**暨南大学本科实验报告专用纸**

课程名称 算法分析与设计实验 成绩评定

实验项目名称 二维0-1背包问题 指导教师 李展

实验项目编号 实验七 实验项目类型 综合性 实验地点

学生姓名 张印祺 学号 2018051948

学院 信息科学技术 系 计算机科学 专业 网络工程

实验时间 2020 年 4 月 29 日

1. 问题描述

给定n个物品和一个背包。物品i的重量是wi，体积是bi，其价值为vi，背包容量为c，容积为d。问应如何选择装入背包中的物品，使得装入背包中物品的总价值最大？其中物品只能选择放和不放。

1. 算法思路

该问题可以迅速列出限制条件：

s.t. ;

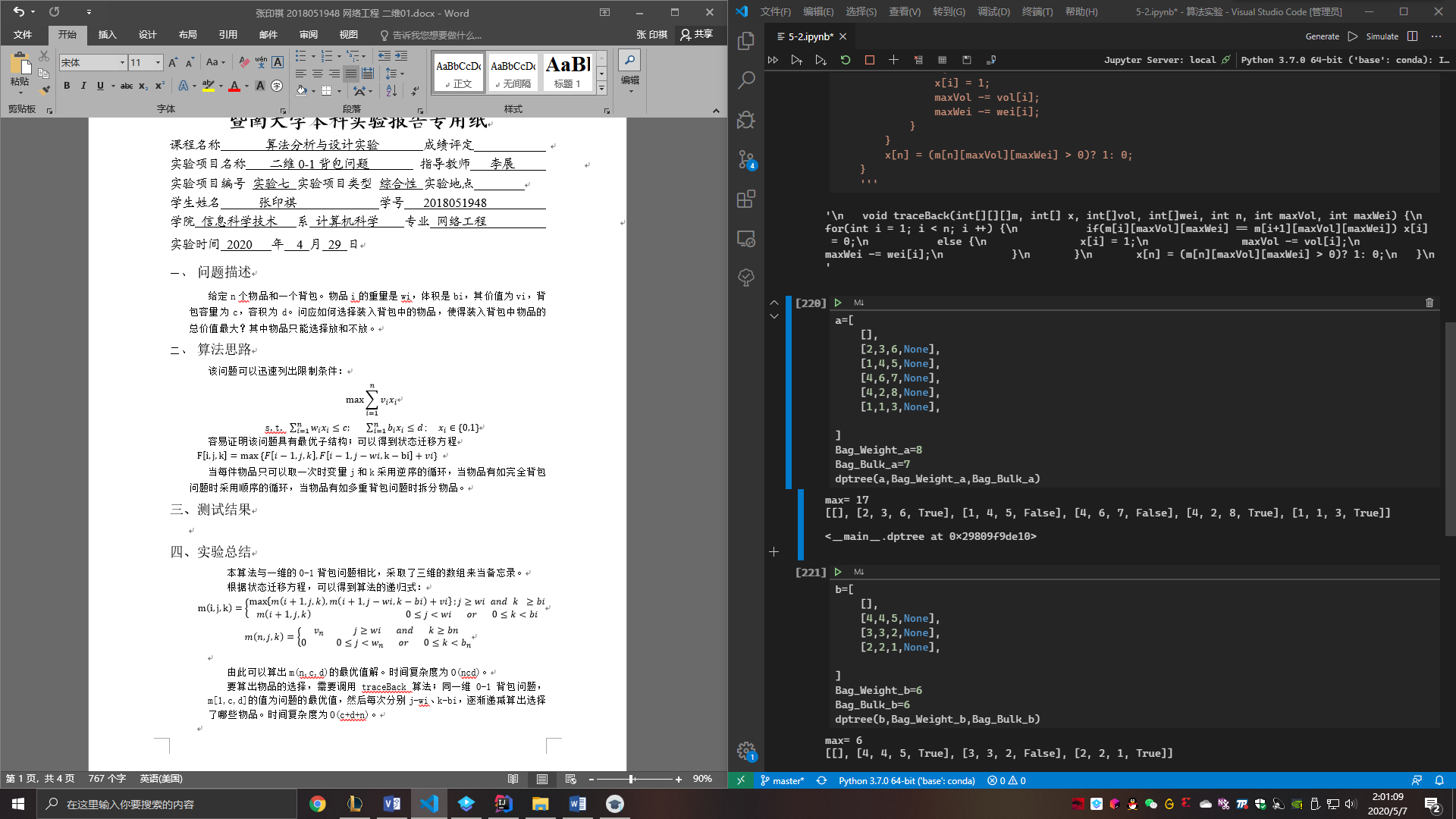
容易证明该问题具有最优子结构；可以得到状态迁移方程

当每件物品只可以取一次时变量j和k采用逆序的循环，当物品有如完全背包问题时采用顺序的循环，当物品有如多重背包问题时拆分物品。

1. 流程图



1. 测试结果



1. 实验总结

本算法与一维的0-1背包问题相比，采取了三维的数组来当备忘录。

根据状态迁移方程，可以得到算法的递归式：

由此可以算出m(n,c,d)的最优值解。时间复杂度为O(ncd)。

要算出物品的选择，需要调用traceBack算法；同一维0-1背包问题，m[1,c,d]的值为问题的最优值，然后每次分别j-wi、k-bi，逐渐递减算出选择了哪些物品。时间复杂度为O(c+d+n)。

六、附录 （程序代码）

'''

a [weight,bulk,val,choose]

'''

class dptree:

    def \_\_init\_\_(self,a:list,MAX\_weight:int,MAX\_bulk:int):

        MAX\_weight+=1

        MAX\_bulk+=1

        Bag=[[[0 for i in range(MAX\_weight+1)]for j in range(MAX\_bulk+1)] for k in range(len(a))]

        n=len(a)-1

        for i in range(a[n][1],MAX\_bulk+1):

            for j in range(a[n][0],MAX\_weight+1):

                Bag[n][i][j]=a[n][2]

        for i in range(len(a)-2,0,-1):

            bulkmax=min(a[i][1]-1,MAX\_bulk)

            weightmax=min(a[i][0]-1,MAX\_weight)

            for j in range(1,bulkmax+1):

                for w in range(1,MAX\_weight+1):

                    Bag[i][j][w]=Bag[i+1][j][w]

            for j in range(bulkmax,MAX\_bulk+1):

                for w in range(1,weightmax+1):

                    Bag[i][j][w]=Bag[i+1][j][w]

            for j in range(a[i][1],MAX\_bulk):

                for w in range(a[i][0],MAX\_weight):

                    Bag[i][j][w]=max(Bag[i+1][j][w],Bag[i+1][j-a[i][1]][w-a[i][0]]+a[i][2])

        print("max=",Bag[1][MAX\_bulk-1][MAX\_weight-1])

        self.traceback(Bag,a,MAX\_bulk,MAX\_weight,n)

    def traceback(self,Bag:list,a:list,MAX\_bulk,MAX\_weight,n):

        for i in range(1,len(a)-1):

            if Bag[i][MAX\_bulk-1][MAX\_weight-1]==Bag[i+1][MAX\_bulk-1][MAX\_weight-1]:

                a[i][3]=False

            else:

                a[i][3]=True

                MAX\_bulk-=a[i][1]

                MAX\_weight-=a[i][0]

        if Bag[n][MAX\_bulk][MAX\_weight]>0:

            a[n][3]=True

        else:

            a[n][3]=False

        print(a)

#测试用例

a=[

    [],

    [2,3,6,None],

    [1,4,5,None],

    [4,6,7,None],

    [4,2,8,None],

    [1,1,3,None],

]

Bag\_Weight\_a=8

Bag\_Bulk\_a=7

dptree(a,Bag\_Weight\_a,Bag\_Bulk\_a)

b=[

    [],

    [4,4,5,None],

    [3,3,2,None],

    [2,2,1,None],

]

Bag\_Weight\_b=6

Bag\_Bulk\_b=6

dptree(b,Bag\_Weight\_b,Bag\_Bulk\_b)