Divide and Conquer 分支法

- 将数组一分为二，继续分解，
  - 最终得到Single Element的数组 开始进行返回
  - 返回途中可以查询每个数组中的特征，并且记录下来
  - 返回特征或者数组内的元素 必须同一类 type
  - 到了顶层后 则完成算法

Self-Defined Template

```
class Solution:

    def mainFunction(self, ...):
        # make use of the helper function
        helper_function(left, right) # parameters may be vary, but be sure of the param before starting it

    def helper_function():
        if base case:
          return 数组中的元素

        #分解 （有必须的话 可以使用二分的方法
        mid = (right - left) // 2 + left
        left = ... (递归)
        right = ... （递归）

        if left == right:
            # 特征条件

        #如果相等的时候 不存在特征条件的话
        #逐步搜索答案，并且找到最终的特征
        #返回特征，完成递归
```

经典题目

169. Majoirty Element

```
class Solution:
    def majorityElement(self, nums, lo=0, hi=None):
        # 缩小到小问题 然后找数组中最大的数值
        def majority_element_rec(lo, hi):
            # base case; the only element in an array of size 1 is the majority
            # element.
            if lo == hi:
                return nums[lo]
              # 当 lo 与 hi 指向了同一个位置时
            # recurse on left and right halves of this slice.
            mid = (hi-lo)//2 + lo
            left = majority_element_rec(lo, mid)
```

```
            right = majority_element_rec(mid+1, hi)
              # 进行分解，然后顺便recursion下去
            # if the two halves agree on the majority element, return it.
            if left == right:
                return left
              # 如左右的数组相同 则return特征
            # otherwise, count each element and return the "winner".
            left_count = sum(1 for i in range(lo, hi+1) if nums[i] == left)
            right_count = sum(1 for i in range(lo, hi+1) if nums[i] == right)
              # 不相同的情况下，找左右的数组特征
            return left if left_count > right_count else right
              # 返回特征
        return majority_element_rec(0, len(nums)-1)
```