基础算法和数据结构高频题 II

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
知乎专栏：http://zhuanlan.zhihu.com/jiuzhang
微博: http://www.weibo.com/ninechapter
官网: www.jiuzhang.com

知识点回顾

- 给一个数组a，s是它的前缀和数组
  - a[i]+a[i+1]+ …..+ a[j] =?

- (sum[id] - sum[id - size]) / size; 改成滚动怎么改？选择题（a or b）
  - a. (sum[mod(id)] - sum[mod(id) - size]) / size;
  - b. (sum[mod(id)] - sum[mod(id – size)]) / size;

知识点回顾

- Read N Characters Given Read4 II - Call multiple times 这题我们用的buffer是什么数据结构？

- 如果我们定义 ’ : ’ 为转义符号，’ :+ ’表示字符串连接符，那么怎样表示’ : ’本身？

概要

- Hash 字符/字符串统计类问题（5题）
- 综合应用问题 （2题）
- 快速点题 （5题）

Hash 字符/字符串统计类问题

Valid Anagram

http://www.lintcode.com/problem/valid-anagram/

http://www.jiuzhang.com/solution/valid-anagram/

Valid Anagram

思路：

- Anagrams的充要条件是？
  - 字符出现的次数一样

- Hash：
  - cntS数组统计S中字符出现的次数
  - cntT数组统计T中字符出现的次数
  - 判断cntS数组是否等于cntT数组

时间复杂度：O(n)

Valid Anagram

- Company Tags：Amazon

考点：

- 简单Hash的应用

Valid Anagram

能力维度：

3. 基础数据结构/算法

Find All Anagrams in a String

http://www.lintcode.com/problem/find-all-anagrams-in-a-string/

http://www.jiuzhang.com/solution/find-all-anagrams-in-a-string/

Find All Anagrams in a String

- **Input:**
  - s: "cbaebabacd" p: “aabc"
- **Output:**
  - [5]

Find All Anagrams in a String

思路：

- 基本的想法：
  - 假设p串的长度为l , s串长度为n
  - 那么就枚举出s中所有长度为l 的子串，并用hash统计它们元素出现的个数

- 基本想法的时间复杂度：
  - n个子串
    - 每次统计子串中元素出现的个数O(l)
    - 每次和p对比元素出现次数是否一样 O(256)
  - 总体O(n * (l + 256)) = O(nl)

Find All Anagrams in a String

- 可以更快吗？
  - 想想相邻的两个子串的差别？

s: “cbaebabacd” p: “aabc”

idx	0	1	2	3	4	5	6	7	8	9
s	c	b	a	e	b	a	b	a	c	d

1	1	1	0	1
a	b	c	d	e

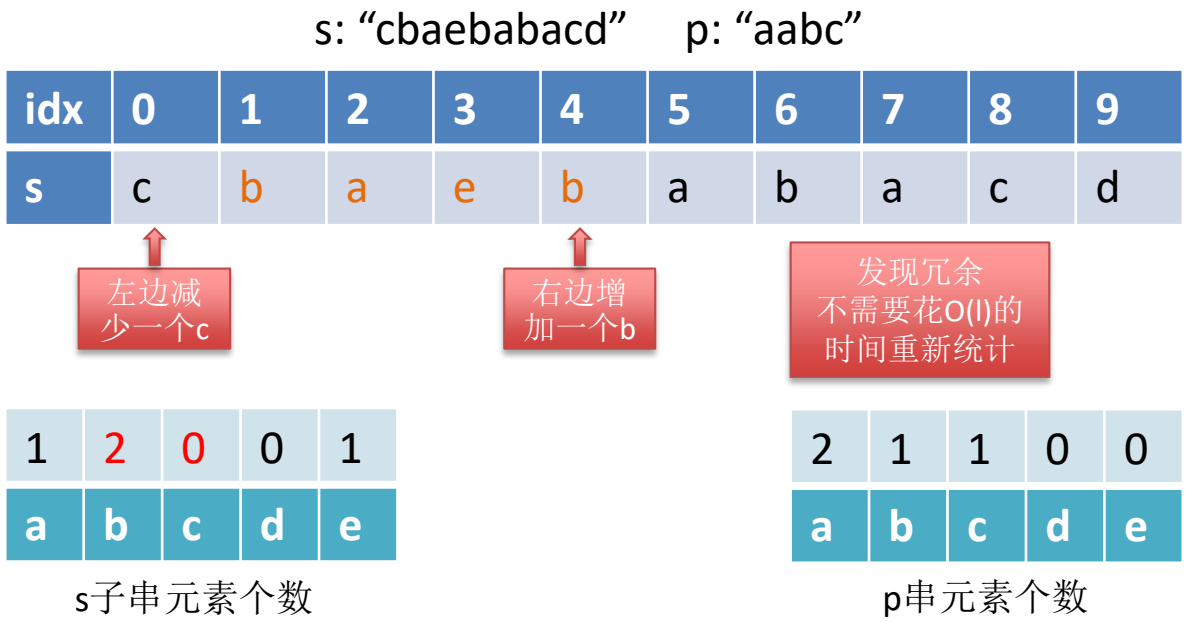
s子串元素个数

2	1	1	0	0
a	b	c	d	e

p串元素个数

Find All Anagrams in a String

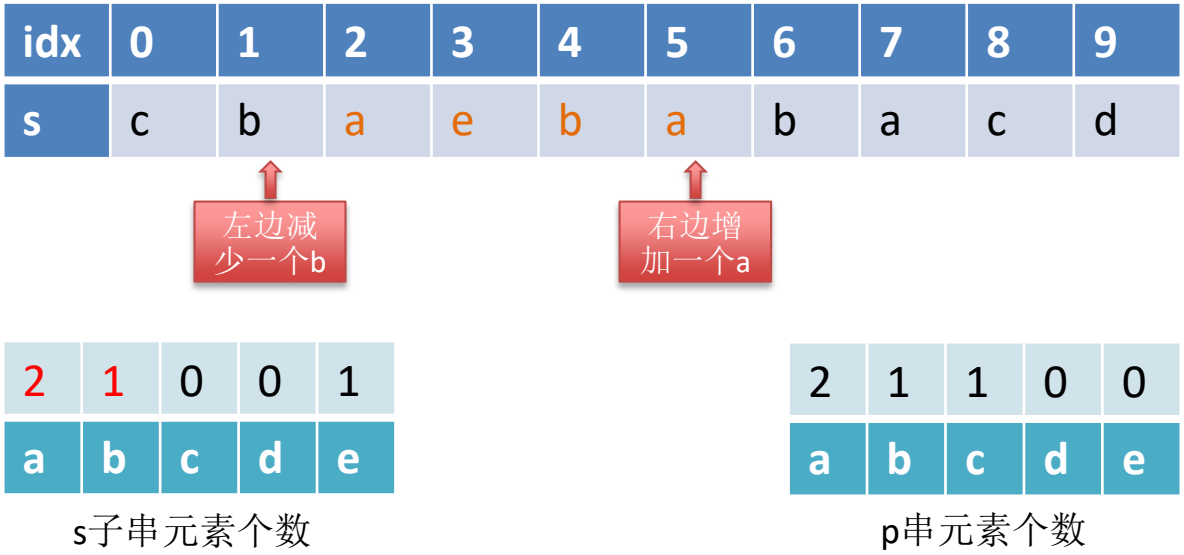
- 可以更快吗？
  - 想想相邻的两个子串的差别？



Find All Anagrams in a String

- 可以更快吗？
  - 想想相邻的两个子串的差别？

s: “cbaebabacd” p: “aabc”



Find All Anagrams in a String

- 相当于一个长度为l 的sliding window 从左往右扫一遍
  - 每次都是右边增加一个，左边减少一个　O(l) → O(1)

s: “cbaebabacd”　p: “aabc”

idx	0	1	2	3	4	5	6	7	8	9
s	c	b	a	e	b	a	b	a	c	d

左边减少一个b

右边增加一个a

2	1	0	0	1
a	b	c	d	e

s子串元素个数

2	1	1	0	0
a	b	c	d	e

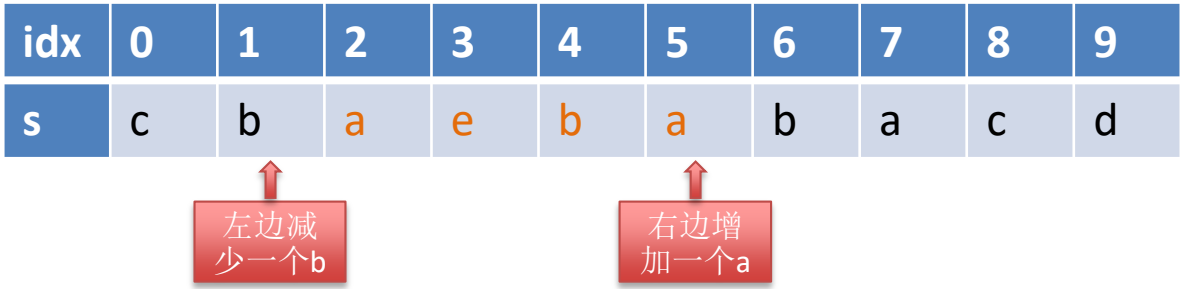
p串元素个数

时间复杂度：
O(n * 256) = O(n)

Find All Anagrams in a String

- 为什么有256这个常数项？可以优化吗？（扩展）

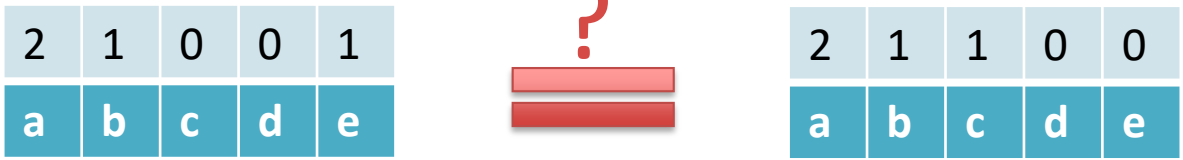
s: “cbaebabacd” p: “aabc”



时间复杂度：
O(n * 256) = O(n)

Find All Anagrams in a String

- 为什么有256这个常数项？可以优化吗？（扩展）
  - 怎样快速的判断下面的两个数组是否相等？ (cntS == cntP ?)



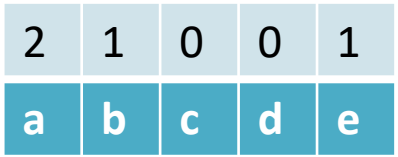
s子串元素个数
cntS

p串元素个数
cntP

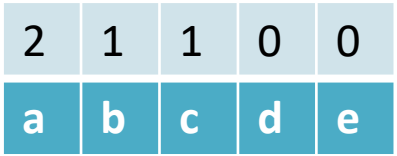
时间复杂度：
O(n * 256) = O(n)

Find All Anagrams in a String

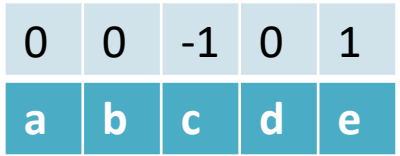
- 定义det = cntS – cntP
  - 如果det全为0 → cntS == cntP



s子串元素个数
cntS



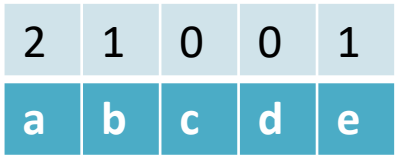
p串元素个数
cntP



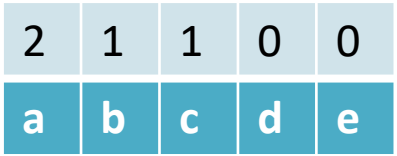
det

Find All Anagrams in a String

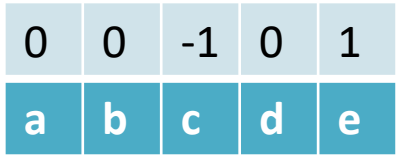
- 怎样判断det全为0？
  - 所有的元素绝对值求和 absSum == 0 → det全为0



s子串元素个数
cntS



p串元素个数
cntP

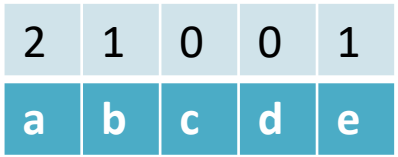


det

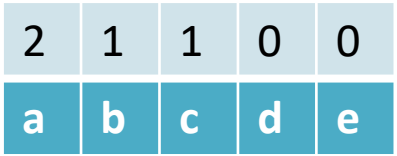
absSum = 0+0+abs(-1)+0+1 = 2

Find All Anagrams in a String

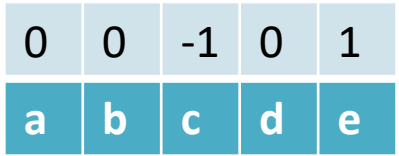
- absSum 怎么更新？
  - 在sliding window向右移动时，减去原有的，加上改变的　O(1)



s子串元素个数
cntS



p串元素个数
cntP

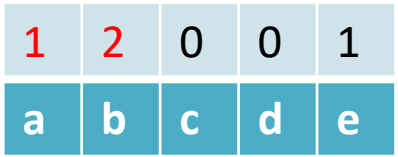


det

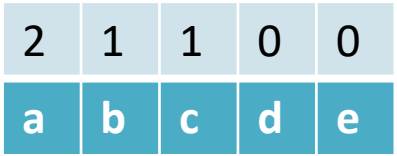
absSum = 0+0+abs(-1)+0+1 = 2

Find All Anagrams in a String

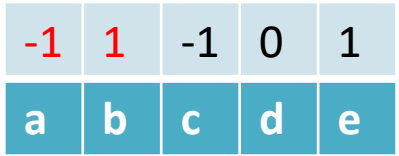
- absSum 怎么更新？
  - 在sliding window向右移动时，减去原有的，加上改变的　O(1)



s子串元素个数
cntS



p串元素个数
cntP



det

absSum = absSum-0-0+abs(-1)+1=4

总体时间复杂度：O(n)

Find All Anagrams in a String

- Company Tags：Amazon

考点：

- Sliding window + hash

相关题目:

- Sliding window median
- Sliding window maximum

Find All Anagrams in a String

能力维度：

1. 理解问题
3. 基础数据结构/算法
4. 逻辑思维/算法优化能力

Word Abbreviation

http://www.lintcode.com/problem/word-abbreviation/

http://www.jiuzhang.com/solution/word-abbreviation/

Word Abbreviation

字符串收缩操作：

apple → a 3 e

apple → a2 le

apple → 4 e

Word Abbreviation

like	god	internal	me	internet	interval	intension	face	intrusion
l2 e	god	i6l	me	i6t	i6l	i7n	f2e	i7n
		in5l			in5l	in6n		in6n
		int4l			int4l	int5n		int5n
		inte3l			inte3l	inte4n		intr4n
		inter2l			inter2l			
		internal			interval			

l2e	god	internal	me	i6t	interval	inte4n	f2e	intr4n

Word Abbreviation

思路：

- 直接模拟

- 求出abbr，有重复就增加prefix继续求abbr

- 怎么判断重复？
  - Hash桶计数

Word Abbreviation

- Company Tags：Google

能力维度:

1. 理解问题
2. 代码基础功力
3. 基础数据结构/算法
4. debug能力

Word Abbreviation Set

http://www.lintcode.com/zh-cn/problem/word-abbreviation-set/

http://www.jiuzhang.com/solutions/word-abbreviation-set/

Word Abbreviation Set

- Given dictionary = ["deer", "door", "cake", "card"]

- isUnique("dear") -> false
- isUnique("cart") -> true
- isUnique("cane") -> false
- isUnique("make") -> true

Word Abbreviation Set

规则解读：

- 假如apple 没在字典中出现过，a3e这个缩写也没出现过
  unique　　（要查找的词在词典中没有出现过）

- 假如 cake 在字典中出现了2次 并且缩写中所有的c2e都是对应cake
  unique　　（要查找的词在词典中出现过，但缩写只对应要查找的词）

- abbr dictionary = ["d2r", "d2r", "c2e", "c2e", "c2d"]　　0　　2
- dictionary = ["deer", "door", "cake", "cake", "card"]　　0　　2

Word Abbreviation Set

思路：

- 两种情况合并在一起，总结起来的规律就是：
  - 单词在字典中出现次数等于对应缩写在字典中出现次数 -> unique
  - 单词在字典中出现次数不等于对应缩写在字典中出现次数 -> not unique

- 用数据结构什么记录单词和缩写出现的次数
  - Hash

Word Abbreviation Set

- Company Tags: Google

考点：

- 理解题目的规则
- Hash的应用

Word Abbreviation Set

能力维度：

1. 理解问题
3. 基础数据结构/算法
5. 细节处理（corner case）

Longest Consecutive Sequence

http://www.lintcode.com/zh-cn/problem/longest-consecutive-sequence/

http://www.jiuzhang.com/solutions/longest-consecutive-sequence/

Longest Consecutive Sequence

思路：

- 不考虑o(n)的时间复杂度要求，大家可以想到什么样的方法？
- 有什么可以改进的？

Longest Consecutive Sequence

- 如果有o(n) 的时间复杂度要求该怎么做呢？
  - 我们一个个看　如果100在答案中 最长能有多长？最长是1 因为101 99 不在数组中（怎么确定？ hash），　如果4 在答案中 最长能有多长？

100	4	200	1	3	2

Longest Consecutive Sequence

- 所以一种简单的方法是对每个数字，向左向右搜一下，看最长能有多长
- 还有一个发现就是，如果4 向左向右搜到了1 2 3 那么1 2 3这三个数字就不用向左向右搜了（发现冗余）。

100	4	200	1	3	2

Longest Consecutive Sequence

- 时间复杂度怎么算？
  - 每个元素只会被访问一遍，所以时间复杂度是O(n)

100	4	200	1	3	2

Longest Consecutive Sequence

- Company Tags：Google Facebook

考点：

- 是否可以跳出排序后扫描的思维定式，以每个元素作为突破点

Longest Consecutive Sequence

能力维度：

3. 基础数据结构/算法
4. 逻辑思维/算法优化能力
6. 算法分析（时间/空间复杂度）

Longest Consecutive Sequence

◆ 小技巧总结：

- Hash可以在O(1)的时间内确定一个元素是否存在，利用这个可以降低时间复杂度
- 计算时间复杂度的方法，“每个元素只会被访问一遍”这句话所代表的方法很常用

综合应用问题

Load Balancer

http://www.lintcode.com/zh-cn/problem/load-balancer/

http://www.jiuzhang.com/solutions/load-balancer/

Load Balancer

思路：

- 要在o(1)的时间内插入删除，只能hash。那hash可以getRandom吗？
  - 不太好做

- 什么数据结构比较好getRandom？
  - 数组

- 考虑hash与数组结合起来用，hash插入一个，数组也插入一个。那么问题来了，数组删除元素怎么办？
  - 与最后插入的一个元素交换

- 那怎么o(1)时间在数组中找到要删除元素（要交换）的位置？
  - 用hash将元素的位置记下来

Load Balancer

算法：

- 插入：
  - 数组末尾加入这个元素
  - Hash这个元素存下数组中的下标

- 删除：
  - 通过hash找到这个元素在数组中的位置
  - 数数组中这个元素和数组的末尾元素交换，交换后删除
  - Hash中删除这个元素，更新数组原末尾元素现在在数组中的位置

- Pick：
  - 数组中random一个返回

Load Balancer

- Company Tags: Google Amazon Facebook

考点：

- 两种数据结构的综合应用

能力维度：

3. 基础数据结构/算法
4. 逻辑思维/算法优化能力
6. 算法分析（时间/空间复杂度）

Find the Celebrity

http://www.lintcode.com/problem/find-the-celebrity/

http://www.jiuzhang.com/solution/find-the-celebrity/

Find the Celebrity

- 名人就是所有人都认识他，他只认识他自己的那个（名人检验）

思路一：

- 一个简单的做法，对每个做一次名人检验，看是不是所有人都认识他，但他不认识所有人
- 时间复杂度$O(n^2)$
- 怎样降低时间复杂度？　冗余在哪里？

Find the Celebrity

思路二：

- 我们询问一次的时候只利用答案为true的情况，如果为false呢？
- 一次询问knows(a, b): true a认识 b a一定不是名人
- false a不认识b b一定不是名人

- 所以一次询问就可以排除一个人，n-1询问后剩下一个人，再对这个做个名人检验就能确定他是否为名人

- 所以实现上就是从左到右扫一遍，每次都是保留下的人和新的人做一次询问，最开始保留的人设为第1个人

Find the Celebrity

- Company Tags：LinkedIn Facebook

能力维度：

4. 逻辑思维/算法优化能力
6. 算法分析（时间/空间复杂度）

Find the Celebrity

◆ 小技巧总结：

- 降时间复杂度 - > 找冗余
- 思维上双向：true时候，false的时候？

例题总结

- Valid Anagram
- Find All Anagrams in a String
  - Sliding window 类问题总结：都是左边减少一个，右边增加一个

- Word Abbreviation
- Word Abbreviation Set

例题总结

- Longest Consecutive Sequence
  - 小技巧总结：
  - Hash可以在O(1)的时间内确定一个元素是否存在，利用这个可以降低时间复杂度
  - 计算时间复杂度的方法，“每个元素只会被访问一遍”这句话所代表的方法很常用
- Load Balancer
- Find the Celebrity

快速点题

First Position Unique Character

九章算法

- http://www.lintcode.com/zh-cn/problem/first-position-unique-character/

- s = "lintcode" return 0
- s = "lovelintcode“ return 2

- Hash基础练习题

- Solution

Repeated DNA

- http://lintcode.com/en/problem/repeated-dna/

- s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"
- Return ["AAAAACCCCC", "CCCCCAAAAA"]

- Hash基础练习题
- 想想什么作为Hash的key

- Solution

Group Anagrams

- http://lintcode.com/zh-cn/problem/group-anagrams/

- ["eat", "tea", "tan", "ate", "nat", "bat"],
- Return [["ate", "eat", "tea"], ["nat", "tan"], ["bat"]]

- Anagrams系列练习题
- 想想anagrams的特征，想想什么作为hash的key

- Solution

Valid Sudoku

九章算法

- http://lintcode.com/zh-cn/problem/valid-sudoku/

- 行、列、块三种验证
- Hash （标记数组）练习题
- 如何算出这个位置属于第几块？

- Solution

Longest Substring Without Repeating Characters

九章算法

- http://lintcode.com/zh-cn/problem/longest-substring-without-repeating-characters/

- s = "abcabcbb“ 其无重复字符的最长子字符串是"abc"，长度为 3
- s = “bbbbb” 其无重复字符的最长子字符串为"b"，长度为1

- Hash + 双指针
- 类似sliding window 一边增加 一边减少

- Solution

Q & A

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
知乎专栏： http://zhuanlan.zhihu.com/jiuzhang
微博: http://www.weibo.com/ninechapter
官网: www.jiuzhang.com