ICPC2011 国内予選問題 C　同色パネル結合

アルゴリズム

- 可能な５回の色変更の組合せを深さ優先探索で全てチェックする。
- 変更後の色 tc を決めたら、change_color 関数で左上の結合パネルの色を tc に変更する。change_color 関数は、指定位置のパネルの色を tc に変更し、そのパネルに隣接する(上下左右の４方向)パネルが結合パネル上のパネルであれば、それらのパネル位置に対して change_color 関数を再帰的に呼び出す。(隣接領域を深さ優先探索して色を変更していることになる) また、change_color 関数は、実際に色 tc に変更したパネルの枚数を返す。
- ５回の色変更で目標の色になったら左上の結合パネルの大きさを求め、必要に応じて結合パネルの大きさ(size)の最大値を更新すれば良い。なお、５回の色変更で目標の色になった結合パネルの大きさは、tc = 0[1] でchange_color関数を再度呼び出すことで求められる。

[プログラム例]

```
// ICPC 2011 国内予選問題 C
// http://icpc2011.ait.kyushu-u.ac.jp/icpc2011/contest/all_ja.html#section_C
//          Filename = pc.c
//          Compile:    cc pc.c
//          Execution:  ./a.out < C0 > C0.result
//          Check:      diff C0.ans C0.result
//
// 深さ優先探索で全ての色の変更の可能性をチェックし、結合パネルの大きさの最大値を求める

#include <stdio.h>
#define MAXH 8                  // 長方形の高さの最大値
#define MAXW 8                  // 長方形の幅の最大値
#define MAXC 6                  // 色数の最大値

int h;                          // 長方形の高さ 1 <= h <= 8
int w;                          // 長方形の幅   1 <= w <= 8
int c;                          // 最終的に結合されるパネルの目標となる色 1 <= c <= 6
int size;                       // 左上の結合パネルの大きさの最大値

int dx[ ] = {1, 0, -1, 0};      // 隣接位置の x 座標のオフセット(右上左下)
int dy[ ] = {0, 1, 0, -1};      // 隣接位置の y 座標のオフセット(右上左下)
```

[1] この時点でのパネルの色は 1～6 であり、色 0 のパネルは存在しない。

```
// map の位置 (x,y) のパネルの色を tc に変更し、(x,y) の隣接位置の色が ltc なら、
// その隣接位置に対して再帰的に change_color を適用する。
// 関数の戻り値は、色 tc に変更したパネルの枚数
int change_color(int map[MAXH][MAXW], int x, int y, int ltc, int tc)
{
   int nx, ny, d, count;
   map[y][x] = tc;                    // まず、位置(x,y) のパネルの色を tc に変更
   count = 1;                         // 色を変更したパネルの枚数 count は 1
   for(d=0; d<4; d++){                // 各隣接位置に対して
      nx = x+dx[d];                   // 隣接位置の x 座標
      ny = y+dy[d];                   // 隣接位置の y 座標
      if(nx<0 || nx>=w || ny<0 || ny>=h) continue;          // マップの長方形の外は無視
      if(map[ny][nx] != ltc) continue;                      // 左上のパネルの色と異なる時も無視
      count += change_color(map, nx, ny, ltc, tc);          // 隣接位置を再帰的にチェック
   }
   return count;                      // 色を tc に変更したパネル枚数を返す
}

// 深さ優先探索で、可能な5回の色の変更全てをチェックする
// 左上のパネル以外の色に変更可能。なお、5回目は最終目標の色に変更
// 5回変更後に、左上の結合パネルの大きさを求める
// level = 今回の変更が何回目の変更かを表す
void dfs(int map[MAXH][MAXW], int level)
{
   int i, j, tc, count;
   int nmap[MAXH][MAXW];              // 色を変更した後のマップ
   for(tc=1; tc<=6; tc++){            // 各色 tc に対して
      if(map[0][0]==tc) continue;     // 左上のパネルと同じ色に変更する必要は無い
      if(level == 5 && tc != c) continue;   // 5回目は最終目標の色に変更
      for(i=0; i<h; i++)
         for(j=0; j<w; j++)
            nmap[i][j] = map[i][j];   // map を nmap にコピー
      change_color(nmap, 0, 0, nmap[0][0], tc); // 左上の結合パネルの色を tc に変更
      if(level < 5) dfs(nmap, level+1);         // 変更回数が5回未満なら、さらに色を変更
      else {  // 5回変更を行ったので、左上の結合パネルの色を 0 に変えることで
         count = change_color(nmap, 0, 0, nmap[0][0], 0);   // その大きさを求める
         if(count > size) size=count; // 今の大きさがこれまでのものよりも大きければ size を更新
      }
   }
}

int main()
{
   int i, j;
   int map[MAXH][MAXW];               // パネルのマップ
   while(1){
      scanf("%d %d %d", &h, &w, &c);  // h, w, c の入力
      if(h==0 && w==0 && c==0) break; // 読み込んだ値が全て0なら終了
      for(i=0; i<h; i++)
         for(j=0; j<w; j++)
```

```
            scanf("%d", &map[i][j]);                // マップデータの読み込み
        size = 0;                                   // 求めるサイズを初期化
        dfs(map, 1);                        // 深さ優先探索で可能な全ての色の変更をチェック
        printf("%d¥n", size);               // 求まった最大の大きさを出力
    }
}
```