

**实 验 报 告**

课 程 名 称 ：   《算法设计与分析》

实 验 日 期 :   2019年5月15日

专 业：     计算机科学与技术

班 级：     计算机 172班

团队名称： 宇宙超强

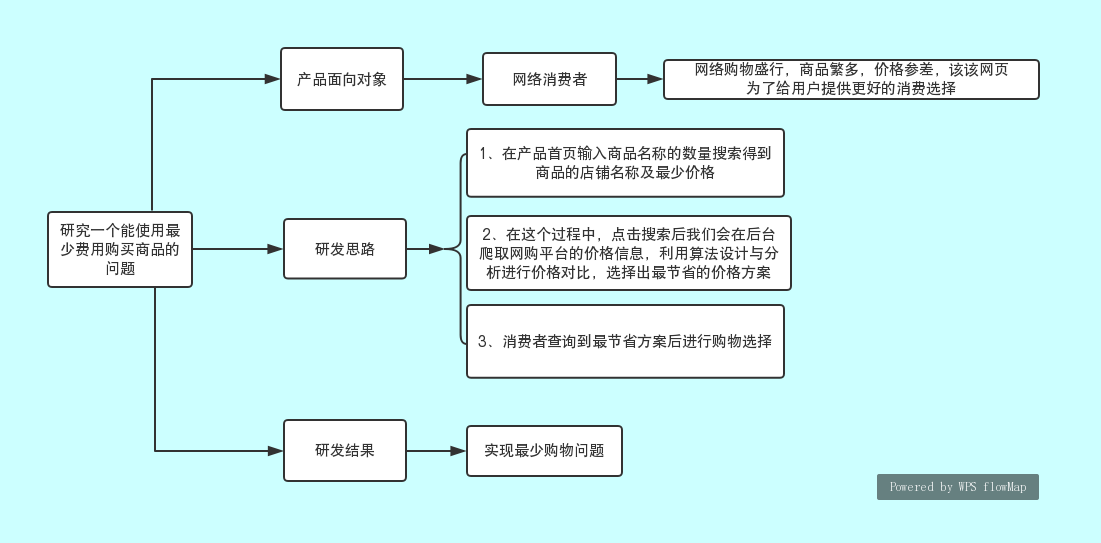
团队成员：    陈景德、李方旦、覃连芳

      潘碧惠、郭文素

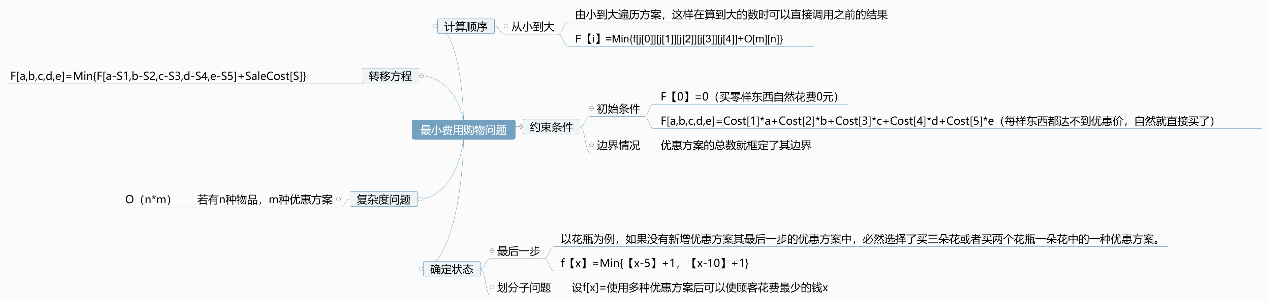
# 一、实验内容：

制作一个小程序（软件）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，即从哪个网站的哪家店铺购买多少什么商品。考虑京东、淘宝、天猫、沃尔玛线上等人们常用的电子商务网站。

# 二、需求分析：

****

# 三、1．问题建模(最少费用购物问题)：

****

## 确定状态

### 规划子问题

对于最小的费用购物问题，我们首先要确定它的状态。什么意思呢？就是说，我们现在的目标是——花最少的钱，买到自己规定数目和商品种类的东西。那么，他肯定要有优惠方案，才有优惠这样一说。比如说，花和花瓶，单买一朵花是2元，单买一个花瓶是5元。那么，商家推出优惠方案：1、买三朵花只要5元。2、买一朵花和两个花瓶要10元。当然还可能有第三，第四种方案等等。我们可以把这些方案，想象成我们平时付钱一样，一个方案可以对应成平时买东西时一张面值固定的钞票，然后我们想要，花最少张数的钞票付完钱。对应到这里呢就是。选择多种优惠方案后，可以使得，我们能花最少的X元买下我们的东西。

那么，我们可以设F【x】=选择多种优惠方案后花费最少的钱X。

### 最后一步

对于动态规划的问题，我们并不需要关心他中间到底选了哪些方案才导致的花钱最少，但是我们可以知道，他最后一步到底有哪些选择。当购买的东西很多之后，可能有的优惠方案使用了多次，但不管怎么使用，如果我们让它一个方案一个方案的购买，那么在它完成购买的最后一步，必然时选择了众多方案中的一个，这里以我们之前举例的两个方案为例，那么他最后一步之前的状态就是：

F[x]=Min{[x-5]+1，[x-10]+1}

它在做最后一步的选择的时候，其必然是从基础方案中的一个做出的最优选择。

## 转移方程

当然，实际描述的时候，基础的优惠方案还是太小，我们把它扩大成五个的形式（也可实际情况更改），那么，对这个问题的最后一步的状态也可以统括为对整个问题的转移方程：

F [a, b, c, d, e] = Min {F [a-S1, b-S2, c-S3, d-S4, e-S5] + SaleCost [S]}

## 初始条件和边界问题

当然，一份算法我们还要将它的初始条件和边界情况考虑进去。初始条件我们可以理解为特殊情况。那么在这个问题里面初始条件（特殊情况）是什么呢？首先，如果我们什么都不买，就自然不用付一分钱，或者我们买的东西数目每一样，都达不到优惠价格，是不是也是原价购买。那么初始条件就界定为：

F[0]=0；

F [a, b, c, d, e] = Cost [1]\*a+Cost [2]\*b+Cost [3]\*c+Cost [4]\*d+Cost [5]\*e

那么边界问题我们也很容易想到：优惠的方案数，就框定了这个问题的边界。

## 计算顺序

动态规划和递归，有些异曲同工之处，那就是遍历。但是，动态规划不同于递归的地方就在于，它会省去重复子问题的计算，按照这个思想，他当然要从小到大计算，这样它才能够在计算到大数量级的时候前面的数据有迹可查。

F【i】=Min{f[j[0]][j[1]][j[2]][j[3]][j[4]]+O[m][n]}

# 2.算法设计：

每一个买商品的用户都会有所需购买的数量，每个商品也会有它对应的价格，所以我们定义一个类来存放它们。

**class** CommodityCommodity{

**int** piece;

**int** price;

}

同时，商品自然要有编码来区分种类，优惠也要有优惠的组合方式在这里，我们将商品编码的最大值定义在三位数以内，虽然并不一定会用到这么多

**private** **static** **int** *MAXCODE* = 99;

同理定义，优惠商品组合数，商品的种类和可以购买的最大值、以及定义两个量来存放实例化的商品种类数和组合优惠数。同时记录购买数量和它们之间的关系。

**private** **static** **int** *SALECOMB* = 99;

**private** **static** **int** *KIND* = 5;

**private** **static** **int** *QUANTITY* = 5;

**private** **static** **int** *b*;

**private** **static** **int** *s*;

**private** **static** **int**[] *num* = **new** **int**[*MAXCODE*+1];

**private** **static** **int**[] *product* = **new** **int**[*KIND*+1];

同时，需要定义一个数组，来记录在某一状态下，该购物的最少总花费

**private** **static** **int**[][] *offer* = **new** **int**[*SALECOMB*+1][*KIND*+1];

**private** **static** CommodityCommodity[]

*purch* = **new** CommodityCommodity[*KIND*+1];

**private** **static** **int**[][][][][] *cost* = **new** **int**[*QUANTITY*+1][*QUANTITY*+1][*QUANTITY*+1][*QUANTITY*+1][*QUANTITY*+1];

当然，算法对问题的处理过程才是最关键的。我们定义一个数组，让让它从0开始，对要购买的商品从小到大的遍历，有优惠方案的就用优惠方案，然后记录下在每一个当前状态下的最优的那个方案，同时如果有更优的则替换它

**private** **static** **void** minicost(){

**int** i,j,k,m,n,p,minm; minm = 0;

**for**(i=1; i<=*b*; i++)

minm += *product*[i]\**purch*[i].price;

**for**(p=1; p<=*s*; p++){

i = *product*[1] - *offer*[p][1];

j = *product*[2] - *offer*[p][2];

k = *product*[3] - *offer*[p][3];

m = *product*[4] - *offer*[p][4];

n = *product*[5] - *offer*[p][5];

**if**(i>=0 && j>=0 && k>=0 && m>=0 && n>=0 && *cost*[i][j][k][m][n]+*offer*[p][0] < minm)

minm = *cost*[i][j][k][m][n] + *offer*[p][0];

}

*cost*[*product*[1]][*product*[2]][*product*[3]][*product*[4]][*product*[5]] = minm;

}

在这里，我们对一个购物方案的选择其实是受前一个状态的影响的，所以从小到的的规划遍历，就省去了很多重复的计算从而降低了算法的复杂度。

最后就是将计算得到的结果输出

System.*out*.println(*cost*[*product*[1]][*product*[2]][*product*[3]][*product*[4]][*product*[5]])

# 3.算法分析：

评价一个算法的好坏，一个在于复杂度的大小，和边界问题及初始化的考虑是否充分。在这里，我们用的是动态规划问题，所以有多少种优惠方案，就会做多少次的遍历，同时也取决于商品种类的多少和数量。假如有J种商品，购买数目为N，优惠方案为M，那么其复杂度就为O（J\*N\*M）。对于边界和初始化问题，如果都不买东西，花费自认就是0元，或者买的东西没有优惠方案，自然只能原价购买，所以就有：

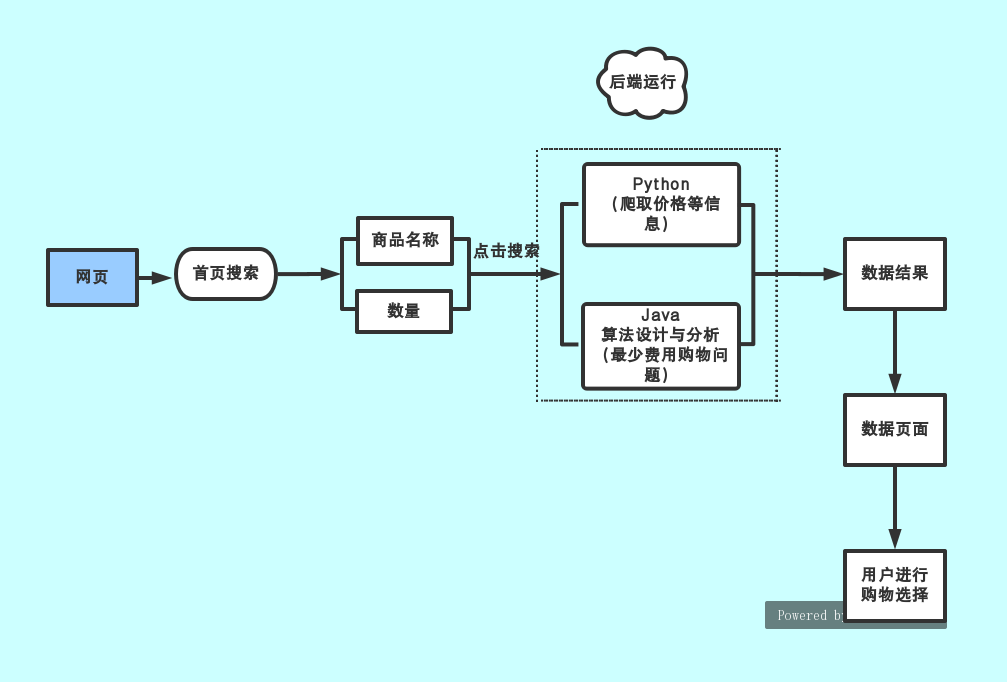
F[0]=0；

F [a, b, c, d, e] = Cost [1]\*a+Cost [2]\*b+Cost [3]\*c+Cost [4]\*d+Cost [5]\*e。

**4.算法实现：**

详见最小购物问题的代码及录屏。

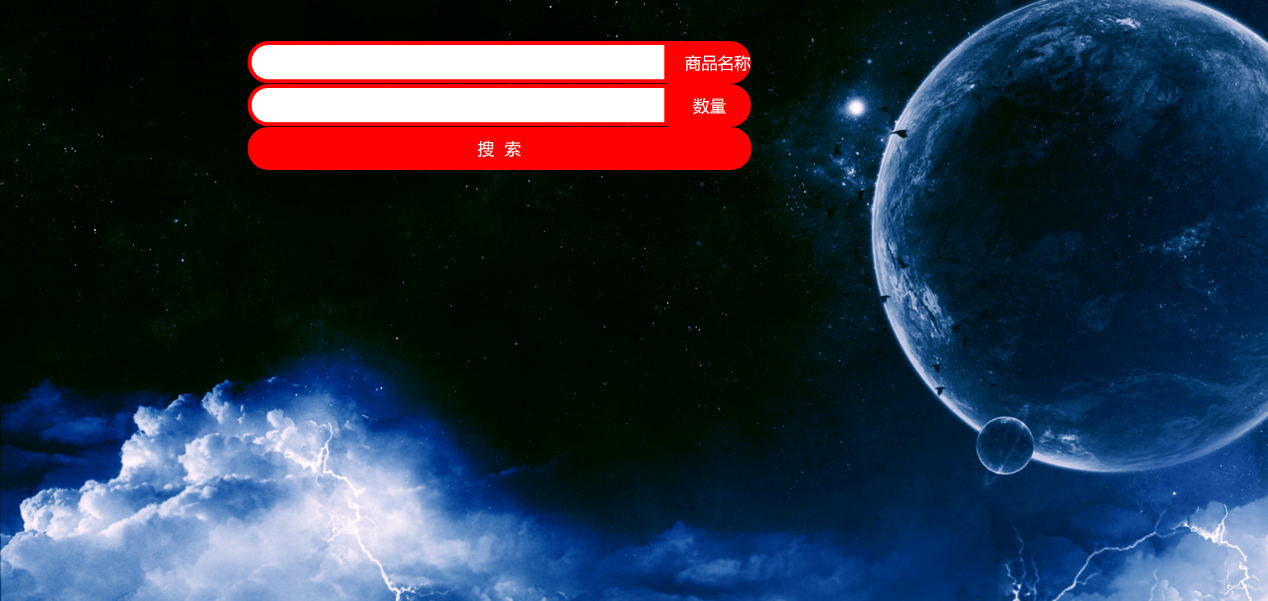
# 四、架构设计：

****

## 1.前端设计：

1. 使用工具：Adobe Dreamweaver CC 2018。
2. 网页总体设计。首页输入搜索条件，点击搜索后跳转到数据结果页面。
3. 首页设计。首页设置两个文本框，一个要求数据商品名称， 另一个要求输入所需数量，再设置一个搜索按钮，实现下一个数据结果页面的跳转。

首页界面如下：



首页编辑代码：

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>宇宙超省搜索</title>

<style type="text/css">

body,ul,li{

margin: 0;

padding: 0;

}

#wrap{

width: 612px;

margin: 50px 300px;

position: relative;

}

input{

width: 460px;

height: 42px;

border: 5px solid red;

float: left;

outline: none;

border-radius: 25px 0 0 25px;

font-size: 18px;

padding: 0 20px;

}

a{

color: darkmagenta;

text-decoration: none;

}

#list{

width: 480px;

position: relative;

left: 25px;

border: 1px solid #CECECE;

display: none;

/\*z-index: -1;\*/

}

#list>li{

/\*height: 40px;\*/

padding: 0 10px;

border-bottom: 1px solid #CECECE;

list-style: none;

line-height: 40px;

cursor: pointer;

}

#list>li:last-child{

border-bottom: none;

}

#list>li:hover{

background: chartreuse;

}

#btn{

float: left;

height: 52px;

width: 610px;

background: red;

color: white;

border: none;

text-align: center;

border-radius:25px 25px 25px 25px;

cursor: pointer;

outline: none;

font-size: 20px;

}

#btn1{

float: left;

height: 52px;

width: 100px;

background: red;

color: white;

border: none;

text-align: center;

border-radius:0 25px 25px 0;

cursor: pointer;

outline: none;

font-size: 20px;

}

.ul2{

width: 250px;

height:100%;

position: absolute;

right: 0;

top: 0;

/\*z-index: 10;\*/

background: #F6F9FC;

box-sizing: border-box;

padding-top: 20px;

display: none;

}

#list>li:hover .ul2{

display: block;

}

.ul2 .li2{

margin: 5px;

float: left;

/\*box-sizing: border-box;\*/

list-style: none;

/\*background: red;\*/

border: 1px solid #cecece;

}

.ul2 .li2 a{

display: block;

line-height: 30px;

width: 60px;

height: 30px;

text-align: center;

}

.ul2 .li2:hover {

background: red;

}

.ul2 .li2:hover a{

color: white;

}

.hover{

color: red;

}

</style>

</head>

<body background="../image/b1.jpg">

<div id="wrap">

<div style="overflow: hidden;">

<input type="text" name="txt" id="txt" value=""/>

<input type="button" name="btn" id="btn1" value="商品名称"/>

<input type="text" name="txt" id="txt" value=""/>

<input type="button" name="btn" id="btn1" value="数量" />

<input type="button" name="btn" id="btn" value="搜 索" onclick="window.location.href='2.html'"/>

</div>

<ul id="list"></ul>

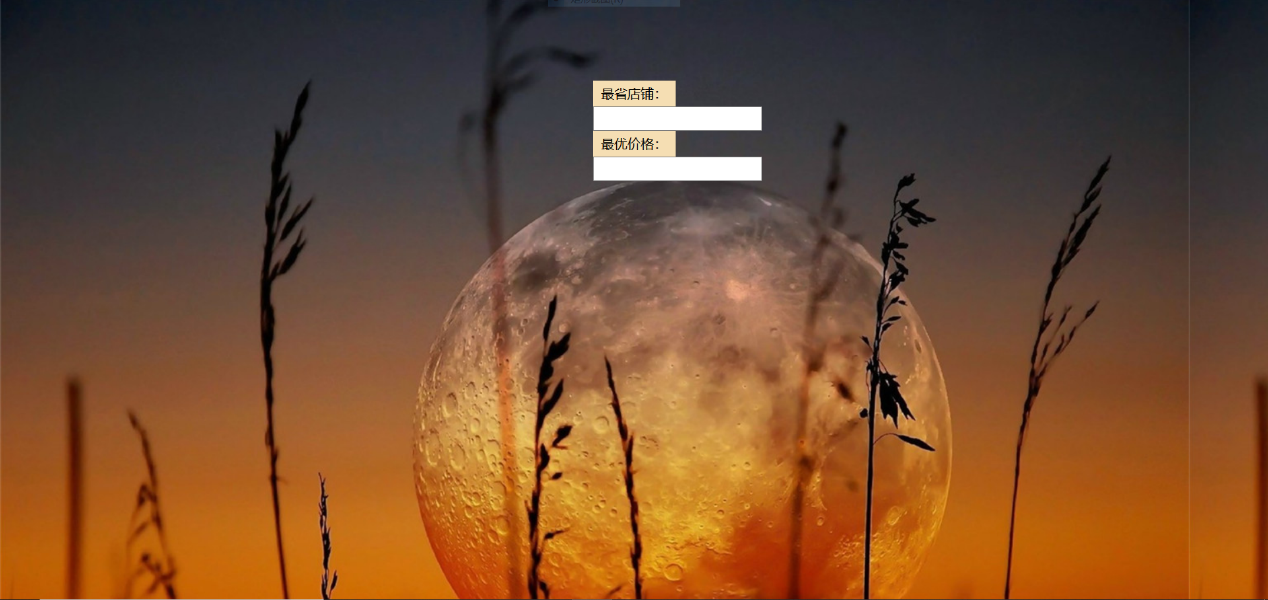
</div>

</body>

</html>

1. 数据结果页面设计。结果页面设置两个文本框，第一个文本框显示筛选后得到的最省的店铺的名称，第二个文本框显示与其对应的商品价格。

数据结果页面如下：



数据结果页面编辑代码：

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>宇宙最优选择</title>

<style>

.item1{

height: 100px;

width: 100px;

background-color: wheat;

text-align: center;

line-height: 30px;

margin: 100px auto;

}

</style>

</head>

<body background="../image/b4.jpg">

<div class="outer">

<div class="item1"><form>

最省店铺：

<input type="text" name="firstname" style="width: 200px;height: 25px">

<br />

最优价格：

<input type="text" name="lastname" style="width: 200px;height: 25px">

</form></div>

</div>

</body>

</html>

## 2.后端设计：

①编写Python，利用Python设计爬虫爬取淘宝、天猫、京东等各大网络消费品牌的商品名称、价格等有关信息。当用户从搜索网页输入商品名称及所需的数量后，点击搜索按钮，爬虫将在后台进行运行操作。

Python爬虫代码如下：

#!coding=utf-8

import requests

import re

import random

import time

import json

from requests.packages.urllib3.exceptions import InsecureRequestWarning

import pandas as pd

requests.packages.urllib3.disable\_warnings(InsecureRequestWarning) ###禁止提醒SSL警告

class tm(object):####手机端

def \_\_init\_\_(self,path): ###保存数据路径

self.path=path

def goodsid(self,url): ###通过店铺URL获取店铺所有ID

shopname = re.search('https://(.\*?).tmall', url).group(1)

searchurl = 'https://{}.m.tmall.com/shop/shop\_auction\_search.do?spm=a1z60.7754813.0.0.301755f0pZ1GjU&sort=defaul'.format(

shopname)

s=requests.session()

headers = {'Accept': '\*/\*',

'Accept-Language': 'zh-CN',

'User-Agent': 'Mozilla/5.0 (iPhone; CPU iPhone OS 10\_3\_3 like Mac OS X) AppleWebKit/603.3.8 (KHTML, like Gecko) FxiOS/10.6b8836 Mobile/14G60 Safari/603.3.8',

'Referer':'https://{}.m.tmall.com/shop/shop\_auction\_search.htm?spm=a1z60.7754813.0.0.301755f0pZ1GjU&sort=default'.format(shopname)

}

s.headers.update(headers)

page1=s.get(url=searchurl,verify=False).text

print(page1)

js=json.loads(page1)

total\_page=int(js['total\_page'])

shop\_id=js['shop\_id']

shop\_title = js['shop\_title']

shop\_id\_list = []

shop\_title\_list = []

item\_id=re.findall('"item\_id":(.\*?),"',page1)

title=re.findall('"title":"(.\*?)","',page1)

sold=re.findall('"sold":"(.\*?)","',page1)

totalSoldQuantity=re.findall('"totalSoldQuantity":(.\*?),"',page1)

skuurl=re.findall('"url":"(.\*?)","',page1)

price=re.findall('"price":"(.\*?)","',page1)

item\_id\_l=len(item\_id)

shop\_id\_list.append(shop\_id)

shop\_id\_list.extend(shop\_id\_list\*(int(item\_id\_l)-1))

shop\_title\_list.append(shop\_title)

shop\_title\_list.extend(shop\_title\_list\*(int(item\_id\_l)-1))

# print(js)

# print(len(shop\_id\_list))

# print(len(shop\_title\_list))

# print(len(item\_id))

# print(len(title))

# print(len(sold))

# print(len(totalSoldQuantity))

# print(len(skuurl))

# print(len(price))

data = {'shop\_id': shop\_id\_list,'shop\_title': shop\_title\_list,'item\_id': item\_id, 'title': title, 'sold':sold, 'totalSoldQuantity':totalSoldQuantity, 'skuurl':skuurl, 'price':price}

df = pd.DataFrame(data=data)

#print(df)

savepath=self.path + r'\tmgoodsid{}.csv'.format(shopname)

print(savepath)

df.to\_csv(savepath, mode='a', index=False, encoding="GB18030")

time.sleep(random.random() \* 2)

if total\_page!=1:

for i in range(2,total\_page+1):

time.sleep(random.random() \* 2)

htmlurl=searchurl+'&p={}'.format(i)

html=s.get(url=htmlurl,verify=False).text

shop\_id\_list = []

shop\_title\_list = []

print(html)

item\_id = re.findall('"item\_id":(.\*?),"',html)

title = re.findall('"title":"(.\*?)","', html)

sold = re.findall('"sold":"(.\*?)","', html)

totalSoldQuantity = re.findall('"totalSoldQuantity":(.\*?),"', html)

skuurl = re.findall('"url":"(.\*?)","', html)

price = re.findall('"price":"(.\*?)","',html)

item\_id\_l = len(item\_id)

shop\_id\_list.append(shop\_id)

shop\_id\_list.extend(shop\_id\_list \* (int(item\_id\_l) - 1))

shop\_title\_list.append(shop\_title)

shop\_title\_list.extend(shop\_title\_list \* (int(item\_id\_l) - 1))

data = {'shop\_id': shop\_id\_list, 'shop\_title': shop\_title\_list, 'item\_id': item\_id, 'title': title,

'sold': sold, 'totalSoldQuantity': totalSoldQuantity, 'skuurl': skuurl, 'price': price}

df = pd.DataFrame(data=data)

df.to\_csv(self.path + r'\tmgoodsid{}.csv'.format(shopname),mode='a', index=False,header=0 ,encoding="GB18030")

df1 = pd.read\_csv(self.path + r'\tmgoodsid{}.csv'.format(shopname), encoding='GB18030')

s.close()

return df1

def getiddata(self,id): ###获取ID数据

time.sleep(random.random() \* 1 + 1)

s = requests.session()

t=int(time.time()\*1000)

url='https://h5api.m.taobao.com/h5/mtop.taobao.detail.getdetail/6.0/' \

'?jsv=2.4.8&appKey=12574478&t={}' \

'&sign=7c9e1dedaa295fdb175d22c99746493b&api=mtop.taobao.detail.getdetail' \

'&v=6.0&dataType=jsonp&ttid=2017%40taobao\_h5\_6.6.0&AntiCreep=true&type=jsonp&callback=mtopjsonp2&' \

'data=%7B%22itemNumId%22%3A%22{}%22%7D'.format(t,id)

headers = {'Accept': '\*/\*',

'Accept-Language': 'zh-CN',

'User-Agent': 'Mozilla/5.0 (iPhone; CPU iPhone OS 10\_3\_3 like Mac OS X) AppleWebKit/603.3.8 (KHTML, like Gecko) FxiOS/10.6b8836 Mobile/14G60 Safari/603.3.8',

'Referer': 'https://detail.m.tmall.com/item.htm?spm=a220m.6910245.0.0.55b17434eiwv4f&id={}'.format(id)

}

print(url)

s.headers.update(headers)

html = s.get(url=url, verify=False).text

html=html.replace('\\','')

time.sleep(0.5)

info=re.search('skuBase":(.\*?),"skuCore',html)

if info!=None:

skuBase=re.search('skuBase":(.\*?),"skuCore',html).group(1) ##SKU+颜色

skuId = re.findall('"skuId":"(.\*?)","', skuBase)

propPath=re.findall('"propPath":"(.\*?)"}',skuBase)

skuBase=json.loads(skuBase)

prop\_list=[]

for i in propPath:

prop = ''

prop1=i.split(';')

for j in prop1:

prop2=j.split(':')

for pid in skuBase['props']:

if pid['pid']==prop2[0]:

#prop=prop+pid['name']

for vid in pid['values']:

if vid['vid']==prop2[1]:

prop=prop+vid['name']

prop\_list.append(str(prop))

sku2info = re.search('"sku2info":(.\*?)},"s', html).group(1) ##价格

sku2info = json.loads(sku2info)

price = []

for i in skuId:

p = sku2info[str(i)]['price']['priceText']

price.append(p)

else:

skuId=[' ']

prop\_list=[' ']

price=[' ']

data = {'skuid': skuId, 'prop': prop\_list,'price':price}

df = pd.DataFrame(data=data)

return df

def iddata(self,id\_df):

df\_l=id\_df.iloc[:,0].size

df=pd.DataFrame()

df.loc[0, "shop\_id"] = ''

df.loc[:, "shop\_title"] = ''

df.loc[:, "item\_id"] = ''

df.loc[:, "title"] = ''

df.loc[:, "sold"] = ''

df.loc[:, "totalSoldQuantity"] = ''

df.loc[:, "skuurl"] = ''

df.loc[:, "price"] = ''

df.loc[:, "skuid"] = ''

df.loc[:, "prop"] = ''

df.loc[:, "skuprice"] = ''

shopid=id\_df['shop\_id'][1]

y=0

for i in range(0,df\_l):

time.sleep(random.random() \* 2.56)

pid=id\_df['item\_id'][i]

data=self.getiddata(pid)

data\_l=data.iloc[:,0].size

for j in range(0,data\_l):

df.at[y, "shop\_id"] = id\_df['shop\_id'][i]

df.at[y, "shop\_title"] = id\_df['shop\_title'][i]

df.at[y, "item\_id"] = id\_df['item\_id'][i]

df.at[y, "title"] = id\_df['title'][i]

df.at[y, "sold"] = id\_df['sold'][i]

df.at[y, "totalSoldQuantity"] = id\_df['totalSoldQuantity'][i]

df.at[y, "skuurl"] = id\_df['skuurl'][i]

df.at[y, "price"] = id\_df['price'][i]

df.at[y, "skuid"] = data['skuid'][j]

df.at[y, "prop"] = data['prop'][j]

df.at[y, "skuprice"] = data['price'][j]

y +=1

df.to\_csv(self.path + r'\tm{}.csv'.format(shopid), index=False, encoding="GB18030")

return df

def urlitem(self,url,\*args): ##通过目录获取 只适合部分

s = requests.session()

headers = {'Accept': '\*/\*',

'Accept-Language': 'zh-CN',

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.15 Safari/537.36'

}

s.headers.update(headers)

itemhtml = s.get(url=url, verify=False).text

#print(itemhtml)

shopid = re.search('class="J\_TModule"(.\*?)"搜索列表"', itemhtml).group(1)

shopid=re.search('data-widgetid="(.\*?)" id',shopid).group(1)

#print(shopid)

id=re.search('category-(.\*?).htm',url).group(1)

nm=re.search('https://(.\*?).tmall.com/',url).group(1)

t=int(time.time()\*1000)

pageurl='https://{}.tmall.com/i/asynSearch.htm?\_ksTS={}\_888&callback=jsonp289&mid=w-{}-0&wid={}&path=/category-{}.htm'.format(nm,t,shopid,shopid,id)

print(pageurl)

time.sleep(random.random() \* 1 + 1)

html = s.get(url=pageurl, verify=False).text

html = html.replace('\\', '')

html=re.sub('\n','',html)

page=re.search('ui-page-s-len">1/(.\*?)</b>',html).group(1)

print(page)

nm\_list=[]

idurl\_list=[]

price\_list=[]

sale\_list=[]

for p in range(1,int(page)+1):

time.sleep(random.random())

pageurl = 'https://{}.tmall.com/i/asynSearch.htm?\_ksTS={}\_888&callback=jsonp289&mid=w-{}-0&wid={}&path=/category-{}.htm'.format(

nm, t, shopid, shopid, id)

html = s.get(url=pageurl, verify=False).text

html = html.replace('\\', '')

html = re.sub('\n', '', html)

print(html)

nm=re.findall('<img alt="(.\*?)" data',html)[:-8]

print(nm)

id=re.findall('<a href="//detail.(.\*?)&rn',html)

idurl=[]

for i in id:

idurl.append('https://detail.'+i)

price=re.findall('class="c-price">(.\*?) ',html)[:-8]

sale=re.findall('sale-num">(.\*?)</span>',html)[:-8]

nm\_list.extend(nm)

idurl\_list.extend(idurl)

price\_list.extend(price)

sale\_list.extend(sale)

print(len(nm\_list))

print(len(idurl\_list))

print(len(price\_list))

print(len(sale\_list))

data={'nm':nm\_list,'idurl':idurl\_list,'price':price\_list,'sale':sale\_list}

df=pd.DataFrame(data)

l=len(args)

for j in range(0,l):

df.loc[:, "col"+str(j)] = args[j]

print(df)

s.close()

return df

# 例子：

# tm = tm()

# url = 'https://shoushanggeshi.tmall.com/category-1310604910.htm'

# # url = 'https://shoushanggeshi.tmall.com/category-674950482.htm'

# tm.urlitem(url, '电脑', 'cpu')

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

path=r'E:\tm'

tm=tm(path)

df=tm.goodsid('https://intel.tmall.com')

tm.iddata(df)

②使用Java编写解决最少费用购物问题的代码，即将Python爬虫爬取到的价格信息进行对比和筛选，得到商品最优的价格以及店铺名称，最终将数据呈现在数据结果页面，为用户提供最好的消费选择。

Java代码如下：

package fczx;

import java.util.Scanner;

class Commodity{

int piece;//购买数量

int price;//购买价格

}

public class MinmShopping {

private static int MAXCODE = 100;//商品编码的最大值

private static int SALECOMB = 50;//优惠商品组合数

private static int KIND = 10; //商品种类

private static int QUANTITY = 10; //购买某种商品数量的最大值

private static int b;//购买商品种类数

private static int s;//当前优惠组合数

private static int[] num = new int[MAXCODE+1];//记录商品编码与商品种类的对应关系

private static int[] product = new int[KIND+1];//记录不同种类商品的购买数量

private static int[][] offer = new int[SALECOMB+1][KIND+1];//offer[i][j]: 商品组合的优惠价(j=0)；某种优惠组合中某种商品需要购买的数量

private static Commodity[] purch = new Commodity[KIND+1];//记录不同商品的购买数量和购买价格

private static int[][][][][] cost = new int[QUANTITY+1][QUANTITY+1][QUANTITY+1][QUANTITY+1][QUANTITY+1];//记录本次购买的总花费

public static void main(String[] args){ init(); comp(1); out();

}

private static void minicost(){

int i,j,k,m,n,p,minm; minm = 0;

for(i=1; i<=b; i++)

minm += product[i]\*purch[i].price;

for(p=1; p<=s; p++){

i = product[1] - offer[p][1];

j = product[2] - offer[p][2];

k = product[3] - offer[p][3];

m = product[4] - offer[p][4];

n = product[5] - offer[p][5];

if(i>=0 && j>=0 && k>=0 && m>=0 && n>=0 && cost[i][j][k][m][n]+offer[p][0] < minm)

minm = cost[i][j][k][m][n] + offer[p][0];

}

cost[product[1]][product[2]][product[3]][product[4]][product[5]] = minm;

}

private static void init(){

Scanner input = new Scanner(System.in);

int i,j,n,p,t,code;

for(i=0; i<51; i++) for(j=0; j<11; j++) offer[i][j] = 0;

for(i=0; i<11; i++){

purch[i] = new Commodity(); purch[i].piece = 0;

purch[i].price = 0; product[i] = 0;

}

b = input.nextInt();

for(i=1; i<=b; i++){

code = input.nextInt();

purch[i].piece = input.nextInt();

purch[i].price = input.nextInt();

num[code] = i;

}

s = input.nextInt();

for(i=1; i<=s; i++){

t = input.nextInt();

for(j=1; j<=t; j++){

n = input.nextInt();

p = input.nextInt();

offer[i][num[n]] = p;

}

offer[i][0] = input.nextInt();

}

}

private static void comp(int i){

if(i > b){ minicost();

return;

}

for(int j=0; j<=purch[i].piece; j++)

{

product[i] = j; comp(i+1);

}

}

private static void out(){

System.out.println(cost[product[1]][product[2]][product[3]][product[4]][product[5]]);

}

}

## 3.测试结果：

①利用Python设计爬虫爬取淘宝价格的代码运行成功。

②利用Java编写进行价格对比，解决最少费用购物问题的代码

运行成功。

③网页的总体设计完成。

④缺少服务器的搭建，无法将Python和Java代码调用到网页

上，故整体项目实施失败。

# 五、心得体会：

2019年春季学期的实践周已经告一段落，本次实践周的一个课题是制作一个能解决最少费用购物问题的小程序或者网页。我们都知道，在这个互联网高速发展的时代背景下，网络购物也应运而生，并在近年来蓬勃发展，更多的人喜欢在淘宝、京东、天猫等这样的购物平台进行消费。网络消费平台的商品也是琳琅满目，价格也参差不齐，因此，我们这个课题就是为了给消费者提供更好的购物选择，用最优惠的价格，享受最优质的购物体验。所以，我们的小组成员经过不懈的努力，同时花费了许多精力去分析和完成这个课题。我们虽然不能将课题做到圆满成功，但是我们也通过各方面的技能学习和实践，使自己在专业知识的层面上有了很大的提升，我们想这也是实践周的最大一个目的，让我们在实践中获取知识和成长。我们通过老师的指导，并在网络上查阅资料和学习，应用到了Python的爬虫技术，制作网页的HTML技术，解决最少费用购物问题的动态规划方法等等。但是，由于我们对各种编程技术的陌生，以及短时间内对知识的掌握太过于浅显，所以我们只能做到一个大致的规模。首先是Python的爬虫技术，在爬取各大网络消费平台的价格等信息并不能做到全面，爬取过程中可能会出现失误。其中一个原因是各大消费平台都有“反爬”机制，解决这个问题完全超出了我们目前的学习范围，让我们无所适从。接着就是利用Java编程出解决最少费用购物问题，其中利用到了算法设计中的动态规划，我们在一些细节分析上出现了疑惑，对于当问题扩大规模化或者遇到特殊情况时的处理方法应该是怎样的这一问题有点无从下手，对于分析算法的复杂度也有些困惑的地方。但是，总体而言还是使得程序运行成功了。最后就是网页的问题，网页也只是设计了一个简洁的页面雏形，以及实现了很基本的页面跳转。但实际上，我们无法完成将Python以及Java等文件的调用，使得整个网页达到目标效果。这个过程的完成，需要有服务器的搭建，但是由于这个技术十分复杂，我们还没能掌握并应用它。整个课题的完成度大体而言，我们把各个部分的雏形都做出来了，但是无法给各个部分建立起联系，最终也使得这个网页不能总体的完成。由于实践周的学习时间太过短暂，我们也只有通过今后的学习中了解和掌握更多的专业知识，才能将这个网页功能呈现出来。但是，我们在这个过程中，学习到更多的可贵的东西，不仅仅是局限于专业知识这一块，还有更多精神层面上的领悟，这也会给我们在今后的学习和成长的路上更多的经验和启发。