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 整数规划问题 指导教师 李展

实验项目编号 实验 X 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

考虑下面的整数线性规划问题：

$$\max \Sigma_{i=1}^{n} c_i x_i$$

$$\text{s.t.}\ \Sigma_{i=1}^{n} a_i x_i \le b;\quad x_i \text{为非负整数}，1 \le i \le n$$

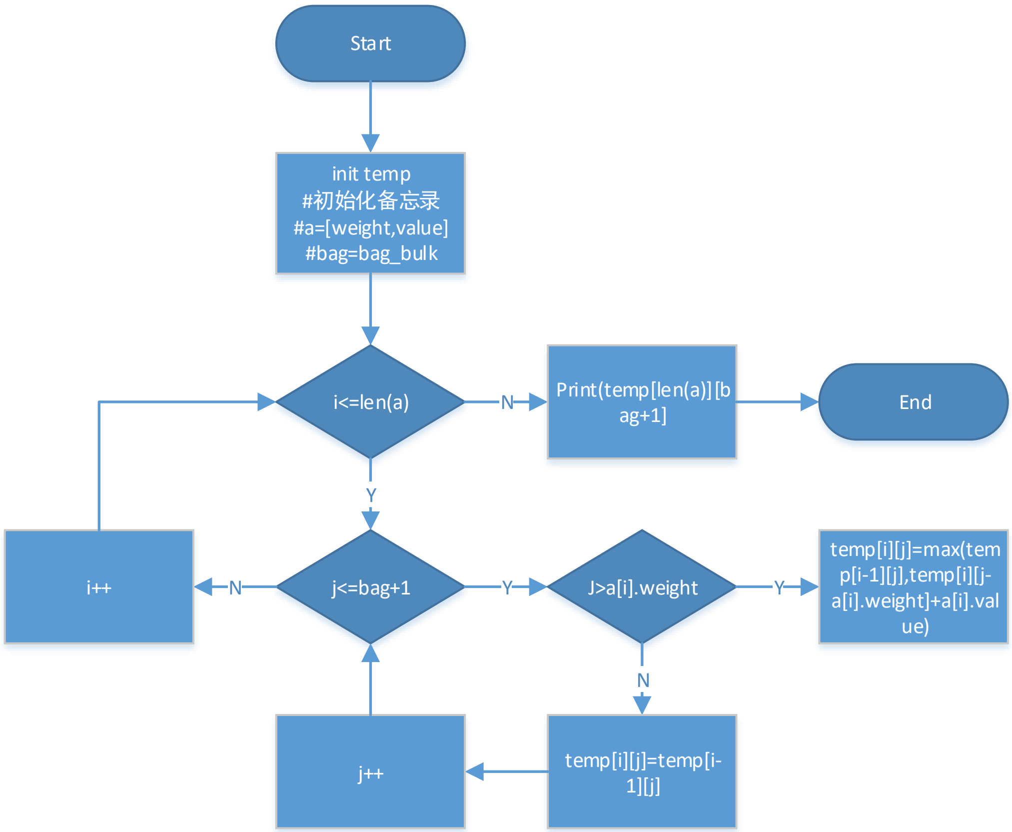
二、算法思路

这题显然是一个完全背包问题。可以把 b 看成背包容量，xi 为物体 i 的数目，ai 为物体 i 的体积，最终要求$\max \Sigma_{i=1}^{n} c_i x_i$。这个问题非常类似于 01 背包问题，所不同的是每种物品有无限件。也就是从每种 物品的角度考虑，与它相关的策略已并非取或不取两种，而是有取 0 件、取 1 件、取 2 件直至装不下为止。可以得到状态迁移方程：

$$\mathrm{F[i,v]} = \max\{\mathrm{F[i-1,v]}, \mathrm{F[i,v-Ci]} + vi\}$$

由此方程得到程序。

三、流程图



四、测试结果

```
[3]
a=[
    [],[2,5],[3,4],[1,2]
]
bag=15
dptree1(a,bag)

[[0, 2, 5, 7, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 37], [0, 2, 5, 7, 10, 12, 15, 17, 20, 2
2, 25, 27, 30, 32, 35, 37], [0, 2, 5, 7, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 37], [0, 2,
 5, 7, 10, 12, 15, 17, 20, 22, 25, 27, 30, 32, 35, 37], [0, 2, 5, 7, 10, 12, 15, 17, 20, 22, 25, 2
7, 30, 32, 35, 37]]
max= 37

<__main__.dptree1 at 0x1e8784ae1d0>
```

```
[4]
b=[[],[2,2],[3,5],[4,5],[1,1],[5,10],[6,5]]
bag=23
dptree1(b,bag)

[[0, 1, 2, 5, 6, 10, 11, 12, 15, 16, 20, 21, 22, 25, 26, 30, 31, 32, 35, 36, 40, 41, 42, 45], [0,
 1, 2, 5, 6, 10, 11, 12, 15, 16, 20, 21, 22, 25, 26, 30, 31, 32, 35, 36, 40, 41, 42, 45], [0, 1,
 2, 5, 6, 10, 11, 12, 15, 16, 20, 21, 22, 25, 26, 30, 31, 32, 35, 36, 40, 41, 42, 45], [0, 1, 2,
 5, 6, 10, 11, 12, 15, 16, 20, 21, 22, 25, 26, 30, 31, 32, 35, 36, 40, 41, 42, 45], [0, 1, 2, 5,
 6, 10, 11, 12, 15, 16, 20, 21, 22, 25, 26, 30, 31, 32, 35, 36, 40, 41, 42, 45], [0, 1, 2, 5, 6, 1
0, 11, 12, 15, 16, 20, 21, 22, 25, 26, 30, 31, 32, 35, 36, 40, 41, 42, 45], [0, 1, 2, 5, 6, 10, 1
1, 12, 15, 16, 20, 21, 22, 25, 26, 30, 31, 32, 35, 36, 40, 41, 42, 45], [0, 1, 2, 5, 6, 10, 11, 1
2, 15, 16, 20, 21, 22, 25, 26, 30, 31, 32, 35, 36, 40, 41, 42, 45]]
max= 45

<__main__.dptree1 at 0x1e8784ae6d8>
```

五、实验总结

对于完全背包问题，假设最优值为 m（i， j），可以建立递归式：

$$m(i,j)=\begin{cases}\max\{m(i-1,j),m(i,j-a_i)+c_i\} & a_i\le j\\ m(i-1,j) & 0\le j<a_i\end{cases}$$

m(0, j) = m(i, 0) = 0;

按此递归计算出的 m(n, b)为最优解，算法的时间复杂度为 O(nb).

完全背包问题也是一个相当基础的背包问题，它有两个状态转移方程。掌握了这个方程，问题就能迎刃而解。

下面说明 dptree1 类，此算法用于查找哪些物品被选择。

在列出的备忘录中，最右侧的值为最优值，只要从上往下走，直到最优值发生变化，说明这个物品被选到，然后减小容量；以此类推。该算法的时间复杂度为 O(n+c)，因为遍历顺序只有往下和往左两个选择。

这里提供思路二：

由于本题是一道特殊的 0-1 背包问题，我们可以计算出每个物品单位质量的价值，用空瓶填物的思想先装石头，后装沙子，再灌水。

我们先装单位价值最大的物品，装至无法装入后，装剩余空间内可装的单位价值最大的物品，依次递推，可以得出最大值。

六、附录 （程序代码）

```
'''
bag=bag_bulk
a=[weight,value]
'''

class dptree1:
    def __init__(self,a:list,bag:int):
        temp=[[0]*(bag+1)]*(len(a)+1)
        for i in range (1,len(a)):
            for j in range (1,bag+1):
                if j>=a[i][0]:
                    temp[i][j]=max(temp[i-1][j],temp[i][j-
a[i][0]]+a[i][1])
                else:
                    temp[i][j]=temp[i-1][j]
        print(temp)
        print("max=",temp[len(a)][bag])
#测试用例：
a=[
    [],[2,5],[3,4],[1,2]
]
```

```
bag=15
dptree1(a,bag)

b=[[],[2,2],[3,5],[4,5],[1,1],[5,10],[6,5]]
bag=23
dptree1(b,bag)
```