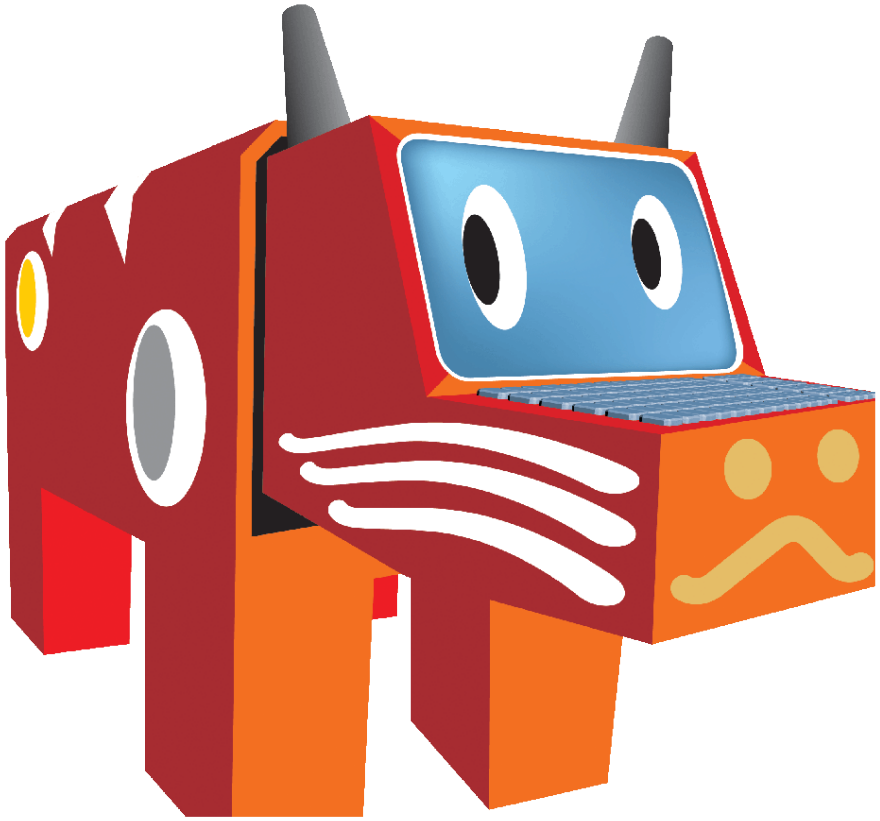
パソコン甲子園2014予選

プログラミング部門　解説



会津大学

問1　椅子の総数

問題概要

- ２つの整数 d (机一つあたりに必要な椅子の数)
  と c (机の数)が与えられる.
- 必要な椅子の総数 d×c を計算する.

講評・解法

- 提出数 907、正答数 564.
- 変数宣言、積の計算、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.

問1　Cによる解答例

```
#include <stdio.h>

int main(){
    int nDesk, nChair;

    scanf("%d %d", &nDesk, &nChair);
    printf("%d¥n", nDesk*nChair);

    return 0;
}
```

問1　C++による解答例

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

int main(){
    int nDesk, nChair;

    cin >> nDesk >> nChair;
    cout << nDesk*nChair << endl;

    return 0;
}
```

問1　Javaによる解答例

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

class Main{
    private void compute(){
        Scanner scanner = new Scanner(System.in);

        int nDesk  = scanner.nextInt();
        int nChair = scanner.nextInt();
        System.out.println(nDesk*nChair);
    }

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

問２　お財布メタボ診断

問題概要

- １円、５円、１０円、５０円、１００円、５００円、
それぞれの枚数が与えられる.
- 硬貨の合計が 1,000 円以上か判定する.

講評・解法

- 提出数 1,025、正答数 533.
- 入出力処理に加えて、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.
- 繰り返し処理を必要としない、最も簡単な部類の問題である.

問2　Cによる解答例: if-else

```
#include <stdio.h>

main(){
  int c1, c5, c10, c50, c100, c500;
  int sum = 0;

  scanf("%d %d %d %d %d %d",
         &c1, &c5, &c10, &c50, &c100, &c500);
  sum = c1 + 5*c5 + 10*c10 + 50*c50 + 100*c100 + 500*c500;

  if ( sum >= 1000 ) printf("1¥n");
  else               printf("0¥n");
}
```

問２　C++による解答例: 配列とループ

```
#include <iostream>
using namespace std;
int main() {
  int nSum = 0, n;
  int nVal[] = {1, 5, 10, 50, 100, 500};
  for (int i=0; i<6; ++i) {
    cin >> n;
    nSum += n*nVal[i];
    if (nSum >= 1000) {
      cout << 1 << endl;
      return 0;
    }
  }
  cout << 0 << endl;
  return 0;
}
```

問2　Javaによる解答例

```
import java.util.*;

class Main{
    int c1, c5, c10, c50, c100, c500, sum;
    void solve(){
       Scanner sc = new Scanner(System.in);
       c1   = sc.nextInt();
       c5   = sc.nextInt();
       c10  = sc.nextInt();
       c50  = sc.nextInt();
       c100 = sc.nextInt();
       c500 = sc.nextInt();
       int sum = c1 +c5*5 +c10*10 +c50*50 +c100*100 +c500*500;
       if ( sum >= 1000 ){ System.out.println(1); }
       else {              System.out.println(0); }
    }

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

問3　残り物には福がある

問題概要

- K 個の石から、P 人が順番に1つずつ石を取るゲームがある.
- P 人目が石を取った時点で、まだ石が残っていれば、また1人目から順番に1つずつ石を取っていく.
- 最後の石を取るのは何人目になるか求める.

講評・解法

- 提出数 1,467、正答数 460.
- 解法１：剰余で求める.
  - K%Pが解だが、割り切れる場合だけPを解とする.
  - または ((K-1)%P) + 1を計算しても良い.
- 解法２：シミュレーションで求める. Kは 1,000 以下なので間に合う.
  - K > 0の間、Kを減らしていき、カウンタjを増やす. jがPより大きくなったら1に戻す.終わったときのjの値が解.

問3　Cによる解答例(解法1)

```
#include <stdio.h>

int main() {
  int n, k, p, j;

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

    scanf("%d %d", &k, &p);

    j = k%p;
    printf("%d¥n", j == 0 ? p : j);
  }
  return 0;
}
```

問3　C++による解答例(解法2)

```
#include <cstdio>

int main() {
  int n, k, p;

  scanf("%d", &n);
  for (int i=0; i<n; ++i) {
    scanf("%d %d", &k, &p);

    int j = 1;
    while (--k) { //simulation
      ++j;
      if (j == p+1) j = 1;
    }
    printf("%d¥n", j);
  }
  return 0;
}
```

問3　Javaによる解答例(解法1)

```
import java.util.*;

class Main{

    void solve(){
        int N, K, P;
        Scanner sc = new Scanner(System.in);
        N = sc.nextInt();
        for ( int i = 0; i < N; i++ ){
            K = sc.nextInt();
            P = sc.nextInt();
            System.out.println((K-1)%P + 1);
        }
    }
    public static void main(String[] a){
        new Main().solve();
    }
}
```

問4　路線バスの時刻表

問題概要

- ２つのバス路線の時刻表を、１つの時刻表としてまとめる.
- 重複した時刻は1回だけ出力する.

講評・解法

- 提出数 1,555、正答数 156.
- ２つの数列を、重複を除いて昇順に結合する.
- 解法１:　全て読んだ後、ソートして重複除去.
- 解法２:　ある時刻にバスが有るか無いかを表すbool配列に、存在するバスを記録.

問4　C++による解答例 (解法1)

```
#include <cstdio>
#include <algorithm>

main() {
  int nBus = 0;
  int aBus[2*23*59];
  int n, h, m;
  for ( int k=0; k<2; ++k ) {
    scanf("%d", &n);
    for ( int i=0; i<n; ++i ) {
      scanf("%d %d", &h, &m);
      aBus[nBus+i] = h*100 + m;
    }
    nBus += n;
  }

  std::sort(aBus, aBus+nBus);
  nBus = std::unique(aBus, aBus+nBus) - aBus;

  for ( int i=0; i<nBus; ++i )
    printf("%d:%02d%c", aBus[i]/100, aBus[i]%100, (i+1) == nBus ? '¥n' : ' ');
}
```

問4　Cによる解答例 (解法2)

```
#include <stdio.h>
int timetbl[24][60];//外部変数は0に初期化される
int main(){
  int i, j, h, m, first;
  for (i=0; i<2; ++i) {
    scanf("%d", &j);
    for (; j>0; --j) {
      scanf("%d %d", &h, &m);
      timetbl[h][m] = 1;
    }
  }
  first = 1;
  for (i=0; i<24; ++i)
    for (j=0; j<60; ++j)
      if ( timetbl[i][j] ){
         if ( !first ) printf(" ");
         else first = 0;
         printf("%d:%02d", i, j);
      }
  printf("¥n");
  return 0;
}
```

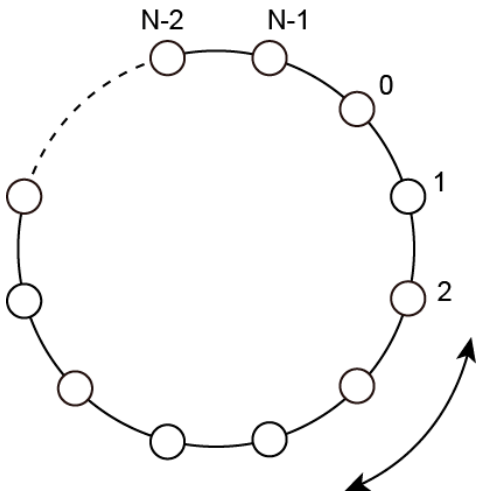
問4　Javaによる解答例 (解法2)

```
class Main {
    boolean exist[][] = new boolean[24][60];

    void solve(){
          for ( int i=0; i<24; ++i )
             for ( int j=0; j<60; ++j ) exist[i][j] = false;

          Scanner sc = new Scanner(System.in);
          for ( int i=0; i<2; ++i ) {
              int n = sc.nextInt();
              for ( int j = 0; j < n; ++j ){
                    int h = sc.nextInt();
                    int m = sc.nextInt();
                    exist[h][m] = true;
              }
          }
          boolean first = true;
          for ( int i=0; i<24; ++i ) {
              for ( int j=0; j<60; ++j ) {
                    if ( exist[i][j] ) {
                        if ( first ) {
                              System.out.printf("%d:%02d", i, j);
                              first = false;
                        }
                        else System.out.printf(" %d:%02d", i, j);
                    }
              }
          }
          System.out.println();
    }

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

問５　鉄道路線II

問題概要

- 右図のように0からN-1までの、N個の駅から成る環状の路線がある.
- １つ隣の駅には１００円で行ける.
- 現在居る駅pから、指定されたM個の駅を訪ねることができる、最小コストを求める.



講評

- 提出数 461、正答数 17.
- N≦100,000、M≦10,000.
- 明らかに全通り(M!)試すわけにはいかない.
- 最短となるには、どのような場合があるか考える.

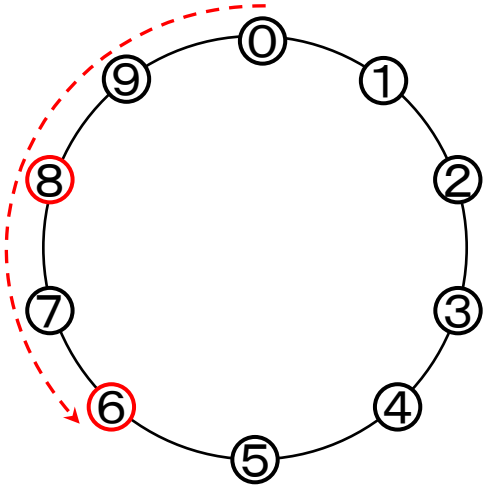
問5　鉄道路線II

解法

- 最短となる可能性がある経路を網羅する.

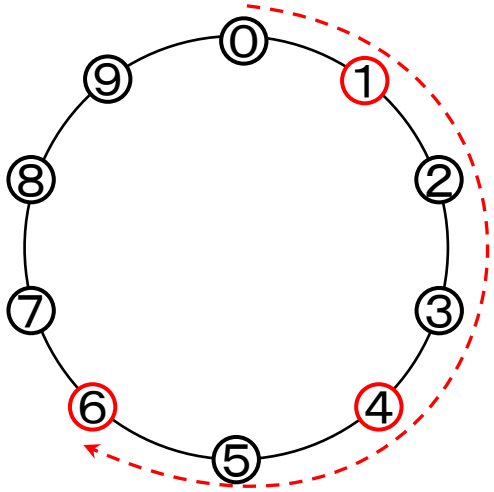
ケース1：　左回りで回りきる

例：$p=0,\ d_1=6,\ d_2=8$



ケース2：　右回りで回りきる

例：$p=0,\ d_1=1,\ d_2=4,\ d_3=6$



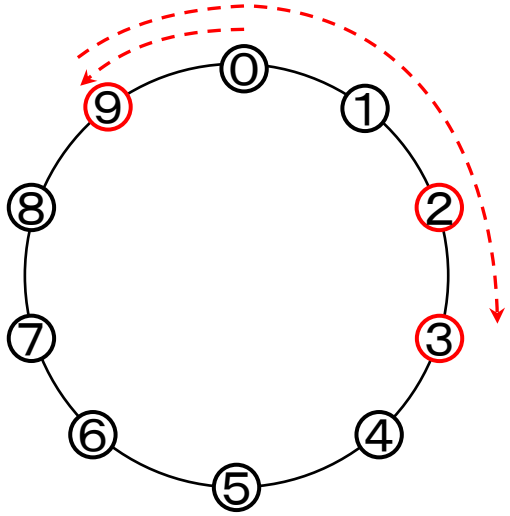
問５　鉄道路線II

解法

- 最短となる可能性がある経路を網羅する.

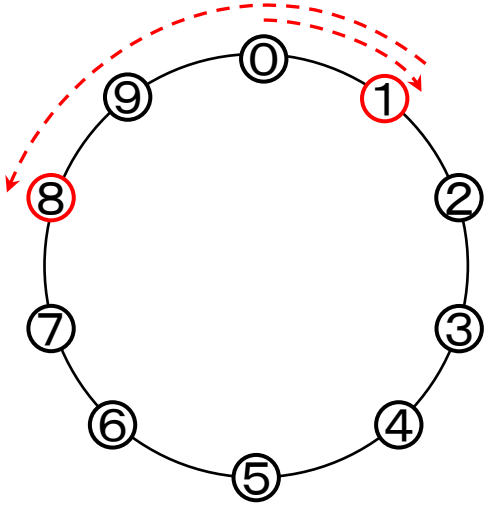
ケース３:
左回りで途中まで回った後、
右回りで回りきる

例 : $p=0,\ d_1=2,\ d_2=3,\ d_3=9$



ケース４:
右回りで途中まで回った後、
左回りで回りきる

例 : $p=0,\ d_1=1,\ d_2=8$



問５　鉄道路線II

解法

- ケース１、ケース２は１通りずつ（計２通り）.
- ケース３、ケース４は、訪ねたい駅Ｍ個全てで、切り返した場合をチェックする(計２Ｍ通り).
- Ｍ≦10,000 なので、多くとも 20,002 通り.
- それぞれの経路の長さは、引き算で求まる.
- ただし、０とN-1の間をまたぐときの引き算に注意する.
- 与えられるdは順番に並んでいない場合があるので、最初にソートしておく.

問5　C++による解答例:

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

int N, M, p;
vector<int> D;
int pdist(int s, int t){ return ( t >= s ) ? (t - s) : ((t+N) - s); }
int ndist(int s, int t){ return pdist(t, s);}

int solve(){
  if ( D.size() == 1 ) return min(pdist(D[0], p), ndist(D[0], p));
  int ans = 2*N;
  for ( int i = 0; i < M; i++ ){
    int b = D[i];
    int e = D[(i+1)%M];
    int d = min(pdist(b, p), min(pdist(e, p), min(ndist(b, p), ndist(e,p))));
    ans = min(d + (N-pdist(b, e)), ans);
  }
  return ans;
}
```

問5　C++による解答例（つづき）

```
int main(){
  int k;
  cin >> N >> M >> p;

  for ( int i = 0; i < M; i++ ){
    cin >> k;

    D.push_back(k);
  }
  sort(D.begin(), D.end());//与えられるdは順番に並んでいない場合があるのでsort
  cout << 100*solve() << endl;
  return 0;
}
```

問5　Javaによる解答例:

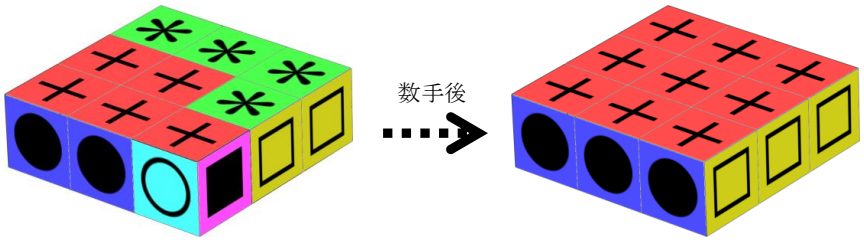
```
class Main {

    int n, m, p;
    int aDest[];
    boolean visit[];

    //sからeへの時計回り距離
    int CWDist ( int s, int e ) {
          return e - s < 0 ? e-s+n : e-s;
    }
    //sからeへの反時計回り距離
    int CCWDist ( int s, int e ) {
          return s - e < 0 ? s-e+n : s-e;
    }
    //訪ねたい駅のうち　時計回りでiの次の駅
    int CWNext ( int i ) {
          do {
              if ( ++i >= n ) i = 0;
          } while ( !visit[i] );
          return i;
    }

    //訪ねたい駅のうち　反時計回りでiの次の駅
    int CCWNext ( int i ) {
          do {
              if ( --i < 0 ) i = n-1;
          } while ( !visit[i] );
          return i;
    }

    //cw で pからtまで + ccw で tからt以前の経由駅eまで
    int CWTurn ( int p, int t, int e ) {
          return CWDist( p, t ) + CCWDist( t, e );
    }

    //ccw で pからtまで + cw で tからt以前の経由駅eまで
    int CCWTurn ( int p, int t, int e ) {
          return CCWDist( p, t ) + CWDist( t, e );
    }
```

問5　Javaによる解答例(つづき):

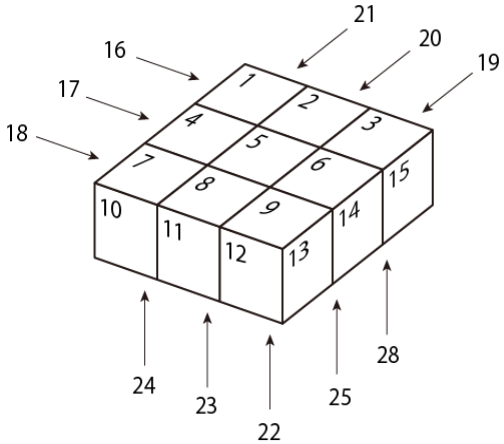
```
void solve(){
      Scanner sc = new Scanner(System.in);
      n = sc.nextInt();
      m = sc.nextInt();
      p = sc.nextInt();

      visit = new boolean[n];
      for ( int i=0; i<n; ++i )
          visit[i] = false;

      for ( int i=0; i<m; ++i ) {
          int d = sc.nextInt();
          visit[d] = true;
      }

      aDest = new int[m];
      for ( int i=0, k=0; i<n; ++i ) {
          if ( visit[i] ) aDest[k++] = i;
      }

      int nDist =
          Math.min( CCWDist(p, CWNext(p)),
                    CWDist(p, CCWNext(p)) );
      for ( int i=0; i<m; ++i ) {
          int e = (i == m-1) ? aDest[0] :
                               aDest[i+1];
          nDist =
             Math.min( nDist,
                       CWTurn(p, aDest[i], e) );

          e = (i == 0) ? aDest[m-1] :
                         aDest[i-1];
          nDist =
             Math.min( nDist,
                       CCWTurn(p, aDest[i], e) );
      }
      System.out.println( 100*nDist );
  }

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

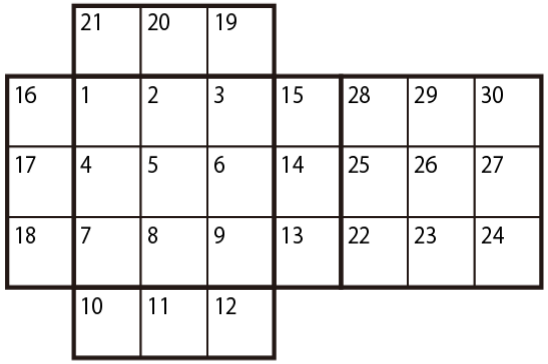
問6　フロッピーキューブ

問題概要



- 9つの立方体が平面的に並んだパズル（フロッピーキューブ）がある.
- 以下のような展開図で表した、現在の面の状態が与えられる.

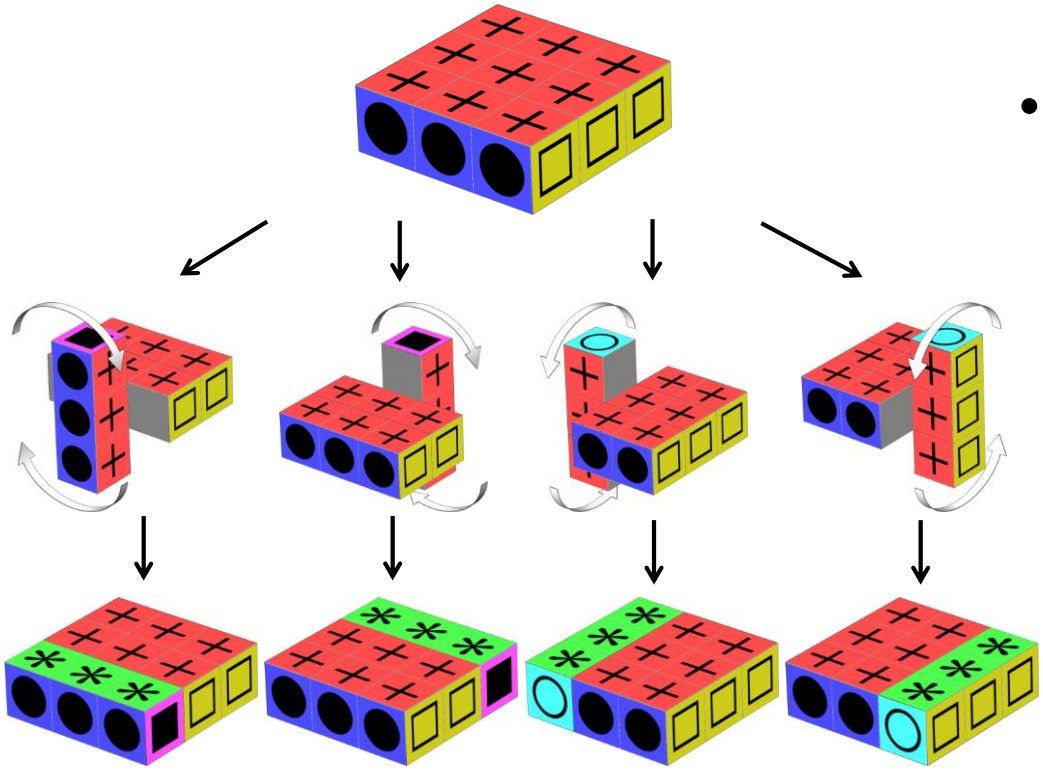




問6　フロッピーキューブ

問題概要

- フロッピーキューブに対しては、上、下、右、左の1列を180°回転させることができ、それぞれ1手と数える



- 全ての面をそろえるのにかかる最短の手数を求める.

問6　フロッピーキューブ

講評

- 提出数50、正答数 22.
- パズルの各列を回転させるには、どうすればよいか？
- 最短手数を求めるにはどうすればよいか？
- 幾何的な思考と、探索アルゴリズムの実装が問われている.

解法

- 回転は、値の入れ替え(swap)で実装する.
- 幅優先探索(BFS)で、全ての面がそろったら終了.
- 8手以内で終わることが保障されているので、枝狩りなどの工夫無しでも $4^8 = 65,536$ 以内の探索で終わる.

問6　C++による解答例

```
class Floppy{
public:
  int f[31], cost;
  Floppy(){ cost = 0; }
  bool operator < ( const Floppy &p ) const{
    for ( int i = 1; i <= 30; i++ ){
      if ( f[i] == p.f[i] ) continue;
      return f[i] < p.f[i];
    }
    return false;
  }

  bool solved(){
    return eq(1, 9) && eq(10, 12) && eq(13, 15) && eq(16, 18) &&
           eq(19, 21) && eq(22, 30);
  }

  bool eq(int b, int e){
    for ( int i = b; i < e; i++ ){
      if ( f[i] != f[i+1] ) return false;
    }
    return true;
  }

  void sw(int i, int j){ swap(f[i], f[j]); }
```

問6　C++による解答例（つづき）

```
  void roll1(){ sw(10, 12); sw(18, 13); sw(8, 23); sw(7, 22); sw(9, 24); }
  void roll2(){ sw(1, 28); sw(3, 30); sw(21, 19); sw(2, 29); sw(15, 16); }
  void roll3(){ sw(16, 18); sw(1, 24); sw(7, 30); sw(4, 27); sw(10, 21); }
  void roll4(){ sw(13, 15); sw(9, 28); sw(3, 22); sw(12, 19); sw(6, 25); }
};

int bfs(Floppy s){
  queue<Floppy> Q;
  set<Floppy> V;
  Q.push(s);
  V.insert(s);

  Floppy u, v;

  while( !Q.empty() ){
    u = Q.front(); Q.pop();
    if ( u.solved() ) return u.cost;
    v = u;
    v.roll1(); v.cost++;
    if ( V.find(v) == V.end() ) Q.push(v);

    v = u;
    v.roll2(); v.cost++;
    if ( V.find(v) == V.end() ) Q.push(v);

    v = u;
    v.roll3(); v.cost++;
    if ( V.find(v) == V.end() ) Q.push(v);
```

問6　C++による解答例（つづき）

```
    v = u;
    v.roll4(); v.cost++;
    if ( V.find(v) == V.end() ) Q.push(v);
  }

  return -1;
}

int main(){
  Floppy F;
  int n; cin >> n;
  for ( int t = 0; t < n; t++ ){
    for ( int i = 1; i <= 30; i++ ) cin >> F.f[i];
    int ans = bfs(F);
    cout << ans << endl;
  }

  return 0;
}
```

問6　Javaによる解答例:

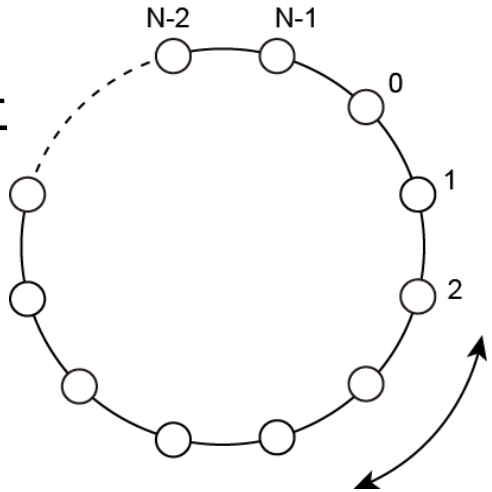
```
class FloppyCube {
    int nCnt = 0;
    int[] aFace = { 1, 1, 1, 1, 1, 1, 1, 1, 1,
          2, 2, 2, 4, 4, 4, 6, 6, 6, 5, 5, 5,
          3, 3, 3, 3, 3, 3, 3, 3, 3 };

    public boolean equals( FloppyCube other ) {
          for ( int i=0; i<30; ++i ) {
              if ( aFace[i] != other.aFace[i] )
                   return false;
          }
          return true;
    }

    void Repl( int m, int n ) {
          int t = aFace[m];
          aFace[m] = aFace[n];
          aFace[n] = t;
    }

    public FloppyCube clone() {
          FloppyCube copy = new FloppyCube();
          copy.nCnt = nCnt;
          for ( int i=0; i<30; ++i )
             copy.aFace[i] = aFace[i];
          return copy;
    }

    public void Rotate( int pos ) {
          switch ( pos ) {
          case 0 :
              Repl(12, 17); Repl(9, 11);
               Repl(6, 21); Repl(7, 22);
               Repl(8, 23);
              break;
          case 1 :
              Repl(14, 15); Repl(18, 20);
               Repl(0, 27); Repl(1, 28);
               Repl(2, 29);
              break;
          case 2 :
              Repl(11, 18); Repl(12, 14);
               Repl(2, 21); Repl(5, 24);
               Repl(8, 27);
              break;
          case 3:
              Repl(9, 20); Repl(15, 17);
               Repl(0, 23); Repl(3, 26);
               Repl(6, 29);
              break;
          }
    }
}
```

問6　Javaによる解答例(つづき):

```
class Main {

    final FloppyCube solution = new FloppyCube();

    void solve(){

          Scanner sc = new Scanner(System.in);
          int n = sc.nextInt();

          while ( n-- > 0 ) {

              FloppyCube current = new FloppyCube();
              for ( int i=0; i<30; ++i )
                    current.aFace[i] = sc.nextInt();

              LinkedList<FloppyCube> check =
                         new LinkedList<FloppyCube>();
              check.offer( current );

              while ( check.peek() != null ) {

                    int i = 0;
                    for ( ; i<4; ++i ) {
                        current =check.peek().clone();
                        if ( current.equals(solution) )
                             break;

                        current.Rotate(i);
                        ++current.nCnt;
                        check.offer( current );
                    }
                    if ( i != 4 ) break;
                    check.poll();
              }

              System.out.println( current.nCnt );
              }

          }

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

問7　バトンリレーゲーム

問題概要



- 0からN-1のゼッケンを付けたN人の生徒が、図のように輪になって並んでいる.
- 最初は生徒0がバトンを持っている.
- 現時点でバトンを持っている生徒が適当な正の整数aを宣言する.

- aが偶数のときは時計回り、奇数のときは反時計回りに隣の生徒にバトンを渡していき、a番目にバトンを受け取った生徒が脱落する.
- 終わった後、Q人の生徒について輪に残っているか調べる.
- 残っていれば1、いなければ0を出力する.
- N ≦ 200,000、M < 200,000、a ≦ 100、Q ≦ 1,000.

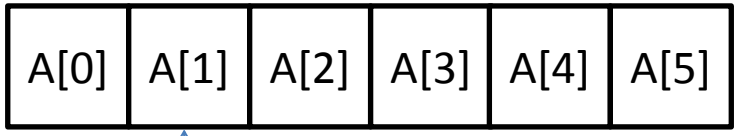
問7　バトンリレーゲーム

講評

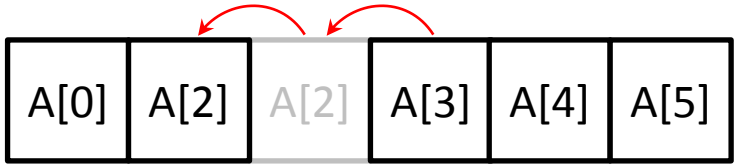
- 提出数 149、正答数 7.
- vectorなどの配列に対して要素の削除を行うと、そのたびに線形時間かかるため、O(N*M)程度の計算量で時間オーバー.
- 脱落する生徒を効率よく扱うためのデータ構造が必要.
- データ構造の特徴を知り、必要ならば独自に実装する、あるいは標準ライブラリのデータ構造を有効利用できるかが問われている.

＊配列の要素を削除するしくみ

配列は、要素のアドレスが連続である必要がある.



そのため、例えば
ここを削除すると



それ以降の要素を一つずつ前に詰めていくため、平均n/2の計算量になる.

問7　バトンリレーゲーム

解法

- 双方向連結リストを作る
- バトンは左にも右にも回るため、双方向にする必要がある.
- リストはランダムアクセスができないため、バトン回しを一つずつ行う必要があるが、バトン回しのパス数aは最大100程度でMが20万程度なので、2,000万(オーダー10^8)程度で間に合う.
- 標準ライブラリのlistや、要素の前(prev)や後(next)関係を記録した2つの配列などで実装可能.

問7　C++による解答例

```
#define MAX 200000
int N, M, Q;

struct Node{
  int prev, next;
  bool f;
};

Node S[MAX];
int A[MAX];
```

```
void simulate(){
  int cur = 0;
  int a;
  bool d;
  for ( int i = 0; i < M; i++ ){
    a = A[i];
    if ( a%2 == 0 ){
      for ( int c = 0; c < a; c++ ){
          cur = S[cur].next;
      }
    } else {
      for ( int c = 0; c < a; c++ ){
          cur = S[cur].prev;
      }
    }
    S[cur].f = false;
    S[S[cur].prev].next = S[cur].next;
    S[S[cur].next].prev = S[cur].prev;
    cur = S[cur].next;
  }
}
```

問7　C++による解答例（つづき）

```
int main(){
  cin >> N >> M >> Q;

  for ( int i = 0; i < N; i++ ) {
    S[i].f = true;
    S[i].next = (i+1)%N;
    S[i].prev = (i+(N-1))%N;
  }
  for ( int i = 0; i < M; i++ ) {
    scanf("%d", &A[i]);
  }

  simulate();

  for ( int i = 0; i < Q; i++ ){
    int q;
    cin >> q;
    printf("%d¥n", S[q].f);
  }

  return 0;
}
```

問7　C++ の list による解答例

```
#define MAX 1000000
int N, M, Q;
int A[MAX];
int V[MAX];

void simulate(){
  list<int> l;
  for ( int i = 0; i < N; i++ ) l.push_back(i);
  list<int>::iterator it = l.begin();

  for ( int i = 0; i < M; i++ ){
    int a = A[i];

    if ( a % 2 == 0 ) {
      for ( int c = 0; c < a; c++ ){
          it++;
          if ( it == l.end() ) it = l.begin();
      }
    } else {
      for ( int c = 0; c < a; c++ ){
          if ( it == l.begin() ) {
            it = l.end();
            it--;
          } else {
            it--;
          }
      }
    }
```

問7　C++ の list による解答例（つづき）

```
    V[*it] = 0;
    it = l.erase(it);
    if ( it == l.end() ) it = l.begin();
  }
}

main(){
  cin >> N >> M >> Q;
  for ( int i = 0; i < M; i++ ) scanf("%d", &A[i]);
  for ( int i = 0; i < N; i++ ) V[i] = 1;
  simulate();
  for ( int i = 0; i < Q; i++ ){
    int q; cin >> q;
    printf("%d¥n", V[q]);
  }
}
```

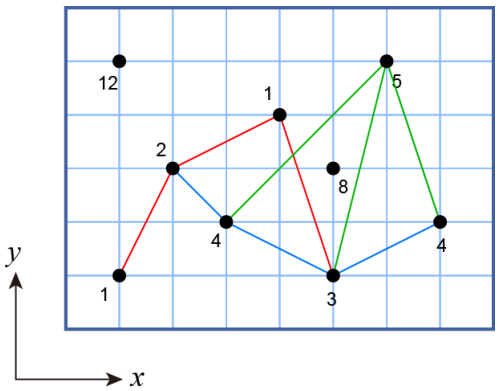
問8　天体観測

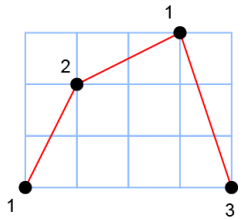
問題概要

- 様々な明るさを持つ星々が窓枠から見える.
- 明るさの差が定数 d 以下の星のグループで星座を作る.
- ある星座の星をすべて含むような、窓枠に平行な辺からなる長方形のうち、面積が最も小さいものを、その星座の大きさとする.
- 窓枠から見える一番大きい星座の面積を求めよ.
- 星の数は 200,000 以下.
- 星の座標は整数で $0 \leqq x_i, y_i \leqq 2{,}000{,}000$.
- $0 \leqq$ 星の明るさ $\leqq 10^9$.

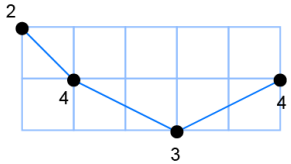
問8　天体観測

問題概要

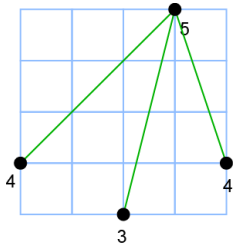




大きさが12の星座



大きさが10の星座



大きさが16の星座
（最大の星座）

講評

- 提出数 35、正答数 8.
- 最大面積を効率的に計算する高等的なデータ構造が必要.

問8　天体観測

解法

- 星の配列 S を明るさの昇順に整列する（O(N log N)）.
- 明るさの差が d 以下の区間に含まれる星について
  - x の最小値 minx
  - x の最大値 maxx
  - y の最小値 miny
  - y の最大値 maxy

  を更新していき、面積 (maxx – minx) × (maxy – miny) の最大値を求める.
- 区間は「しゃくとり法」により O(N) で走査することができる.
- minx, maxx, miny, maxy は、それぞれの「セグメント木（Range Minimum Query）」を用いて O(log N) で更新することができる.
- よって、O(N log N) のアルゴリズムで解くことができる.

問8　C++による解答例

```
class RMQ{
public:
  int n;
  ll D[2*MAX_N-1];
  RMQ(){}
  RMQ(int n_){
    n = 1;
    while( n < n_ ) n *= 2;
    for ( int i = 0; i < 2*n-1; i++ ) D[i] = INFTY;
  }

  void update(int k, ll a){
    k += n - 1;
    D[k] = a;
    while( k > 0 ){
      k = (k - 1)/2;
      D[k] = min(D[k*2+1], D[k*2+2]);
    }
  }

  // [a, b]
  ll findMin(int a, int b){
    return query(a, b+1, 0, 0, n);
  }

  // [a, b)
  ll query(int a, int b, int k, int l, int r){
    if ( r <= a || b <= l ) return INFTY;
    if ( a <= l && r <= b ) return D[k];
    else{
      ll vl = query(a, b, k*2 + 1, l, (l + r)/2);
      ll vr = query(a, b, k*2 + 2, (l + r)/2, r);
      return min(vl, vr);
    }
  }
};
```

```
class Star{
public:
  ll x, y, l;
  Star(){}
  Star(ll x, ll y, ll l):x(x), y(y), l(l){}
  bool operator < ( const Star &s) const{
    return l < s.l;
  }
};

Star S[MAX_N];
RMQ minXQ = RMQ(MAX_N);
RMQ minYQ = RMQ(MAX_N);
RMQ maxXQ = RMQ(MAX_N);
RMQ maxYQ = RMQ(MAX_N);
```

問8　C++による解答例（つづき）

```
int main(){
  int N;
  ll D, x, y, l;

  cin >> N >> D;

  for ( int i = 0; i < N; i++ ){
    scanf("%lld %lld %lld", &x, &y, &l);
    S[i] = Star(x, y, l);
  }

  sort( S, S+N);

  for ( int i = 0; i < N; i++ ){
    minXQ.update(i, S[i].x);
    minYQ.update(i, S[i].y);
    maxXQ.update(i, S[i].x*(-1));
    maxYQ.update(i, S[i].y*(-1));
  }

  int t = 0;
  ll maxarea = -1;

  for ( int s = 0; s < N; s++ ){
    while( t < N && S[t].l - S[s].l <= D) t++;
    t--;

    ll a = ((-1)*maxXQ.findMin(s, t)-minXQ.findMin(s, t))*((-1)*maxYQ.findMin(s, t) - minYQ.findMin(s, t));
    maxarea = max(maxarea, a);
  }

  cout << maxarea << endl;

  return 0;
}
```

問9　力持ち

問題概要

- N人の力持ち 1, 2, …, N が横一列に並んで行進する.
- 力持ち i の体重は w_i で最大 c_i の重量まで持つことができる.
- 他人を持ち上げることで実際に歩く人数を減らしたい.
- 以下の条件をすべて満たすとき、隣に立っている左右どちらかの力持ちを持ちあげることができる.
  - 自分の上下には力持ちはいない. つまり、誰かに持ち上げられてもいないし、誰かを持ちあげてもいない.
  - 隣の力持ちの体重が、自分が持つことのできる最大の重量以下である. ただし、隣の力持ちが既に誰かを持ち上げているなら、縦に積み上がった力持ちたちの体重の合計が、自分が持つことのできる最大の重量以下でなければならない.
- 一番下で歩く力持ちの最小の人数を求めよ.

問9　力持ち

問題概要

- $1 \leqq N \leqq 1{,}000.$
- $1 \leqq c_i \leqq 100{,}000.$
- $1 \leqq w_i \leqq 100{,}000.$

講評

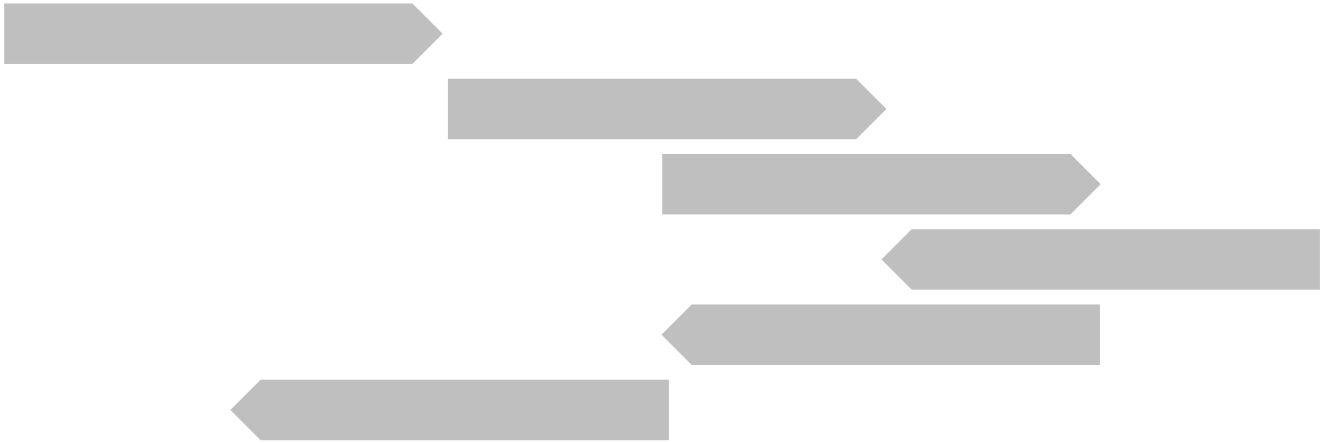
- 提出数 15、正答数 5.
- 組み合わせの最適解を効率よく求めるプログラミングテクニックが必要.

問9　力持ち

解法

- 力持ち i から力持ち j までをグループにできるかを配列 P[i][j] に記録する.

	1	2	3	4	5	6
持てる重さ	10	2	15	4	4	8
重さ	5	10	20	2	3	5



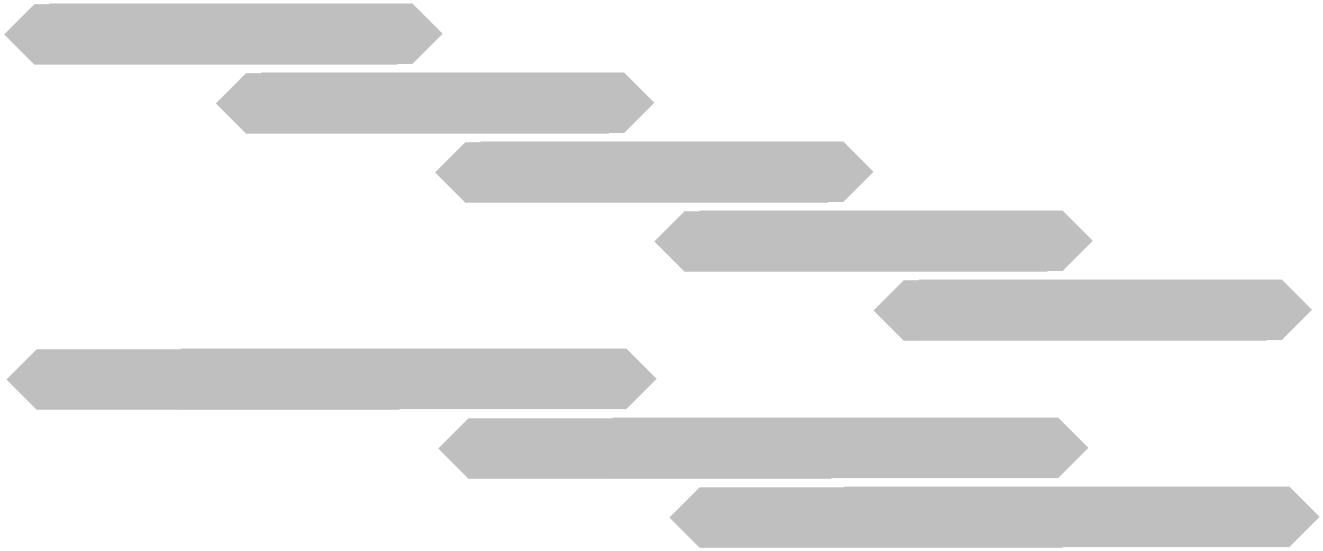
隣あっている人を持てるかどうかは分かる

問9　力持ち

解法

- 力持ち i から力持ち j までをグループにできるかを配列 P[i][j] に記録する.

	1	2	3	4	5	6
持てる重さ	10	2	15	4	4	8
重さ	5	10	20	2	3	5



２つ隣の人まで持てるかどうか分かると、３つ隣の人で持てるかどうか分かる

問9　力持ち

解法（つづき）

- M 個隣の人まで持てるかどうか分かると、M+1 個隣の人で持てるかどうかが分かる.
- 配列 P は $O(N^2)$ で作成することができる.
- これにより、力持ち i から力持ち j までの区間をグループにできるかどうかを O(1) で判定することができる.

- 次に、P を利用して、「i 番目の力持ちまでを考慮した、歩く人の人数の最小値 dp[i]」を動的計画法によって求める.
- すべての区間 P[b][e] を考慮して dp[e] を更新していく $O(N^2)$ のアルゴリズムとなる.
  - P[b][e] が true なら、dp[e] = min(dp[e], dp[b-1]+1).

問9　C++による解答例

```
struct P{ int c, w; };

int N;
P S[MAX+1];
bool P[MAX+1][MAX+1];
int dp[MAX+1];
int L[MAX+1];

int main(){
  cin >> N;
  for ( int i = 1; i <= N; i++ ) cin >> S[i].c >> S[i].w;
  for ( int i = 1; i <= N; i++ ){
    for ( int j = 1; j <= N; j++ ){
      P[i][j] = (i==j)?true:false;
    }
  }
  L[0] = 0;
  for ( int i = 1; i <= N; i++ ) L[i] = S[i].w + L[i-1];

  for ( int l = 1; l <= N; l++ ){
    for ( int i = 1; i <= N-l+1; i++ ){
      int j = i + l - 1;
      if ( !P[i][j] ) continue;
      if ( j + 1 <= N ){
        if ( L[j] - L[i-1] <= S[j+1].c ){
          P[i][j+1] = true;
        }
      }
      if ( i - 1 >= 1 ){
        if ( L[j] - L[i-1]  <= S[i-1].c ){
          P[i-1][j] = true;
        }
      }
    }
  }
  for ( int i = 0; i <= N; i++ ) dp[i] = INF;
  dp[0] = 0;

  for ( int b = 1; b <= N; b++ ){
    for ( int e = 1; e <= N; e++ ){
      if ( !P[b][e] ) continue;
      dp[e] = min(dp[e], dp[b-1]+1);
    }
  }

  cout << dp[N] << endl;

  return 0;
}
```

問10　学食

問題概要

- 学食に高校生が並ぶ.
- 高校生は1からNまでの番号で識別される.
- 高校生のペア(a_i, b_i)に関して、以下の1つ目と2つ目を組み合わせた制約がC個与えられる.
  - １つ目の制約は順序に関するもので以下のいずれか
    - a_iはb_iよりも先、または同じ位置に並ばなくてはならない
    - a_iはb_iよりも後、または同じ位置に並ばなくてはならない
    - a_iはb_iより先でも、同じ位置でも、後でもよい
  - ２つ目の制約は距離に関するもので以下のいずれか
    - a_iとb_iはd_iメートル以上離れなければならない
    - a_iとb_iはd_iメートル以内に並ばなければならない
- 高校生の列が、最大でどのくらいの距離になるか求める.

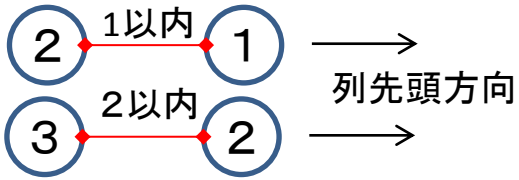
問10　学食

問題概要

- 入出力例1(N=3, C=2):

制約1：1は2より先か同じ位置で、距離1以内に並ぶ.

制約2：2は3より先か同じ位置で、距離2以内に並ぶ.

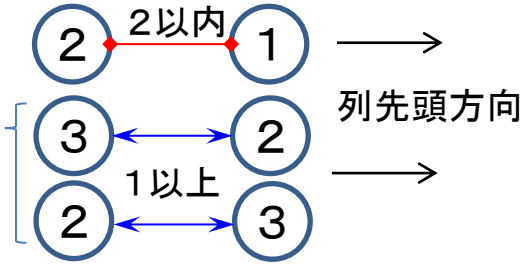


解：1は2より先か同じ位置で、2は3より先か同じ位置なので前後関係が決まる. 各々の間の距離の最大値は、両方とも～以内の制約なので、3になる.

- 入出力例3(N=3, C=2):

制約1：1は2より先か同じ位置で、距離2以内に並ぶ.

制約2：2と3の前後関係は任意で、距離1以上に並ぶ.



解：1は2より先か同じ位置. 2が3より先に並ぶと、2と3の間の距離は、いくらでも大きくできるので、最大距離はinfになる.

問10　学食

講評

- 提出数14、正答数3.
- 制約となる不等式が与えられた状態で最適化する問題なので、一見、線形最適化問題に見える.
- 実はグラフにできる.

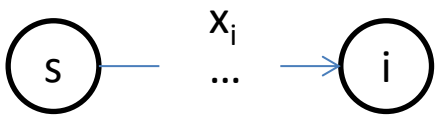
解法

- 変数が2つしかない不等式は、グラフの問題にできる.
- 前後関係が決まったものについては、それらの不等式を使う.
- 前後関係が決まっていないものについては、不等式を2通りとも試す.
- グラフの最短距離が、列の最大距離になる.

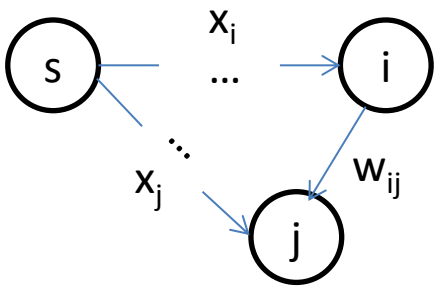
問10　学食

グラフは不等式で表せる

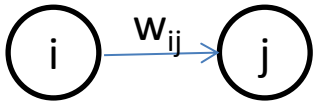
- 始点がsのグラフで、節点iへの最短距離がx_iだとする（複数の節点を経由することもある）.



- 節点jへの最短距離がx_jで、iとjを直接結ぶ辺の距離がw_{ij}だとする.



- x_jは明らかにx_i+w_{ij}以下なので、iからjへの有向辺w_{ij}は不等式で表せる.



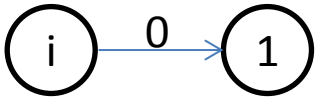
$\Longleftrightarrow$ $x_j - x_i \leqq w_{ij}$

問10　学食

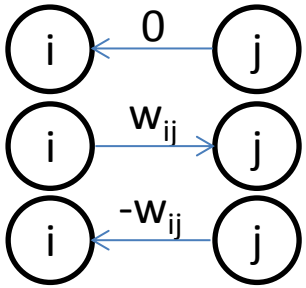
与えられる条件を不等式→グラフで表す

- 列の先頭（節点1）を基準とすると、任意の節点iは節点1より後：

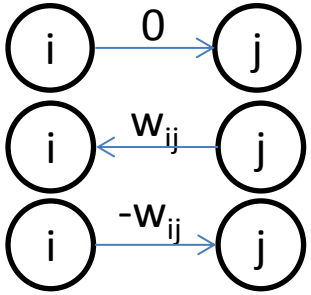
$x_1 - x_i \leqq 0$



- iがjより先か同じ場所に並ぶ：　$x_i - x_j \leqq 0$
- それが距離w_{ij}以内：　$x_j - x_i \leqq w_{ij}$
- それが距離w_{ij}以上：　$x_j - x_i \geqq w_{ij}$　→ $x_i - x_j \leqq -w_{ij}$



- iがjより後か同じ場所に並ぶ：　$x_j - x_i \leqq 0$
- 距離w_{ij}以内：　$x_i - x_j \leqq w_{ij}$
- 距離w_{ij}以上：　$x_i - x_j \geqq w_{ij}$　→ $x_j - x_i \leqq -w_{ij}$



以上の辺を全て張って最短距離を求めたとき、列の最大距離になる.

問10　学食

- 負の辺があるのでベルマンフォード法などを使う.
- 負の閉路がある場合、不可能.
- ただし、順番が任意な条件が6つまで与えられるので、先の場合と後の場合の組み合わせをグラフで作って全て試す(最大2^6=64通り).
- 全ての組み合わせでグラフを作って試したとき、infにできるものがあるならinf、infが無ければ有限の値のうちの最大値、全て不可能なら-1になる.
- 計算量は最大2^6×(節点数)×(辺の数)程度.