# 问题建模

目录

[问题建模 1](#_Toc8983987)

[一、问题描述 1](#_Toc8983988)

[二、问题分析 1](#_Toc8983989)

[三、问题假设 2](#_Toc8983990)

[四、模型设计 2](#_Toc8983991)

[五、模型求解 4](#_Toc8983992)

一、问题描述

商店中每种商品都有标价。例如，一朵花的价格是2元。一个花瓶的价格是5 元。为了吸引顾客，商店提供了一组优惠商品价。优惠商品是把一种或多种商品分成一组，并降价销售。例如，3朵花的价格不是6元而是5元。2 个花瓶加1 朵花的优惠价是10 元。试设计一个算法，计算出某一顾客所购商品应付的最少费用。

二、问题分析

首先，要明确各个商品的单价，用户即使购买一种商品，也要从中给出最优惠的方案。其次，当有优惠方案是，充分考虑到组合后的价格，注意不要出现无优惠状况。当商品组合并无优惠方案时，也要考虑到这样的情况。对于给定欲购商品的价格和数量，以及优惠商品价，编程计算所购商品应付的最少费用。下面的代码就是使用穷举方法，但是使用动态规划中的备忘录法减少穷举过程中的重复计算。穷举过程的递归式为cost[A][B][C][D][E]=min{cost[A-Ai][B-Bi][C-Ci][D-Di][E-Ei]+Pi},0=<i<=s，Ai,Bi,Ci,Di,Ei,为第i种优惠方案中，商品种类1-5对应的组合数量，Pi为第i种组合优惠方案的优惠价格

三、问题假设

1、用户购买的种类商品数量及种类存在优惠方案

2、存在此类商品

四、模型设计

文件的第1行中有1 个整数B（0≤B≤5），表示所购商品种类数。

接下来的B 行，每行有3 个数C，K 和P。C 表示商品的编码（每种商品有唯一编码），1≤C≤999。

K 表示购买该种商品总数，1≤K≤5。P 是该种商品的正常单价（每件商品的价格），1≤P≤999。

请注意，一次最多可购买5\*5＝25件商品。

由文件offer.txt提供优惠商品价数据。

文件的第1行中有1 个整数S（0≤S≤99），表示共有S 种优惠商品组合。

接下来的S 行，每行的第一个数描述优惠商品组合中商品的种类数j。接着是j 个数字对（C，K），其中C 是商品编码，1≤C≤999。K 表示该种商品在此组合中的数量，1≤K≤5。每行最后一个数字P（1≤ P≤9999）表示此商品组合的优惠价。

结果输出:程序运行结束时，将计算出的所购商品应付的最少费用输出到文件output.txt中。

输入文件示例

input.txt

2 //用户欲购买两种商品

7 3 2 //编号为7的商品欲购买3个，单价为2

8 2 5 //编号为8的商品欲购买2个，单价为5

offer.txt

2 //两种优惠组合方案

1 7 3 5 //商品种类数为1，编号为7，欲购买3个，总价格为5

2 7 1 8 2 10 //商品种类数为2，编号分别是7和8，编号为7的商品欲购买1个，编号为8的商品欲购买2个，总价格为10

输出文件示例

output.txt

14

五、模型求解

代码如下：

#include<stdio.h>

#include<malloc.h>

#include<fstream>

#include<iostream>

using namespace std;

#define MAX 10000000

//商品编号用0,1,2,3,4代替

typedef struct{

//每件商品的单价

int price;

//所需购买的商品的总量,1=<amount<=5

int amount;

}Goods;

typedef struct{

//组合促销方案中的商品种类数

int kinds\_num;

//组合商品的总的优惠价格

int price;

//组合方案中，每种商品的的编号和数量

int amount[5];

}Promote;

int s;

Goods goods[5];

Promote \*promote;

int cost[6][6][6][6][6];

int mincost(int A,int B,int C,int D,int E);

int find\_match(int c[],int num)

{

int i=0;

for(i=0;i<5;i++)

{

if(c[i]==num)

return i;

}

return -1;

}

int main()

{

int b;

int i,j;

int c[5];

ifstream in("input.txt");

in>>b;

for(i=0;i<5;i++)

{

if(i<b)

{

in>>c[i];

in>>goods[i].amount;

in>>goods[i].price;

}else{

c[i]=0;

goods[i].amount=0;

goods[i].price=0;

}

}

ifstream in1("offer.txt");

in1>>s;

promote=new Promote[s];

for(j=0;j<s;j++)

{

memset(&promote[j],0,sizeof(Promote));

in1>>promote[j].kinds\_num;

int num;

for(i=0;i<promote[j].kinds\_num;i++)

{

in1>>num;

int index=find\_match(c,num);

if(index>=0)

in1>>promote[j].amount[index];

else {

cout<<"data err"<<endl;

exit(1);

}

}

in1>>promote[j].price;

}

int res=mincost(goods[0].amount,goods[1].amount,goods[2].amount,goods[3].amount,goods[4].amount);

cout<<"res is "<<res<<endl;

}

//备忘录法进行递归穷举

int mincost(int A,int B,int C,int D,int E)

{

int i=0;

int min=MAX;

for(i=0;i<s;i++)

{

int A1=promote[i].amount[0];

int B1=promote[i].amount[1];

int C1=promote[i].amount[2];

int D1=promote[i].amount[3];

int E1=promote[i].amount[4];

if(A>=A1&&B>=B1&&C>=C1&&D>=D1&&E>=E1)

{

if(min>cost[A-A1][B-B1][C-C1][D-D1][E-E1]+promote[i].price)

{

if(cost[A-A1][B-B1][C-C1][D-D1][E-E1]>0)

min=cost[A-A1][B-B1][C-C1][D-D1][E-E1]+promote[i].price;

else min=mincost(A-A1,B-B1,C-C1,D-D1,E-E1)+promote[i].price;

}

}else{

if(min>A\*goods[0].price+B\*goods[1].price+C\*goods[2].price+D\*goods[3].price+E\*goods[4].price)

min=A\*goods[0].price+B\*goods[1].price+C\*goods[2].price+D\*goods[3].price+E\*goods[4].price;

}

}

cost[A][B][C][D][E]=min;

return cost[A][B][C][D][E];

}