背包方案数问题

求方案总数

对于一个给定了背包容量、物品费用、物品间相互关系（分组、依赖等）的背包问题，除了再给定每个物品的价值后求可得到的最大价值外，还可以得到装满背包或将背包装至某一指定容量的方案总数。对于一个给定了背包容量、物品费用、物品间相互关系（分组、依赖等）的背包问题，除了再给定每个物品的价值后求可得到的最大价值外，还可以得到装满背包或将背包装至某一指定容量的方案总数。对于这类改变问法的问题，一般只需将状态转移方程中的max改成sum即可。例如若每件物品均是完全背包中的物品，转移方程即为

$$f[i][j] = \sum f[i-1][j], f[i][j-w[i]]$$

初始条件$f[0][0] = 1$。事实上，这样做可行的原因在于状态转移方程已经考察了所有可能的背包组成方案。

最优方案的总数

这里的最优方案是指物品总价值最大的方案。以01背包为例。 结合求最大总价值和方案总数两个问题的思路，最优方案的总数可以这样求：$f[i][j]$意义同前述，$g[i][j]$表示这个子问题的最优方案的总数，则在求$f[i][j]$的同时求$g[i][j]$

题目描述

有N件物品和一个容量是V的背包。每件物品只能使用一次。

第i件物品的体积是v_i，价值是w_i。

求解将哪些物品装入背包，可使这些物品的总体积不超过背包容量，且总价值最大。

输出**最优选法的方案数**。注意答案可能很大，请输出答案模$10^9 + 7$的结果。

输入格式

第一行两个整数，N，V，用空格隔开，分别表示物品数量和背包容积。

接下来有N行，每行两个整数v_i, w_i，用空格隔开，分别表示第i件物品的体积和价值。

输出格式

输出一个整数，表示**方案数**模$10^9 + 7$的结果。

数据范围

$0 < N, V \leq 1000$

$0 < vi, wi \leq 1000$

输入样例

```
4 5
1 2
2 4
3 4
4 6
```

输出样例:

```
2
```

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 fnspack(){
        freopen("dp04beibao09_01.in","r",stdin);
        // 初始化时: f[0] = 0, 其余为-1
        // f[i]表示volume恰好是i时的最值。
        const int N = 1001, MOD = 1e9 + 7;
        int n, V;
        int v[N], w[N];
        int f[N], cnt[N];
        scanf("%d %d",&n,&V);
        for (int i = 0; i < n; i++) {
            scanf("%d %d",&v[i],&w[i]);
        }
        memset(f,-1,sizeof(f));
        f[0]=0;
        cnt[0] = 1;
        for (int i = 0; i < n; i++) {
            for (int j = V; j >= v[i]; j--) {
                if (f[j - v[i]] + w[i] > f[j]) {
                    cnt[j] = cnt[j - v[i]];
                } else if (f[j] == f[j - v[i]] + w[i]) {
                    cnt[j] += cnt[j - v[i]];
                    cnt[j] %= MOD;
                }
                f[j] = Max(f[j], f[j - v[i]] + w[i]);
            }
        }

        int maxw =0;

        for(int k=0;k<V+1;++k){
            if(f[k]>maxw)maxw=f[k];
        }
        int res = 0;
        for (int i = 0; i <= V; i++) {
            if (f[i] == maxw) {
                res += cnt[i];
                res %= MOD;
            }
        }
        printf("%d",res);
    }
}
#endif
```

Out[1]:

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

```
2
```

Out[2]: (void) nullptr

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 fnspack01(){
        freopen("dp04beibao09_01.in","r",stdin);
        int n,v;
        int vi[1001],wi[1001];
        int dp[1001],num[1001];
        const int mod=1e9+7;
        int i,j;
        while(scanf("%d %d",&n,&v)==2){
            for(i=1;i<=n;++i){
                scanf("%d %d",&vi[i],&wi[i]);
            }
            for(i=0;i<1001;++i){
                num[i] = 1;
            }
            memset(dp,0,sizeof(dp));
            for(i=1;i<=n;++i){
                for(j=v;j>=vi[i];--j){
                    if(dp[j]<dp[j-vi[i]]+wi[i]){   ///当加入当前物品价值更大
                        dp[j]=dp[j-vi[i]]+wi[i]; ///当前价值修改
                        num[j]=num[j-vi[i]]%mod; ///当前价值的数目也修改
                    }else if(dp[j]==dp[j-vi[i]]+wi[i]){///找到与最大价值一样的
                        num[j]=(num[j-vi[i]]+num[j])%mod;///最大价值数目增加
                    }
                }
            }
            printf("%d",num[v]);
        }
    }
}
#endif
```

Out[1]:

In [2]:
```
fnspack01();
```

```
2
```

Out[2]: (void) nullptr

泛化物品

定义

考虑这样一种物品，它并没有固定的费用和价值，而是它的价值随着你分配给它的费用而变化。这就是泛化物品的概念。

更严格的定义之。在背包容量为V VV的背包问题中，泛化物品是一个定义域为$0\cdots V$中的整数的函数h，当分配给它的费用为v时，能得到的价值就是$h(v)$。

这个定义有一点点抽象，另一种理解是一个泛化物品就是一个数组$h[0\cdots V]$，给它费用v vv，可得到价值$h[V]$。

一个费用为c价值为w的物品，如果它是01背包中的物品，那么把它看成泛化物品，它就是除了$h(c)=w$其它函数值都为0的一个函数。如果它是完全背包中的物品，那么它可以看成这样一个函数，仅当v被c整除时有$h(v)=v/c*w$，其它函数值均为0。如果它是多重背包中重复次数最多为n的物品，那么它对应的泛化物品的函数有$h(v)=v/c*w$仅当v被c整除且$v/c<=n$，其它情况函数值均为0。一个物品组可以看作一个泛化物品h。对于一个$0\cdots V$中的v，若物品组中不存在费用为v的的物品，则$h(v)=0$，否则$h(v)$为所有费用为v的物品的最大价值。有依赖的背包问题中每个主件及其附件集合等价于一个物品组，自然也可看作一个泛化物品。

泛化物品的和

如果面对两个泛化物品h和l，要用给定的费用从这两个泛化物品中得到最大的价值，怎么求呢？事实上，对于一个给定的费用v，只需枚举将这个费用如何分配给两个泛化物品就可以了。同样的，对于$0\cdots V$的每一个整数v，可以求得费用v分配到h和l中的最大价值$f(v)$。也即

$$f(v)=max(h(k)+l(v-k)|0<=k<=v)$$

可以看到，f也是一个由泛化物品h和l决定的定义域为$0\cdots V$的函数，也就是说，f是一个由泛化物品h和l决定的泛化物品。

由此可以定义泛化物品的和：h、l都是泛化物品，若泛化物品f满足以上关系式，则称f是h与l的和。这个运算的时间复杂度取决于背包的容量，是$O(V^2)$ 。 泛化物品的定义表明：在一个背包问题中，若将两个泛化物品代以它们的和，不影响问题的答案。事实上，对于其中的物品都是泛化物品的背包问题，求它的答案的过程也就是求所有这些泛化物品之和的过程。设此和为s，则答案就是$s[0\cdots V]$中的最大值。

背包问题的泛化物品

一个背包问题中，可能会给出很多条件，包括每种物品的费用、价值等属性，物品之间的分组、依赖等关系等。但肯定能将问题对应于某个泛化物品。也就是说，给定了所有条件以后，就可以对每个非负整数v求得：若背包容量为v，将物品装入背包可得到的最大价值是多少，这可以认为是定义在非负整数集上的一件泛化物品。这个泛化物品——或者说问题所对应的一个定义域为非负整数的函数——包含了关于问题本身的高度浓缩的信息。一般而言，求得这个泛化物品的一个子域（例如$0\cdots V$）的值之后，就可以根据这个函数的取值得到背包问题的最终答案。

综上所述，一般而言，求解背包问题，即求解这个问题所对应的一个函数，即该问题的泛化物品。而求解某个泛化物品的一种方法就是将它表示为若干泛化物品的和然后求之。

例题：

HDU 3810 Magina (http://acm.hdu.edu.cn/showproblem.php?pid=3810)

背包演示

背包演示网站 (http://karaffeltut.com/NEWKaraffeltutCom/Knapsack/knapsack.html)

In [ ]: