西安交通大学实验报告

课程名称： 算法设计与问题求解 实验名称： 动态规划-1

学 院： 机械工程学院 实验日期 2020 年 11 月 14日

班 级： **机械97班** 姓 名： 杨逢诜 学号： 2193712613

#### 一、实验内容和结果

* **题目1**

假设有多个金矿，其编号、产量见下表：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 产金量  人数 | 800  50 | 1500  70 | 6700  800 | 5800  760 | 5678  800 | 1200  120 | 500  30 | 900  49 | 7000  1000 | 8888  1500 |

要求：

* 每座金矿必须挖或者不挖，不能只挖一部分；
* 每个工人只能分配到一个金矿劳动。

已知总人数：10000，如果希望挖到的金子越多越好，请问那些金矿需要挖？

用给出的解法1或解法2编写程序，解决挖金矿问题，用本节开始所给出的数据验证。

【源程序】

#include<stdio.h>

int maxgold[100][10001]; //声明全局变量（二维数组）：用于储存在所有情形下可能获得的最大收益

int golddigging[100]; //声明全局变量：用于储存最佳决策下的收益策略

int DynamicProg(int n,int maxinv,int \*invest,int \*benefit) //声明函数：动态规划，用于计算最大收益

{

int i,j; //声明变量，用于控制循环

for(i=0;i<=maxinv;i++) //根据已知信息对第一行赋初值

{

if(i<invest[0])

{

maxgold[0][i]=0; //即对于第一座金矿，在投入人力数不足限额的情形下得不到收益

}

else maxgold[0][i]=benefit[0]; //在投入人力数充足的情形下得到全部收益

}

for(i=1;i<n;i++) //计算逐渐增多投入生产的金矿数的情形下的收益

{

for(j=0;j<=maxinv;j++) //遍历所有可能的投入人数，计算投入不同人数时的最大收益

{

if(j<invest[i]) //若提供的人力资源数不足以开挖新的金矿，则它的最大收益与不开挖此金矿时的最大收益一致

{

maxgold[i][j]=maxgold[i-1][j];

}

else //否则，则将投入相同人力资源而不开挖此金矿时的收益与“在投入的人力资源中安排一部分人开挖此金矿而其他人按照最大收益模式开采其他金矿”的收益相比并计算二者中的最大值，其最大值作为此时的最大收益

{

maxgold[i][j]=maxgold[i-1][j]>=(maxgold[i-1][j-invest[i]]+benefit[i])?maxgold[i-1][j]:(maxgold[i-1][j-invest[i]]+benefit[i]);

}

}

}

int t=maxinv; //声明参量，用于倒推最优开采策略

for(i=n-1;i>=0;i--)

{

if(i==0) //在倒推到最后一座金矿时，若剩余人力资源所对应的最大收益数等于0则意味着该矿未被开采，赋0

{

if(maxgold[i][t]==0)

{

golddigging[i]=0;

}

else golddigging[i]=1; //否则，则意味着该矿被开采，赋1

}

else if(maxgold[i][t]==maxgold[i-1][t]) //若在投入对应人力资源时，考虑此矿和不考虑此矿时的最大收益相同，则意味着该矿未被开采

{

golddigging[i]=0;

}

else //否则，该矿则被开采；同时向后倒推，找出该情形对应的前一个形态，同时将人力资源重置为该形态对应的人力资源数

{

golddigging[i]=1;

t=t-invest[i];

}

}

return maxgold[n-1][maxinv]; //返回计算得到的最大收益——在开掘全部金矿、投入全部人力情形下的最大收益

}

int main() //主函数开始

{

int hr[100],gd[100],n,hit,i,benefit; //声明变量：需要投入的人力资源数、可以获得的最大收益数、金矿总数、人力资源总数、循环控制变量、收益

printf("请输入金矿总数："); //根据提示依次输入金矿总数、工人总数、各矿需要的工人数、各矿的最大收益并存入内存中

scanf("%d",&n);

printf("请输入工人总数：");

scanf("%d",&hit);

printf("请输入各矿需要的工人数目：");

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

{

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

}

printf("请输入各矿的总产出：");

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

{

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

}

benefit=DynamicProg(n,hit,hr,gd); //调用动态规划，计算最大收益

printf("可挖出的金的最大数额为%d.\n",benefit); //输出最大收益

printf("金矿建议挖法:(0--不装，1--装)\n"); //输出提示

for(i=0;i<n;i++) //输出结论：第一行指示金矿对应编码，第二行指示决策，1表示挖，0表示不挖

{

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

}

printf("\n");

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

{

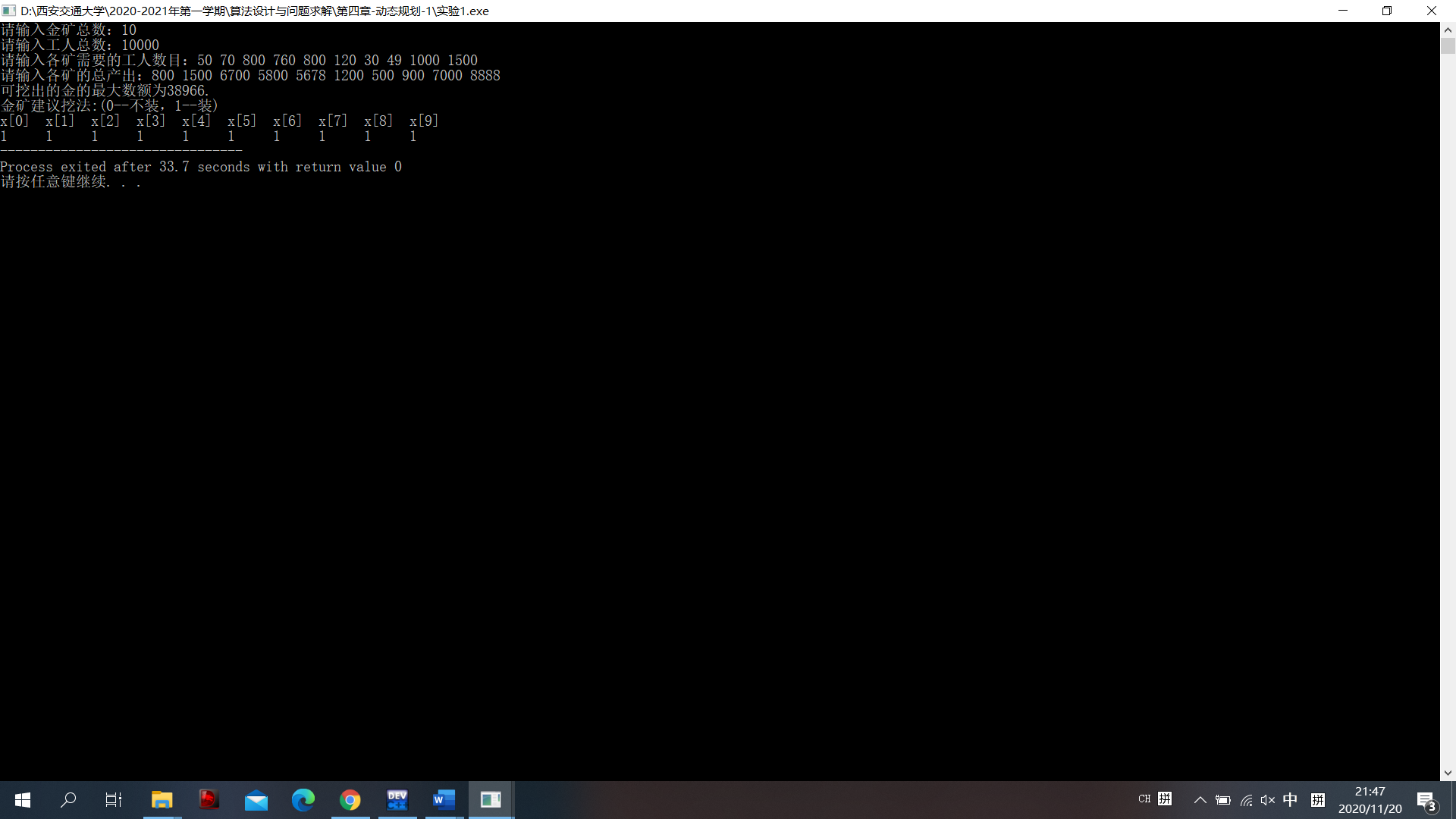
printf("%d ",golddigging[i]);

}

return 0; //主函数结束

}

【运行结果】



* **题目2**

小明要远行，他有一个容量为15kg的背包，另外有4个物品，物品的质量和价值如下表所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **物品** | A1 | A2 | A3 | A4 |
| **质量*w*** | 3 | 4 | 5 | 6 |
| **价值*v*** | 4 | 5 | 6 | 7 |

小明希望能用他的背包带走的物品总价值最大，你能告诉他应该怎么做吗？

用给出的动态规划法编写程序，解决01背包问题，用本节开始所给出的数据验证。

【源程序】

#include<stdio.h>

struct itemstruct //声明结构体，用于存储物品信息，包括质量和价值两项

{

int mass;

int value;

};

int DynamicProg(struct itemstruct \*iteminfo,int n,int mass,int \*items) //声明动态规划函数，用于计算在背包空间有限的情形下的最优选择策略

{

int i,j; //声明循环控制变量

int maxval[n][mass+1]; //声明二维数组，用于储存在所有情形下可能获得的最大收益

for(i=0;i<=mass;i++) //根据已知信息对第一行赋初值

{

if(i<iteminfo[0].mass)

{

maxval[0][i]=0; //即对于第一件物品，在投入空间数不足限额的情形下不能选择装入，对应价值失效

}

else maxval[0][i]=iteminfo[0].value; //在投入空间数充足的情形下物品价值有效

}

for(i=1;i<n;i++) //计算逐渐增多纳入考虑的物品数时的收益

{

for(j=0;j<=mass;j++) //遍历所有可能的投入空间数，计算投入不同空间的最大收益

{

if(j<iteminfo[i].mass) //若提供的空间不足以放新物品，则它的最大收益与不放新物品时的最大收益一致

{

maxval[i][j]=maxval[i-1][j];

//printf("%d %d %d\n",i,j,maxval[i][j]);

}

else //否则，则将投入相同空间而不装入此物品时的收益与“在投入的空间中安排一部分空间装入此物品而其他人按照最大收益模式装入其他物品”的收益相比并计算二者中的最大值，其最大值作为此时的最大收益

{

maxval[i][j]=maxval[i-1][j]>=(maxval[i-1][j-iteminfo[i].mass]+iteminfo[i].value)?maxval[i-1][j]:(maxval[i-1][j-iteminfo[i].mass]+iteminfo[i].value);

//printf("%d %d %d\n",i,j,maxval[i][j]);

}

}

}

int t=mass; //声明参量，用于倒推最优开采策略

for(i=n-1;i>=0;i--)

{

if(i==0) //在倒推到最后一座金矿时，若剩余空间所对应的最大价值数等于0则意味着该物品未被装入，赋0

{

if(maxval[i][t]==0)

{

items[i]=0;

}

else items[i]=1; //否则，则意味着该物品被装入，赋1

}

else if(maxval[i][t]==maxval[i-1][t]) //若在投入对应人空间时，考虑此物品和不考虑此物品时的最大收益相同，则意味着该物品未被装入

{

items[i]=0;

}

else //否则，该物品则被装入；同时向后倒推，找出该情形对应的前一个形态，同时将空间重置为该形态对应的空间

{

items[i]=1;

t=t-iteminfo[i].mass;

}

//printf("%d %d %d\n",i,t,items[i]);

}

return maxval[n-1][mass]; //返回计算得到的最大收益——在考虑全部物品、投入全部空间下的最大收益

}

int main() //主函数开始

{

int n,i,packagemass,packagevalue,items[20]; //声明变量：物品总数、循环控制变量、背包空间、背包可承装的最大价值、物品的装入策略

struct itemstruct iteminfo[20]; //声明变量：物品信息

printf("请输入物品总数："); //根据提示依次输入物品总数、背包空间、各物品的质量、各物品的价值并存入内存中

scanf("%d",&n);

printf("请输入背包可容纳的最大质量：");

scanf("%d",&packagemass);

printf("请输入各物品质量：");

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

{

scanf("%d",&iteminfo[i].mass);

}

printf("请输入各物品价值：");

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

{

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

}

packagevalue=DynamicProg(iteminfo,n,packagemass,items); //调用动态规划，计算最大收益

printf("可装入的物品最大价值为%d.\n",packagevalue); //输出最大收益

printf("物品建议取法:(0--不装，1--装)\n"); //输出提示

for(i=0;i<n;i++) //输出结论：第一行指示金矿对应编码，第二行指示决策，1表示挖，0表示不挖

{

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

}

printf("\n");

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

{

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

}

return 0; //主函数结束

}

【运行结果】

