哈希表

什么时候想到用哈希表?

需要判断一个元素是否出现在集合里/判断一个元素是否出现过，以及涉及到类别/字母种类统计时，可以考虑哈希表

字符串的写法总结

访问字符串中某个字母：如s.charAt(j)；

把字符串转换成字符数组（排序需要用）：如char [ ] array = str.toCharArray( );

对字符数组排序：Arrays.sort(array)；

哈希表的写法总结（Java版，持续更新）

1、数组型哈希表：

（1）声明：如int [ ] record = new int[26]；

（2）把字符在哈希表中的键投射为整数：如record[s.charAt(j) - 'a']++;

（3）判断两个哈希表是否相等：如Arrays.equals(sCount, pCount)；

（4）确定数组的输出范围：如Arrays.copyOfRange(intersection, 0, index);

（5）把若干个元素组合起来组成数组：如quadruplets.add(Arrays.asList(nums[i], nums[j], nums[left], nums[right]));

2、map型哈希表：

（1）声明：如Map<String, List<String>> map = new HashMap <String, List<String>>( )；

（2）从哈希表中获取与键对应的列表，如果不存在则创建一个新的列表：

如List<String> list = map.getOrDefault(key, new ArrayList<String>( ))；

（3）往数组/列表中添加元素：如list.add(str)；ans.add(i);

（4）将哈希表更新的键值对放回map中：如map.put(key, list)；

（5）访问map中的所有值：如List<List<String>> arrays = new ArrayList<List<String>>(map.values())；

（6）移除map中某个键：如map.remove(num);

（7）检查map中有无某个键：如map.containsKey(target - nums[i])；

3、set型哈希表：

（1）声明：如Set<Integer> set = new HashSet<Integer>();

（2）往set中加入元素：如set.add(i)；

（3）检查set中有无某值：如set.contains(i)；

（4）set哈希表大小：如resSet.size()；

（5）set型哈希表的元素以数组形式输出：如resSet.stream().mapToInt(x -> x).toArray();

字母异位词题型

字母异位词要想到哈希表！滑动窗口也应想到！

如果涉及异位词的统计：

通常用map型哈希表，把排序的字符串array作为键，实际字符串str加入对应键的值中

如果涉及判断、比较型的情况（如判断两个字符串是否为异位词，判断其中一个字符串能否用另一字符串表示等，且只涉及小写字母）：

通常用数组型哈希表，参考字符串的字符数在record先加，目标字符串的字符数在record再减，再考察record每个元素的值，视题目具体情况做出结果判断

此外有些需要用到滑动窗口，此时通常需要两个数组型哈希表储存和比较信息

题目：

242. 有效的字母异位词（简单）

383. 赎金信（简单）

49. 字母异位词分组（中等）

438. 找到字符串中所有字母异位词（中等）

两个数组的交集

什么时候想到set型哈希表：遇到**不重复**统计数组元素要考虑set型哈希表；且当数组元素不多，但值的范围巨大时，用数组型哈希表统计不方便，此时要用set型哈希表

什么时候想到map型哈希表：如果要统计数组元素的出现次数，要用map型哈希表（和统计字符串字母种类数量区分），若用数组型哈希表，无法确定该表的大小，造

成空间浪费

题目细节详情看leetcode中的标注

题目：

349. 两个数组的交集（简单）

350. 两个数组的交集II（简单）

快乐数

两个关键点：如何表示一个数各位数字的平方之和；如何退出死循环

由于退出死循环的判断需要保证哈希表无重复元素，因此自然想到用set型哈希表

题目细节详情看leetcode中的标注

题目：

202. 快乐数（简单）

数组“x数之和”题型

两数之和：应考虑map型哈希表（需要储存元素值作为key，元素下标作为value），当需要统计元素下标时应当想到map型哈希表（set型和数组型都不合适）。

四数相加II：是两数之和的变体，重点在a+b一起考虑，c+d一起考虑（两个for-for循环）。由于不去重，需要精确得到a+b的出现次数（map型哈希表），再让c+d的值（实则为相反数）去哈希表中找，进而统计结果。

三数之和&四数之和：这两个题用哈希表会更加耗时，且更复杂容易出错，更适合的方法是双指针。

首先需要对数组排序。三数之和是固定a，双指针赋给b和c；四数之和则需for-for先固定a和b，再将双指针赋给c和d。这两个题需要对元素去重，且可以加上剪枝操作节省循环时间。

（两数之和和四数相加II就不适合用双指针，因为两数之和需要精确统计下标位置，一旦数组排序后顺序就乱了；四数相加II涉及多个数组，用双指针不方便）

题目细节详情看leetcode中的标注

题目：

1. 两数之和（简单）

454. 四数相加（中等）

15. 三数之和（中等）

18. 四数之和（中等）