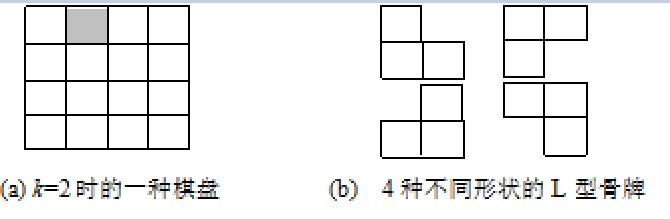
棋盘覆盖问题

在一个2^k×2^k （0<k<10）个方格组成的棋盘中，恰有一个方格与其他方格不同，称该方格为特殊方格。显然，特殊方格在棋盘中可能出现的位置有4^k种，因而有4^k种不同的棋盘，图(a)所示是k=2时16种棋盘中的一个。棋盘覆盖问题要求用如图所示的4种不同形状的L型骨牌覆盖给定棋盘上除特殊方格以外的所有方格，且任何2个L型骨牌不得重叠覆盖。

输入：

输入一个数（是2的k次幂），使棋盘由2^k×2^k个方格组成，

输出：

由骨牌编码覆盖的棋盘（数据间用制表符’\t’隔开）

示例输入：

4

示例输出：

2 0 3 3

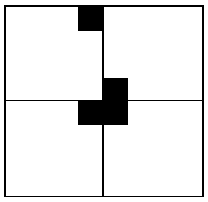
2 2 1 3

4 1 1 5

4 4 5 5

传说中的解题报告。。

首先，易知，在任何一个2^k\*2^k 的棋盘中，用到的 L 型骨牌个数恰为(4^k-1)/3 。当 k>0时，可以先将2^k\*2^k的棋盘分割为4个2^(k-1)\*2^(k-1)的子棋盘，则特殊方格必位于 4 个子棋盘之一中，其余3个子棋盘中无特殊方格。**根据分治法，该问题需要分解为若干个规模较小的相同问题再递归求解。**要将这 3 个无特殊方格的子棋盘转化为特殊棋盘，我们可以用一个L型骨牌覆盖这3个较小的棋盘的汇合处，如下图所示。这 3 个子棋盘上被L型骨牌覆盖的方格就成为该棋盘上的特殊方格，从而将原问题化为4个较小规模的棋盘覆盖问题。使用这种分割，直至棋盘简化为 1x1 棋盘（递归边界）。



代码如下：

#include<iostream>

using namespace std;

long long int tile=1; //L型骨牌的编号(递增)

long long int board[1000][1000]; //棋盘

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 递归方式实现棋盘覆盖算法

\* 输入参数：

\* tr--当前棋盘左上角的行号

\* tc--当前棋盘左上角的列号

\* dr--当前特殊方格所在的行号

\* dc--当前特殊方格所在的列号

\* size：当前棋盘的:2^k

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void chessBoard ( int tr, int tc, int dr, int dc, int size )

{

if ( size==1 ) //棋盘方格大小为1,说明递归到最里层

return;

long long int t=tile++; //每次递增1

int s=size/2; //棋盘中间的行、列号(相等的)

//检查特殊方块是否在左上角子棋盘中

if ( dr<tr+s && dc<tc+s ) //在

chessBoard ( tr, tc, dr, dc, s );

else //不在，将该子棋盘右下角的方块视为特殊方块

{

board[tr+s-1][tc+s-1]=t;

chessBoard ( tr, tc, tr+s-1, tc+s-1, s );

}

//检查特殊方块是否在右上角子棋盘中

if ( dr<tr+s && dc>=tc+s ) //在

chessBoard ( tr, tc+s, dr, dc, s );

else //不在，将该子棋盘左下角的方块视为特殊方块

{

board[tr+s-1][tc+s]=t;

chessBoard ( tr, tc+s, tr+s-1, tc+s, s );

}

//检查特殊方块是否在左下角子棋盘中

if ( dr>=tr+s && dc<tc+s ) //在

chessBoard ( tr+s, tc, dr, dc, s );

else //不在，将该子棋盘右上角的方块视为特殊方块

{

board[tr+s][tc+s-1]=t;

chessBoard ( tr+s, tc, tr+s, tc+s-1, s );

}

//检查特殊方块是否在右下角子棋盘中

if ( dr>=tr+s && dc>=tc+s ) //在

chessBoard ( tr+s, tc+s, dr, dc, s );

else //不在，将该子棋盘左上角的方块视为特殊方块

{

board[tr+s][tc+s]=t;

chessBoard ( tr+s, tc+s, tr+s, tc+s, s );

}

}

int main()

{

int size;

cout<<"输入棋盘的size(大小必须是2的n次幂): ";

cin>>size;

int x,y;

cout<<"输入特殊方格位置的坐标: ";

cin>>x>>y;

chessBoard ( 0,0,x,y,size);

for ( int i=0; i<size; i++ )

{

for ( int j=0; j<size; j++ )

cout<<board[i][j]<<"\t";

cout<<endl;

}

}