基础算法和数据结构高频题 I

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;

- Read Characters From File - multiple calls 这题我们用的buffer是什么数据结构？

知识点回顾

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

- 把x转成k进制存放到digit数组中（填空）

  digit[i]= x____;

  x_____;

概要

- 区间类问题（3题）
- Hash 字符/字符串统计类问题（4题）
- 综合应用问题（1题）

区间类问题

Missing Interval

http://www.lintcode.com/zh-cn/problem/missing-interval/

http://www.jiuzhang.com/solutions/missing-interval/

Missing Interval

Example:

- 区间：[0, 99]　　挖去的点：[0, 1, 3, 50, 75]
- Output: ["2", "4->49", "51->74", "76->99"]

思路：

- 简单的模拟题
  - 两端点和一头一尾形成的区间 + for循环扫描中间形成的区间
  - 利用函数让自己的代码更简洁 （见代码）

- 特殊输入？
  - 实现时可能出现中间值超过int 范围
  - 去掉的点为空

时间复杂度：O(n)

Missing Interval

- Company Tags：Google

考点：

- 快速实现简单问题
- 特殊情况的处理

Missing Interval

能力维度：

2. 代码基础功力
5. 细节处理（corner case）

Merge intervals

http://www.lintcode.com/zh-cn/problem/merge-intervals/

http://www.jiuzhang.com/solutions/merge-intervals/

Merge intervals

Example:

- Given [1,3],[2,6],[8,10],[15,18],
  return [1,6],[8,10],[15,18]

思路一：

- 类似桶排序 mark数组，出现在线段内就标true
- 可是时间复杂度？

Merge intervals

Example:

- Given [1,3],[2,6],[8,10],[15,18],
  return [1,6],[8,10],[15,18]

思路二：

- 直接合并，比如[1,3] [2,6] 合并成[1,6]，不断合并，直到不能合并为止

Merge intervals

九章算法

- 问题：以什么样的顺序合并比较方便？

- 区间左端点从小到大排个序，从左往右扫一遍：
  - 不能合并 ->直接下一个
  - 能合并 -> 就合并

- Coding time

- 特殊输入？

时间复杂度：O(nlogn)

Merge intervals

- Company Tags: LinkedIn Google Facebook

考点：

- 是否想到直接合并
- 是否可以想到排序来简化问题

Merge intervals

能力维度：

2. 代码基础功力
3. 基础数据结构/算法
7. debug能力

Insert interval

http://www.lintcode.com/zh-cn/problem/insert-interval/

http://www.jiuzhang.com/solutions/insert-interval/

Insert interval

Example:

- Given intervals [1,3],[6,9], insert and merge [2,5] in as [1,5],[6,9].

思路：

- 做了merge interval 这一题就很简单了
- 先插入，然后直接套用merge interval

- 特殊输入？

时间复杂度：O(n)

Insert interval

- Company Tags：LinkedIn Google Facebook

考点：

- 是否可以通过上一题来解决follow up 问题

Insert interval

能力维度：

2. 代码基础功力
3. 基础数据结构/算法
7. debug能力

区间类问题总结

- 把区间在数轴上画出来 （脑海中 or 纸上）

- 往往会将区间按照左端点从小到大排个序

Hash 字符/字符串统计类问题

First Position Unique Character

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

http://www.jiuzhang.com/solutions/first-position-unique-character/

First Position Unique Character

- Given a string, find the first non-repeating character in it and return it's index. If it doesn't exist, return -1.

- **Examples:**
- s = "lintcode"
- return 0.
- s = "lovelintcode",
- return 2.

First Position Unique Character

思路：

- 扫一遍统计每个字符出现的次数（用什么统计？Hash）
- 再扫一遍找出第一个出现次数=1的字符

时间复杂度：O(n)

First Position Unique Character

- Company Tags：Amazon

考点：

- Hash的应用

First Position Unique Character

能力维度：

3. 基础数据结构/算法

Substring Anagrams

http://www.lintcode.com/zh-cn/problem/substring-anagrams/

http://www.jiuzhang.com/solutions/substring-anagrams/

Substring Anagrams

- **Input:**
  - s: "cbaebabacd" p: “aabc"
- **Output:**
  - [5]
- Anagrams 的充要条件？
  - 元素出现的次数一样就好了

Substring Anagrams

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)

Substring Anagrams

- 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

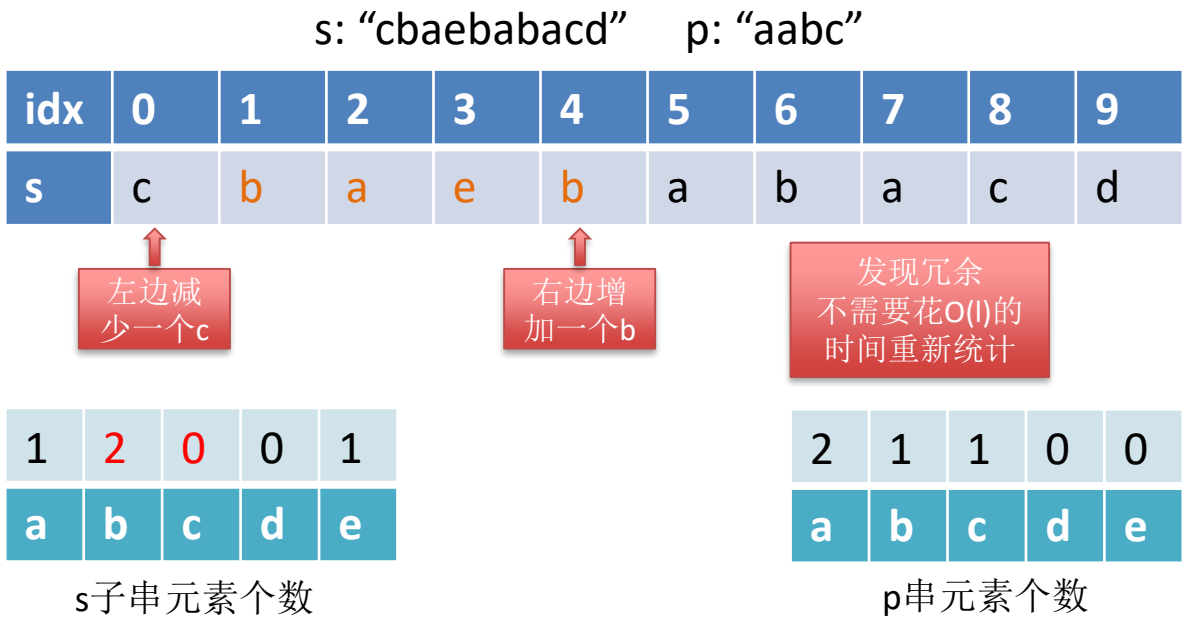
s子串元素个数

2	1	1	0	0
a	b	c	d	e

p串元素个数

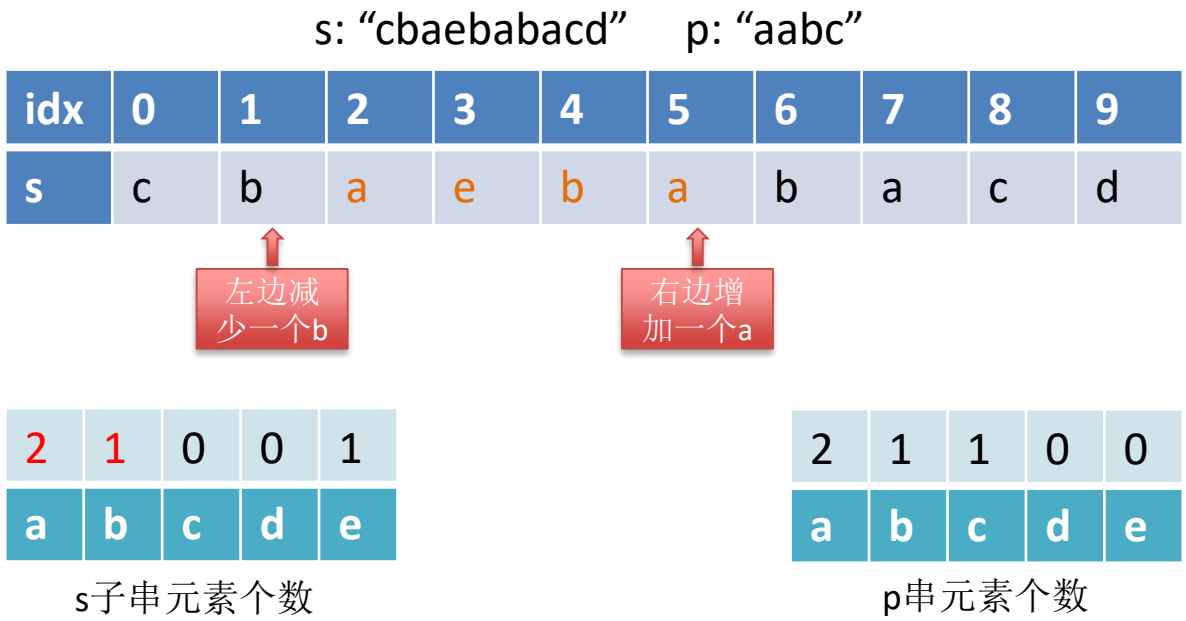
Substring Anagrams

- 可以更快吗？
  - 想想相邻的两个子串的差别？



Substring Anagrams

- 可以更快吗？
  - 想想相邻的两个子串的差别？



Substring Anagrams

- 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

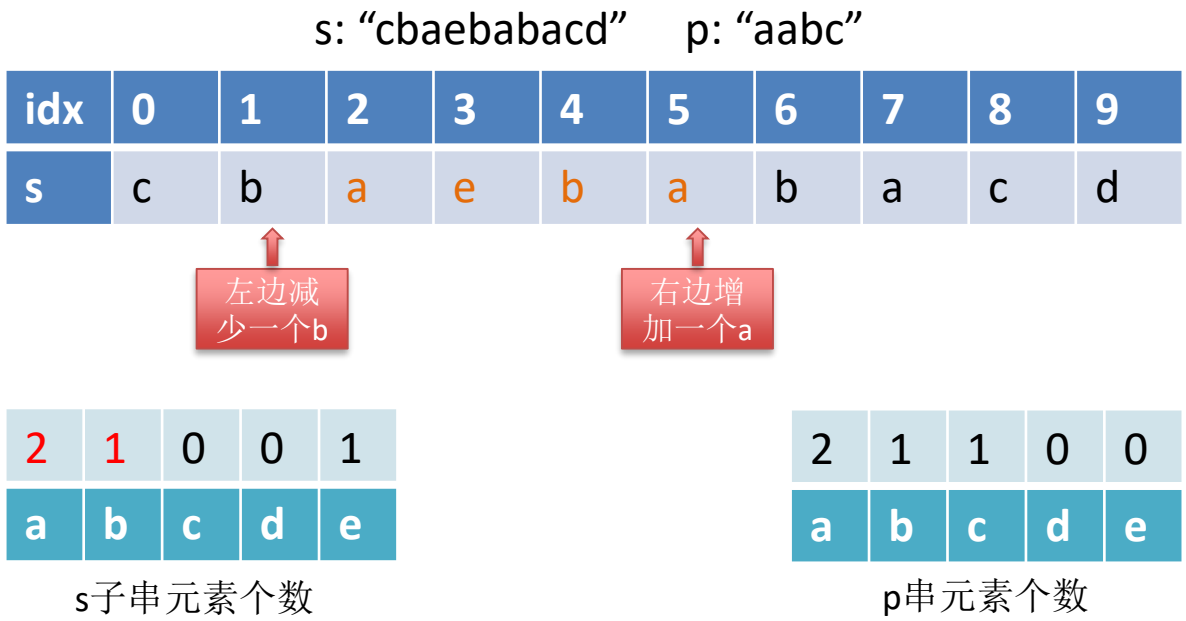
2	1	1	0	0
a	b	c	d	e

p串元素个数

时间复杂度：
O(n * 256) = O(n)

Substring Anagrams

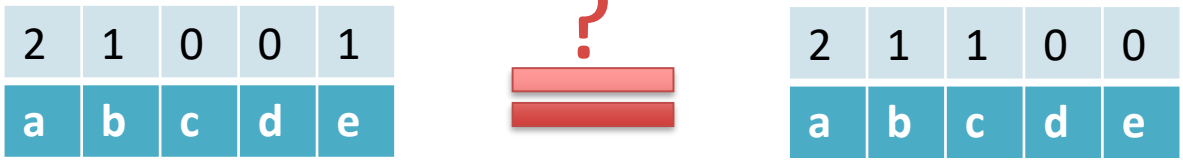
- 为什么有256这个常数项？可以优化吗？（扩展）



时间复杂度：
O(n * 256) = O(n)

Substring Anagrams

- 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 ?)



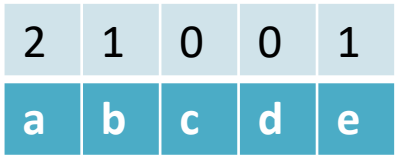
s子串元素个数
cntS

p串元素个数
cntP

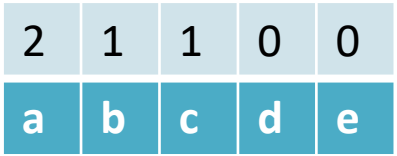
时间复杂度：
O(n * 256) = O(n)

Substring Anagrams

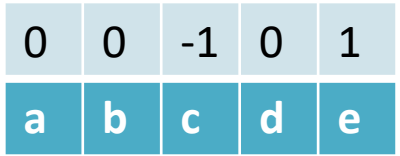
- 定义det = cntS – cntP
  - 如果det全为0 → cntS == cntP



s子串元素个数
cntS



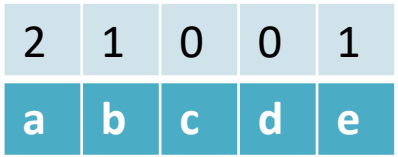
p串元素个数
cntP



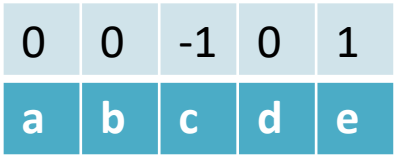
det

Substring Anagrams

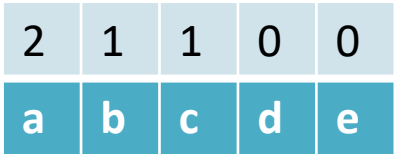
- 怎样判断det全为0？
  - 所有的元素绝对值求和 absSum == 0 → det全为0



s子串元素个数
cntS



det

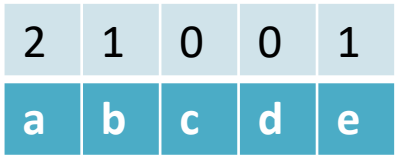


p串元素个数
cntP

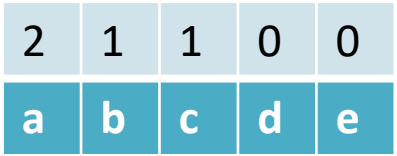
absSum = 0+0+abs(-1)+0+1 = 2

Substring Anagrams

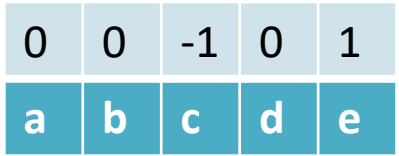
- absSum 怎么更新？
  - 在sliding window向右移动时，减去原有的，加上改变的　O(1)



s子串元素个数
cntS



p串元素个数
cntP

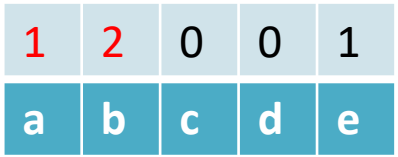


det

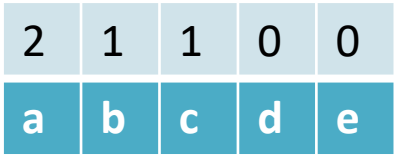
absSum = 0+0+abs(-1)+0+1 = 2

Substring Anagrams

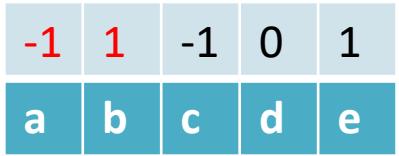
- absSum 怎么更新？
  - 在sliding window向右移动时，减去原有的，加上改变的　O(1)



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)

Substring Anagrams

- Company Tags：Amazon

考点：

- Sliding window + hash

相关题目:

- Sliding window median
- Sliding window maximum

Substring Anagrams

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

休息5分钟

Word Abbreviation Set

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

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

Word Abbreviation Set

- a) it --> it (no abbreviation)
- b) d|o|g --> d1g
- c) l | nternationalizatio | n --> i18n
- d) l | ocalizatio | n --> l10n

- 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

为什么考：

- 包装了一下的hash，不简单的裸考

Word Abbreviation Set

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
5. 细节处理（corner case）
7. debug能力

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，从小到大排序，然后从左到右扫一遍。
- 这样的方法有多少同学会想到？
- 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

Load Balancer

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
6. 算法分析（时间/空间复杂度）

本节课知识点总结

- Missing Interval
- Merge intervals
- Insert interval
- First Position Unique Character
- Substring Anagrams
  - Sliding window 类问题总结：都是左边减少一个，右边增加一个

本节课知识点总结

- Word Abbreviation Set
- 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

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