理解：

贪心法在解决问题的策略上目光短浅，只根据当前已有的信息就做出选择，而且一旦做出了选择，不管将来有什么结果，这个选择都不会改变。

一句话：不求最优，只求可行解。

判断：

 对于一个具体的问题，怎么知道是否可用贪心算法解此问题，以及能否得到问题的最优解?

　　我们可以根据贪心法的2个重要的性质去证明：**贪心选择性质和最优子结构性质**。

　　1、贪心选择

　　什么叫贪心选择？从字义上就是贪心也就是目光短线，贪图眼前利益，在算法中就是只根据当前已有的信息就做出选择，而且以后都不会改变这次选择。（这是和动态规划法的主要区别）

　　所以对于一个具体问题，要确定它是否具有贪心选择性质，必须证明每做一步贪心选择是否最终导致问题的整体最优解。

　　2、最优子结构

　　当一个问题的最优解包含其子问题的最优解时，称此问题具有最优子结构性质。

这个性质和动态规划法的一样，最优子结构性质是可用动态规划算法或贪心算法求解的关键特征。

**区分动态规划**

**动态规划**算法通常以自底向上的方式解各子问题，是递归过程。

**贪心算法**则通常以自顶向下的方式进行，以迭代的方式作出相继的贪心选择，每作一次贪心选择就将所求问题简化为规模更小的子问题。

**以二叉树遍历为例：**

**贪心法**是从上到下只进行深度搜索，也就是说它从根节点一口气走到黑的，它的代价取决于子问题的数目，也就是树的高度，每次在当前问题的状态上作出的选择都是1，不进行广度搜索，所以最终它得出的解不一定是最优解，很有可能是近似最优解。

　　而**动态规划法**在最优子结构的前提下，从树的叶子节点开始向上进行搜索，并且在每一步都根据叶子节点的当前问题的状况作出选择，从而作出最优决策，所以她的代价是子问题的个数和可选择的数目，它求出的解一定是最优解。

**一般求解过程**

　　使用贪心法求解可以根据以下几个方面进行（最终也对应着每步代码的实现），以找零钱为例：

　　1、候选集合(C)

　　　　通过一个候选集合C作为问题的可能解。（最终解均取自于候选集合C）

　　　　例如，在找零钱问题中，各种面值的货币构成候选集合。

　　2、解集合(S)

　　　　每完成一次贪心选择，将一个解放入S，最终获得一个完整解S

　　3、解决函数(solution)

　　　　检查解集合S是否构成问题的完整解。

　　　　例如，在找零钱问题中，解决函数是已付出的货币金额恰好等于应付款。

　　4、选择函数(select)

　　　　即贪心策略，这是贪心法的关键，选择出最有希望构成问题的解的对象。（这个选择函数通常和目标函数有关）

     　　   例如，在找零钱问题中，贪心策略就是在候选集合中选择面值最大的货币。

　　5、可行函数(feasible)

　　　　检查解集合中加入一个候选对象是否可行。（加入下一个对象后是不是满足约束条件）

　　　　例如，在找零钱问题中，可行函数是每一步选择的货币和已付出的货币相加不超过应付款。

实现：

1. Greedy(C)  *//C是问题的输入集合即候选集合*
2. {
3. S={ }; *//初始解集合为空集*
4. while (not solution(S))  *//集合S没有构成问题的一个解*
5. {
6. x=select(C);    *//在候选集合C中做贪心选择*
7. if feasible(S, x)  *//判断集合S中加入x后的解是否可行*
8. S=S+{x};
9. C=C-{x};
10. }
11. return S;
12. }