01背包问题

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 vi，价值是 wi。

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。
输出最大价值。

朴素版解法：二维空间解法
每件物品只能选一次，对于每种物品，我们有两种选择

1.不选 -> dp[i][j]=dp[i-1][j]
等于选前i-1个物品，空间为j情况下的最优解
2.选 -> dp[i][j]=dp[i-1][j-v[i]]+w[i]
如果选的话，前i-1个物品的体积最多为j-v[i]

```
//dp[i][j]表示前i个物品，背包容量是j的情况下的最大价值。
  for(int i = 1;i <=n; i++)
      for(int j = 0; j <= V; j++) //要从0开始
         if(v[i] <= j) dp[i][j] = max(dp[i-1][j],dp[i-1][j-v[i]]+w[i]);
          else dp[i][j] = dp[i-1][j];//这句容易忘
```

一维数组

```
for(int i = 1;i <=n; i++)
        for(int j = V;j>= v[i];j--)
            dp[j] = max(dp[j],dp[j-v[i]]+w[i]);
```

计数问题：

给定N个正整数$A_1, A_2, \ldots, A_N$，从中选出若干个数，使它们的和为M，求有多少种选择方案。

输入格式

第一行包含两个整数N和M。

第二行包含N个整数，表示$A_1, A_2, \ldots, A_N$。

```
for (int i = i; i <= n; i ++ )
    {
        for (int j = m; j >= v[i]; j -- ) dp[j] += dp[j - v[i]];
    }
```

完全背包问题

```
for(int i = 1; i <= n; i++){
        for(int j = v[i]; j <= V; j++)
            f[j] = max(f[j], f[j-v[i]] + w[i]);
    }
```

背包问题求具体方案

```
per(i, n, 1){//从后往前枚举，保证字典序
    rep(j, 0, m){
        f[i][j] = f[i+1][j];
        if(j >= v[i]) f[i][j] = max(f[i][j], f[i+1][j-v[i]] + w[i]);
    }
}
int vol = m;
rep(i, 1, n){//检查第i个物品是否选择
    if(vol < v[i]) continue;
    if(f[i][vol] == f[i+1][vol - v[i]] + w[i]){
        cout << i << ' ';
        vol -= v[i];
    }
}
```

求最优解方案数

```
rep(i, 0, n) p[i] = 1; // 注意设置为1
    rep(i, 1, n){
        per(j, m, v[i]){
            int t = f[j - v[i]] +w[i];
            if(f[j] < t){
                f[j] = t;
                p[j] = p[j-v[i]];
            }
            else if(f[j] == t){
                p[j] += p[j-v[i]];
            }
            p[j]%= mod;
        }
    }
```