

# 背包问题 by iuiou

所谓背包问题，有一个非常经典的模型：现在有 $n$ 个物品，第 $i$ 个物品有一个权值 $e_i$ 有一个费用 $w_i$ ，每种物品只能选取1次，现你手中

只有 $m$ 元钱，问如何选取物品才会使手中物品的权值达到最大？

简要分析：这种有代价，有选择然后取最值的题目，一般只有两种思路，一种是贪心，一种是动态规划。两者的区别我一直非常迷惑，知道看到一个大佬多写的文章。总的来说，贪心注重由固定的规律(多是能证明或者显而易见)由局部解推出最终解，每一步都能保证前一步的答案是最优的，而动态规划，是由每种状态的转移，来从每一个比当前小的状态转移出大状态的解。并不能保证前一个状态的结果是最优的，所以要遍历所有情况。其实就相当于找一种方法来遍历所有情况，而其核心在于初始化和转移。

回到这题，既然放在动态规划肯定是动态规划啦，这个题目明显找不到一个确定的贪心策略，能保证每一步都是最优的，至少我看不出来，所以考虑动态规划。

核心转移 $dp[m] = \max(dp[m], dp[m - w_i] + e_i)$

上面这个式子太显然了没啥好说的 $dp[i]$ 表示 $i$ 元钱能得到的最大权值，然后有花费和赚。

循环：

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

上面是01背包，每个物品选一次。

可重背包：只需要改成for(int j=a[i];j<=v;j++)即可

多重背包：即限定了每种个数，可以考虑转化为01背包，把相同的看成一个，还可以采用二进制分组的方法，简化复杂度。

分组背包：有许多组，每组有若干个，每组只能选一个。最外面枚举组别，考虑在最里面一层再加入一个循环判断选这个组哪个物品。

有依赖背包：即选一个必选依附的那一个，转化状态即可，将不同的状态看成物品，即加入选 $b$ 或选 $c$ 就要选 $a$ 可以直接转化为只选 $a$ 只选 $a$ 和 $b$ 只选 $a$ 和 $c$ 选则 $a,b,c$ 这样四种情况，分别看作一种物品，做分组背包即可。

背包问题大致就上面这些。

From:  
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:  
<https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:manespace:%E8%83%8C%E5%8C%85%E9%97%AE%E9%A2%98&rev=1594886743>

Last update: 2020/07/16 16:05