2010 Hangzhou Regional Online Problem G Tetris Solution

write by Gestalti Lur
2012-09-06
原题：http://acm.hdu.edu.cn/showproblem.php?pid=3647

题目大意

问用给定顺序下落的俄罗斯方块能否拼成 N * M(N*M=40)的矩形。方块可以旋转，并且下落的时候不能受到已有方块的阻碍。

算法分析

用 DFS 即可，搜索状态记录当前每一列上方块的高度(数量)。然后扫描判断连续的一段上方块的高度是否能和落下的方块匹配即可。注意 input 中首先给出矩形宽度其次是其高度。

另外注意这样的数据（OnlineJudge 上似乎没有）：

40 1
I I I I I I I I I J

参考代码

```
/*
 2010 HANGZHOU ONLINE G
 gestapolur
 2012-09-06
 ACCEPTED
*/
#include<cstdio>
#include<cstring>

int n , m;
int r[ 42 ];
char t[ 11 ];

inline bool insert( int p , char x , int rr )
{
 switch ( x )
  {
  case 'I' :
   if( rr == 0 and p + 3 <= m and r[ p ] + 1 <= n and r[ p ] == r[ p + 1 ] and r[ p ] == r[ p + 2 ] and r[ p ] == r[ p + 3 ] )
    { ++ r[ p ]; ++ r[ p + 1 ]; ++ r[ p + 2 ]; ++ r[ p + 3 ]; return true; }
   if( rr == 1 and r[ p ] + 4 <= n ) { r[ p ] += 4; return true;}
   return false;
  case 'J' :
```

```
    if( rr == 0 and p + 2 <= m and r[ p ] + 2 <= n and r[ p ] == r[ p + 1 ] and r[
p ] == r[ p + 2 ] )
    { r[ p ] += 2; ++ r[ p + 1 ]; ++ r[ p + 2 ]; return true; }
    if( rr == 1 and p + 1 <= m and r[ p ] + 3 <= n and r[ p + 1 ] + 1 <= n and
r[ p ] + 2 == r[ p + 1 ] )
    { r[ p ] += 3; ++ r[ p + 1 ]; return true; }
    if( rr == 2 and p + 2 <= m and r[ p ] == r[ p + 1 ] and r[ p ] == r[ p + 2 ] +
1 and r[ p ] + 1 <= n and r[ p + 2 ] + 2 <= n)
    { ++ r[ p ]; ++ r[ p + 1 ]; r[ p + 2 ] += 2; return true; }
    if( rr == 3 and p + 1 <= m and r[ p + 1 ] + 3 <= n and r[ p ] == r[ p + 1 ] )
     { ++ r[ p ]; r[ p + 1 ] += 3; return true;}
    return false;
  case 'L' :
    if( rr == 0 and p + 2 <= m and r[ p + 2 ] + 2 <= n and r[ p ] == r[ p + 1 ]
and r[ p ] == r[ p + 2 ] )
    { ++ r[ p ]; ++ r[ p + 1 ]; r[ p + 2 ] += 2; return true; }
    if( rr == 1 and p + 1 <= m and r[ p ] + 3 <= n and r[ p + 1 ] <= n and r[ p ]
== r[ p + 1 ] )
    { r[ p ] += 3 ; ++ r[ p + 1 ]; return true; }
    if( rr == 2 and p + 2 <= m and r[ p ] + 2 <= n and r[ p + 1 ] + 1 <= n and
r[ p ] + 1 == r[ p + 1 ] and r[ p + 1 ] == r[ p + 2 ] )
    { r[ p ] += 2; ++ r[ p + 1 ]; ++ r[ p + 2 ]; return true;}
    if( rr == 3 and p + 1 <= m and r[ p + 1 ] + 3 <= n and r[ p + 1 ] + 2 ==
r[ p ] )
     { ++ r[ p ]; r[ p + 1 ] += 3; return true; }
    return false;
  case 'O' :
    if( rr == 0 and p + 1 <= m and r[ p ] + 2 <= n and r[ p ] == r[ p + 1 ] )
    { r[ p ] += 2 ; r[ p + 1 ] += 2 ; return true;}
    return false;
  case 'S' :
    if( rr == 0 and p + 2 <= m and r[ p + 1 ] + 2 <= n and r[ p + 2 ] + 1 <= n
and r[ p ] == r[ p + 1 ] and r[ p ] + 1 == r[ p + 2 ] )
    { ++ r[ p ]; r[ p + 1 ] += 2; ++ r[ p + 2 ]; return true;}
    if( rr == 1 and p + 1 <= m and r[ p ] + 2 <= n and r[ p + 1 ] + 2 <= n and
r[ p ] == r[ p + 1 ] + 1 )
    { r[ p ] += 2; r[ p + 1 ] += 2; return true;}
    return false;
  case 'T' :
    if( rr == 0 and p + 2 <= m and r[ p + 1 ] + 2 <= n and r[ p ] == r[ p + 1 ]
and r[ p ] == r[ p + 2 ])
    { ++ r[ p ]; r[ p + 1 ] += 2 ; ++ r[ p + 2 ]; return true; }
    if( rr == 1 and p + 1 <= m and r[ p ] + 3 <= n and r[ p + 1 ] + 1 <= n and
r[ p ] + 1 == r[ p + 1 ] )
    { r[ p ] += 3 ; ++ r[ p + 1 ]; return true; }
    if( rr == 2 and p + 1 <= m and r[ p ] + 1 <= n and r[ p + 1 ] + 3 <= n and
r[ p + 1 ] + 1 == r[ p ] )
    { ++ r[ p ]; r[ p + 1 ] += 3; return true; }
```

```
    if( rr == 3 and p + 2 <= m and r[ p ] + 1 <= n and r[ p + 1 ] + 2 <= n and
r[ p ] == r[ p + 2 ] and r[ p + 1 ] + 1 == r[ p ] )
     { ++ r[ p ]; r[ p + 1 ] += 2; ++ r[ p + 2 ]; return true;}
     return false;
   case 'Z' :
     if( rr == 0 and p + 2 <= m and r[ p + 1 ] + 2 <= n and r[ p + 1 ] == r[ p +
2 ] and r[ p ] + 1 <= n and r[ p ] == r[ p + 1 ] + 1 )
     { ++ r[ p ]; r[ p + 1 ] += 2 ; ++ r[ p + 2 ]; return true;}
     if( rr == 1 and p + 1 <= m and r[ p ] + 2 <= n and r[ p + 1 ] + 2 <= n and
r[ p ] + 1 == r[ p + 1 ] )
     { r[ p ] += 2 ; r[ p + 1 ] += 2; return true; }
     return false;
   }
}
/*
void test()
{
 for( int i = 1 ; i <= m ; ++ i ) printf("%d " , r[ i ] ); printf("\n");
 for( int i = n ; i > 0 ; -- i )
   {
    for( int j = 1 ; j <= m ; ++ j )
    printf( "%d" , i <= r[ j ] ? 1:0 );
    printf("\n");
   }
 return ;
}
*/
bool dfs( int x )
{
 int rec[ 42 ] , ro;
 memcpy( rec , r , sizeof( rec ) );
 for( int i = 1 ; i <= m ; ++ i )
   for( ro = 0 ; ro < 4 ; ++ ro )
    if( insert( i , t[ x ] , ro ) )
    {
      if( x == 10 or dfs( x + 1 ) )
        return true;
      memcpy( r , rec , sizeof( rec ) );
    }
 return false;
}

bool init()
{
 int i;
 char ch;
 scanf("%d %d" , &m , &n );
 if( n == 0 and m == 0 ) return false;
```

```
 getchar();
 for( i = 1 ; i <= 10 ; ++ i )
   {
    scanf("%c" , &t[ i ] );
    if( i < 10 ) getchar();
   }
 memset( r , 0  , sizeof( r ) );
 return true;
}

int main()
{
 while( init() )
  printf( dfs( 1 ) ? "Yes\n" : "No\n" );
 return 0;
}
```

Python 随机数据生成：

```
import random
for i in range( 0 , 10 ):
   s = True
   while s == True:
      j = random.randint( 1 , 41 )
      if 40 % j == 0:
         print j , 40 / j
         s = False
   for j in range( 0 , 10 ):
      print random.choice("IJLOSTZ"),
   print
print 0 , 0
```