パソコン甲子園 2010

【本選問題＊解説】

問題 01 ヒットアンドブロー

問題のポイント

組み合わせを網羅して調べることができるプログラムを作成できるかを問う問題です。

問題の解き方

一般的には、２重ループで４x４の全ての組み合わせを調べ、ヒットとブローの数を数えます。

講評

提出数:34、 正答数:21

全てのチームが正解しました。

解答例(C++)

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

main(){
   string r, a;
   while(1){
       cin >> r >> a;
       if ( a == "0" ) break;
       int hit = 0, blow = 0;
       // 4 x 4 パターン試す
       for ( int i = 0; i < 4; i++ ){
           for ( int j = 0; j < 4; j++ ){
               if ( r[i] != a[j] ) continue;
               if ( i == j ) hit++; // ヒット
               else blow++;        // ブロー
           }
       }
       cout << hit << " " << blow << endl;
   }
}
```

解答例(Java)

```
import java.util.Scanner;

public class P01{
   public static void main(String[] args){
       String r, a;
       Scanner scan = new Scanner(System.in);
       while(true){
           r = scan.next();
           a = scan.next();
           if( a == "0" ) break;
           int hit = 0, blow = 0;
```

```
            for( int i = 0 ; i < 4 ; i++ ){
                for( int j = 0 ; j < 4 ; j++ ){
                    if( r.charAt(i) != a.charAt(j) ) continue;
                    if( i == j ) hit++;
                    else blow++;
                }
            }
        }
        System.out.println( hit + " " + blow );
    }
}
```

問題 02 お客様大感謝祭

問題のポイント

簡単な並べ替えを行うプログラムを作成できるかを問う問題です。

問題の解き方

解答アルゴリズムの一つです。まず、全体の価格を予め計算しておき、値段が大きい順に野菜を整列します。そして、m の倍数番目の野菜の値段を全体の価格から引いた値を求めます。

講評

提出数: 27、正答数: 19

典型的なグリーディー（貪欲）法の問題です。このような問題を解くときにはソートを実装する必要がありますが、C++なら STL の sort 関数を、Java なら Collection.sort を使うことによって実装時間を大幅に短縮することが出来ます。競技プログラミングは時間勝負ですので、プログラミング言語の機能やライブラリで使えるものは積極的に使って実装時間を短縮しましょう。

解答例(C++)

```
#include<iostream>
#include<algorithm>
using namespace std;
#define MAX 1000

main(){
   int n, m, P[MAX], sum;
   while(1){
       cin >> n >> m;
       if ( n == 0 && m == 0 ) break;
       sum = 0;
       for ( int i = 0; i < n; i++ ) {
           cin >> P[i];
           sum += P[i];
       }
       sort(P, P+n, greater<int>());
       for ( int i = m-1; i < n; i += m ) sum -= P[i];
       cout << sum << endl;
   }
}
```

解答例(Java)

```
import java.util.Scanner;
import java.util.ArrayList;
import java.util.Collections;

public class P02{
```

```
    public static void main(String[] argc){
        Scanner scan = new Scanner(System.in);
        while( true ){
            int n = scan.nextInt();
            int m = scan.nextInt();
            if( n == 0 && m == 0 ) break;

            ArrayList<Integer> P = new ArrayList<Integer>();
            int sum = 0;
            for( int i = 0 ; i < n ; i++ ){
                P.add(scan.nextInt());
                sum += P.get(i);
            }
            Collections.sort(P);
            Collections.reverse(P);
            for( int i = m-1 ; i < n ; i += m ) sum -= P.get(i);
            System.out.println( sum );
        }
    }
}
```

問題 03　7セグメント

問題のポイント

現在の状態と変換後の状態との差を求められるかを問う問題です。

問題の解き方

一つのアルゴリズムとして、現在の数字を保持しつつ、読み込まれた数字との排他的論理和を出力していきます。各数字を表すビット列を 10 進数表現で予め準備しておくと、C/C++の排他的論理和の演算子^を直接使用することができます。

講評

提出数:31、正答数:20

ビット演算や逐次計算等、それぞれのアルゴリズムで実装していました。不正解の解答では、セグメントの並びの定義を間違えて入力していたものがありました。このような間違いは、入力例のみに頼らず、独自に全通りの入力を確認することで防ぐことができますので、テストする習慣をつけましょう。

解答例(C++)

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

void printBit(int x, int w){
   if (!w) return;
   printBit(x/2, w-1);
   cout << x%2;
}

main(){
   int B[11] = {63, 6, 91, 79, 102, 109, 125, 39, 127, 111, 0};
   int n, p;
   while( cin >> n && n != -1 ){
       for ( int i = 0, cur = 10; i < n; i++ ){
           cin >> p;
           printBit(B[cur]^B[p], 7);
           cout << endl;
           cur = p;
       }
   }
}
```

解答例(Java)

```
import java.util.Scanner;

public class P03{
    public static void printBit(int x, int w){
        if(w == 0) return;
```

```
        printBit(x/2, w-1);
        System.out.print(x%2);
    }

    public static void main(String[] argc){
       int B[] = {63, 6, 91, 79, 102, 109, 125, 39, 127, 111, 0};
        int n, p;
        Scanner scan = new Scanner(System.in);
        while( scan.hasNextInt() ){
           n = scan.nextInt();
           if( n == -1 ) break;
           for( int i = 0, cur = 10 ; i < n ; i++ ){
                p = scan.nextInt();
                printBit(B[cur]^B[p], 7);
                System.out.println();
                cur = p;
           }
        }
    }
}
```

問題 04 大当たり！

問題のポイント

問題の仕様を文章から正確に読み取りアルゴリズムを組み立てられるかを問う問題です。

問題の解き方

ボーナスゲームの数、通常ゲームの数をそれぞれ求め、手元に残ったメダルの枚数を求める計算式を立てます。

講評

提出数: 21、正答数: 18

最も正答率が高かった問題で、概ね上記の方法で解かれていました。

解答例(C++)

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

main(){
   char n;
   int b, r, g, c, s, t;
   while(1){
       cin >> n >> b >> r >> g >> c >> s >> t;
       if ( n == '0' ) break;
       cout << n << " ";
       int nb = b*5 + r*3; // ボーナスゲームの数
       int no = t - nb;    // 通常ゲームの数
       int cur = 100;
       int result = cur + 15*b + 15*r + 15*nb + 7*g + 2*c - 3*(no-s) - 2*nb;
       cout << result << endl;
   }
}
```

解答例(Java)

```
import java.util.Scanner;

public class P04{
   public static void main(String[] argc){
      String n;
      int b, r, g, c, s, t;
      Scanner scan = new Scanner(System.in);

      while( scan.hasNextInt() ){
         b = scan.nextInt(); r = scan.nextInt(); g = scan.nextInt();
         c = scan.nextInt(); s = scan.nextInt(); t = scan.nextInt();
         if( b == 0 && r == 0 && g == 0 && c == 0 && s == 0 && t == 0 ) break;
         int nb = b*5 + r*3;
         int no = t - nb;
         int cur = 100;
         int result = cur + 15*b + 15*r + 15*nb + 7*g + 2*c - 3*(no-s) - 2*nb;
         System.out.println(result);
```

```
            }
        }
    }
```

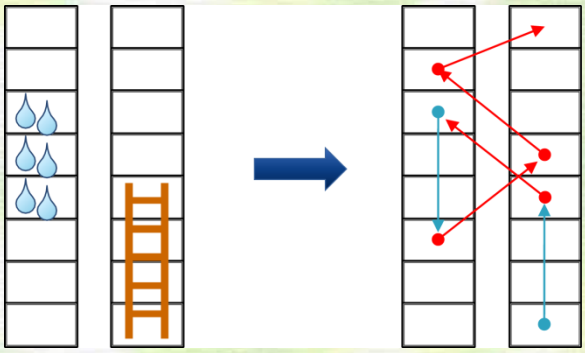
問題 05 忍者のビル登り

問題のポイント

問題の設定を読み取り、適切なモデルを選択し、探索を行えるかを問う問題です。

問題の解き方

グラフ上の幅優先探索で解くアルゴリズムを説明します。下図のように現在の階 p とビルの種類 b のペアをノードとした有向グラフ上で幅優先探索（BFS）を行います。グラフのノード（状態空間）は、ビルの高さを n とすれば、例えば bool V[n][2]と表すことができます。



講評

提出数:35、正答数:9

BFS で解けば良いことに多くのチームが気づいていました。不正解の解答では、梯子やすべる壁の処理がうまくできていないものが多くありました。この場合も、テスト用入出力データを作れば発見できた間違いです。せっかく二人チームなのですから、一人が実装している間にもう一人はテストデータを作る、などのチームワークも効果的です。

解答例(C++)

```
#include<iostream>
#include<queue>
using namespace std;
#define rep(i, n) for ( int i = 0; i < n; i++)
#define MAX 100

class State{
   public:
   int p, b, cost;
   State(int p=0, int b=0, int cost=0):p(p), b(b), cost(cost){}
};

int n, B[MAX][2];
```

```
int bfs(){
   bool V[MAX][2];
   queue<State> Q;
   rep(i, n) rep(j, 2) V[i][j] = false;
   int s0 = 0, s1 = 0;
   if ( B[s0][0] == 1 ) while( s0+1 < n && B[s0+1][0] == 1) s0++;
   if ( B[s1][1] == 1 ) while( s1+1 < n && B[s1+1][1] == 1) s1++;
   V[s0][0] = V[s1][1] = true;
   Q.push(State(s0, 0, 0));
   Q.push(State(s1, 1, 0));

   State u, v;
   while(!Q.empty()){
       u = Q.front(); Q.pop();
       if ( u.p == n-1 ) return u.cost;
       for ( int r = 0; r < 3; r++ ){
           int np = u.p + r;
           int nb = (u.b + 1)%2;
           if ( np >= n ) continue;
           if ( B[np][nb] == 2 ) while( B[np][nb] == 2 ) np--;
           if ( B[np][nb] == 1 ) while( np+1 < n && B[np+1][nb] == 1 ) np++;
           if ( V[np][nb] ) continue;
           V[np][nb] = true;
           Q.push(State(np, nb, u.cost+1));
       }
   }

   return -1;
}

main(){
   while( cin >> n ){
       if ( n == 0 ) break;
       rep(i, n) cin >> B[i][0];
       rep(i, n) cin >> B[i][1];
       int cost = bfs();
       if ( cost < 0 ) cout << "NA" << endl;
       else cout << cost << endl;
   }
}
```

解答例(Java)

```
import java.util.*;
import java.util.concurrent.*;

public class P05{
   static final int MAX = 100;
   static int n;
   static int B[][] = new int[MAX][2];

   public static class State{
      int p, b, cost;
      State(int p, int b, int cost){
         this.p = p;
         this.b = b;
         this.cost = cost;
      }
   }
```

```
    public static int bfs(){
        boolean V[][] = new boolean[MAX][2];
        Queue<State> Q = new ConcurrentLinkedQueue<State>();
        for( int i = 0 ; i < n ; i++ )
            for( int j = 0 ; j < 2 ; j++ )
                V[i][j] = false;
        int s0 = 0, s1 = 0;
        if( B[s0][0] == 1 ) while( s0+1 < n && B[s0+1][0] == 1 ) s0++;
        if( B[s1][1] == 1 ) while( s1+1 < n && B[s1+1][1] == 1 ) s1++;
        V[s0][0] = V[s1][1] = true;
        Q.add(new State(s0, 0, 0));
        Q.add(new State(s1, 1, 0));

        State u, v;
        while(!Q.isEmpty()){
            u = Q.poll();
            if( u.p == n-1 ) return u.cost;
            for( int r = 0 ; r < 3 ; r++ ){
                int np = u.p + r;
                int nb = (u.b + 1)%2;
                if( np >= n ) continue;
                if( B[np][nb] == 2 ) while( B[np][nb] == 2 ) np--;
                if( B[np][nb] == 1 ) while( np+1 < n && B[np+1][nb] == 1 ) np++;
                if( V[np][nb] ) continue;
                V[np][nb] = true;
                Q.add(new State(np, nb, u.cost+1));
            }
        }

        return -1;
    }

    public static void main(String[] argc){
        Scanner scan = new Scanner(System.in);
        while( scan.hasNextInt() ){
            n = scan.nextInt();
            if( n == 0 ) break;
            for( int i = 0 ; i < n ; i++ ) B[i][0] = scan.nextInt();
            for( int i = 0 ; i < n ; i++ ) B[i][1] = scan.nextInt();
            int cost = bfs();
            if( cost < 0 )
                System.out.println("NA");
            else
                System.out.println(cost);
        }
    }
}
```

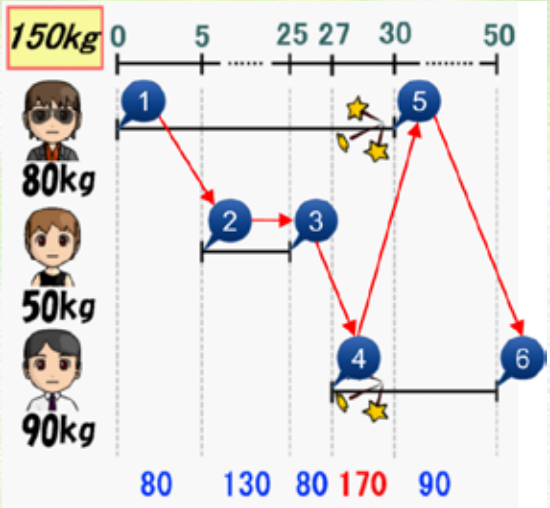
問題 06 危ない橋

問題のポイント

問題の仕様から適切なデータ構造を判断し、データの並べ替えを行うプログラムを作ることができるかを問う問題です。

問題の解き方

この解答例では、ある人が渡り始める時刻、渡り終える時刻を”イベント”とします。下図のように入力情報を元にイベントの列をつくり時刻順に整列します。現在橋の上にいる人の総体重を W とし、イベントの順番に取り出し：それが「渡り始める」なら対応する人の体重を W に足し、それが「渡り終える」なら対応する人の体重を W から引いていきます。この過程で W が強度 150 を超えたら NG と出力します。イベントをソートする場合、時間が同じ場合は「渡り終える」イベントを優先してソートをしなければいけないことに注意が必要です。



講評

提出数: 31、正答数：14

不正解の解答では、時間を一単位ずつ進めて全ての時間で橋を渡っている人を判定するものが多くありました。この場合、時間を t、橋を渡る人の数を n とすると計算量は O(nt)程度になります。この問題の場合、n が 100、t が 20,000,000 程度ですからこの解法は現実的でないことが分かります。実装しようとするアルゴリズムの計算量をあらかじめ見積もることも効率の上で重要です。

解答例(C++)

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

#define OUT 0
#define IN 1
#define MAX 100

class Event{
   public:
   int id, type, t;
   Event(int id=0, int type=0, int t=0):id(id), type(type), t(t){}
   bool operator < ( const Event &e) const{
       if ( t == e.t ) return type < e.type;
       return t < e.t;
   }
};

main(){
   int s = 150, n, a, b, M[MAX];

   while( cin >> n && n ){
       Event events[MAX*2];
       int ne = 0;
       for ( int i = 0; i < n; i++ ){
           cin >> M[i] >> a >> b;
           events[ne++] = Event(i, IN, a);
           events[ne++] = Event(i, OUT, b);
       }

       sort(events, events + ne);

       bool fail = false;
       int cur = 0;

       for ( int i = 0; i < ne; i++){
           if ( events[i].type == OUT ) cur -= M[events[i].id];
           if ( events[i].type == IN )  cur += M[events[i].id];
           if ( cur > s ) fail = true;
       }

       if ( fail ) cout << "NG" << endl;
       else cout << "OK" << endl;
   }
}
```

解答例(Java)

```
import java.util.Scanner;
import java.util.Arrays;
import java.util.ArrayList;
import java.lang.Comparable;
import java.util.Collections;

public class P06{
   static final int OUT = 0;
   static final int IN  = 1;
   static final int MAX = 100;

   public static class Event implements Comparable<Event> {
       int id, type, t;
       public Event( int id, int type, int t ){
```

```
            this.id = id;
            this.type = type;
            this.t = t;
        }

        public int compareTo(Event e){
            if( this.t == e.t )
                return this.type - e.type;
            else
                return this.t - e.t;
        }
    }

    public static void main(String[] argc){
        int s = 150, n, M[] = new int[MAX];
        Scanner scan = new Scanner(System.in);

        while( scan.hasNext() ){
            n = scan.nextInt();
            if( n == 0 ) break;
            ArrayList<Event> events = new ArrayList<Event>();
            for( int i = 0 ; i < n ; i++ ){
                M[i] = scan.nextInt();
                events.add(new Event(i, IN,  scan.nextInt()));
                events.add(new Event(i, OUT, scan.nextInt()));
            }
            Collections.sort( events );

            boolean fail = false;
            int cur = 0;

            for( int i = 0 ; i < events.size() ; i++ ){
                if( events.get(i).type == OUT ) cur -= M[events.get(i).id];
                if( events.get(i).type == IN )  cur += M[events.get(i).id];
                if( cur > s ) fail = true;
            }

            if( fail ) System.out.println("NG");
            else System.out.println("OK");
        }
    }
}
```

問題 07 人生ゲーム

問題のポイント

動的計画法によって確率が計算できるかを問う問題です。

問題の解き方

動的計画法を用いた解答アルゴリズム例です。dp[i][j]を i 円持った状態でマス j にいる確率とし、dp[0][0]（0 円持った状態で最初のマスにいる確率）= 1.0 より開始し、j が 0 から最後のマス Y までについて dp[i][j]を動的計画法により求めていきます。dp[i][Y]（i=1, 2, ... 5000）より期待値を計算します。

講評

提出数: 15、正答数:5

動的計画法は慣れていないと難しいですが、比較的実装量が少ないので効率良く解くことができます。普段の練習で動的計画法に慣れておくと良いでしょう。

解答例(C++)

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

#define MAX 100
#define MMAX 5001

int X, Y, Z, A[MAX], V[MAX], E[MAX];

void compute(){
   float P[MMAX][MAX];
   for ( int i = 0; i < MMAX; i++ ){
       for ( int j = 0; j <= Y; j++ ) P[i][j] = 0;
   }
   P[0][0] = 1.0;

   int np, nv;

   for ( int j = 0; j < Y; j++ ){
       for ( int i = 0; i < MMAX; i++ ){
           if ( P[i][j] == 0 ) continue;

           for ( int k = 0; k < X; k++ ){
               np = min(Y, j + V[k]);
               nv = i;
               if ( E[np] == 1 ) {
                   np = min(Y, np + A[np]);
               } else if ( E[np] == 2 ){
                   nv = i + A[np];
               } else if ( E[np] == 3 ){
```

```
                    nv = max(0, i - A[np]);
                }

                P[nv][np] += P[i][j]*(1.0/X);
            }
        }
    }

    double ex = 0;
    for ( int i = 0; i < MMAX; i++ ) ex += P[i][Y]*i;
    cout << (int)ex << endl;
}

main(){
    int n, e, a;
    while(1){
        cin >> X >> Y >> Z;
        if ( X == 0 && Y == 0 && Z == 0 ) break;
        for ( int i = 0; i <= Y; i++ ) A[i] = E[i] = 0;
        for ( int i = 0; i < X; i++ ) cin >> V[i];
        for ( int i = 0; i < Z; i++ ){
            cin >> n >> e >> a;
            A[n] = a;
            E[n] = e;
        }
        compute();
    }
}
```

解答例(Java)

```
import java.util.Scanner;

public class P07{
    static final int MAX = 100;
    static final int MMAX = 5001;

    static int X, Y, Z;
    static int A[] = new int[MAX], V[] = new int[MAX], E[] =newint[MAX];

    public static void compute(){
        double P[][] = new double[MMAX][MAX];
        for( int i = 0 ; i < MMAX ; i++ )
            for( int j = 0 ; j <= Y ; j++ )
                P[i][j] = 0.0;
        P[0][0] = 1.0;

        int np, nv;

        for( int j = 0 ; j < Y ; j++ ){
            for( int i = 0 ; i < MMAX ; i++ ){
                if( P[i][j] == 0 ) continue;
                for( int k = 0 ; k < X ; k++ ){
                    np = Math.min(Y, j + V[k]);
                    nv = i;
                    if( E[np] == 1 )
                        np = Math.min(Y, np + A[np]);
                    else if( E[np] == 2 )
                        nv = i + A[np];
                    else if( E[np] == 3 )
                        nv = Math.max(0, i - A[np]);
```

```
                    P[nv][np] += P[i][j]*(1.0/X);
                }
            }
        }

        double ex = 0;
        for( int i = 0 ; i < MMAX ; i++ ) ex += P[i][Y]*i;
        System.out.println( (int)ex );
    }

    public static void main(String[] argc){
        int n, e, a;
        Scanner scan = new Scanner(System.in);
        while(scan.hasNextInt()){
            X = scan.nextInt();
            Y = scan.nextInt();
            Z = scan.nextInt();
            if( X == 0 && Y == 0 && Z == 0 ) break;
            for( int i = 0 ; i <= Y ; i++ ) A[i] = E[i] = 0;
            for( int i = 0 ; i < X ; i++ )  V[i] = scan.nextInt();
            for( int i = 0 ; i < Z ; i++ ){
                n = scan.nextInt();
                e = scan.nextInt();
                a = scan.nextInt();
                A[n] = a;
                E[n] = e;
            }
            compute();
        }
    }
}
```

問題 08 図書整理

問題のポイント

10 進数の数をマイナス 10 進数に変換するアルゴリズムを考案できるかを問う問題です。

問題の解き方

対象とする数値が非常に大きな数なので、最初に範囲内のマイナス 10 進数を生成する方法は適用できません。そこで、ここでは各桁（1、-10、100、-1000、...）がそれぞれいくつ必要なのかを、小さい桁から順番に求めていきます。

講評

提出数：26、 正答数：4

通常の基数変換の方法は適用できませんが、そもそも n 進数がどういった数の表現方法なのかを考えてみれば素直に解ける問題です。

解答例(C++)

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

void convert(int x, int b){
   if ( x == 0 ) return;
   int p, l = x%10;
   if ( l < 0 ) l *= -1;
   if ( l == 0 ){
       p = 0;
   } else if ( x*b < 0 ){
       p = 10-l;
       x -= b*10;
   } else {
       p = l;
   }
   convert(x/10, b*(-1));
   cout << p;
}

main(){
   int x;
   while(cin >> x && x ){
       convert(x, 1);
       cout << endl;
   }
}
```

解答例(Java)

```
import java.util.Scanner;

public class P08{
    public static void convert(int x, int b){
       if( x == 0 ) return;
```

```
        int p, l = x%10;
        if( l < 0 ) l *= -1;
        if( l == 0 ){
            p = 0;
        } else if( x*b < 0 ){
            p = 10-l;
            x -= b*10;
        } else {
            p = l;
        }
        convert(x/10, b*(-1));
        System.out.print(p);
    }

    public static void main(String[] argc){
        int x;
        Scanner scan = new Scanner(System.in);
        while( scan.hasNextInt() ){
            x = scan.nextInt();
            if( x == 0 ) break;
            convert(x, 1);
            System.out.println();
        }
    }
}
```

問題 09 サージェント・ライアン

問題のポイント

木構造を体系的に走査するプログラムを作成できるかを問う問題です。

問題の解き方

最後にいる島はどこでも良いことから、最後の島全通りについて必要な時間を調べ、最小値を求めます。最後の島を s とし、s を根とする木を生成し、スタート地点である島 1 から深さ優先探索で橋を渡る時間の合計を求めます。深さ優先探索の過程で、(1)渡った島のさらに先に島（子ノード）がなければ、もとの島にもどり、(2)渡った島のさらに先に島（子ノード）があれば、渡ってきた橋のコストの 2 倍を加算し、(3)親ノードに移動する場合は、親ノードに渡るための橋のコストを加算します。

講評

提出数:10、　正答数:7

一見難しそうにですが、それほど難しくはありません。特に DFS などの縦型の探索は実際の計算量がどの程度か、どれくらい枝刈りが出るかという点に着目するようにしましょう。

解答例(C++)

```
#include<iostream>
#include<vector>
using namespace std;
#define rep(i, n) for ( int i = 0; i < n; i++ )
#define MAX 20
#define INFTY (1<<21)

class Node{
   public:
   int P;
   vector<int> C;
   Node(){
       P = -1;
       C.clear();
   }
};

int N, G[MAX][MAX], total;
Node node[MAX];
bool visited[MAX];

void dfs(int u, int p){
   visited[u] = true;
   node[u].P = p;
   rep(v, N){
       if ( G[u][v] == INFTY || visited[v]) continue;
       node[u].C.push_back(v);
       dfs(v, u);
   }
```

```
}

void parse(int u, int cost){
   visited[u] = true;
   if ( node[u].C.size() == 0 ) return;

   for ( int i = 0; i < node[u].C.size(); i++ ){
       int v = node[u].C[i];
       if ( !visited[v] ) parse(v, G[u][v]);
   }
   total += cost*2;
}

int getCost(){
   rep(i, N) visited[i] = false;
   int cur = 0;
   total = 0;
   while( cur != -1 ){
       parse(cur, 0);
       if ( node[cur].P != -1 ) total += G[cur][node[cur].P];
       cur = node[cur].P;
   }
   return total;
}

int compute(){
   int minc = INFTY;
   for ( int s = 0; s < N; s++ ){
       rep(i, N) { visited[i] = false; node[i] = Node(); }
       dfs(s, -1);
       minc = min(minc, getCost());
   }
   return minc;
}

main(){
   int s, t, c;
   while( cin >> N && N ){
       rep(i, N) rep(j, N) G[i][j] = INFTY;
       rep(i, N-1){
           cin >> s >> t >> c;
           s--; t--;
           G[s][t] = G[t][s] = c;
       }
       cout << compute() << endl;
   }
}
```

解答例(Java)

```
import java.util.Scanner;
import java.util.ArrayList;

public class P09{
   public static final int MAX = 20;
   public static final int INFTY = 1 << 21;

   public static class Node{
       int P;
       ArrayList<Integer> C = new ArrayList<Integer>();

       Node(){
       this.P = -1;
       this.C.clear();
```

```
    }
}

public static int N, total, G[][] = new int[MAX][MAX];
public static Node node[] = new Node[MAX];
public static boolean visited[] = new boolean[MAX];

public static void dfs(int u, int p){
    visited[u] = true;
    node[u].P = p;
    for( int v = 0 ; v < N ; v++ ){
        if( G[u][v] == INFTY || visited[v] ) continue;
        node[u].C.add(v);
        dfs(v, u);
    }
}

public static void parse(int u, int cost){
    visited[u] = true;
    if( node[u].C.size() == 0 ) return;

    for( int i = 0 ; i < node[u].C.size() ; i++ ){
        int v = node[u].C.get(i);
        if( !visited[v] ) parse(v, G[u][v]);
    }
    total += cost*2;
}

public static int getCost(){
    for( int i = 0 ; i < N ; i++ ) visited[i] = false;
    int cur = 0;
    total = 0;
    while( cur != -1 ){
        parse(cur, 0);
        if( node[cur].P != -1 ) total += G[cur][node[cur].P];
        cur = node[cur].P;
    }
    return total;
}

public static int compute(){
    int minc = INFTY;
    for( int s = 0 ; s < N ; s++ ){
        for( int i = 0 ; i < N ; i++ ){
            visited[i] = false;
            node[i] = new Node();
        }
        dfs(s, -1);
        minc = Math.min(minc, getCost());
    }
    return minc;
}

public static void main(String[] argc){
    int s, t, c;
    Scanner scan = new Scanner(System.in);
    while( scan.hasNextInt() ){
        N = scan.nextInt();
        if( N == 0 ) break;

        for( int i = 0 ; i < N ; i++ )
        for( int j = 0 ; j < N ; j++ )
            G[i][j] = INFTY;
```

```
            for( int i = 0 ; i < N-1 ; i++ ){
                s = scan.nextInt();
                t = scan.nextInt();
                c = scan.nextInt();
                s--; t--;
                G[s][t] = G[t][s] = c;
            }
            System.out.println( compute() );
        }
    }
}
```

問題 10 会津の埋蔵金

問題のポイント

最短経路を求めるアルゴリズムを応用できるかを問う問題です。また、探索の空間を絞り込む工夫も要求されます。

問題の解き方

解答アルゴリズム例の一つです。まず探索の空間を絞り込むことを考えます。

1.上に戻ることは無いので、今いる高さの横の状態を記録しておけばよい、

2.横の動き方は連続的であり、あるセルを飛ばして掘るような状態は存在しないため、今いる高さで行ったことのある左端と右端のみを記録しておけばよい、

ことから、状態の上限は $H \times W \times W \times W \times m = 10 \times 10 \times 10 \times 10 \times 50 = 500000$ に落とすことができます。

講評

提出数：26、　正答数：1

動的計画法やダイクストラ等で最短経路を求める、というアプローチ自体はどのチームもわかっていたようですが、この問題で重要になるのは探索空間をどのようにとるかです。不正解の解答の多くは、そもそも答えが得られないような探索空間を探している（探索空間がグリッドのサイズと同じ）という間違いがありました。ある程度の探索の問題の場合、探索空間と目に見えているグラフなどのデータ構造が一致することはほとんどありません。本当に探索しなくてはいけない対象を見極められるようにしましょう。

解答例(C++)

```
#include<iostream>
#include<queue>
#include<algorithm>
#include<set>
#include<vector>
using namespace std;
#define rep(i, n) for ( int i = 0; i < n ;i++ )
#define MAX 100

int H, W;
int dy[3] = {0, 0, 1};
int dx[3] = {1, -1,0};

class State{
   public:
   int l, r, y, x, o;
   State(int y=0, int x=0, int o=0):y(y), x(x), o(o){
       reset();
   }
```

```
    void reset(){ l = W; r = -1; }
    void visit(int j){
        l = min(l, j);
        r = max(r, j);
    }
    bool visited(int j){
        return l <= j && j <= r;
    }

    bool operator < ( const State &s) const{
        if ( o != s.o ) return o < s.o;
        if ( l != s.l ) return l < s.l;
        if ( r != s.r ) return r < s.r;
        if ( y != s.y ) return y < s.y;
        if ( x != s.x ) return x < s.x;
        return false;
    }
};

class QState{
    public:
    State state;
    int cost;
    QState(){}
    QState(State state, int cost): state(state), cost(cost){}
    bool operator < ( const QState &s) const{
        return cost > s.cost;
    }
};

int F, M, O, C[MAX][MAX];

int dijkstra(){
    if ( O == 0 ) return -1;
    int ncost;

    priority_queue<QState> PQ;
    set<State> visited;

    rep(j, W){
        State source = State(0, j, O);
        source.visit(j);
        source.o--;
        if ( source.o == 0 ) continue;

        if ( C[0][j] >= 0 ){
            source.o = min(M, source.o + C[0][j]);
            ncost = 0;
        } else {
            ncost = C[0][j]*(-1);
        }
        visited.insert(source);
        PQ.push(QState(source, ncost));
    }

    QState u, v;
    int nx, ny;
    while(!PQ.empty()){
        u = PQ.top(); PQ.pop();
        if ( u.cost > F ) continue;
        if ( u.state.y == H-1) return u.cost;

        rep(r, 3){
            nx = u.state.x + dx[r];
            ny = u.state.y + dy[r];
```

```
            if ( nx < 0 || ny < 0 || nx >= W || ny >= H ) continue;

            v = u;
            v.state.o--;
            if ( v.state.o == 0 ) continue;
            v.state.x = nx;
            v.state.y = ny;

            if ( r == 2 ) v.state.reset();

            if ( !v.state.visited(nx) ){
                if ( C[ny][nx] >= 0 ){
                    v.state.o = min(M, v.state.o + C[ny][nx]);
                    v.state.visit(nx);
                } else if ( C[ny][nx] < 0 ){
                    v.cost = u.cost + (C[ny][nx]*(-1));
                    v.state.visit(nx);
                }
            }

            if( visited.find(v.state) == visited.end()){
                visited.insert(v.state);
                PQ.push(v);
            }
        }
    }

    return -1;
}

main(){
    while( cin >> W >> H && W && H ){
        cin >> F >> M >> O;
        rep(i, H) rep(j, W) cin >> C[i][j];
        int cost = dijkstra();
        if ( cost < 0 || F < cost) cout << "NA" << endl;
        else cout << cost << endl;
    }
}
```

問題 11 宇宙人のきまぐれメッセージ

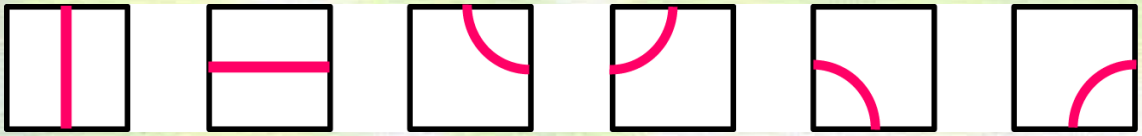
問題のポイント

全探索によって解が存在するか否かを判定できるかを問う問題です。

問題の解き方

このアルゴリズムでは、基本的には、バックトラックにより全通りの線の引き方を再帰によって試します。ただし明らかに必要のない探索を行わないように”枝刈り”を行う必要があります。サイズが小さいので特別な高速化を行う必要はありません。閉路が複数できる場合や空白のセルがないケースはどちらも NO と判定しなければならないことに注意しましょう。

以下の解答例では、下図に示す 6 つのタイルをそれぞれ、10、5、3、6、12、9 の数字で表しています。これは、セルの右の辺から反時計回りに 1、2、4、8 を割り当て、線が存在する辺に対応した数字を足し合わせたものです。例えば、左のタイルは上と下の数字である 2 と 8 を足し合わせて 10 となります。



講評

提出数：56、　正答数：2

このような縦型の探索を用いる問題は、横型の探索に比べて不得意な傾向が見られますが、枝刈りを用いる探索ではさらにその傾向が強いようです。探索では常に枝刈りを意識して、あり得ない状態を見極める習慣をつけましょう。

解答例(C++)

```
#include<iostream>
#include<cstdio>
using namespace std;
#define rep(i,n) for ( int i = 0; i < n; i++ )
#define MAX 20

static const int T[6] = {10, 5, 3, 6, 12, 9};
int G[MAX][MAX], H, W, ncell, vcell, V[MAX][MAX];

bool eq(int a, int i, int b, int j){
   return ((a & (1<<i)) > 0) == ((b & (1<<j)) > 0);
}

void dfs(int i, int j){
```

```
    V[i][j] = true;
    vcell++;
    static int di[4] = {0, -1, 0, 1};
    static int dj[4] = {1, 0, -1, 0};

    rep(s, 4){
        int ni = i + di[s];
        int nj = j + dj[s];
        if ((G[i][j] & (1<<s)) && !V[ni][nj]) dfs(ni, nj);
    }
}

bool rec(int pos){
    if ( pos == H*W ) {
        vcell = 0;
        rep(i, H) rep(j, W) V[i][j] = false;
        rep(i, H) rep(j, W){
            if ( G[i][j] == 0 ) continue;
            dfs(i, j);
            return ncell == vcell;
        }
        return false;
    }
    int i = pos/W;
    int j = pos%W;
    if ( G[i][j] == 0 ){
        int v = 0;
        if ( j && !eq( G[i][j-1], 0, v, 2 ) )  return false;
        if ( i && !eq( G[i-1][j], 3, v, 1 ) )   return false;
        if ( rec(pos+1) ) return true ;
    } else {
        rep(t, 6){
            int v = T[t];
            if ( j && !eq( G[i][j-1], 0, v, 2 ) ) continue;
            if ( i && !eq( G[i-1][j], 3, v, 1 ) ) continue;
            if ( i == 0 && (v&(1<<1)) ) continue;
            if ( i == H-1 && (v&(1<<3)) ) continue;
            if ( j == 0 && (v&(1<<2)) ) continue;
            if ( j == W-1 && (v&1) ) continue;
            G[i][j] = v;
            if ( rec(pos+1) ) return true;
        }
    }
    return false;
}

main(){
    while(1){
      cin >> W >> H;
        if ( H == 0 && W == 0 ) break;
        ncell = 0;
        rep(i, H) rep(j, W){
            cin >> G[i][j];
            if ( !G[i][j] ) ncell++;
            G[i][j] = (G[i][j] ? 0 : 1);
        }
        if (rec(0)) cout << "Yes" << endl;
        else cout << "No" << endl;
    }
```

```
}
```

解答例(Java)

```
import java.util.*;

public class P11{
   static final int MAX = 20;
   static final int T[] = {10, 5, 3, 6, 12, 9};
   static int G[][] = new int[MAX][MAX], V[][] = new int[MAX][MAX];
   static int H, W, ncell, vcell;
   public static boolean eq(int a, int i, int b, int j){
      return ((a & (1<<i)) > 0) == ((b & (1<<j)) > 0);
   }

   public static void dfs(int i, int j){
      V[i][j] = 1;//true;
      vcell++;
      int di[] = {0, -1, 0, 1};
      int dj[] = {1, 0, -1, 0};

      for( int s = 0 ; s < 4 ; s++ ){
         int ni = i + di[s];
         int nj = j + dj[s];
         if((G[i][j] & (1<<s)) > 0 && V[ni][nj] == 0) dfs(ni, nj);
      }
   }

   public static boolean rec(int pos){
      if( pos == H*W ){
         vcell = 0;
         for( int i = 0 ; i < H ; i++ )
            for( int j = 0 ; j < W ; j++ )
               V[i][j] = 0;//false;
         for( int i = 0 ; i < H ; i++ ){
            for( int j = 0 ; j < W ; j++ ){
               if( G[i][j] == 0 ) continue;
                  dfs(i, j);
                  return ncell == vcell;
            }
         }
         return false;
      }
      int i = pos/W;
      int j = pos%W;
      if( G[i][j] == 0 ){
         int v = 0;
         if( j > 0 && !eq( G[i][j-1], 0, v, 2 ) ) return false;
         if( i > 0 && !eq( G[i-1][j], 3, v, 1 ) ) return false;
         if( rec( pos+1 ) ) return true;
      } else {
         for( int t = 0 ; t < 6 ; t++ ){
            int v = T[t];
            if( j > 0   && !eq( G[i][j-1], 0, v, 2 ) ) continue;
            if( i > 0   && !eq( G[i-1][j], 3, v, 1 ) ) continue;
            if( i == 0  && (v&(1<<1)) > 0 ) continue;
            if( i == H-1 && (v&(1<<3)) > 0 ) continue;
            if( j == 0  && (v&(1<<2)) > 0 ) continue;
            if( j == W-1 && (v&1) > 0 )     continue;
            G[i][j] = v;
            if( rec(pos+1) ) return true;
         }
      }
      return false;
```

```
    }

    public static void main(String[] argc){
    Scanner scan = new Scanner(System.in);
    while( scan.hasNextInt() ){
       W = scan.nextInt(); H = scan.nextInt();
       if( H == 0 && W == 0 ) break;
       ncell = 0;
       for( int i = 0 ; i < H ; i++ ){
          for( int j = 0 ; j < W ; j++ ){
             G[i][j] = scan.nextInt();
             if( G[i][j] == 0 ) ncell++;
             G[i][j] = (G[i][j] > 0 ? 0 : 1);
          }
       }
       if(rec(0)) System.out.println("Yes");
       else      System.out.println("No");
       }
    }
}
```

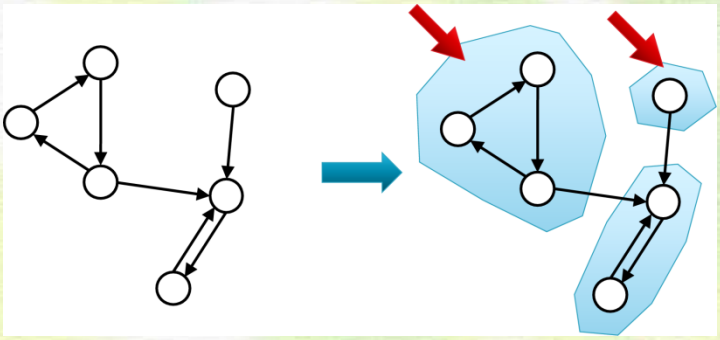
問題 12 最後の扉

問題のポイント

計算幾何学の実装力とグラフの応用知識を問う問題です。

問題の解き方

各三角形について、形成される長方形と他の全ての三角形について交差判定を行い有向グラフを生成します。三角形 u の光によって三角形 v が発光するとき、ノード u からノード v へ向かってエッジを追加します。このグラフを強連結成分分解し、入力エッジを含まない成分の数を求めます。



講評

提出数：8、　正答数:0

残念ながら正解解答はありませんでした。提出された解答も正解まではほど遠いようでした。このレベルの問題を時間内に解くのは非常に困難ですので、気落ちせずに楽しみとして解いてみてください。

解答例(C++)

```
#include<iostream>
#include<vector>
#include<cfloat>
#include<cassert>
#include<cmath>
#include<vector>
#include<algorithm>

using namespace std;
#define rep(i, n) for ( int i = 0; i < (int)n; i++)
#define EPS (1e-8)
#define equals(a, b) (fabs((a) - (b)) < EPS )
#define dle(a, b) (equals(a, b) || a < b )

#define MAX 100

class Point{
   public:
```

```
   double x, y;

   Point ( double x = 0, double y = 0): x(x), y(y){}
   Point operator + ( Point p ){ return Point(x + p.x, y + p.y); }
   Point operator - ( Point p ){ return Point(x - p.x, y - p.y); }
   Point operator * ( double a ){ return Point(x*a, y*a); }

   double abs() { return sqrt(norm());}
   double norm() { return x*x + y*y; }

   bool operator < ( const Point &p ) const {
       return x != p.x ? x < p.x : y < p.y;
   }

   bool operator == ( const Point &p ) const {
       return fabs(x-p.x) < EPS && fabs(y-p.y) < EPS;
   }
};
typedef Point Vector;
typedef vector<Point> Polygon;

double norm( Vector a ){ return a.x*a.x + a.y*a.y; }
double dot( Vector a, Vector b ){ return a.x*b.x + a.y*b.y; }
double cross( Vector a, Vector b ){ return a.x*b.y - a.y*b.x; }
static const int COUNTER_CLOCKWISE = 1;
static const int CLOCKWISE = -1;
static const int ONLINE_BACK = 2;
static const int ONLINE_FRONT = -2;
static const int ON_SEGMENT = 0;

int ccw( Point p0, Point p1, Point p2 ){
   Vector a = p1 - p0;
   Vector b = p2 - p0;
   if ( cross(a, b) > EPS ) return COUNTER_CLOCKWISE;
   if ( cross(a, b) < -EPS ) return CLOCKWISE;
   if ( dot(a, b) < -EPS ) return ONLINE_BACK;
   if ( norm(a) < norm(b) ) return ONLINE_FRONT;
   return ON_SEGMENT;
}
bool isIntersect(Point p1, Point p2, Point p3, Point p4){
   return ( ccw(p1, p2, p3) * ccw(p1, p2, p4) <= 0 &&
            ccw(p3, p4, p1) * ccw(p3, p4, p2) <= 0 );
}

bool isInside(Polygon gon, Point p ){
   for ( int i = 0; i < gon.size(); i++ ){
       if ( ccw(gon[i], gon[(i+1)%gon.size()], p) == CLOCKWISE ) return false;
   }
   return true;
}

class Triangle{
   public:
   vector<Point> P;
   Triangle(){ P.resize(3);}

   void normalize(){
       vector<Point> tmp(3);
       if ( norm(P[0]-P[1]) == norm(P[0]-P[2]) ){
           tmp[0] = P[1]; tmp[1] = P[2]; tmp[2] = P[0];
       } else if ( norm(P[1]-P[0]) == norm(P[1]-P[2]) ){
           tmp[0] = P[0]; tmp[1] = P[2]; tmp[2] = P[1];
       } else if ( norm(P[2]-P[0]) == norm(P[2]-P[1]) ){
           tmp[0] = P[0]; tmp[1] = P[1]; tmp[2] = P[2];
       }
```

```
        P = tmp;
        if ( ccw(P[0], P[1], P[2]) != COUNTER_CLOCKWISE ) {
            swap(P[0], P[1]);
        }
    }
};

class Graph{
public:
    int n;
    vector<vector<int> > adj;
    Graph(int n = 0):n(n){
        adj.resize(n);
        rep(i, n) adj[i].clear();
    }
    void connect(int i, int j){ adj[i].push_back(j); }
    void transpose(){
        vector<vector<int> > nadj;
        nadj.resize(n);
        rep(i, n) nadj[i].clear();
        rep(i, n) rep(j, adj[i].size()) nadj[adj[i][j]].push_back(i);
        rep(i, n) adj[i] = nadj[i];
    }
};

int n;
double d;
Triangle T[MAX];

Graph g;
int visited[MAX], finish[MAX], t;
vector<int> com;

bool overlapPolygon(Polygon p1, Polygon p2){
    rep(i, p1.size()) if ( isInside(p2, p1[i]) ) return true;
    rep(i, p2.size()) if ( isInside(p1, p2[i]) ) return true;
    rep(i, p1.size()) rep(j, p2.size()){
      if ( isIntersect(p1[i], p1[(i+1)%p1.size()], p2[j], p2[(j+1)%p2.size()]))
        return true;
    }
    return false;
}

bool overlap( int t1, int t2){
    Triangle s = T[t1];
    vector<Point> rect, tri;
    Point c = Point((s.P[0].x+s.P[1].x)/2, (s.P[0].y+s.P[1].y)/2);
    Vector v = s.P[2] - c;
    double a = v.abs();
    v.x = d*v.x/a;
    v.y = d*v.y/a;
    Point l = s.P[0] + v;
    Point r = s.P[1] + v;
    rect.push_back(s.P[1]);
    rect.push_back(r);
    rect.push_back(l);
    rect.push_back(s.P[0]);
    tri = T[t2].P;
    return overlapPolygon(rect, tri);
}

void dfs(int &u){
    visited[u] = t++;
    rep(i, g.adj[u].size()){
        int v = g.adj[u][i];
```

```
        if ( visited[v] == -1 ) dfs(v);
    }
    finish[u] = t++;
}

void dfsT(int &u){
    com.push_back(u);
    visited[u] = 1;
    rep(i, g.adj[u].size()){
        int v = g.adj[u][i];
        if ( visited[v] == 0 ) dfsT(v);
    }
}

void makeGraph(){
    g = Graph(n);
    rep(i, n) rep(j, n){
        if ( i != j && overlap(i, j) ) {
            g.connect(i, j);
        }
    }
}

int scc(){
    int id[MAX];
    bool indeg[MAX];

    rep(i, n) visited[i] = finish[i] = -1;
    t = 0;
    rep(i, n) if ( visited[i] == -1 ) dfs(i);

    g.transpose();

    vector<pair<int, int> > order;
    rep(i, n) order.push_back(make_pair(finish[i], i));
    sort(order.begin(), order.end());

    rep(i, n) visited[i] = 0;

    int ncom = 0;

    for ( int i = n-1; i >= 0; i-- ){
        int u = order[i].second;
        com.clear();
        if ( visited[u] == 0 ) {
            dfsT(u);
            rep(c, com.size()) id[com[c]] = ncom;
            ncom++;
        }
    }

    int cnt = 0;
    g.transpose();

    rep(i, ncom) indeg[i] = false;

    rep(u, n) rep(i, g.adj[u].size()){
        int v = g.adj[u][i];
        if ( id[v] != id[u] ) indeg[id[v]] = true;
    }
    rep(i, ncom) if (!indeg[i]) cnt++;

    return cnt;
}
```

```
main(){
   while( cin >> n >> d && n ){
       rep(i, n){
           Triangle t;
           rep(j, 3) cin >> t.P[j].x >> t.P[j].y;
           t.normalize();
           T[i] = t;
       }
       makeGraph();
       cout << scc() << endl;
   }
}
```