**山东大学计算机科学与技术学院  
《数据结构与算法》课程设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201705130120 | 姓名： 苑宗鹤 | | 班级： 17.1 |
| 上机学时：4 | | 日期： 2019.4.28 | |
| 课程设计题目：关键活动 | | | |
| 软件开发环境：win10 x64 vs2019 | | | |
| 报告内容：  1.需求描述  1.1 问题描述  残缺棋盘(defective chessboard)：是一个有 2k×2k个方格的棋盘，其中恰有一个方格残缺。对于任 意k，恰好存在22k种不同的残缺棋盘。 在残缺棋盘中，要求用三格板(triominoes)覆盖 残缺棋盘。在覆盖中，任意两个三格板不能重叠，任 意一个三格板不能覆盖残缺方格，但三格板必须覆盖 其他所有方格。  1.2 基本要求  输入棋盘大小和残缺方格的位置，输出覆盖后的棋 盘，输出棋盘时要着色，共享同一边界的覆盖应着不 同的颜色。棋盘是平面图，因此最多只需4种颜色，为 覆盖着色，要求设计贪婪着色启发式方法，以尽量使 用较少的颜色  1.3 输入说明  输入k,x,y  其中k为棋盘的2^k次幂大小 x,y为残缺坐标  1.4 输出说明  输出棋盘的涂色方案  2.设计  2.1 系统结构设计    2.2 设计思路  对于一个残缺棋盘  使用分治的方法可以将其转化为1个残缺棋盘与三个完整棋盘  其中3个完整棋盘相连的L型部分被取消 则转化为3个残缺棋盘  递归求解直到被划分的区域=2\*2  涂色按照贪心进行涂色  2.3 数据及数据类(型)定义  int mapp[5000][5000]; //棋盘区域划分  int totid = 1;//自增区域标号  int color[5000000];//区域id对应颜色  2.4.算法设计及分析  贪心涂色算法规则:  首先将输入的残缺棋盘的残缺位 转化为棋盘右下角优先的对称模式  即 如果残缺坐标是(1,1) 则棋盘与残缺(n,n)的情况等价  按照n,n的情况计算  然后 按照先行后列的方式 进行可能颜色序号从小到大的贪心涂色  则 涂色之后 必然能仅仅花费3种颜色将棋盘涂满  贪心涂色算法的证明:  首先 最小单元是2\*2 当单元大小达到4\*4时(共16种情况) 考虑到对称只有三种情况    这三种情况最坏状况是边界会出现两种颜色加一个颜色空缺  最好情况是两两相间的颜色  而将这三种组合任意两两拼接可以发现 一个边界上只有两种颜色 所以必定能合法拼接  则对于n>4的情况 必然由 4\*4的状态衍生而来 所以合法  而n<=4的时候通过枚举也可以知道结果全部可以涂3色  所以只需要三种颜色  3. 测试结果  输入:  4 3 1  输出:      4. 分析与探讨  由于题目特殊 所以只用三种颜色就可以吐满  5. 附录：实现源代码  **main.cpp**  #include<iostream>  #include<vector>  #include<algorithm>  #include<iomanip>  #include<fstream>  using namespace std;  bool xyin(int x, int y, int arex, int arey, int len) {  return x >= arex && x < arex + len && y >= arey && y < arey + len;  }  int mapp[5000][5000];  int totid = 1;  int color[5000000];  void dfs(int bx, int by, int sx, int sy, int len) {  //已经占用的xy 区块起始坐标sx sy  if (len == 1) {  ++totid;  for (int a = 0; a <= 1; ++a) {  for (int b = 0; b <= 1; ++b) {  if (mapp[sx + a \* len][sy + b \* len] == 0) {  mapp[sx + a \* len][sy + b \* len] = totid;  }  }  }  return;  }  int xid = ++totid;  for (int a = 0; a <= 1; ++a) {  for (int b = 0; b <= 1; ++b) {  if (xyin(bx, by, sx + a \* len, sy + b \* len, len)) {  dfs(bx, by, sx + a \* len, sy + b \* len, len / 2);  }  else {  mapp[sx + len - (!a)][sy + len - (!b)] = xid;  dfs(sx + len - (!a), sy + len - (!b), sx + a \* len, sy + b \* len, len / 2);  }  }  }  }  int k;  int colorid = 1;  void dfscolor() {  for (int a = 1; a <= 1 << k; ++a) {  for (int b = 1; b <= 1 << k; ++b) {  int& nowcol = color[mapp[a][b]];  int id = mapp[a][b];  if (nowcol != 0 || id==1)continue;  //cout << "涂色坐标x=" << a << " y=" << b;  for (int col = 1; col <= 4; ++col) {  //尝试涂4种颜色  int ok = 1;  for (int i = -1; i <= 1; ++i) {  for (int j = -1; j <= 1; ++j) {  if (mapp[a + i][b + j] != id)continue;  for (int ii = -1; ii <= 1; ++ii) {  for (int jj = -1; jj <= 1; ++jj) {  if (ii \* jj != 0 || (ii == 0 && jj == 0))continue;  if (color[mapp[a + i + ii][b + j + jj]] == col && mapp[a + i + ii][b + j + jj] != id)ok = 0;  }  }  }  }  if (ok == 1) {  nowcol = col;  break;  }  }  //cout << "涂色结果" << color[mapp[a][b]] << endl;  }  }  }  int main() {  cin >> k;//2^k棋盘  int sx, sy;  cin >> sx >> sy; //残缺坐标 1~2^n  mapp[sx][sy] = 1;  dfs(sx, sy, 1, 1, 1 << (k - 1));  for (int a = 1; a <= 1 << k; ++a) {  for (int b = 1; b <= 1 << k; ++b) {  cout << setw(2) << mapp[a][b];  }  cout << endl;  }  dfscolor();  for (int a = 1; a <= 1 << k; ++a) {  for (int b = 1; b <= 1 << k; ++b) {  cout << setw(2) << color[mapp[a][b]];  }  cout << endl;  }  fstream fout("out.txt", ios::out);  fout << k << endl;  for (int a = 1; a <= 1 << k; ++a) {  for (int b = 1; b <= 1 << k; ++b) {  fout << mapp[a][b]<<" ";  }  fout << endl;  }  for (int a = 1; a <= 1 << k; ++a) {  for (int b = 1; b <= 1 << k; ++b) {  fout << color[mapp[a][b]]<<" ";  }  fout << endl;  }  fout.close();  return 0;  }  可视化部分  Vers.py  with open('out.txt','r') as f:  k=int(f.readline())  blocks=[]  for i in range(1<<k):  x=f.readline().replace('\n','').replace(' ',' ').strip().split(' ')  blocks.append([x for x in map(int,x)])  colors=[]  for i in range(1 << k):  x = f.readline().replace('\n', '').replace(' ',' ').strip().split(' ')  colors.append([x for x in map(int,x)])    print(blocks)  print(colors)  import cv2  import numpy as np  from matplotlib import pyplot as plt  img = np.zeros((1024,1024,3), np.uint8)  img.fill(255)  lens=30  for i in range(1<<k):  for j in range(1 << k):  if colors[i][j]==1:  colx=(240,50,0)  elif colors[i][j]==2:  colx=(0,240,50)  elif colors[i][j]==3:  colx=(50,0,240)  else:  colx=(125,0,0)  cv2.rectangle(img, (i \* lens, j \* lens), (i \* lens + int(lens / 1), j \* lens + int(lens / 1)),  (200, 200, 200), thickness=-1)  cv2.rectangle(img,(i\*lens+1,j\*lens+1),(i\*lens+int(lens/1)-2,j\*lens+int(lens/1)-2),colx,thickness=-1)    cv2.imshow('image', img)  cv2.waitKey(0)  cv2.destroyAllWindows() | | | |