## 01背包

最基本的背包问题

最大W的重量，N种物品 每个只能拿一件

每件物品有重量以及价值v[i] (以是0可以相同)

比如

W=10 N=5

2 1

0 4

6 5

2 4

4 3

设dp[i][j]是前i个物品，前j的重量的情况下，最大能拿的价值总和

自然而然对于前0个物品，dp[0][j]永远等于0

对于前1个物品，dp[1][j]要分情况，如果第一个物品的种类w小于等于j,那么dp[1][j]=v[i],否则拿不了就是0

后面的规律可以推出来，下面是表

前几个物品

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号/前j的重量 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 3 | 4 | 5 | 5 | 5 | 5 | 5 | 9 | 10 | 10 | 10 |
| 4 | 4 | 5 | 8 | 9 | 9 | 9 | 9 | 9 | 13 | 14 |
| 5 | 4 | 5 | 8 | 9 | 9 | 9 | 11 | 12 | 13 | 14 |

Dp[i][j]=max( dp[i-1][j] , dp[i-1][j-w[i]]+v[i] )

求dp[N][W]即可

优化成一维数组：发现每次内层循环，只要使用2层的数组，每次使用的是上一层j靠前的

**从后向前推**即可

## 完全背包：

最大W的重量，N种物品 每个只能拿无限件

每件物品有重量以及价值v[i] (不能0可以相同)

可以把它拆成很多个（W/w[i]个）01背包，这样理论上可以求解， w[i]很小时候，时间复杂度太大了，且价值W给个10000，N给个100，10000\*10000/w[i]\*100空间估计都得多了

正确方法是从新推dp方程

f[i+1][j]=max(f[i][j-k\*weight[i+1]]+k\*value[i+1])，其中0<=k<=V/weight[i+1]

和那个差不多，只不过乘了个k，表示尽可能多的取物品

优化成一维数组：发现每次内层循环，只要使用2层的数组，每次使用的是上一层j靠后的，**正着从前向后**推即可

多重背包：完全背包的转移方程：

f[i+1][j]=max(f[i][j-k\*weight[i+1]]+k\*value[i+1])，其中0<=k<=V/weight[i+1]

中，变一下，限制k的数量