# 棋盘覆盖问题

- 本题采用分治法的主要思路将棋盘不断四等分，其中对于*含特殊点的部分继续递归*，对*不含特殊点的部分分区域选择骨牌覆盖处*，即可保证每次在同一递归深度的覆盖方式能覆盖出一个三角骨牌形状，且最终能够实现非特殊点处的全覆盖；

```c
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_BOARD_SIZE 1024


//tile为骨牌序号，board为棋盘，二者为全局变量
int tile = 1;

int board[MAX_BOARD_SIZE][MAX_BOARD_SIZE];

void chessboard(int tr, int tc, int dr, int dc, int size) {
  //如果size为1，则意味着递归完成，直接退出
  if (size == 1) {
    return;
  }

  int t = tile++;
  int s = size / 2;
  //对于特殊点在分区内则直接进行递归，反之，若为左上分区则在右下处覆盖骨牌，若为右上分
  //在左下处覆盖骨牌，以此类推；覆盖将覆盖处记为特殊点进行下一轮递归
  if (dr < tr + s && dc < tc + s) {
    chessboard(tr, tc, dr, dc, s);
  } else {
    board[tr + s - 1][tc + s - 1] = t;
    chessboard(tr, tc, tr + s - 1, tc + s - 1, s);
  }
```

```c
  if (dr < tr + s && dc >= tc + s) {
    chessboard(tr, tc + s, dr, dc, s);
  } else {
    board[tr + s - 1][tc + s] = t;
    chessboard(tr, tc + s, tr + s - 1, tc + s, s);
  }

  if (dr >= tr + s && dc < tc + s) {
    chessboard(tr + s, tc, dr, dc, s);
  } else {
    board[tr + s][tc + s - 1] = t;
    chessboard(tr + s, tc, tr + s, tc + s - 1, s);
  }

  if (dr >= tr + s && dc >= tc + s) {
    chessboard(tr + s, tc + s, dr, dc, s);
  } else {
    board[tr + s][tc + s] = t;
    chessboard(tr + s, tc + s, tr + s, tc + s, s);
  }
}

int main() {
  int n;
  printf("Input the size of the board. \n");
  scanf("%d", &n);

  int boardSize = (int)pow(2, n);

  for (size_t i = 0; i < boardSize; i++) {
    for (size_t j = 0; j < boardSize; j++) {
      board[i][j] = 0;
    }
  }

  int uniqRow, uniqCol;
  printf("Input the unique row of the board. \n");
  scanf("%d", &uniqRow);
  printf("Input the unique col of the board. \n");
  scanf("%d", &uniqCol);

  chessboard(0, 0, uniqRow, uniqCol, boardSize);
```

```
  for (size_t i = 0; i < boardSize; i++) {
    for (size_t j = 0; j < boardSize; j++) {
      printf("%2d\t", board[i][j]);
    }
    printf("\n");
  }

  return 0;
}
```