背包专题之分组背包

1272：分组背包

题目描述

一个旅行者有一个最多能装V公斤的背包，现在有n件物品，它们的重量分别是W1，W2，...,Wn，它们的价值分别为C1,C2,...,Cn。这些物品被划分为若干组，每组中的物品互相冲突，最多选一件。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

输入

第一行：三个整数，V(背包容量，V≤200)，N(物品数量，N≤30)和T(最大组号，T≤10)；

第2..N+1行：每行三个整数Wi,Ci,P，表示每个物品的重量，价值，所属组号。

输出

仅一行，一个数，表示最大总价值。

输入样例

```
10 6 3
2 1 1
3 3 1
4 8 2
6 9 2
2 8 3
3 9 3
```

输出样例

```
20
```

解析

分组背包与01背包最大的区别就是：每组内的物品仅能使用一件。

根据条件，我们建立数组dp[i][j]代表在前i组中，在j容量下的最大价值。

现在，假设前i-1组j容量的最大价值已经计算完毕，即dp[i-1][j]。第i组共有s件物品，那么我们所要做的，就是尝试在向背包中放入第i组的物品。那么放入的可能性如下：

不放入任何物品，放入1号，放入2号……放入s号，共计s+1种组合，然后比较这些可能性中价值最大的即可。

编码

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

using namespace std;

int bagV, n, m, c, t;

int w[1005];                    //物品的重量
int v[1005];                    //物品的价值
int g[1005][1005];     //物品的分组信息
int f[1005][1005];     //动态规划表

//读取物品信息
void ReadItemInfo() {
    //记录最大承重、物品数量、最大分组
    cin >> bagV >> n >> m;
    //记录每个物品的重量和价值
    for (int i = 1; i <= n; i++) {
        cin >> w[i] >> v[i] >> t;
        c = ++g[t][0];      //用数组g的第二维的0记录当前组下总的物品数量
       g[t][c] = i;        //用数组g的第二维1以后的数字，记录当前物品的实际索
    }
}

//动态规划
void Dp() {
    //1、从第一组开始遍历
    for (int k = 1; k <= m; k++) {
        //2、模拟01背包从第一个重量开始
       for (int j = 1; j <= bagV; j++) {
            //遍历每一组的数据
          //3、当a=0的时候，实际上就是不放入当前组的任何物品
          for (int a = 0; a <= g[k][0]; a++) {
                //取出组内的每一个物品,i存储的是这个物品的实际索引值
             int i = g[k][a];
                //如果能装的下，则装入
             if (j >= w[i]) {
                    //分别尝试向背包加入当前组内的物品
                 f[k][j] = max(f[k][j], f[k - 1][j - w[i]] + v[i]);
                }
                //装不下的时候，则需要做一次比较，很可能本组的其他物品已经计算
             else {
                    f[k][j] = max(f[k - 1][j], f[k][j]);
                }
            }
        }
    }
}
```

```
int main() {
    ReadItemInfo();
    Dp();
    //背包的最大值在最后一个格子中
    cout << f[m][bagV];
    return 0;
}
```

一维优化

同样，我们可以按照01背包一维优化的原理对分组背包进行优化，代码如下：

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

using namespace std;

int bagV, n, m, c, t;

int w[1005];                //商品的体积
int v[1005];                //商品的价值
int g[1005][1005];     //商品的分组信息
int f[1005];     //动态规划表


//读取物品信息
void ReadItemInfo() {
    //记录最大承重、物品数量、最大分组
    cin >> bagV >> n >> m;
    //记录每个物品的重量和价值
    for (int i = 1; i <= n; i++) {
        cin >> w[i] >> v[i] >> t;
        c = ++g[t][0];      //g[t][0] 记录当前组下有多少物品
        g[t][c] = i;        //记录当前组的物品id，索引从1开始
    }
}

//动态规划
void Dp() {
    //从第一组开始遍历
    for (int k = 1; k <= m; k++) {
        //模拟01背包倒序模拟重量
        for (int j = bagV; j >= 1; j--) {
            //遍历每一组的数据
            //a从0开始，包含不放入当前组物品的情况
            for (int a = 0; a <= g[k][0]; a++) {
                //取出组内的每一个物品
                int i = g[k][a];
                if (j >= w[i]) {
                    //组内比较
```

```
                    f[j] = max(f[j], f[j - w[i]] + v[i]);
                }
            }
        }
    }
}


int main() {
    ReadItemInfo();
    Dp();
    //背包的最大值在最后一个格子中
   cout << f[bagV];
    return 0;
}
```


g[1][0]　存储总量