# **[动态规划：分组背包和有依赖的背包](https://www.cnblogs.com/aininot260/p/9308736.html)**

我们先来看分组背包问题

有N件物品和一个容量为V的背包，第i件物品的重量为c[i]，价值为w[i],这些物品被划分成了若干组，每组中的物品互相冲突，最多选一件

问将哪些物品放入背包中可以使背包获得最大的价值

我们用f[k][v]表示前k种物品花费费用v所能取得的最大价值

给出状态转移方程：

f[k][v]=max{f[k-1][v],f[k-1][v-c[i]]+w[i]|物品i属于第k组}

可以看出，这个问题还是很显然的，下面给出完整的实现：

在实现的时候稍微皮了一下~~

#include<iostream>

#include<algorithm>

#include<cstring>

using namespace std;

const int maxn=105;

const int maxv=105;

const int maxt=15;

int N,V,T;

int v[maxn],w[maxn];

int f[maxv];

int a[maxt][maxn];

int main()

{

cin>>V>>N>>T;

for(int i=1;i<=N;i++)

{

int p;

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

a[p][++a[p][0]]=i; //存每一组的所有物品的编号

//a[p][0]表示第p组一共有几个物品

}

for(int i=1;i<=T;i++)

for(int j=V;j>=0;j--)

for(int k=1;k<=a[i][0];k++)

if(j-v[a[i][k]]>=0)

f[j]=max(f[j],f[j-v[a[i][k]]]+w[a[i][k]]);

cout<<f[V]<<endl;

return 0;

}

下面介绍有依赖的背包，参考题目为NOIP金明的预算方案

在这种背包中引入了主件和附件的概念，只有放了主件之后才能放与之相关联的附件

最开始我是用生成组合的形式来解决这个问题的

即以主件为基准，和每一种附件的情况进行捆绑，再放01背包

但是我们有更皮的解决方法

将每一个主件对应的附件集合先做一次01背包

下面给出一道例题来实现这个方法，本例题中的主件是没有价值的，如果主件有价值可以在这个的基础上进行简单的修改

#include<iostream>

#include<algorithm>

#include<cstring>

using namespace std;

const int maxn=55;

const int maxv=100005;

int tmp[maxv];

int f[maxn][maxv];

int N,V; //主件的数量，背包的容量

int main()

{

while(cin>>N>>V)

{

memset(f,0,sizeof(f));

for(int i=1;i<=N;i++)

{

memset(tmp,-1,sizeof(tmp));

int p,m; //主件的费用，附件的数量

cin>>p>>m;

for(int j=p;j<=V;j++)

tmp[j]=f[i-1][j-p];

int v,w;

for(int j=1;j<=m;j++)

{

cin>>v>>w; //每一个附件的重量和价值

for(int k=V;k>=v;k--)

if(tmp[k-v]!=-1)

tmp[k]=max(tmp[k],tmp[k-v]+w);

}

for(int j=V;j>=0;j--)

f[i][j]=max(tmp[j],f[i-1][j]);

}

cout<<f[N][V]<<endl;

}

return 0;

}