分析代码、计算问题的复杂性

# data

2 import requests

3 import re

4 def get\_one\_page(url):

5 headers={

6 'User-Agent':'Mozilla/5.0(Macintosh;Intel Mac OS X 10\_13\_3) AppleWebKit/537.36(KHTML,like Gecko) Chrome/65.0.3325.162 Safari/537.36'

7 }#模拟登陆，防止1688禁止获取

8 response = requests.get(url,headers=headers)#请求URL，成功后返回200

9 if response.status\_code==200:

10 return response.text#访问成功返回源代码

11 return None

12 def main():

13 goods=input('请输入产品名称：')

14 goods=goods.encode('gbk')#将中文转换成gbk

15 goods=str(goods)#转换成字符串

16 k=len(goods)#计算字符串长度

17 goods=goods[2:k-1]#取2到k-1个字符串

18 goods=goods.upper()#将str大写

19 goods=eval(repr(goods).replace('\\', '%'))#将\\转换成%

20 goods=eval(repr(goods).replace('%%X', '%'))#将%%X转换成%

21 url = "https://s.1688.com/selloffer/offer\_search.htm?keywords="+goods#网址拼接

22 html=get\_one\_page(url)#调取get\_one\_page方法得到源代码

23 goodlist=[]#空的列表

24 plist = re.findall(r'\"sm-offer-priceNum sw-dpl-offer-priceNum.\*?\"\>[\d]\*',html)#正则匹配商品价格

25 for i in range(len(plist)):#循环列表

26 price = re.findall(r'\d+\.?\d\*',plist[i])#列表内的每个元素取数字即价格

27 goodlist.append(price)#向列表goodlist添加元素

28 print(goodlist)

29 main()#主函数，执行的开始

一、商品价格爬取Python版的代码，选择使用该版本的代码作为商品价格的爬取，我们在这里用的是Jupyter notebook，通过分析这段代码，我们从def main(): 这段代码可以看出该问题的时间复杂性为O（1）。

import java.util.Scanner;  
class Commodity{  
    int piece;//购买数量  
    int price;//购买价格  
}  
public class ZuiShaoFeiYongGouWu {  
    private static int MAXCODE = 999;//商品编码的最大值  
    private static int SALECOMB = 99;//优惠商品组合数  
    private static int KIND = 5;     //商品种类  
    private static int QUANTITY = 5; //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(j>0)  
    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<100; i++)  
            for(j=0; j<6; j++)  
                offer[i][j] = 0;  
  
        for(i=0; i<6; 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]]);  
    }  
}

二、商品价格爬取java版的代码，选择使用该版本的代码作为商品价格的爬取，通过分析这段代码，从以上这段代码可以看出该问题的时间复杂性为O（n^9）。

<html>

<head>

<script type="text/javascript">

function showHint(str)

{

var xmlhttp;

if (str.length==0)

{

document.getElementById("txtHint").innerHTML="";

return;

}

if (window.XMLHttpRequest)

{// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{// code for IE6, IE5

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("txtHint").innerHTML=xmlhttp.responseText;

}

}

xmlhttp.open("GET","/ajax/gethint.asp?q="+str,true);

xmlhttp.send();

}

</script>

</head>

<body>

<h3>请输入产品的型号或价格：</h3>

<form action="">

产品型号：<input type="text" id="txt1" value="香水" />

</form>

<p>产品价格：

<input type="text" id="txt1" value="225" /><br>

<div><input type="submit"value="提交" onClick="window.open('https://list.jd.com/list.html?cat=1316,1387,11932&jth=yi#J\_crumbsBar')"></div><br>

</form>

</p>建议：<span id="txtHint"></span></p>

</body>

</html>

三、我们最终是使用HTML语言来实现我们的问题，并在IE浏览器上得到了相关的界面信息。通过分析这段代码，可以看出该问题的时间复杂性为O（1）。