パソコン甲子園 2009

【本選問題＊解説】

問題 01　じゃんけん

問題のポイント

プログラミング言語の基本的な構文である条件分岐と繰り返しそして基本データ構造である配列を使い、簡単なアルゴリズムが実装できるかを問う問題です。

問題の解き方

様々なアルゴリズムが考えられる興味深い問題の１つです。ここでは、２重ループを使ったアルゴリズムを説明します。各人について他の４人に対する勝ち・負けの数をそれぞれ数え、「勝ちが１つ以上でかつ負けが 0」の場合は勝ち（1)、「負けが１つ以上でかつ勝ちが 0」の場合は負け（2)、その他の場合は引き分け（3)、と判定することができます。２人に対する勝敗判定については、1（グー）→ 2 （チョキ）→ 3（パー）→ 1（グー)・・・という関係があるので、A君の手を a、B君の手を b とすれば、a%3 が b-1 に等しいとき、A君がB君に勝ったと判定することができます。ここで、a%3 は a を 3 で割った余りを示します。

データセットごとに処理しなければならない入力については、一人目の手を読み込みそれが 0 でない場合処理を行うという繰り返し構造とし、処理の最初に残りの４人分の手を読み込むとう手順で記述すると良いでしょう。

講評

提出数：45、正解数：19

不正解の解答で最も目立ったのは、勝敗判定における判定ミスです。判定ミスの原因は多岐にわたりますが、あいこの判定においては、直前または直後の手とだけではなく、全体での判定も忘れずに行いましょう。

解答例

```
#include<iostream>
using namespace std;
#define N 5

main(){
   int T[N];
   while( cin >> T[0] && T[0] ){
        for ( int i = 1; i < N; i++ ) cin >> T[i];
        for ( int i = 0; i < N; i++ ){
           int w = 0, l = 0;
           for ( int j = 0; j < N; j++ ){
                if ( T[i]%3 == T[j]-1 ) w++;
                if ( T[j]%3 == T[i]-1 ) l++;
```

```
            }
            if ( w && !l ) cout << 1 << endl;
            else if ( l && !w ) cout << 2 << endl;
            else cout << 3 << endl;
        }
    }
}
```

問題 02　旅行はいつ？

問題のポイント

基本的なプログラミングの能力を問う問題です。配列等のデータ構造も必要なく、単純ループと条件分岐で実装することができるので、アルゴリズムは問題 01 よりも単純になります。

問題の解き方

合計金額 S が目的の金額 L に達したとしても 12 カ月分（i: 1～12）の M と N の組を読み込まなければならないことに注意しながら、(M - N)の合計金額 S が目的の金額 L に達する月 t を求めます。t の初期値を 0 とし、S が L 以上でかつ t が 0（初期値から未だ変更されていないことを示す）のとき、t に現在の月 i を代入します。

講評

提出数：47、正解数：18

入力では 12 ヶ月分の収支情報を読み込まなければなりませんが、目標金額に達した時点で入力の読み込みを終了してしまっているために不正解になっている解答が非常に多くありました。その他には、目標金額に達したかどうかの判定で、「以上」とすべきところを「より大きい」と判定している間違いも多くありました。

解答例

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

main(){
    int L, M, N;
    while( cin >> L && L ){
        int t = 0, S = 0;
        for ( int i = 1; i <= 12; i++ ){
            cin >> M >> N;
            S += M - N;
            if ( S >= L && t == 0 ) t = i;
        }
        if ( S < L ) cout << "NA" << endl;
        else cout << t << endl;
    }
}
```

問題 03　ブロック

問題のポイント

2 次元グリッド上のマスを体系的に訪問するアルゴリズム、つまりグラフの探索アルゴリズムを実装できるかが問われています。

問題の解き方

この問題は深さ優先探索または幅優先探索によって解くことができます。以下に示す解答プログラムでは深さ優先探索を実装しています。深さ優先探索は、一度訪問したマスを二度訪問しないように注意しながら、再帰的に隣接するマスを訪問していくアルゴリズムです。プログラムの関数 dfs に示すように、グリッドにおける探索の問題では、現在地からの x 方向及び y 方向への移動距離を示す dx[4]、dy[4]を宣言しておくと、マスの間の移動を簡潔に記述することができます。

講評

提出数：42、正解数：8

不正解解答のうち、スタート位置にブロックが無いときの処理を考慮していないために不正解となっている解答が最も多かったです。幅優先探索はしつつも、ゴール条件が間違っているために不正解となっている解答も目立ちました。

解答例

```
#include<iostream>
using namespace std;
#define MAX 100
static const int dx[4] = {1, 0, -1, 0};
static const int dy[4] = {0, -1, 0, 1};

int w, h, sx, sy, gx, gy, G[MAX+2][MAX+2];

bool dfs( int x, int y, int color ){
   if ( color == 0 || G[x][y] != color ) return false;
   if ( x == gx && y == gy ) return true;
   G[x][y] = 0;
   for ( int r = 0; r < 4; r++ ){
       if ( dfs(x + dx[r], y + dy[r], color) ) return true;
   }
   return false;
}

main(){
   int n, c, d, px, py, ww, hh;
   while( cin >> w >> h && w ){
       cin >> sx >> sy >> gx >> gy >> n;
       for ( int y = 0; y < h+2; y++ )
```

```
            for ( int x = 0; x < w+2; x++ ) G[x][y] = 0;
        for ( int i = 1; i <=n; i++ ){
            cin >> c >> d >> px >> py;
            if ( d == 0 ) { ww = 4; hh = 2; }
            else { ww = 2; hh = 4; }
            for ( int y = 0; y <= hh-1; y++ )
                for ( int x = 0; x <= ww-1; x++ ) G[px+x][py+y] = c;
        }
        if ( dfs(sx, sy, G[sx][sy]) ) cout << "OK" << endl;
        else cout << "NG" << endl;
    }
}
```

問題 04　病院の部屋番号

問題のポイント

プログラミング能力に加え、数学的な思考、あるいは柔軟な対応力によって、アルゴリズムを導きだす能力が問われています。

問題の解き方

n の値が非常に大きいため、数列を生成して解くことはできません。4 と 6 の 2 つの数字を用いてはいけないので、10 – 2 = 8 進数で n を表せばよいことになります。ただし、基数変換後の数の各桁の数字を d とすると、d が 0～3 の場合はそれぞれ 0～3、d が 4 の場合は d+1(4 の分)= 5、d が 5 以上の場合は d+2(4 と 6 の分)に変換して出力します。文字列”01235789”の d 番目の文字を返す関数を定義しても良いでしょう。

講評

提出数：23、正解数：8

不正解の解答で一番多かったのは 1 から順に 4 または 6 を含まない部屋番号を数えていく方法の解答ですが、この方法では処理に時間がかかりすぎて時間制限エラーになってしまい不正解となります。

解答例

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

int getValue(int x){
   if ( x == 4 ) return 5;
   return (x < 4) ? x : x+2;
}

void convert( int n ){
   if ( n == 0 ) return;
   convert( n/8 );
   cout << getValue(n%8);
}

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

問題 05　写真に写っている景色は？

問題のポイント

多重ループ構造やグリッドの回転など、やや複雑な制御構造の実装と２次元配列のインデックス操作が行えるかが問われています。

問題の解き方

m×m のグリッドを回転しながら､全ての場所について一致しているかを調べていきます。n×n のグリッドにおける m×m の領域について、上から下、左から右の順番で調べていきます。各領域について、m×m のグリッドを回転させ、重なった部分のマスが全て一致するか（問題の仕様上-1 は無視します）をチェックします。

また、この解答例では、典型的な繰り返し処理を簡潔に記述するために、for ループを rep として define しています。一般的なプログラミングにはあまり用いられませんが、競技プログラミングにおいてはコードの簡略化、バグ埋め込みの回避、可読性向上のためのテクニックとして使われる場合があります。

講評

提出数：26、正解数：3

不正解の解答では、写真をすべての方向に回転できていない解答、-1 が縦または横 1 列に並んだ場合を考慮しようとして（問題の仕様上は考慮する必要なし）ミスをしている解答が目立ちました。問題を注意深く読むことで防げる間違いもありますので落ち着いて問題を読み、正確に読み取る練習も大切です。

解答例

```
#include<iostream>
using namespace std;
#define rep(i, n) for ( int i = 0; i < (int)n; i++ )
#define NMAX 200
#define MMAX 100
#define INFTY (1<<21)
int n, m, T[NMAX][NMAX], P[MMAX][MMAX];

void valid(int sx, int sy, int &lx, int &ly ){
   int ax = -1, ay;
   rep(y, m) rep(x, m){
        if ( P[x][y] == -1 ) continue;
        if ( P[x][y] != T[sx+x][sy+y] ) return;
        if ( ax == -1 ) {ay = y; ax = x;}
   }
   if ( ay < ly || (ay == ly && ax < lx) ) {ly = sy+ay; lx = sx+ax; }
}
```

```
bool rotate(int sx, int sy){
    int tmp[MMAX][MMAX];
    int ly = INFTY, lx;
    rep(r, 4){
        valid(sx, sy, lx, ly);
        rep(y, m) rep(x, m) tmp[x][y] = P[x][y];
        rep(y, m) rep(x, m) P[y][m-x-1] = tmp[x][y];
    }
    if ( ly == INFTY ) return false;
    cout << lx+1 << " " << ly+1 << endl;
    return true;
}

void compute(){
    rep(y, n-m+1) rep(x, n-m+1) if ( rotate(x, y) ) return;
    cout << "NA" << endl;
}

main(){
    while( cin >> n >> m && n ){
        rep(y, n) rep(x, n) cin >> T[x][y];
        rep(y, m) rep(x, m) cin >> P[x][y];
        compute();
    }
}
```

問題 06　ザ・スクエアーズ

問題のポイント

問題文に定義されたやや複雑な仕様を理解し、正しくシミュレーションを行うプログラムを素早く実装できるかがポイントです。

問題の解き方

問題文に書かれている手順でアルゴリズムを実装します。最初のステップで、各人についてその人の方向を定義されたルールに従い決定します。次のステップにて、各人について現在の方向に動けるか否かを定義されたルールに従い決定し、動ける場合は移動します。

講評

提出数：10、正解数：4

不正解となった解答のうち、細かなミスによる不正解が多く、複雑な仕様を丹念に実装する練習も大切です。

解答例

```
#include<iostream>
#include<string>
using namespace std;
#define rep(i,n) for ( int i = 0; i < n; i++ )
#define REP(i, b, e) for ( int i = b; i <= e; i++ )
#define MAX 100
#define LIMIT 180
#define BLOCK '#'
#define SPACE '.'
#define EXIT 'X'
static const string DIR = "ENWS";
static const int di[4] = {0, -1, 0, 1};
static const int dj[4] = {1, 0, -1, 0};
int H, W;
char G[MAX+2][MAX+2];

bool isPerson(char ch){
   rep(d, 4) if ( ch == DIR[d] ) return true;
   return false;
}

int getDirection(char ch){
   rep(d, 4) if ( ch == DIR[d] ) return d;
}

int chengeDirection(){
   int nperson = 0;
   REP(i, 1, H) REP(j, 1, W){
       if ( !isPerson(G[i][j]) ) continue;
       nperson++;
```

```
        int curd = getDirection(G[i][j]);
        int v = curd;
        curd = ((curd == 0) ? 3 : curd-1);
        for ( int r = curd; r < curd + 4; r++ ){
            char ch = G[i+di[r%4]][j+dj[r%4]];
            if ( ch == SPACE || ch == EXIT ){
                v = r%4;
                break;
            }
        }
        G[i][j] = DIR[v];
    }
    return nperson;
}

void move(){
    int dir[MAX+2][MAX+2];
    rep(i, H+2) rep(j, W+2 ) dir[i][j] = -1;

    REP(i, 1, H) REP(j, 1, W){
        if ( G[i][j] == SPACE || G[i][j] == EXIT){
            for ( int r = 0; r < 4; r++ ){
                char target = G[i+di[r]][j+dj[r]];
                if ( isPerson(target) && target == DIR[(r+2)%4] ){
                    dir[i+di[r]][j+dj[r]] = (r+2)%4;
                    break;
                }
            }
        }
    }

    REP(i, 1, H) REP(j, 1, W){
        if ( !isPerson(G[i][j]) || dir[i][j] == -1) continue;
        int ni, nj;
        ni = i + di[dir[i][j]];
        nj = j + dj[dir[i][j]];

        if ( G[ni][nj] == EXIT ){
            G[i][j] = SPACE;
        } else if ( G[ni][nj] == SPACE ){
            G[ni][nj] = G[i][j];
            G[i][j] = SPACE;
        }
    }
}

int main(){
    while(1){
        cin >> W >> H;
        if ( H == 0 && W == 0 ) break;
        rep(i,H+2) rep(j,W+2) G[i][j] = BLOCK;
        REP(i, 1, H) REP(j, 1, W) cin >> G[i][j];
        int t = 0;
        while(1){
            if ( t > LIMIT ) break;
            if( !chengeDirection() ) break;
            move();
            t++;
        }
```

```
            if ( t > LIMIT ) cout << "NA" << endl;
            else cout << t <<  endl;
        }

        return 0;
    }
```

問題 07　みんなでジョギング

問題のポイント

整数に関するアルゴリズムの知識が有効な問題です。

問題の解き方

n 人をそれぞれ 0〜n-1 の番号 i で表し、それぞれのコースの 1 周を di(km)、走る速度を vi(km/h)で表すことにします。まず、計算途中でのオーバーフローを避けるために、それぞれの vi/di を約分します。次に vi/di (i = 0 〜 n-1)を通分します。全ての分母 di に対する最小公倍数 LCM を分母とすると、それぞれの分子は vi×LCM/di となります。最後に全ての分子をそれらの最大公約数で割ると、最初に全員が出会う周回数となります。最大公約数を求めるためにはユークリッドの互除法を用います。

講評

提出数：9、正解数：3

解答例

```
#include<iostream>
#include<algorithm>
using namespace std;
typedef unsigned long long ullong;
#define MAX 10

ullong gcd(ullong a, ullong b){
   return ( b == 0 ) ? a : gcd(b, a%b);
}

ullong lcm( ullong a, ullong b ){
   if ( b > a ) swap(a, b);
   return a/gcd(a, b)*b;
}

void compute(int n){
   ullong d1, d2, v1, v2, G;
   pair<ullong, ullong> S[MAX];
   for ( int i = 0; i < n ; i++ ){
        cin >> S[i].first >> S[i].second;
        G = gcd(S[i].first, S[i].second);
        S[i].first /= G;
        S[i].second /= G;
   }

   ullong l = lcm(S[0].first, S[1].first);
   for ( int i = 2; i < n; i++ ) l = lcm(l, S[i].first);

   ullong A[MAX];
   for ( int i = 0; i < n; i++ ) A[i] = S[i].second*(l/S[i].first);
```

```
    G = gcd(A[0], A[1]);
    for ( int i = 2; i < n; i++ ) G = gcd(G, A[i]);
    for ( int i = 0; i < n; i++ ) cout << A[i]/G << endl;
}

main(){
    int n;
    while( cin >> n && n ) compute(n);
}
```

問題 08　高速バス

問題のポイント

グラフの最短経路を求めるダイクストラのアルゴリズムを応用して解くことができます。

問題の解き方

基本的にはダイクストラのアルゴリズムで解きます。ただし、このダイクストラのアルゴリズムにおけるグラフのノードは、与えられたグラフのノード i（バス停）と残りのチケットの枚数 k の組になります。つまり、２次元配列 D[i][k]に「k 枚のチケットを持ちバス停 i にいるスタート地点からの最小コスト」をダイクストラのアルゴリズムで記録します。

講評

提出数：21、正解数：3

不正解の解答には、グラフを作る際にチケットの残り枚数を記録していない、または最短経路探索のアルゴリズムに間違いがある解答が多くありました。

解答例

```
#include<iostream>
using namespace std;
#define rep(i, n) for ( int i = 0; i < n; i++ )
#define MAX 100
#define INFTY (1<<21)
int G[MAX][MAX], c, n, s, d;

int dijkstra(){
   int D[MAX][11];
   bool V[MAX][11];
   rep(i, n) rep(k, c+1) { D[i][k] = INFTY; V[i][k] = false; }
   D[s][c] = 0;
   while(1){
       int u, w, minv = INFTY;
       rep(i, n) rep(k, c+1){
          if ( !V[i][k] && minv > D[i][k] ){
              minv = D[i][k]; u = i; w = k;
          }
       }
       if ( minv == INFTY ) break;
       V[u][w] = true;
       rep(v, n){
          if ( G[u][v] == INFTY ) continue;
          D[v][w] = min(D[v][w], D[u][w] + G[u][v]);
          if ( w ) D[v][w-1] = min(D[v][w-1], D[u][w] + G[u][v]/2);
       }
   }
   int minv = INFTY;
   rep(k, c+1) minv = min(minv, D[d][k]);
   return minv;
```

```
}

main(){
    int m, a, b, f;
    while( cin >> c >> n >> m >> s >> d && c){
        s--; d--;
        rep(i, n) rep(j, n) G[i][j] = INFTY;
        rep(i, m){
            cin >> a >> b >> f; a--; b--;
            G[a][b] = G[b][a] = f;
        }
        cout << dijkstra() << endl;
    }
}
```

問題 09　土地分割

問題のポイント

パズルを解くように、バックトラックをしながら探索を行うプログラムを実装することが必要です。

問題の解き方

全ての区画について順番に、要求された大きさの長方形で分譲地を埋めていきます。各区画の長方形の大きさと位置については、すでに決定した他の区画と重ならず、かつ分譲地からはみ出さないような全てのパターンについて調べます。この処理は、再帰関数によって実装します。全ての区画について長方形が配置でき、かつ全てのマスが 0 以外で埋まっている場合は解としてカウントし、そうでない場合はバックトラックでさらに探索を行います。

講評

提出数：3、正解数：0

解答例

```
#include<iostream>
#include<cassert>
using namespace std;
#define rep(i, n) for ( int i = 0; i < n; i++ )
#define MAX 10
struct Cell{ int i, j, x, id; };

int H, W, n, b, k, ans, V[MAX+1];
Cell P[MAX*MAX], M[MAX][MAX], A[MAX][MAX];

bool placeable(int si, int sj, int r, int c ){
   rep(i, r) rep(j, c){
        int ii = si + i;
        int jj = sj + j;
        if ( ii < 0 || jj < 0 || ii >= H || jj >= W ) return false;
        if ( M[ii][jj].x ) return false;
   }
   return true;
}

void solve(int pos){
   if ( pos >= n ){
        rep(i, H) rep(j, W) if ( M[i][j].x == 0 ) return;
        rep(i, H) rep(j, W) A[i][j] = M[i][j];
        ans++;
        return;
   }
```

```
    for(int r = 1; r <= P[pos].x; r++){
        if ( P[pos].x % r != 0 ) continue;
        int c = P[pos].x / r;
        rep(di, r) rep(dj, c){
            int si = P[pos].i - di;
            int sj = P[pos].j - dj;
            if ( placeable( si, sj,  r, c ) ){
                rep(i, r) rep(j, c) M[si+i][sj+j] = P[pos];
                solve( pos + 1 );
                rep(i, r) rep(j, c) M[si+i][sj+j].x = 0;
            }
        }
    }
}

int main(){
    int x;
    while( cin >> W >> H && !(H == 0 && W == 0) ){
        cin >> n;
        rep(i, n){
            cin >> b >> k;
            P[b-1].x = k;
        }

        rep(i, H) rep(j, W){
            cin >> x;
            if ( x ){  P[x-1].i = i; P[x-1].j = j; P[x-1].id = x-1; }
            M[i][j].x = 0;
        }
        ans = 0;
        solve(0);
        if ( ans == 1 ) {
            rep(i, H){
                rep(j, W){
                    if ( j ) cout << " ";
                    cout << A[i][j].id+1;
                }
                cout << endl;
            }
        } else {
            cout << "NA" << endl;
        }
    }
    return 0;
}
```

問題 10　秋のイルミネーション

問題のポイント

基本的な計算幾何学アルゴリズムの実装とグラフの連結成分分解の実装ができるかが問われています。幾何学オブジェクトに対しての基本的な操作を行うライブラリ（プログラムの集まり）を事前準備しておけば、簡単に解くことができる問題です。

問題の解き方

２つの四角形が接触しているかどうかを判定するプログラムを使ってグラフをつくり、連結成分の数を深さ優先探索で求めます。四角形Ａのいずれかの辺と四角形Ｂのいずれかの辺が接触している、四角形Ａが四角形Ｂの頂点を含む、または四角形Ｂが四角形Ａの頂点を含む場合、四角形Ａと四角形Ｂが同じグループに属すると判定することができます。

講評

提出数：11、正解数：2

不正解の解答の多くは、四角形の中に別の四角形が完全に入っている場合の判定を間違えていました。

解答例

```
#include<iostream>
#include<cfloat>
#include<cmath>
#include<vector>
using namespace std;
#define rep(i, n) for ( int i = 0; i < (int)n; i++ )
#define EPS (1e-8)

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); }
};
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 pol, Point p){
   rep(i, 4){
       if ( ccw(pol[i], pol[(i+1)%4], p ) == COUNTER_CLOCKWISE ) return
false;
   }
   return true;
}

bool isIntersectSQ(Polygon p1, Polygon p2 ){
   rep(i, 3) rep(j, 3){
       if ( isIntersect(p1[i], p1[i+1], p2[j], p2[j+1] ) ) return true;
   }
   rep(i, 4) if(isInside(p1, p2[i])) return true;
   rep(i, 4) if(isInside(p2, p1[i])) return true;
   return false;
}

Polygon P[100];
int n, G[100][100];
bool V[100];

void dfs( int u ){
   V[u] = true;
   rep(v, n) if ( G[u][v] && !V[v] ) dfs(v);
}

int compute(){
   rep(i, n) rep(j, n ) G[i][j] = 0;
   rep(i, n) rep(j, n ) if ( isIntersectSQ(P[i], P[j]) ) G[i][j] = 1;
   rep(i, n) V[i] = false;
   int ncomponent = 0;
   rep(i, n){
       if ( V[i] ) continue;
       ncomponent++;
       dfs(i);
   }
   return ncomponent;
}

int main(){
   int N;
```

```
    double x, y;
    while( cin >> N && N ){
        rep(i, N){
            cin >> n;
            rep(j, n){
                Polygon pol;
                rep(k, 4){
                    cin >> x >> y;
                    pol.push_back(Point(x, y));
                }
                P[j] = pol;
            }
            cout << compute() << endl;
        }
    }
    return 0;
}
```

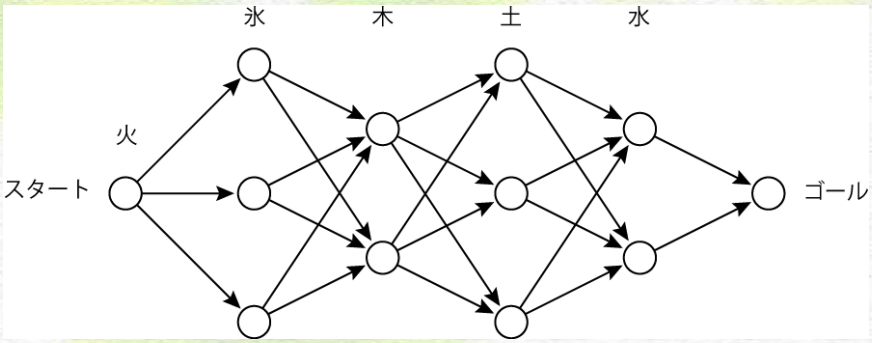
問題 11　パチモンクリーチャー

問題のポイント

入力の上限を考慮して計算効率を見積もり、効率の良いアルゴリズムを設計する能力が問われています。ここでは、典型的な幅優先探索またはダイクストラのアルゴリズムではなく、動的計画法で実装するところがポイントです。

問題の解き方

最初に 1 つのパチクリを選んだ後のパチクリを捕まえる順番は、上記属性の関連の順番になります。例えば最初に火の属性をもつパチクリを持っていれば、氷、木、土、水の属性をもつパチクリを順番に捕まえてゴールに行けばよいので、下図に示す DAG（Directed Acyclic Graph）の最短コストを求めることになります。下図は、3 体の氷のパチクリ、2 体の木のパチクリ、3 体の土のパチクリ、2 体の水のパチクリを含むグリッドを DAG で表したものです。捕まえられないパチクリがいるマスを通っても現在保持しているパチクリの状態に影響はないので、ノード間の距離は対応するマス間のマンハッタン距離（x 座標の差と y 座標の差の和）となります。スタート地点で選ぶパチクリを変えて（5 種類分ありますが、下図では最初に火のパチクリを選んでいます)、動的計画法によってゴールまでの最短コストを求めます。



講評

提出数：20、正解数：2

不正解の解答の多くは、グリッド上でのダイクストラのアルゴリズムまたは深さ優先探索で解こうとして失敗している解答が多くありました。

解答例

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

#define rep(i, n) for (int i = 0; i < n; i++ )
#define MAX 1000
```

```
#define INFTY (1<<21)
struct Point{ int x, y; };
int H, W, sy, sx, gy, gx, cnt[5];
Point P[5][MAX];
int D[5][MAX];

inline int mdist(int x1, int y1, int x2, int y2){
   return max(x1, x2) - min(x1, x2) + max(y1, y2) - min(y1, y2);
}

void compute(){
   int opts, optc = INFTY;
   int t;
   rep(s, 5){
      rep(i, 5) rep(j, cnt[i]) D[i][j] = INFTY;
      t = (s+1)%5;
      rep(j, cnt[t]) D[t][j] = mdist(sx, sy, P[t][j].x, P[t][j].y );
      for ( int i = s+1; i < s + 5; i++ ){
         int b = (i)%5;
         int e = (i+1)%5;
         rep(k, cnt[b]) rep(l, cnt[e]) {
            D[e][l] = min(D[e][l], D[b][k] + mdist(P[b][k].x, P[b][k].y,
P[e][l].x, P[e][l].y));
         }
      }
       t = (s+4)%5;
      rep(j, cnt[t] ){
         if ( D[t][j] + mdist(gx, gy, P[t][j].x, P[t][j].y ) < optc ){
            optc = D[t][j] + mdist(gx, gy, P[t][j].x, P[t][j].y);
            opts = s;
         }
      }
   }
   if ( optc == INFTY ) cout << "NA" << endl;
   else cout << opts+1 << " " << optc << endl;
}

main(){
   char ch;
   while(cin >> W >> H && W){
      rep(i, 5) cnt[i] = 0;
      rep(y, H) rep(x, W) {
          cin >> ch;
          if ( ch == '.' ) continue;
          if ( ch == 'S' ){
              sy = y; sx = x;
          } else if ( ch == 'G' ){
              gy = y; gx = x;
          } else {
              int p = ch -'0' - 1;
              P[p][cnt[p]].x = x;
              P[p][cnt[p]++].y = y;
          }
      }
      compute();
   }
}
```