《面向对象程序设计与训练》实验报告

信息 学院 计算机科学与技术 专业 2019 级

实验时间 2020 年 11 月 6 日

姓名 学号 分工

姓名 学号 分工

姓名 学号 分工

姓名 学号 分工

姓名 学号 分工

实验名称 leetcode前十题

实验成绩

一、实验目的

在leetcode上进行算法题的训练

二、实验仪器设备及软件

Windows2000 或XP，JDK1.6与Jcreator4.0

三、实验方案

1.遍历两个数组，计算nums[i]+nums[j]，若等于target，记录i，j，返回结果，否则继续遍历

2.若l1非空则取出l1当前点的值，l2同理，设l1.val+l2.val+进位c的值为val，若val>=10，则val-=10，且进位c置一，将val插入新链表。当l1,l2均空时，若进位c不为零，那么需要在新链表中插入一个新节点来存储进位c

3.

1. 要求时间复杂度O(log(m+n))，第一时间想到二分查找的方法

每次取nums1的中间元素mid1与nums2的中间元素mid2，有下面三种情况。

mid1<mid2,这时最多有mid1-1+mid2-1个数比mid1小，并且mid1-1+mid2-1=（m+n）/2-2，所以nums1中mid1及前面的均不可能是中位数。

mid1>mid2，同上，nums2中mid2及前面的均不可能是中位数。

mid1=mid2，可以归到上面两种情况钟的任意一种

要注意，本题两输出元素总数有奇偶两种情况

5.设fij表示字符串i到j的子串是否为一个回文串，当s.charAt(i)==s.charAt(j))&&(L<=2||f[i+1][j-1])时，fij=true。特殊地字符串长度为1时，最长回文子串为他自身。在这个过程中记录最长回文子串地长度和开始位置，最后用substring方法即可找到该子串。

6.逐行整体观察，发现其由两段组成，一段是上下方向，一段是斜线方向。对于上下方向的，每一行相邻两个差2\*numRows-2个位置，对于斜线方向，满足i!=0 && i!=numRows-1 && (cnt+(numRows-1-i)\*2)<len时存在，且间隔cnt+(numRows-1-i)\*2，cnt为当前计算到的坐标。由上述计算方法计算即可得到答案

7. 通过逐位倒序ans=ans\*10+x%10;x/=10;得到逆序后的数值，这里ans为long类型防止溢出，可通过将ans强制转换为int类型，对比数值有没有发生改变来判断是否超过范围。

8.首先滤过空格，读到第一个非空格字符，若全空格或者第一个字符不是+-及数字（+：fuhao=1；-：fuhao=-1），那么返回0。如果+-之后的不是数字也返回0，将数字全部读完后计算出其值存在ans中，ans为long类型防溢出，将ans强制转换为int类型若数值发生改变说明超出边界。

9.求得该整数的倒序，与该整数比较，若相等则说明是回文数，否则不是

1. 用动态规划的方法，设f[i][j]表示s中前i位与p中前j位是否匹配
2. p(j-1)不为’\*’时，只要s(i-1)与p(j-1)相等或者p(j-1)为’.’时

有f[i][j]=f[i-1][j-1];

1. p(j-1)不为’\*’时，若s(i-1)与p(j-2)相等或者p(j-2)为’.’时

有f[i][j]=f[i-1][j] || f[i][j-2] || f[i][j-1];

否则，有f[i][j]=f[i][j-2];

（3）其他情况f[i][j]均为false，初始f[0][0]为true

四、实验步骤

1.

1. **class** Solution {
2. **public** **int**[] twoSum(**int**[] nums, **int** target) {
3. **int** len=nums.length;
4. **int**[] ans=**new** **int**[2];
5. **for**(**int** i=0;i<=len;i++){
6. **for**(**int** j=0;j<len;j++){
7. **if**(j!=i){
8. **if**(nums[i]+nums[j]==target){
9. ans[0]=i;
10. ans[1]=j;
11. **return** ans;
12. }
13. }
14. }
15. }
16. **return** ans;
17. }
18. }

2.

1. /\*\*
2. \* Definition for singly-linked list.
3. \* public class ListNode {
4. \*     int val;
5. \*     ListNode next;
6. \*     ListNode() {}
7. \*     ListNode(int val) { this.val = val; }
8. \*     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
9. \* }
10. \*/
11. **class** Solution {
12. **public** ListNode addTwoNumbers(ListNode l1, ListNode l2) {
13. ListNode L=**null**;
14. ListNode p;
15. p=L;
16. **int** c=0;//进位
17. **while**(l1!=**null** || l2!=**null**){
18. **int** val1=0,val2=0;
19. **if**(l1!=**null**){
20. val1=l1.val;
21. l1=l1.next;
22. }
23. **if**(l2!=**null**){
24. val2=l2.val;
25. l2=l2.next;
26. }
27. **int** val=val1+val2;
28. **if**(c>0){
29. val+=c;
30. c=0;
31. }
32. **if**(val>=10){
33. c=1;
34. val-=10;
35. }
36. **if**(L==**null**){
37. p=L=**new** ListNode(val);
38. }**else**{
39. p.next=**new** ListNode(val);
40. p=p.next;
41. }
43. }
44. **if**(c>0){
45. p.next=**new** ListNode(c);
46. }
47. **return** L;
48. }
49. }

3.

1. **class** Solution {
2. **public** **int** lengthOfLongestSubstring(String s) {
3. **int** len=s.length();
4. **int** i=0,j=0;
5. **int** ans=0;
6. **for**(;j<len;j++){
7. **for**(**int** k=i;k<j;k++){
8. **if**(s.charAt(j)==s.charAt(k)){
9. i=k+1;
10. **break**;
11. }
12. }
13. **if**(j-i+1>ans) ans=j-i+1;
14. }
15. **return** ans;
16. }
17. }

4.

1. **class** Solution {
2. **public** **int** findk(**int** nums1[],**int** nums2[],**int** k){
3. **int** start1=0,start2=0;
4. **int** len1=nums1.length,len2=nums2.length;
5. **while**(**true**){
6. **if**(start1==len1){
7. **return** nums2[start2+k-1];
8. }
9. **if**(start2==len2){
10. **return** nums1[start1+k-1];
11. }
12. **if**(k==1){
13. **return** nums1[start1]<nums2[start2]?nums1[start1]:nums2[start2];
14. }
15. **int** index1=len1<start1+k/2?len1-1:start1+k/2-1;
16. **int** index2=len2<start2+k/2?len2-1:start2+k/2-1;
17. **if**(nums1[index1]<=nums2[index2]){
18. k-=(index1-start1+1);
19. start1=index1+1;
20. }**else**{
21. k-=(index2-start2+1);
22. start2=index2+1;
23. }
24. }
25. }
26. **public** **double** findMedianSortedArrays(**int**[] nums1, **int**[] nums2) {
27. **int** len1=nums1.length,len2=nums2.length;
28. **int** len=len1+len2;
29. **double** ans=0.0;
30. **if**(len%2==1){
31. ans=findk(nums1,nums2,len/2+1);
32. }**else**{
33. ans=(findk(nums1,nums2,len/2)+findk(nums1,nums2,len/2+1))/2.0;
34. }
35. **return** ans;
36. }
37. }

5.

1. **class** Solution {
2. **public** String longestPalindrome(String s) {
3. **int** len=s.length();
4. **if**(len<=1) **return** s;
5. **int** max=0,l=0;
6. **boolean** f[][]=**new** **boolean**[len][len];
7. **for**(**int** i=len-1;i>=0;i--){
8. **for**(**int** j=i;j<len;j++){
9. **int** L=j-i+1;
10. f[i][j]=(s.charAt(i)==s.charAt(j))&&(L<=2||f[i+1][j-1]);
11. **if**(f[i][j]&& L>max){
12. max=L;
13. l=i;
14. }
15. }
16. }
17. **return** s.substring(l,l+max);
18. }
19. }

6.

1. **class** Solution {
2. **public** String convert(String s, **int** numRows) {
3. **int** len=s.length();
4. **if**(numRows==1) **return** s;
5. **int** ctr=numRows-2;
6. String ans="";
7. **int** jiange=numRows-1+ctr+1;
8. **int** cur=0;
9. **for**(**int** i=0;i<numRows;i++){
10. **int** cnt=i;
11. **while**(cnt<len){
12. ans+=s.charAt(cnt);
14. **if**(i!=0 && i!=numRows-1 && (cnt+(numRows-1-i)\*2)<len){
15. ans+=s.charAt(cnt+(numRows-1-i)\*2);
16. }
17. cnt+=jiange;
18. }
19. }
20. **return** ans;
21. }
22. }

7.

1. **class** Solution {
2. **public** **int** reverse(**int** x) {
3. **long** ans=0;
4. **while**(x!=0){
5. ans=ans\*10+x%10;
6. x/=10;
7. }
8. **return** (**int**)ans==ans?(**int**)ans:0;
9. }
10. }

8.

1. **class** Solution {
2. **public** **int** myAtoi(String s) {
3. **int** len=s.length();
4. **int** fuhao=1;
5. **long** ans=0;
6. **int** cnt=0;
7. **while**(cnt<len && s.charAt(cnt)==' '){
8. cnt++;
9. }
10. **if**(cnt>=len){
11. **return** 0;
12. }
13. **if**(s.charAt(cnt)=='-'){
14. fuhao=-1;
15. cnt++;
16. **if**(cnt>=len||s.charAt(cnt)<'0'||s.charAt(cnt)>'9')
17. **return** 0;
18. }**else** **if**(s.charAt(cnt)=='+'){
19. fuhao=1;
20. cnt++;
21. **if**(cnt>=len||s.charAt(cnt)<'0'||s.charAt(cnt)>'9')
22. **return** 0;
23. }**else** **if**(s.charAt(cnt)<'0'||s.charAt(cnt)>'9'){
24. **return** 0;
25. }
26. **while**(cnt<len && s.charAt(cnt)>='0'&&s.charAt(cnt)<='9'){
27. ans=ans\*10+s.charAt(cnt)-'0';
28. **if**((**int**)ans\*fuhao!=ans\*fuhao){
29. **if**(fuhao<0) **return** -2147483648;
30. **else** **return** 2147483647;
31. }
32. cnt++;
33. }
34. **return** (**int**)ans\*fuhao;
35. }
36. }

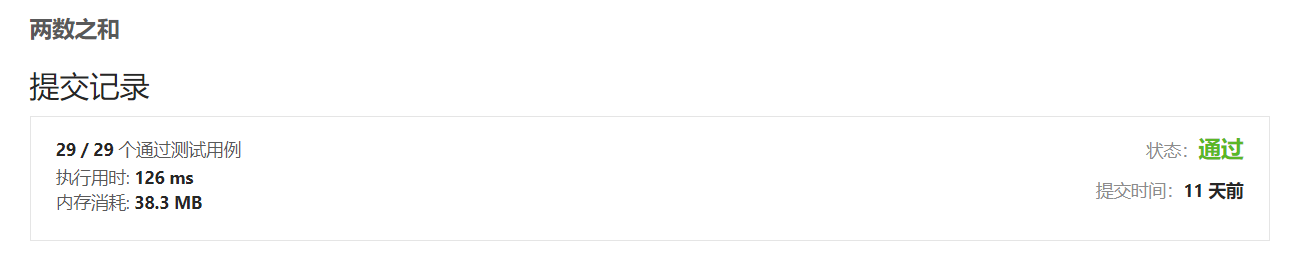
9.

1. **class** Solution {
2. **public** **boolean** isPalindrome(**int** x) {
3. **if**(x<0) **return** **false**;
4. **int** tmp=x,rx=0;
5. **while**(x!=0){
6. rx=rx\*10+x%10;
7. x/=10;
8. }
9. System.out.println(rx);
10. **if**(tmp==rx) **return** **true**;
11. **else** **return** **false**;
12. }
13. }

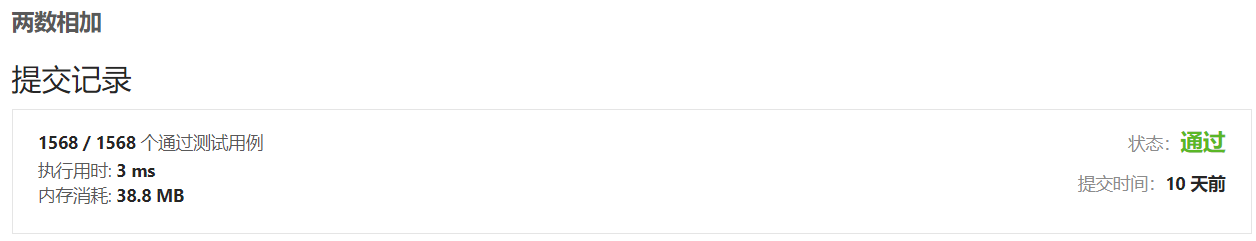
10.

1. **class** Solution {
2. **public** **boolean** isMatch(String s, String p) {
3. **boolean** f[][]=**new** **boolean**[21][31];
4. f[0][0]=**true**;
5. **int** lens=s.length(),lenp=p.length();
6. **for**(**int** i=0;i<=lens;i++){
7. **for**(**int** j=1;j<=lenp;j++){
8. **if**(p.charAt(j-1)!='\*'){
9. **if**(i>=1 && (s.charAt(i-1)==p.charAt(j-1) || p.charAt(j-1)=='.')){
10. f[i][j]=f[i-1][j-1];
11. }
12. }**else**{
13. **if**(i>=1 && (s.charAt(i-1)==p.charAt(j-2) || p.charAt(j-2)=='.')){
14. f[i][j]=f[i-1][j] || f[i][j-2] || f[i][j-1];
15. }**else**{
16. f[i][j]=f[i][j-2];
17. }
18. }
19. }
20. }
21. **return** f[lens][lenp];
22. }
23. }
24. 实验结果及分析

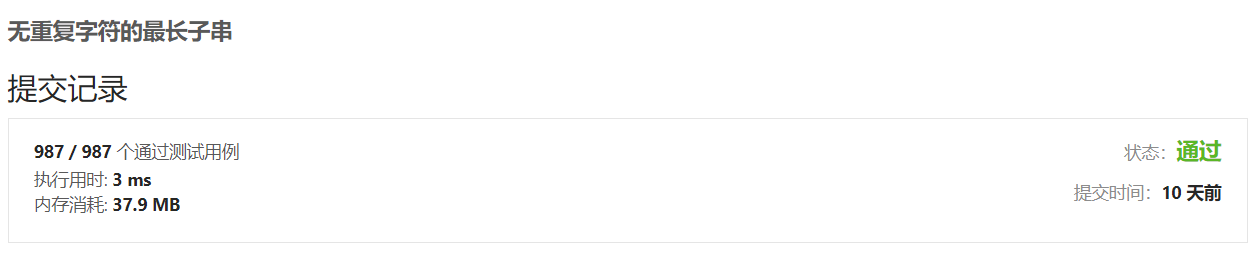
1.



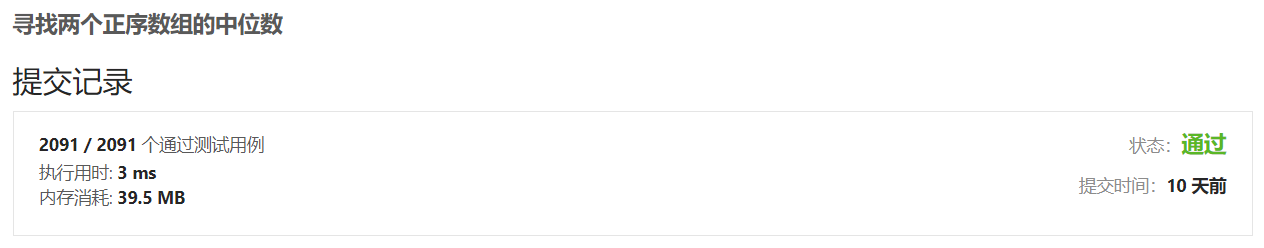
2.



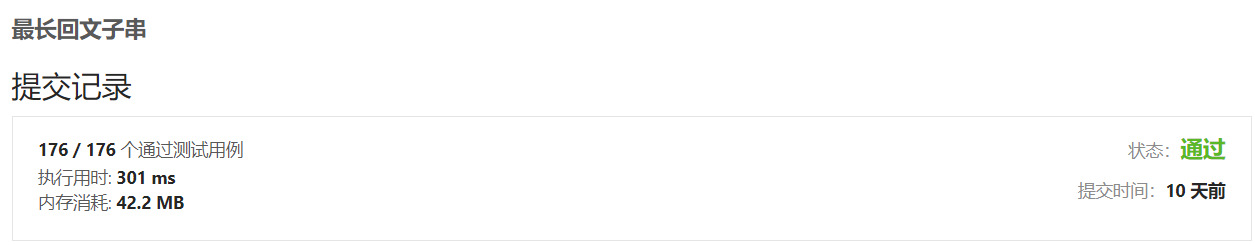
3.



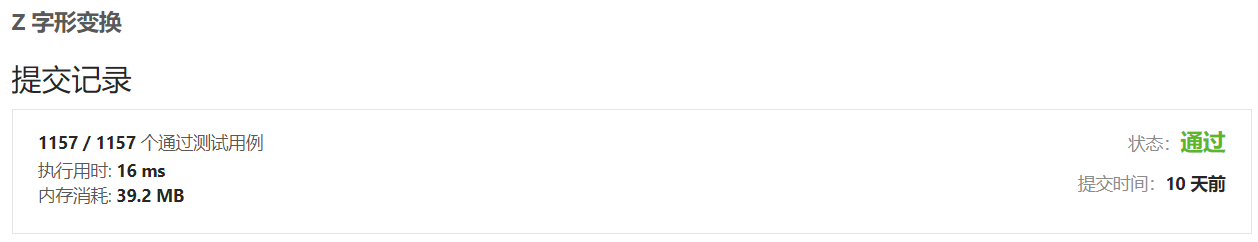
4.



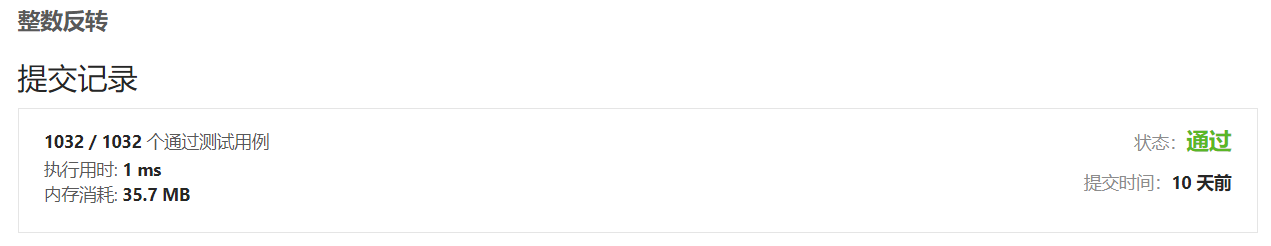
5.



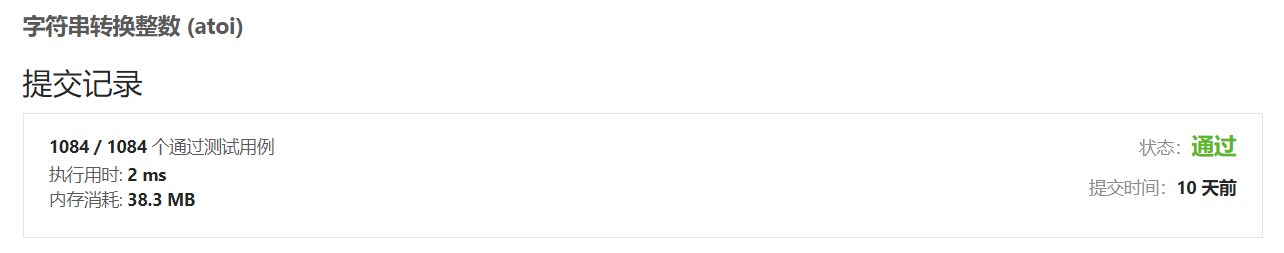
6.



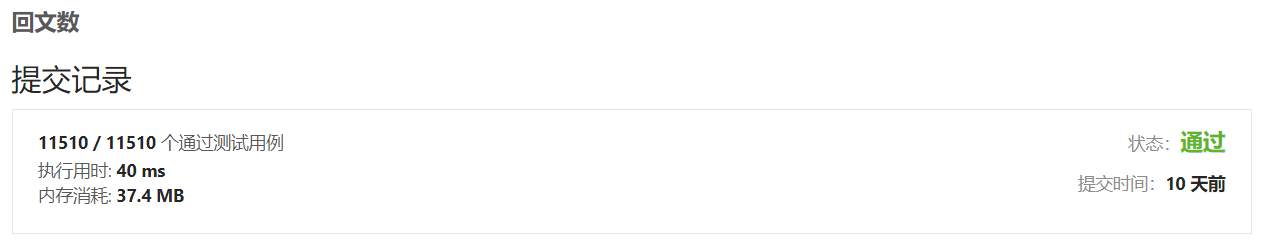
7.



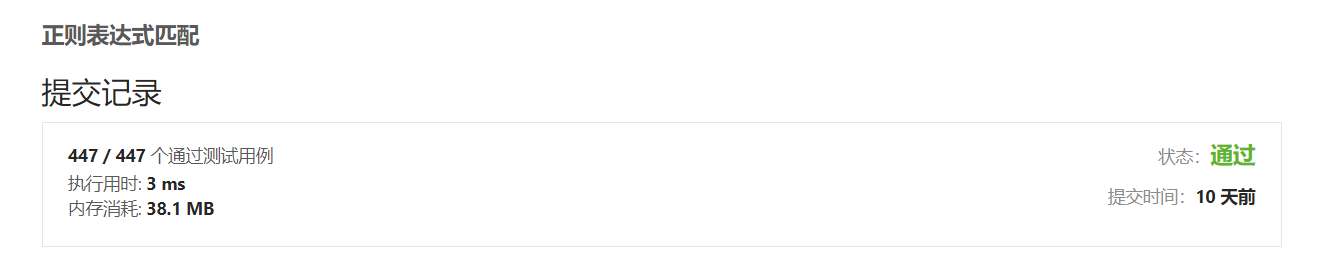
8.



9.



10.



六、实验总结及体会

(简单总结实验中所碰到的问题、解决方案、心得体会等，请删除此行)

七、教师评语

(中文字体宋体、英文字体Times New Roman，小四，1.5倍行距，首行缩进两个字符，请删除此行)