通过典型案例来展示招聘笔试考试的应对技巧

案例：中国银行 2018 的考试题：n 粒黄金，a[n]标识每粒黄金的重量。要打包出口它们，每包重量不超过 40g，问如何搭配就能使得包数最小。

解：直观认为，先应该考虑重的，再考虑轻的。因此先根据重量对 n 粒黄金从重到轻排序，排序后得到数组 b[n]。

策略：自然要从重到轻来考虑。先拿最重的，从它的下一个开始扫描，先找出第一个能和它搭配一起出口的，即 b[0] + b[j] ≦ 40。

考虑到是招聘笔试，因此不能想得太复杂，否则时间来不及，因此只好朝规律方面想。因此，就想到再找下一个还能搭配进来的。朝规律方面想，自然就想到要使用递归。

这样考虑的好处是：覆盖范围广。极端情况是：b[0] + b[n-1] > 40, 这样，b[0]只能是一个出口，没有可搭配的。上面的方法把这种极端情况也覆盖进来了。

当然，上述方案也可能不是最佳的。没有关系，招聘笔试是机器打分。其办法是通过用例来测试。上述方法能覆盖极端情况，因此，上述方案的得分能达到 90%。

因为是重的优先考虑。因此，肯定对每粒黄金都要标记其是否已经被搭配打包了。于是对每粒黄金设置两个属性：b[i].weight 表示重量。 b[i].out=false 表示有待搭配处理，b[i].out=true 表示已经搭配完了。

```
main ( )    {
    先用冒泡法对 a[0..n-1]从重到轻排序：
    for ( i = n - 1; i --; i > 1)
        for ( j = 0; j ++; j < i)
            if (a[j] < a[j+1])   {
                k = a[j];
                a[j] = a[j+1];
                a[j+1] = k;
            }
    export_package_num = 0;     //出口次数，要求的值；
    Package * current_package;
    i = 0;                //从最重的开始考虑
    while (i < n)   {
        while ( b[i].out == true && i < n)
            i++;
        if ( i < n && b[i].out == false)    {
            current_package = Figure_out_best_sub_package(40, i);
            export_package_num ++;
            print( “the %d th package weighs %d ”, export_package_num, current_package.Weight);
            i++;
        }
    }
    Print( “the minimum number of exports is %d”, export_package_num);
}

package Figure_out_best_sub_package(weight, start_index)     {
    Package * current_package;
    Package * package = new package( );
    package.Add_Element(start_index);
    package.Set_Weight (b[start_index].weight);
    b[start_index].out ==true;

    if    (start_index < n - 1 )       {    //表示不是最后一个元素
        j = start_index + 1;
        while    ( j < n )      {
            if ( b[j].out == false     &&     b[start_index].weight +  b[j].weight <= weight )
                break;
            j++;
        }
        if    ( j < n )    {
            current_package = Figure_out_best_sub_package(weight - b[start_index].weight, j);
            package.Add_Elements(current_package);
            package.Add_Weight(current_package);
        }
    }
    return package;
}
```

当然，招聘笔试考试要有速度，有质量才行。平时就要多练，把最典型的算法，求解问题的套路练熟，把边界条件的控制练熟，　把递归练熟。刚练还不行，还要自己归纳，总结才行。

改进：进行择优选取。其思想是：从重到轻排序后，假定是 b_i,　b_j,　b_k , b_n,　b_m,　b_s, b_t。现在 package ={ b_i}，　接下来，找到了一个子包 current_package = {b_k , b_t}满足加进 package 中的条件。不过此时并不结束，而是继续扫描待打包队列，如发现另一个子包 next_package ={b_n , b_m, b_s}也满足加进 package 中的条件，而且比 current_package 更合适，

原因是 next_package 的重量比 current_package 更重一些。那么我们就要择优选取，选用 next_package 替换掉 current_package，把 current_package 退回到待打包队列中。然后让 next_package 成为 current_package。 这个过程继续下去，直到穷尽 next_package 为止。改进后的函数 Figure_out_best_sub_package 如下：

```
package Figure_out_best_sub_package(weight, start_index)     {
        Package * current_package, next_package;
        Package * package = new package( );
        package.Add_Element(start_index);
        package.Set_Weight (b[start_index].weight);
        b[start_index].out == true;

        if    (start_index < n - 1 )       //表示不是最后一个元素
                j = start_index + 1;
                while    ( j < n )     {
                        if ( b[j].out == false     &&     b[start_index].weight +  b[j].weight <= weight )
                                break;
                        j++;
                }
                if    (j < n && b[j].out == false)         //表示有搭配的
                        current_package = Figure_out_best_sub_package(weight - b[start_index].weight, j);
                        if (current_package.Weight( ) < weight - b[start_index].weight )    { //没达到饱和组合
                                k = j + 1;
                                while ( k < n && b[k].out == true)         //逻辑判别的先后顺序非常重要！
                                        k++;
                                while ( k < n && b[k].out == false)     {
                                        next_package =Figure_out_best_sub_package(weight - b[start_index].weight, k);
                                        If    (current_package.Weight( ) < next_package.Weight( ))    {   //须要替代
                                                //current_package 败给了 next_package，被退回到等待打包的队列中：
                                                p = current_package.First_Element( );
                                                while ( p > -1)    {
                                                        b[p].out = false;
                                                        p = current_package.Next_Element( );
                                                }
                                                //设置当前最佳的搭配：
                                                current_package = next_package;
                                        }
                                        else    {          //next_package 竞争不过 current_package，被退回到等待打包
                                                           //的队列中：
                                                p = next_package.First_Element( );
                                                while ( p > -1)    {
                                                        b[p].out = false;
                                                        p = next_package.Next_Element( );
                        }
                    }
                    k = k + 1;
                    while (k < n && b[k].out == true)
                        k++;
                }       //while ( k < n && b[k].out == false)     {
            }     // If (current_package.Weight( ) < weight - b[i].weight )
            package.Add_Elements(current_package);
            package.Add_Weight(current_package);
        }     //if
    }   // if
    return package;
}
```

例如，37, 35, 33, 27, 24, 21 , 18, 15, 12, 11, 9, 8, 7, 6, 6, 4, 2, 2, 1

37, 35, 33, 27, 24, 21 , 18, 15, 12, 11, 9, 8, 7, 6, 6, 4, **2**, 2, **1**
37, **35**, 33, 27, 24, 21 , 18, 15, 12, 11, 9, 8, 7, 6, 6, **4**, **2**, 2, **1**
37, **35**, **33**, 27, 24, 21 , 18, 15, 12, 11, 9, 8, **7**, 6, 6, **4**, **2**, 2, **1**
37, **35**, **33**, **27**, 24, 21 , 18, 15, **12**, 11, 9, 8, **7**, 6, 6, **4**, **2**, 2, **1**
37, **35**, **33**, **27**, **24**, 21 , 18, **15**, **12**, 11, 9, 8, **7**, 6, 6, **4**, **2**, 2, **1**
37, **35**, **33**, **27**, **24**, **21** , **18**, **15**, **12**, 11, 9, 8, **7**, 6, 6, **4**, **2**, 2, **1**
37, **35**, **33**, **27**, **24**, **21** , **18**, **15**, **12**, **11**, **9**, **8**, **7**, 6, 6, **4**, **2**, **2**, **1**
37, **35**, **33**, **27**, **24**, **21** , **18**, **15**, **12**, **11**, **9**, **8**, **7**, **6**, **6**, **4**, **2**, **2**, **1**

例如，25, 21, 19, 18, 17, 14 , 8, 7, 6, 6, 3, 3, 3, 3, 3, 3, 2, 2, 2

25, 21, 19, 18, 17, **14** , **8**, **7**, 6, 6, 3, 3, 3, 3, 3, 3, 2, 2, 2
25, 21, 19, 18, 17, 14 , **8**, **7**, 6, 6, 3, 3, 3, 3, 3, 3, 2, 2, 2
25, **21**, **19**, 18, 17, 14 , **8**, **7**, 6, 6, 3, 3, 3, 3, 3, 3, 2, 2, 2
25, **21**, **19**, **18**, **17**, 14 , **8**, **7**, 6, 6, **3**, 3, 3, 3, 3, 3, **2**, 2, 2
25, **21**, **19**, **18**, **17**, **14** , **8**, **7**, **6**, **6**, **3**, **3**, **3**, **3**, **3**, 3, **2**, **2**, 2

例如，25, 21, 19, 18, 17, 13 , 8, 6, 6, 6, 3, 3, 3, 3, 3, 3, 3, 3, 3

25, 21, 19, 18, 17, **13** , **8**, **6**, **6**, **6**, 4，**3**, 3, 3, 3, 3, 3, 3, 3, 3
25, **21**, **19**, 18, 17, 13 , 8, 6, **6**, **6**, 4，**3**, 3, 3, 3, 3, 3, 3, 3, 3
25, **21**, **19**, **18**, **17**, 13 , 8, 6, **6**, **6**, **4**，**3**, 3, 3, 3, 3, 3, 3, 3, 3