《算法分析与设计》

实 验 报 告

学 号

姓 名

年 级 2020

专 业 软件工程

院 系 计算机与人工智能学院

二0二二年三月

目 录

实验一 求两个圆相交部分的面积······································1

实验二 算法效率分析与比较·········································11

实验三 穷举法设计与实验···········································11

实验四 分治法设计与实验···········································11

# 实验4.3 棋盘覆盖问题

1. 实验目的

|  |
| --- |
| 1. 理解分治算法的求解过程。 2. 通过范例学习分治策略设计技巧，学会分析分治算法的时间复杂度。 3. 掌握用分治算法求解具体问题，了解其面临的瓶颈。 |

1. 实验任务

|  |
| --- |
| 在一个2kX2k (k>=O)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。棋盘覆盖问题要求用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. 实验环境

|  |
| --- |
| * 1. 硬件环境  1. 计算机：拯救者R7000P 2020H 2. CPU: AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz 3. RAM：16GB    1. 软件环境 4. 操作系统：Windows11家庭中文版 5. 开发工具：Visual Studio Code |

1. 实验步骤及结果

|  |
| --- |
| * 1. 实验预习   4.1.1分析  分治法求解棋盘覆盖问题的技巧在于划分棋盘，使划分后的子棋盘的大小相同，并且每个子棋盘均包含一个特殊方格，从而将原问题分解为规模较小的棋盘覆盖问题。  当K>0时，可将2^k\*2^k的棋盘划分为4个2^(k-1)\*2^(k-1)的子棋盘, 这样划分后，由于原棋盘只有一个特殊方格，所以，这4个子棋盘中只有一个子棋盘包含该特殊方格，其余3个子棋盘中没有特殊方格。为了将这3个没有特殊方格的子棋盘转化为特殊棋盘，以便采用递归方法求解，可以用一个L型骨牌覆盖这3个较小棋盘的会合处，从而将原问题转化为4个较小规模的棋盘覆盖问题。递归地使用这种划分策略，直至将棋盘分割为1X1的子棋盘。  4.1.2伪代码  **chessBoard**:  input:tr,tc,dr,dc,size  output:board[size][size]  **begin**  棋子标号t随递归次数递增；   1. 覆盖左上角的子棋盘）   **If**特殊方格在此棋盘中**then**  递归**chessBoard**；  **Else**  用t号L型棋子覆盖右下角；  递归**chessBoard**；   1. 覆盖右上角子棋盘）   **If**特殊方格在此棋盘中**then**  递归**chessBoard**；  **Else**  用t号L型棋子覆盖左下角；  递归**chessBoard**；   1. 覆盖左下角子棋盘）   **If**特殊方格在此棋盘中**then**  递归**chessBoard**；  **Else**  用t号L型棋子覆盖右上角；  递归**chessBoard**；   1. 覆盖右下角子棋盘）   **If**特殊方格在此棋盘中**then**  递归**chessBoard**；  **Else**  用t号L型棋子覆盖左上角；  递归**chessBoard**；  **end**  4.1.3时间复杂度分析  设T（K）是算法chessBoard覆盖一个2^k\*2^k棋盘所需的时间，由分治策略可知，T（K）满足如下方程：  T(k)=O(1), k=0 (1)  T(k)=4T(k-1)+O(1), k>0 (2)  解此递归方程可得：  T(k)=O(4^k) (3)  所以分治法解决棋盘覆盖问题的时间复杂度为，O(4^k)。  4.1.4程序代码  #include<stdio.h>  #include<Windows.h>  int tile = 1;  int k = 0;  int dr = 0; *//特殊方格所在的行号*  int dc = 0; *//特殊方格所在的列号*  int tr = 0; *//棋盘左上角方格的行号*  int tc = 0; *//棋盘左上角方格的列号*  int board[10][10];  void  chessBoard(int *tr*, int *tc*, int *dr*, int *dc*, int *size*)  {      if(1 == *size*)   return ;      int t = tile++, *//L型骨牌号*      s = *size*>>1; *//分割棋盘*  *//覆盖左上角子棋盘*      if(*dr* <*tr* + s && *dc* < *tc* + s)  *//特殊方格在此棋盘中*          chessBoard(*tr*, *tc*, *dr*, *dc*, s);      else      {  *//用t号L型骨牌覆盖右下角*          board[*tr*+s - 1][*tc* + s -1] = t;  *//覆盖其余方格*          chessBoard(*tr*,*tc*,*tr*+s-1,*tc*+s-1,s);      }  *//覆盖右上角子棋盘*      if(*dr*<*tr*+s && *dc* >= *tc*+s)  *//特殊方格在此棋盘中*          chessBoard(*tr*,*tc*+s, *dr*,*dc*,s);      else *//此棋盘中无特殊方格*      {  *//用t号L型骨牌覆盖左下角*          board[*tr*+s-1][*tc*+s] = t;  *//覆盖其余方格*          chessBoard(*tr*, *tc*+s, *tr*+s-1,*tc*+s, s);      }  *//覆盖左下角子棋盘*      if(*dr*>=*tr*+s && *dc*<*tc* + s)  *//特殊方格在此棋盘中*          chessBoard(*tr*+s,*tc*,*dr*,*dc*,s);      else *//用t号L型骨牌覆盖右上角*      {          board[*tr*+s][*tc*+s-1] = t;  *//覆盖其余方格*          chessBoard(*tr*+s,*tc*,*tr*+s,*tc*+s-1,s);      }  *//覆盖右下角子棋盘*      if(*dr*>=*tr*+s && *dc*>=*tc*+s)  *//特殊方格在此棋盘中*          chessBoard(*tr*+s,*tc*+s,*dr*,*dc*,s);      else *//用t号L型骨牌覆盖左上角*      {          board[*tr*+s][*tc*+s] = t;  *//覆盖其余方格*          chessBoard(*tr*+s,*tc*+s,*tr*+s,*tc*+s,s);      }  }  int main()  {      printf("请输入k的大小:");      scanf("%d", &k);      int size = 2;      for (int i = 1; i < k; i++)          size \*= 2; *// size = 2^k,棋盘规格为2^k \* 2^k*      printf("请输入特殊方格的横纵坐标：");      scanf("%d %d", &dr, &dc);      chessBoard(0, 0, dr, dc, size);      printf("输出结果为：\n");      for (int i = 0; i < size; i++)      {          for (int j = 0; j < size; j++)          {              printf("%2d", board[i][j]);              printf("  ");          }          printf("\n");      }      system("pause");      return 0;  }  4.1.5求解过程分析  采用分治策略，对分割后的四个小方块进行判断，判断特殊方格是否在里面。每次先记录下整个大方块的左上角方格的横纵坐标tr，tc，与特殊方格横纵坐标dr，dc进行比较；如果特殊方块在，直接递归即可；如果不在，根据分割的四个方块的不同位置，把右下角、左下角、右上角或者左上角的方格标记为特殊方块，然后继续递归。变量s记录边的方格数，每次递归，边的方格数都会减半。变量t记录L 型骨牌的编号。  4.2上机实验  4.2.1算法测试  测试数据  3  2 2  4.2.2测试结果及其分析  对应测试数据，其程序执行过程中间变量的值如下截图所示；        上图所示为部分程序执行过程，其中t为棋子编号，size为棋盘大小，输入k=3，size分割由8->4->2,dr、dc为每个棋盘的特殊棋子的横纵坐标，tr、tc为每个棋盘的左上角横纵坐标；  程序最终输出结果如图所示：    其中，特殊棋子用‘0’表示，L型骨牌用递增的三个相同的数字表示，成‘L’型；经验证可知，特殊棋子坐标为（2，2），整个棋盘大小为2^3=8,每次分治即将当前棋盘四等分，并保证每个子棋盘中都有一个特殊棋子，依次类推，由输出结果可知，此算法成功实现‘L’型棋牌覆盖整个棋盘问题。  5.实验总结  通过棋盘覆盖问题了解了分治算法的分析求解过程，时间复杂度分析方法,以及如何设计分治算法解决实际问题。通过本次实验加深理解了分治算法的特点以及在实际应用中如何正确设计并使用分治算法。 |