**问题描述:**

商店t每利商品都有标价。例如，一朵花的价格是2元、一个花瓶的价格是5元。为了吸引顾客，商店提供了一组优惠成品价,优惠商品是把一种或多种商品分成一组, 并降价销售。例如，3朵花的价格不是6元而是5元2个花瓶加1朵花的优惠价是10元。设计一个算法，计算出某一顾客所购商品应付的最少费用。

**动态规划的思想:**

动态规划方法的基本思想是：将待求解的问题分解成若干个相互联系的子问题，然后按顺序求解各子问题，最后一个子问题就是初始问题的解。

动态规划最关键的地方在于，存在一种子问题和最理想子结构重叠的情况。这个说法非常抽象，简单来说就是子问题的最优解被重复使用。

当相同的子问题的解决方案被重复利用。在动态规划中，对于重复出现的子问题，只在第一次遇到的时候对它进行求解，并把答案保存起来，让以后再次遇到时直接引用答案，不必重新求解。因此，如果这个问题是没有共同的(币叠)子问题，动态规划是没有用的。

**适合动态规划解决的问题的特点:**

动态规划算法的问题及决策应该具有两个性质:最优化原理、无后向性。

-1)最优化原理(或称为最佳原则、最优子结构)；

-2)无后向性(无后效性):某阶段状态一旦确定以后，就不受这个状态以后决策的影响，即某状态以后的过程不会影响以前的状态，只与当前状态有关。

**问题分析:**

设欲可俱购买的物品有n种，每种物品购买的数量为a1,a2,a3,...,an设最小购买费用为mincost(a1,a2,a3,...,an).

设可以采用的优惠组合有m种，第m种优惠组合所需要的物品个数为a1m,a2m,a3m,...,anm在没有使用优惠组合的情况下，mincost(a1,a2,a3,...,an)为即a1\*c1 + a2\*c2 + … + an\*cn

显然这不是本向题的最优解。

在采用优惠组合的情况下，假设采用第1种优惠组合，则首先待到的花费为cost(a1 a11,a2-a21,a3-a31,…an-an1)+offer(1).这样一来就产生了一个子问题，即为求cos t(a11,a2-a21,a3-a31,…an-an1)。同理可知cos t(a11,a2-a21,a3-a31,…an-an1)可以继续推出子问题。求由此可见本问题可以划分为多个子问题。

**1.最优子结构:** 最优子结构在这里即是子问题的最优解。  
 mincost(a1,a2,a3...an)=min(mincost(a1 a11,a2-a21,a3-a31,…an-an1)+offer(1),……, min(mincost(a1 a1m,a2-a2m,a3-a3m,…an-anm)+offer(m),可知当前决策下的mincost 一定是m种优惠组合中的最优选择。因此本问题的决策满足最优子结构。而求出mincost(a1,a2,a3...an)前我们还需要求出min(mincost(a1 a11,a2-a21,a3-a31,…an-an1)+offer(1),……, min(mincost(a1 a1m,a2-a2m,a3-a3m,…an-anm)+offer(m)。同理可知我们最终要求mincost(1,0,0…,0)或mincost(0,1,...,0)因此本问题出现了子问题的最优解被重复使用的情况。  
  
**2.无后效性:** 显而易见，mincost(a1-a11,a2-a21,a3-a33,...,an-an1),......,mincost(a1-a1m,a2-a2m,a3-a3m,....,an-anm的取值与mincost(a1,a2,a3,…,an)无关。后一项的选择对前面几项不会造成影响，因此本问题具有无后效性。  
 综上所述，本问题的决策同时满足最优子结构的存在和无后效性，同时又有重叠子问题的情况，因此本问题是典型的动态规划决策问题。  
  
**算法描述:**  
 动态规划的主要思想是以空间换时间，即以一定的空间存储规划决策过程中的各个阶段的状态，在以后的决策迭代过程中就可以直接调用之前各阶段的结果,这样大大减低了时间复杂度。  
  
**动态规划和核心算法伪代码形式:**

Mincost(){  
   for(依次遍历优惠组合)

{  
 if(num1>=offer\_num1且…… 且numm>=offer\_numm)

{  
 if(min>mincost[num1-offer\_num1] …… [numm-offer\_num1]+offer()

{

if(mincost[num1-offer\_num1] …… [numm-offer\_num2]>0)  
       将min设为当前计算出的费用:  
 else  
 将min设为Mincost(余F的购物数量) +offer();

}  
 }  
 else  
 比较或将min设为购买数量乘以单价:

}

将mincost[]设为当前计算得到的min;

返回当前计算的子问题F的mincost[0];

}

**本算法的状态转移方程为:** mincost(a1,a2,a3...an) = min(mincost(a1-a1m,a2-a2m,a3-a3m,…an-anm)+offer(m))