**《算法设计与分析》周练记录表-第2周**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **班学号** | **姓名** | **OJ账号** | **练习平台** | **完成题目数量** | **题号** |
| 11117233 | 徐鸿飞 | X-cool | LeetCode | 6 | 46，47，215，4，23，53 |
| Xhofe | LintCode | 2 | 1095，1639 |
|  | 牛客网 |  |  |
|  | CCF-CSP |  |  |

LeetCode：https://leetcode.com/ 或https://leetcode-cn.com/

LintCode：https://www.lintcode.com/

牛客网（剑指Offer）：https://www.nowcoder.com/ta/coding-interviews?page=1

CCF-CSP：http://www.cspro.org/lead/application/ccf/login.jsp

\*注：本课程作业报告每周提交一次，请所有同学每周日晚21:00前交给各班学习委员，由学习委员统计并上传至微云，课堂上将随机抽取同学讲解自己的练习（5分钟）。

## 一、练习题目及代码

#### 【题目1】LeetCode46

给定一个没有重复数字的序列，返回其所有可能的全排列。

**【代码】**

**class** Solution:

**def** permute(self, nums: list[int]) -> list[list[int]]:

self.len = len(nums)

**if** self.len==1:

**return** [nums]

self.list=list()

self.per(nums)

**return** self.list

**def** per(self,nums):

**if** len(nums)==1:

**return** [[nums[0]]]

**elif** self.len==len(nums):

**for** i **in** range(len(nums)):

temp=self.per(nums[:i]+nums[i+1:])

**for** j **in** temp:

self.list.append([nums[i]]+j)

**else**:

a=list()

**for** i **in** range(len(nums)):

temp=self.per(nums[:i]+nums[i+1:])

**for** j **in** temp:

a.append([nums[i]]+j)

**return** a

**【提交记录截图】**



**【遇到的问题及解决办法】**

递归，对于n个数的全排列，即提取其中任意一个数在加上剩下n-1个数的每一种排列。

第一次没有考虑输入数组长度为1的情况。

**【对比学习心得】**

发现还可以用深搜做。

#### 【题目2】LeetCode47

给定一个可包含重复数字的序列，返回所有不重复的全排列。

**【代码】**

**class** Solution:

**def** permuteUnique(self, nums: list[int]) -> list[list[int]]:

self.len=len(nums)

**if** self.len==1:

**return** [nums]

self.list=list()

self.per(nums)

**return** self.list

**def** per(self,nums):

**if** len(nums)==1:

**return** [[nums[0]]]

**elif** self.len==len(nums):

s1=set()

**for** i **in** range(len(nums)):

oldlen=len(s1)

s1.add(nums[i])

newlen=len(s1)

**if** newlen>oldlen:

temp=self.per(nums[:i]+nums[i+1:])

**for** j **in** temp:

self.list.append([nums[i]]+j)

**else**:

a=list()

s2=set()

**for** i **in** range(len(nums)):

oldlen=len(s2)

s2.add(nums[i])

newlen=len(s2)

**if** newlen>oldlen:

temp=self.per(nums[:i]+nums[i+1:])

**for** j **in** temp:

a.append([nums[i]]+j)

**return** a

**【提交记录截图】**



**【遇到的问题及解决办法】**

在上一题的基础上加了一个判断条件，即提取出的数是不是已经被提取过了，同样是递归的方法。

**【对比学习心得】**

#### 【题目3】LeetCode215

在未排序的数组中找到第 **k** 个最大的元素。请注意，你需要找的是数组排序后的第 k 个最大的元素，而不是第 k 个不同的元素。

**【代码】**

**class** Solution:

**def** findKthLargest(self, nums, k) :

mid=nums[len(nums)//2]

nums.pop(len(nums)//2)

left=list()

middle=list()

right=list()

**for** item **in** nums:

**if** item<mid:

left.append(item)

**elif** item==mid:

middle.append(item)

**else**:

right.append(item)

**if** k>=len(right)+1 **and** k<=len(right)+len(middle)+1:

**return** mid

**elif** k<=len(right):

**return** self.findKthLargest(right,k)

**else**:

**return** self.findKthLargest(left,k-len(middle)-len(right))

a=Solution()

a.findKthLargest([-1,2,0],2)

**【提交记录截图】**



**【遇到的问题及解决办法】**

刚开始尝试用第二种方法：

**class** solution:

**def** findKthLargest(self, nums, k):

length=len(nums)

**if**(length<6):

nums.sort()

print(**'sort'**)

**return** nums[length-k]

l=length//5

j=0

li=list()

**for** i **in** range(l):

temp=list()

**for** i **in** range(5):

temp.append(nums[j])

j+=1

li.append(temp)

M=list()

**for** item **in** li:

item.sort()

M.append(item[2])

m=self.findKthLargest(M,l//2+1)

*# M.sort()*

*# m=M[int(l/2)]*

A=list()

S1=list()

S2=list()

**for** item **in** li:

**if** item[2]<m:

S1+=item[:3]

A+=item[3:]

**elif** item[2]>m:

S2+=item[2:]

A+=item[:2]

**else**:

S1+=item[:2]

S2+=item[3:]

**for** i **in** range(length%5):

temp=nums[5\*l+i]

**if**(temp<m):

S1.append(temp)

**else**:

S2.append(temp)

**if** k==len(S2)+1:

**return** m

**elif** k<len(S2):

**return** self.findKthLargest(S2,k)

**else**:

**return** self.findKthLargest(S1,k-len(S2)+1)

但是一直执行出错，然后就用了第一种。还需要慢慢排查。

**【对比学习心得】**

可以借助最大堆求解。

#### 【题目4】LeetCode4

给定两个大小为 m 和 n 的有序数组 nums1 和 nums2。

请你找出这两个有序数组的中位数，并且要求算法的时间复杂度为 O(log(m + n))。

你可以假设 nums1 和 nums2 不会同时为空。

**【代码】**

**import** sys

**class** Solution:

**def** findMedianSortedArrays(self, nums1, nums2):

len1=len(nums1)

len2=len(nums2)

length=len1+len2

**if** (length)%2!=0:

**return** self.findk(nums1,nums2,length//2+1)

**else**:

**return** (self.findk(nums1,nums2,length//2)+self.findk(nums1,nums2,length//2+1))/2

**def** findk(self, nums1, nums2,k):

**if** len(nums1)==0:**return** nums2[k-1]

**if** len(nums2)==0:**return** nums1[k-1]

**if** k==1:

**return** min(nums1[0],nums2[0])

mid1=sys.maxsize

mid2=sys.maxsize

**if** (k+1)//2-1<len(nums1):mid1=nums1[(k+1)//2-1]

**if** (k+1)//2-1<len(nums2):mid2=nums2[(k+1)//2-1]*#第一次错了，k没有加1*

**if** mid1==mid2:**return** mid1

**elif** mid1<mid2:

**return** self.findk(nums1[k//2:],nums2,k-k//2)

**else**:

**return** self.findk(nums1,nums2[k//2:],k-k//2)

**【提交记录截图】**



**【遇到的问题及解决办法】**

利用ppt上所给的方法，第一次判断条件有误，无限递归。第二次没有考虑好k为基数时的index。

**【对比学习心得】**

#### 【题目5】LeetCode23

合并 k 个排序链表，返回合并后的排序链表。请分析和描述算法的复杂度。

**【代码】**

**package** LeetCode;

**public class** leetcode23 {

**public** ListNode mergeKLists(ListNode[] lists) {

**int** len=lists.**length**;

**if**(lists==**null** || len==0)**return null**;

**return** fz(lists,0,len-1);

}

**public** ListNode fz(ListNode[] lists,**int** j,**int** k){

**if**(k>=j)**return** lists[j];

**int** mid=(k-j)/2+j;

ListNode l1=fz(lists,j,mid);

ListNode l2=fz(lists,mid+1,k);

**return** merge2Lists(l1,l2);

}

**public** ListNode merge2Lists(ListNode l1,ListNode l2){

**if**(l1==**null**)**return** l2;

**if**(l2==**null**)**return** l1;

ListNode p1=l1,p2=l2;

ListNode current=**new** ListNode(0);

ListNode first=current;

**while** (p1!=**null**&&p2!=**null**){

**if**(p1.**val**<=p2.**val**){

current.**next**=p1;

p1=p1.**next**;

current=current.**next**;

}

**else** {

current.**next**=p2;

p2=p2.**next**;

current=current.**next**;

}

}

ListNode p=(p1!=**null**)? p1:p2;

**while** (p!=**null**){

current.**next**=p;

p=p.**next**;

current=current.**next**;

}

**return** first.**next**;

}

}

**【提交记录截图】**



**【遇到的问题及解决办法】**

这道题上周就已经尝试过了，当时没有使用分治的方法，而是直接合并：

**public class** Solution23 {

**public** ListNode mergeKLists(ListNode[] lists) {

**int** n=lists.**length**;

**if**(n==0)**return null**;

ListNode temp;

**int** min=2^31-1;

**int** index=0;

**boolean** end=**true**;

**for** (**int** i=0;i<n;i++){

**if** (lists[i]==**null**)**continue**;

**if**(min>lists[i].**val**){

min=lists[i].**val**;

index=i;

}

end=**false**;

}

**if** (end)**return null**;

lists[index]=lists[index].**next**;

ListNode head=**new** ListNode(min);

temp=head;

**while** (**true**){

min=2^31-1;

index=0;

end=**true**;

**for** (**int** i=0;i<n;i++){

**if** (lists[i]==**null**)**continue**;

**if**(min>lists[i].**val**){

min=lists[i].**val**;

index=i;

}

end=**false**;

}

**if**(end)**break**;

**if**(lists[index]!=**null**)lists[index]=lists[index].**next**;

temp.**next**=**new** ListNode(min);temp=temp.**next**;

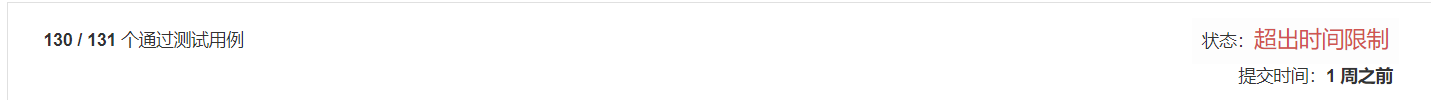
}

**return** head;

}

}

但是在输入的所有链表只有一个元素的时候，时间复杂度太大。



然后改用两两合并的方法，但是没有使用递归，而是循环合并，时间复杂度还是太高。

**【对比学习心得】**

#### 【题目6】LeetCode53

给定一个整数数组 nums ，找到一个具有最大和的连续子数组（子数组最少包含一个元素），返回其最大和。

**【代码】**

**class** Solution:

**def** maxSubArray(self, nums: list[int]) -> int:

length=len(nums)

dp=[0 **for** i **in** range(length)]

dp[0]=nums[0]

res=nums[0]

**for** i **in** range(length):

**if**(dp[i-1]<0):

**continue**

**else**:

dp[i]=nums[i]+dp[i-1]

res=max(res,dp[i])

**return** res

**【提交记录截图】**



**【遇到的问题及解决办法】**

刚开始想用分治的方法来做：

**class** Solution:

**def** maxSubArray(self, nums: List[int]) -> int:

length=len(nums)

**if** length==1:

**return** nums[0]

**else**:

**return** self.maxSub(nums,0,length-1)

**def** maxSub(self,nums,left,right):

**if**(left==right):

**return** nums[left]

mid=(right-left)//2+left

sumL=self.maxSub(nums,left,mid)

sumR=self.maxSub(nums,mid+1,right)

maxL=nums[mid]

temp=maxL

**for** i **in** range(mid):

temp+=nums[mid-i-1]

**if** temp>maxL:

maxL=temp

maxR=nums[mid+1]

temp=maxR

**for** i **in** range(mid+2,right+1):

temp+=nums[i]

**if** temp>maxR:

maxR=temp

sum=maxL+maxR

**if** sumL>sum:

sum=sumL

**if** sumR>sum:

sum=sumR

**return** sum

但是复杂度太高，看评论区都说用动态规划来做，所以简单的学习了一下dp，递推关系为f(n) = max(f(n-1) + A[n], A[n]);然后用res纪录一下结果即可。第一次错误为多算了一次首位数。

**【对比学习心得】**

#### 【题目7】LintCode1095

给定一个非负整数，你可以交换两个数位**至多**一次来获得最大的合法的数。返回最大的合法的你能够获得的数。

**【代码】**

**def** maximumSwap(num):

*# Write your code here*

s = str(num)

temp = s

a = 0

**while** temp != **''**:

index = 0

max=0

**for** i **in** range(len(temp)):

**if**(eval(temp[i])>=max):

max = eval(temp[i])

index=i

**if** eval(temp[0]) == max:

a += 1

temp = temp[1:]*#上一次错了，写反了*

**else**:

temp=temp[:index]+temp[0]+temp[index+1:]

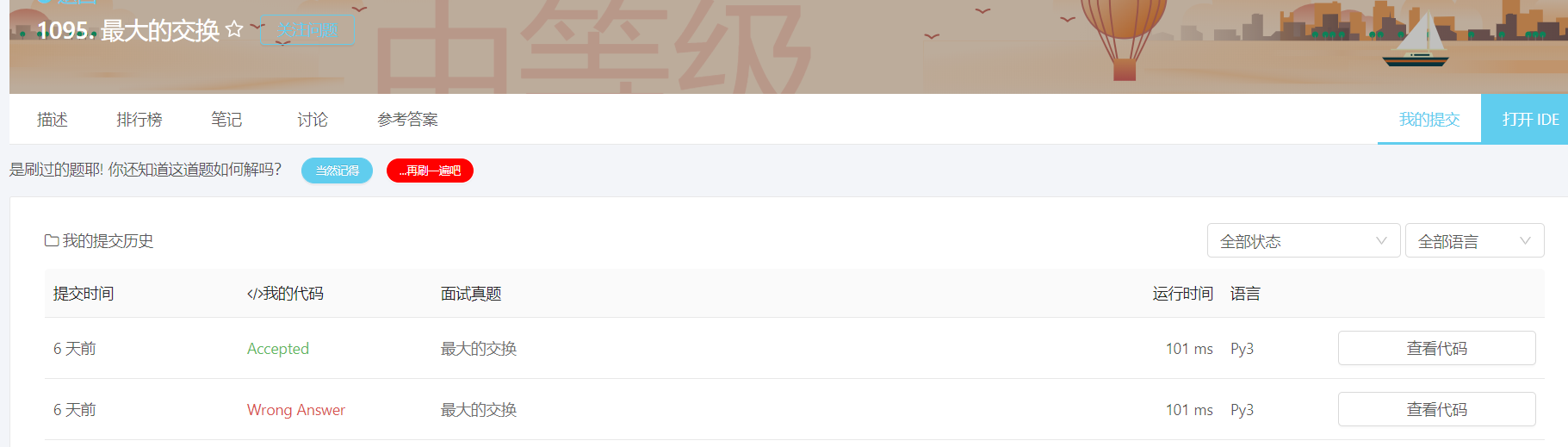
temp=str(max)+temp[1:]

res=s[0:a]+temp

**return** eval(res)

**return** eval(s)

**【提交记录截图】**



**【遇到的问题及解决办法】**

从首位开始查找最大值如果首位不是最大的则交换首位，首位是最大的，交换下一位，且要交换的数字是所有位数中值最大的，如果有相同值，取索引相对大的.第一次提交有些下标粗心了。

**【对比学习心得】**

#### 【题目8】LintCode1639

给定字符串S和整数K.

计算长度为K且包含K个不同字符的子串数

**【代码】**

**def** KSubstring(stringIn, K):

S=set()

l=len(stringIn)

**for** i **in** range(l-K+1):

temp=stringIn[i:i+K]

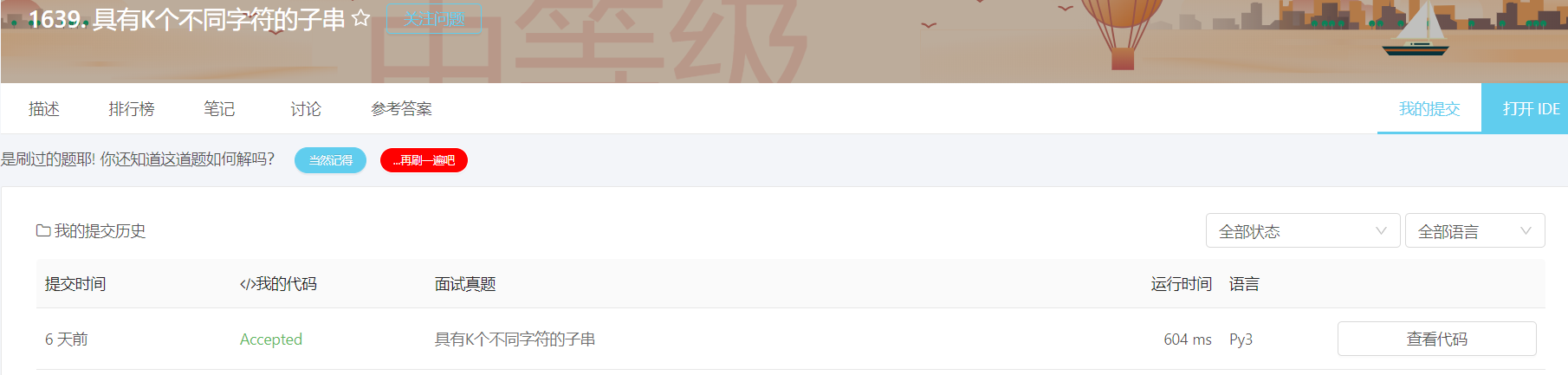
**if** len(temp)==len(set(temp)):

S.add(temp)

print(S)

**return** len(S)

**【提交记录截图】**



**【遇到的问题及解决办法】**

用python的set来做，很好写，遍历一遍字符串长度为K的子串，判断是否含重复字符，若没有则加入集合中。

**【对比学习心得】**

## 二、本周小结

巩固了上课所学的分治的算法，并初步了解了动态规划的思想。原来的思想有点固化，分治的算法不容易想到，希望通过刷题改进。

-------------------------------

成绩评定：

教师签名：

批改日期：