春节7天练讲Day4：散列表和字符串

你好，我是王争。初四好！

为了帮你巩固所学，真正掌握数据结构和算法，我整理了数据结构和算法中，必知必会的30个代码实现，分7天发布出来，供你复习巩固所用。今天是第四篇。

和昨天一样，你可以花一点时间，来完成测验。测验完成后，你可以根据结果，回到相应章节，有针对性地进行复习。

前几天的内容。如果你错过了，点击文末的“上一篇”，即可进入测试。

关于散列表和字符串的4个必知必会的代码实现

散列表

- 实现一个基于链表法解决冲突问题的散列表
- 实现一个LRU缓存淘汰算法

字符串

- 实现一个字符集，只包含a～z这26个英文字母的Trie树
- 实现朴素的字符串匹配算法

对应的LeetCode练习题（@Smallfly 整理）

字符串

- Reverse String （反转字符串）

英文版：https://leetcode.com/problems/reverse-string/

中文版：https://leetcode-cn.com/problems/reverse-string/

- Reverse Words in a String（翻转字符串里的单词）

英文版：https://leetcode.com/problems/reverse-words-in-a-string/

中文版：https://leetcode-cn.com/problems/reverse-words-in-a-string/

- String to Integer (atoi)（字符串转换整数 (atoi)）

英文版：https://leetcode.com/problems/string-to-integer-atoi/

中文版：https://leetcode-cn.com/problems/string-to-integer-atoi/

做完题目之后，你可以点击“请朋友读”，把测试题分享给你的朋友，说不定就帮他解决了一个难题。

祝你取得好成绩！明天见！

精选留言

李皮皮皮皮皮

散列表的核心是散列函数和冲突解决算法，以及装载因子过大时如何扩容。散列函数的设计较为复杂，一般使用现有的函数，如murmur散列。冲突解决一般有开放寻址法和链表法。查看开源项目的源码实现很有意思，例如lua的table实现，是结合了两个方法的非常优雅的实现。根据装载因子扩容一般保持在2，在占用空间较大时慢慢缩减为1.5，1.25……如golang的实现。为了避免rehash时的延迟，可以使用先分配，后逐步散列的方法，redis就是使用这个方法的。

字符串是编程中一定会出现的问题，变种非常多，反转，反转单词，字串，最长字串，最长子序列等等，有时解决问题需要多种数据结构与算法的结合。

2019-02-08 07:34

kai

实现一个 LRU 缓存淘汰算法:

```
import java.util.HashMap;
import java.util.Iterator;

public class LRU<K,V> {

private Node head;
private Node tail;
private HashMap<K, Node> map;
private int maxSize;

private class Node {
Node pre;
Node next;
K k;
V v;
public Node(K k, V v) {
this.k = k;
this.v = v;
}
}

public LRU(int maxSize) {
this.maxSize = maxSize;
this.map = new HashMap<>(maxSize * 4 / 3);
head = new Node(null, null);
tail = new Node(null, null);
head.next = tail;
tail.pre = head;
}

public V get(K key) {
if (!map.containsKey(key)) {
return null;
}
Node node = map.get(key);
unlink(node);
appendToHead(node);
return node.v;
}
public void put(K key, V value) {
if (map.containsKey(key)) {
Node node = map.get(key);
unlink(node);
}
Node node = new Node(key, value);
appendToHead(node);
map.put(key, node);
if (map.size() > maxSize) {
Node toRemove = removeTail();
```

```
map.remove(toRemove.k);
}
}
private Node removeTail() {
Node node = tail.pre;
Node pre = node.pre;
tail.pre = pre;
pre.next = tail;
node.next = null;
node.pre = null;
return node;
}
private void appendToHead(Node node) {
Node next = head.next;
node.next = next;
node.pre = head;
next.pre = node;
head.next = node;
}
private void unlink(Node node) {
Node pre = node.pre;
Node next = node.next;
pre.next = next;
next.pre = pre;
node.pre = null;
node.next = null;
}

}
```

2019-02-11 10:36

kai

哇塞，老师太牛了，过年都在更新，一直在跟着老师的课程在总结归纳，同时找来题目在练习，这个专栏很牛~

2019-02-08 21:15

你看起来很好吃

字符串转换整数python实现：

```
import math

class Solution:
def myAtoi(self, str: 'str') -> 'int':
result = 0

i, N, former = 0, len(str), 1

while i < N:
if str[i] != ' ':
break
i += 1

if i < N and (str[i] == '-' or str[i] == '+'):
former = -1 if str[i] == '-' else 1
i += 1
```

```
while i < N:
if str[i].isdigit():
result = result * 10 + int(str[i])
i += 1
else:
break

result = result * former
if result > (math.pow(2, 31) * -1) and result < (math.pow(2,31) - 1):
return result
elif former > 0 :
return int(math.pow(2,31) - 1)
else:
return int(math.pow(2,31) * -1)
```

2019-02-10 00:14

纯洁的憎恶

1.从两端向中间两两对调，时间复杂度O（n）。

2.先去空格，O（n^2）。从两端向中间查找单词，找到一对单词s、t（s在前t在后），保存这两个单词，如果s长t短，把它们之间的字符串整体左移长度差个字符，反之整体右移长度差个字符，再把s和t按调整后位置向原数组赋值，O（n^2）。

3.如果字符串全为空、全为空格、首个非空格字符非法，则返回0。若首个合法字符位“-”则记录。int num=0；逐个读取数字部分字符a，若a合法，则num*=10，然后num+=a-‘0’，直到读取结束或者读到非法字符，此时如果记录的首个合法字符为“-”返回num*（-1），否则返回num。不知int型运算过程中结果值溢出，是否自动将值设置为边界值。如果不能就要在每次乘10的时候结合“-”考察一下是否越界。

2019-02-09 18:57

你看起来很好吃

反转字符串python实现：

```
class Solution:
def reverseString(self, s: 'List[str]') -> 'None':
"""
Do not return anything, modify s in-place instead.
"""
i, N = 0, len(s)
while i < N//2:
s[i], s[N-1-i] = s[N-1-i], s[i]
i += 1

print(s)
```

2019-02-09 16:19

molybdenum

老师新年好，这是我第四天的作业

https://blog.csdn.net/github_38313296/article/details/86818634

2019-02-09 16:17

ext4

反转字符串

```
class Solution {
public:
string reverseString(string s) {
int length = s.length();
if (length < 2) {
```

```
return s;
}
int i = 0, j = length - 1;
char temp;
while (i < j) {
temp = s[i];
s[i] = s[j];
s[j] = temp;
i++;
j--;
}
return s;
}
};
```

2019-02-09 07:37

黄丹

王争老师，新年的第四天快乐，已经很晚了，祝您好梦！
关于基于链表法解决冲突的散列表，就是使用一个数组，将值散列到数组下标上，但数组的每个值又是一个链表的头结点，当遇到冲突时就遍历该头结点后链表。其实java中hashmap底层的实现原理就是一个基于链表解决冲突的动态扩容的数组。大家有兴趣可以自己实现一下hashmap的底层数据结构，还是很有收获的。
今天leetcode上的三题都是关于字符串的，下面是我的解题思路和代码
1. Reverse String （反转字符串）
解题思路：这一题要求使用O(1)的空间将字符串进行反转，就是原地反转字符串，对字符串s[0…n-1]来说当i<n/2;将i与n-1-i位置的字符进行互换就行.
代码： https://github.com/yyxd/leetcode/blob/master/src/leetcode/strings/Problem344_ReverseString.java
2. Reverse Words in a String （翻转字符串里的单词）
解题思路：这一题我用的是java中的StringBuilder处理字符串，先用split函数将字符串按空格分开，但是当有多个连续空格时，一定要注意这种不能当做单词处理，要检查一下。
代码： https://github.com/yyxd/leetcode/blob/master/src/leetcode/strings/Problem151_ReverseWordsInString.java
3. String to Integer (atoi) 字符串转换整数 (atoi))
解题思路：将字符串转化为整数,首先是对数字前面的+/-进行处理，遍历字符串，如果不是数字字符就break，自己不懂得地方在于如何将大于INT.MAX 的值转化为 INT.MAX,将INT.MIN的值化为 INT.MIN，我自己想到的解法是用更高精度的long去保存，然后转化成int类型的值
代码： https://github.com/yyxd/leetcode/blob/master/src/leetcode/strings/Problem8_atoi.java

2019-02-08 23:18

虎虎

itoa

```
public class Solution {
public int myAtoi(String str) {
if (str.isEmpty())
return 0;
str = str.trim();
int i = 0, ans = 0, sign = 1, len = str.length();
if (str.charAt(i) == '-' || str.charAt(i) == '+')
sign = str.charAt(i++) == '+' ? 1 : -1;
for (; i < len; ++i) {
int tmp = str.charAt(i) - '0';
if (tmp < 0 || tmp > 9)
break;
```

```
if (ans > Integer.MAX_VALUE / 10
ll (ans == Integer.MAX_VALUE / 10 && Integer.MAX_VALUE % 10 < tmp))
return sign == 1 ? Integer.MAX_VALUE : Integer.MIN_VALUE;
else
ans = ans * 10 + tmp;
}
return sign * ans;
}
}
```

2019-02-08 23:13

_CountingStars

反转字符串 go 语言实现

```
package main

import "fmt"

func reverseString(s []byte) {
length := len(s)
for i := 0; i < length/2; i++ {
s[i], s[length-i-1] = s[length-i-1], s[i]
}
}

func main() {
testString := []byte{'h', 'e', 'l', 'l', 'o'}
fmt.Println(string(testString[:]))
reverseString(testString)
fmt.Println(string(testString[:]))
}
```

2019-02-08 22:02

失火的夏天

LRU缓存淘汰算法

```
private class Node{
private Node prev;
private Node next;
private int key;
private int value;

Node(int key,int value){
this.key = key;
this.value = value;
}
}

private Node head;// 最近最少使用，类似列队的头，出队
private Node tail;// 最近最多使用，类似队列的尾，入队
private Map<Integer,Node> cache;
private int capacity;

public LRUCache(int capacity) {
```

```
this.cache = new HashMap<>();
this.capacity = capacity;
}

public int get(int key) {
Node node = cache.get(key);
if(node == null){
return -1;
}else{
moveNode(node);
return node.value;
}
}

public void put(int key, int value) {
Node node = cache.get(key);
if (node != null){
node.value = value;
moveNode(node);
}else {
removeHead();
addNode(new Node(key,value));
}
cache.put(key,node);
}

private void removeHead(){
if (cache.size() == capacity){
Node tempNode = head;
cache.remove(head.key);
head = head.next;
tempNode.next = null;
if (head != null)
head.prev = null;
}
}

private void addNode(Node node){
if (head == null)
head = tail = node;
else
addNodeToTail(node);
}

private void addNodeToTail(Node node){
node.prev = tail;
tail.next = node;
tail = node;
}

private void moveNode(Node node){
```

```
if(head == node && node != tail){
head = node.next;
head.prev = null;
node.next = null;
addNodeToTail(node);
}else if (tail == node){
}else {
node.prev.next = node.next;
node.next.prev = node.prev;
node.next = null;
addNodeToTail(node);
}
}
}
```

2019-02-08 21:41

峰

反转字符串

```
class Solution {
public void reverseString(char[] s) {
int start = 0;
int end = s.length - 1;
while(start < end){
swap(s,start,end);
start++;
end--;
}
}

public void swap(char[] array,int a,int b){
char tmp = array[a];
array[a] = array[b];
array[b] = tmp;
}



}
```

2019-02-08 19:33

老杨同志

//字符串转换整数

```
package com.jxyang.test.geek.day4.Solution;

class Solution2 {
public int myAtoi(String str) {
if(str==null){
return 0;
}
char[] arr= str.toCharArray();
boolean flag = false;
boolean numBegin = false;
int result = 0;
for(int i =0;i<arr.length;i++){
```

```
if(numBegin && (arr[i]=='-'||arr[i]=='+'||arr[i]==' ')){
break;
}else if(arr[i]==' ') {
continue;
}else if(arr[i]=='+'){
numBegin = true;
continue;
}else if(arr[i]=='-'){
flag = true;
numBegin = true;
continue;
}else if(arr[i]>='0'&&arr[i]<='9'){
numBegin = true;
if(result==0){
result = flag?('0'-arr[i]):(arr[i]-'0');
}else{
try{
result = Math.multiplyExact(result,10);
result = Math.addExact(result,flag?('0'-arr[i]):(arr[i]-'0'));
}catch (Exception e){
if(flag){
return Integer.MIN_VALUE;
}else{
return Integer.MAX_VALUE;
}
}
}
}else{
break;
}
}
return result;
}
public static void main(String[] args) {
Solution2 solution2 = new Solution2();
System.out.println(solution2.myAtoi("42"));
System.out.println(solution2.myAtoi(" +0 123"));//期望123
System.out.println(solution2.myAtoi(" -42"));
System.out.println(solution2.myAtoi("4193 with words"));
System.out.println(solution2.myAtoi("words and 987"));
System.out.println(solution2.myAtoi("-91283472332"));//期望-2147483648
System.out.println(solution2.myAtoi("+1"));//期望-2147483648
}
}
```

2019-02-08 18:23

老杨同志

```
class Solution {
//反转字符串
public void reverseString(char[] s) {
if(s==nullls.length<2){
return;
```

```
}
int l=0;
int r=s.length-1;
while (l<r){
char tmp = s[l];
s[l] = s[r];
s[r] = tmp;
l++;
r--;
}
}
}
```

2019-02-08 17:12

C_love

Reverse Words in a String

```
public class Solution {
public String reverseWords(String s) {
final List<String> words = new ArrayList<>();
final char[] charArray = s.toCharArray();

int start = 0;
int end = 0;
while (end < s.length()) {
if (' ' == charArray[end]) {
if (start != end) {
words.add(getWord(charArray, start, end));
start = end;
}
start++;
end++;
} else {
end++;
}
}

if (start != end) {
words.add(getWord(charArray, start, end));
}

Collections.reverse(words);
return String.join(" ", words);
}

private String getWord(final char[] charArray, final int start, final int end) {
char[] tmp = new char[end - start];
int pos = 0;
for(int i = start; i < end; i++) {
tmp[pos++] = charArray[i];
}
return new String(tmp);
```

```
}
}
```
2019-02-08 12:40