**MinCost算法:**

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))

**分析:**

**1.时间复杂度分析**

设基本语句执行的执行次数为f(n)，有

f(n) = c1 \* n2

因为T(n) = f(n) = c1 \* n2 <= cn2,其中c1,c均为常数，所以该算法的时间复杂度为：

T(n) = O(n2)

**2.空间复制度分析**

本算法需要开辟n个空间进行存放n个商品的购买数量和单价，因此本算法的空间复杂度为：

S(n) = O(n)