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

一、 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。

二、 算法思路

该问题可以迅速列出限制条件：

$$\max\sum_{i=1}^{n} v_i x_i$$

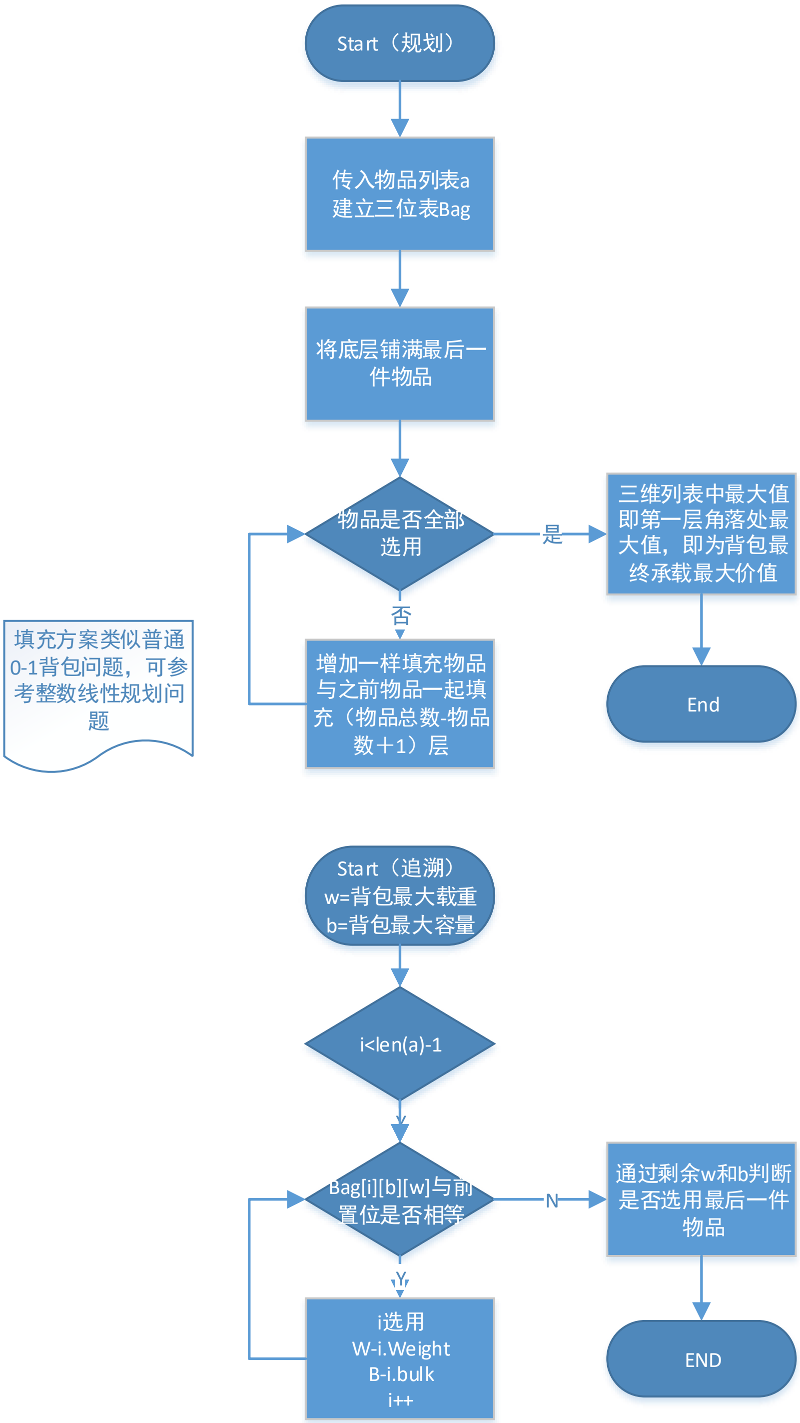
$$\text{s.t.}\ \sum_{i=1}^{n} w_i x_i \le c;\quad \sum_{i=1}^{n} b_i x_i \le d;\quad x_i \in \{0,1\}$$

容易证明该问题具有最优子结构；可以得到状态迁移方程

$$\mathrm{F[i,j,k]} = \max\{F[i-1,j,k], F[i-1,j-wi,\mathrm{k}-\mathrm{bi}] + vi\}$$

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。

三、流程图



Start（规划）
传入物品列表a
建立三位表Bag
将底层铺满最后一件物品
物品是否全部选用
是
三维列表中最大值即第一层角落处最大值，即为背包最终承载最大价值
否
增加一样填充物品与之前物品一起填充（物品总数-物品数＋1）层
End
填充方案类似普通0-1背包问题，可参考整数线性规划问题
Start（追溯）
w=背包最大载重
b=背包最大容量
i<len(a)-1
Bag[i][b][w]与前置位是否相等
N
通过剩余w和b判断是否选用最后一件物品
Y
i选用
W-i.Weight
B-i.bulk
i++
END

四、测试结果

```
[220]
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)
```

```
max= 17
[[], [2, 3, 6, True], [1, 4, 5, False], [4, 6, 7, False], [4, 2, 8, True], [1, 1, 3, True]]

<__main__.dptree at 0x29809f9de10>
```

```
[221]
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)
```

```
max= 6
[[], [4, 4, 5, True], [3, 3, 2, False], [2, 2, 1, True]]
```

五、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：

$$\mathrm{m(i,j,k)} = \begin{cases} \max\{m(i+1,j,k), m(i+1,j-wi,k-bi)+vi\}; j \geq wi \ \ and \ \ k \ \geq bi \\ \quad m(i+1,j,k) \qquad\qquad\qquad\qquad 0 \leq j < wi \quad or \quad 0 \leq k < bi \end{cases}$$

$$m(n,j,k) = \begin{cases} \quad v_n \qquad j \geq wi \quad and \quad k \geq bn \\ 0 \qquad 0 \leq j < w_n \quad or \quad 0 \leq k < b_n \end{cases}$$

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)
```