# 

## 《算法分析理论及应用》课程实验报告

**班级：软工182 姓名：邓棋 学号：2018081062**

**一、实验题目**

1. 图着色问题
2. 普里姆算法最小生成树问题。
3. 0/1背包问题。
4. 活动兼容问题。

**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=(V, E)，求图G的最小色数k，使得用k种颜色对G中的顶点着色，可使任意两个相邻顶点着色不同。

描述：

（1）任选一顶点着颜色1，在图中寻找尽可能多的顶点用颜色1着色；

（2）选取不能用颜色1着色的顶点，用颜色2着色，在图中寻找尽可能多的顶点用颜色2着色；

（3）直到所有顶点都被着色停止算法。

请写出算法时间复杂度、算法策略，算法伪代码以及代码实现。

**2. Prim算法最小生成树问题。（完成实验代码、伪代码）**

1）以某一个点开始，寻找当前该点可以访问的所有的边；

2）在已经寻找的边中发现最小边，这个边必须有一个点还没有访问过，将还没有访问的点加入我们的集合，记录添加的边；

3）寻找当前集合可以访问的所有边，重复2的过程，直到没有新的点可以加入；

4）此时由所有边构成的树即为最小生成树。

请解释Prim算法是如何采用贪心策略的，请写出算法时间复杂度、算法策略，算法伪代码以及代码实现。

**3. 0/1背包问题。（完成实验代码、伪代码）**

设有编号为1、2、…、n的n个物品，它们的重量分别为w1、w2、…、wn，价值分别为v1、v2、…、vn，其中wi、vi（1≤i≤n）均为正数。有一个背包可以携带的最大重量不超过W。

求解目标：在不超过背包负重的前提下，使背包装入的总价值最大（即效益最大化），与0/1背包问题的区别是，这里的每个物品可以取一部分装入背包。

请写出算法时间复杂度、算法策略，算法伪代码以及代码实现。

**4. 活动兼容问题。（完成实验代码、伪代码）**

假设有一个需要使用某一资源的n个活动所组成的集合S，S={1，…，n}。该资源任何时刻只能被一个活动所占用，活动i有一个开始时间bi和结束时间ei（bi<ei），其执行时间为ei-bi，假设最早活动执行时间为0。一旦某个活动开始执行，中间不能被打断，直到其执行完毕。若活动i和活动j有bi≥ej或bj≥ei，则称这两个活动兼容。

设计算法求一种最优活动安排方案，使得所有安排的活动个数最多。

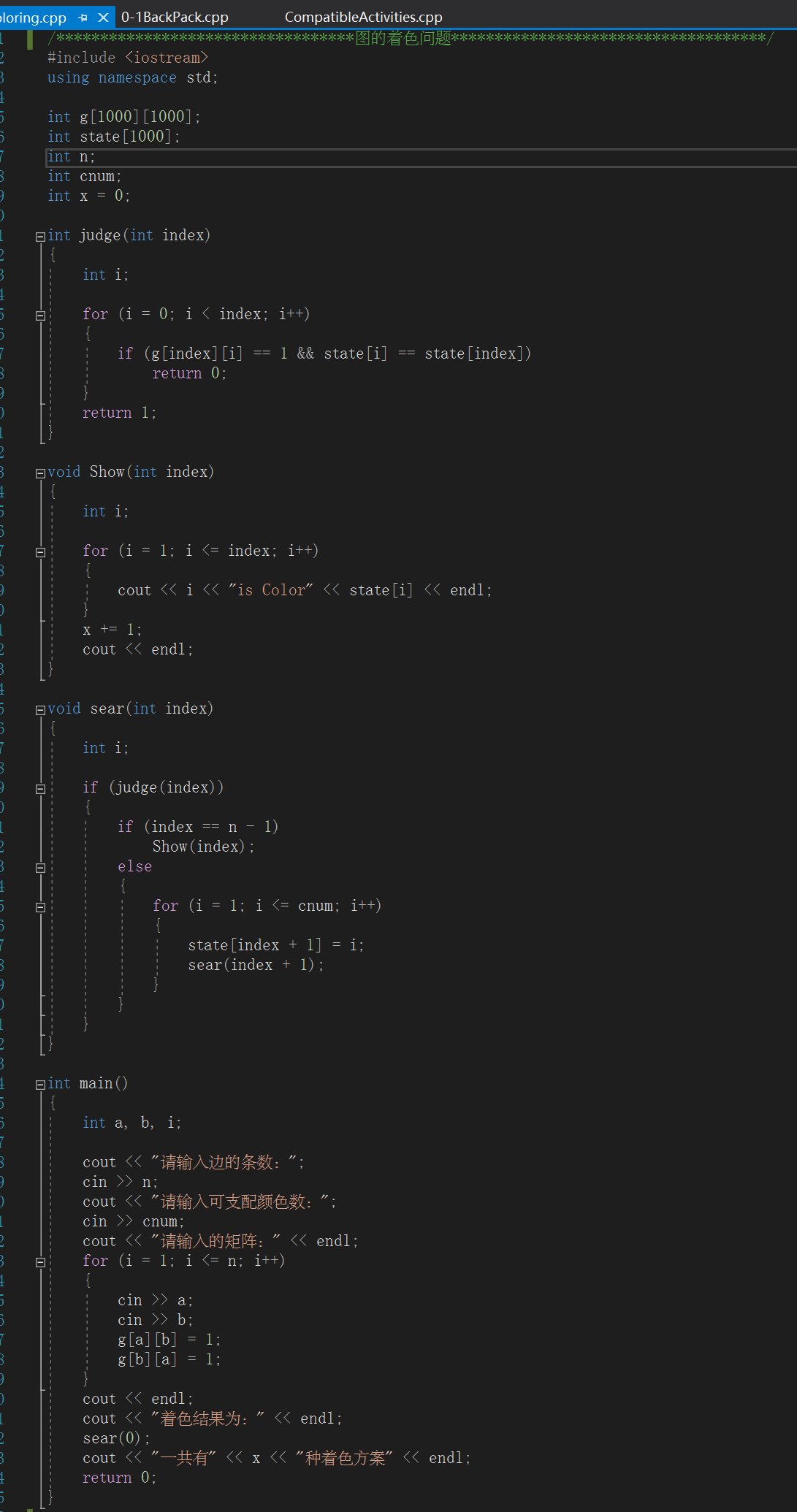
请写出算法时间复杂度、算法策略，算法伪代码以及代码实现。

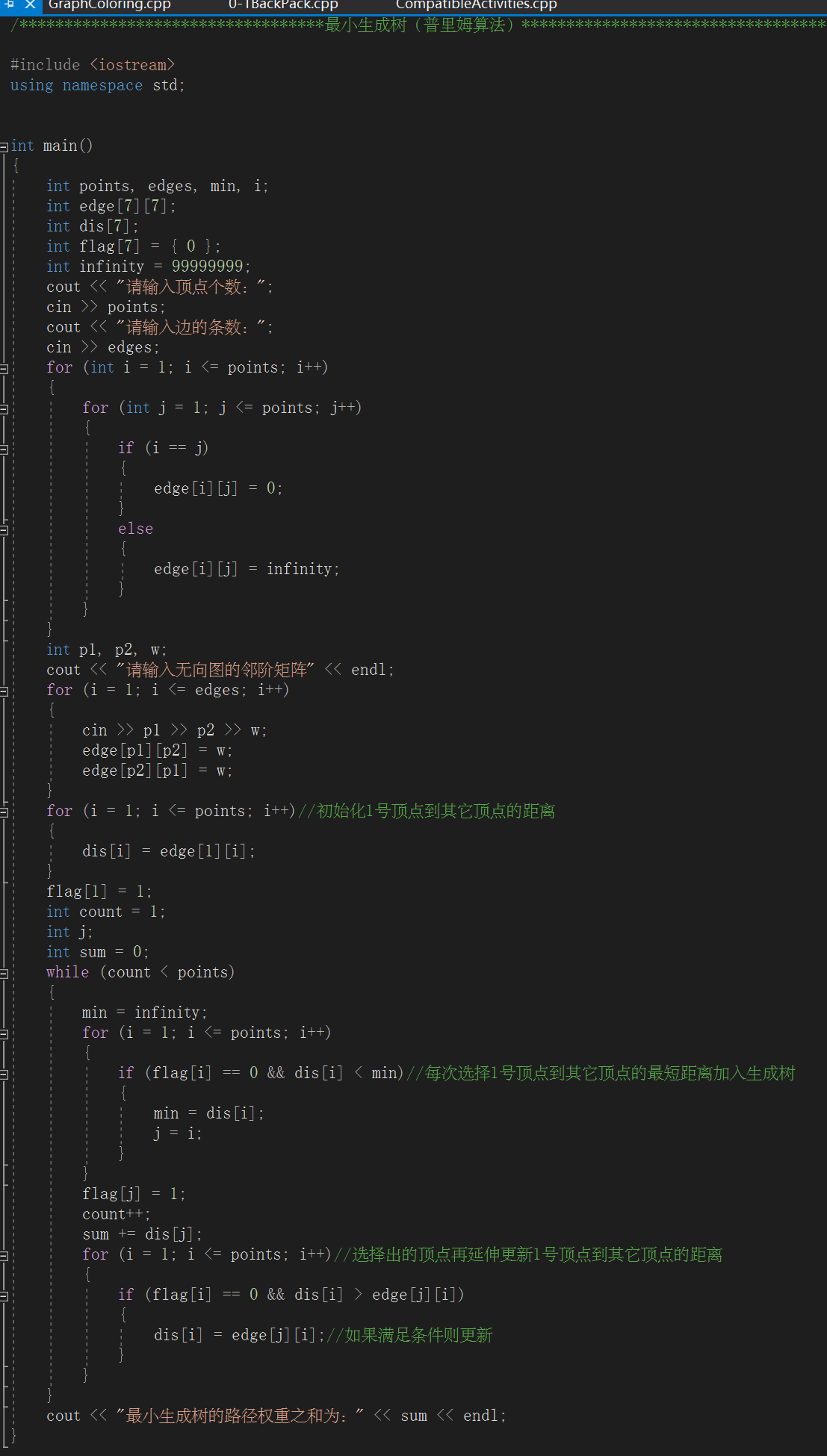
**三、实验目的**

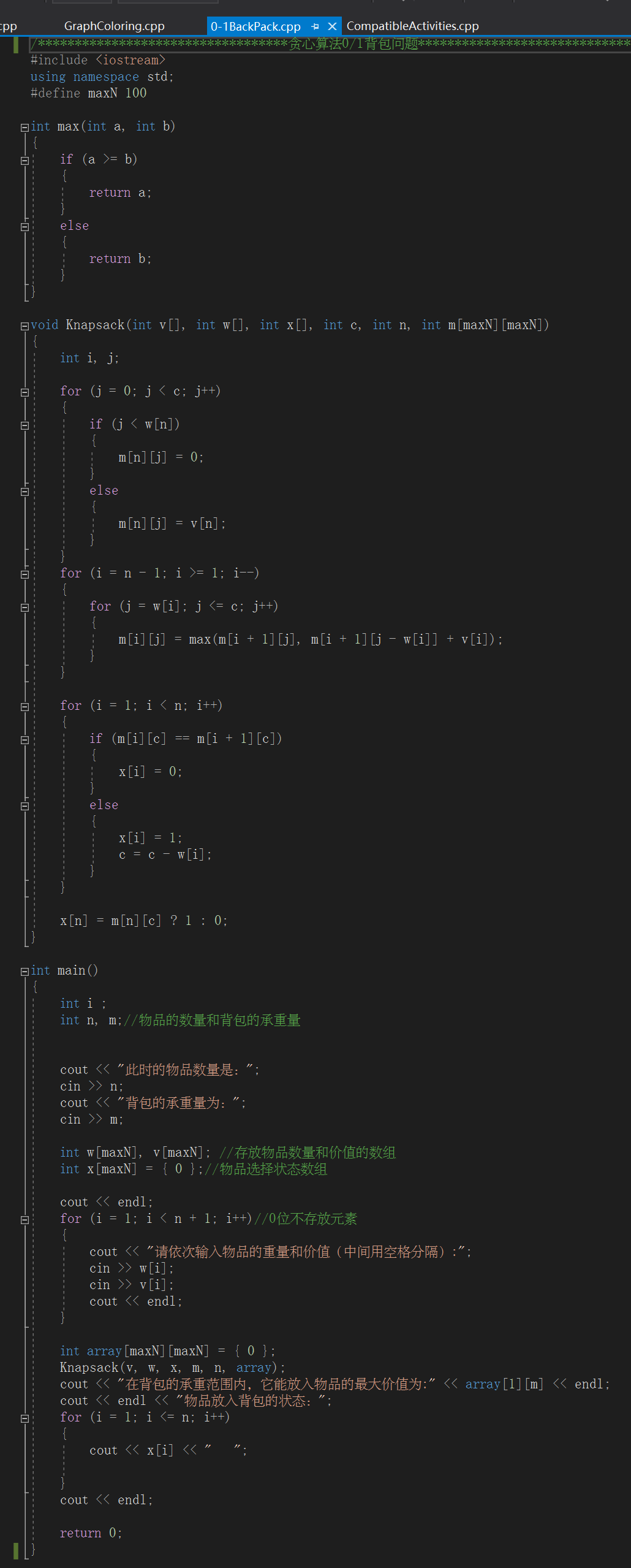
1. 理解贪心问题的思想，算法策略。

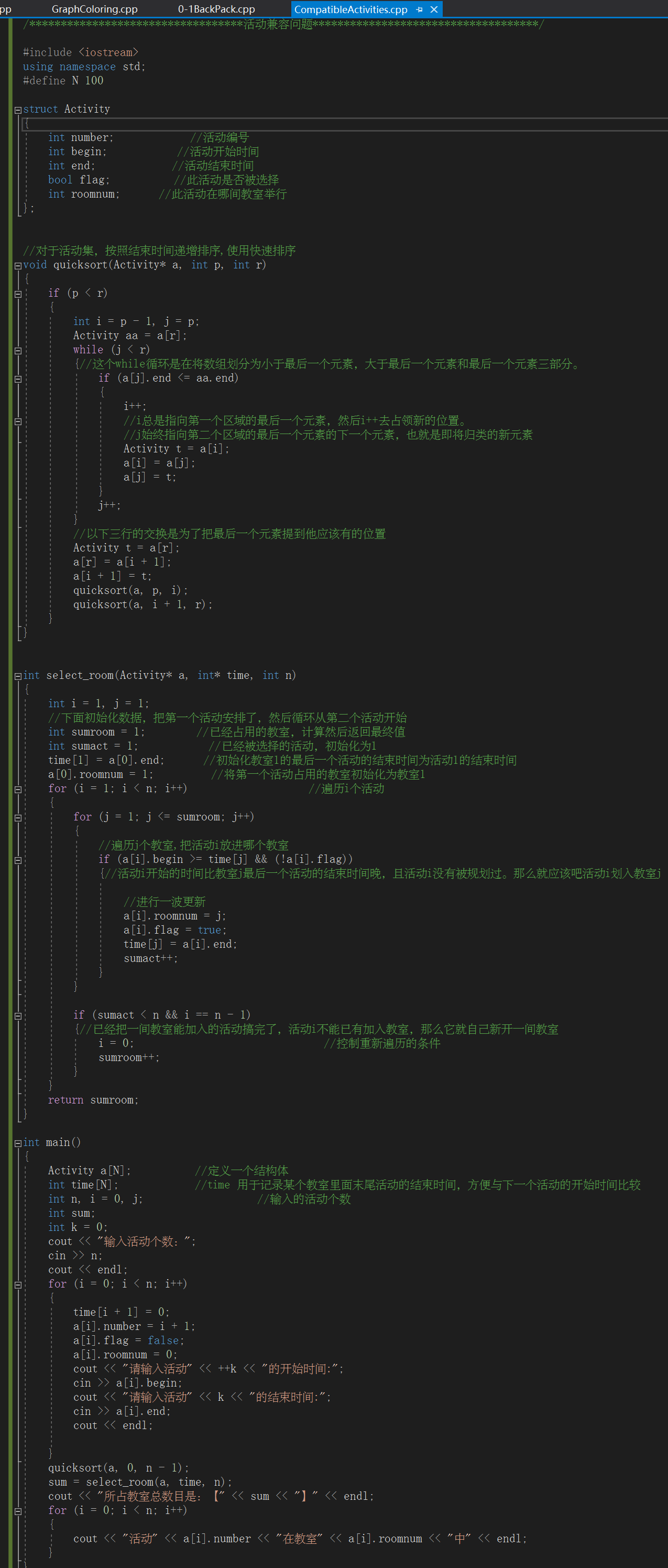
2. 掌握利用贪心解决问题的基本思想，会用高级语言对算法进行描述，并对算法复杂度（时间和空间）进行分析。

**四、实验代码**

**图着色问题：**

**Prim算法最小生成树问题**

**0/1背包问题：**

**活动兼容问题：**

**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.color[1] = 1; //定点1着颜色1

2.for (i = 2; i <= n; i++) //其他所有定点置未着色状态

Color[i] = 0;

3.k = 0;

4.循环直到所有顶点都着色

4.1 k++; //取下一个颜色

4.2 for (i = 2; i <= n; i++) //用颜色k为尽量多的顶点着色

4.2.1 若顶点i已着色，则转步骤4.2，考虑下一个顶点；

4.2.2 若图中与顶点i邻接的顶点着色与顶点i着色不冲突，则color[i] = k;

5.输出k;

**Prim算法最小生成树问题：**

1.输入顶点个数和无向图边的条数

2.for i = 1 to 顶点个数：

for j = 1 to 顶点个数：

将邻接矩阵对角线上的数字都标记为0

其他位置都标记为infinity

3.输入无向图的邻接矩阵及其权值，将有联系的点在邻接矩阵上的值直接表示为它的权值，然后对于它的转置矩阵赋予同样的值。

4. 初始化1号顶点到其它顶点的距离

5.只要顶点没有遍历完，就一直处于循环状态

每次选择1号顶点到其它顶点的最短距离加入生成树

遍历过的顶点用一个flag标记一下，记录此时邻接矩阵上的距离权值

选择出的顶点再延伸更新1号顶点到其它顶点的距离

6.输出记录下的所有距离权值之和

**0/1背包问题：**

1.输入物品的水量和背包的最大承重，再依次输入物品的重量和价值

2.改变数组w和v的排列顺序，使其按单位重量价值v[i]/w[i]降序排列;

3.将数组x[n]初始化为0; //初始化解向量

4.i = 1;

5.循环直到(w[i] > weight);

5.1 x[i] = 1; //将第i个物品放入背包

5.2 weight = weight – w[i];

5.3 i++;

6.x[i] = weight / w[i];

**活动兼容问题：**

1.定义一个结构体存放活动的相关信息，输入活动的个数，依次输入活动的开始和结束时间；

2. 对于活动集，按照结束时间使用快速排序进行递增排序；

3.依次遍历排序后的活动

for (i = 1; i < n; i++)

3.1 for (j = 1; j <= sumroom; j++)

遍历j个教室,把活动i放进哪个教室

活动i开始的时间比教室j最后一个活动的结束时间晚，且活动i没有被规划过。那么就应该吧活动i划入教室j

3.2 已经把一间教室能加入的活动搞完了，活动i不能已有加入教室，那么它就自己新开一间教室

4.返回所用的教室总数，并依次打印活动在哪个教室举办

1. **算法设计策略描述**

**图着色问题：**

选择一种颜色，以任意顶点作为开始顶点，依次考察图中的未被着色的每个顶点，如果一个顶点可以用颜色1着色，换言之，该顶点的邻接点都还未被着色，则用颜色1为该顶点着色，当没有顶点能以这种颜色着色时，选择颜色2和一个未被着色的顶点作为开始顶点，用第二种颜色为尽可能多的顶点着色，如果还有未着色的顶点，则选取颜色3并为尽可能多的顶点着色，以此类推。

**Prim算法最小生成树问题：**

Prim算法使生成树以一种自然的方式生长，即从任意顶点开始，每一步为这棵树添加一个分枝，直到生成树中包含全部顶点。设连通网中有n个顶点，则第一个进行初始化的循环语句需要执行n-1次，第二个循环共执行n-1次，内嵌两个循环，其一是在长度为n的数组中求最小值，需要执行n-1次，其二是调整辅助数组，需要执行n-1次。

**0/1背包问题：**

应用贪心策略，每次从物品集合中选择单位重量价值最大的物品，如果其重量小于背包重量，就可以把它装入，并将背包容量减去该物品的重量，然后我们就面临了一个最优子问题——它同样是背包问题，只不过背包容量减少了，物品集合减少了。因此背包问题具有最优子结构性质。

**活动兼容问题：**

总是选择这样一个活动来占用资源，它能够使得余下的未调度的时间最大化，使得兼容的活动尽可能多。因此，将活动按结束时间递增排序，每次总是选择最早结束的兼容活动。然后再比较活动开始的时间，在如果时间不重合，就不会再使用新的教室。

1. **算法时空复杂度分析**

**图着色问题：**

**时间复杂度：** O（nlog2n）

**空间复杂度：** O（n）

**Prim算法最小生成树问题：**

**时间复杂度：** O（n^2）

**空间复杂度：** O（n）

**0/1背包问题：**

**时间复杂度：** O(nlog2n)

**空间复杂度：** O（n）

**活动兼容问题：**

**时间复杂度：** O(nlog2n)

**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:**

Choose a color, with any vertex as the starting vertex, and examine each uncolored vertex in the graph in turn. If a vertex can be colored with color 1, in other words, the adjacent points of the vertex have not been colored, use color 1 color the vertex. When no vertex can be colored in this color, select color 2 and an uncolored vertex as the starting vertex. Use the second color to color as many vertices as possible. Vertices, pick color 3 and color as many vertices as possible, and so on.

**Prim algorithm minimum spanning tree proble**m:

The Prim algorithm makes the spanning tree grow in a natural way, starting from any vertex and adding a branch to this tree at each step until the spanning tree contains all the vertices. Assuming there are n vertices in the connected network, the first loop statement to be initialized needs to be executed n-1 times, the second loop is executed n-1 times in total, and two loops are embedded, one of which is n To find the minimum value in an array, it needs to be performed n-1 times. The second is to adjust the auxiliary array, which needs to be performed n-1 times.

**0/1 backpack problem:**

Applying the greedy strategy, each time you select the item with the highest value per unit weight from the collection of items, if its weight is less than the weight of the backpack, you can load it and subtract the weight of the item from the capacity of the backpack, and then we face Yuko problem-it is also a backpack problem, but the capacity of the backpack is reduced and the collection of items is reduced. Therefore, the knapsack problem has optimal substructure properties.

**Activities compatible problem:**

Always choose such an activity to occupy resources, it can maximize the remaining unscheduled time and make as many compatible activities as possible. Therefore, the activities are sorted in ascending end time, and always the earliest compatible activity is selected. Then compare the start time of the event. If the time does not coincide, the new classroom will not be used again.