背包专题之依赖背包

洛谷P1064 金明的预算方案

题目描述

金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间金明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，金明就开始做预算了，他把想买的物品分为两类：主件与附件，附件是从属于某个主件的，下表就是一些主件与附件的例子：

主件	附件
电脑	打印机，扫描仪
书柜	图书
书桌	台灯，文具
工作椅	无

如果要买归类为附件的物品，必须先买该附件所属的主件。每个主件可以有0个、1个或2个附件。每个附件对应一个主件，附件不再有从属于自己的附件。金明想买的东西很多，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10元的整数倍）。他希望在不超过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vj，重要度为wj，共选中了k件物品，编号依次为j1,j2,…,jk，则所求的总和为：

$$v_{j_1} \times w_{j_1} + v_{j_2} \times w_{j_2} + \cdots + v_{j_k} \times w_{j_k}。$$

请你帮助金明设计一个满足要求的购物单。

输入

第一行有两个整数，分别表示总钱数n和希望购买的物品个数m。

第2到第(m+1)行，每行三个整数，第(i+1)行的整数vi，pi，qi分别表示第i件物品的价格、重要度以及它对应的的主件。如果qi=0，表示该物品本身是主件。

输出

输出一行一个整数表示答案。

输入样例

```
1000 5
800 2 0
400 5 1
300 5 1
400 3 0
500 2 0
```

输出样例

```
2200
```

解析

这道题目中做了很多限制，使得我们解决起来非常的容易。

首先，每个主件最多只有两个附件，那么我们一共就只存在四种可能性：

a、只买主件
b、购买主件 + 附件1
c、购买主件 + 附件2
d、购买主件 + 附件1 + 附件2

还是有点晕对不对？我换另外一种描述：现在一共有N组物品，每组物品只能购买一件，求最大价值是多少？

怎么样，有没有一些思路？我们是不是可以把主附件的各种组合关系看成是分组背包中一组物品，只能选择其中的一个？

接下来，我们将题目中的数据与基础的背包做个映射

物品 -> 想要购买的物品
重量 -> 物品的价格
价值 -> 物品的价格 x 物品的重要程度
背包上限 -> 妈妈给的最大预算

编码

```cpp
#include<bits/stdc++.h>

using namespace std;

int dp[32100];        //一维优化表
int mitem[61];     //存储主件物品信息的数组
int item[61][5];   //存储附件物品信息的数组
int w[61], v[61];  //物品的价格和价值
int n, m, mcnt;  //总预算，希望购买的数量，主件物品的索引
```

```
int main() {
    //读入预算和物品总数量
   scanf("%d%d", &n, &m);
    //拍平的过程
   for (int i = 1; i <= m; i++) {
        int a, b, c;
        //读入物品的价格，重要程度、分组
       scanf("%d%d%d", &a, &b, &c);
        //找到一个主件，进行存储
       if (c == 0) {
            mitem[++mcnt] = i;  //mitem[1] = i;
        }
            //附件
       else {
            //item[c][0] 存储的是数量
          //1、主件下有几个附件
          //2、每个附件的实际索引
          item[c][++item[c][0]] = i;
            //item[1][1] = 0  //主件1的第一件附件的实际索引为2
            //item[1][2] = 3
        }
        w[i] = a;  //存储价格
       v[i] = a * b;  //存储价值
   }
    //遍历全部的主件
   for (int i = 1; i <= mcnt; i++) {
        //遍历全部的预算，一维优化，倒序
       for (int j = n; j >= 0; j--) {
            //当前主件的索引
          int mindex = mitem[i];
            //尝试只购买主件
          if (j - w[mindex] >= 0)
                dp[j] = max(dp[j - w[mindex]] + v[mindex], dp[j]);

            //尝试购买主件和附件1：
          int sIndex1 = item[mindex][1]; //读取附件1的索引
          //判断附件1是否存在，并判断当前的费用是否够买主件+附件1
            //sIndex1 == 0 说明，不存在这个附件
          if (sIndex1 && j - w[mindex] - w[sIndex1] >= 0)
                //计算购买主件和附件1的最大价值
              dp[j] = max(dp[j - w[mindex] - w[sIndex1]] + v[mindex]

            //尝试购买主件和附件2：
          int sIndex2 = item[mindex][2]; //读取附件2的索引
          //判断附件1是否存在，并判断当前的费用是否够买主件+附件2
            if (sIndex2 && j - w[mindex] - w[sIndex2] >= 0)
                //计算购买主件和附件2的最大价值
              dp[j] = max(dp[j - w[mindex] - w[sIndex2]] + v[mindex]
```

```
             //尝试购买主件和附件2+附件1：
            if (sIndex1 && sIndex2 && j - w[mindex] - w[sIndex1] - w[s
                    //计算购买主件和附件1、附件2的最大价值
                dp[j] = max(dp[j - w[mindex] - w[sIndex1] - w[sIndex2]]
                                    dp[j]);
        }
    }
    //打印最大价值
  printf("%d", dp[n]);
   return 0;
}
```

逻辑航线培优教育，信息学奥赛培训专家。

扫码添加作者获取更多内容。

```
+ v[mindex] + v[sIndex1] + v[sIndex2],
```