深 圳 大 学 实 验 报 告

课程名称：算法设计与分析

实验项目名称：回溯法（地图填色问题）

学院：计算机与软件学院

专业：计算机科学与技术

指导教师：杨烜

报告人：郑雨婷 学号：2021150122 班级：高性能

实验时间：2023/04/6——2023/04/27

实验报告提交时间：2023/04/27

教务部制

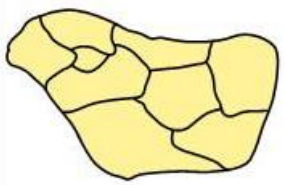
实验目的与要求：

一、实验目的：

1.掌握回溯法算法设计思想。
2.掌握地图填色问题的回溯法解法。

二、实验要求：

1、对下面这个小规模数据，利用四色填色测试算法的正确性；



2、对附件中给定的地图数据填涂，解释为什么填涂颜色超过了四色；
3、设计剪枝策略，分析不同的剪枝策略的效率。
4、随机产生不同规模的图，分析算法效率与图规模的关系。

方法、步骤：

一、问题描述

将地图转换为平面图，每个地区变成一个节点，相邻地区用边连接，为这个图形的顶点着色，并且两个顶点通过边连接时必须具有不同的颜色。

二、步骤

1.设计回溯法实现地图填色，在小规模地图上验证算法的正确性。
2.设计不同的剪枝策略进行改进，对附件中的文件进行填涂，根据实测数据，对不同的剪枝策略进行比较分析。
3.最后，随机生成不同规模的图来分析算法效率与图规模的关系。

实验过程及内容：

一、基本回溯法

1.1 回溯法原理：

在包含问题的所有解的解空间树中，按照深度优先搜索的策略，从根结点出发深度探索解空间树。

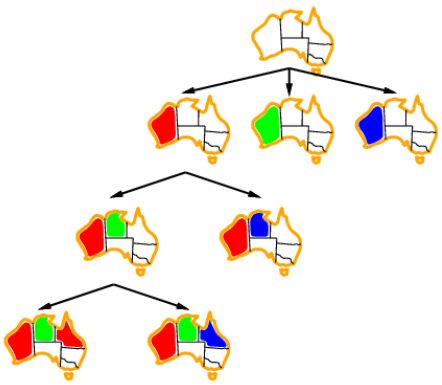


图 1 回溯法原理示意图

结合地图填色问题，回溯法的基本思想是：在对某个节点进行填色时，如果所填颜色合法，则前往下一层节点进行操作；如果所填颜色非法，就需要回溯到上一层节点重新考虑填色方案。如此循环操作直至所有节点均被填色，完成地图填色任务。

1.2 核心伪代码

```
void base_dfs(x) {
if x > v_num
    ans++;
    for i = 1 to   c_num
      pp[x].color = i;
      if isvalid(x)   base_dfs(x + 1);
}
```

1.3 在小地图上验证算法正确性

对给定的地图按图 2 标号，将其抽象为由 9 个点和 17 条边组成的无向图，并使用 4 种颜色进行填色。在 10 次测试后，平均每次运行时间为 0.07ms，总共得到了 480 组解。运行结果如表 1 所示，验证该算法正确。

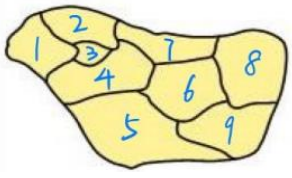


图 2 对小地图进行标号

运行时间（ms）	le9_4	le450_5	le450_15	le450_25
基本回溯	0.07	∞	∞	∞

表 1 基本回溯法的运行时间

二、剪枝策略

最基础的回溯法无法在短时间内得到附件中的解，因为回溯法实际上就是穷举法。只有进行剪枝才可能在较短的时间内得到解，下面给出三种剪枝策略：选择填涂区域、选择填涂颜色、向前探测。

2.1 选择填涂区域

2.1.1 思想

区域的选择遵循两个准则：

1.MRV 准则——选择可选颜色数量最少的区域。通过寻找可涂颜色数量最少的区域进行染色，由于可能涂上的颜色数量越少，则必须的尝试次数也越少。

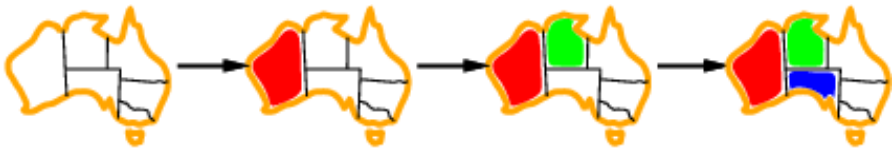


图 3 MRV 填涂示例

2.DH 准则——选择度数最大的区域。度最大选择是优先选择度最多的节点，因为度数越多的节点受到其他节点的约束就越大，若优先填涂了该节点那么可以影响周围节点可选择的颜色。

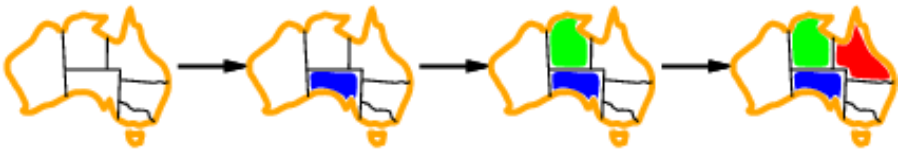


图 4 DH 填涂示例

两个准则都是让容易导致失败的变量先赋值，如果这些变量注定无法赋值，我们希望早些发现。优先利用 MRV 选择，当多个区域具有相同的可选颜色数时，利用 DH 选取度数最大的区域。

2.1.2 核心伪代码

```
void dfs(x) {
if x > v_num
      ans++;
      xuan=MRV_DH();
      for i = 1 to   c_num
         pp[xuan].color = i;
         if isvalid(x)   dfs(x + 1);
}
int MRV_DH() {
for i = 1 to v_num
      if pp[i].color == -1
            if (i 点的可用颜色少) OR (i 的可用颜色相等并且度大)
                  选择点 i
return x;
}
```

2.1.3 实测数据

使用选择填涂区域的剪枝策略，对 le9_4.txt 求出全部解，对 le450_5.txt、le450_15.txt、le450_25 求出一个解，所用的时间如下表 2：

文件	le9_4	le450_5	le450_15	le450_25
运行时间(ms)	4	1241	1354	7

表 2 选择区域的运行时间

其中，15 色的第一个点选择 432 点，否则会陷入完全子图中，导致很久都运行不出结果。根据表 2 可以看出，选择填涂区域的剪枝策略有效。

2.2 选择填涂颜色

2.2.1 思想

选择对其他区域影响少的颜色。在选择颜色时我们需要选择该节点影响周围节点剩余颜色最少的颜色，也就是说要尽量不要选择周围节点可使用的颜色。如果相邻节点的剩余颜色中有该颜色，则进行计数，最后优先选择计数最少的颜色。

图 5 选择颜色填涂示例

2.2.2 核心伪代码

```
void dfs(x) {
if x > v_num
    ans++;
//得到按序排列的 sortcolor[]
for i = 1 to c_num
    pp[x].color = sortcolor[i];
    if isvalid(x)  dfs(x + 1);
}
int MCV(x, vector  c) {
for  i = 1  to c_num;
  cnt = 0;
  //遍历相邻节点，有可使用该颜色的，cnt++
  colorSelect a(i, cnt);
  c.push_back(a);
}
sort(c.begin(), c.end());      //按照 cnt 升序排序
}
```

2.2.3 实测数据

使用选择填涂区域的剪枝策略，对 le9_4.txt 求出全部解，对 le450_5.txt、le450_15.txt、le450_25 求出一个解，所用的时间如下表 3：

文件	le9_4	le450_5	le450_15	le450_25
运行时间(ms)	4	∞	∞	∞

表 3 选择颜色的运行时间

根据表 3 可以看出，选择填涂颜色的剪枝策略在大规模地图并没有使求解时间有明显的提升，反而，在小规模数据上还增加了对颜色的排序过程，使得运行更慢。

2.3 向前探测

2.3.1 思想

如果当前涂色使得相邻的某个区域无色可选，则回退。

具体实现是当某节点选择了一个颜色之后，则将邻接节点的可选颜色中，将该颜色删除。如果在删除的过程中发现删除后已经没有颜色可以填涂，那么就放弃该节点的那一颜色。这种方法可以提前知道某节点的选择是否正确，可以提前试错。也就是加大了剪枝操作。

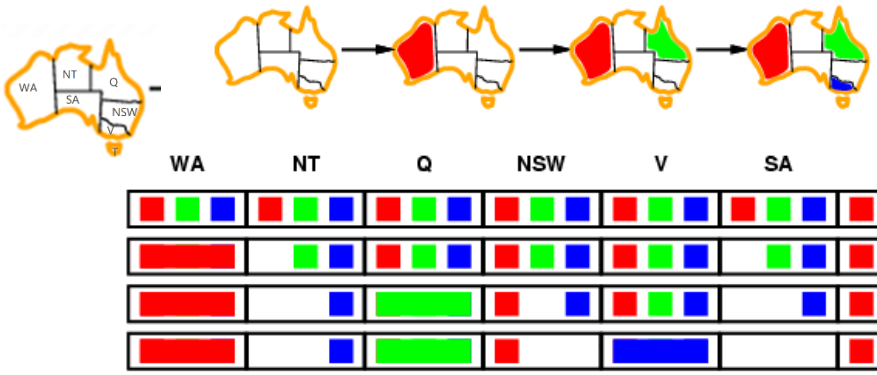


图 6 向前探测填涂示例

如图 6 所示，Q 选择绿色导致 SA 没有颜色可涂选了，提前发现失败进行回溯。

2.3.2 核心伪代码

```
bool paint(x, se) {
    if (!x.ok_color[se]) return false;
    x.color = se;
    for   i = 0 to x.du -1
        id = x.connect[i];
        if (pp[id].ok_color[se] && pp[id].color == -1) {
          if (pp[id].ok_num == 1) break;      //如果相邻点只有剩该颜色可涂
                                              被涂了就没有可用颜色了，发现失败，回溯
          pp[id].ok_color[se] = false;
          pp[id].ok_num--;
          }
      }
  return true;
}
```

2.3.3 实测数据

使用向前探测的剪枝策略，对 le9_4.txt 求出全部解，对 le450_5.txt、le450_15.txt、le450_25 求出一个解，所用的时间如下表 4：

文件	le9_4	le450_5	le450_15	le450_25
运行时间(ms)	0	∞	∞	∞

表 4 向前探测的运行时间

根据表 4 可以看出，向前探测的剪枝策略在大规模地图并没有使求解时间有明显的提升。

数据处理分析：

一、不同剪枝对比

将不同的剪枝策略组合，并对地图进行测试，其中 le450_5、le450_15、le450_25 只找一个解，其中，15 色的第一个点选择 432 点，所用的时间如下表 5：

文件	le9_4	le450_5	le450_15	le450_25
基本	0	∞	∞	∞
纯选颜色	4	∞	∞	∞
纯选点	4	1241	1354	7
纯向前探测	0	∞	∞	∞
选点+向前探测	1	620	1205	5
选颜色+向前探测	4	∞	∞	∞
选颜色+选点	5	869	57102	15
三种全	4	830	52257	12

表 5 不同剪枝策略的运行时间对比

观察数据比较得出结论：

三种剪枝策略里 MRV_DH 选择下一个涂色的区域可以很明显的剪枝，向前探测的剪枝程度较低，颜色选择反而会使得运行时间增加。原因是向前探测只是向前了一步，所以只提前一点点发现失败。颜色选择导致运行时间更长是因为只寻找了一组解，而每次的排序需要耗时，只有求大规模的所有解时会看到明显的时间缩短。

二、不同规模地图测试

（1）边/点固定，颜色不同

固定 50 个点，100 条边，颜色数从 5-9，运行时间和解个数如下表 6：

	50，100，5	50，100，6	50，100，7	50，100，8	50，100，9
解个数	40960	262500	1179360	4168136	12386304
运行时间/ms	15	47	187	641	1797
平均每个/ms	0.00036621	0.000179	0.0001586	0.00015379	0.00014508

表 6 不同颜色数的运行时间对比

将平均每个解所耗的时间绘制成图 7：

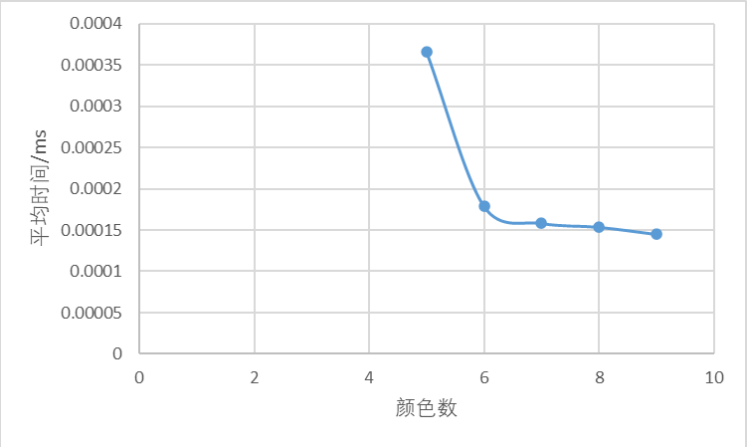


图 7 颜色数不同的平均运行时间

得到结论当边和点固定不变时颜色数越多越快找到解，因为颜色很多时没有相对不会那么容易失败，回溯次数减少。

（2）颜色数固定，边/点不同

固定 10 种颜色，改变边与点的比例，运行时间和解个数如下表 7：

	50,200,10	50,230,10	50,250,10	50,275,10	50,280,10
解个数	211392	623616	177030	207104	744548
运行时间/ms	47	125	31	46	173
平均每个/ms	0.00022234	0.0002004	0.0001751	0.00022211	0.00023236

表 7 不同边点比的运行时间对比

将平均每个解所耗的时间绘制成图 8：

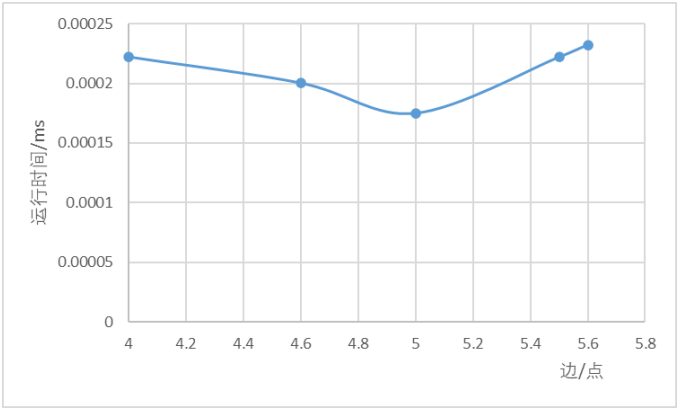


图 8 不同边点比的平均运行时间

得到结论当颜色数固定不变时，运行时间先减少后增加，当边/点约为 5 的时候平均运行时间最短。与想象的不太一样，本以为运行时间会一直递减，因为度数越大可行解就越少了，就能更快剪枝。结果表明不能太依赖惯性思维，事实和想象的是有区别的。

实验结论：

本次实验熟悉了回溯法并且体验了使用回溯法怎么进行剪枝操作，对于回溯法有了更深层次的了解，也懂得了剪枝和选择节点对于回溯法提升效率的重要性。给定的附件从一开始无法跑出来到几秒可以明显地体会到优化的重要性。其中，选择下一个填涂区域的方法是十分有效的。

指导教师批阅意见：

成绩评定：

指导教师签字：

年　　月　　日

备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。