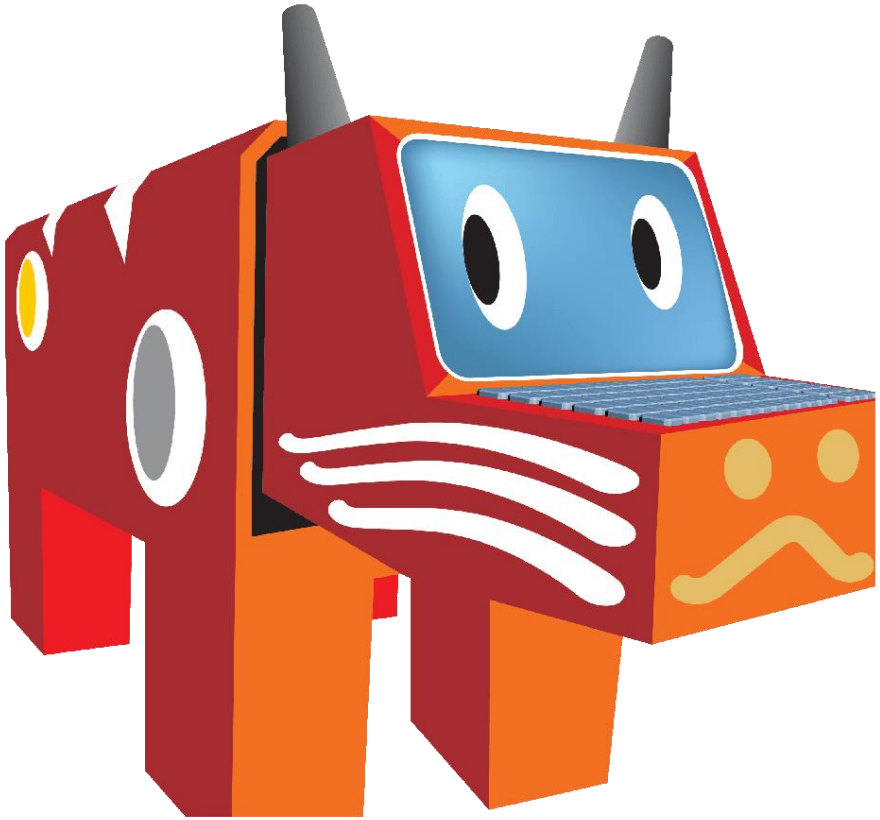
パソコン甲子園２０１６本選
プログラミング部門　解説



会津大学
2016年11月12日

概要

AIZU
THE UNIVERSITY OF AIZU

#	タイトル	分野	実装	思考	得点	正答数
1	長方形	基礎	☆	☆	3	24
2	棒で作る直方体	基礎	★	☆	4	24
3	有理式最大化	全探索・貪欲	★	★	5	19
4	必勝７ならべ	探索・シミュレーション・貪欲	★☆	★☆	8	13
5	環状すごろく	グラフ	★☆	★☆	8	14
6	実数既約分数化	文字列処理・数学	★☆	★☆	8	14
7	静かな町	グラフ・最短経路	★★☆	★★☆	10	10
8	勢力の予想	動的計画法	★★	★★★	10	9
9	ソート	データ構造	★★☆	★★★★	11	1
10	蟻	シミュレーション・データ構造	★★★	★★★☆	11	3
11	文字列ゲーム	データ構造・文字列処理	★★★☆	★★★★☆	11	3
12	夕暮れ	計算幾何学	★★★★	★★★	11	1

総評

AIZU
THE UNIVERSITY OF AIZU

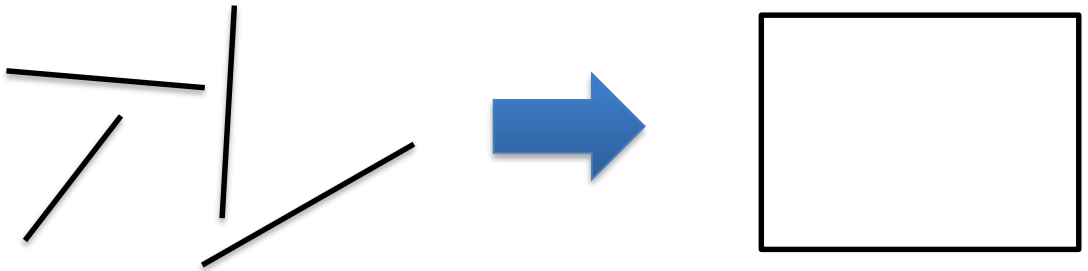
- すべてのチームが２問以上解いた.
- 各問題が１チーム以上に解かれた.
- 早い時間帯に全完するチームはなかった.

問１　長方形

AIZU
THE UNIVERSITY OF AIZU

問題概要

- ４つの整数が与えられる.
- それらの整数をそれぞれの辺の長さとする長方形が作れるか判定せよ.



総評

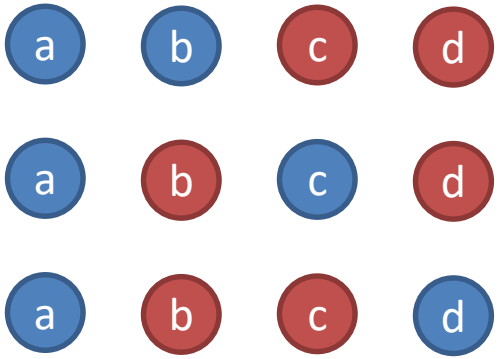
- 正答数 24.
- 変数宣言、標準入出力処理、分岐処理を行えるかを問う最も基本的な問題である.

問１　長方形

AIZU
THE UNIVERSITY OF AIZU

解法１

- ４つの整数をa, b, c, d とする.
- a = b かつ c = d または
  a = c かつ b = d または
  a = d かつ b = c　ならば「yes」.



解法２

- ４つの整数を配列A[4]に格納し整列する.
- A[0] = A[1] かつ A[2] = A[3] ならば「yes」.

問１　長方形

AIZU
THE UNIVERSITY OF AIZU

解答例１（C++）

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

main(){
  int a, b, c, d;
  cin >> a >> b >> c >> d;
  if ( a == b && c == d ||
       a == c && b == d ||
       a == d && b == c ) cout << "yes" << endl;
  else cout << "no" << endl;

  return 0;
}
```

問１　長方形

AIZU
THE UNIVERSITY OF AIZU

解答例２（C++）

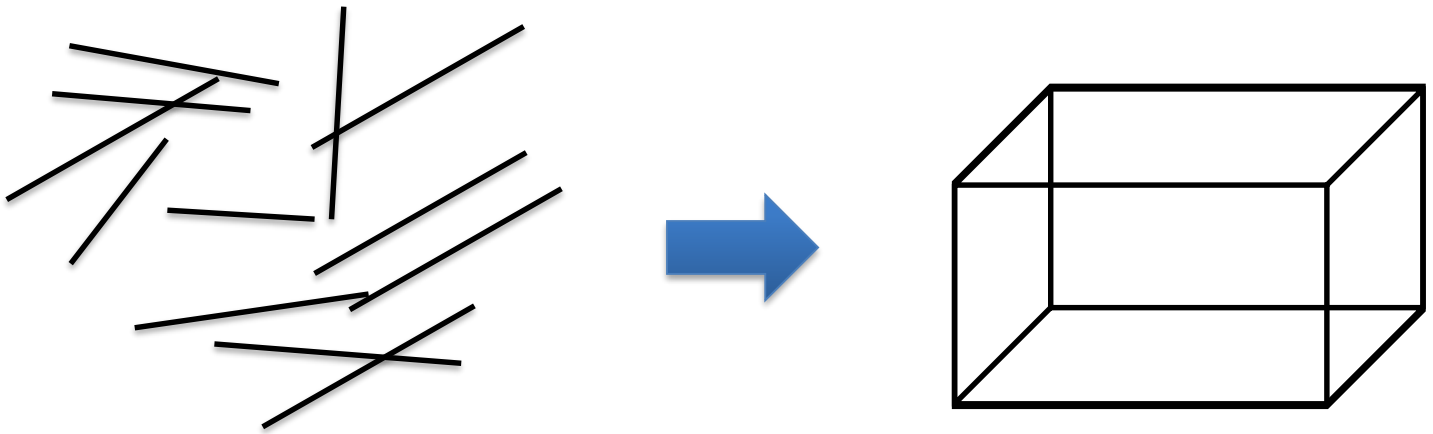
```
#include<bits/stdc++.h>
using namespace std;


int main(){
  int A[4];
  cin >> A[0] >> A[1] >> A[2] >> A[3];
  sort( A, A+4 );
  if( A[0] == A[1] && A[2] == A[3] ) cout << "yes" << endl;
  else cout << "no" << endl;
}
```

問２　棒で作る直方体

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 12本の棒のそれぞれの長さが整数で与えられる.
- これらの棒を使って、直方体（フレーム）が作れるか判定せよ.



総評

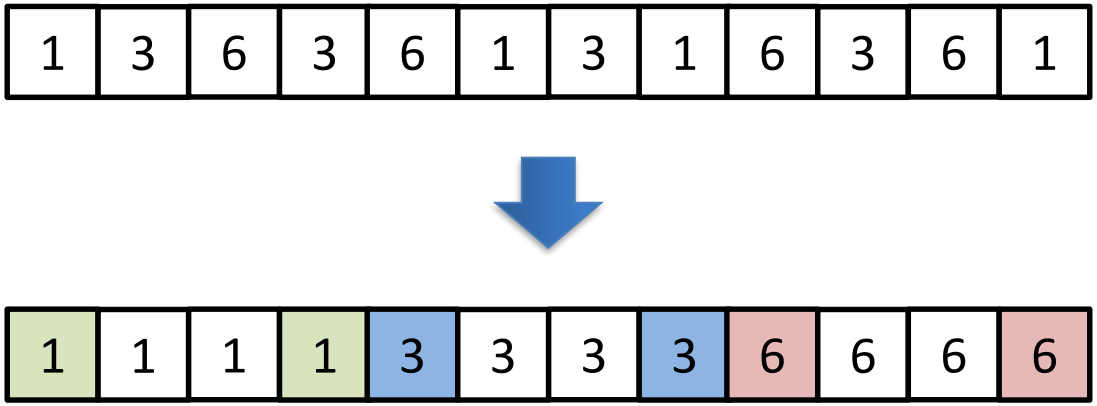
- 正答数 24.
- 基礎的なアルゴリズムを応用して、問題を解決できるかが問われている.

問２　棒で作る直方体

AIZU
THE UNIVERSITY OF AIZU

解法1

- 同じ長さの棒が４本含まれる、３つのグループを作れるか判定する.
- 入力を整列し、最初の４つが同じ整数、次の４つが同じ整数、次の４つが同じ整数かどうかを調べる.
- 各グループの先頭と末尾が一致するか調べるだけでよい.



問２　棒で作る直方体

AIZU
THE UNIVERSITY OF AIZU

解答例１（Ｃ＋＋）

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

int main(){
  int a[12];
  for(int i=0;i<12;i++) cin>>a[i];
  sort(a,a+12);
  if(a[0]==a[3]&&a[4]==a[7]&&a[8]==a[11]) cout<<"yes"<<endl;
  else cout<<"no"<<endl;
  return 0;
}
```

問２　棒で作る直方体

AIZU
THE UNIVERSITY OF AIZU

解法2

- 同じ長さの棒が４本含まれる、３つのグループを作れるか判定する.
- 与えられた整数について、1, 2, 3, .. 100 の個数を数える.
  - 配列A[i]に整数iの個数を記録.
- ４で割り切れるかどうかを判定する.
  - 割り切れないものがある　→「no」.

問２　棒で作る直方体

AIZU
THE UNIVERSITY OF AIZU

解答例2（C＋＋）

```
#include<iostream>
using namespace std;
const int MAX=100;
const int EDGE=12;
int ary[MAX];

int main(){
  int in,x=0,i=EDGE;

  while(i--){
    cin >> in;
    ary[in-1]++;
  }
  for(i=0;i<MAX;i++)
    if(ary[i]&3) break;

  cout << (i!=MAX?"no":"yes") << endl;
  return 0;
}
```

問３　有理式最大化

AIZU
THE UNIVERSITY OF AIZU

問題概要

- N個の異なる自然数 $a_1, a_2, \ldots a_N$ から異なる４つを選んで、それらをA, B, C, D としたとき、次の式の最大値を求よ.

$$\frac{A+B}{C-D}$$

- $4 \leqq N \leqq 1000, 1 \leqq a_i \leqq 100000000$

講評

- 正答数19.
- 計算量を見積もり、問題の制約を考慮した基礎的なアルゴリズムを考え、実装できるかが問われている.

問３　有理式最大化

AIZU
THE UNIVERSITY OF AIZU

誤答

- 簡単な貪欲法では解くことができない.
- 例えば、A + B を最大化するように選び、残ったもので C – D を最小化するように選ぶ方法では正しく答えを求めることができない.

```
sort(a, a+n);
cout << (a[n-2] + a[n-1])/(a[1] - a[0] ) << endl;
```

入力例：　1　50　51　100

$$\frac{51+100}{50-1}$$

$$\frac{1+100}{51-50}$$

問３　有理式最大化

AIZU
THE UNIVERSITY OF AIZU

解法１

- 入力を整列する（昇順、降順どちらでもよい）.
- C と D のすべての組み合わせについて（C と Dを決め打ちして選び）、選ばれていない要素の中から、大きい順に２つの要素を選び、それらを A、B とする。
- C と D のすべての組み合わせを試してO(N^2).
- 入力を予め整列しておけば、A と B はO(1)で探すことができる.
- 全体の計算量はO(N^2).

問３　有理式最大化

AIZU
THE UNIVERSITY OF AIZU

解答例１（Ｃ＋＋）

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

int main(){
  int n,i,j,k;
  cin>>n;
  double a[n],ans=0,x,y;
  for(i=0;i<n;i++) cin>>a[i];
  sort(a,a+n);
  for(i=0;i<n;i++){
    for(j=i+1;j<n;j++){
      k=n-1;
      while(k==i||k==j) k--;
      x=a[k];k--;
      while(k==i||k==j) k--;
      y=a[k];
      ans=max(ans,(x+y)/(a[j]-a[i]));
    }
  }
  printf("%6lf\n",ans);
  return 0;
}
```

問３　有理式最大化

AIZU
THE UNIVERSITY OF AIZU

解法2

- 入力を整列する（昇順）.
- DをO(N)で決め打ちするとC が決まる.
  - CはDよりも大きい（そうでなければ、式の結果が負になる）.
  - CがA,Bの候補にもなる場合は、Cが小さくなるように選ぶ.
  - CとDは隣あっている要素となる.
- 入力を予め整列しておけば、A と B はO(1)で探すことができる.
- 全体の計算量は整列のアルゴリズムに依存してO(N log N).

問３　有理式最大化

AIZU
THE UNIVERSITY OF AIZU

解答例2（C＋＋）

```
main(){
  int n, S[1000];
  scanf("%d", &n);
  for ( int i = 0; i < n; i++ ) scanf("%d", &S[i]);

  double res = 0.0;
  sort(S, S + n);

  for ( int d = 0; d < n-1; d++ ){
    int c = d+1;
    int a = n-1;
    int b = n-2;
    if ( d == n-3 ){
      b = d-1;
    } else if ( d == n-2 ){
      a = d-2;
      b = d-1;
    }
    res = max(res, 1.0*(S[a]+S[b])/(S[c]-S[d]));
  }

  printf("%.8lf\n", res);
}
```

問４　必勝７並べ

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 7並べ(簡易版)で二人が勝負する.
- 先手のカードの番号が与えられたとき、後手がどのようにカードを出してきても、先手が勝つ手順が少なくとも一つあるか判定せよ.

総評

- 正答数13.
- ゲーム理論に関連する問題である.
- ゲームの性質を理解し、シミュレーション、探索、貪欲法などを用いて問題を解決できるかが問われている.

問４　必勝７並べ

AIZU
THE UNIVERSITY OF AIZU

解法１

- 再帰を用いてシミュレーションを行う.
- 先手は、その局面で可能な手をすべて試す.
  - １つでも成功すれば、先手が勝つことができる（trueを返す）.
- 後手も、その局面で可能な手をすべて試す.
  - １つでも成功すれば、先手が勝つことはできない（falseを返す）.
- 計算回数は3^{12}程度.　実際はもっと少ない.

問４　必勝７並べ

AIZU
THE UNIVERSITY OF AIZU

解答例１（Ｃ＋＋）

```
// プレイヤー、先手インデックス、後手インデックス、場の最小・最大値
bool dfs(int p, int ap, int aq, int bp, int bq, int l, int r){
  if ( p == 1 ){ // 先手
    if ( ap >= AP.size() && aq >= AQ.size() ) return true; // 先手の勝ち
  } else { // 後手
    if ( bp >= BP.size() && bq >= BQ.size() ) return false; // 先手の負け
  }
  int cnt = 0;
  if ( p == 0 ) { // A 先手
    if ( check(AP, ap, l-1) ){ // 小さいカードを出せる
      cnt++;
      if ( dfs((p+1)%2, ap+1, aq, bp, bq, l-1, r) ) return true;
    }
    if ( check(AQ, aq, r+1) ){ // 大きいカードを出せる
      cnt++;
      if ( dfs((p+1)%2, ap, aq+1, bp, bq, l, r+1) ) return true;
    }
    if ( cnt == 0 ){ // パス
      if ( dfs((p+1)%2, ap, aq, bp, bq, l, r) ) return true;
    }
    return false;
  } else { // B
    ...
```

問４　必勝７並べ

AIZU
THE UNIVERSITY OF AIZU

解答例1（C＋＋）つづき

```
    ...
    ...
    } else { // B 後手
      bool possible = true; // 先手が勝てる
      if ( check(BP, bp, l-1) ){
        cnt++;
        if ( !dfs((p+1)%2, ap, aq, bp+1, bq, l-1, r) ) possible = false;
      }
      if ( check(BQ, bq, r+1) ){
        cnt++;
        if ( !dfs((p+1)%2, ap, aq, bp, bq+1, l, r+1) ) possible = false;
      }
      if ( cnt == 0 ){ // pass
        if ( !dfs((p+1)%2, ap, aq, bp, bq, l, r) ) possible = false;
      }
      return possible;
    }
}
```

問４　必勝７並べ

AIZU
THE UNIVERSITY OF AIZU

解法２

- 「後手のどのようにカードを出してきても、先手が勝つ手順が少なくとも一つあるか判定せよ」 ⇔ 「両者が最適な戦略をとった場合、先手が勝てるか？」.
- 貪欲法でシミュレーションを行う.
  - １と13を両方持っているプレイヤーは絶対に勝てない.
  - １と13を両方持っていないプレイヤーは必ず勝てる.
  - 考えるべきは、１と１３がそれぞれ別のプレイヤーに渡されたとき.
  - １だけを持っているプレイヤーは小さいカードから優先的に出すのが最適な戦略である.
  - 13だけを持っているプレイヤーは大きいカードから優先的に出すのが最適な戦略である.
  - この戦略のもと、どちらが先に手札を場にだせるかのシミュレーションを行えばよい.

問４　必勝７並べ

AIZU
THE UNIVERSITY OF AIZU

解答例2（C＋＋）

```
bool solve(){
  ...
  if ( in[13] && in[1] ) return false;
  if ( !in[13] && !in[1] ) return true;

  int minv = 7, maxv = 7;

  if ( in[1] ){ // 1 を持っている
    while(1){
      if ( AP[0] == minv-1 ) { minv--; AP.erase(AP.begin()); } // 小さい方から
      else if ( AQ[0] == maxv+1 ) { maxv++; AQ.erase(AQ.begin()); } // 大きい方から
      if ( AP.size() + AQ.size() == 0 ) return true;
      if ( BQ[0] == maxv+1 ) { maxv++; BQ.erase(BQ.begin()); } // 大きい方から
      else if ( BP[0] == minv-1 ) { minv--; BP.erase(BP.begin()); } // 小さい方から
      if ( BP.size() + BQ.size() == 0 ) return false;
    }
  } else { // 13 を持っている
    while(1){
      if ( AQ[0] == maxv+1 ) { maxv++; AQ.erase(AQ.begin()); } // 大きい方から
      else if ( AP[0] == minv-1 ) { minv--; AP.erase(AP.begin()); } // 小さい方から
      if ( AP.size() + AQ.size() == 0 ) return true;
      if ( BP[0] == minv-1 ) { minv--; BP.erase(BP.begin()); } // 小さい方から
      else if ( BQ[0] == maxv+1 ) { maxv++; BQ.erase(BQ.begin()); } // 大きい方から
      if ( BP.size() + BQ.size() == 0 ) return false;
    }
  }
}
```

問５　環状すごろく

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 環状にマスが並んだすごろくについて、あがりにたどりつけるマスの個数を求めよ.
- 各マスには１つの整数が書かれており、その数だけ駒を時計回り進める.
- あるマスを出発して駒を進め、始点に戻ってくることができたらあがりとなる.
- マスの数Nは100000以下.

総評

- 正答数14.
- 有向グラフの閉路を見つける問題である.
- グラフのアルゴリズムを選択し、それを実装できるかが問われている.
- 問題の特性（出次数が１である）を利用して、よりシンプルなアルゴリズムを実装することができる.

問５　環状すごろく

AIZU
THE UNIVERSITY OF AIZU

解法１

- 深さ優先探索
  - 探索中にBackedgeが存在するかどうかで閉路を検出する.

解法２

- 幅優先探索
  - 入次数が０のノードをキューに入れていく幅優先探索.

問５　環状すごろく

AIZU
THE UNIVERSITY OF AIZU

解答例１（Ｃ＋＋）幅優先探索

```
main(){
  cin >> n;
  for ( int i = 0; i < n; i++ ) in[i] = 0;
  for ( int i = 0; i < n; i++ ) {
    int v; cin >> v;
    v = (i+v)%n;
    P[i] = v;
    in[v]++;
  }

  queue<int> Q;
  for ( int i = 0; i < n; i++ )
    if ( in[i] == 0 ) Q.push(i); // 入次数が1の頂点をキューに入れる

  int res = 0;
  while(!Q.empty()){
    int u = Q.front(); Q.pop();
    res++;
    int v = P[u];
    in[v]--;
    if ( in[v] == 0 ) Q.push(v); // 入次数が1の頂点をキューに入れる
  }
  cout << n - res << endl;

  return 0;
}
```

問５　環状すごろく

AIZU
THE UNIVERSITY OF AIZU

解答例2（Ｃ＋＋） ループ検出

```
int n;
int a[100005], x[100005], y[100005];

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

  memset(x,-1,sizeof(x));
  memset(y,-1,sizeof(y));

  int ans=0;
  for(int i=0;i<n;i++){
    int p=i,c=0;
    while(x[p]==-1){
      x[p]=c++; // 未訪問なら-1，深さ
      y[p]=i;   // 未訪問なら-1，どこを出発点にして訪れたか
      p=(p+a[p])%n;
    }
    if(y[p]==i)ans+=(c-x[p]); // 閉路上にある頂点の数を加算
  }

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

問６　実数既約分数化

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 分数で表すことができる実数を入力し、その実数と等しい既約分数を出力する.
- 入力される実数が、循環小数の場合は、循環する部分が (...) で与えられる.

総評

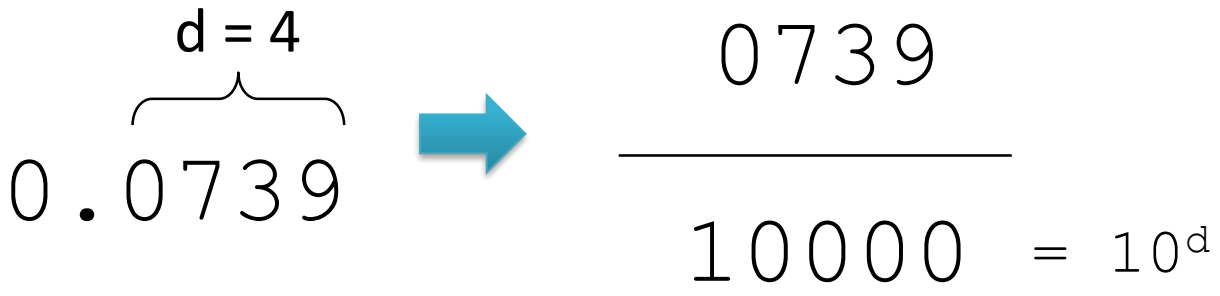
- 正答数14.
- 高校数学の知識に加えて、プログラムによる文字列処理が行えるかが問われている.
  - 文字列中の文字の扱い
  - 文字列の整数への変換
  - 構文解析

問６　実数既約分数化

AIZU
THE UNIVERSITY OF AIZU

解法（循環がない場合）

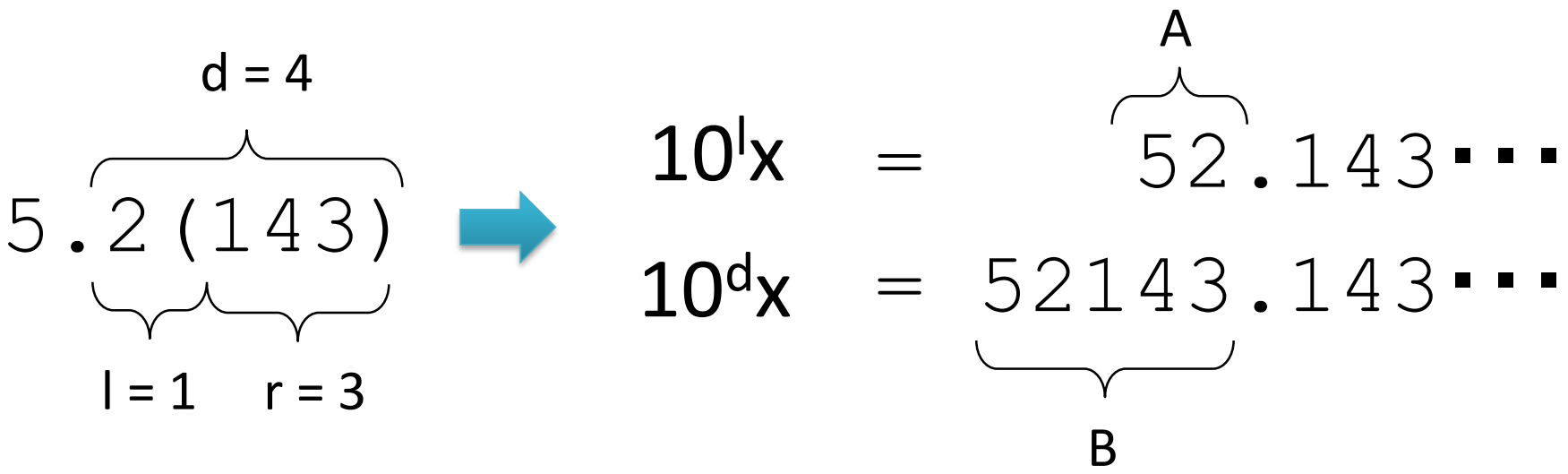
- 小数点以下の桁数を求める (= d).
- 入力 x を整数として表し、x / 10^d を既約分数化する.
- 既約分数化するためには、分母と分子をそれらの最大公約数で割る.



問６　実数既約分数化

解法（循環小数の場合）

- 小数点以下の桁数 (= d)、循環部分の桁数 (= r) を求める.
- 循環しない部分の桁数を求める (l = d − r).
- 入力値xをそれぞれ10^l倍10^d倍した10^lxと10^dxを求める.

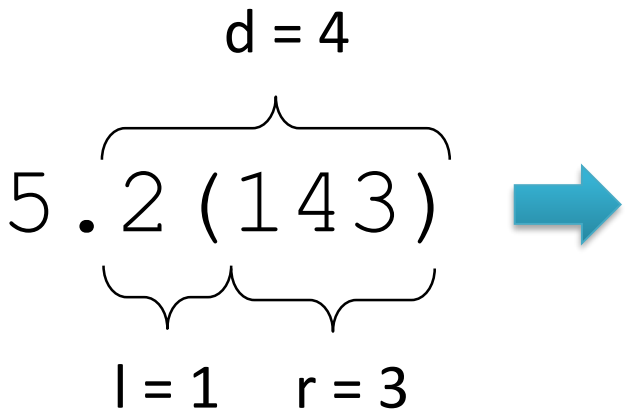


問６　実数既約分数化

AIZU
THE UNIVERSITY OF AIZU

解法（循環小数の場合）（つづき）

- $10^l x$と$10^d x$の小数点以下が同じになる.
- $10^d x - 10^l x$ は整数となる.



$$10^l x = 52.143\cdots$$

$$10^d x = 52143.143\cdots$$

A, B

$$10^d x - 10^l x = B - A$$

問６　実数既約分数化

AIZU
THE UNIVERSITY OF AIZU

解答例１（Ｃ＋＋）

```
int print(int x, int y){
  int g = gcd(x, y);
  cout << x/g << "/" << y/g << endl;
}

main(){
  string in; cin >> in;
  int o, l, r;
  l = r = in.size(); r++;
  string all = "", sub = "";
  for ( int i = 0; i < in.size(); i++ ){
    if ( in[i] == '.' ){
      o = i;
    } else if ( in[i] == '(' ) {
      l = i;
      sub = all;
    } else if ( in[i] == ')' ){
      r = i;
    } else {
      all += in[i];
    }
  }
  if ( r-l-1 == 0 ){
    print(atoi(all.c_str()), pow10(l-o-1) );
  }else {
    print(atoi(all.c_str()) - atoi(sub.c_str()), pow10(l-o-1 + r-l-1) - pow10(l-o-1) );
  }
}
```

問６　実数既約分数化

AIZU
THE UNIVERSITY OF AIZU

解答例２（Ｃ＋＋）

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

int main(){
  string s;
  cin>>s;
  int a=0,b=0,c=0,k=0;
  for(int i=0;i<s.size();i++){
    if(s[i]=='.'){k=1;continue;}
    if(s[i]=='('){
      while(s[++i]!=')')b*=10,c*=10,b+=s[i]-'0',c+=9;
    }
    else k*=10,a*=10,a+=s[i]-'0';
  }
  if(b)a=b+a*c,k*=c;
  int g=__gcd(a,k);
  cout<<a/g<<'/'<<k/g<<endl;

  return 0;
}
```

問7　静かな町

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 重み付き無向グラフが与えられる.
- ２つの頂点の組み合わせの中で、最短距離の最大値をDとする.
- 最短距離がDであるような２つの頂点の組み合わせ(s, t)について、s から t までの最短経路上になり得ない頂点を列挙せよ.
- 頂点数Nは1500以下、辺の数Rは3000以下.

総評

- 正答数10.
- グラフアルゴリズムの応用問題である.
- 最短経路のアルゴリズムの仕組みを理解し、応用することができるかが問われている.

問7　静かな町

AIZU
THE UNIVERSITY OF AIZU

誤答（時間制限）

- ワーシャルフロイドで全対最短距離を求める.
- 始点 i 、終点 j 、経由点 k の組み合わせを全探索し、d[i][k] + d[k][j] が最大距離Dと一致する経由点 k を候補から外す.
- $O(N^3)$ で制限時間を超えてしまう.

問7　静かな町

AIZU
THE UNIVERSITY OF AIZU

解法 1

- ダイクストラ法で最短距離を求める.
  - 最短経路の最大値Dを求める.
  - Dをグラフの直径と呼ぶ.
- 始点sを全探索し、最短距離がDとなるような終点から深さ優先探索または幅優先探索を行う.
  - ダイクストラ法の距離配列を保持しておくか、最短経路木を復元するための親頂点の配列を用いる.
  - 最短距離が同じになる親頂点をリスト等で保持しておく.
- N回の探索とN回のダイクストラ法を行うので O(NR log N)のアルゴリズムとなる.

問7　静かな町

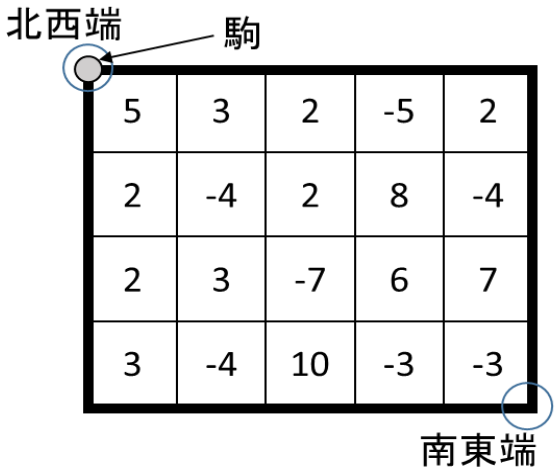
AIZU
THE UNIVERSITY OF AIZU

解答例（C/C＋＋）

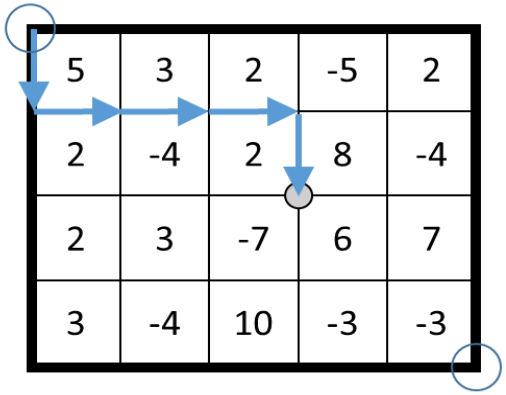
```
void dijkstra(int s){
  int u, v, c;
  priority_queue<pair<int, int> > PQ;
  ....
  d[s] = 0;
  PQ.push(make_pair(0, s));
  pair<int, int> f;
  while(!PQ.empty()){
    f = PQ.top(); PQ.pop();
    u = f.second;
    c = (-1)*f.first;
    vis[u] = true;
    if ( d[u] < c ) continue;
    for ( int j = 0; j < adj[u].size(); j++ ){
      v = adj[u][j].first;
      if ( vis[v] ) continue;
      if ( d[v] == d[u] + adj[u][j].second ){
        P[v].push_back(u); // 最短経路木における親のリストに追加
        PQ.push(make_pair(d[v]*(-1), v));
      } else if ( d[v] > d[u] + adj[u][j].second ){
        d[v] = d[u] + adj[u][j].second;
        P[v].clear(); // 最短経路木における親のリストを初期化し、追加
        P[v].push_back(u);
        PQ.push(make_pair(d[v]*(-1), v));
      }
    }
  }
}
```

問８　勢力の予想

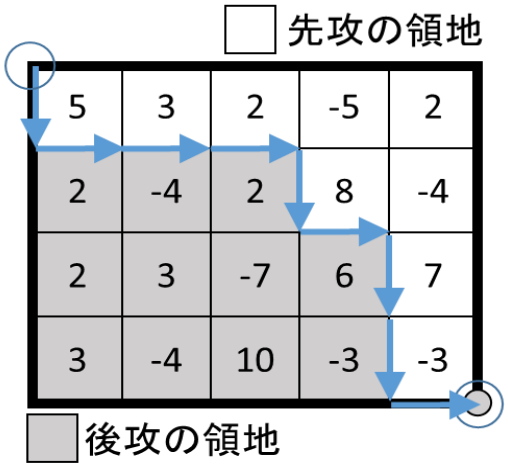
問題概要



①島と区画



②ゲーム中



③ゲーム終了後の領地

- 二人のプレイヤーがグリッドの格子点のコマを、下または右に移動して境界線を作る.
- 境界線から上（右）・下（左）の領域を先手・後手の領域とする.
- プレイヤーは自分の領域の区画に書かれた整数の合計の損得を得る.

問８　勢力の予想

AIZU
THE UNIVERSITY OF AIZU

問題概要（つづき）

- 両者がそれぞれ、自分のスコアから相手のスコアを引いた値を最大化するために最適な行動をとる.
- 両者のスコアの差の絶対値を求めよ.

総評

- 正答数9.
- ゲーム理論に関する問題.
- ゲームの性質を理解し、効率的な方法で問題を解決できるかが問われている.

問８　勢力の予想

AIZU
THE UNIVERSITY OF AIZU

解法

- 駒の場所を状態にした動的計画法．dp[i][j]を以下のように定める．
- dp[i][j] := 駒を南に i 回、東に j 回動かした時点の、自分がその後の操作で獲得できるスコア − 相手がその後の操作で獲得できるスコア
- dp[i][j] :=

  0　　(i = H, 駒が南端)

  score(i, j) − dp[i+1][j]　　(j = W, 駒が東端)

  max(-dp[i][j+1], score(i, j) − dp[i+1][j])
- dp[0][0] が答えになる．
- このdp配列を末尾（南東端）から計算していく．

問８　勢力の予想

AIZU
THE UNIVERSITY OF AIZU

解法（つづき）

- -dp[i][j+1] は、駒を東に進めた際の「相手のスコア − 自分のスコア」である.
  - 東に進めるという操作ではスコアを獲得しない.
  - dp[i][j+1]ではプレイヤーが逆転しているので正負を逆転したものになる.
- score(i, j) − dp[i+1][j] は、駒を南に進めた際の「相手のスコア − 自分のスコア」である.
  - score(i, j) は(i, j)から南に進めるという操作で生じる、「自分のスコア − 相手のスコア」の差分を表す.
  - dp[i+1][j]ではプレイヤーが逆転しているので正負を逆転したものになる.
- score(i, j) : = i+1行目の区画（入力データ）において、

  (j+1)列目からW列目までの区画の整数の和　= R

  1列目からj列目までの区画の整数の和 = L , としたとき

  score(i, j) := R − L　　（先手の場合）

  score(i, j) := L - R　　（後手の場合）

問８　勢力の予想

解法（つづき）

- これら２種類の遷移でプレイヤーが行える操作は網羅されているため、最適な操作を行う場合は大きい方を選べばよい.
- dp[i][j]の計算がdp[i][j+1]とdp[i+1][j]の計算結果に依存するが、これらに循環はない.
- dp[H][W] から始めるボトムアップな動的計画法．または、再帰＋メモ化で始点(0, 0)のスコアを求めることができる.
- 計算量は O(HW).
- score(i, j) の計算を O(1) で行うため、区画の累積和をとっておく.

5	3	2	-5	2
2	-4	2	8	-4
2	3	-7	6	7
3	-4	10	-3	-3

5	8	10	5	7
2	-2	0	8	4
2	5	-2	4	11
3	-1	9	6	3

問８　勢力の予想

AIZU
THE UNIVERSITY OF AIZU

解答例（Ｃ＋＋）

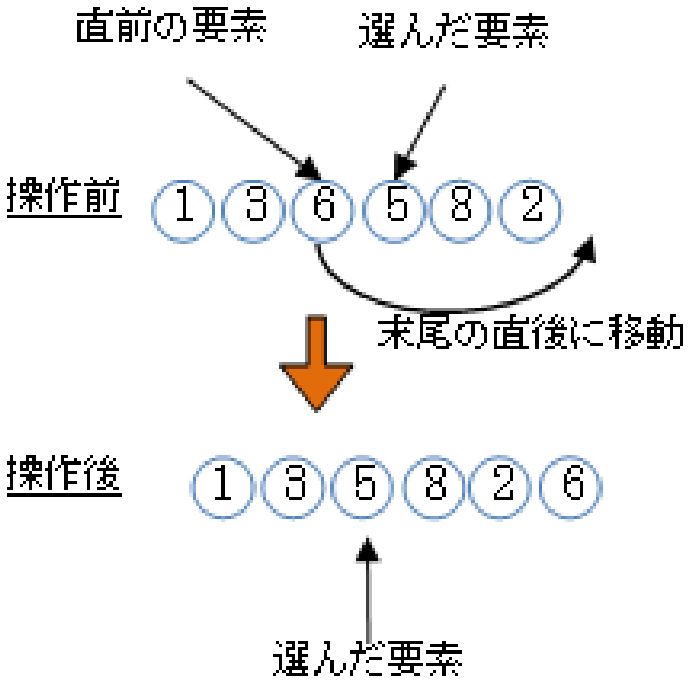
```
long long score(int i, int j){
  if ( (i+j)%2 == 0 ) return (G[i+1][W] - G[i+1][j]) - G[i+1][j];
  return G[i+1][j] - (G[i+1][W] - G[i+1][j]);
}

main(){
  cin >> W >> H;
  for ( int i = 0; i <= H; i++ ) G[i][0] = 0;
  for ( int i = 1; i <= H; i++ ){
    for ( int j = 1; j <= W; j++ ) {
      scanf("%lld",&G[i][j]);
      G[i][j] += G[i][j-1];
    }
  }
  for ( int i = H; i >= 0; i-- ){
    for ( int j = W; j >= 0; j-- ){
      if ( i == H ) {
        dp[i][j] = 0;
      } else if ( j == W ){
        dp[i][j] = score(i, j) - dp[i+1][j];
      } else {
        dp[i][j] = max(-dp[i][j+1], score(i, j) - dp[i+1][j]);
      }
    }
  }
  cout << abs(dp[0][0]) << endl;
```

問９　ソート

問題概要

- 次に示すソートアルゴリズムの、要素の移動回数を求めよ.

1. 数列の先頭の要素を選ぶ.
2. 選んだ要素の直前の要素（ある場合）が、選んだ要素よりも小さい間、直前の要素を数列の末尾に移動する.
3. 選んだ要素が最後尾なら終了．そうでなければ、選んだ要素の直後の要素を新たに選び、２．へ戻る



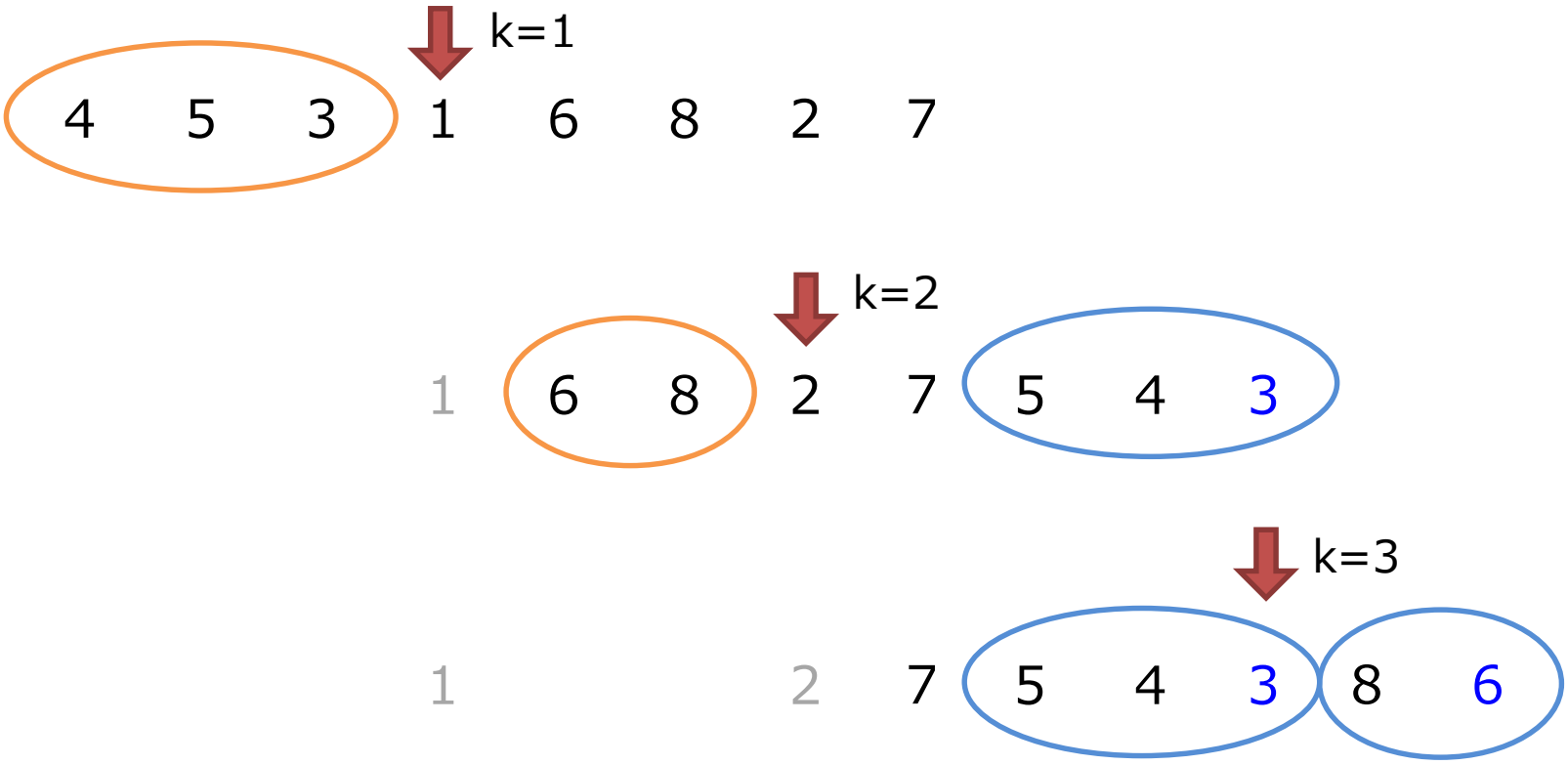
総評

- 正答数１.
- 高度なアルゴリズムの設計力と実装力が問われている.

問９　ソート

考察

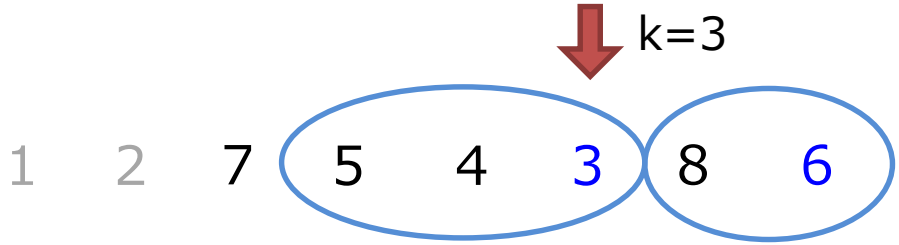
- 前処理として、入力の数列は 1, 2, … N の集合に圧縮しておく.
- 先頭から順番に、k = 1, 2, … N を決めていく.



問９　ソート

考察

- 前処理として、入力の数列は 1, 2, … N の集合に圧縮しておく.
- 先頭から順番に、k = 1, 2, … N を決めていく.



集合の中で、最も小さい要素が右に来る.
最も小さい値の場所が分かればよい.

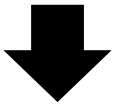
問９　ソート

AIZU
THE UNIVERSITY OF AIZU

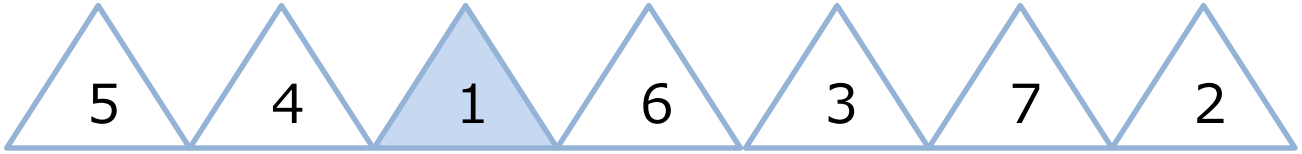
解法

- データ構造(集合)のマージ.

5　4　1　6　3　7　2

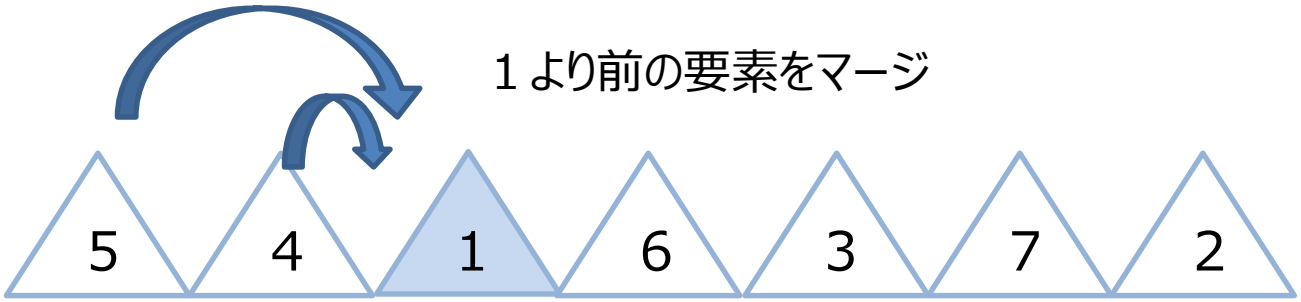


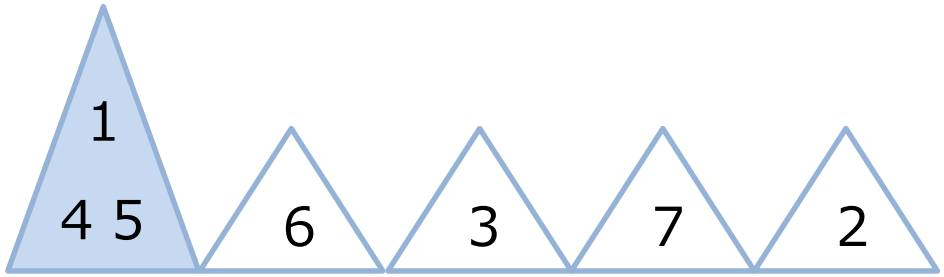
それぞれの要素を集合に入れる



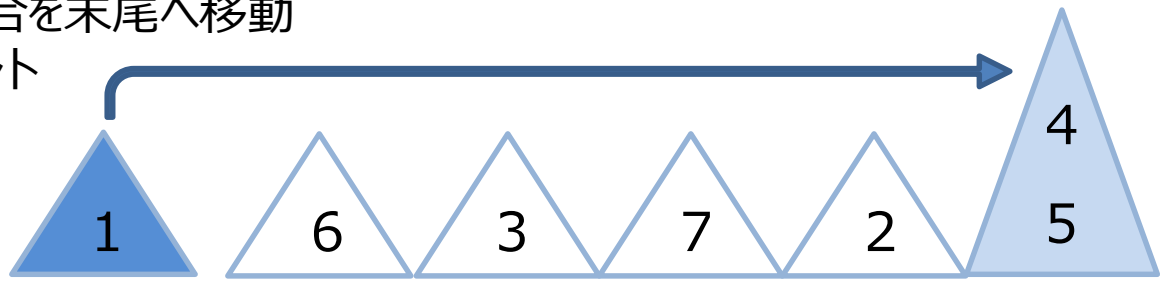
1 (=k)から順番に見ていく

問９　ソート

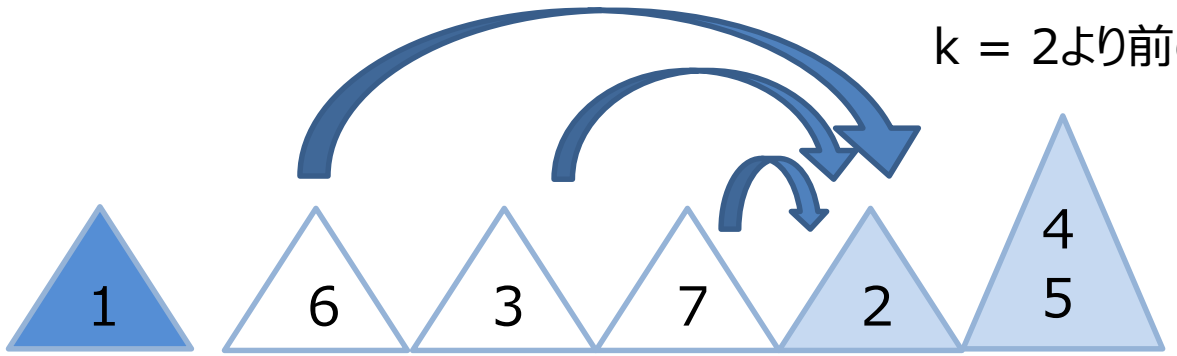




１を残して、残りの集合を末尾へ移動
<u>集合の要素数</u>をカウント



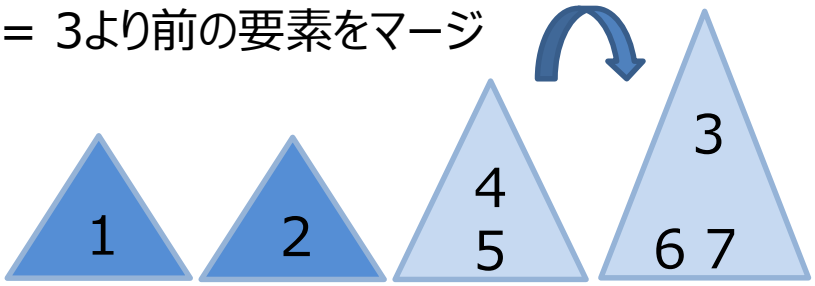
問９　ソート

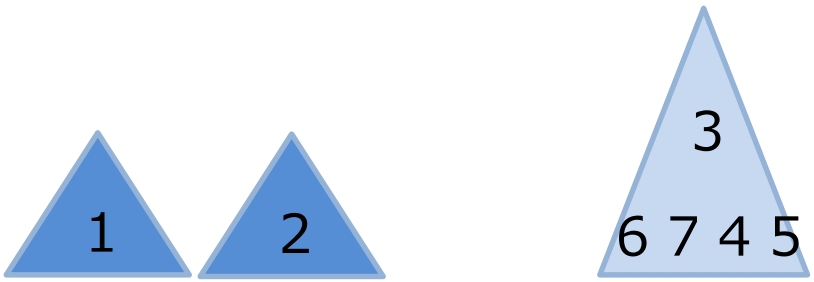


2を残して、残りの集合を末尾へ移動
<u>集合の要素数</u>をカウント

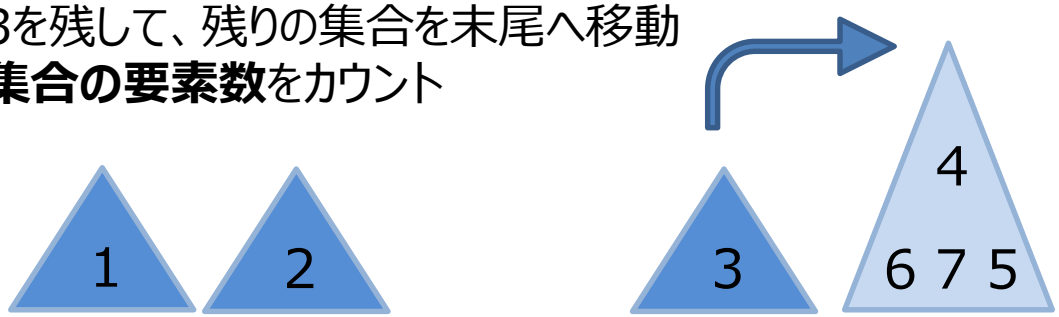
問９　ソート

k ＝ 3より前の要素をマージ





3を残して、残りの集合を末尾へ移動
集合の要素数をカウント



問９　ソート

AIZU
THE UNIVERSITY OF AIZU

解法

- 集合の移動をO(1)で行う実装が必要.
  - リスト、キューの操作.
  - 集合へのポインタを要素とする.
- 計算量はデータ構造のマージに依存する O(N log N).
  - ２つの要素をマージするときは、要素数が小さい方を大きい方へマージする.
  - ある要素（整数）に注目すると、自分が所属するデータ構造の要素数は最初は１、次は少なくとも２、次は少なくとも４、という具合に、２の指数で大きくなっていく.
  - 各整数について、高々　log N 回の移動.
  - 最初の要素数が N なので全体で O(N log N)となる.
- 集合として　priority_queue, set などを使い O(N $\log^2$ N)の実装.

問９　ソート

AIZU
THE UNIVERSITY OF AIZU

解答例（Ｃ＋＋）

```
typedef set<int> G;
G* make(int a){ G *g = new G(); g->insert(a); return g; }


// 大きい集合へ小さい集合の要素をマージする
G* merge(G* u, G* v){
  if ( u->size() < v->size() ) swap(u, v);

  for ( G::iterator it = v->begin(); it != v->end(); it++ ) u->insert(*it);
  return u;
}
```

問９　ソート

AIZU
THE UNIVERSITY OF AIZU

解答例（Ｃ＋＋）

```
int main(){
  int N; cin >> N;
  queue<G*> Q;
  vector<int> in;
  for ( int i = 0; i < N; i++ ){int a; cin>> a; in.push_back(a); }
  in = compress(in, N); // convert 1, 2, ..N

  for ( int i = 0; i < N; i++ ) Q.push(make(in[i]));
  long long ans = 0;
  for ( int k = 1; k <= N; k++ ){
    G* u = Q.front(); Q.pop();
    while(1){
      if ( *(u->begin()) == k ){
        u->erase(u->begin());
        ans += u->size();
        if ( u->size() ) Q.push(u);
        break;
      }
      G* v = Q.front(); Q.pop();
      u = merge(u, v);
    }
  }
  cout << ans << endl;
  return 0;
}
```

問10　蟻

AIZU
THE UNIVERSITY OF AIZU

問題概要

- 大きなチェス盤に、N匹の蟻がいる.
- 蟻は１単位時間に東または南に進む.
- ２匹の蟻が白マスで出会うと、お互いの方向を入れ替える.
- ２匹の蟻が黒マスで出会うと、すれ違い、方向は維持する.
- チェス盤から落下する順番に蟻の番号を報告せよ.
- N ≦ 200,000, チェス盤の大きさH, W ≦ 1,000,000,000

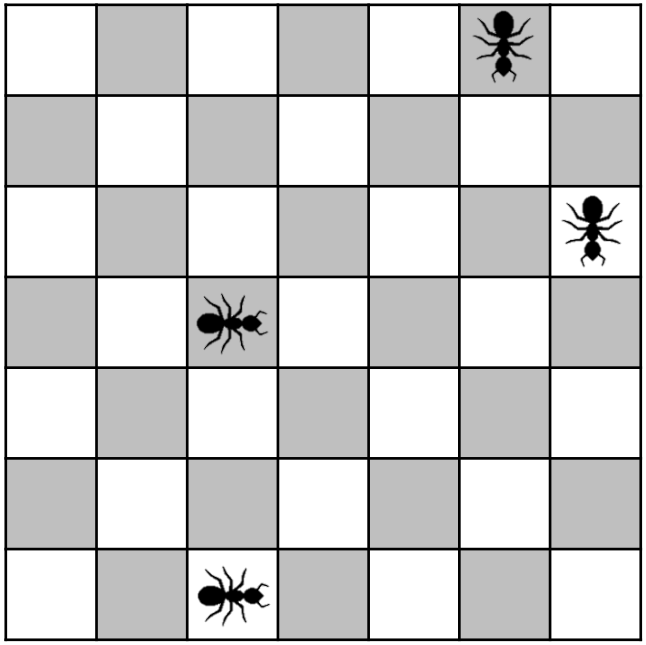
総評

- 正答数３.
- データ構造（キューなど）を応用して効率的なシミュレーションを行えるかが問われている.

問10　蟻

解法

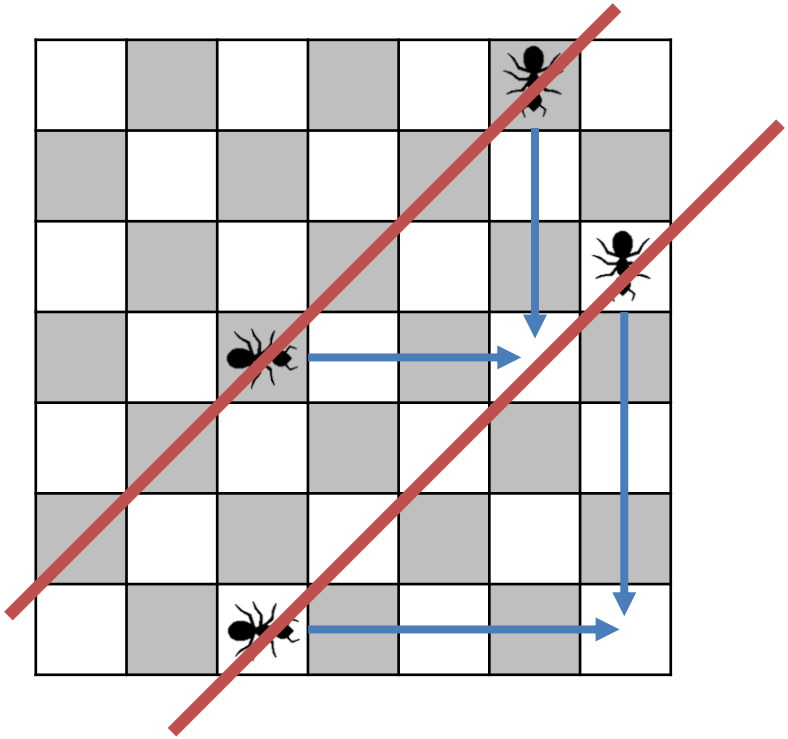
- 考察を行うと、x+yの値が等しい蟻どうしでしか、同時に同じマスに入りえないことが分かる.
- x+yの値ごとに蟻をグループ分けして、各グループごとに独立して蟻のシミュレーションを行えばよい.



問10　蟻

解法

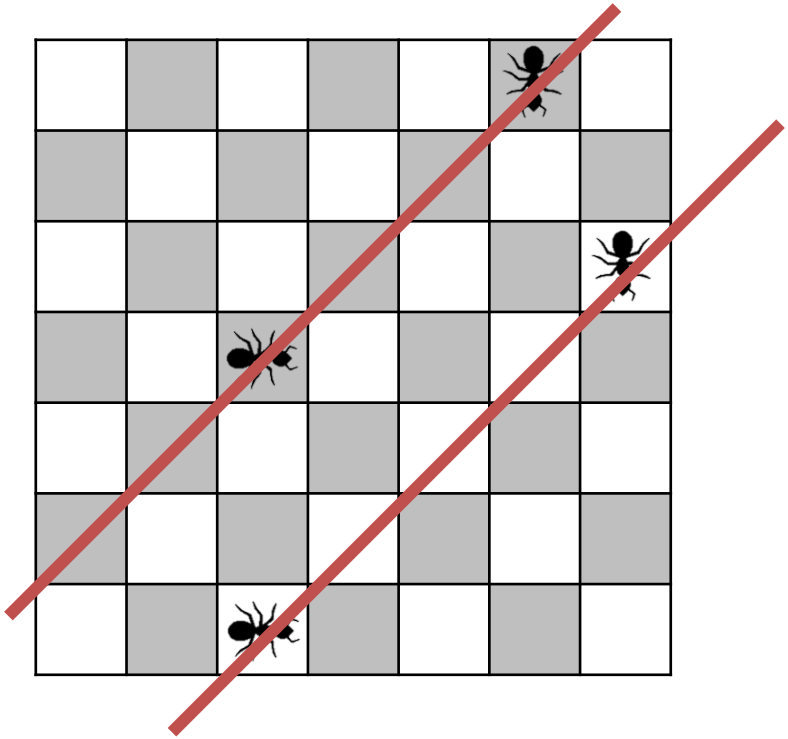
- 考察を行うと、x+yの値が等しい蟻どうしでしか、同時に同じマスに入りえないことが分かる.
- x+yの値ごとに蟻をグループ分けして、各グループごとに独立して蟻のシミュレーションを行えばよい.



問10　蟻

解法

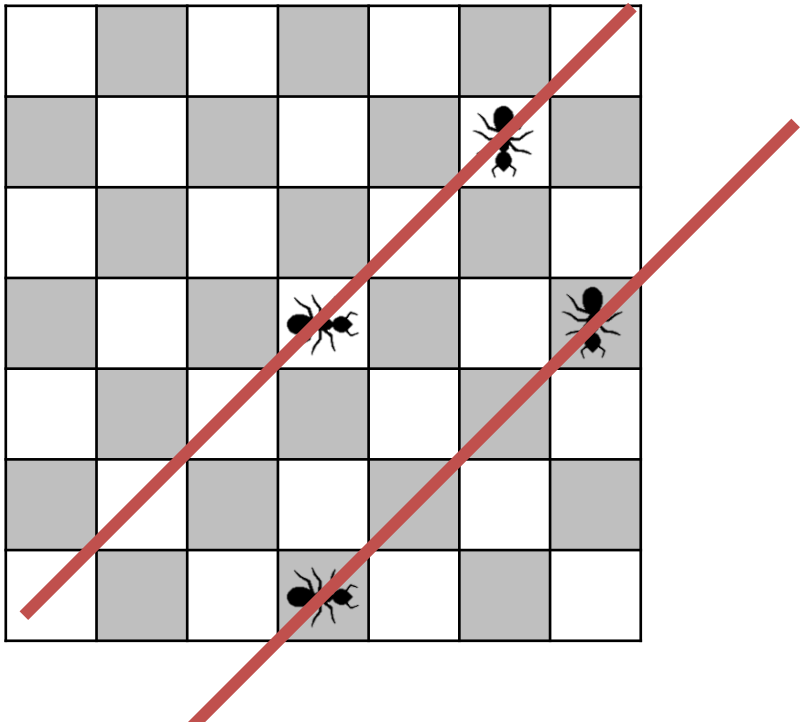
- 考察を行うと、x+yの値が等しい蟻どうしでしか、同時に同じマスに入りえないことが分かる.
- x+yの値ごとに蟻をグループ分けして、各グループごとに独立して蟻のシミュレーションを行えばよい.



問10　蟻

解法

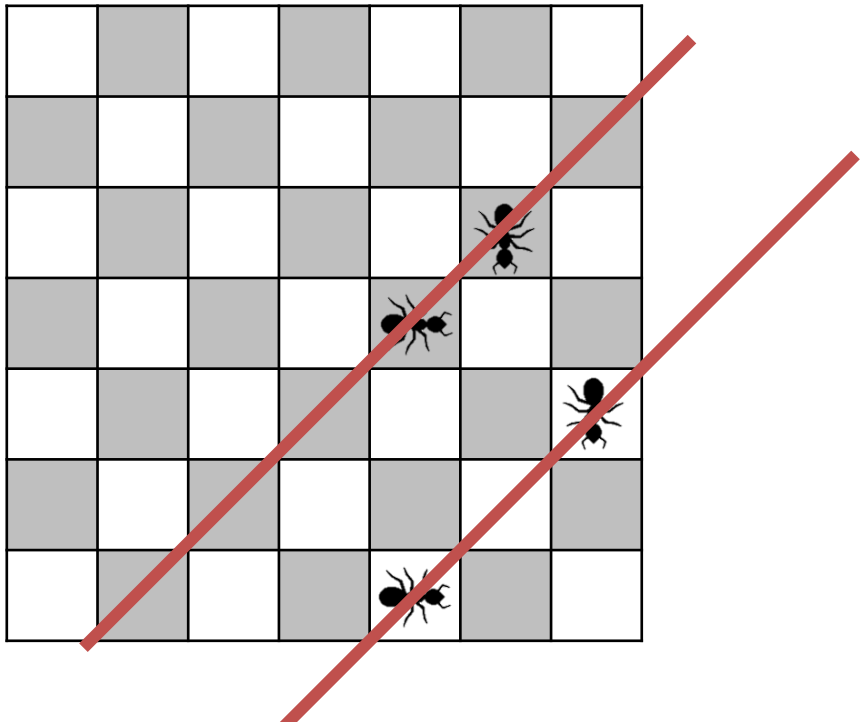
- 考察を行うと、x+yの値が等しい蟻どうしでしか、同時に同じマスに入りえないことが分かる.
- x+yの値ごとに蟻をグループ分けして、各グループごとに独立して蟻のシミュレーションを行えばよい.



問10　蟻

解法

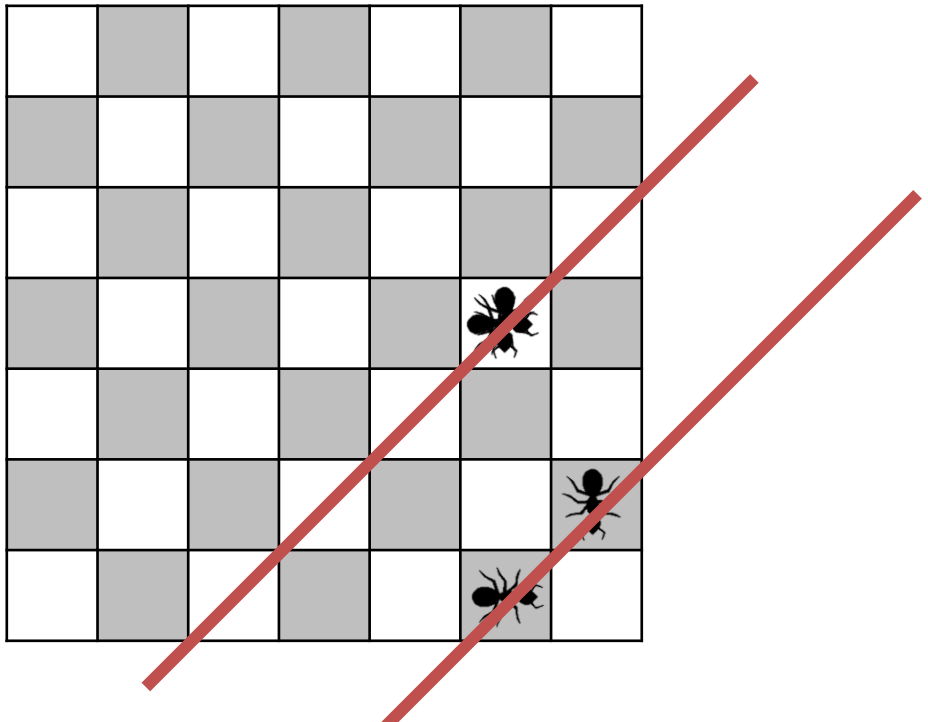
- 考察を行うと、x+yの値が等しい蟻どうしでしか、同時に同じマスに入りえないことが分かる.
- x+yの値ごとに蟻をグループ分けして、各グループごとに独立して蟻のシミュレーションを行えばよい.



問10　蟻

解法

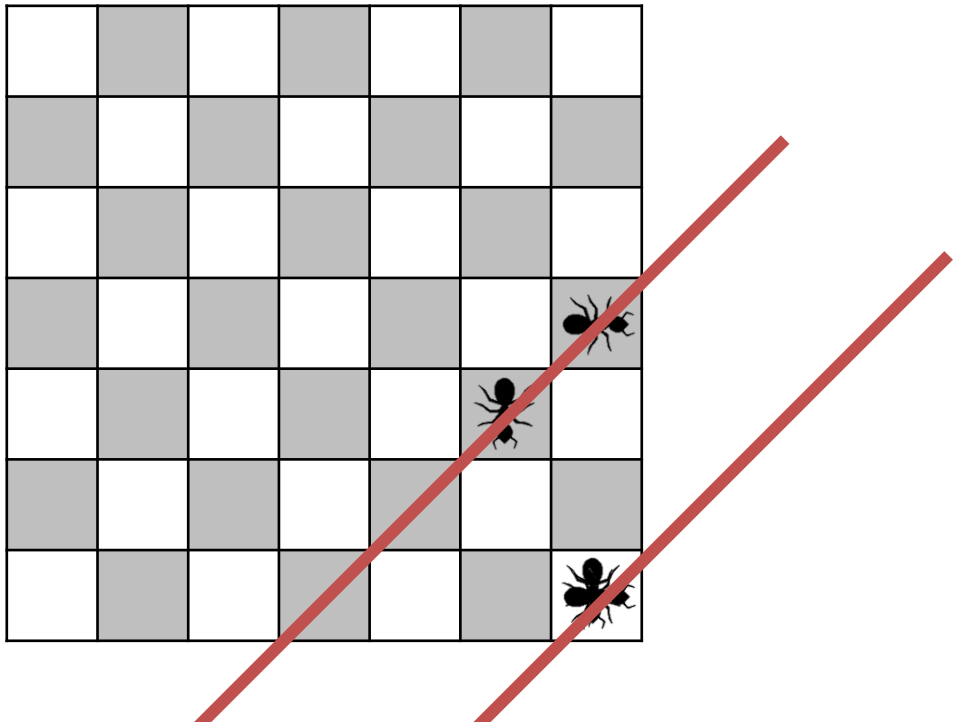
- 考察を行うと、x+yの値が等しい蟻どうしでしか、同時に同じマスに入りえないことが分かる.
- x+yの値ごとに蟻をグループ分けして、各グループごとに独立して蟻のシミュレーションを行えばよい.



問10　蟻

解法

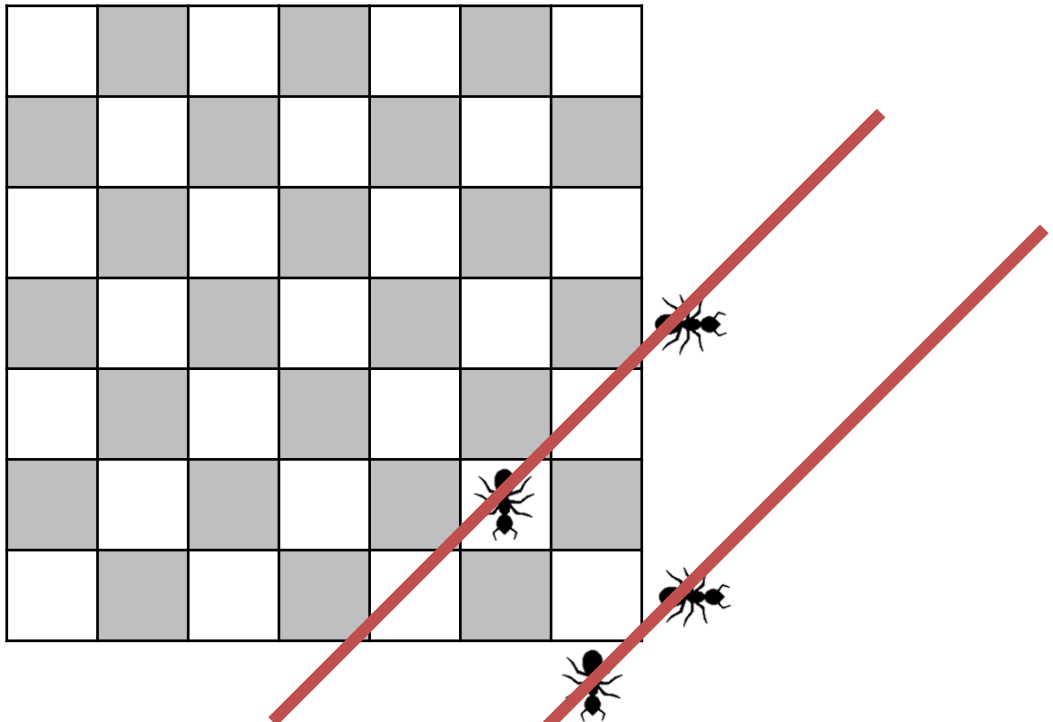
- 考察を行うと、x+yの値が等しい蟻どうしでしか、同時に同じマスに入りえないことが分かる.
- x+yの値ごとに蟻をグループ分けして、各グループごとに独立して蟻のシミュレーションを行えばよい.



問10　蟻

解法

- 考察を行うと、x+yの値が等しい蟻どうしでしか、同時に同じマスに入りえないことが分かる.
- x+yの値ごとに蟻をグループ分けして、各グループごとに独立して蟻のシミュレーションを行えばよい.

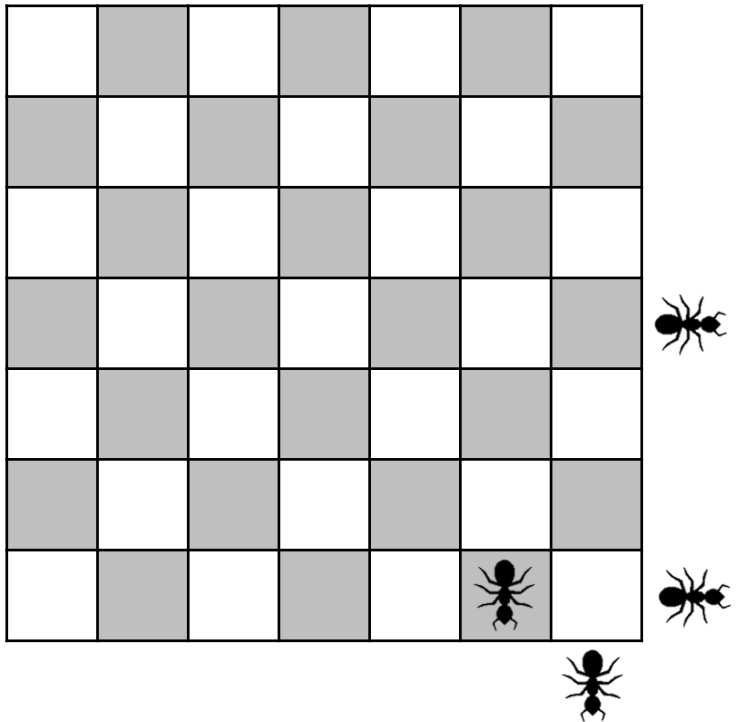


問10　蟻

AIZU
THE UNIVERSITY OF AIZU

解法

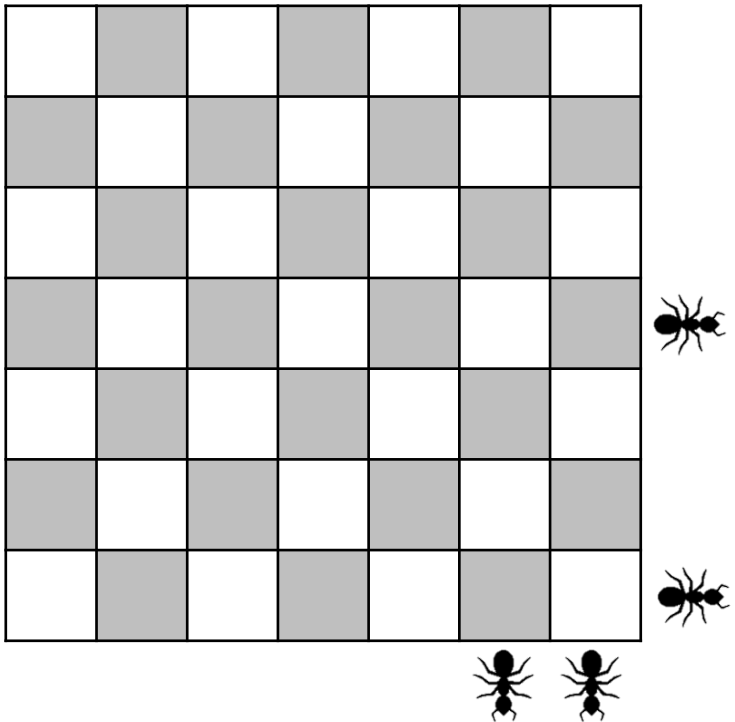
- 考察を行うと、x+yの値が等しい蟻どうしでしか、同時に同じマスに入りえないことが分かる.
- x+yの値ごとに蟻をグループ分けして、各グループごとに独立して蟻のシミュレーションを行えばよい.



問10　蟻

解法

- 考察を行うと、x+yの値が等しい蟻どうしでしか、同時に同じマスに入りえないことが分かる.
- x+yの値ごとに蟻をグループ分けして、各グループごとに独立して蟻のシミュレーションを行えばよい.



問10　蟻

AIZU
THE UNIVERSITY OF AIZU

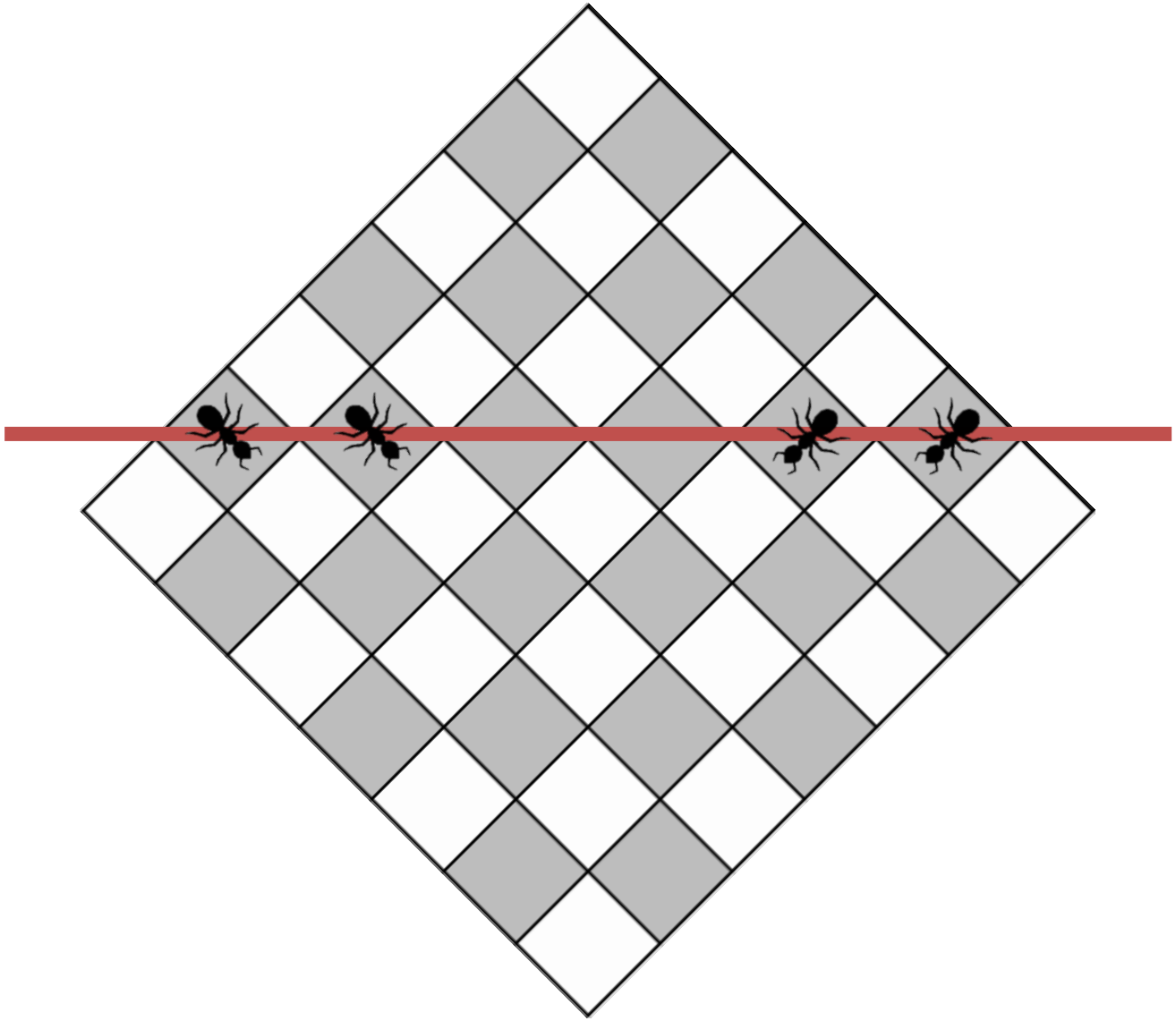
解法

- より分かり易くするため、以下のように問題を置き換える.
  - 蟻が同時に白マスへと入った場合は、互いに向きを変更しているのではなく、向きを維持したまま2匹の蟻のゼッケンを入れ替えていると考える.
    - こうした場合,答えはゼッケンが盤面の外へ出る順にソートしたものになる.
    - どのゼッケンがどのタイミングで盤面の外へ出たのかを知るためには,
      最終的にどのゼッケンがどの蟻に付けられているのかが分かればよい.
  - 盤面を45度回転させて、左向きと右向きの蟻が,1次元の直線上に並んでいると考える.
  - 右向きの蟻は停止していて、その分左向きの蟻のみが移動していると考える.
    - 蟻のゼッケンの入れ替わりだけをシミュレーションすればよいため.

問10　蟻

AIZU
THE UNIVERSITY OF AIZU

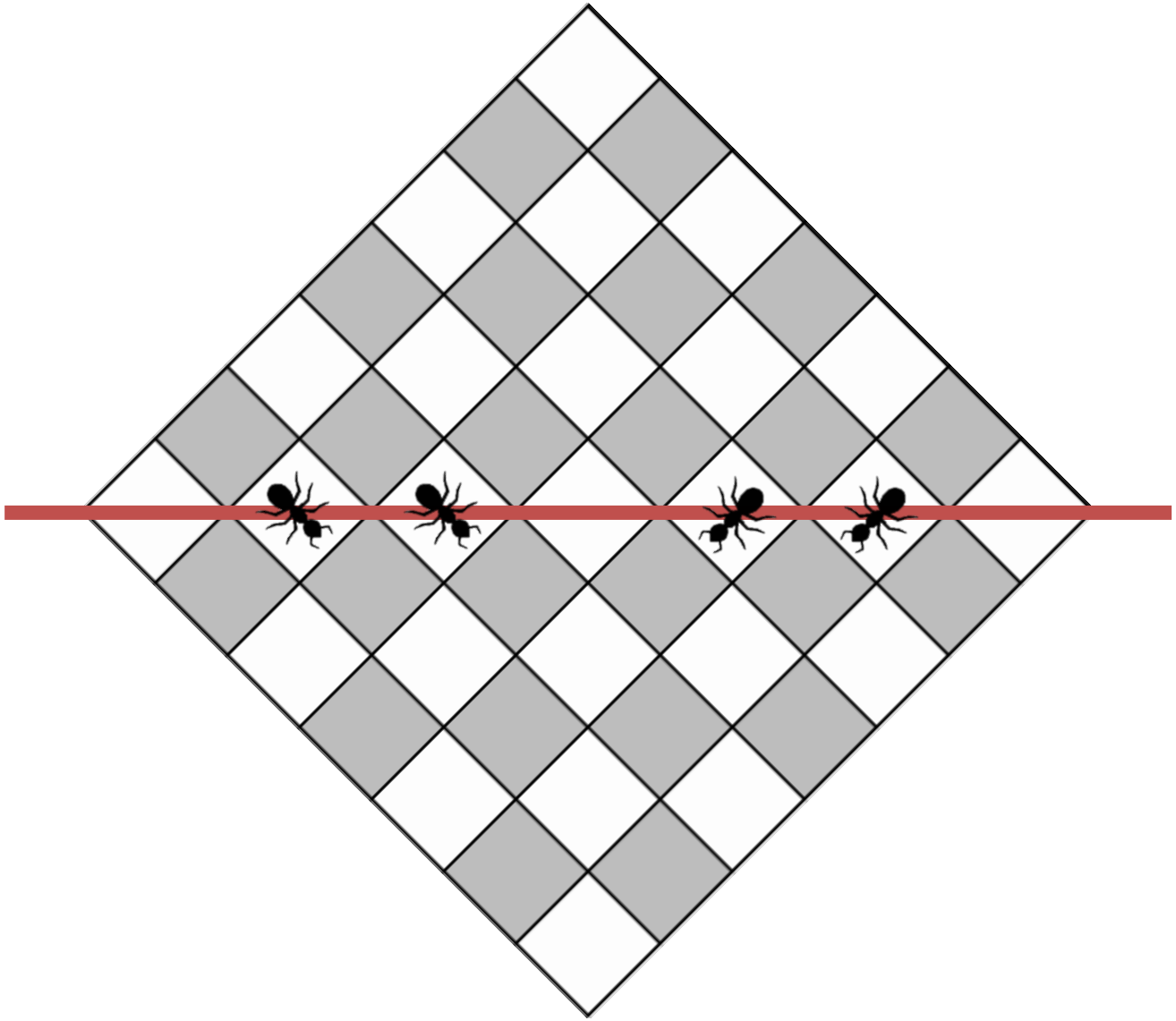
解法



問10　蟻

AIZU
THE UNIVERSITY OF AIZU

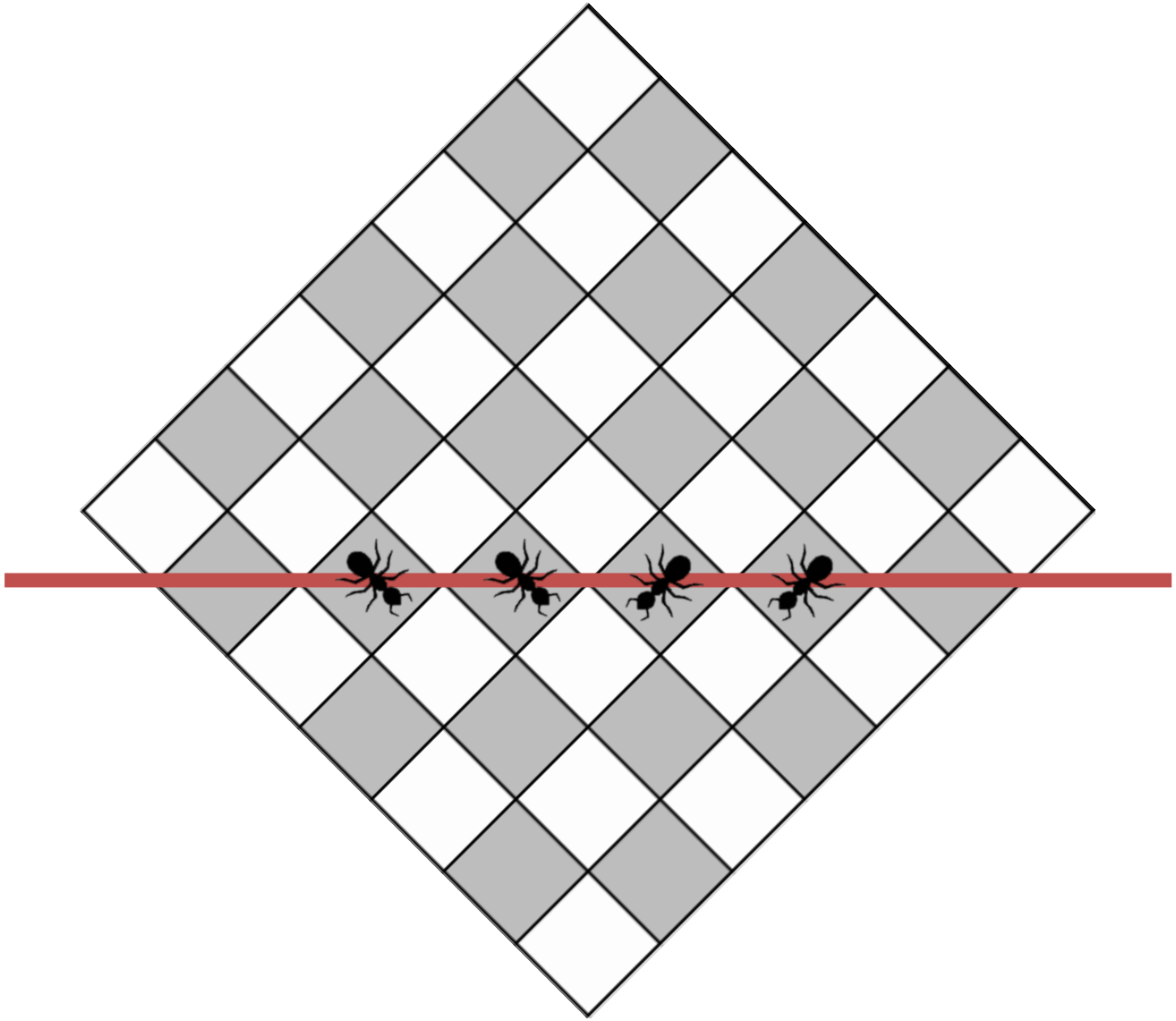
解法



問10　蟻

AIZU
THE UNIVERSITY OF AIZU

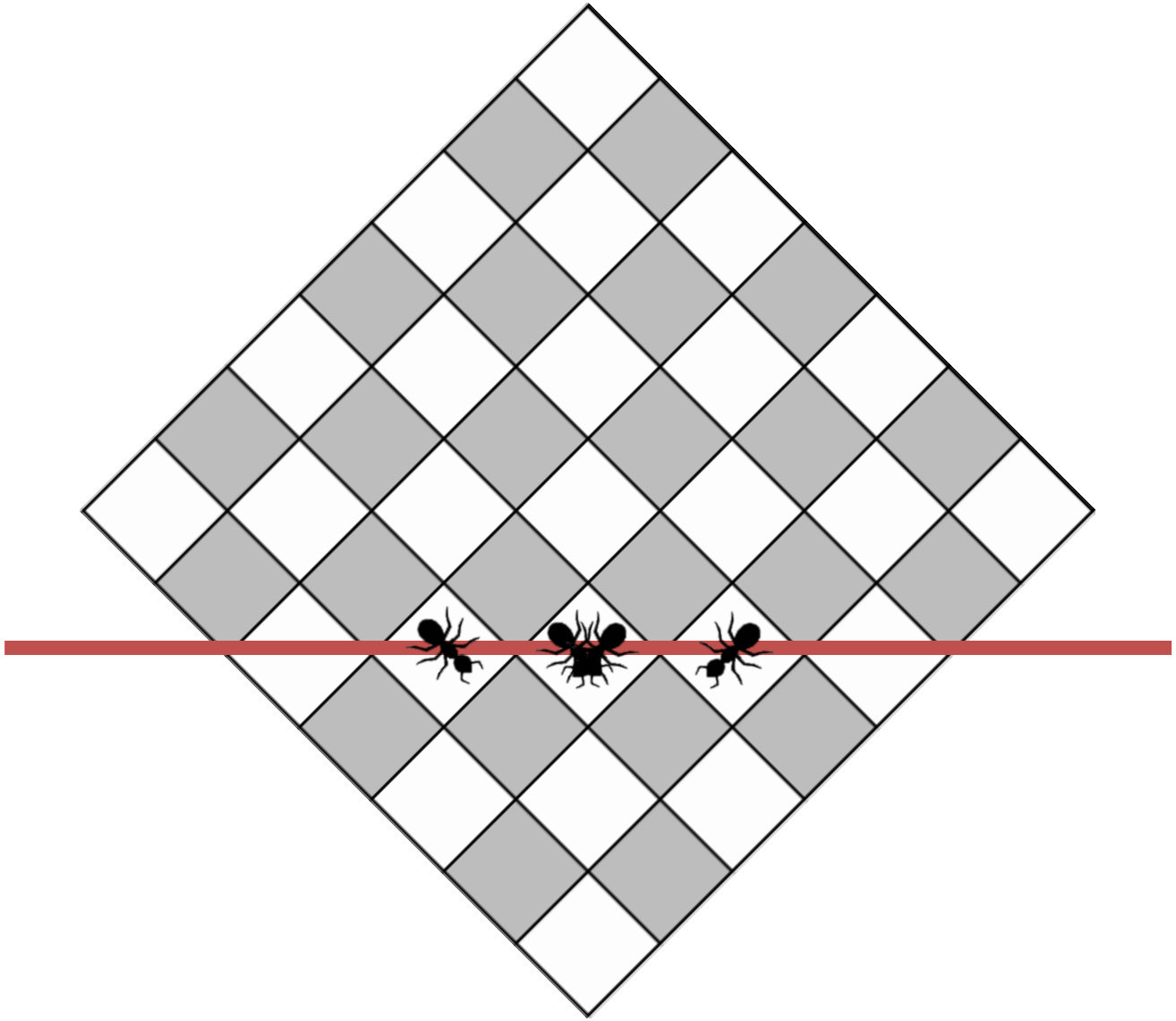
解法



問10　蟻

AIZU
THE UNIVERSITY OF AIZU

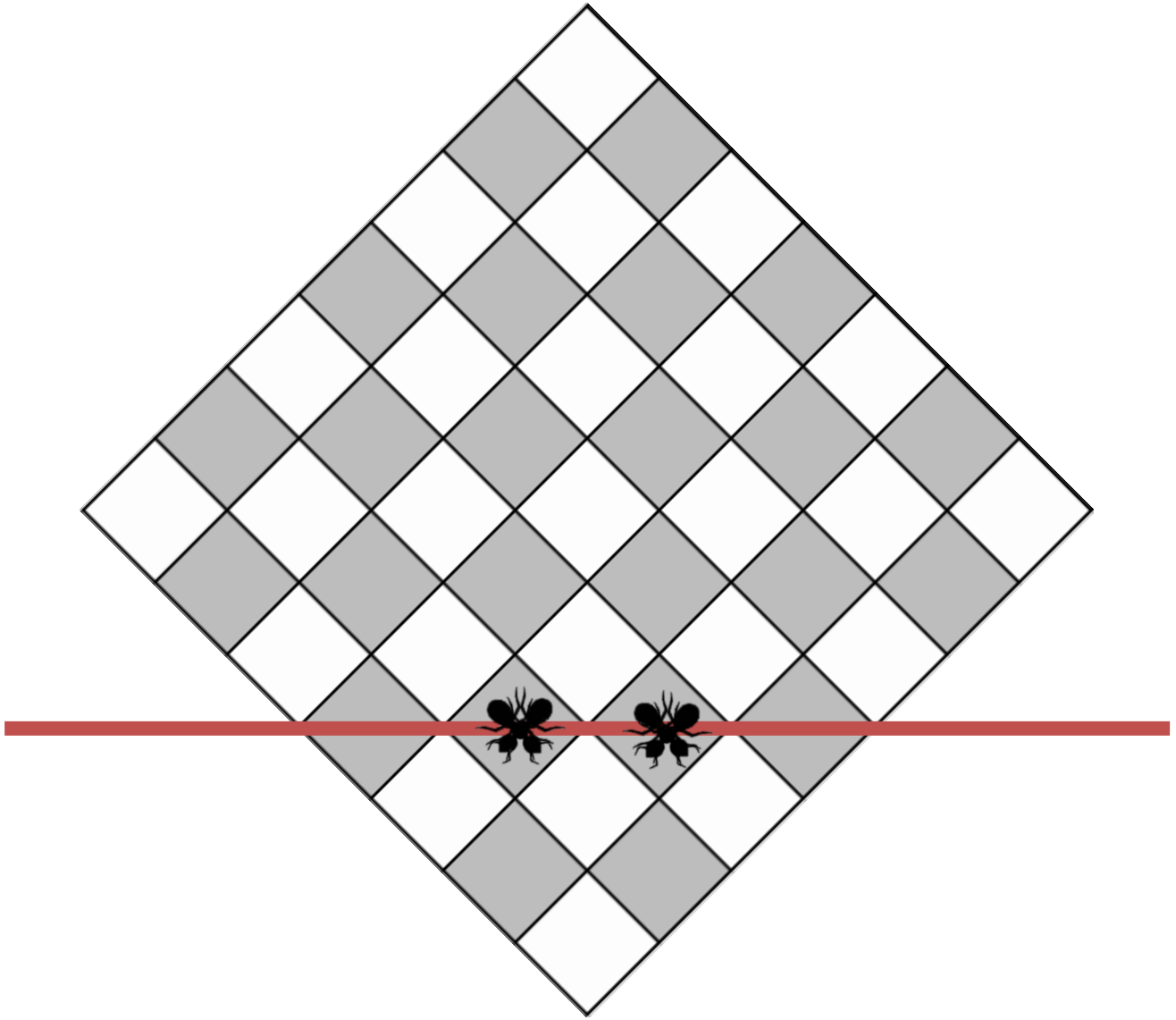
解法



問10　蟻

AIZU
THE UNIVERSITY OF AIZU

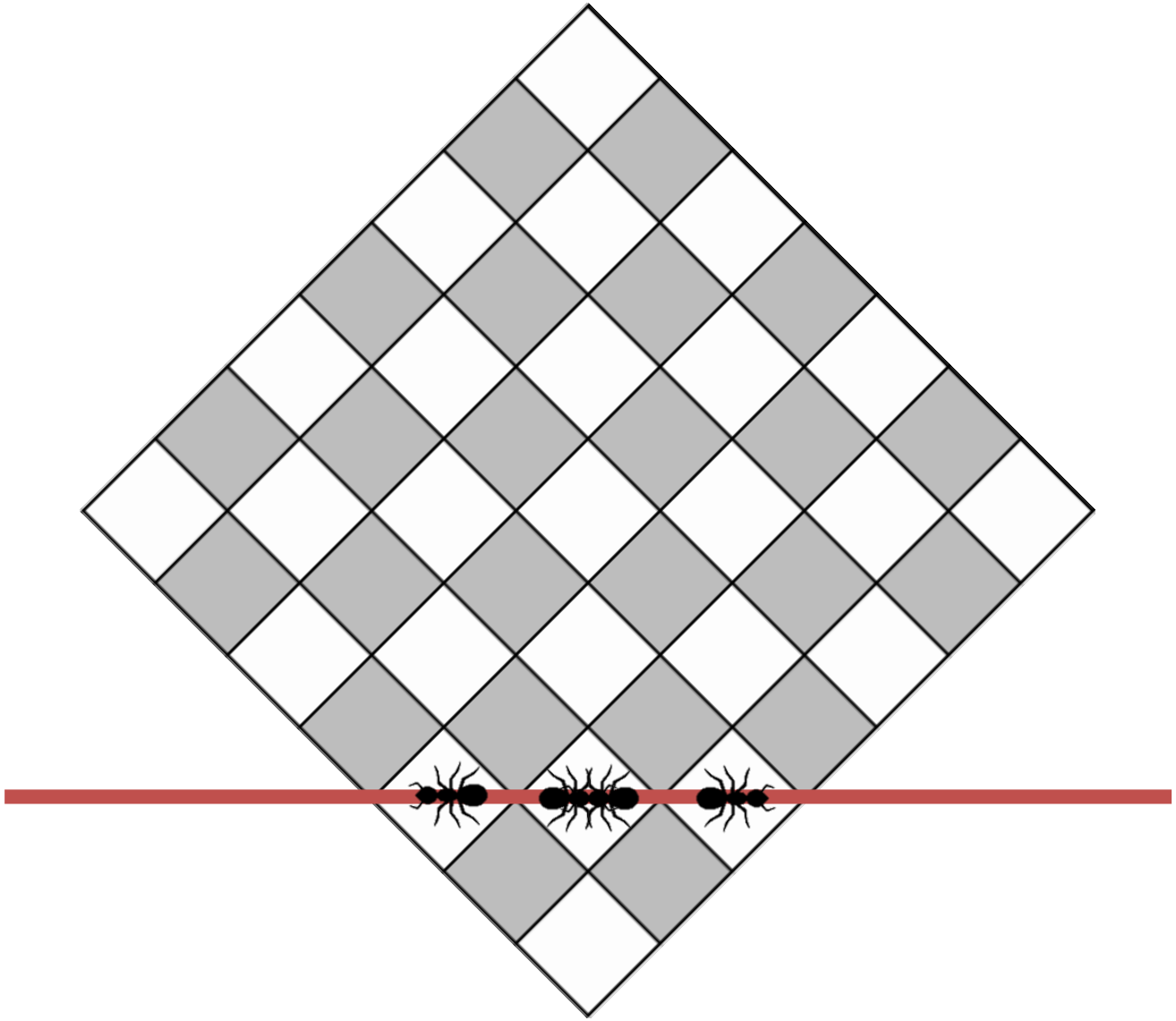
解法



問10　蟻

AIZU
THE UNIVERSITY OF AIZU

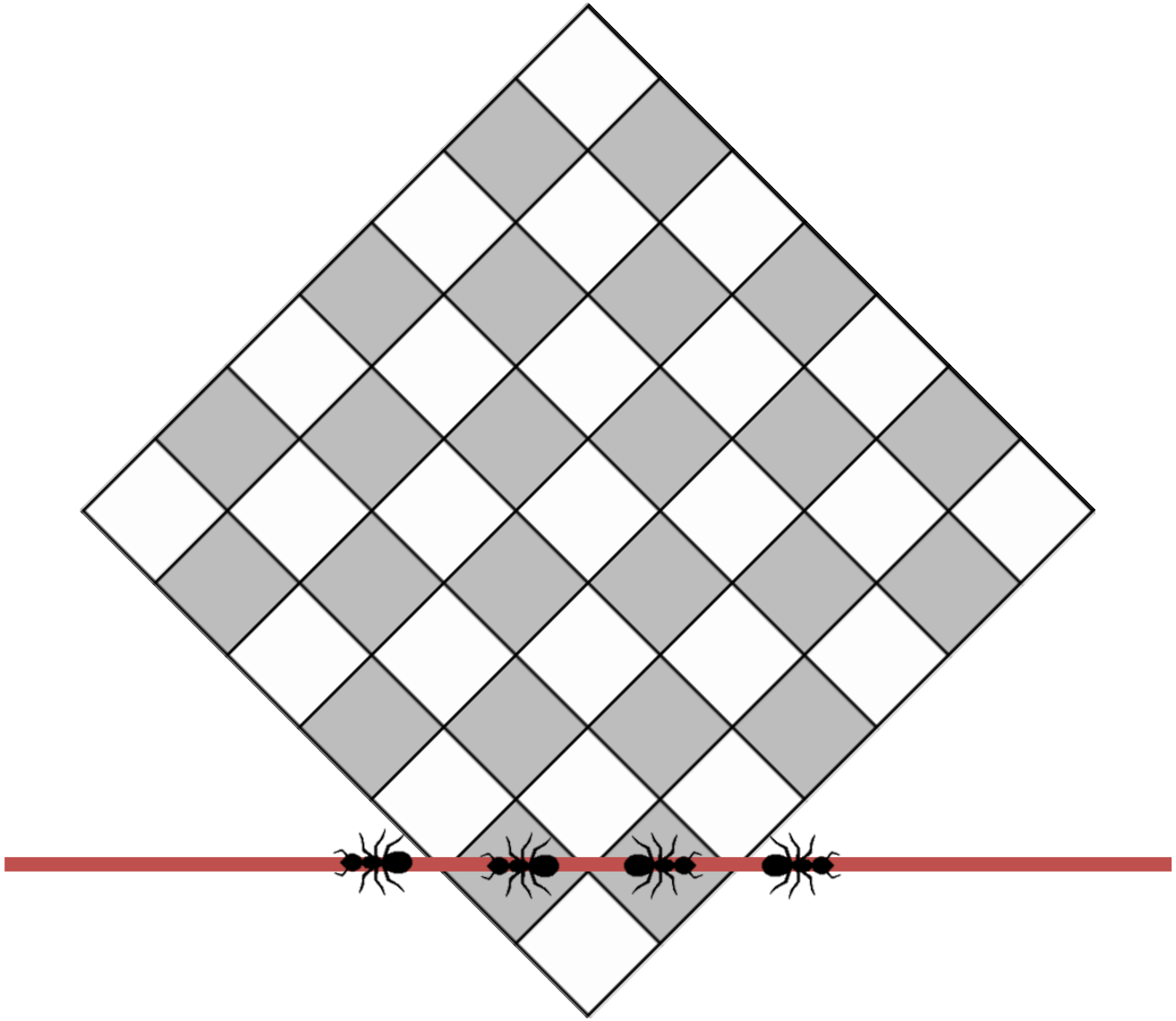
解法



問10　蟻

AIZU
THE UNIVERSITY OF AIZU

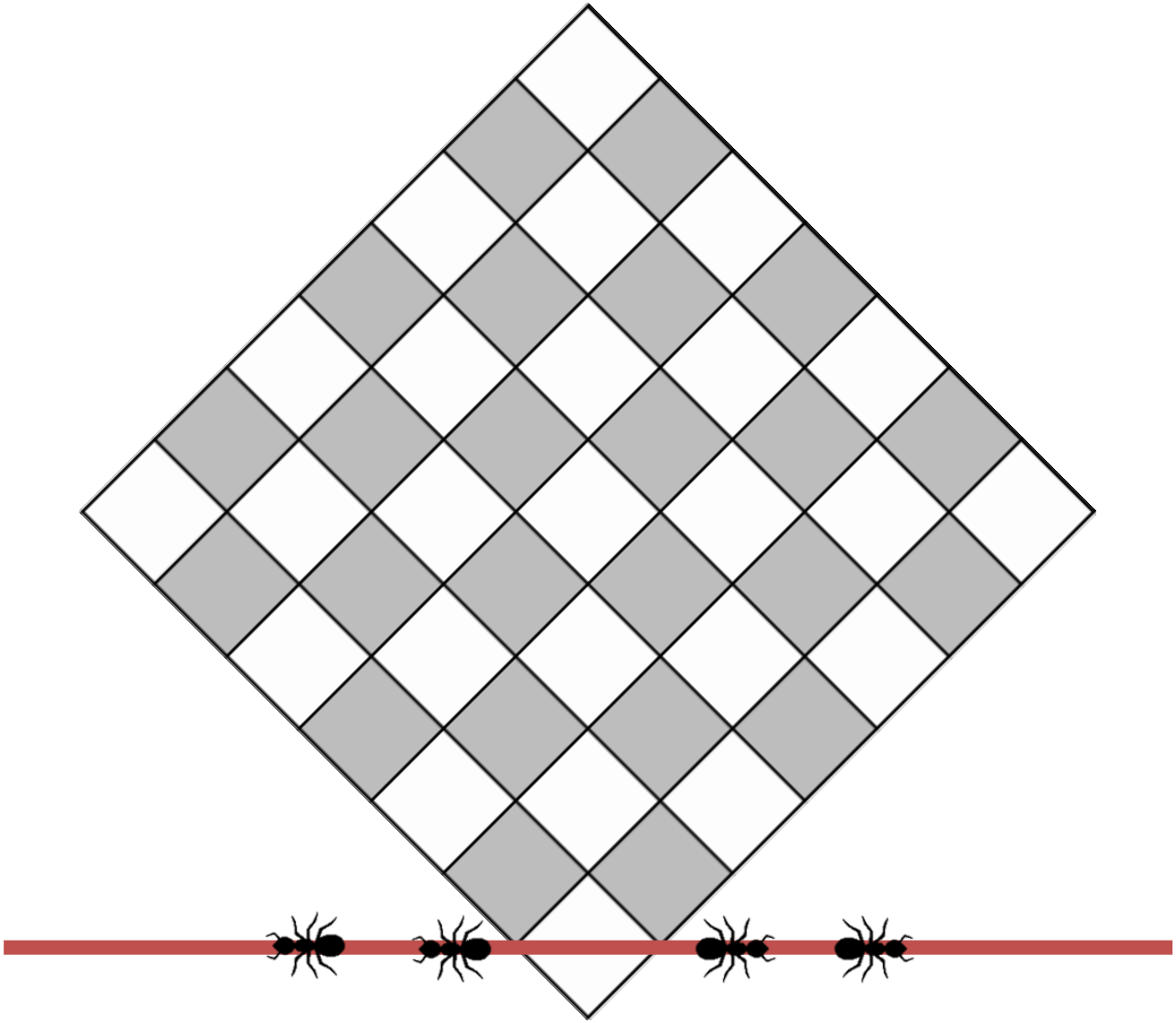
解法



問10　蟻

AIZU
THE UNIVERSITY OF AIZU

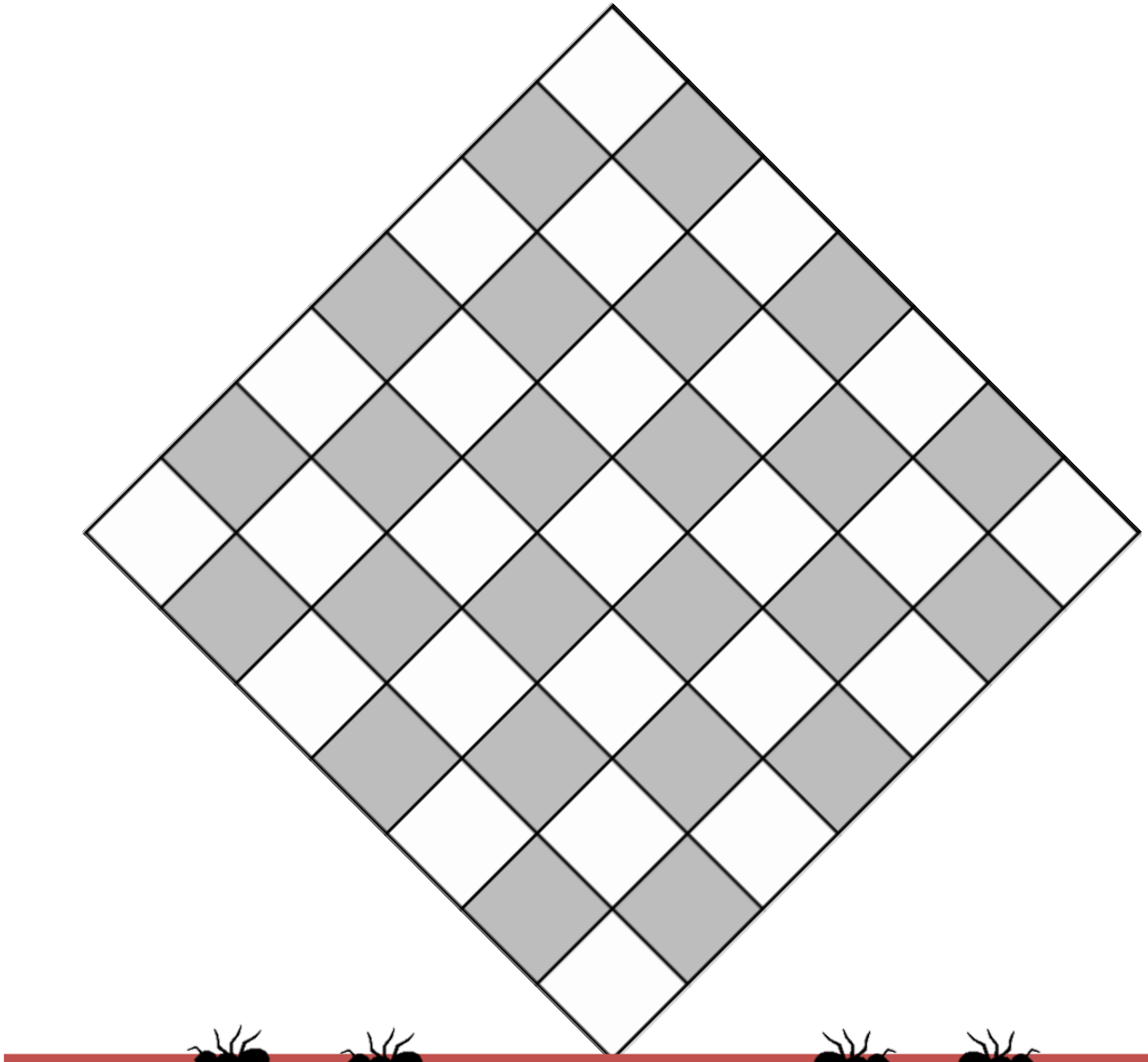
解法



問10　蟻

AIZU
THE UNIVERSITY OF AIZU

解法



問10　蟻

AIZU
THE UNIVERSITY OF AIZU

解法

- 各グループごとにゼッケンの入れ替わりをシミュレーションする.
- すべてのマスが白いマスならば,キューを用いることで高速に処理できる.

問10　蟻

AIZU
THE UNIVERSITY OF AIZU

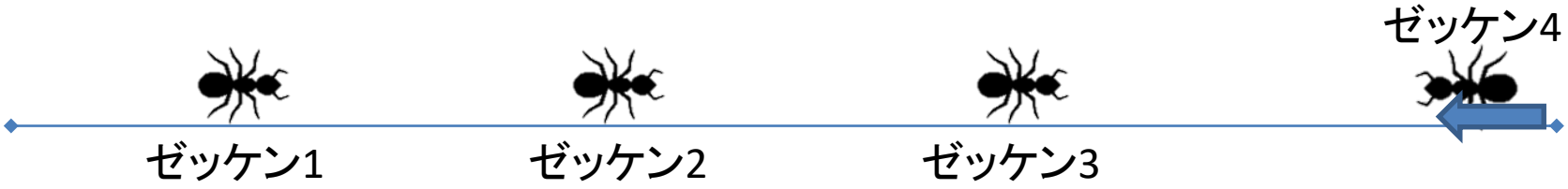
解法

- 左側にいる蟻から順番に蟻を1匹ずつ見ていく.現在見ている蟻の向きで以下の場合分けを行う。

- 蟻が右向きの場合
  - その蟻の情報をキューに追加.
- 蟻が左向きの場合
  - その左向きの蟻が最終的に付けているゼッケンは、キューの先頭にいる蟻がつけているゼッケンになる.
  - キューの末尾にいる蟻がつけているゼッケンは、その左向きの蟻がもともと付けていたゼッケンになる.
  - それ以外のキューの中にいる蟻のゼッケンは、キューの1つ後ろの蟻が付けていたゼッケンになる.

問10　蟻

解法



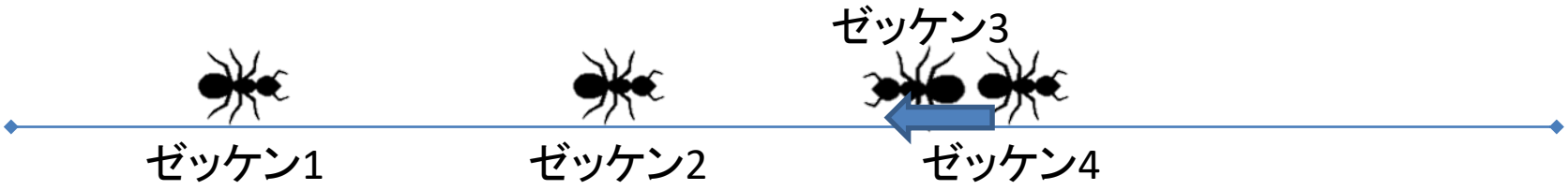
問10　蟻

解法

問10　蟻

AIZU
THE UNIVERSITY OF AIZU

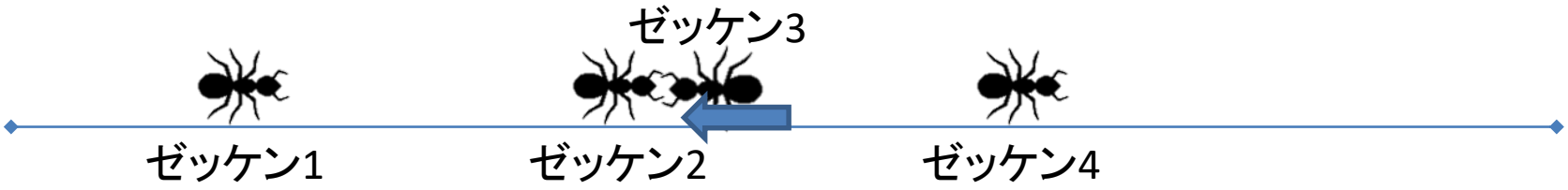
解法



問10　蟻

AIZU
THE UNIVERSITY OF AIZU

解法



問10　蟻

解法

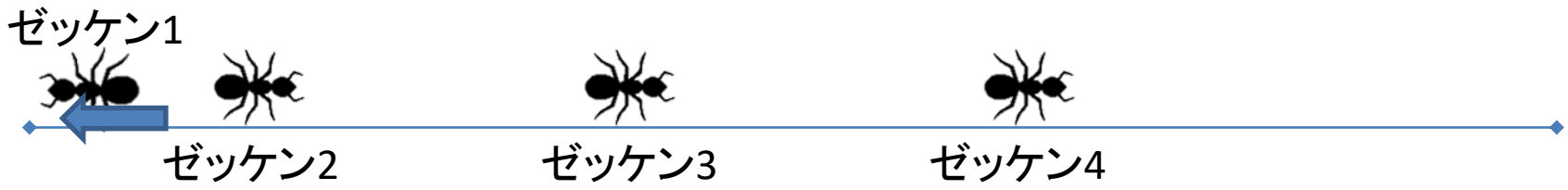
問10　蟻

解法

問10　蟻

AIZU
THE UNIVERSITY OF AIZU

解法



問10　蟻

AIZU
THE UNIVERSITY OF AIZU

解法

- 蟻がはじめにいたマスの色と、x座標の偶奇で場合分けが必要.
- 蟻がはじめにいたマスの色が白ならば単純に、蟻のデータを保存するキューを2倍用意して、 x座標の偶奇ごとに分けて管理すればよい.
  - x座標の偶奇が同じ蟻どうしでしか同時に白いマスに入りえないため.
- 蟻がはじめにいたマスの色が黒の場合でも、蟻のデータを保存するキューを2倍用意してx座標の偶奇ごとに分けて管理すればよいのだが,右向きの蟻だけ(もしくは左向きの蟻だけ)はx座標の偶奇を反転させる必要がある.
  - x座標の偶奇が異なる蟻どうしでしか同時に白いマスに入りえないため.

- 蟻のソートに最も時間がかかるので、O(N log N) となる.

問11　文字列ゲーム

AIZU
THE UNIVERSITY OF AIZU

問題概要

- １つの文字列が与えらる.
- その文字列に対して以下の種類のクエリを処理せよ.
  - 指定した範囲をある文字に置き換える.
  - 指定した範囲の部分文字列２つを比較する.
- 文字列の長さは最大 200,000.
- クエリの数は最大 100,000.

総評

- 正答数３.
- 文字列処理を高速に行う高度なデータ構造とアルゴリズムを実装する力が問われてる.

問11　文字列ゲーム

AIZU
THE UNIVERSITY OF AIZU

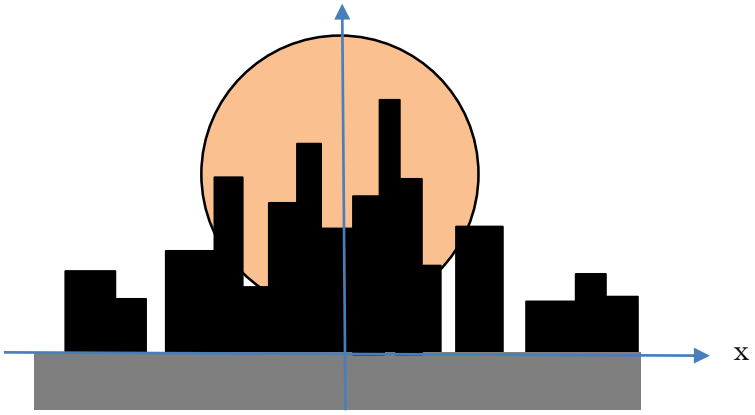
解法

- セグメントツリーで部分文字列のハッシュ値を管理する.
- 更新処理：
  - 範囲が指定されるため、O(log N)で高速に処理するためにはセグメントツリーの要素更新における遅延評価が必要.
- ２つの部分文字列の比較：
  - ２つの部分文字列に対して、各開始位置から何文字目までのハッシュ値が一致するかを、二分探索で求める.
  - 初めて異なった文字を比較して、２つの部分文字列の大小関係を求める.
  - Q個のクエリを処理、セグメントツリーで部分文字列のハッシュ値を求めながら二分探索を行うのでO(Q $\log^2$ N)のアルゴリズムとなる.

問12　夕暮れ

問題概要

- 地平線上(x軸上)にいくつかのビル(長方形)が並んでいる.
- 太陽(円)がｙ軸に沿って落下する.
- 太陽はビルによって遮られる.
- 円の見えている部分の面積（遮られている部分の面積）が半分になるときの円の高さを求めよ.



総評

- 正答数１.
- 計算幾何学の応用力と実装力が問われている.

問12 夕暮れ

AIZU
THE UNIVERSITY OF AIZU

解法

- 太陽の中心の高さを求める.
  - 太陽が沈むにつれて、ビル（地面）に遮られる面積は増加していく（非減少）.

  → 高さ h を二分探索することができる.
- 中心の高さが h の円と長方形の共通面積の和を求める.

  → 円と多角形の共通面積を求める.

  → 円と三角形の共通面積を求める.

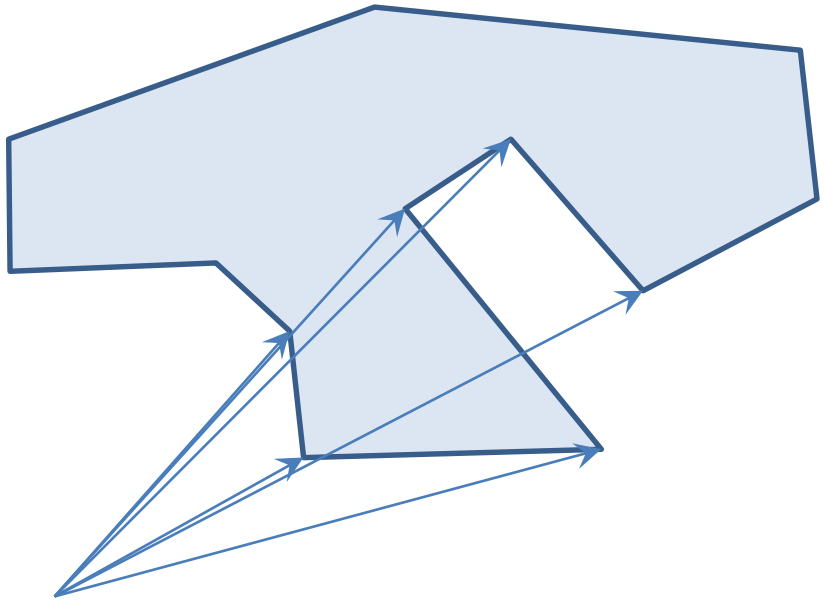
問12 夕暮れ

AIZU
THE UNIVERSITY OF AIZU

円と多角形の共通面積

- よく知られている多角形の面積の求め方を応用.
- 原点と多角形の点が作る符号付き三角形の面積の和.

```
double getArea(Polygon p){
    double sum = 0.0;
    for(int i = 0; i < p.size(); i++){
        sum += cross(p[i], p[(i+1)%p.size()]);
    }
    return abs(sum/2);
}
```

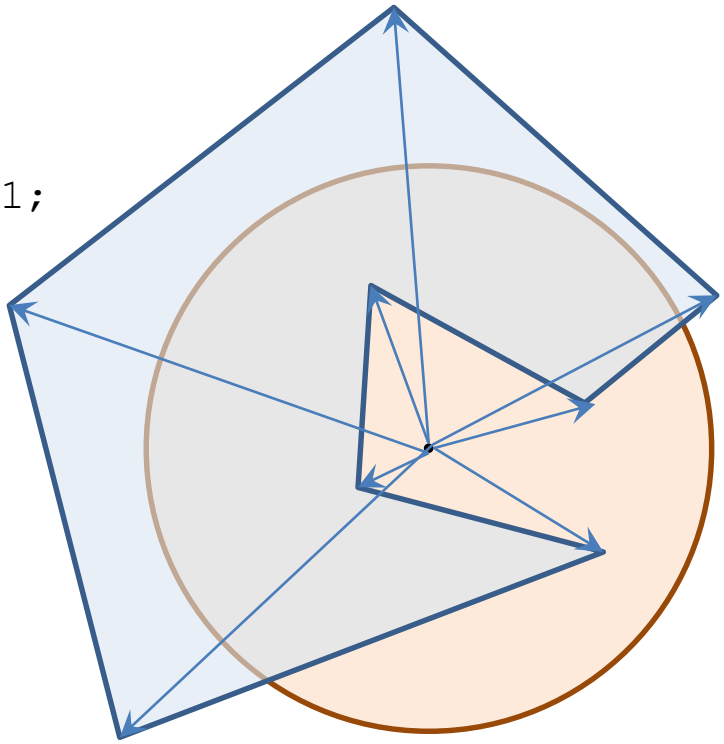


問12 夕暮れ

円と多角形の共通面積

- 円の中心を１点とする符号付き三角形の面積の和.

```
double getAreaCP(Circle c, Polygon g){
  double area = 0.0;
  for ( int i = 0; i < g.size(); i++ ){
    Point p1 = g[i];
    Point p2 = g[(i+1)%g.size()];
    double a = getAreaCT(c, p1, p2);
    if (cross(p1-c.c, p2-c.c) < 0) a *= -1;
    area += a;
  }

  return area;
}
```



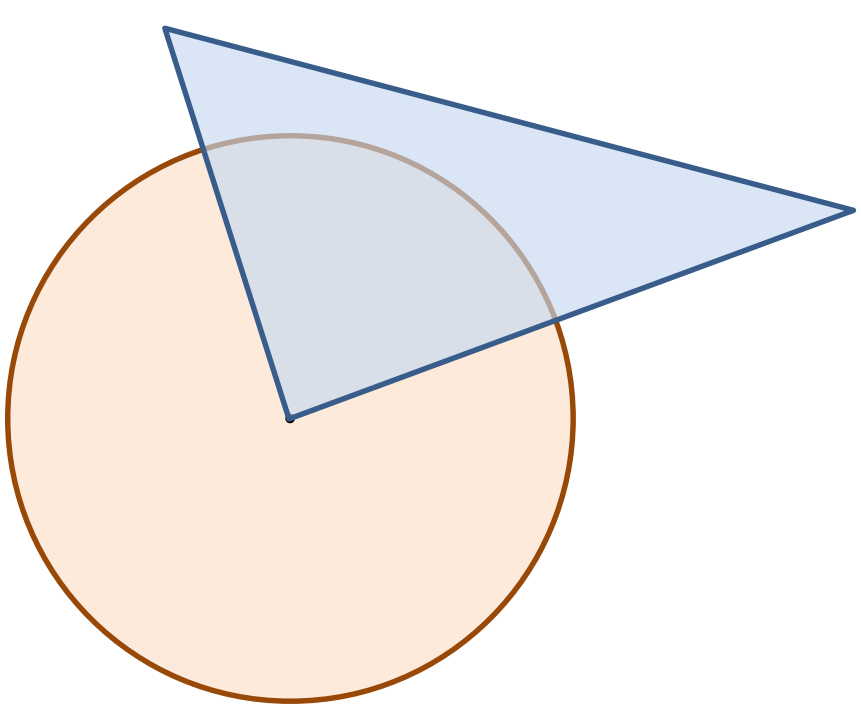
問12　夕暮れ

AIZU
THE UNIVERSITY OF AIZU

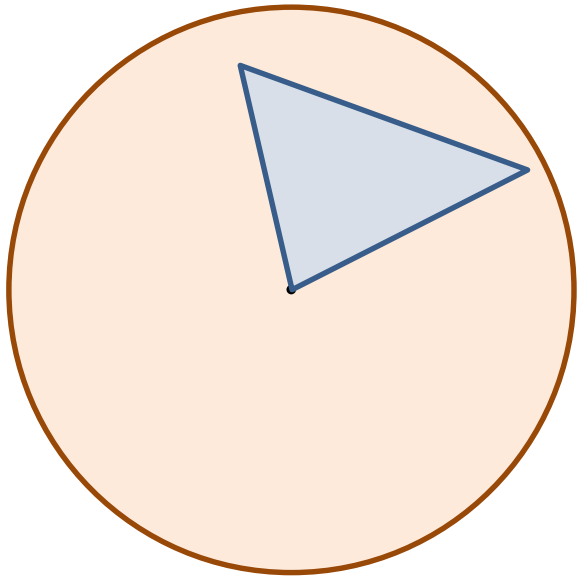
円と多角形の共通面積

- 円とその円の中心を頂点の一つとする三角形の共通面積.

```
double getAreaCT(Circle c, Point p1, Point p2){
```



場合分けを行い実装する.

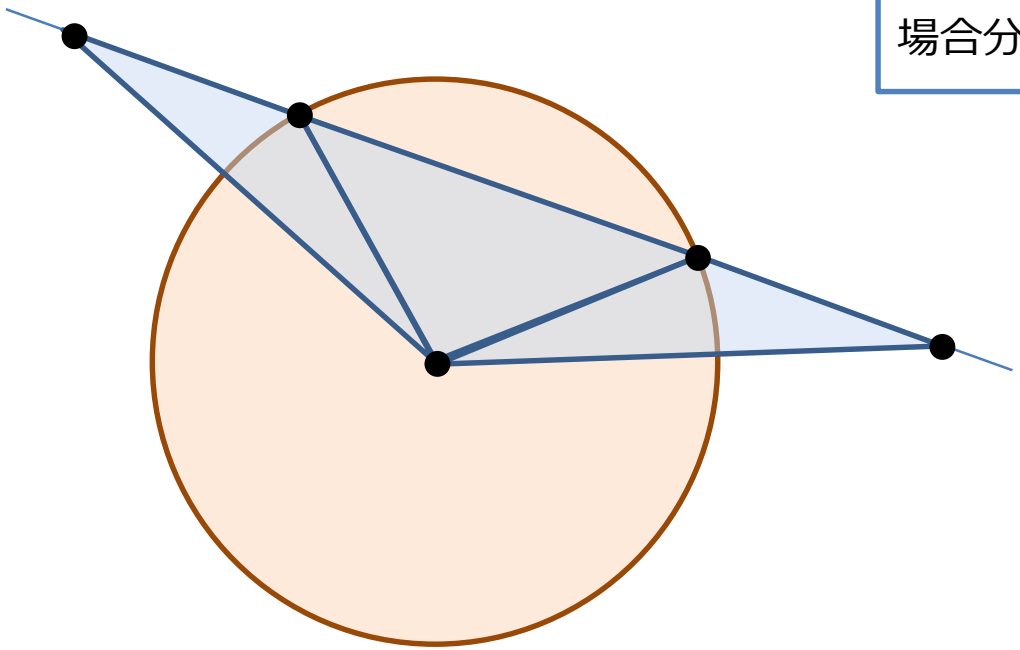


問12　夕暮れ

円と多角形の共通面積

- 円とその円の中心を頂点の一つとする三角形の共通面積.

```
double getAreaCT(Circle c, Point p1, Point p2){
```



場合分けを行い実装する.

問12 夕暮れ

AIZU
THE UNIVERSITY OF AIZU

解答例（C＋＋）

```
double getACT(Circle c, Point p1, Point p2){
  if ( contains(c, p1) != OUT && contains(c, p2) != OUT ){
    return abs(getArea(c.c, p1, p2)); // 三角形
  } else {
    return abs(c.r*c.r*getAngle(p1, c.c, p2)/2.0); // 扇形
  }
}

double getAreaCT(Circle c, Point p1, Point p2){
  if ( p1 == p2 || c.c == p1 || c.c == p2 || equals(0.0, c.r) ) return 0.0;
  if ( !isIntersect(c, Line(p1, p2) ) ) return getACT(c, p1, p2);
  if ( contains(c, p1) > 0 && contains(c, p2) > 0 ) return getACT(c, p1, p2);

  pair<Point, Point> pp = getCrossPoints(c, Line(p1, p2));

  if ( p1 < pp.second == p2 < pp.second && p1 < pp.first == p2 < pp.first ){
    return getACT(c, p1, p2);
  } else if ( contains(c, p1) > 0 ){
    return getACT(c, p1, pp.first) + getACT(c, pp.first, p2);
  } else if ( contains(c, p2) > 0 ){
    return getACT(c, p2, pp.second) + getACT(c, pp.second, p1);
  } else {
    return getACT(c, p1, pp.second) + getACT(c, pp.second, pp.first) + getACT(c, pp.first, p2);
  }
}
```

場合分けを行い実装する.