**最少费用购物问题**

1. **问题描述：**当用户输入想购买的商品及数量时，计算出花费最少的购物方案及费用。购物方案即从哪个网站（如淘宝，天猫，京东等）哪家店铺购买的多少商品，使得用户的花费最少。

输入 输入

商品数量

商品型号

商品

价格

店铺

1. **遇到的问题：**如何让用户输入的商品型号及数量在点击提交后与各个网站中的商品直接链接响应上；点击提交按钮后如何获取得价格并筛选出最优惠价格；
2. **问题分析：**要解决的问题同我们做的背包问题差不多。只是背包问题是给定容量，求最大价值的东西。而这道题目是给定所放的东西，求最小的费用（对应背包问题为最小的容量）。恰好是一个求最值的“逆问题”。背包问题是经典的动态规划问题，由于动态规划要满足无后效性和最优化原理，所以我们来分析此题是否满足以上两点。先来状态表示的方法，商品不超过5种，而每种采购的数量又不超过5，那么用一个五元组来表示第I种商品买AI的最小费用。：F（A1，A2，A3，A4，A5） （1）考虑这个状态的由来，当然，我们不用优惠商品也可以买，显然这样不是最优。于是如果我们能够使用第I条商品组合的话，状态就便为了：F（A1-SI1，A2-SI2，A3-SI3，A4-SI4，A5-SI5） （2）这样的话，状态1的费用为状态2的费用加上SI的费用，而状态2的费用必须最低（很显然，用反证法即可），同时，我们也不管状态2是如何来的（因为每一个优惠商品组合的使用是没有限制的），所以本题既满足无后效性，又符合最优化原理，同时还有大量重叠子问题产生，动态规划解决此题是最好不过了。通过对问题的分析，我们知道了状态的表示和转移的基本方法，我们很容易得到一个状态转移方程：F [a, b, c, d, e] = Min {F [a-S1, b-S2, c-S3, d-S4, e-S5] + SaleCost [S]}

初始条件为：F [a, b, c, d, e] = Cost [1]\*a+Cost [2]\*b+Cost [3]\*c+Cost [4]\*d+Cost [5]\*e，即不用优惠的购买费用。

1. **算法分析：**
2. python爬取JD商品的代码：

import requests

from bs4 import BeautifulSoup

import urllib

import xlsxwriter

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36"

}

def get\_good\_urls(word):

url\_str = urllib.parse.quote(word)

urls = (

"https://search.jd.com/Search?keyword={}&enc=utf-8&qrst=1&rt=1&stop=1&vt=2&offset=4&page={}&s=1&click=0

".format(

url\_str, i) for i in range(1, 12, 2))

return urls

def get\_html(url):

html = requests.get(url, headers=headers)

html.encoding = html.apparent\_encoding

soup = BeautifulSoup(html.text, 'lxml')

return soup

def get\_info(soup):

all\_titles = soup.find\_all(class\_="p-name p-name-type-2")

all\_prices = soup.find\_all(class\_="p-price")

all\_commits = soup.find\_all(class\_="p-commit")

all\_imgs = soup.find\_all(class\_="p-img")

titles = []

prices = []

commits = []

links = []

imgs = []

for title in all\_titles:

titles.append(title.text.strip())

for price in all\_prices:

prices.append(price.text.strip())

for commit in all\_commits:

commits.append(commit.text.strip())

for link in all\_imgs:

links.append(link.find\_all('a')[0].get("href"))

for img in all\_imgs:

imgs.append(img.find\_all('img')[0].get("src"))

return titles, prices, commits, links, imgs

if \_\_name\_\_ == '\_\_main\_\_':

good = input("请输入你要查询的商品\n")

links = get\_good\_urls(good)

workbook = xlsxwriter.Workbook(good + '.xlsx') # 创建新表

worksheet = workbook.add\_worksheet()

bold = workbook.add\_format({'bold': True}) # 建立粗体格式

worksheet.write('A1', 'Title', bold) # 写入标题，粗体

worksheet.write('B1', 'Price', bold)

worksheet.write('C1', 'Commit', bold)

worksheet.write('D1', 'Link', bold)

worksheet.write('E1', 'Img', bold)

worksheet.set\_column('A:A', 100) # 改变列宽度

worksheet.set\_column('B:B', 10)

worksheet.set\_column('C:C', 18)

worksheet.set\_column('D:D', 27)

worksheet.set\_column('E:E', 100)

all\_row = 1

col = 0

for link in links:

html = get\_html(link)

ti, pr, co, li, im = get\_info(html)

row = all\_row

for t in ti:

worksheet.write(row, col, t)

row += 1

row = all\_row

for p in pr:

worksheet.write(row, col + 1, p)

row += 1

row = all\_row

for c in co:

worksheet.write(row, col + 2, c)

row += 1

row = all\_row

for l in li:

worksheet.write(row, col + 3, l)

row += 1

row = all\_row

for i in im:

worksheet.write(row, col + 4, i)

row += 1

all\_row += len(ti)

print('Done One page')

workbook.close()