实验 6 算法设计与实现（2）

1. 实验目的

（1）了解比较复杂的算法设计

（2）掌握用 Python 语言去实现算法。

2. 实验任务

实验任务 6-1 任选课堂上所讲授的一种排序算法，编写一个排序的函数

调用你写的这个函数，能够轻松实现对一年之内北京市日均最高气温、日均最低气温、平均降水量等项数据按数值大小排序输出。

表：

全年平均气温	一月	二月	三月	四月	五月	六月	七月	八月	九月	十月	十一月	十二月
日均最高气温(℃)	2℃	5℃	12℃	20℃	26℃	30℃	31℃	30℃	26℃	19℃	10℃	3℃
日均最低气温(℃)	-9℃	-6℃	0℃	8℃	14℃	19℃	22℃	21℃	15℃	8℃	0℃	-6℃
平均降水总量(mm)	3	6	9	22	36	74	179	177	53	23	8	2
平均降水天数(天)												

实验任务 6-20/1 背包问题

背包问题(Knapsack problem)是一种组合优化的 NP 完全问题。问题可以描述为：给定一组物品，每种物品都有自己的重量和价格，在限定的总重量内，我们如何选择，才能使得物品的总价格最高。问题的名称来源于如何选择最合适的物品放置于给定背包中。相似问题经常出现在商业、组合数学，计算复杂性理论、密码学和应用数学等领域中。

下面请你编辑并运行求解 0/1 背包问题的程序。

本程序要解决的问题是一个背包的承重量为 10，有五件物品的重量分别是 2、2、6、5、4，这五件物品的价值分别是 6、3、5、4、6。按照要求挑选几件物品装入背包，满足以下条件：（1）放入背包的物品总重要不能超重（2）装入背包的物品的价值最大且不能分割物品中。

实验指导：

（1）本程序中，令 Load[i][j]表示在前 i 个物体中，能够装入承重为 j 的背包中的物品的总价值，j=1,2,3,…,m

（2）表达式 Load[i][0]= Load[0][j]表示:

承重为 0 的背包装入前面 i 个物品，或者 0 个物品装入承重为 j 的背包，背包中物品的价值都是 0

（3）表达式 Load[i][j]= Load[i-1][j]（$j<w_i$）表示：

如果第 i 个物品的重量超过了背包承重，则装入前面 i 个物品与装入前面 i-1 个物品的总价值是相同的（因第 i 个物品并未能装入）

（4）表达式 Load[i][j]= max{Load[i-1][j], Load[i-1][j-w_i]+p_i }(j>=w_i)

表示：当第 i 个物品的重量小于背包承重时（可以装入），如果将它装入承重为 j 的背包后总价值上升才装入，否则作不装入的决策。

```
#6-2
#动态规划法解 0-1 背包问题，定义函数 zeroOneknapsack
def zeroOneknapsack(w,p,m,x):
请将函数补充完整
    return v

m=10
w=[0,2,2,6,5,4]
p=[0,6,3,5,4,6]
#计算 n 的个数
n=len(w)-1
#初始化 x 列表，该列表表示每个物品是否装入背包的状态，初始状态的时候均为“未装”
x=["未装" for raw in range(n+1)]
#调用函数解背包问题
totalV=zeroOneknapsack(w,p,m,x)
#输出结果
print("装入背包中物品的总价值：",totalV)
print("物品是否装入背包的状态：",x[1:])
```

实验 6-3 （选作）四色问题

有形如下列图形的地图，图中每一块区域代表一个省份，现请你用红（1）、兰（2）、黄（3）、绿（4）四种颜色给这些省份填上颜色，要求每一省份用一种颜色，且任意两个相邻省份的颜色不能相同，请给出一种符合条件的填色方案。地图用无向图的形式给出，每个省份代表图上的一个顶点，边代表两个省份是相邻的。

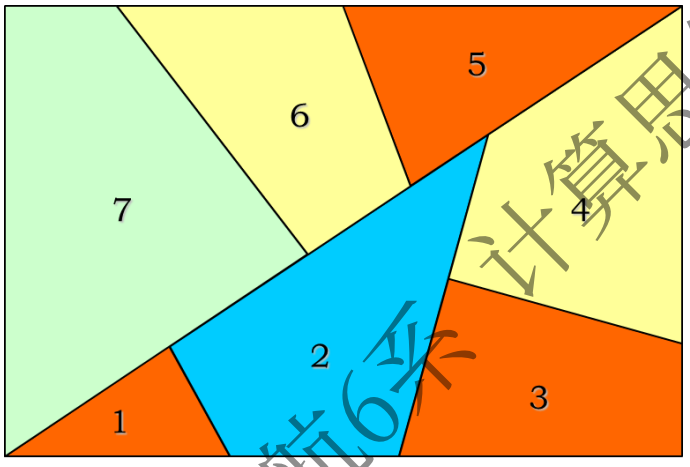
输入

0 1 0 0 0 0 1
1 0 1 1 1 1 1
0 1 0 1 0 0 0
0 1 1 0 1 0 0
0 1 0 1 0 1 0
0 1 0 0 1 0 1
1 1 0 0 0 1 0

输出

1 2 1 3 1 3 4

请你编写一个程序，解决该涂色问题



提示：

要得到一种可行的填色方案，比较直接的做法就是一个省一个省的填色，在填色的过程中去检查是否和周围省份的颜色存在冲突。

从第一个省份开始，对每个省份依次尝试四种颜色，判断当然选择的颜色是否和相邻的省份相同。

若都不相同，则将此省份填上当前的颜色。

若存在相同，则取下一种颜色继续尝试。

若四种颜色都不能填上，则退回上一个省份并选择下一种颜色继续尝试。

当最后一个省份也被填色后就找到一组可行解。

四色问题的解空间是所有的填色方案。

实验指导：

（1）用图论法对该问题进行建模，将该地图视为无向图，对其用邻接矩阵进行表示，其中 0 表示两省份不相邻，1 表示两省份相邻。

0 1 0 0 0 0 1
1 0 1 1 1 1 1
0 1 0 1 0 0 0
0 1 1 0 1 0 0
0 1 0 1 0 1 0
0 1 0 0 1 0 1
1 1 0 0 0 1 0

用嵌套的列表存储该矩阵

（2）参考程序（解法不唯一，欢迎大家补充解法）

```
#6-3
#定义函数 deal
def deal(mat, num, result, index):
请补充代码

if __name__=="__main__":
    #mat 为方阵，mat[i][j]=1 表示城市 i 与城市 j 相邻，=0 表示不相邻
    mat=[[0,1,0,0,0,0,1],
         [1,0,1,1,1,1,1],
         [0,1,0,1,0,0,0],
         [0,1,1,0,1,0,0],
         [0,1,0,1,0,1,0],
         [0,1,0,0,1,0,1],
         [1,1,0,0,0,1,0]]
    num=len(mat)
    result = []
    index=0
  print(deal(mat, num, result, index))#num 为方阵的 row、col，result 为结果列表，index 为当前处理到的城市标号
```