思考题 1

孟妍廷 2015202009

2017 年 9 月 27 日

分析: 对于多数问题, 给定一个大小为 n 的数组, 则 n 只有为偶数和为基数两种可能, 分情况讨论:
1. 偶数: 当 n 为偶数时, 若一个数出现的次数要超过 $\frac{n}{2}$, 该数起码有两个是相邻的, 示例如下:

121314 不满足

113141 满足

2. 奇数: 当 n 为奇数时, 若一个数出现的次数要超过 $\frac{n}{2}$, 该数在数组中出现的最离散的情况如下:

1213141 每两个1之间有一个不为1的数间隔

根据以上特性, 可以设置一个变量 count(初值为 1), 假定当前数组存在一个数出现次数超过 $\frac{n}{2}$, 首先假设数组第一位的数就是这个数, 向后遍历, 当这个数再次出现时 count++, 当这个数没有出现时 count − , 当 count 减到零时, 假定数组下一位是这个数.
若到最后一位前都未找到，则假定数组最后一位就是这个数, 进行检查.
该算法的时间复杂度为 O(n). 伪码如下:

```
int majority(int num[ ], int begin, int end) {
 if(begin == end && check(end) == 1)
  return end;
 int count = 1;
 int i = begin;
 while(count > 0 && i < end) {
  if(num[++i] == num[begin])
   count++;
  else
   count--;
 }
 if(count > 0)
  return begin;
 else
  majority(num[n], ++begin, end);
}
```

因此，首先以 begin 为 0，end 为 n 开始调用 majority 函数即可，空间复杂度为 O(1).