

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

- 业余爱好：音乐剧，浙大灵韵音乐剧社创始人，常春藤盟校春晚导演

- ZOOM 的用法
- 老师/助教回答提问方式
- 课程Ladder: <http://www.lintcode.com/zh-cn/ladder/14/>
- 九章QA: <http://www.jiuzhang.com/qa/>
- 错过第一节课不要紧，可以上下期课的第一节课（第一节课均免费）
- 答疑QQ群: 554891950

- 以下哪种情况可以直接使用数组作为hash (多选)
 - a. '1' b. 1 c. "123" d. 'A' e. "db"

- Corner Case处理技巧 (2题)
- 数组、字符串、栈与队列 (4题)
- 快速点题 (9题)

Corner Case处理技巧

Missing Ranges

<http://www.lintcode.com/problem/missing-ranges/>

<http://www.jiuzhang.com/solutions/missing-ranges/>

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: $O(n)$

- Company Tags: Google

考点:

- 快速实现简单问题
- 特殊情况的处理

能力维度:

- 2. 代码基础功力
- 5. 细节处理 (corner case)

Valid Number

<http://www.lintcode.com/zh-cn/problem/valid-number/>

<http://www.jiuzhang.com/solutions/valid-number/>

Example:

- “0” true
- “ 0.1 ” true
- “1 a” false
- “-2.5e+10” true

思路:

- 一个数怎么构成:
 - 符号 + 浮点数 + e + 符号 + 整数
- coding time

时间复杂度: $O(n)$

- Company Tags: LinkedIn

考点:

- 特殊情况的处理

能力维度:

- 2. 代码基础功力
- 5. 细节处理 (corner case)

- 常见Corner Case:
 1. 溢出MaxInt
 2. 输入数据为空
 3. 数组越界
- 解决Corner Case技巧:
 1. 先写程序，最后考虑Corner Case
 2. 64位长整型long 解决溢出MaxInt
 3. 数组稍微开大一点（比如加空格、dummy 0等）



数组、字符串、栈与队列

Moving Average from Data Stream

<http://www.lintcode.com/problem/moving-average-from-data-stream/>

<http://www.jiuzhang.com/solution/moving-average-from-data-stream/>

Moving Average from Data Stream

Example:

- size=3
- $m.next(1) = 1$
- $m.next(10) = (1 + 10) / 2$
- $m.next(3) = (1 + 10 + 3) / 3$
- $m.next(5) = (10 + 3 + 5) / 3$

思路:

- 我们先来一个最简单的做法
 - 来一个数就存数组
 - for 循环最近size 个数求和取平均返回
- 时间复杂度是多少呢?
 - 每次 $O(\text{size})$
- 怎样优化算法——如何快速求和?
 - 前缀和数组

- 什么是前缀和数组？
 - 下表中a是原始数组
 - 定义s是a的前缀和数组， $s[i] = a[1] + a[2] + a[3] \dots + a[i]$

idx	0	1	2	3	4	5	6	7	8	9
a		3	4	11	1	-2	3	2	1	5
s	0	3	7	18	19	17	20	22	23	28

- 方便快速求a数组中某一段的和
 - $a[k] + a[k + 1] + \dots + a[j] = s[j] - s[k - 1]$ 时间复杂度 $O(1)$
- 怎样快求s数组？
 - $s[i] = s[i - 1] + a[i]$ 时间复杂度 $O(n)$

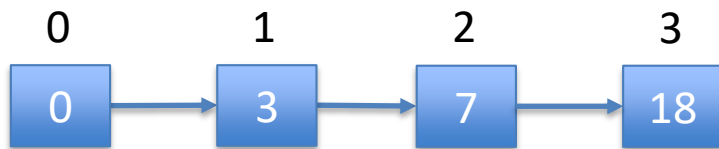
Moving Average from Data Stream

- 如何节省储存空间呢？（2种方法）

1. 链表保存sum

2. 数组滚动

Sum链表:

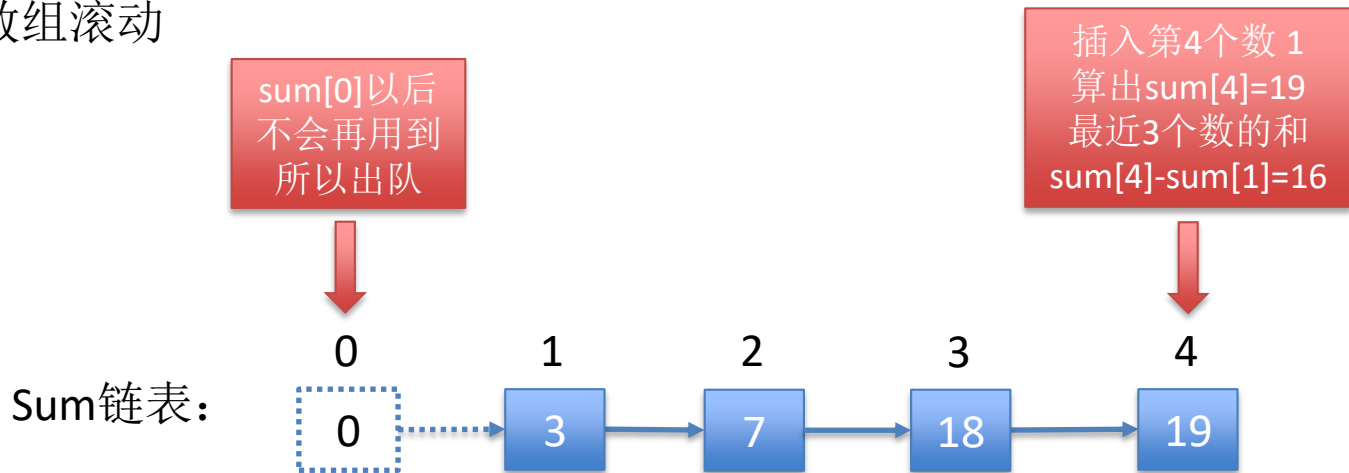


插入第3个数 11
算出 $\text{sum}[3]=18$
最近3个数的和
 $\text{sum}[3]-\text{sum}[0]=18$

- 如何节省储存空间呢？（2种方法）

1. 链表保存sum

2. 数组滚动



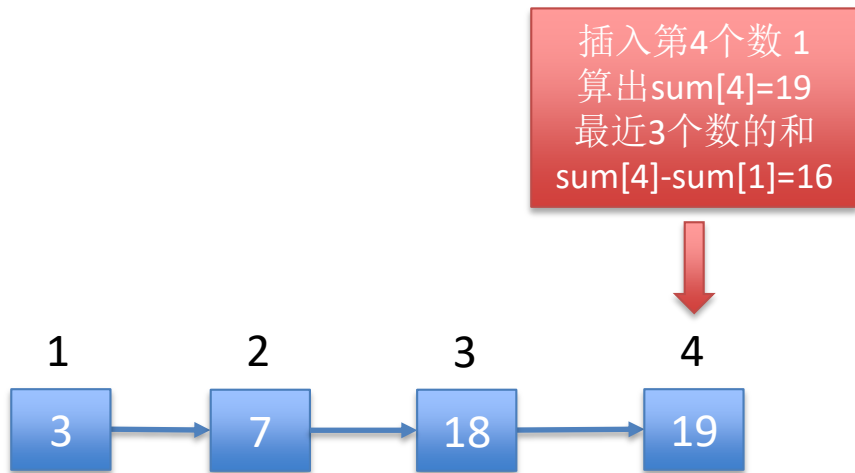
Moving Average from Data Stream

- 如何节省储存空间呢？（2种方法）

1. 链表保存sum

2. 数组滚动

Sum链表:

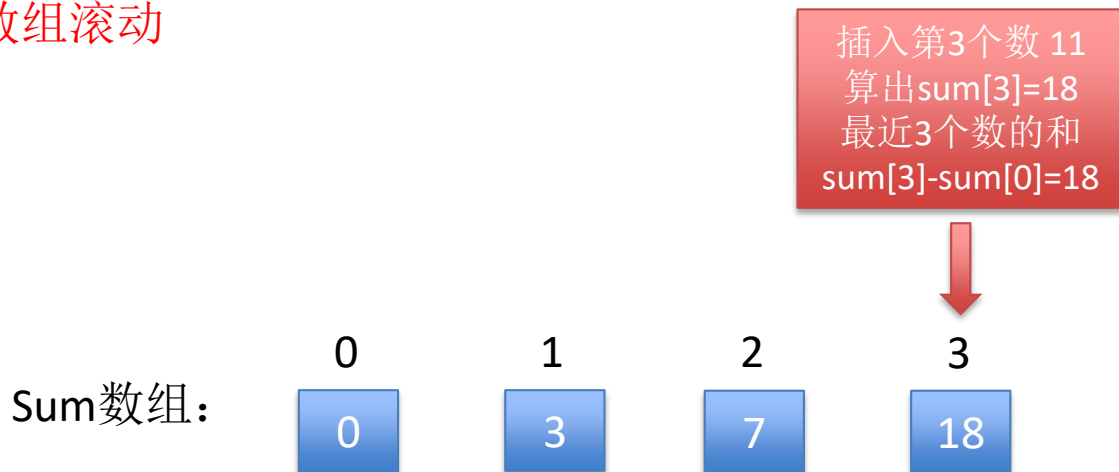


空间复杂度: $O(\text{size})$

- 如何节省储存空间呢？（2种方法）

1. 链表保存sum

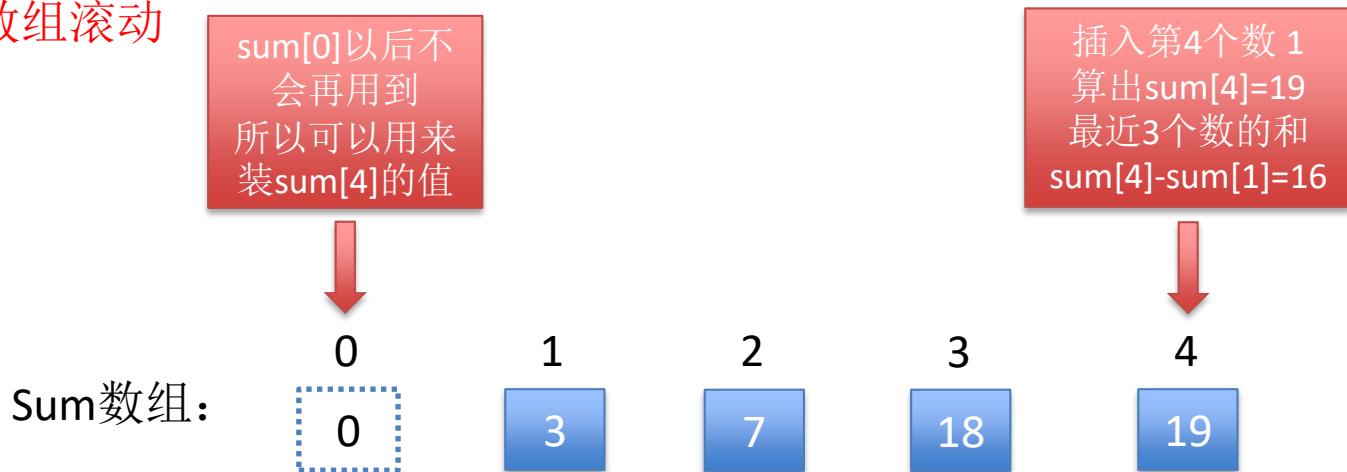
2. 数组滚动



- 如何节省储存空间呢？（2种方法）

1. 链表保存sum

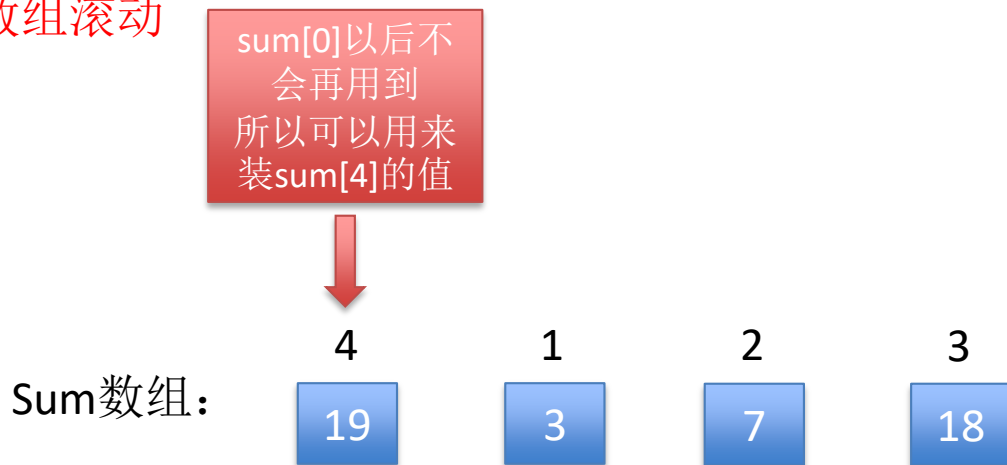
2. 数组滚动



- 如何节省储存空间呢？（2种方法）

1. 链表保存sum

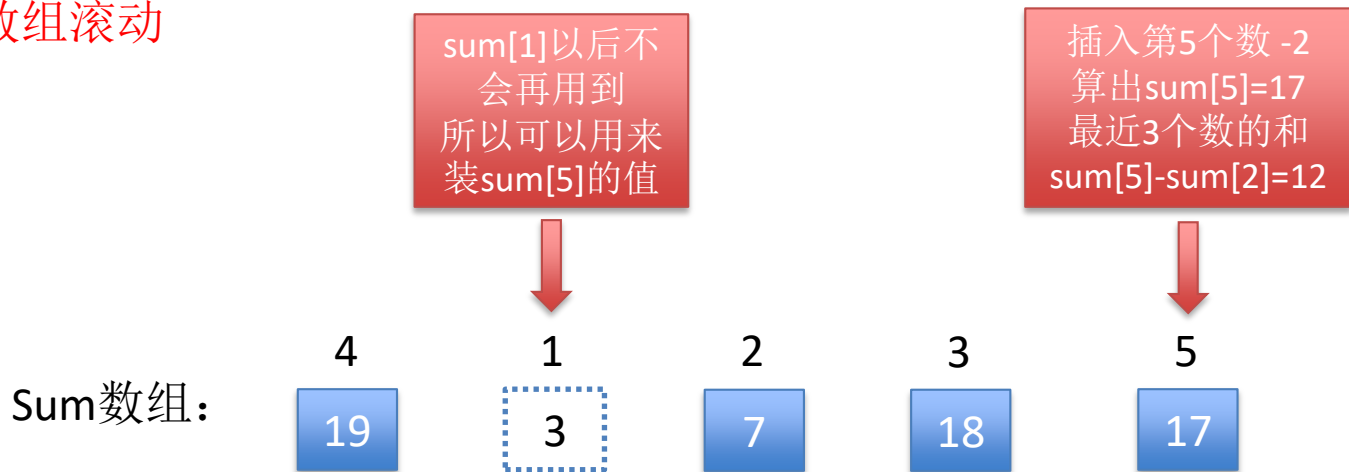
2. 数组滚动



- 如何节省储存空间呢？（2种方法）

1. 链表保存sum

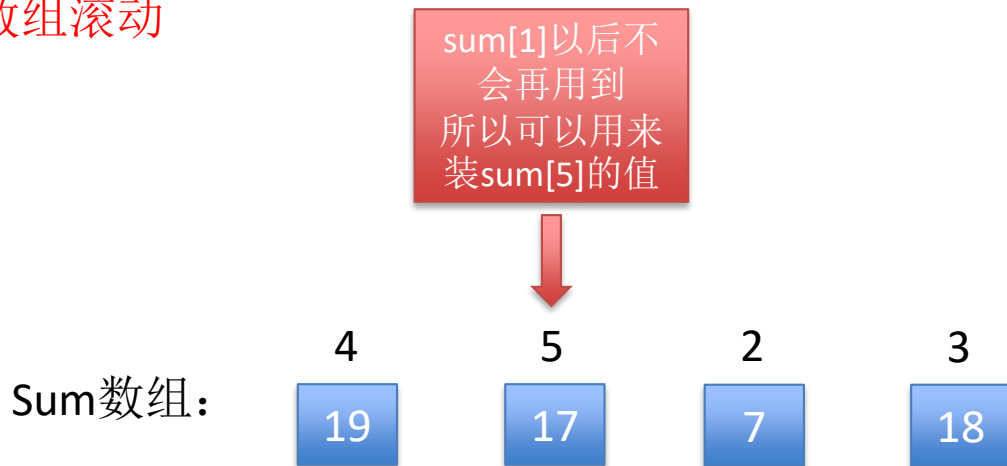
2. 数组滚动



- 如何节省储存空间呢？（2种方法）

1. 链表保存sum

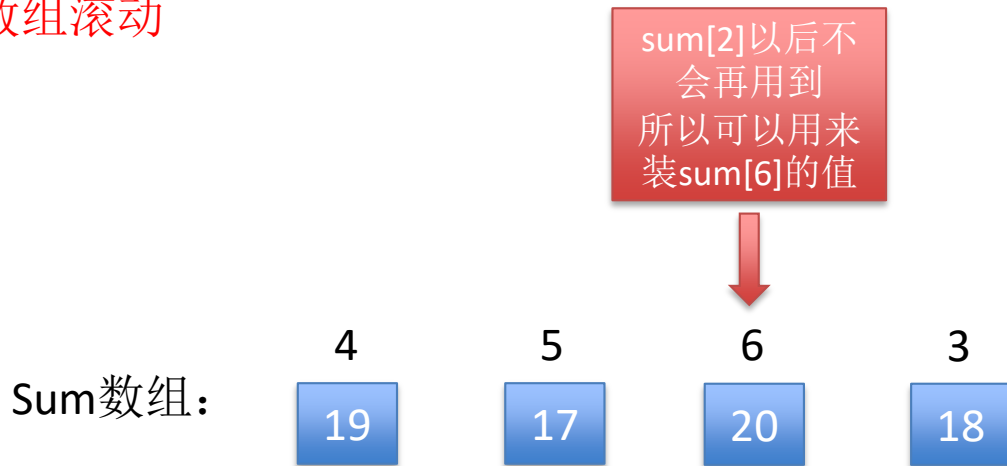
2. 数组滚动



- 如何节省储存空间呢？（2种方法）

1. 链表保存sum

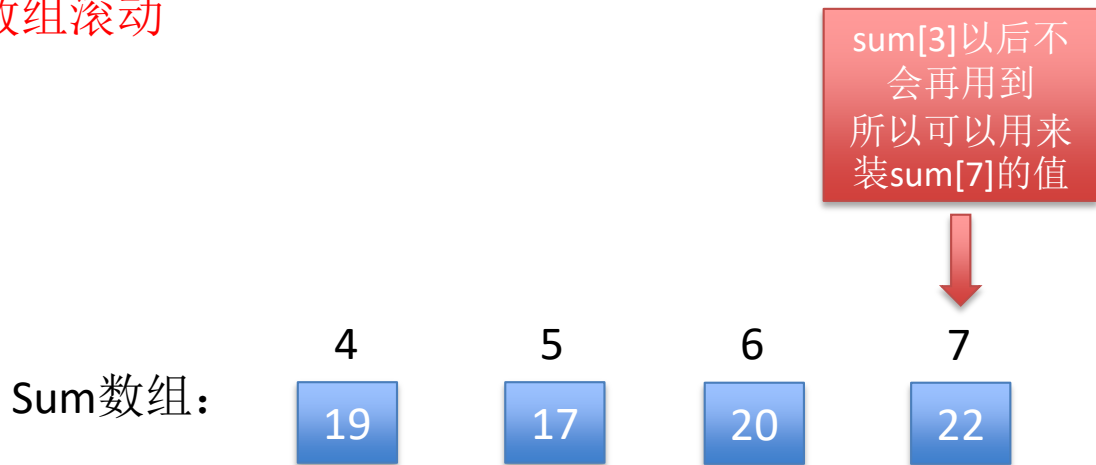
2. 数组滚动



- 如何节省储存空间呢？（2种方法）

1. 链表保存sum

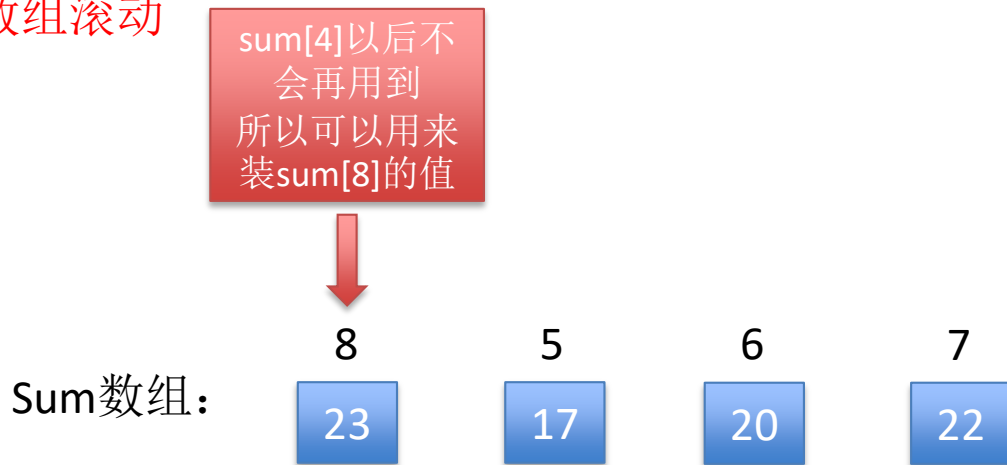
2. 数组滚动



- 如何节省储存空间呢？（2种方法）

1. 链表保存sum

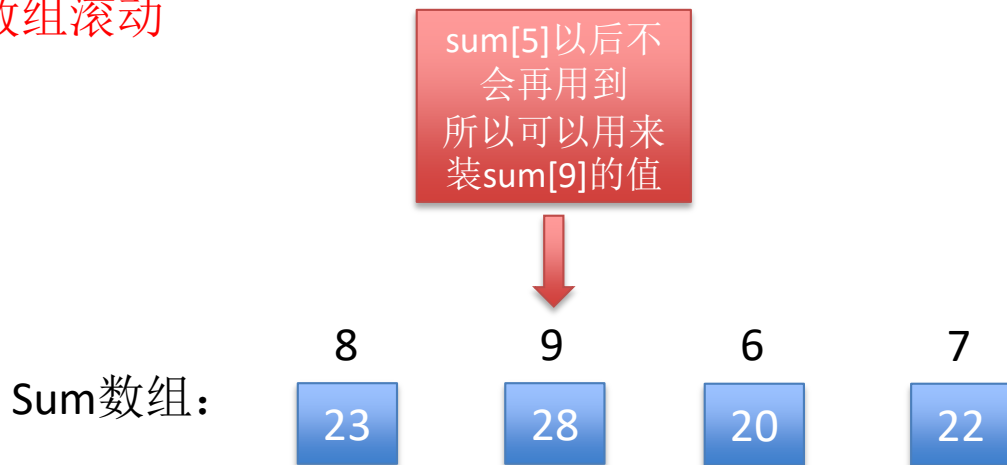
2. 数组滚动



- 如何节省储存空间呢？（2种方法）

1. 链表保存sum

2. 数组滚动



Moving Average from Data Stream

- 如何节省储存空间呢？（2种方法）

1. 链表保存sum

2. 数组滚动

逻辑位置	0	1	2	3	4	5	6	7	8	9
实际位置	0	1	2	3	0	1	2	3	0	1

- 怎样得到实际位置？
 - 逻辑位置取mod，这一题mod (size+1)

空间复杂度：O(size)

- Company Tags: Google

考点:

- 能否想到前缀和优化
- 能否进一步想到空间优化
- 15分钟bug free 写出来
- 电面题，筛选基本代码能力的面试者

能力维度:

3. 基础数据结构/算法

◆ 小技巧总结:

- 如何快速求和? 前缀和数组(dummy 0)
 - 如何节省储存空间呢? 链表/滚动
 - 写滚动的技巧 先写程序最后加滚动
-
- 滚动的应用:
 - DP 背包问题 **backpack** 系列
 - BFS 中的循环队列

休息5分钟

Encode and Decode Strings

<http://www.lintcode.com/problem/encode-and-decode-strings/>

<http://www.jiuzhang.com/solution/encode-and-decode-strings/>

思路:

- 简单的想法, 用'; ' (或者 '+' 等) 将字符串连起来
- 如果字符串中本身就有 ';' 呢? ';' 是连接符还是原有字符?
- 考虑\n \\ 这一类转义字符的原理
- 用 ':' 表示转义, 那么连接符就是': ; ' 表示 ':' 本身就是 ': : '

- abc def -> abc::def::
- ab:c def -> ab::c::def::
- ab::c def -> ab:::c::def::

思路:

- Encode:
 - “:” 替换成 “::”
 - “::” 将两个字符串连接起来
- Decode:
 - 从左往右扫
 - 遇到 “:” 则看其后一个字符
 - 遇到其他的则原文照搬

- Company Tags: Google
- 能力维度:
 4. 逻辑思维/算法优化能力

Longest Absolute File Path

<http://www.lintcode.com/problem/longest-absolute-file-path/>


<http://www.jiuzhang.com/solution/longest-absolute-file-path/>

思路:


- 有点小麻烦的纯模拟
- 技巧一：用`split('\n')` 将原串分割开，相当于一次读一行
- 技巧二：利用`'\t'`的个数来当前目录/文件 在第几层
- 技巧三：从上到下一行一行顺序读入，用类似栈操作，把前面几层的字符串长度都记下来

Longest Absolute File Path

	Level	Length
dir	0	3
subdir1	1	7
file1.ext	2	9
subsubdir1	2	10
sdir2	1	5
ssdir2	2	6
f2.ext	3	6



Level	0	1	2	3	4
Length	3				



Longest Absolute File Path

	Level	Length
dir	0	3
subdir1	1	7
file1.ext	2	9
subsubdir1	2	10
sdir2	1	5
ssdir2	2	6
f2.ext	3	6



Level	0	1	2	3	4
Length	3	7			




Longest Absolute File Path

	Level	Length
dir	0	3
subdir1	1	7
file1.ext	2	9
subsubdir1	2	10
sdir2	1	5
ssdir2	2	6
f2.ext	3	6


求和					
Level	0	1	2	3	4
Length	3	7	9		

Longest Absolute File Path

	Level	Length
dir	0	3
subdir1	1	7
file1.ext	2	9
subsubdir1	2	10
sdir2	1	5
ssdir2	2	6
f2.ext	3	6



Level	0	1	2	3	4
Length	3	7	10		



Longest Absolute File Path

	Level	Length
dir	0	3
subdir1	1	7
file1.ext	2	9
subsubdir1	2	10
sdir2	1	5
ssdir2	2	6
f2.ext	3	6



Level	0	1	2	3	4
Length	3	5	10		



Longest Absolute File Path

	Level	Length
dir	0	3
subdir1	1	7
file1.ext	2	9
subsubdir1	2	10
sdir2	1	5
ssdir2	2	6
f2.ext	3	6




Level	0	1	2	3	4
Length	3	5	10		


无效值

Longest Absolute File Path

	Level	Length
dir	0	3
subdir1	1	7
file1.ext	2	9
subsubdir1	2	10
sdir2	1	5
ssdir2	2	6
f2.ext	3	6



Level	0	1	2	3	4
Length	3	5	6		



Longest Absolute File Path

	Level	Length
dir	0	3
subdir1	1	7
file1.ext	2	9
subsubdir1	2	10
sdir2	1	5
ssdir2	2	6
f2.ext	3	6



	<div>求和</div>				
Level	0	1	2	3	4
Length	3	5	6	6	

Longest Absolute File Path

	Level	Length
dir	0	3
subdir1	1	7
file1.ext	2	9
subsubdir1	2	10
sdir2	1	5
ssdir2	2	6
f2.ext	3	6



Level	0	1	2	3	4
Length	3	5	6	6	
sum	3	8	14	20	



- Company Tags: Google

考点:

- 是否可以形象化的思考这个问题
- 是否可以熟练的处理字符串

能力维度：

1. 理解问题
2. 代码基础功力
5. 细节处理（corner case）
7. debug能力

Read N Characters Given Read4 II - Call multiple times

<http://www.lintcode.com/problem/read-n-characters-given-read4-ii-call-multiple-times/>

<http://www.jiuzhang.com/solution/read-n-characters-given-read4-ii-call-multiple-times/>

Read N Characters Given Read4 II-Call multiple times 九章算法

Example : 1 2 3 4 5 6 7 8 9

- 每次读的个数 6 1 4
- 第一次: 1 2 3 4 5 6
- 第二次: 7
- 第三次: 8 9

难点:

- 如果只读3个，剩下的一个怎么处理？（读多的怎么留着给下次用？）
- Read4 这个函数只读了2个怎么办？（读到末尾时，没有读全4个）

Read N Characters Given Read4 II-Call multiple times



九章算法

思路:

- 计算机中硬盘是怎么读数据的?



- 本题类似内存从硬盘读数据
 - 输入字符数组: 硬盘
 - `read4`: 读硬盘一整块到缓冲区
 - `read(n)`: 内存实际需求
 - 需要我们做的: 实现一个缓冲区及相应功能

Read N Characters Given Read4 II-Call multiple times



九章算法

思路:

- 缓冲区用什么数据结构?
 - 队列（**FIFO**），因为要保持数据顺序不变
- 缓冲区队列的逻辑?
 - 队列为空时就进队（**read4**）
 - 队列不为空时就满足内存的请求，也就是出队

Read N Characters Given Read4 II-Call multiple times 九章算法

思路:

- Example : 1 2 3 4 5 6 7 8 9
- 每次读的个数: 6 1 4



请求	result
6	



队列为空

Read N Characters Given Read4 II-Call multiple times 九章算法

思路:

- Example : 1 2 3 4 5 6 7 8 9
- 每次读的个数: 6 1 4



请求	result
6	



Read N Characters Given Read4 II-Call multiple times 九章算法

思路:

- Example : 1 2 3 4 5 6 7 8 9
- 每次读的个数: 6 1 4



请求	result
6	1 2 3 4



队列非空：出队

Read N Characters Given Read4 II-Call multiple times 九章算法

思路:

- Example : 1 2 3 4 5 6 7 8 9
- 每次读的个数: 6 1 4



请求	result
6	1 2 3 4

5	6	7	8
---	---	---	---

队列为空: 进队

Read N Characters Given Read4 II-Call multiple times 九章算法

思路:

- Example : 1 2 3 4 5 6 7 8 9
- 每次读的个数: 6 1 4



请求	result
6	1 2 3 4 5 6



队列非空：出队

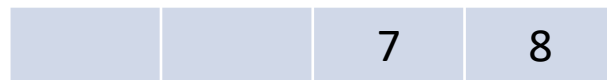
Read N Characters Given Read4 II-Call multiple times 九章算法

思路:

- Example : 1 2 3 4 5 6 7 8 9
- 每次读的个数: 6 1 4



请求	result
6	1 2 3 4 5 6
1	



队列非空

Read N Characters Given Read4 II-Call multiple times 九章算法

思路:

- Example : 1 2 3 4 5 6 7 8 9
- 每次读的个数: 6 1 4



请求	result
6	1 2 3 4 5 6
1	7



Read N Characters Given Read4 II-Call multiple times 九章算法

思路:

- Example : 1 2 3 4 5 6 7 8 9
- 每次读的个数: 6 1 4



请求	result
6	1 2 3 4 5 6
1	7
4	



队列非空

Read N Characters Given Read4 II-Call multiple times 九章算法

思路:

- Example : 1 2 3 4 5 6 7 8 9
- 每次读的个数: 6 1 4



请求	result
6	1 2 3 4 5 6
1	7
4	8



Read N Characters Given Read4 II-Call multiple times 九章算法

思路:

- Example : 1 2 3 4 5 6 7 8 9
- 每次读的个数: 6 1 4



请求	result
6	1 2 3 4 5 6
1	7
4	8




队列为空: 进队

Read N Characters Given Read4 II-Call multiple times 九章算法

思路:

- Example : 1 2 3 4 5 6 7 8 9
- 每次读的个数: 6 1 4



请求	result
6	1 2 3 4 5 6
1	7
4	8 9



队列非空: 出队

Read N Characters Given Read4 II-Call multiple times 九章算法

思路:

- Example : 1 2 3 4 5 6 7 8 9
- 每次读的个数: 6 1 4



请求	result
6	1 2 3 4 5 6
1	7
4	8 9



队列为空: 进队

全部读完: 程序
退出

时间复杂度: $O(n)$

Read N Characters Given Read4 II-Call multiple times 九章算法

- Company Tags: Google Facebook

考点:

- 对队列这种数据结构的理解
- 对细节的处理，大家都会但不容易写对
- Onsite
- 25min

Read N Characters Given Read4 II-Call multiple times 九章算法

能力维度:

- 3. 基础数据结构/算法
- 5. 细节处理 (corner case)
- 7. debug能力

◆ Missing Ranges

◆ Valid Number

– 常见Corner Case:

1. 溢出MaxInt
2. 输入数据为空
3. 数组越界

– 解决Corner Case技巧:

1. 先写程序，最后考虑Corner Case
2. 长整型long 解决溢出MaxInt
3. 数组稍微开大一点（比如加空格、dummy 0等）

◆ Moving Average from Data Stream

◆ 小技巧总结:

- 如何快速求和? 前缀和数组
- 如何节省储存空间呢? 链表/滚动
- 写滚动的技巧 先写程序最后加滚动

◆ Encode and Decode Strings

◆ Longest Absolute File Path

◆ Read N Characters Given Read4 II - Call multiple times

快速点题

String to Integer (atoi)

- <http://www.lintcode.com/problem/string-to-integer-atoi/>
- “+123” 123
- “-456” -456
- 类似Valid Number
- 只考虑整数，不用考虑浮点数
- [Solution](#)

- <http://www.lintcode.com/problem/one-edit-distance/>
- s="abcd" t="abce" output: true
- s="abcd" t="abcef" output: false
- 想一想特殊情况：两字符串长度相差太大时？两个字符串长度一样？
- 三种情况： 长度差 >1 长度差 $=0$ 长度差 $=1$
- [Solution](#)

- <http://www.lintcode.com/zh-cn/problem/merge-intervals/>
- Given [1,3], [2,6], [8,10], [15,18]
- return [1,6], [8,10], [15,18]
- 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
- 区间左端点从小到大排个序
- [Solution](#)

- <http://www.lintcode.com/zh-cn/problem/insert-interval/>
- Given [1,3], [6,9], insert and merge [2,5]
- return [1,5], [6,9]
- 做了merge interval 这一题就很简单了
- 先插入，然后直接套用merge interval
- [Solution](#)

- <http://www.lintcode.com/problem/strobogrammatic-number/>
- "69", "88", and "818" are all mirror numbers
- 纯模拟：从左到右扫一遍，看对应的位置是不是反着的
- 沟通清楚哪些数是对称的，比如“5”是不是对称的
- [Solution](#)

- <http://www.lintcode.com/zh-cn/problem/interval-sum/>
- 对于数组 $[1, 2, 7, 8, 5]$, 查询 $[(1, 2), (0, 4), (2, 4)]$
- 返回 $[9, 23, 20]$
- 前缀和数组应用
- 注意dummy 0
- [Solution](#)

- <http://www.lintcode.com/zh-cn/problem/min-stack/>
- push(1), pop(), push(2), push(3), min(), push(1), min()
- 返回 1, 2, 1
- 类似前缀和数组，实现一个“前缀min”数组
- 注意dummy 0
- [Solution](#)

- <http://www.lintcode.com/zh-cn/problem/valid-parentheses/>
- []([]{}) 匹配
- [](不匹配
- 经典栈应用问题，从左往右扫，左括号对应入栈，相匹配右括号对应出栈
- 注意三种不匹配的情况：
 - a. 扫完后栈中还有元素
 - b. 扫描过程中栈是空的但还要执行出栈操作
 - c. 要执行出栈但是括号不匹配，比如 [])
- [Solution](#)

- <http://www.lintcode.com/problem/evaluate-reverse-polish-notation/>
- ["2", "1", "+", "3", "*"] -> ((2 + 1) * 3) -> 9
- ["4", "13", "5", "/", "+"] -> (4 + (13 / 5)) -> 6
- 栈在表达式求值中的运用
- 遇到数字直接push进栈
- 遇到+ - * /时pop出栈顶2个数并相应运算求值，结果push进栈
- [Solution](#)



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://zhuankan.zhihu.com/jiuzhang>

微博: <http://www.weibo.com/ninechapter>

官网: www.jiuzhang.com