**算法训练 开心的金明**

时间限制：1.0s   内存限制：256.0MB

问题描述

　　金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间他自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说：“你的房间需要购买哪些物品，怎 么布置，你说了算，只要不超过N元钱就行”。今天一早金明就开始做预算，但是他想买的东西太多了，肯定会超过妈妈限定的N元。于是，他把每件物品规定了一 个重要度，分为5等：用整数1~5表示，第5等最重要。他还从因特网上查到了每件物品的价格（都是整数元）。他希望在不超过N元（可以等于N元）的前提 下，使每件物品的价格与重要度的乘积的总和最大。  
　　设第j件物品的价格为v[j]，重要度为w[j]，共选中了k件物品，编号依次为 j1，j2，……，jk，则所求的总和为：  
　　v[j1]\*w[j1]+v[j2]\*w[j2]+ …+v[jk]\*w[jk]。（其中\*为乘号）  
　　请 你帮助金明设计一个满足要求的购物单。

输入格式

　　输入文件 的第1行，为两个正整数，用一个空格隔开：  
　　N m  
　　（其中N（<30000）表示总钱 数，m（<25）为希望购买物品的个数。）  
　　从第2行到第m+1行，第j行给出了编号为j-1的物品的基本数据，每行有2个非负整数  
　　v p  
　　（其中v表示该物品的价格(v<=10000)，p表示该物品的重要度(1~5)）

输出格式

　　输出文件只有一个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值（<100000000）。

样例输入

1000 5  
800 2  
400 5  
300 5  
400 3  
200 2

样例输出

3900

数据规模和约定

锦囊1

动态规划。

锦囊2

将每个物品看成大小为v[i]，价值为v[i]\*w[i]，则变成了一个背包问题，使用经典的动态规划解决。

本题的C++参考代码如下：

#include<iostream>

#include<cstdio>

#include<string.h>

using namespace std;

int dp[30000];

int main()

{

int N,m,v[300],w[300];

//freopen("d://1.txt","r",stdin);

cin>>N>>m;

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

{

cin>>v[i]>>w[i];

}

fill(dp,dp+N,0);

dp[0]=1;

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

for(int j=N;j>=v[i];j--)

dp[j]=max(dp[j],dp[j-v[i]]+w[i]\*v[i]);

cout<<dp[N]<<endl;

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

long mnum[30000];

long objprice[25]; //第i件商品的价格

long objk[25]; //第i件商品的价格与权值的积

int M,N; //M表示金钱总数，N表示商品总数

int main()

{

int i,m;

int maxp = 0;

scanf("%u%u", &M, &N); //M表示金钱总数，N表示商品总数

for(i=0;i<N;++i) //依次输入第i件商品的价格与权值，并计算出商品的价格与权值之积

{

scanf("%u%u", &objprice[i], &m);

objk[i] = m\*objprice[i];

}

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

{

for(m=0;m<M-objprice[i];++m)

if (mnum[m+objprice[i]] + objk[i] > mnum[m])

mnum[m] = mnum[m+objprice[i]] + objk[i];

}

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

if (mnum[i] > maxp)

maxp = mnum[i];

printf("%u", maxp);

return 0;

}

本题的Java参考代码如下：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main{

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String[] str = br.readLine().split(" ");

int money = Integer.parseInt(str[0]);

int count = Integer.parseInt(str[1]);

int[][] arr = new int[count][2];

for (int i = 0; i < count; i++) {

String[] order = br.readLine().split(" ");

for (int j = 0; j < 2; j++) {

arr[i][j] = Integer.parseInt(order[j]);

}

}

int[] temp = new int[money + 1];

for (int i = 0; i < count; i++) {

for (int j = money; j > 0; j--) {

if (j > arr[i][0]) {

if (temp[j - arr[i][0]] + arr[i][0] \* arr[i][1] > temp[j]) {

temp[j] = temp[j - arr[i][0]] + arr[i][0]

\* arr[i][1];

} else {

continue;

}

}

}

}

System.out.println(temp[money]);

}

}