链滴

记一次动态规划的算法问题

作者：18380422102
原文链接：https://ld246.com/article/1535439413556
来源网站：链滴

题目

题目描述

王强今天很开心，公司发给N元的年终奖。王强决定把年终奖用于购物，他把想买的物品分为两类：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 个附件附件不再有从属于自己的附件。王强想买的东西很多，为了不超出预算，他把每件物品规定了一个重度，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 10 元的整数倍）。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 j 1 ， j
2 ，……， j k ，则所求的总和为：

v[j 1]*w[j 1]+v[j
]*w[j 2]+ … +v[j k]*w[
k] 。 （其中 * 为乘号）

请你帮助王强设计一个满足要求的购物单。

输入描述:

输入的第 1 行，为两个正整数，用一个空格隔开：N m

（其中 N （ <32000 ）表示总钱数， m （ <60 ）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 3 个非负整数 v p q

（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 ）， q 表示该物品主件还是附件。如果 q=0 ，表示该物品为主件，如果 q>0 ，表示该物品为附件， q 是所属主件的编）

输出描述:

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（ <200000 。

实例:

输入:

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

输出:
2200

解法思路

这个题是一个动态规划的问题，假定总价格为FP,商品数量为FN，单个商品的价格为P，重要程度为I编号为Q;

先不考虑附件问题。当总价格不变，且商品价格都不超过FP时(P < FP):

一件商品　　f(1,FP) = P1 * I1

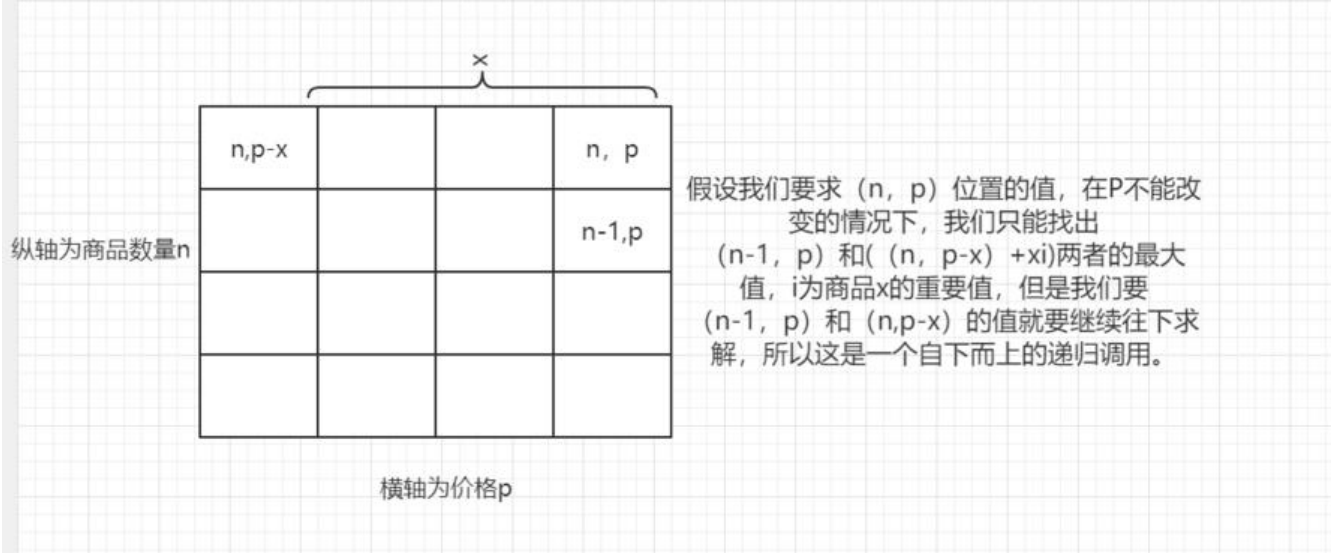
两件商品　　f(2,FP) = Max(f(1,FP), f(1,FP-P2) + P2 * I2)

三件商品　　f(3,FP) = Max(f(2,FP), f(2,FP-P3) + P3 * I3)

...

FN件商品　　f(FN,FP) = Max(f(FN-1,FP), f(FN-1,FP-FNP) + FNP * FNI)

上面的意思不知道能不能看懂，不过没关系，我这里解释一下，在决定是否要买一个价值500的物品，我们要判断（FP-500）的价格下的最大值加上这500的值是否大于不买这500的时候的最大值，转方程就是f(FN,FP) = Max(f(FN-1,FP), f(FN-1,FP-P(FN-1)) + FNP * FNI)其中FNP为FN的价格，就是面的500，FNI为500的商品的重要程度。



代码

```
import java.util.Scanner;

public class demo16 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        //题目规定价格都是10的整数，所以所有价格都除10，方便后面的遍历
        int fullPrice = scanner.nextInt()/10;
        int shopNum = scanner.nextInt();
        int[] prices = new int[shopNum+1];
        int[] importance = new int[shopNum+1];
        int[] q = new int[shopNum+1];
        for (int i = 1;i<=shopNum;i++){
            prices[i] = scanner.nextInt() / 10;
            importance[i] = scanner.nextInt() * prices[i];
            q[i] = scanner.nextInt();
        }
        int[][] res = new int[shopNum+1][fullPrice+1];
        for (int i = 1;i<=shopNum;i++){
            for (int j=1;j<=fullPrice;j++){
                //不是附件的情况
                if (q[i] == 0) {
                    if (prices[i] <= j){
                        //从数量1到N到价格1到N，依次求出所有情况的最大值，一直到要求的价格和数量
                        res[i][j] = Math.max(res[i-1][j],res[i-1][j-prices[i]]+importance[i]);
                    }
                } else {
                    //如果是附件，则需要考虑该附件的的价格
                    if (prices[i] + prices[q[i]] <= j){
                        res[i][j] = Math.max(res[i-1][j],res[i-1][j-prices[i]]+importance[i]);
                    }
                }
            }
        }
        System.out.println(res[shopNum][fullPrice]*10);
    }
```

原文链接：记一次动态规划的算法问题

```
}
```

原文链接：记一次动态规划的算法问题