算法设计与分析文档

多维背包问题也称为多约束背包问题或者多背包问题，它是带有一组约束（重量 尺寸 可靠性等）的背包问题。该问题可以简单描述为n个物品要放入m个称重不同的背包，他与0-1背包问题不同的是，物品放入不同背包的重量是不同的。显然，在多约束背包的问题中，除了确定每个物品是否被放入背包之外，还需要确定他需要放入哪个背包。多维多选择背包问题是一类特殊的0-1背包问题。问题的描述如下：存在m个背包，其称重分别是Wk（k=1，2，3…m）和n类物品，每类物品又分别有Ii（i=1,2,…n）个物品，其价值分别为Vij（j=1,2,…）而对每一个物品，由于其装入的背包不同而其重量也有所不同，分别为Wijk,该问题要求每一类只选择装入一个物品，在满足背包称重的限制下最大化装入背包的物品总价值。多维背包问题也简称为MKP，MKP数据测试如下：

6 10 3800//6表示商品数、10表约束、3800表最优解；

100 600 1200 2400 500 2000//价值

8 12 13 64 22 41//属性加起来不超过80

8 12 13 75 22 41//属性加起来不超过96

3 6 4 18 6 4//属性加起来不超过20

5 10 8 32 6 12//属性加起来不超过36

5 13 8 42 6 20//属性加起来不超过44

5 13 8 48 6 20//属性加起来不超过48

0 0 0 0 8 0//属性加起来不超过10

3 0 4 0 8 0//属性加起来不超过18

3 2 4 0 8 4//属性加起来不超过22

3 2 4 8 8 4//9属性加起来不超过24

80 96 20 36 44 48 10 18 22 24//属性的总量

想要算出价值的最优解，那么可以利用回溯法来解决。回溯法的思想就是深度优先搜索，是一种组织得井井有条的、能避免不必要重复搜索的穷举式搜索算法。它的基本思想是：从一条路往前走，能进则进，不能进则退回来，换一条路再试。这时可通过回溯法来求解每组属性满足的约束条件，当第一组求出满足的属性后，再求第二组，这样就拿第一组满足的属性和第二组的属性求它们的交集，以此类推，就可求出最优解3800。

在这过程中，想要得出以上的数据，我是从python3中导出excel表格中的数据，数据如下：

6 10

100 600 1200 2400 500 2000

8 12 13 64 22 41

8 12 13 75 22 41

3 6 4 18 6 4

5 10 8 32 6 12

5 13 8 42 6 20

5 13 8 48 6 20

0 0 0 0 8 0

3 0 4 0 8 0

3 2 4 0 8 4

3 2 4 8 8 4

80 96 20 36 44 48 10 18 22 24

然后刚才所说的回溯法来求解最优解，得到如下数据：

3800

主要是利用python语言来写，代码如下：

import xlrd

import numpy as np

import pandas as pd

from numpy import \*

from pandas import DataFrame #导入包

data = pd.read\_excel("C:\\Users\\Administrator\\Desktop\\data.xlsx") #读取第excel表格中的数据

v=data.columns

price=np.array(v[:6])

print(price) #读取价值

data=np.array(data)

print(data) #读取的其他属性以及约束条件

qy=data[:,:-1]

print(qy) #从data中读取属性 qy :属性

cn=data[:,6:7]

print(cn) #读取约束条件 cn : 条件

n=6 # 物品的个数

m=10 #物品的属性

qy=pd.DataFrame(qy)

cn=pd.DataFrame(cn) # 格式转化，将array格式转换成dataFrame格式

v=0 #定义一个常量v 并赋值给0

mv = 0 #定义一个数组 mv最大价值

def bag(a):

# 背包中最多可以放下a个物品（物品最多6个，所以a<=6）

global mv,v #在函数内定义全局变量 mv, v

if a > n-1:

if v>mv:

mv=v # a于物品总数的时候，错误

else:

p=0 #定义一个P，令它的值等于约束条件中的值减去属性中的值:data2[0]- data[a]

p=(cn[0]- qy[a])>=0

if p.all():

cn[0]=cn[0]-qy[a] #约束条件减去属性，得到剩余的重量

v=v+price[a] #如果满足，加上根结点的价值

bag(a+1)

v=v-price[a] #不满足条件时，减去根节点的价值，继续回溯

cn[0]=cn[0]+qy[a] #继续回溯，需要加上再加上根节点的价值，才能得到剩余的重量

bag(a+1)

bag(0)

# bag（0） 最大的值序号为0

print("最大价值为%f元"% mv) #打印输出最大价值