深 圳 大 学 实 验 报 告

课程名称：算法设计与分析

实验名称：回溯法地图填色问题

学院：计算机与软件学院 专业：腾班

报告人：叶茂林 学号：2021155015 班级：腾班

同组人：叶茂林

指导教师：李炎然

实验时间：2023.4.27

实验报告提交时间：2023.5.4

教务处制

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．实验步骤与结果

1、小规模地图如图 1 所示，利用四色填色测试算法的正确性；

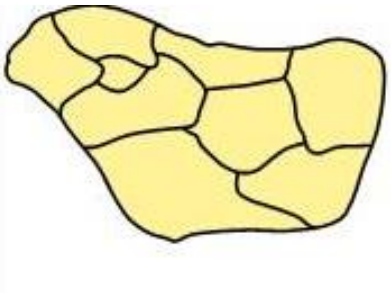

图 1 小规模地图

先数学建模，把地图变成平面图，给每个区域编号，如图 2 所示。

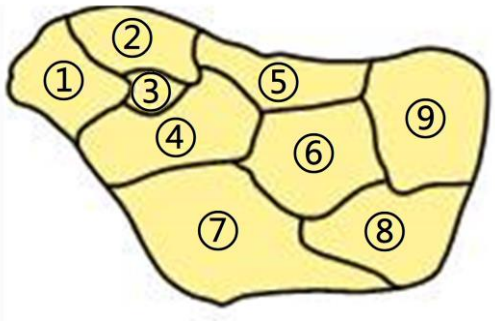

图 2 地图区域编号

然后将每个区域变成图的一个顶点，区域之间用边连接，如图 3 所示。

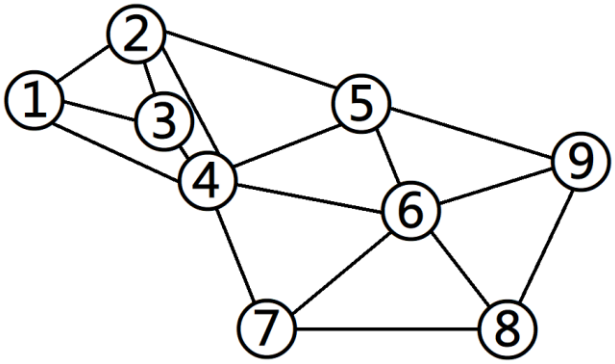

图 3 小规模地图平面图化

然后我们就可以用邻接矩阵来存储地图，问题转化为给图的顶点着色，约束是边连接的两个顶点的颜色不能相同。

回溯法

回溯法的基本思想是采用递归和深度优先搜索的方法，尝试在一组可能的解中搜索出符合要求的解，在搜索过程中，若发现当前所选的方案不能得到正解，就回溯到前面的某一步（即撤销上一次的选择），换一种可能性继续尝试，直到找到符合要求的解或者所有的可能性都已尝试完毕。

在地图填色中，回溯法从某一区域开始，如图 4 所示，尝试使用不同的颜色进行填充，然后递归地尝试填充相邻的区域，如果发现当前填充颜色与相邻区域的颜色冲突，则回溯到之前的状态重新选择一种颜色进行填充，如此往复直到所有的区域都被填充上颜色或者无解。

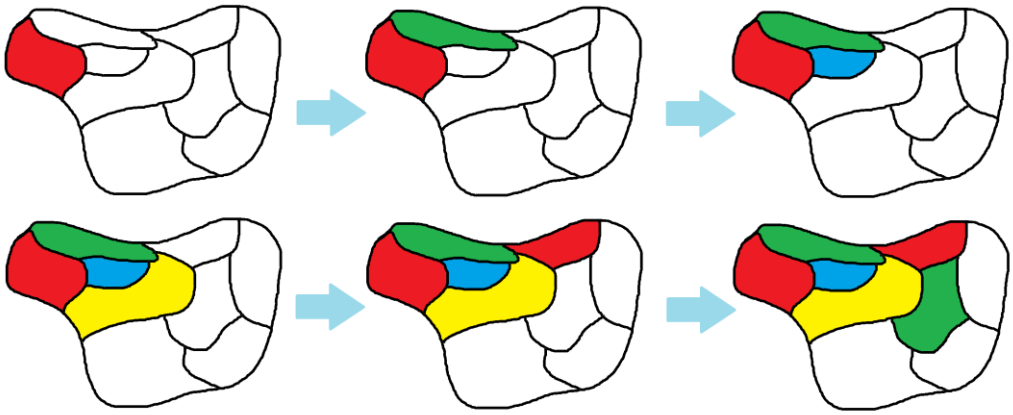

图 4 回溯法地图填色示例

伪代码

Algorithm 1 fillColor(vertex)

```
1: choseOneColor(vertex)
2: if !conflict(vertex) then
3:     fillColor(nextvertex)
4: end if
```

运行结果

如图 5 所示，对于小规模地图，回溯法成功在 323 毫秒内找出 480 个解，并将每个解打印出来，验证了算法的正确性。

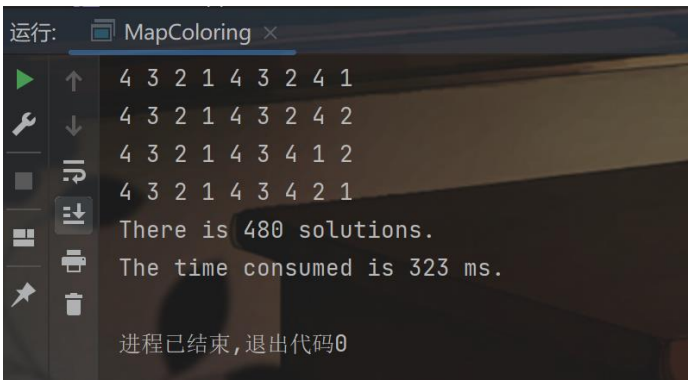

图 5 回溯法小规模地图填色

2、对附件中给定的地图数据填涂；

首先还是用经典回溯法试跑一下，只找一个解的情况，如表 1 所示。

经典回溯法			
地图	le450_5a	le450_15b	le450_25a
时间 (ms)	∞	∞	∞

表 1 经典回溯法大规模地图填色

由结果可以看出，当规模大时，回溯法的搜索空间会变得非常庞大，从而需要耗费大量的时间和内存资源来完成搜索过程，这将导致算法的运行时间呈指数级增长，短时间内无法求解。因此，我们需要对回溯法进行优化。

（1） 最大度优先

经典回溯法的问题在于解的空间太大，回溯次数太多，而优先选择邻边个数最多的顶点进行填色则会对剩下未填色的顶点产生更多的限制，从而减少回溯的次数，如图 6 所示，每次填色，我们都优先填度最大的区域。

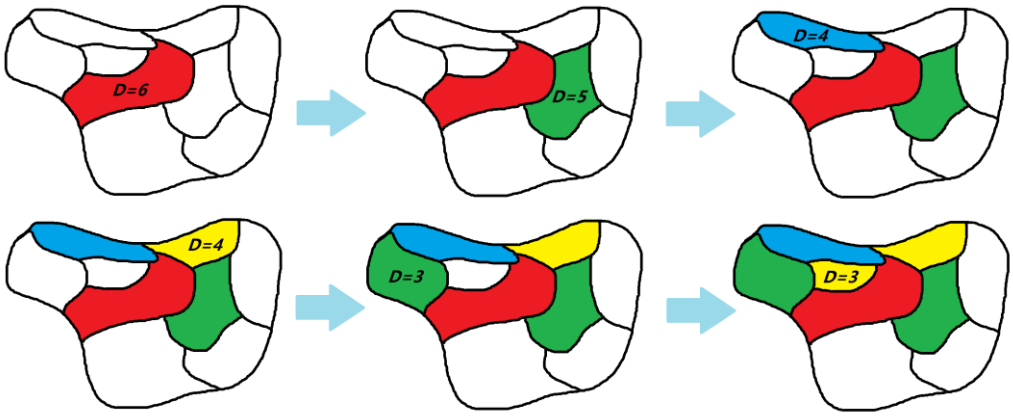

图 6 最大度优先地图填色示例

伪代码

Algorithm 2 fillColor(vertex)

```
1: choseOneColor(vertex)
2: if !conflict(vertex) then
3:     nextvertex←MaxDegreeVertex()
4:     fillColor(nextvertex)
5: end if
```

运行结果

先在小规模地图上验证算法的正确性，如图 7 所示，最大度优先可以在 325 毫秒内找出 480 个解。

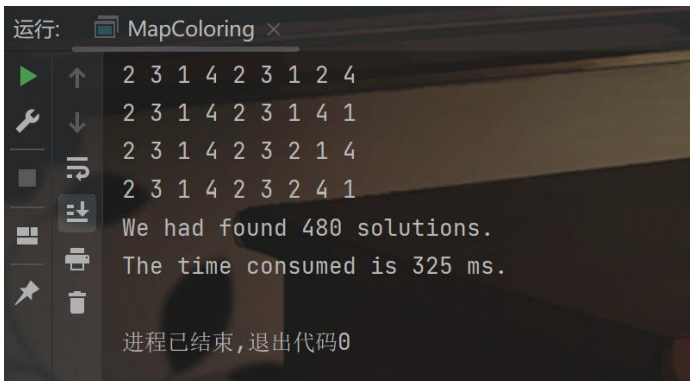

图 7 最大度优先小规模地图填色

然后尝试填涂三个大规模地图，只找一个解的情况，如表 2 所示。

最大度优先			
地图	le450_5a	le450_15b	le450_25a
时间 (ms)	∞	∞	1

表 2 最大度优先大规模地图填色

由结果可知，我们的最大度优先优化策略略显成效，但是第一个和第二个地图还是无法在短时间内找到解，我们需要继续努力。

（2） 最少可选颜色优先

每次选择区域进行填色时优先选择剩余可用颜色最少的区域进行填色，这样可以减少剩余可用颜色最多的地区需要尝试不同颜色的次数，如图 8 所示，每填完一个区域就更新邻近区域的可选颜色，然后优先选择可选颜色最少的区域进行填色。

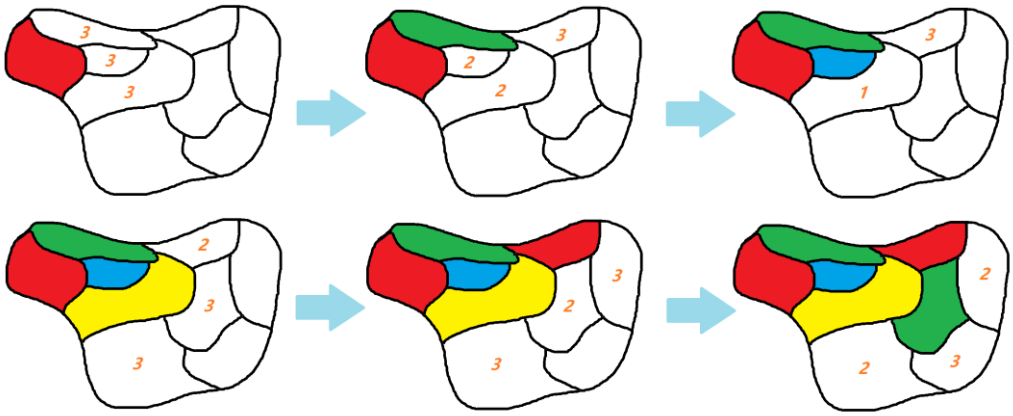

图 8 最少可选颜色优先地图填色示例

伪代码

Algorithm 3 fillColor(vertex)

```
choseOneColor(vertex)
if !conflict(vertex) then
    nextvertex←MinRemainingValueVertex()
    fillColor(nextvertex)
end if
```

运行结果

先在小规模地图上验证算法的正确性，如图 9 所示，可以在 321 毫秒内找出 480 个解。

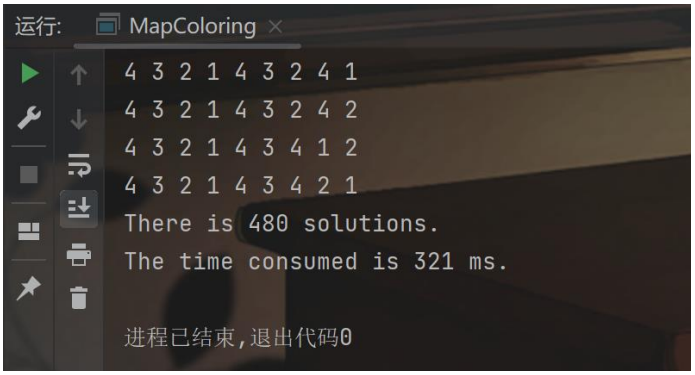

图 9 最少可选颜色小规模地图填色

然后尝试填涂三个大规模地图，结果如表 3 所示

最少可选颜色优先			
地图	le450_5a	le450_15b	le450_25a
时间 (ms)	2000	∞	10

表 3 最少可选颜色优先大规模地图填色

由结果可知，最少可选颜色优先的优化策略使得第一个图也可以在 2 秒内找到解了，通过算法的优化，原本短时间内无解的问题可以迅速解决。

然后我们尝试将最大度优先和最少可选颜色优先结合去填涂三个大规模地图，结果如表 4 所示。

最少可选颜色优先+最大度优先			
地图	le450_5a	le450_15b	le450_25a
时间 (ms)	600	3000	11

表 4 最少可选颜色+最大度地图填色

由结果可知，将最少可选颜色优先和最大度优先相结合后，三个地图均可以迅速找到解，其中第一个地图需要 600 毫秒，而第二个地图在 3 秒内终于找到了一个解。

继续测试，对第一个地图找全部解，对第二个和第三个地图找 10 万个解，结果如表 5 所示，可知该优化策略可以迅速找解。

最少可选颜色优先+最大度优先			
地图	le450_5a	le450_15b	le450_25a
时间 (ms)	3840解80000	10万解4200	10万解2000

表 5 最少可选颜色+最大度找多解

（3） 向前探测

每次选择区域进行填色的时候，先判断该填涂的颜色是否会导致邻近的区域无色可填，如果导致了邻近区域无色可填则直接换一种颜色填涂，如图 10 所示，每填一个区域就更新邻近区域的可用颜色，如果可用颜色为 0 则说明此处不能填这个颜色，进行剪枝。

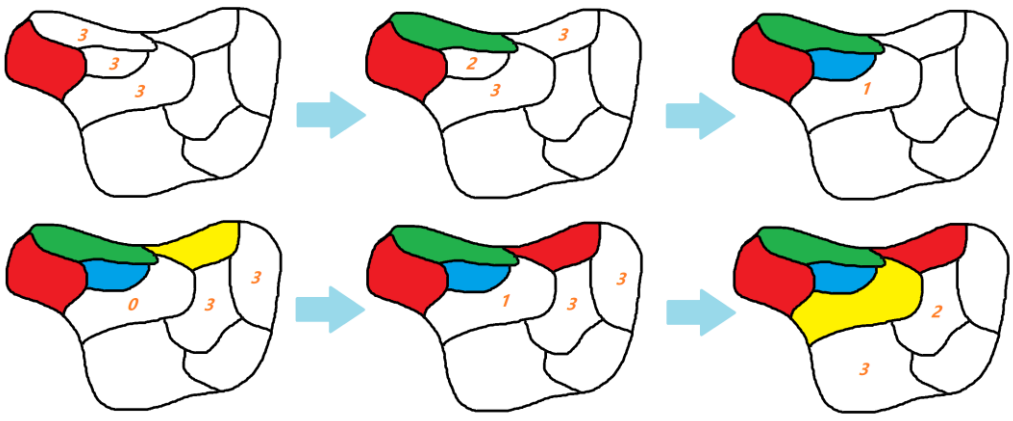

图 10 向前探测地图填色示例

伪代码

Algorithm 4 fillColor(vertex)

```
1: choseOneColor(vertex)
2: if ForwardChecking() and !conflict(vertex) then
3:     fillColor(nextvertex)
4: end if
```

运行结果

先在小规模地图上验证算法的正确性，如图 11 所示，可以在 412 毫秒内找出 480 个解。

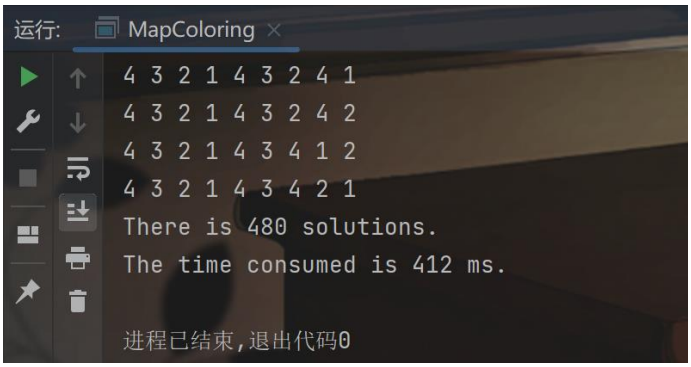

图 11 向前探测小规模地图填色

然后尝试填涂三个大规模地图，结果如表 6 所示。

向前探测			
地图	le450_5a	le450_15b	le450_25a
时间微秒	∞	∞	∞

表 6 向前探测大规模地图填色

由结果可知，单纯的向前探测无法在短时间内找出三个地图的解，下面我们将向前探测和最大度优先结合起来，填涂三个大规模地图，结果如表 7 所示。

向前探测+最大度优先			
地图	le450_5a	le450_15b	le450_25a
时间 (ms)	∞	∞	11

表 7 向前探测+最大度地图填色

再加上最少可选颜色优先，填涂三个大地图，结果如表 8 所示。

向前探测+最少可选颜色优先+最大度优先			
地图	le450_5a	le450_15b	le450_25a
时间 (ms)	780	4826	21

表 8 向前探测+最少可选颜色+最大度地图填色

对第一个地图找全部解，对第二个和第三个地图找 10 万个解，结果如表 9 所示。

向前探测+最少可选颜色优先+最大度优先			
地图	le450_5a	le450_15b	le450_25a
时间 (ms)	3840解130586	10万解7600	10万解4000

表 9 向前探测+最少可选颜色+最大度找多解

由此可知，与最大度优先和最少可选颜色优先相比，向前探测的优化效果不是特别明显。

3、随机产生不同规模的图，分析算法效率与图规模的关系（四色）。

（1）固定边

固定图的边数为 1000 条边，然后随机生成顶点数为 100 到 1000 的平面图，测试多组数据取众数，结果如图 12 所示。

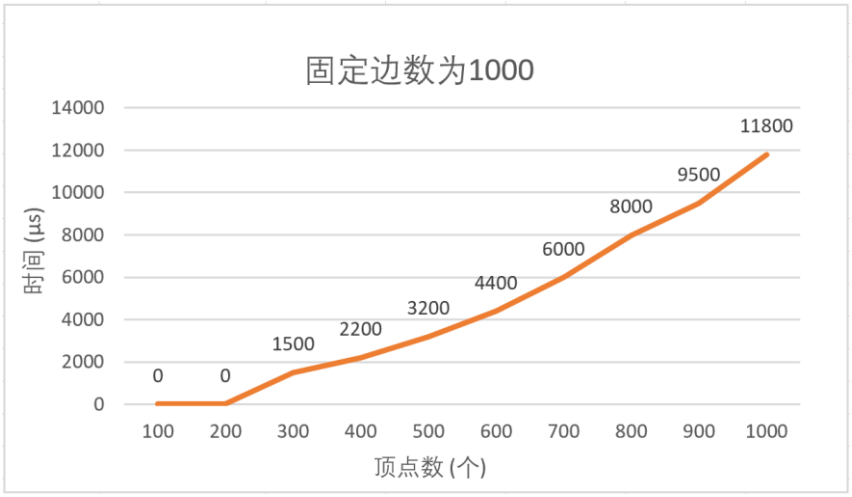

图 12 固定边为 1000 不同顶点数的地图填色

具体数据如表 10 所示。

固定边为1000										
顶点数	100	200	300	400	500	600	700	800	900	1000
时间 (μs)	--	--	1500	2200	3200	4400	6000	8000	9500	11800

表 10 固定边为 1000 不同顶点数的地图填色

由结果可知，边数固定的情况，顶点数越多，消耗的时间和资源也更多，解的搜索空间变大，搜索时间更长。

（2）固定点

固定图的顶点数为 100，随机生成边数为 100 到 1000 的平面图，测试多组数据取众数，结果如图 13 所示。

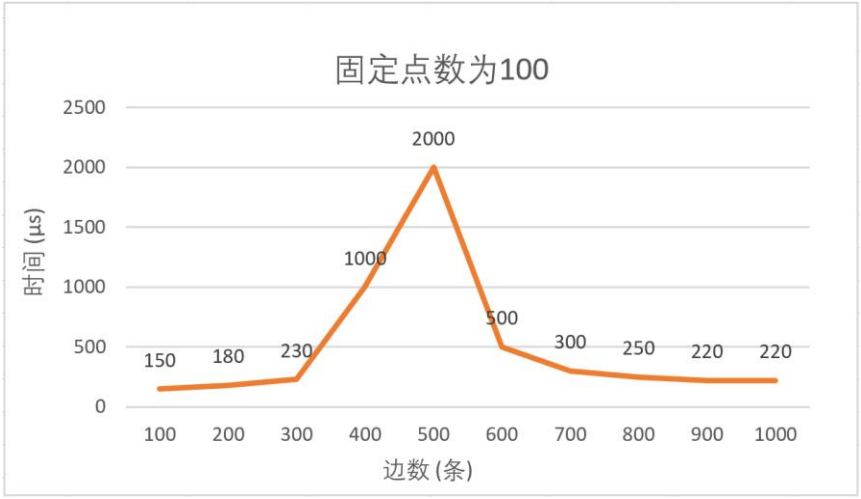

图 13 固定点为 100 不同边数的地图填色

具体数据如表 10 所示。

固定点为100										
边数	100	200	300	400	500	600	700	800	900	1000
时间 (μs)	150	180	230	1000	2000	500	300	250	220	220

表 10 固定点为 100 不同边数的地图填色

由结果分析，算法执行的时间先是随着边数的增加而增加，这是因为解的搜索空间增加了，而后当边数达到一定程度，边密度越大，图变得更加复杂，可选的颜色减少，算法剪枝的效率更高，所以搜索效率会更高。

三．实验心得

这次实验，我先是在小规模地图上验证了回溯法解决地图填色的正确性，然后在填涂三个大规模的地图时，使用最大度优先、最少可选颜色优先以及向前探测等策略对经典回溯法进行优化，大大提高了算法效率。实验的最后，我通过固定边数和固定顶点数来随机生成不同规模的图，分析了算法效率和图规模的关系，在边数固定的情况下，顶点数越多，算法的执行效率也就越低，而在顶点数固定的情况下，算法的执行效率会随着边数的增加先降低，当边密度增加到一定程度的时候，图的复杂程度进一步提高，剪枝的效果会更加明显，算法的执行效率会逐渐提高。

通过这次实验，我深刻认识到了算法优化对算法效率的重要性，通过算法的优化，使得原来可能在短时间内无解的问题能够迅速解决，同时也明确了针对不同的问题，需要考虑不同的优化方案。将理论知识与实践相结合，不断进行实验和调整，才能取得最优的实验结果，这对我的编程能力和自学能力都是一种很好的锤炼。

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。