上海大学实验报告

专业：计算机科学
姓名：颜乐春
学号：15124542
日期：2017-10-09
地点：704

课程名称：算法设计於分析　指导老师：神人　成绩：59
实验名称：棋盘覆盖问题　实验类型：Null　同组学生姓名：None

一、 问题描述於实验目的

设 $n = 2^k (k > 0)$。在一个 $n \times n$ 个方格组成的棋盘中，恰有 1 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 4k 种，因而有 n2 种不同的棋盘，下图所示是 $k = 2$ $n = 4$ 时 16 种棋盘中的一个。

棋盘覆盖问题要求用下图所示的 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。

1. 输入

测试数据有若干行，每行 3 个整数 k x y，其中 $n = 2k$ 是棋盘的规模，$(x\ y)$ 是特殊方格的位置坐标，$(k > 1)$。

2. 输出

对输入中的每个正整数 k，第一行上先输出 “Case Num: n=”，接着输出 n 的值，其中 Num 是测试数据的序号，从 1 开始。第 2 到第 $n+1$ 行，输出对于规模为 $n = 2^k$ 的棋盘的一个覆盖。在该棋盘覆盖中，同一个骨牌用 3 个相同的数字表示。各骨牌表示的数字从 1 开始编号。特殊方格用 @ 表示。

3. 输入样例

```
1 2 1
2 2 3
```

4. 输出样列

```
Case 1: n=2
1 @
1 1
Case 2: n=4
1 1 2 2
1 5 # 2
3 5 5 4
```

```
3 3 4 4
```

二、 实验环境

Ubuntu 17.04 + gcc 6.3

三、 实验内容和步骤

1. 设计思路

当 $k > 0$ 时，将 2k* 2k 的棋盘分割为 4 个 2（k-1）*2（k-1）子棋盘。特殊方格必位于 4 个较小棋盘之一中，其余 3 个子棋盘中无特殊方格。为了将这 3 个无特殊方格的子棋盘化为特殊棋盘，我们用一个 L 型骨牌覆盖这 3 个较小的棋盘的回合处，这 3 个子棋盘被 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 个小规模的棋盘覆盖问题。递归地使用这种分割，直至棋盘化简为 1*1 棋盘。

2. 算法描述

四、 实现程序

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

const int BOARD_SZ = 8;
static int tile = 1;
static int board[BOARD_SZ][BOARD_SZ] = {0};

void chess_board(int tr, int tc, int dr, int dc, int size)
{
    if(size == 1)
        return;

    int t = tile++; //tile means 瓦片，基石，覆盖的步骤
    int sz = size / 2; //每次进行划分

    //cover top left corner
    if(dr < tr+sz && dc < tc+sz) //notice < < //注意一共四种情况，<>=这几个符号要控制好边界
        chess_board(tr, tc, dr, dc, sz);
    else{
        board[tr+sz-1][tc+sz-1] = t;
        chess_board(tr, tc, tr+sz-1, tc+sz-1, sz);
    }

    //cover top right corner
    if(dr < tr+sz && dc >= tc+sz) //notice < >=
        chess_board(tr, tc+sz, dr, dc, sz);
```

```
    else{
        board[tr+sz-1][tc+sz] = t;
        chess_board(tr, tc+sz, tr+sz-1, tc+sz, sz);
    }

    //cover lower left corner
    if(dr >= tr+sz && dc < tc+sz) //notice >= <
        chess_board(tr+sz, tc, dr, dc, sz);
    else{
        board[tr+sz][tc+sz-1] = t;
        chess_board(tr+sz, tc, tr+sz, tc+sz-1, sz);
    }

    //cover lower right corner
    if(dr >= tr+sz && dc >= tc+sz) //notice >= >=
        chess_board(tr+sz, tc+sz, dr, dc, sz);
    else{
        board[tr+sz][tc+sz] = t;                //标记一个假设的特殊点
        chess_board(tr+sz, tc+sz, tr+sz, tc+sz, sz); //递归该部分
    }
}

void print_chess_board()
{
    cout.setf(ios::left); //左对齐
    for(int i=0; i<BOARD_SZ; ++i){
        for(int j=0; j<BOARD_SZ; ++j){
            cout.width(3); //打印宽度为3
            cout<<board[i][j];
        }
        cout<<endl;
    }
}

int main()
{
    chess_board(0, 0, 3, 4, BOARD_SZ);
    print_chess_board();
    return 0;
}
```