LeetCode 精选 TOP 面试题（6）

344. 反转字符串

思路

(双指针) $O(n)$

- 1、定义两个指针 `i` 和 `j`，初始化 `i = 0`，`j = n - 1`。
- 2、`i` 从前往后，`j` 从后往前，只要 `i < j`，我们交换 `s[i]` 和 `s[j]`。

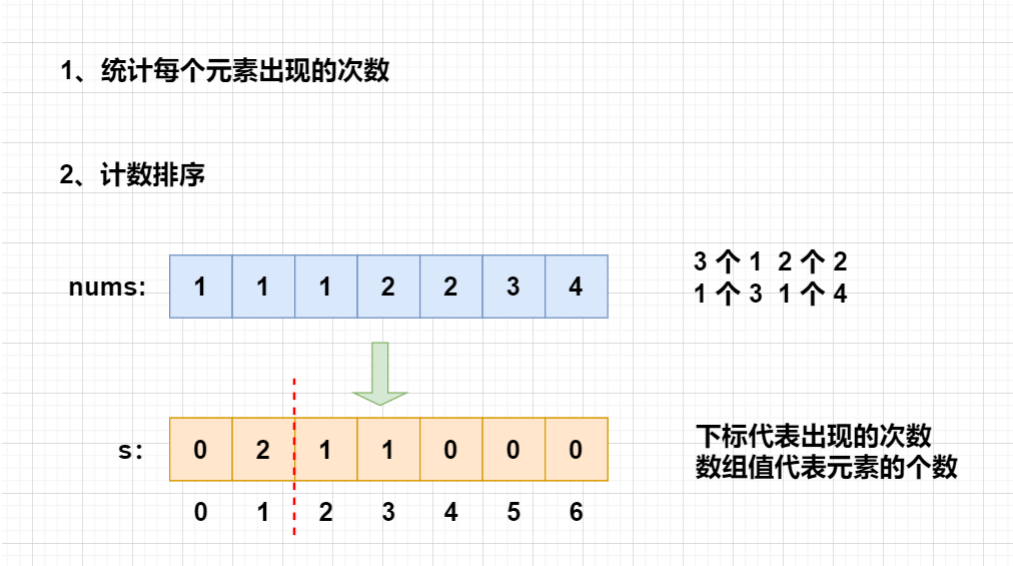
c++代码

```
class Solution {
public:
    void reverseString(vector<char>& s) {
        int n = s.size();
        for(int i = 0, j = n - 1; i < j; i++, j--){
            swap(s[i], s[j]);
        }
    }
};
```

347. 前 K 个高频元素

思路

(计数排序) $O(n)$



我们可以先统计每个数字出现了多少次，在统计一下出现次数为 `t` 次的元素各有多少个，然后利用计数排序的思想判断一下出现次数前 `k` 多的数字最少出现多少次，求出这个下界 `i`，最后再遍历一次哈希表，将所有出现次数大于等于这个下界的元素加入答案。

具体过程：

- 1、先统计每个元素出现次数。
- 2、用 `s` 数组，`s[i]` 表示出现了 `i` 次的元素有 `s[i]` 个。
- 3、根据 `k` 在 `s` 数组中找到一个分界线 `i`，使得前 `k` 个高频元素的出现次数都 `>i` 次。

c++代码

```cpp
class Solution {
public:
    vector<int> topKFrequent(vector<int>& nums, int k) {
        int n = nums.size();
        unordered_map<int, int> cnt;
        for(int x : nums) cnt[x]++;
        vector<int>s(n + 1);
        for(auto p : cnt) s[p.second]++;
        int i = n, t = 0;
        while(t < k) t += s[i--];
        vector<int> res;
        for(auto p : cnt){
            if(p.second > i)
                res.push_back(p.first);
        }
        return res;
    }
};
```

350. 两个数组的交集 II

思路

(哈希) $O(n+m)$

- 1、定义一个`hash`表，遍历`nums1`数组，记录每个元素出现的次数。
- 2、遍历`nums2`数组，如果`hash[x]`存在，则将`x`添加到结果数组`res`中，同时`hash[x]--`。
- 3、最后返回`res`。

时间复杂度分析： `nums1`数组和`nums2`数组分别遍历一次，因此时间复杂度为$O(n+m)$。

c++代码

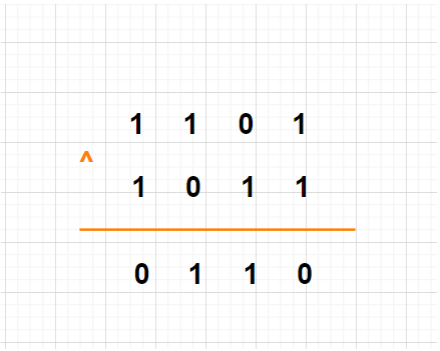
```cpp
class Solution {
public:
    vector<int> intersect(vector<int>& nums1, vector<int>& nums2) {
        unordered_map<int, int> hash;
        for(int x : nums1) hash[x]++;
        vector<int> res;
        for(int x : nums2){
            if(hash[x]){
                res.push_back(x);
                hash[x]--;
            }
        }
        return res;
    }
};
```

371. 两整数之和

思路

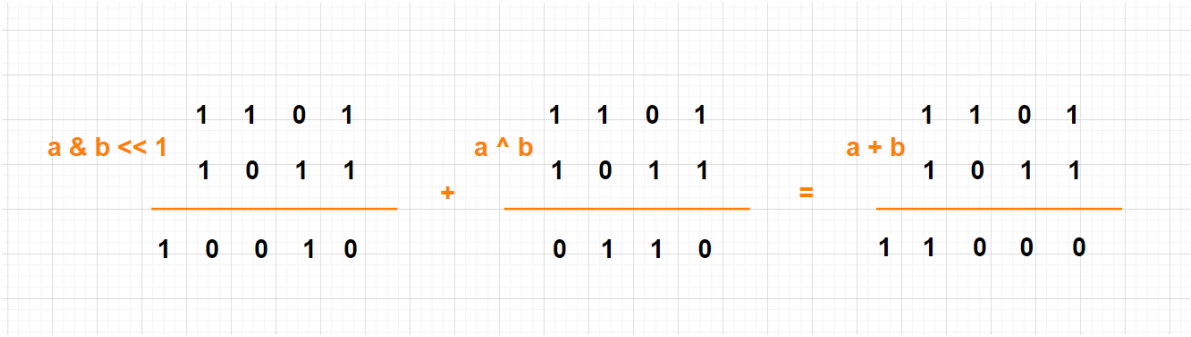
(位运算) $O(1)$

异或运算相当于不进位的加法，比如：`1101` ^ `1011` = `0110`。



不进位的加法再加上进位就等于两个数相加，而进位和`&`运算一样，我们只需要把`a & b`的结果左移一位即可得到进位，因此

`a + b == ((a & b) << 1) + (a ^ b)`。



但是我们发现计算`(a ^ b) + ((a & b) << 1)`时，又用到了加法，但其实本质上还是两个数相加。因此我们递归或者循环计算，直到进位`(a & b) << 1`为`0`为止。

具体过程如下：

- 1、我们计算出不进位`sum = a ^ b`，进位`carry = (unsigned)(a & b) << 1`。
- 2、递归调用`getSum(carry, sum)`。
- 3、直到`a == 0`时，我们返回`b`，即为答案。

实现细节：

在c++中，计算`(a & b) << 1`，可能发生溢出，因此我们做一个类型转换，即计算`(unsigned)(a & b) << 1`。

时间复杂度分析： $O(1)$。

c++代码

```cpp
class Solution {
public:
    int getSum(int a, int b) {
        if(!a) return b;
        int sum = a ^ b, carry = (unsigned)(a & b) << 1;
        return getSum(carry, sum);
    }
};
```

Java代码

```java
class Solution {
    public int getSum(int a, int b) {
       if(a == 0) return b;
       int sum = a ^ b , carry = (a & b) << 1;
       return getSum(carry, sum);
    }
}
```

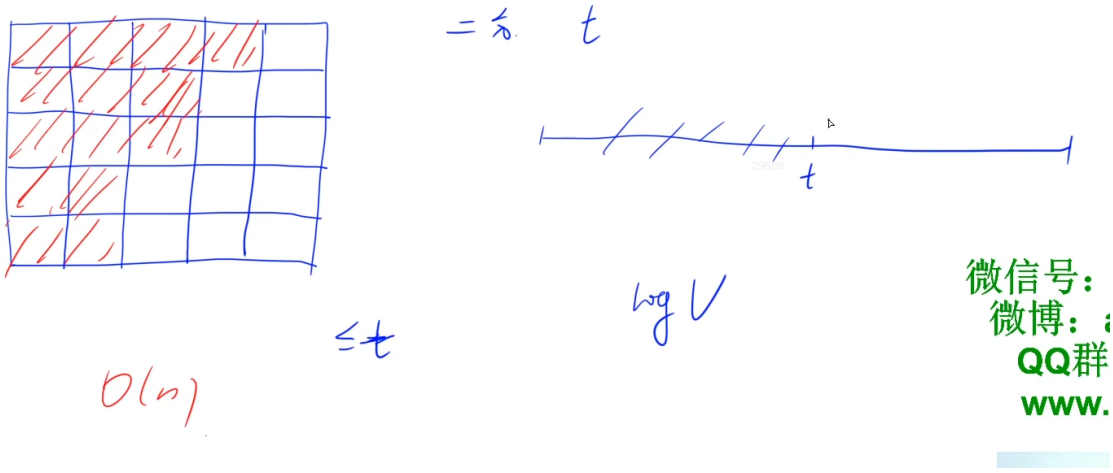
378. 有序矩阵中第 K 小的元素

思路

(值域二分) $nlog(V)$

数组的最小值是左上角的`matrix[0][0]`，最大值是右下角的`matrix[n-1][n-1]`，那么第`k`小的数一定在这个区间内。

二分的区间为`matrix[0][0] ~ matrix[n-1][n-1]`，假设我们二分出来的答案为`target`，那么遍历整个数组，统计`<= target`的个数`cnt`，如果`cnt < k`个，那么说明第`k`小的数比`target`大。如果`cnt >= k`，就说明第`k`小的数`<= target`。



根据矩阵性质，每一行`<= target`的个数一定是**非递增**的。

c++代码

```cpp
class Solution {
public:
    int kthSmallest(vector<vector<int>>& matrix, int k) {
        int n = matrix.size();
        int l = matrix[0][0], r = matrix[n - 1][n - 1];
        while(l < r){
```

```
            int mid = l + r  >> 1;
            int j = n - 1, cnt = 0;
            for(int i = 0; i < n; i++){
                while(j >= 0 && matrix[i][j] > mid) j--;
                cnt += j + 1;
            }
            if(cnt >= k) r = mid;
            else l = mid + 1;
        }
        return r;
    }
};
```

387. 字符串中的第一个唯一字符

思路

(哈希) $O(n)$

- 1、定义一个`hash`表，记录每个字符出现的次数。
- 2、遍历`s`数组，如果`hash[s[i]] == 1`，我们返回`i`。

c++代码

```
class Solution {
public:
    int firstUniqChar(string s) {
        unordered_map<char, int>hash;
        for(char c : s) hash[c]++;
        for(int i = 0; i < s.size(); i++){
            if(hash[s[i]] == 1) return i;
        }
        return -1;
    }
};
```

412. Fizz Buzz

思路

(模拟) $O(n)$

c++代码

```
class Solution {
public:
    vector<string> fizzBuzz(int n) {
        vector<string> res;
        for(int i = 1; i <= n; i++){
            if(i % 3 == 0 && i % 5 == 0) res.push_back("FizzBuzz");
            else if(i % 3 == 0) res.push_back("Fizz");
            else if(i % 5 == 0) res.push_back("Buzz");
            else res.push_back(to_string(i));
        }
        return res;
    }
};
```