# 

## 《算法分析理论及应用》课程实验报告S

**班级：软工182 姓名：邓棋 学号：2018081062**

**一、实验题目**

1. 图着色问题
2. 0/1背包问题。
3. 八皇后问题。

**二、实验内容**

**1. 图着色问题。（完成实验代码、伪代码）**

给定无向连通图G和m种不同的颜色。用这些颜色为图G的各顶点着色，每个顶点着一种颜色。如果有一种着色法使G中每条边的两个顶点着不同颜色，则称这个图是m可着色的。图的m着色问题是对于给定图G和m种颜色，找出所有不同的着色法。

【输入格式】第1行有3个正整数n、k和m，表示给定的图G有n个顶点和k条边，m种颜色。顶点编号为1，2，…，n。接下来的k行中，每行有两个正整数u、v，表示图G的一条边（u，v）。

【输出格式】程序运行结束时，将计算出的不同的着色方案数输出。如果不能着色，程序输出-1。

【输入样例】

4 4 3

1 2

1 3

1 4

2 4

【输出样例】

12

请写出算法时间复杂度、算法策略(基于回溯法)，算法伪代码以及代码实现。

**2. 0/1背包问题。（完成实验代码、伪代码）**

有n个重量分别为{w1，w2，…，wn}的物品，它们的价值分别为{v1，v2，…，vn}，给定一个容量为Q的背包。

设计从这些物品中选取一部分物品放入该背包的方案，每个物品要么选中要么不选中，要求选中的物品不仅能够放到背包中，而且重量和恰好为Q具有最大的价值。

0/1背包问题（Q=6）：

|  |  |  |
| --- | --- | --- |
| **物品编号** | **重量** | **价值** |
| **1** | **5** | **4** |
| **2** | **3** | **4** |
| **3** | **2** | **3** |
| **4** | **1** | **1** |

请写出算法时间复杂度、算法策略(基于回溯法)，算法伪代码以及代码实现。

**3. 八皇后问题。（完成实验代码、伪代码）**

八皇后问题是十九世纪著名的数学家高斯于1850年提出的。问题是：在8×8的棋盘上摆放八个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上。可以把八皇后问题扩展到n皇后问题，即在n×n的棋盘上摆放n个皇后，使任意两个皇后都不能处于同一行、同一列或同一斜线上。

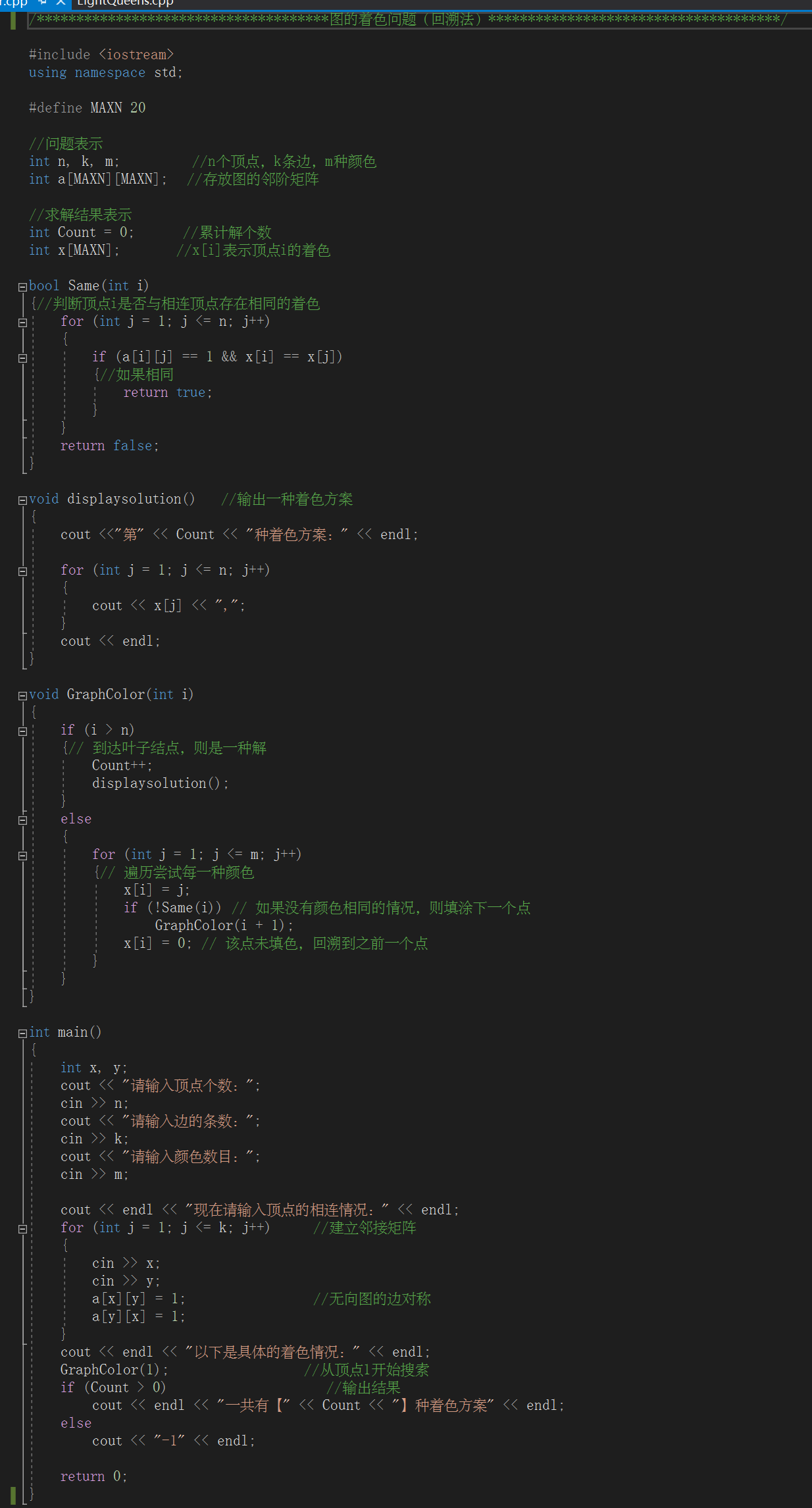
请写出算法时间复杂度、算法策略(基于回溯法)，算法伪代码以及代码实现。

**三、实验目的**

1. 理解回溯问题的思想，算法策略。

2. 掌握利用回溯解决问题的基本思想，会用高级语言对算法进行描述，并对算法复杂度（时间和空间）进行分析。

**四、实验代码**

**图着色问题：**

**0/1背包问题：**

**八皇后问题：**

**五、实验总结**

书写以下实验总结：1、算法伪代码编写；2、算法设计策略描述；3、算法时空复杂度分析。4、遇到的问题及解决方法。

1. **算法伪代码编写**

**图着色问题：**

1.输入无向图的顶点个数、边的条数、可染色的颜色数目，输入无向图的邻接矩阵

2.遍历

2.1 if （i > n ） //到达了叶子结点

输出结果

2.2 else

尝试遍历每一种颜色

2.2.1 if （！Same(i)） //如果没有颜色相同的情况

填涂下一个点

2.2.2 else 回溯到之前的一个点

3.输出结果

**0/1背包问题：**

1.输入物品数量、背包的承重量，输入每件物品的重量和价值

2.建立一个临时数组

3.遍历

3.1 if (i > n) //找到一个叶子结点

3.1.1 if 找到一个满足条件的更优解

更新已有信息，保存

* 1. else 尚未找完所有物品

3.2.1 if 满足剪枝条件

左孩子结点剪枝

3.2.2 else 不选取第i个物品

回溯

4.最后输出最佳装填方案。

**八皇后问题：**

1.输入n皇后问题的规模

2.位置变量i既表示当前行，也表示放置的第i个皇后，同时用一个数组x[]来记录皇后放置的列号

3.while (i >= 1) //尚未回溯到头，循环

3.1 x[i]++ //原位置后移一列

3.2 考察皇后i放置在x[i]列是否发生冲突

冲突的话继续后移一列

* 1. if 皇后还没有放置完

3.3.1 if 皇后刚好放置完了

输出一个解，并标记该规模下问题有解

3.3.2 else 皇后还没放置完

i++ 考虑新皇后的放置

3.4 第i个皇后找不到合适的位置，回溯到上一个皇后

4.标记无解，输出无解

5.标记有解，最后输出解的个数

1. **算法设计策略描述**

**图着色问题：**

图中的顶点编号为1～n，着色编号为1～m。对于图G中的每一个顶点，可能的着色为1～m，所以对应的解空间是一棵m叉树，高度为n，层次i从1开始。

回溯法求解图着色问题，首先把所有顶点的颜色初始化为0，然后依次为每个顶点着色。在图着色问题的解空间树中，如果从根结点到当前结点对应一个部分解，也就是所有的颜色指派都没有冲突，则在当前结点处选择第一棵子树继续搜索，也就是为下一个顶点着颜色1，否则，对当前子树的兄弟子树继续搜索，也就是为当前顶点着下一个颜色，如果所有m种颜色都已尝试过并且都发生冲突，则回溯到当前结点的父结点处，上一个顶点的颜色被改变，依此类推。

**0/1背包问题：**

用x[1..n]数组存放最优解。对第i层上的某个分枝结点，对应的状态为Knapsack(i，tw，tv，op)，其中tw表示装入背包中的物品总重量，tv表示背包中物品总价值，op记录一个解向量。该状态的两种扩展如下：

（1）选择第i个物品放入背包：op[i]=1，tw=tw+w[i]，tv=tv+v[i]，转向下一个状态Knapsack (i+1，tw，tv，op)。该决策对应左分枝。

（2）不选择第i个物品放入背包：op[i]=0，tw不变，tv不变，转向下一个状态Knapsack (i+1，tw，tv，op)。该决策对应右分枝。

叶子结点表示已经对n个物品做了决策，对应一个解。对所有叶子结点进行比较求出满足tw=W的最大tv(用maxv表示)，对应的最优解op存放到x中。

**八皇后问题：**

用数组x[]存放皇后的位置，x[i]表示第i个皇后放置的位置，n皇后问题的一个解是x[1]，x[2]，…，x[n]，数组x的下标从1开始。先放置第1个皇后，然后依2、3、…、n的次序放置其他皇后，当第n个皇后放置好后产生一个答案解。为了找所有解，还需要继续试探第n个皇后的下一个位置。第i（i<n）个皇后放置后，接着放置第i+1个皇后，在试探第i+1个皇后的位置时，都是从第1列开始的。当第i个皇后试探了所有列都不能放置时，则回溯到第i-1个皇后，此时与第i-1个皇后的位置x[i-1]有关，如果第i-1个皇后的列号小于n即x[i-1]<n，则将其移到下一列，继续试探；否则再回溯到第i-2个皇后，依此类推。若第1个皇后的所有位置回溯完毕，则算法结束。放置第i个皇后应与前面已经放置的i-1个皇后不发生冲突。

1. **算法时空复杂度分析**

**图着色问题：**

**时间复杂度：** O(mn)

**空间复杂度：** O(n)

**0/1背包问题：**

**时间复杂度：** O(2n)

**空间复杂度：** O(n)

**八皇后问题：**

**时间复杂度：**O(n!)

**空间复杂度：**O(n)

1. **遇到的问题及解决方法**

无。

**六、算法策略的英文描述（字数>200）**

**Graph coloring problem:**

The vertices in the figure are numbered 1 to n, and the coloring numbers are 1 to m.For each vertex in the graph G, the possible coloring is 1 to m, so the corresponding solution space is an m-ary tree, the height is n, and the level i starts from 1.

Backtracking solves the problem of graph coloring, first initializes the colors of all vertices to 0, and then colors each vertex in turn.In the solution space tree of the graph coloring problem, if there is a partial solution from the root node to the current node, that is, all color assignments do not conflict, then the first subtree is selected at the current node to continue searching, that is, Color 1 for the next vertex, otherwise, continue searching for the siblings of the current subtree, that is, the next color for the current vertex. If all m colors have been tried and all have conflicts, then back to the current node At the parent node of, the color of the previous vertex is changed, and so on.

**0/1 backpack problem:**

Use x [1..n] array to store the optimal solution.

For a branch node on the i-th layer, the corresponding state is Knapsack (i, tw, tv, op), where tw represents the total weight of the items in the backpack, tv represents the total value of the items in the backpack, op records A solution vector.

The two extensions of this state are as follows:

(1) Select the i-th item into the backpack: op [i] = 1, tw = tw + w [i], tv = tv + v [i], and go to the next state Knapsack (i + 1, tw, tv, op).

This decision corresponds to the left branch.

(2) Do not select the i-th item in the backpack: op [i] = 0, tw is not changed, tv is not changed, and the next state is Knapsack (i + 1, tw, tv, op).

This decision corresponds to the right branch.Leaf nodes indicate that decisions have been made for n items, corresponding to one solution.Compare all leaf nodes to find the maximum tv (represented by maxv) that satisfies tw = W, and the corresponding optimal solution op is stored in x.

**Eight queens problem:**

Use the array x [] to store the position of the queen, and x [i] represents the position where the ith queen is placed. One solution to the n-queens problem is x [1], x [2], ..., x [n], and the array x Subscripts start at 1. Place the first queen first, then place other queens in the order of 2, 3, ..., n. When the nth queen is placed, an answer solution is generated. In order to find all the solutions, it is necessary to continue to test the next position of the nth queen. After the i (i <n) queen is placed, then the i + 1 queen is placed. When testing the position of the i + 1 queen, it starts from the first column. When the i-th queen tried that all columns could not be placed, it traced back to the i-1th queen, at this time related to the position x [i-1] of the i-1 queen, If the column number of is less than n, that is, x [i-1] <n, then move it to the next column and continue testing; otherwise, go back to the i-2th queen, and so on. If all positions of the first queen have been traced back, the algorithm ends. The i-th queen should be placed without conflict with the i-1 queen that has been placed before.