01背包和完全背包问题

主讲人：邓哲也

01背包问题

有 N 件物品和一个容量为 V 的背包。

第 i 件物品的费用是 c[i]，价值是 w[i]。

求将哪些物品装入背包可以使得价值总和最大。

01背包问题

这是最基础的背包问题。

“01”就是指每种物品要么是选，要么是不选。

我们定义状态 f[i][j] 表示从前 i 件物品中选出容量为 j 的背包能获得的最大价值。

思考转移方程。

01背包问题

f[i][j] = max(f[i-1][j], f[i-1][j-c[i]]+w[i])

也就是考虑第 i 个物品是选还是不选。

时空复杂度都是 O(NV)

空间优化

f[i][j] = max(f[i-1][j], f[i-1][j-c[i]]+w[i])

注意到每次我们都是从 f[i-1] 递推到 f[i]

可以只用 O(V) 的空间存下每一步的 f 吗?

```
for i = 1 .. n
      for j = V .. 0
            f[j] = max(f[j], f[j-c[i]]+w[i])
```

思考为什么是倒序。

空间优化

```
for i = 1 .. n
      for j = V .. 0
            f[j] = max(f[j], f[j-c[i]]+w[i])
```

因为每次算 f[j] 的时候，f[V..j+1] 都已经是新的一轮的值了，而 f[j] 和 f[j-c[i]] 是上一轮的值，分别对应原来的 f[i-1][j] 和 f[i-1][j-c[i]]。

01背包问题

初始化时 f[0]=0, f[1..V]=+∞

如果要求背包要装满，答案就是 f[V]

如果可以不装满，答案就是 max{f[1..V]}

完全背包问题

有 N 件物品和一个容量为 V 的背包。

第 i 件物品的费用是 c[i]，价值是 w[i]。

每种物品都有无限件可用。

求将哪些物品装入背包可以使得价值总和最大。

完全背包问题

与 01背包问题不同的是，这里每种物品可以选任意件出来放进背包。

我们定义状态 f[i][j] 表示从前 i 件物品中选出容量为 j 的背包能获得的最大价值。

思考转移方程。

完全背包问题

f[i][j] = max(f[i-1][j-k•c[i]] + k•w[i] | 0≤k•c[i]≤j)

这样的时间复杂度已经到了 O(NV•Σ(V/c[i]))

类比一下 01背包问题的一维数组解法，看看能不能优化。

完全背包问题

```
for i = 1 .. n
        for j = 0 .. V
                f[j] = max(f[j], f[j-c[i]]+w[i])
```

只要把 01背包问题里枚举体积的部分倒着枚举即可。

因为每次算 f[j] 的时候，f[j-c[i]] 表示的是用前 i 个物品（可能已经拿过第 i 个物品了）凑出体积为 j-c[i] 的最大价值。

不仅空间变成了O(V)，时间复杂度也变为了 O(NV)。

类比

01背包问题代码

```
for i = 1 .. n
    for j = V .. 0
        f[j] = max(f[j], f[j-c[i]]+w[i])
```

完全背包问题代码

```
for i = 1 .. n
    for j = 0 .. V
        f[j] = max(f[j], f[j-c[i]]+w[i])
```

下节课再见