众数问题

本文探讨一些和众数有关的问题。

摩尔投票法

摩尔投票法应该是最经典的一个众数有关的算法。它可以在线性时间内利用常数的时间复杂度解决这样一个问题：求出某个长为 n 的序列中的出现次数 $> \frac{n}{2}$ 的数（方便起见称为众数）。

我们考虑相互抵消这一策略。即序列中的不同的数相互抵消，抵消掉的一对数消失，那么只要这个众数存在，无论是众数和非众数、还是非众数和非众数抵消，最后必然只会是这个众数剩余。

按照这种想法可以维护当前等待抵消的数 x 以及其需要被抵消的次数 cnt。初始 $cnt = 0$。扫描整个序列，对于序列当前的数 y，讨论：

1. $cnt = 0$，则 $x := y, cnt := 1$。
2. $cnt > 0$，则：
    1. $x = y$，则 cnt 自增。
    2. $x \neq y$，则 cnt 自减。

注意，这个算法能够正确地找出众数的前提是众数真的存在。因此对于最后剩下的数，还需要回到原序列中检查其的出现次数是否 $> \frac{n}{2}$。

摩尔投票法扩展

如果要求出某个长为 n 的序列中出现次数 $> \frac{n}{3}$ 的数（称为众数），该怎么做?

还是和前面一样，考虑相互抵消的策略。只不过这次由于可能有两个这样的众数，所以要考虑的不是两两抵消，而是三个三个抵消。只有三个三个抵消剩下来的才**可能**会是众数。因此，维护的等待被抵消的数应该为两个，对应的计数器也要维护两个。

确定了维护方法和维护对象后就可以按照和之前类似的方式更新、维护。时间复杂度仍是线性，空间复杂度仍是常数级别。同样要注意：对于最后剩下来的数，必须回到原序列中检查它是不是真的满足出现次数 $> \frac{n}{3}$。

按照这种方法可以扩展到任意的求出现次数 $> \frac{n}{k}$ 的数的情况。

区间众数查询

如果现在问题变成：给定一个序列，有若干的询问，询问形如要求查询 $[l, r]$ 这段区间上的众数，还可能有修改操作。应该怎么处理?

问题 1

问题：不带修改，查询的众数限定为出现次数 $> \frac{L}{2}$ 的数，L 为查询区间长度。要求在线。（Leetcode 1157)

这个问题可以考虑分块解决。考虑对询问分块，假设对长度 $\leq S$ 的询问可以暴力解决。对于长度 $> S$ 的询问，如果答案存在那么其出现次数应当 $> \frac{S}{2}$。整个序列中最多会有 $\frac{2n}{S}$ 个出现次数满足这个条件的数，因此可以预先统计这些数出现次数的前缀和，然后查询时 $O(\frac{2n}{S})$ 扫一遍，检查是否有答案。

由均值不等式，取 $S=\sqrt{2n}$ 可以让这两部分的时间复杂度均取 $O(\sqrt{n})$。因此算上预处理，总的时间复杂度为 $O((n+q)\sqrt{n})$。

（以下为 Leetcode 1157 的代码。由于该题与这里描述略有不同，因此仅作参考）

```
class MajorityChecker {
    int n, bsize;
    int sum[210][20005], lis[210], tot;
    int cnt[20005];
    vector<int>& a;
public:
    MajorityChecker(vector<int>& arr) : a(arr){
        memset(sum, 0, sizeof(sum)) ;
        n = arr.size();
        bsize = static_cast<int>(floor(sqrt(n * 2) + 0.5));
        memset(cnt, 0, sizeof(cnt));
        for (int x: arr)
            ++cnt[x];
        tot = 0;
        for (int i = 1; i <= 20000; ++i){
            if (cnt[i] >= 2 * n / bsize){
                lis[++tot] = i;
                for (int j = 1; j <= n; ++j)
                    sum[tot][j] = sum[tot][j - 1] + (arr[j - 1] == i ? 1: 0);
            }
        }
    }

    int query(int left, int right, int threshold) {
        if (right - left + 1 <= bsize){
            int x = 0, ccnt = 0;
            for (int i = left; i <= right; ++i) {
                if (!ccnt) ccnt = 1, x = a[i];
                else ccnt = (x == a[i] ? ccnt + 1: ccnt - 1);
            }
            ccnt = 0;
            for (int i = left; i <= right; ++i)
                if (x == a[i]) ++ccnt;
            return (ccnt >= threshold ? x: -1);
        }else {
            for (int i = 1; i <= tot; ++i)  {
                if (sum[i][right + 1] - sum[i][left] >= threshold)
                    return lis[i];
            }
            return -1;
        }
    }
};
```

另一种方法是将摩尔投票法搬到区间上。由摩尔投票法的流程可以推知，如果将整个序列分割成若干个部分分别做摩尔投票法，然后将每一个部分的结果再做摩尔投票法，那么如果众数真的存在，最后的结果还是原序列的众数。这符合线段树区间可加的条件，因此可以用线段树进行维护。维护的对象即为当前众数和计数器。

当然，仍有必要对结果检查其是否真的出现次数 $>\frac{L}{2}$。这个可以离散化后用 vector 维护。这样总的时间复杂度是 $O(n+q\log n)$。

问题 2

（等待补充）

参考资料

1. A Fast Majority Vote Algorithm。这篇论文正是摩尔投票法名称的由来。
2. Leetcode 1157 题解
3. 《区间众数解题报告》By 陈立杰