TIMUS 1301 Cube in Labyrinth Solution

write by Gestalt Lur

2012-07-15

题目链接 http://acm.timus.ru/problem.aspx?space=1&num=1301

题目大意

在一个 X*Y 的网格上有一个与网格等宽的方块。给定开始和结束的位置，方块要从开始位置滚动到结束位置，并且在一些网格之间因为有障碍而不可达。要求在结束位置时方块向上的一面必须和开始时一样。求从开始到结束的最短路径的长度。

算法分析

搜索即可，方块的朝向可以预先定义其向四个方向翻转时各个面的变化情况。

参考代码

```
/*
 TIMUS 1301
 2012-07-10
 ACCEPTED
 gestapolur
*/
#include<cstdlib>
#include<cstdio>
#define MAXN 11
#define INF 1 << 30

const int rol[ 4 ][ 7 ] = { { 0 , 6 , 2 , 5 , 4 , 1 , 3 } ,
                            { 0 , 5 , 2 , 6 , 4 , 3 , 1 } ,
                            { 0 , 2 , 3 , 4 , 1 , 5 , 6 } ,
                            { 0 , 4 , 1 , 2 , 3 , 5 , 6 } };

int n , m;
int s1 , s2 , e1 , e2;
bool lnk[ MAXN ][ MAXN ][ MAXN ][ MAXN ];
int d[ MAXN ][ MAXN ][ 7 ];

void init()
{
 int i , j , k;
 int t1 , t2;
 char ch;
 scanf( "%d%d%d%d%d%d%c" , &n , &m , &s1 , &s2 , &e1 , &e2 , &ch );
 ch = getchar();

 for( i = 1 ; i <= n ; ++ i )
  for( j = 1 ; j <= m ; ++ j )
   for( k = 1 ; k <= 6 ; ++ k )
    d[ i ][ j ][ k ] = INF;
 d[ s1 ][ s2 ][ 1 ] = 0;

 while( scanf( "%d" , &t1 ) )
```

```
   {
    if( t1 == int( 'h' ) )
     break;
    scanf( "%d" , &t2 );
    lnk[ t1 ][ t2 ][ t1 + 1 ][ t2 ] = true;
    lnk[ t1 + 1 ][ t2 ][ t1 ][ t2 ] = true;
   }

 getchar();
 while( scanf("%d%d" , &t1 , &t2 ) not_eq EOF )
   {
    lnk[ t1 ][ t2 ][ t1 ][ t2 + 1 ] = true;
    lnk[ t1 ][ t2 + 1 ][ t1 ][ t2 ] = true;
   }
 return ;
}

void dfs( int p1 , int p2 , int ups )
{
 //down
 if( p1 + 1 <= n and not lnk[ p1 ][ p2 ][ p1 + 1 ][ p2 ]
    and d[ p1 + 1 ][ p2 ][ rol[ 0 ][ ups ] ] > d[ p1 ][ p2 ][ ups ] + 1 )
   {
    d[ p1 + 1 ][ p2 ][ rol[ 0 ][ ups ] ] = d[ p1 ][ p2 ][ ups ] + 1;
    dfs( p1 + 1 , p2 , rol[ 0 ][ ups ] );
   }
 //up
 if( p1 - 1 > 0 and not lnk[ p1 ][ p2 ][ p1 - 1 ][ p2 ]
    and d[ p1 - 1 ][ p2 ][ rol[ 1 ][ ups ] ] > d[ p1 ][ p2 ][ ups ] + 1 )
   {
    d[ p1 - 1 ][ p2 ][ rol[ 1 ][ ups ] ] = d[ p1 ][ p2 ][ ups ] + 1;
    dfs( p1 - 1 , p2 , rol[ 1 ][ ups ] );
   }
 //right
 if( p2 + 1 <= m and not lnk[ p1 ][ p2 ][ p1 ][ p2 + 1 ]
    and d[ p1 ][ p2 + 1 ][ rol[ 2 ][ ups ] ] > d[ p1 ][ p2 ][ ups ] + 1 )
   {
    d[ p1 ][ p2 + 1 ][ rol[ 2 ][ ups ] ] = d[ p1 ][ p2 ][ ups ] + 1;
    dfs( p1 , p2 + 1 , rol[ 2 ][ ups ] );
   }
 //left
 if( p2 - 1 > 0 and not lnk[ p1 ][ p2 ][ p1 ][ p2 - 1 ]
    and d[ p1 ][ p2 - 1 ][ rol[ 3 ][ ups ] ] > d[ p1 ][ p2 ][ ups ] + 1 )
   {
    d[ p1 ][ p2 - 1 ][ rol[ 3 ][ ups ] ] = d[ p1 ][ p2 ][ ups ] + 1;
    dfs( p1 , p2 - 1 , rol[ 3 ][ ups ] );
   }
```

```
  return ;
}

int main()
{
  init();
  dfs( s1 , s2 , 1 );
  if( d[ e1 ][ e2 ][ 1 ] not_eq INF )
    printf( "%d\n" , d[ e1 ][ e2 ][ 1 ] );
  else
    printf( "No solution\n" );
  return 0;
}
```