**1. Find First Unique Object in a list of objects. Objects can be any type.**

Go though the list with unordered\_map, key is objects, value is the frequency.

Go though the list again, find the first one whose frequency == 1;

**Folow up: solve in one pass**

Doublelinkedlist + unordered\_map, the head of the list is the result.

**2. int array里重复出现最多**

use a unordered\_map, and a value to track the most frequent one.

**Follow up: without a map?**

Sort and Count

**3. 给一个string，判断是有duplicate character**

unordered\_set

**Follow up:** 不许用任何data structure

int[256]

**4. Given a int array，每个数字的范围都是1到10000**

(1)找出最小的不存在的数 array={3,1,4} return 2

(2)判断有没有duplication

**1.unodered\_set**

**2.array bool[n] (10000 byte)**

**3.都是positive number，可以取negative来标示该index的数是不是已经出现过**

**Follow up: Reduce Memory**

**4.bitset** (1-10billon 的input)

the size of the bitset is **10 billion bit** = 1.25 billion byte = 1.25 GB

1GB = 1 billion byte

**5. bloom filter** (save space, but there may be some error, "possibly in set" or "definitely not in set")

**5.Use an unordered\_map / prefix tree**

(1)给一个句子, 输出每个单词的频率

(2)给你一个novel, 输出单词出现的次数

(3)给一个大的文件, 里面有不少的ip address, 写个程序统计一下不同ip出现的次数

**6. Single Number数组中1个出现奇数次的元素，假定其他元素都出现2次。**

1.unordered\_map (如果有范围0-N，则用array)

2.xor

3.sort + binary search O(1) space

**Follow up: 可能有多个奇数次元素**

unordered\_map

**7. Given two string, find the common chars**

**一个BST和一个string，怎么找到common的element**

1. unordered\_set

2. int[256] (ASCII code)

结果用unordered\_set 表示

**8. Given two int/string array, find the common elements (Intersection of two arrays)**

**array1[1，2，2，3，4] array2[2，2] output: [2]**

unordered\_set<int>

go through array1, insert everyone

go through array2, if in table, res.push\_back(curr), **table.earse()(防止array2 has duplicate);**

结果用vector<int> 表示

**Follow up 1: output duplicate are considered output[2,2]**

1.unordered\_map<int, vector<int>> vector<int> count for array 1 and array2

2.unordered\_map<int, int> table

go though the array1, count for everyone, do++;

go though the array2, if table[arrary[i]] > 0, res.push\_back(arrary[i]), do--;

**Follow up 2: space O(1) or have been sorted (Intersection of two sorted arrays)**

Sort and count with pointer, each time moves the smaller one.

Quick sort because it uses constant space

**9. Given n vector<sting>, find common elements**

**(1)找n个vector<string> 的Common string**

**a,a**

**a,**

**输出一个a**

**a,a,a**

**a,a,a**

**输出3个a**

**(2)Write a function that takes in email lists and results in a new email list that contains only the email addresses that are in all lists**

List1(kindle) : foo@amazon.com

List2(aws) : foo@amazon.com, jason@amazon.com

List3(videogames): foo@amazon.com, jason@amazon.com, annabel@amazon.com

Unordered\_map<string, int> count the frequency if there are not duplicate in each list.

**If there are duplicate in each list.**

Unordered\_map<string, int> if( i == table[array[i][k]]) table[array[i][k]]++;

Unordered\_map<string, vector<bool>> vector<bool> to mark everyone

**11. anagram**

0. 判断第一个string是否能够construct第二个string

1. 判断一个string是不是另外一个string的permutation

2. LeetCode Group Anagrams（我有一本字典 怎么找到所有anagram然后输出）

3. LeetCode valid Anagrams

**12.Set**

1.给两个set，返回一个SET包含第一个SET里第二个没有的 和第二个SET里第一个没有的

Time O(m+n) 分别过一遍两个set

2.给定三个集合，找出三个集合的重合

Time O(min(m,n,k)) 看每个元素在不在其他两个.