# 最小购物问题建模

## 确定状态

### 规划子问题

对于最小的费用购物问题，我们首先要确定它的状态。什么意思呢？就是说，我们现在的目标是——花最少的钱，买到自己规定数目和商品种类的东西。那么，他肯定要有优惠方案，才有优惠这样一说。比如说，花和花瓶，单买一朵花是2元，单买一个花瓶是5元。那么，商家推出优惠方案：1、买三朵花只要5元。2、买一朵花和两个花瓶要10元。当然还可能有第三，第四种方案等等。我们可以把这些方案，想象成我们平时付钱一样，一个方案可以对应成平时买东西时一张面值固定的钞票，然后我们想要，花最少张数的钞票付完钱。对应到这里呢就是。选择多种优惠方案后，可以使得，我们能花最少的X元买下我们的东西。

那么，我们可以设F【x】=选择多种优惠方案后花费最少的钱X。

### 最后一步

对于动态规划的问题，我们并不需要关心他中间到底选了哪些方案才导致的花钱最少，但是我们可以知道，他最后一步到底有哪些选择。当购买的东西很多之后，可能有的优惠方案使用了多次，但不管怎么使用，如果我们让它一个方案一个方案的购买，那么在它完成购买的最后一步，必然时选择了众多方案中的一个，这里以我们之前举例的两个方案为例，那么他最后一步之前的状态就是：

F[x]=Min{[x-5]+1，[x-10]+1}

它在做最后一步的选择的时候，其必然是从基础方案中的一个做出的最优选择。

### 转移方程

当然，实际描述的时候，基础的优惠方案还是太小，我们把它扩大成五个的形式（也可实际情况更改），那么，对这个问题的最后一步的状态也可以统括为对整个问题的转移方程：

F [a, b, c, d, e] = Min {F [a-S1, b-S2, c-S3, d-S4, e-S5] + SaleCost [S]}

### 初始条件和边界问题

当然，一份算法我们还要将它的初始条件和边界情况考虑进去。初始条件我们可以理解为特殊情况。那么在这个问题里面初始条件（特殊情况）是什么呢？首先，如果我们什么都不买，就自然不用付一分钱，或者我们买的东西数目每一样，都达不到优惠价格，是不是也是原价购买。那么初始条件就界定为：

F[0]=0；

F [a, b, c, d, e] = Cost [1]\*a+Cost [2]\*b+Cost [3]\*c+Cost [4]\*d+Cost [5]\*e

那么边界问题我们也很容易想到：优惠的方案数，就框定了这个问题的边界。

### 计算顺序

动态规划和递归，有些异曲同工之处，那就是遍历。但是，动态规划不同于递归的地方就在于，它会省去重复子问题的计算，按照这个思想，他当然要从小到大计算，这样它才能够在计算到大数量级的时候前面的数据有迹可查。

F【i】=Min{f[j[0]][j[1]][j[2]][j[3]][j[4]]+O[m][n]}