2. Add two Number

public ListNode addTwoNumbers(ListNode l1, ListNode l2) {

//创建一个头结点

ListNode prev= new ListNode(0);

//进位计数器

int carry=0;

ListNode head=prev;

while(l1!=null||l2!=null||carry!=0)

{ //  计算当前位置的值

int sum=((l1==null)?0:l1.val)+((l2==null)?0:l2.val)+carry;

//计算有无进位

carry=sum/10;

//创建prev下一个结点，val=sum%10,并将prev指针指向这一节点

prev.next=new ListNode(sum%10);

//移动指针到prev的下一结点

prev=prev.next;

//移动 l1,l2到下一个节点

l1=(l1==null)?l1:l1.next;

l2=(l2==null)?l2:l2.next;

}

//返回头结点

return head.next;

}

3.Longest Substring without Repeating Characaters

Use HashMap:

public int lengthOfLongestSubstring(String s) {

int n=s.length();

int ans=0;

Map<Character,Integer> map=new HashMap<>();

for (int j=1,i=0;j<=n;j++)

{

//如果当前位置字母已存在于Map中

if(map.containsKey(s.charAt(j-1)))

{

i=Math.max(i,map.get(s.charAt(j-1)));

}

//计算先前这个字母出现位置和现在这个字母位置距离,和之前存放的最大值比较取大值

ans=Math.max(ans,j-i);

map.put(s.charAt(j-1),j);

}

return ans;

}

4. **Longest Palindromic Substring** 最长回文子字符串

public String longestPalindrome(String s) {

int length = s.length();

boolean[][] P = new boolean[length][length];

int maxLen = 0;

String maxPal = "";

for (int len = 1; len <= length; len++) //遍历所有的长度

for (int start = 0; start < length; start++) {

int end = start + len - 1;

if (end >= length) //下标已经越界，结束本次循环

break;

P[start][end] = (len == 1 || len == 2 || P[start + 1][end - 1]) && s.charAt(start) == s.charAt(end); //长度为 1 和 2 的单独判断下

if (P[start][end] && len > maxLen) {

maxPal = s.substring(start, end + 1);

}

}

return maxPal;

}

5. ZigZag Conversion

public String convert(String s, int numRows) {

char[] c= s.toCharArray();

int len=c.length;

StringBuilder[] sb=new StringBuilder[numRows];

for(int i=0;i<sb.length;i++)

{sb[i]=new StringBuilder();}

int i=0;

while(i<len)

{

for(int index=0;index<numRows&&i<len;index++)

sb[index].append(c[i++]);

for(int index=numRows-2;index>=1&&i<len;index--)

sb[index].append(c[i++]);

}

for(int idx=1;idx<sb.length;idx++)

{

sb[0].append(sb[idx]);

}

return sb[0].toString();

}

6.Reverse Integer

class Solution {

public int reverse(int x) {

int rev=0;

while(x!=0)

{

int newrev=x%10+rev\*10;

if((newrev-x%10)/10 != rev) return 0;

rev=newrev;

x/=10;

}

return rev;

}

}

7. String to Integer

class Solution {

public int myAtoi(String str) {

int sign = 1;

int ans = 0, pop = 0;

boolean hasSign = false; //代表是否开始转换数字

for (int i = 0; i < str.length(); i++) {

if (str.charAt(i) == '-' && !hasSign) {

sign = -1;

hasSign = true;

continue;

}

if (str.charAt(i) == '+' && !hasSign) {

sign = 1;

hasSign = true;

continue;

}

if (str.charAt(i) == ' ' && !hasSign) {

continue;

}

if (str.charAt(i) >= '0' && str.charAt(i) <= '9') {

hasSign = true;

pop = str.charAt(i) - '0';

//和上道题判断出界一个意思只不过记得乘上 sign 。

if (ans \* sign > Integer.MAX\_VALUE / 10 || (ans \* sign == Integer.MAX\_VALUE / 10 && pop \* sign > 7))

return 2147483647;

if (ans \* sign < Integer.MIN\_VALUE / 10 || (ans \* sign == Integer.MIN\_VALUE / 10 && pop \* sign < -8))

return -2147483648;

ans = ans \* 10 + pop;

} else {

return ans \* sign;

}

}

return ans \* sign;

}

}

8.Palindrome Number

class Solution {

public boolean isPalindrome(int x) {

if(x<0) return false;

int div=1; int num=x;

while(num/div>=10)

{

div\*=10;

}

while(num!=0)

{

int left=num/div;

int right=num%10;

if(left!=right) return false;

num=num%div/10;

div/=100;

}

return true;

}

}

9. Regular Expression Matching

public boolean isMatch(String text, String pattern) {

if (pattern.isEmpty()) return text.isEmpty();

boolean first\_match = (!text.isEmpty() &&

(pattern.charAt(0) == text.charAt(0) || pattern.charAt(0) == '.'));

//只有长度大于 2 的时候，才考虑 \*

if (pattern.length() >= 2 && pattern.charAt(1) == '\*'){

//两种情况

//pattern 直接跳过两个字符。表示 \* 前边的字符出现 0 次

//pattern 不变，例如 text = aa ，pattern = a\*，第一个 a 匹配，然后 text 的第二个 a 接着和 pattern 的第一个 a 进行匹配。表示 \* 用前一个字符替代。

return (isMatch(text, pattern.substring(2)) ||

(first\_match && isMatch(text.substring(1), pattern)));

} else {

return first\_match && isMatch(text.substring(1), pattern.substring(1));

}

}

10 Most Water

public int maxArea(int[] height) {

if (height==null||height.length<2) return 0;

int area=0;

int left=0;

int right=height.length-1;

while(left<right)

{

area=Math.max(area,(right-left)\*Math.min(height[left],height[right]));

if(height[left]>height[right]) right--;

else{

left++;

}

}

return area;

}