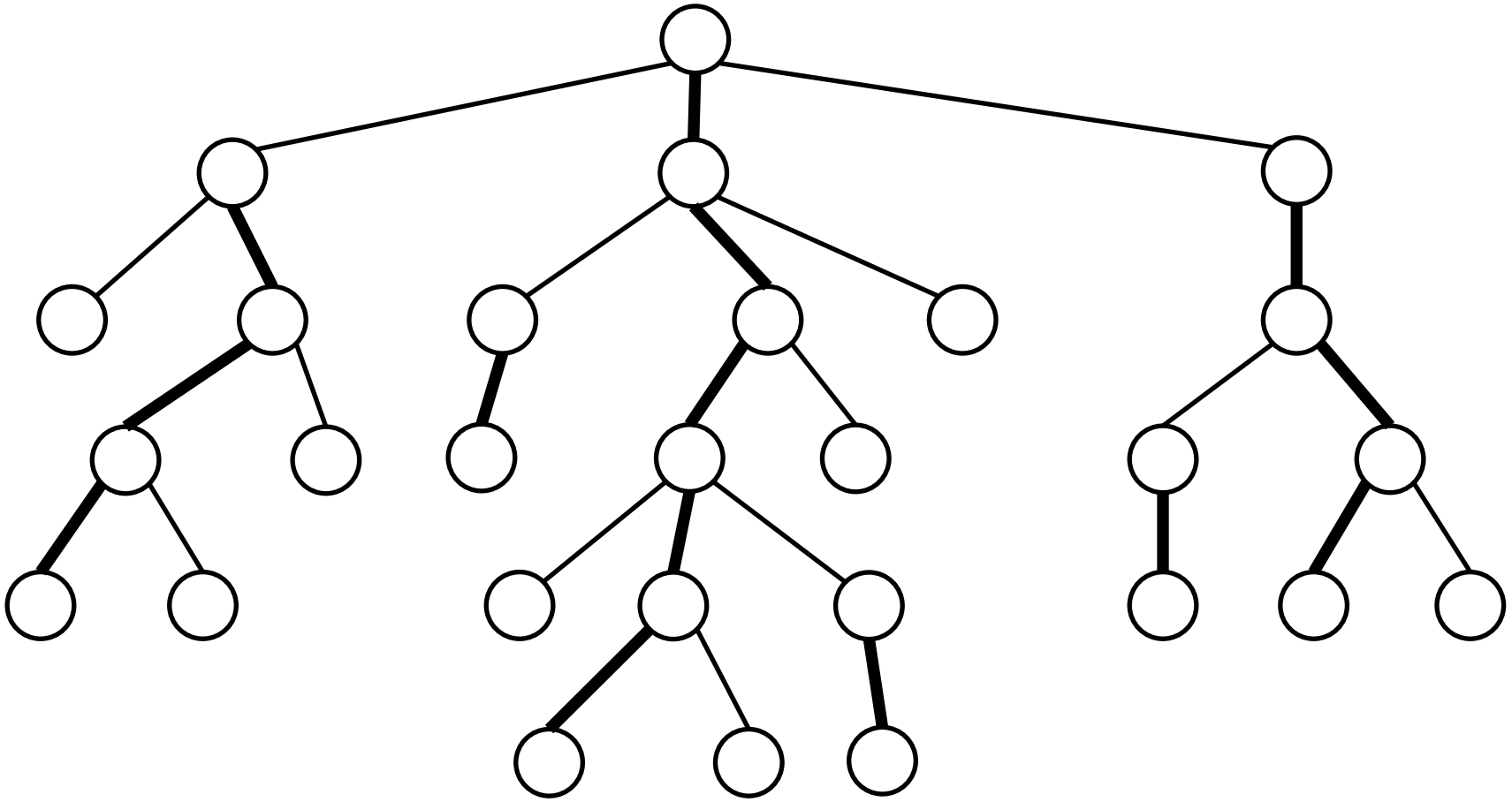
問題１１ ネットワークの課金システム

解法

- HL分解を行った後の木の高さは O(log N)になる.
- よってLCAはO(log N)で求まる.
- addクエリはセグメント木を更新してO(log N).
- sendクエリはO(log N) 個の区間に対してそれぞれクエリ処理を行い O(log N^2).
- 全体の計算量は O(Q log N^2)

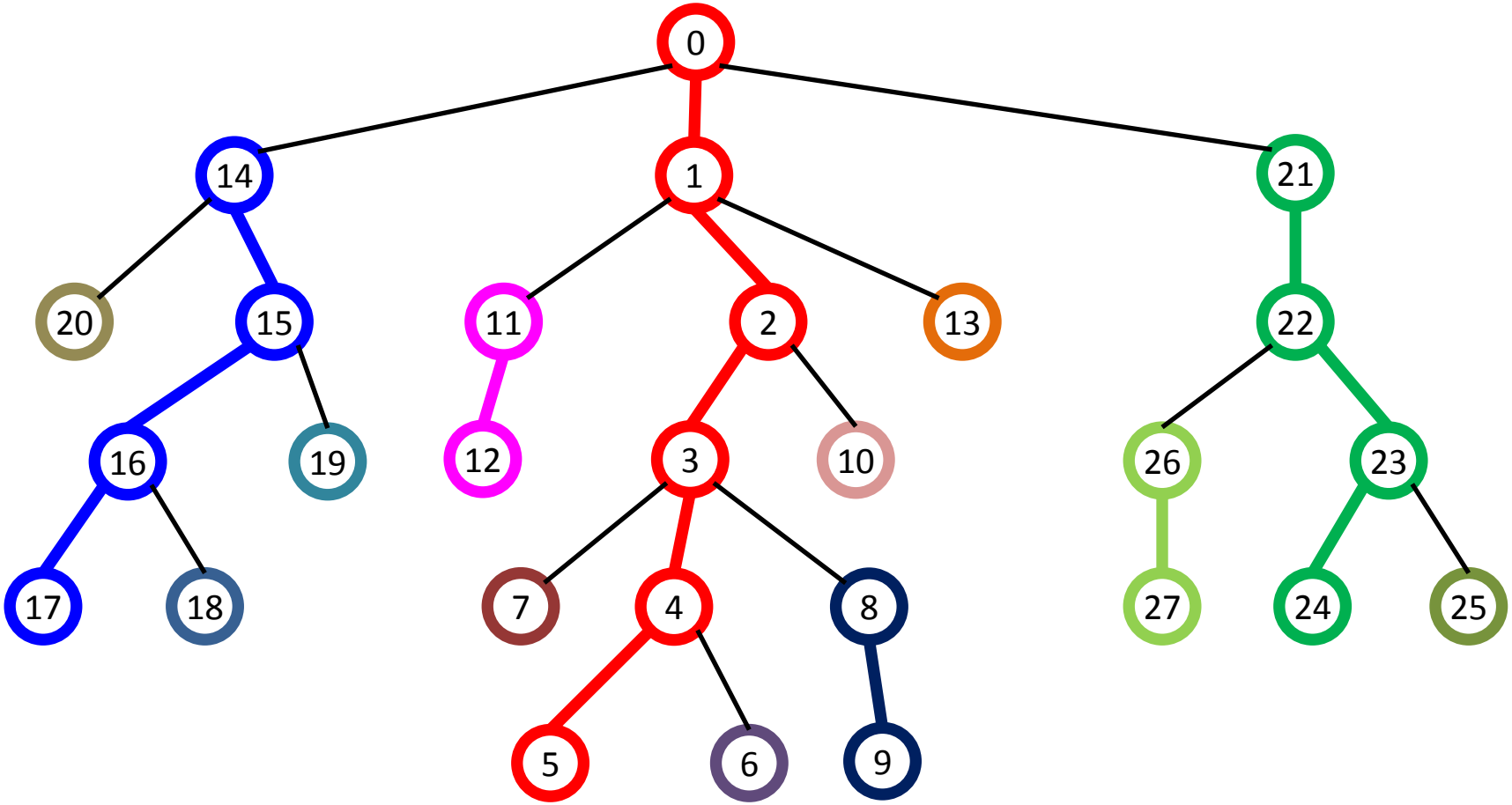
問題１１ ネットワークの課金システム

- 木に対して Heavy-Light Decomposition （HL分解）を行う.



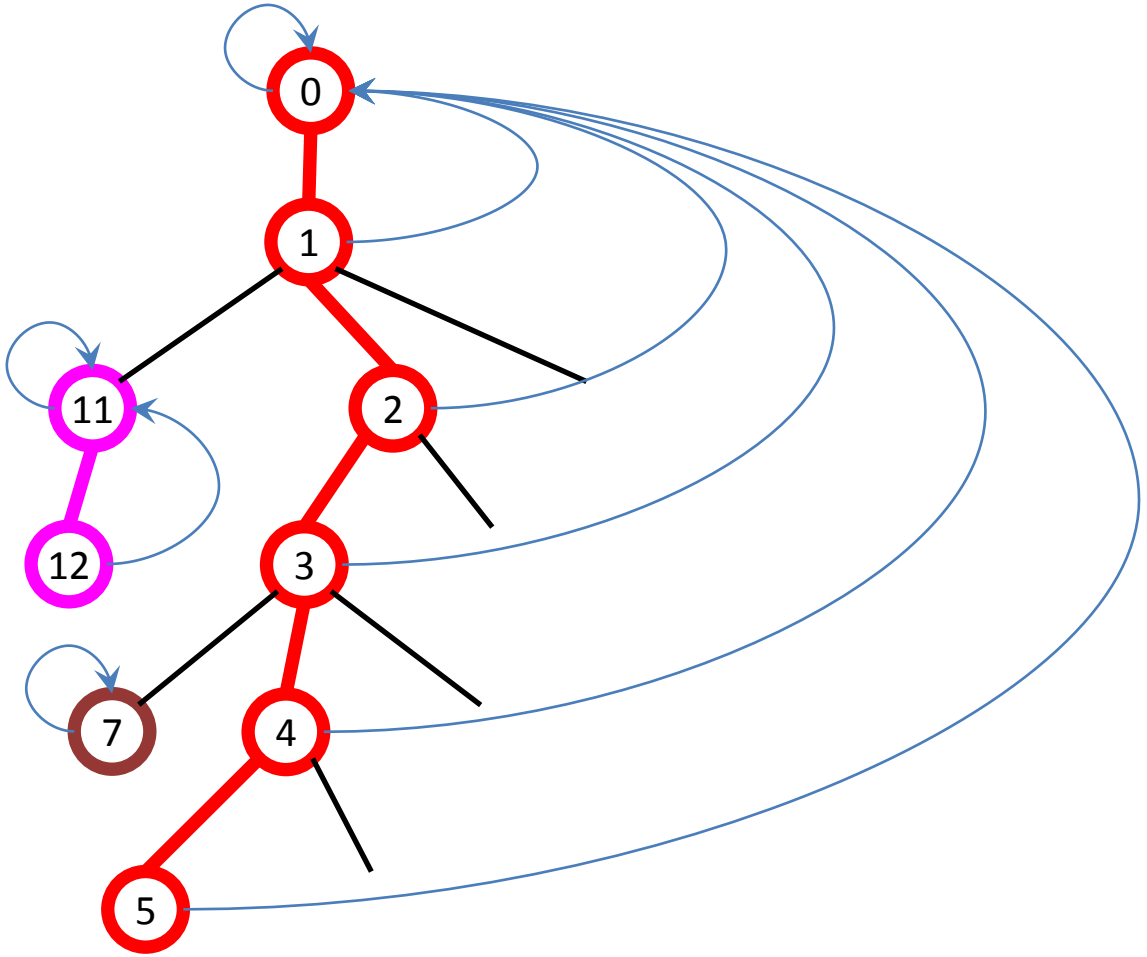
問題１１ ネットワークの課金システム

- Heavy辺を辿ってリストに分解し、セグメントツリーの対応するインデックスを振る．同じ色＝１つの節点



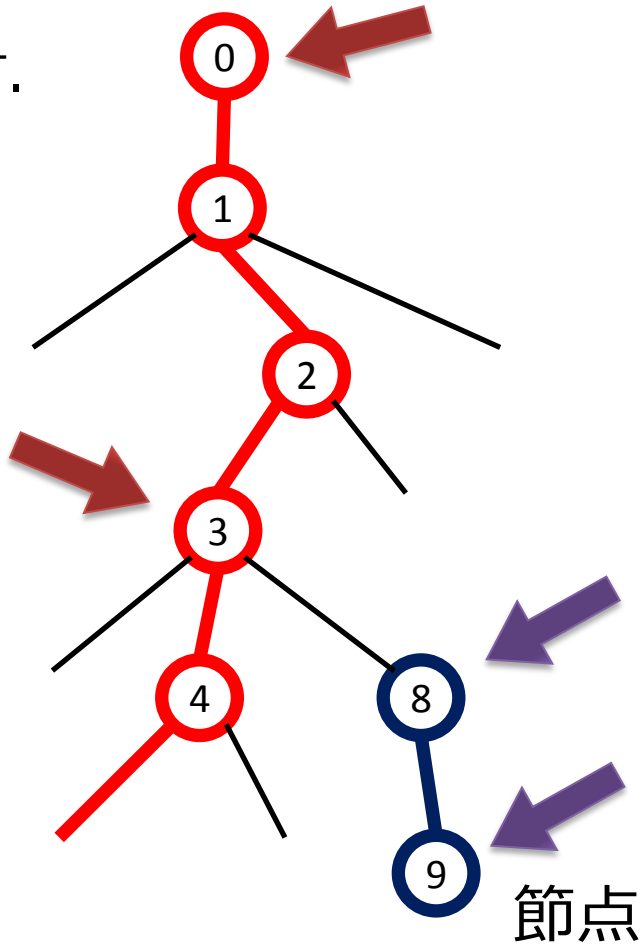
問題１１ ネットワークの課金システム

- 各節点について、リスト（区間）の先頭となる節点（ヘッダ）へのポインタを保持しておく.



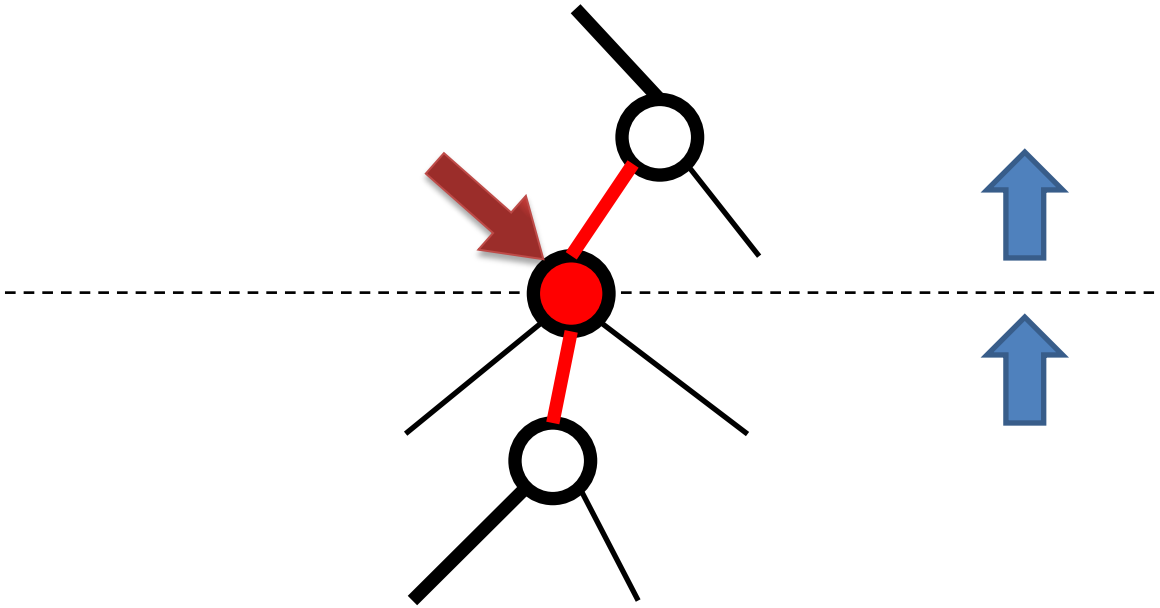
問題１１ ネットワークの課金システム

- ある節点から根までの辿り方：
  - 節点からそのヘッダまでの範囲を対象とする.
  - ヘッダが根であれば終了.
  - ヘッダの親を次の節点に設定し繰り返す.



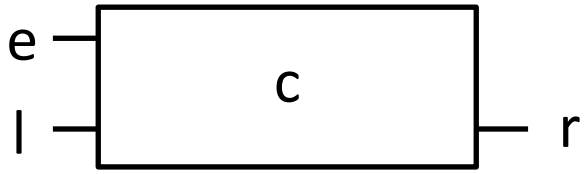
問題１１ ネットワークの課金システム

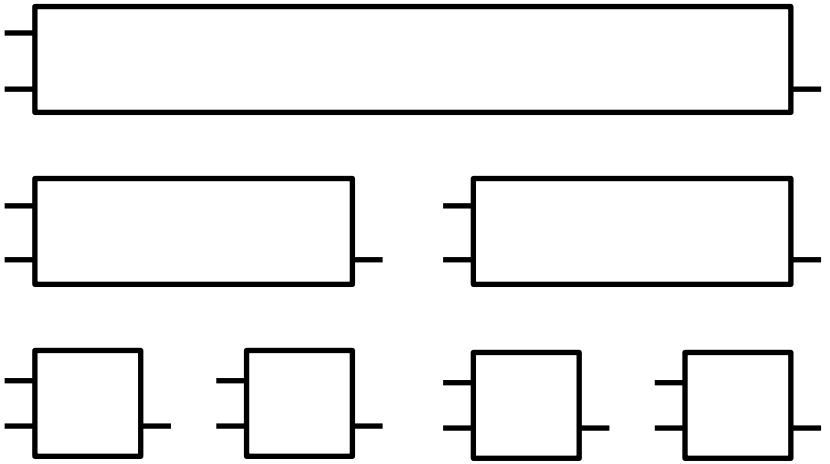
- 加算クエリの遅延処理
  - 節点を通る道は下から上への一方通行.
  - 節点に値を累積し、その節点を通るときに上下の辺に加算すればよい.
  - 接する全ての辺にそのつど加算する必要はない.
  - 区間で処理するためセグメントツリーは更新する必要がある.



問題１１ ネットワークの課金システム

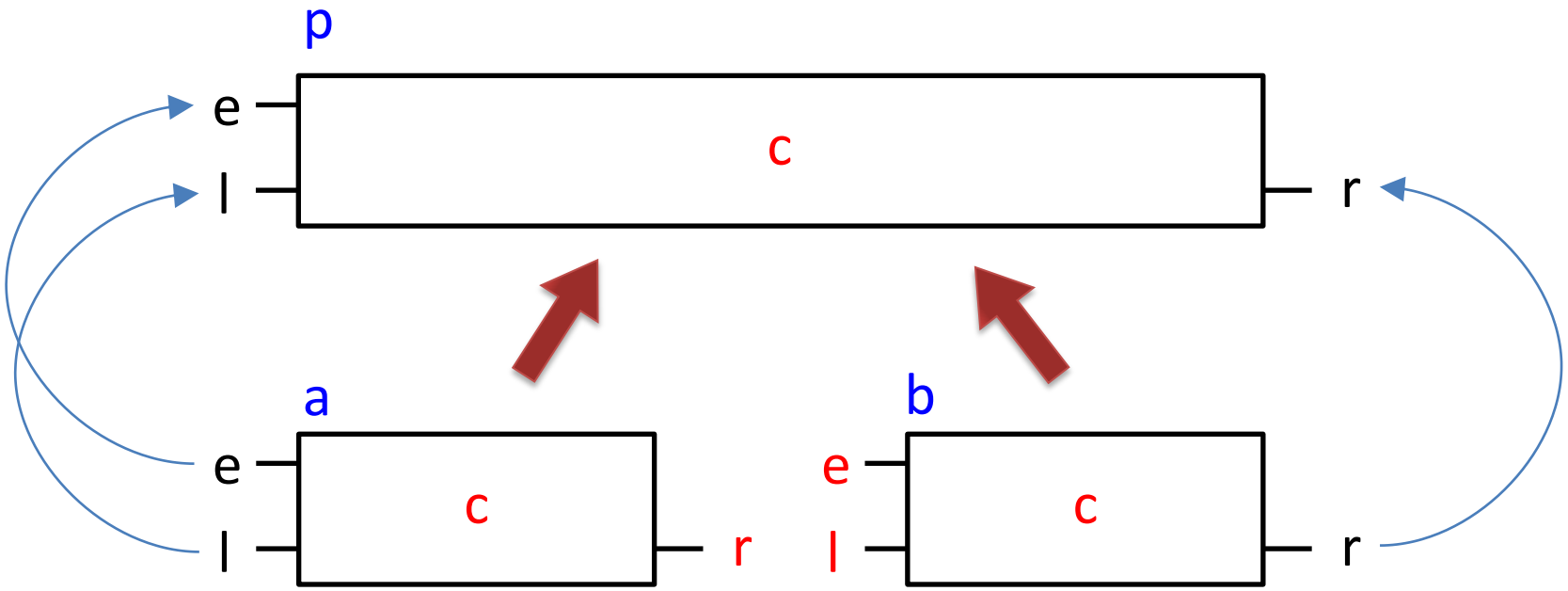
- セグメント木のノードの要素
  - l: 節点・区間の根に向かう方向の辺（区間の左側）に累積された太さ.
  - r: 節点・区間の葉に向かう方向の辺（区間の右側）に累積された太さ.
  - e: 節点・区間の根（親）に向かう方法の辺の、オリジナルの太さ
  - c: 区間におけるコスト.





問題１１ ネットワークの課金システム

- セグメント木のノードのマージ(addクエリ、sendクエリ共通)



p.l ← a.l
p.r ← b.r
p.e ← a.e

p.c ← a.c + b.c
cost ← a.r + b.l + b.e
p.c ← p.c + (cost%K>0?cost:0)