《面向对象程序设计与训练》实验报告

信息学院 学院 计算机科学与技术 专业 19 级

实验时间 2020 年 11 月 7 日

姓名 白文强 学号 20191060064 分工 1~10 ，整理实验报告、PPT

姓名 王迪 学号 20191060232 分工 1~10 ，整理实验报告

姓名 王童尧 学号 20191060050 分工 1~10

姓名 谭人玮 学号 20191060133 分工 1~10

实验名称 LeetCode 前十题

实验成绩

一、实验目的

磨炼算法技术、夯实 Java 基础

二、实验仪器设备及软件

个人 PC、LeetCode 在线网页

三、实验方案

第一题：采用双重循环暴力破解，在内层循环对遍历的部分进行优化，减少遍历次数。

第二题：类比 C 语言里面的链表操作，这里是对每一个节点实例化了一个对象。

第三题：从字符串的各位置出发往后找不重复的字符，遇到重复字符时结束，得到以该位置开始的无重复字串长度

第四题：归并排序，合并两个数组，使其保持有序，找出中位数即可。

第五题：对于一个子串而言，如果它是回文串，并且长度大于 2，那么将它首尾的两个字母去除之后，它仍然是个回文串。

第六题：使用布尔变量作为标志，分别表示向上和向下写入数据，最后按题意读取即可。

第七题：使用循环和取余进行逐位剥离并相乘相加，使用 long 类型存储，最后看结果转换成 int 类型是否丢失精度判断能否成功。

第八题：先判断第一位的符号，然后逐位转换成数字并相乘相加，直到后面可以被忽略。

第九题：首先负数不可能为回文数；对于正数采取类似第七题的做法，直接逆转数字并判断是否相等即可。

第十题：采用动态规划，我们每次从字符串 p 中取出一个字符或者字符+星号的组合，并在 s 中进行匹配

dp[i][j]表示 s[0:i]和 p[0:j]能否匹配

（1） p[j]是普通字符，则：

dp[i][j] = dp[i-1][j-1] && s[i]==p[j]

（2）p[j]是'.'，则：

dp[i][j] = dp[i-1][j-1]

（3）p[j]是'*'，则：

if s[i] == p[j-1]

dp[i][j] = dp[i][j-2] or dp[i-1][j]

if s[i] != p[j-1]

dp[i][j] = dp[i][j-2]

四、实验步骤

```
//  1、两数之和
import java.util.Arrays;
public class Solution {
    public int[] twoSum(int[] nums, int target) {
        for(int i=0; i<nums.length; i++){
            for(int j = i + 1 ; j < nums.length; j++ ){
                if( nums[i] + nums[j] == target ){
                    return new int []{i, j};
                }
            }
        }
        return new int[0];
    }
}
```

```java
//  2、两数相加
class Solution {
    public ListNode addTwoNumbers(ListNode l1, ListNode l2) {
        ListNode head = new ListNode(0), cur = head;
        int isBigThanTen = 0;               //是否大于 10

        while(l1 != null || l2 != null){
            int n1 = l1 == null ? 0 : l1.val;
            int n2 = l2 == null ? 0 : l2.val;
            int num = n1 + n2 + isBigThanTen;
            isBigThanTen = num/10;
            cur.next = new ListNode(num % 10);
            cur = cur.next;
            if(l1 != null){
                l1 = l1.next;
            }
            if(l2 != null){
                l2 = l2.next;
            }
        }
        if(isBigThanTen > 0){
            cur.next = new ListNode(1);
        }
        return head.next;
    }
}
```

```java
//  3、无重复字符的最长子串
class Solution {
    public int lengthOfLongestSubstring(String s) {
        int i = 0, j = 0;
        int maxLength = 0;
        while(j < s.length()){
            for(int k = i; k < j; k++)
            {
                if(s.charAt(k) == s.charAt(j)){
                    i = k+1;
                    break;
                }
            }
            if( j-i+1 > maxLength ){
                maxLength = j-i+1;
            }
            j++;
        }
        return maxLength;
    }
}
```

```java
//  4、寻找两个正序数组的中位数
class Solution {
    public double findMedianSortedArrays(int[] nums1, int[] nums2) {
        int length = nums1.length + nums2.length;
        int[] arr = new int[length];
        int i = 0, j = 0, k = 0;
        while (k < length) {
            if (i < nums1.length && j < nums2.length && nums1[i] <= nums2[j]) {
                arr[k++] = nums1[i++];
                if (k == length / 2 +1) {
                    break;
                }
            }
            if (i < nums1.length && j < nums2.length && nums1[i] > nums2[j]) {
                arr[k++] = nums2[j++];
                if (k == length / 2 +1) {
                    break;
                }
            }
            if (i >= nums1.length && j < nums2.length) {
                arr[k++] = nums2[j++];
                if (k == length / 2 + 1) {
                    break;
                }
            }
            if (i < nums1.length && j >= nums2.length) {
                arr[k++] = nums1[i++];
                if (k == length / 2 +1) {
                    break;
                }
            }
        }
        if (length % 2 == 0) {
            return (arr[k - 2] + arr[k-1]) / 2.0;
        } else {
            return (double) arr[k-1];
        }
    }
}
```

```java
//  5、最长回文子串
class Solution {
    public String longestPalindrome(String s) {
        int n = s.length();

        boolean [][]dp = new boolean [n][n];
        for(int i = 0; i < n; i++){
            dp[i][i] = true;
        }
        String ans = s.substring(0,1);
        for(int j = 1; j < n; j++){
            for(int i = 0; i < j; i++){
                if( s.charAt(i) != s.charAt(j) ){
                    dp[i][j] = false;
                }else{
                    if( j - i + 1 < 3 ){
                        dp[i][j] = true;
                    }else{
                        dp[i][j] = dp[i+1][j-1] && s.charAt(i) == s.charAt(j);
                    }
                }
                if( dp[i][j] == true && j - i + 1 > ans.length() ){
                    ans = s.substring(i,j+1);
                }
            }
        }
        return ans;
    }
}
```

```java
//  6、Z 字形变换
class Solution {
    public String convert(String s, int numRows) {
        if(numRows == 1){
            return s;
        }
        char result[][] = new char[numRows][s.length()];
        int N[] = new int[numRows];

        int num = 0;
        boolean upOrDown = false;            //false 向下，true 向上。
        for(int i = 0; i < s.length(); i++){
            if(upOrDown == false){
                result[num][N[num]++] = s.charAt(i);
                num++;
            }else{
                result[num][N[num]++] = s.charAt(i);
                num--;
            }
            if(num == numRows-1 || num == 0){
                upOrDown = !upOrDown;
            }
        }
        String S = "";
        for(int i = 0; i < numRows; i++){
            S += (String.valueOf(result[i])).substring(0,N[i]);
        }
        return S;
    }
}
```

```java
//  7、整数反转
class Solution {
    public int reverse(int x) {
        long result = 0;
        while(x != 0){
            result = result * 10 + x % 10;
            x = x/10;
        }
        return (int)result == result ? (int)result : 0;
    }
}
```

```java
//  8、字符串转整函数
class Solution {
    public int myAtoi(String s) {
        s = s.trim();
        int number = 0;
        boolean isBigThanZero = true;
        for( int i = 0; i < s.length(); i++ ){
            if( i == 0 ){
                if( s.charAt(i) == '-' ){
                    isBigThanZero = false;
                    continue;
                }else if(s.charAt(i) == '+'){
                    isBigThanZero = true;
                    continue;
                }else if(Character.isDigit(s.charAt(i))){
                    isBigThanZero = true;
                }else{
                    return 0;
                }
            }

            if( s.charAt(i) >= '0' && s.charAt(i) <= '9' ){
                if( number > (Integer.MAX_VALUE - (s.charAt(i)-'0')) / 10 ){
                    return isBigThanZero ? Integer.MAX_VALUE : Integer.MIN_VALUE;
                }
                number = number * 10 + (s.charAt(i) - '0');
            }else{
                break;
            }
        }
        return isBigThanZero? number : -number;
    }
}
```

```java
//  9、回文数
class Solution {
    public boolean isPalindrome(int x) {
        if(x < 0){
            return false;
        }
        int origin = x;
        int y = 0, tem=0 ;
        while( origin != 0 ){
            tem = origin % 10; // 余数
            y = y*10+tem;       //进位
            origin /= 10;       //去掉最低位
        }
        return y == x;
    }
}
```

```java
//  10、正则表达式匹配
class Solution {
    public boolean isMatch(String s, String p) {
        int m = s.length();
        int n = p.length();
        boolean[][] dp = new boolean [m+1][n+1];
        dp[0][0] = true;
        for(int i = 0; i <= m; i++){
            for(int j = 1; j <= n; j++){
                if( p.charAt(j-1) == '*' ){
                    dp[i][j] = dp[i][j - 2];
                    if( ismatch(s,p,i,j-1) )
                        dp[i][j] = dp[i][j] || dp[i-1][j];
                }else{
                    if( ismatch(s,p,i,j) )
                        dp[i][j] = dp[i-1][j-1];
                }
            }
        }
        return dp[m][n];
    }
    public boolean ismatch(String s, String p, int i, int  j){
        if(i == 0)
            return false;
        if( p.charAt(j-1) == '.' )
            return true;
        return s.charAt(i-1) == p.charAt(j-1);
    }
}
```

五、实验结果及分析

✔	1	两数之和	7879	49.3%	简单	🔒
✔	2	两数相加	4193	38.9%	中等	🔒
✔	3	无重复字符的最长子串	4148	35.9%	中等	🔒
✔	4	寻找两个正序数组的中位数	1978	39.1%	困难	🔒
✔	5	最长回文子串	2164	32.4%	中等	🔒
✔	6	Z 字形变换	1419	49.0%	中等	🔒
✔	7	整数反转	3129	34.8%	简单	🔒
✔	8	字符串转换整数 (atoi)	1761	21.0%	中等	🔒
✔	9	回文数	2968	58.5%	简单	🔒
✔	10	正则表达式匹配	722	30.4%	困难	🔒
✔	11	盛最多水的容器	1744	64.4%	中等	🔒
✔	12	整数转罗马数字	908	64.5%	中等	🔒
✔	13	罗马数字转整数	2234	62.1%	简单	🔒
✔	14	最长公共前缀	2323	38.9%	简单	🔒

六、实验总结及体会

实践是掌握一种知识最好的方式，通过这些题目，对动态规划等算法有了更好的理解，写完题目后通过查看题解，了解自己设计的算法存在的不足，进而对自己的算法进行改进，在实践中学习，在实践中促进自己的学习，二者相辅相成。

七、教师评语