算法设计与分析

计算机171何诗淇201706401128

1. 实验目的

为了更好的学习算法知识，巩固本学期所学内容，掌握算法的思想，加强记忆与应用能力和提高创新能力。

1. 实验要求

综合运用学习过的算法设计与分析方法：设计MKP问题的一种算法，分析复杂性，实现算法，调试测试，讲解代码。

1. 实验内容

问题描述：把一个约束的01背包问题改成多个约束的多维背包问题，就是二维背包问题的继续拓展。多个物品，多个属性，多个约束的条件下，背包能获得的最大价值。

输入示例： 输出示例：

6 10

100 600 1200 2400 500 2000 3800

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

数据输入：输入数据由文件名为input.txt的文本文件提供。

第一行为n物品书和m属性书

第二行为物品价值

第三到到十二行是每个属性的值

最后一行为背包的约束条件

结果输出：背包最高最优价值为3800元。

1. 实验过程和结果

#include <stdio.h>

#include <stdlib.h>

#include "stdafx.h"

#include<iostream>

//用来挑选物品序列集合

int array[64][6];

//用于存放物品的价格

int value[6];

//用于存放各个物品的各个属性

int attribute[10][6];

//背包的约束条件

int YueShu[10];

//最优解序列，用1来标志物品存放入背包，0表示未存放入，初始值都是0

int x[] = { 0,0,0,0,0,0 };

//用于存放中间变量

int result[10] = { 0,0,0,0,0,0,0,0,0,0 };

//中间输出变量变量

int outArry[6] = { 0,0,0,0,0,0 };

//价值最高最优解

int MaxValue = 0;

//解析最大值，求两个数中的最大值

int max(int a, int b);

//二进制数据转换

int\* IntToBiary(int num);

//创建数组，表示物品序列集合

void CreateArray();

//初始化输入物品属性价格

void initData(void);

//清除缓存

void ClearArray(void);

//该两个函数用于测试

//解析是否成功，并返回是否解析出当前约束条件下的最优解

int IsSucceed(int\* arry, int\* outArry);

//计算，只是用来测试一下是否正确

int Calculator(int\* arr);

int main()

{

initData(); //初始化物品属性价格

CreateArray(); //清除缓存

for (int i = 0; i < 64;i++)

{

int sumValue = 0; //用来存放当前价值

ClearArray();

int n = 0;

for (int j = 0; j < 10;j++)

{

for (int k = 0; k < 6;k++)

{

if (array[i][k] == 1)

{

result[j] += attribute[j][k]; //中间变量加等于物品的该属性

}

else if (array[i][j] == 0)

{

result[j] += 0;

}

}

}

if (result[0] < YueShu[0] && //当每个中间变量的值都不超过背包的约束条件时

result[1] < YueShu[1] &&

result[2] < YueShu[2] &&

result[3] < YueShu[3] &&

result[4] < YueShu[4] &&

result[5] < YueShu[5] &&

result[6] < YueShu[6] &&

result[7] < YueShu[7] &&

result[8] < YueShu[8] &&

result[9] < YueShu[9]

)

{

for (int k = 0;k < 6;k++)

{

if (array[i][k] == 1)

{

sumValue += value[k];

outArry[k] = 1;

}

else

{

outArry[k] = 0;

}

}

}

else

{

sumValue = 0;

}

if (MaxValue < sumValue)

{

MaxValue = sumValue;

x[0] = outArry[0];

x[1] = outArry[1];

x[2] = outArry[2];

x[3] = outArry[3];

x[4] = outArry[4];

x[5] = outArry[5];

}

}

printf("产品的价格为：\n");

for (int i = 0;i < 6;i++)

{

printf(" %d\n", value[i]);

}

printf("最优解为：%d %d %d %d %d %d \n", x[0], x[1], x[2], x[3], x[4], x[5]);

MaxValue = 0;

printf("最优解的价格分别为：\n");

for (int i = 0;i < 6;i++)

{

if (x[i] == 1)

{

printf(" %d ", value[i]);

MaxValue += value[i];

}

}

printf("最优解结果为：%d \n", MaxValue);

system("pause");

return 0;

}

//初始化输入物品属性价格

void initData(void)

{

memset(value, 0, sizeof(value[6])); //memset初始化函数。

memset(attribute, 0, sizeof(attribute[10][6]));

memset(YueShu, 0, sizeof(YueShu[10]));

printf("请依次输入物品价格： \n");

for (int i = 0;i < 6;i++)

{

scanf("%d", &value[i]);

}

printf("请依次输入物品对的每个属性： \n");

for (int i = 0;i < 10;i++)

{

printf("请依次每个物品的第%d属性： \n",i+1);

for (int j = 0; j < 6;j++)

{

scanf("%d", &(attribute[i][j]));

}

}

printf("请依次输入背包的约束条件： \n");

for (int i = 0;i < 10;i++)

{

scanf("%d", &YueShu[i]);

}

memset(x, 0, sizeof(x[6])); //memset初始化函数。作用是将某一块内存中的内容全部设置为指定的值， 这个函数通常为新申请的内存做初始化工作

memset(outArry, 0, sizeof(outArry[6]));

MaxValue = 0; //价值最高最优解初始化为0

}

int\* IntToBiary(int num)

{

int b[16];

char\* arrory = "";

int n = 0;

for (int n = 0; num; num >>= 1, n++)

{

b[n] = num & 1;

}

return b;

}

void CreateArray() //创建数组，表示物品序列集合

{

int b = 0;

memset(array, 0, sizeof(array));

for (int i = 0;i < 64;i++)

{

for (int j = 5, k = 0; j >= 0, k < 6;j--, k++)

{

int\* arr = IntToBiary(i);

array[i][j] = arr[k];

}

}

for (int i = 0;i < 64;i++)

{

for (int j = 0; j < 6;j++)

{

if (array[i][j] != 1)

{

array[i][j] = 0;

}

}

}

}

int IsSucceed(int\* arry, int\* outArry)

{

int sum = 0;

int result[10] = { 0,0,0,0,0,0,0,0,0,0 };

int n = 0;

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 6; j++)

{

if (arry[j] == 1)

{

result[n++] += widget[i][j];

}

else if (arry[j] == 0)

{

result[n++] += 0;

}

}

}

if (result[0] <= YueShu[0] &&

result[1] <= YueShu[1] &&

result[2] <= YueShu[2] &&

result[3] <= YueShu[3] &&

result[4] <= YueShu[4] &&

result[5] <= YueShu[5] &&

result[6] <= YueShu[6] &&

result[7] <= YueShu[7] &&

result[8] <= YueShu[8] &&

result[9] <= YueShu[9]

)

{

//return 1;

for (int i = 0; i < 6; i++)

{

outArry[i] = arry[i];

if (arry[i] == 1)

{

sum += value[i];

}

}

}

else

{

sum = 0;

}

return sum;

}

int Calculator(int\* arr)

{

int sum = 0;

for (int i = 0; i < 6; i++)

{

if (arr[i] == 1)

{

sum += value[i];

}

}

return sum;

}

int max(int a, int b)

{

return a > b ? a : b;

}

void ClearArray(void) //清空缓存

{

result[0] = 0;

result[1] = 0;

result[2] = 0;

result[3] = 0;

result[4] = 0;

result[5] = 0;

result[6] = 0;

result[7] = 0;

result[8] = 0;

result[9] = 0;

result[10] = 0;

outArry[0] = 0;

outArry[1] = 0;

outArry[2] = 0;

outArry[3] = 0;

outArry[4] = 0;

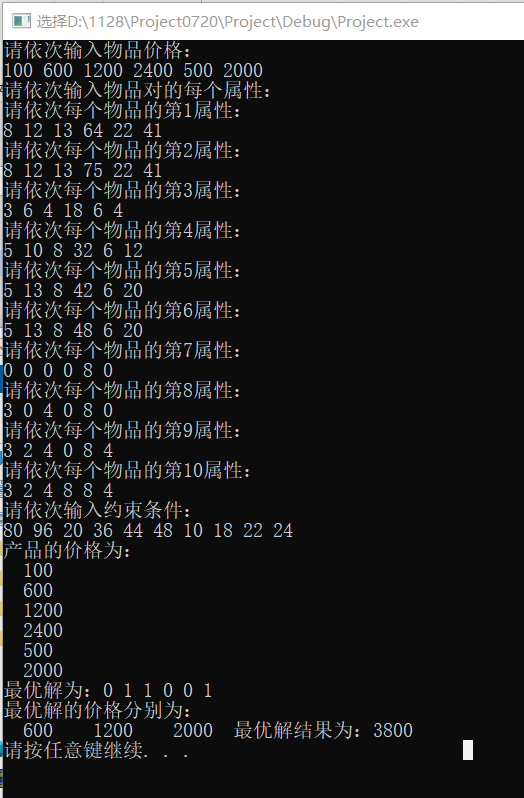
outArry[5] = 0;

}

1. 算法复杂度分析

这个代码的物品数n和属性数m我都直接写进去了，如果按n,m来算，代码运用循环不断的递归去找到最优解，时间复杂度为O（nm）。

1. 实验结果



1. 实验总结

通过这次试验让我对算法的基础知识有了一定的了解。尤其是对于递归和分治算法思想有了一定了了解，也学应用在一些问题上。