分组背包问题

问题

有N件物品和一个容量为V的背包。第 i 件物品的费用是$w[i]$，价值是$v[i]$。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。

算法

这个问题变成了每组物品有若干种策略：是选择本组的某一件，还是一件都不选。也就是说设$f[k][j]$表示前k组物品花费费用j能取得的最大权值，则有：

$$f[k][j] = max(f[k-1][j], f[k-1][j-c[i]] + w[i] | \text{物品} i \text{属于组} k)$$

题目描述

有N组物品和一个容量是V的背包。

每组物品有若干个，同一组内的物品最多只能选一个。

每件物品的体积是v_{ij}，价值是w_{ij}，其中i是组号，j是组内编号。

求解将哪些物品装入背包，可使物品总体积不超过背包容量，且总价值最大。

输出最大价值。

输入格式

第一行有两个整数N，V，用空格隔开，分别表示物品组数和背包容量。

接下来有N组数据：

每组数据第一行有一个整数S_i，表示第i个物品组的物品数量；

每组数据接下来有S_i行，每行有两个整数v_{ij},w_{ij}，用空格隔开，分别表示第i个物品组的第j个物品的体积和价值；

输出格式

输出一个整数，表示最大价值。

数据范围

$0 < N, V \leq 100$

$0 < S_i \leq 100$

$0 < v_{ij}, w_{ij} \leq 100$

输入样例

```
3 5
2
1 2
2 4
1
3 4
1
4 5
```

输出样例:

```
8
```

In [1]:
```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#ifdef __cplusplus  //曾经的C/C++, 使用这个宏
#define Max(a, b) ((a > b) ? (a) : (b))
extern "C" {
    void zupack(){
        freopen("dp04beibao06_01.in","r",stdin);
        int f[110];
        int v[110],c[110];
        memset(f,0,sizeof(f));  //初始化很重要
        int n,m;
        scanf("%d %d",&n,&m);
      //  printf("nm----> %d %d\n",n,m);
        for(int i=0;i<n;i++){
            int s;
            scanf("%d",&s);
            //printf("s------->%d\n",s);
            for(int j=1;j<=s;j++){
                scanf("%d %d",&v[j],&c[j]); //获取每一组数据
               // printf("%d %d\n",v[j],c[j]);
            }
            for(int j=m;j>=0;j--){ //01背包问题
                for(int k=1;k<=s;k++){ //从第i组选择一个最大的
                    if(j>=v[k]){
                        f[j]=Max(f[j],f[j-v[k]]+c[k]);
                    }
//printf("%d\n",f[j]);
                }
            }
        }
        printf("%d",f[m]);
    }
}
#endif
```

Out[1]:

In [2]:
```
zupack();
```

```
8
```

Out[2]: (void) nullptr

F[i][j]=max(F[i-1][j],F[i-1][j-V[i][k]]+W[i][k]) (k表示选择第i组里的第k件) <---------前i组，容积为j的背包获得的最大价值

In [1]:
```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<vector>
#include<algorithm>
using namespace std;
    static const int maxN=40;
    static const int maxV=300;
    static const int maxT=15;
    static const int inf=2147483647;
class zupack_v1{
public:
    class Item{
    public:
        int weight,value;
    };
    int n,V,T;
    vector<Item> C[maxT];
    int F[maxT][maxV]={0};
    int zupack_v2(){
        freopen("dp04beibao06_01.in","r",stdin);
        int a,b,c;
        cin>>T>>V;    //组数，最大容积
        for (int i=1;i<=T;i++){
            cin>>c;
            for(int j=1;j<=c;++j){
                cin>>a>>b;   //体积，价值
                C[i].push_back((Item){a,b});
            }
        }
        int Ans=0;
        for (int i=1;i<=T;i++){      //一组一组的递推
            for (int k=0;k<=V;k++){    //消费，体积，从小向大，从前向后推。不是恰好，是尽量好
                for (int j=0;j<C[i].size();j++){ //该组中的物品一个一个的拿出来
                    if (k-C[i][j].weight>=0)
                        F[i][k]=max(F[i][k],max(F[i-1][k],F[i-1][k-C[i][j].weight]+C[i][j].value));
                    else
                        F[i][k]=max(F[i][k],F[i-1][k]);
                    Ans=max(Ans,F[i][k]);
                }
                //cout<<F[i][k]<<' ';
            }
            //cout<<endl;
        }
        cout<<Ans<<endl;
        return 0;
    }
}
```

Out[1]:

In [2]:
```
zupack_v1 * pzupack_v1 = new zupack_v1();
pzupack_v1->zupack_v2();
delete pzupack_v1;
```

```
8
```

Out[2]: (void) @0x7fd4f4a9fb38

小结

分组的背包问题将彼此互斥的若干物品称为一个组，这建立了一个很好的模型。不少背包问题的变形都可以转化为分组的背包问题（例如有依赖的背包），由分组的背包问题进一步可定义“泛化物品”的概念，十分有利于解题。

例题：

Luogu 1757 通天之分组背包 (https://www.luogu.org/problemnew/show/P1757#sub)
HDU 1712 ACboy needs your help (http://acm.hdu.edu.cn/showproblem.php?pid=1712)

In [ ]: