**1. Two Sum**

**class Solution {**

**public int[] twoSum(int[] nums, int target) {**

**int[] number= new int[2];**

**label: for (int i=0;i<nums.length-1;i++){**

**for (int j=i+1;j<=nums.length;j++){**

**if(nums[i]+nums[j]==target){**

**number[0]=i;**

**number[1]=j;**

**break label;**

**}**

**}**

**}**

**return number;**

**}**

**}**

**2. Add Two Numbers**

**class Solution {**

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

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

**ListNode i=l1,j=l2,sum=n;**

**int plus=0;**

**int value=0;**

**while(i!=null || j!=null){**

**int x=(i!=null)? i.val : 0;**

**int y=(j!=null)? j.val : 0;**

**if(x+y>=10){**

**value=(x+y)%10+plus;**

**plus=1;**

**}else{**

**value=(x+y)%10+plus;**

**plus=0;**

**}**

**sum.next=new ListNode(value);**

**sum=sum.next;**

**i=i.next;**

**j=j.next;**

**}**

**return n.next;**

**}**

**}**

**4. Median of Two Sorted Arrays**

**class Solution {**

**public double findMedianSortedArrays(int[] nums1, int[] nums2) {**

**int[] temp=new int[100];**

**double median=0.0;**

**double n1median=0.0;**

**double n2median=0.0;**

**int i=0,j=0;**

**if(nums1==null){**

**n1median=0.0;**

**n2median=(nums2[nums2.length/2]+nums2[(nums2.length/2)+1])/2;**

**median=(n1median+n2median)/2;**

**//return median;**

**}else if(nums2==null){**

**n2median=0.0;**

**n1median=(nums1[nums1.length/2]+nums1[(nums1.length/2)+1])/2;**

**median=(n1median+n2median)/2;**

**//return median;**

**}else{**

**while(i<nums1.length && j<nums2.length){**

**int n=0;**

**temp[n]=(nums1[i]>=nums2[j])? nums1[i]:nums2[j];**

**n++;**

**if(nums1[i]>=nums2[j])**

**i++;**

**else**

**j++;**

**median=(temp[temp.length/2]+temp[(temp.length/2)+1])/2;**

**//return median;**

**}**

**}**

**return median;**

**}**

**}**

**7. Reverse Integer**

**class Solution {**

**public int reverse(int x) {**

**int sign = 1;**

**int ans = 0;**

**if (x == 0)**

**return ans;**

**sign = x > 0 ? 1 : -1;**

**x \*= sign;**

**if (x > (Math.pow(2,31)-1) || x < Math.pow(2,-31)){**

**return ans;**

**}**

**while (x != 0){**

**ans = (ans \* 10) + (x % 10);**

**x /= 10;**

**}**

**ans \*= sign;**

**return ans;**

**}**

**}**

**8. String to Integer (atoi)**



**9. Palindrome Number**

**class Solution {**

**public boolean isPalindrome(int x) {**

**int temp = 0;**

**int tempx = x;**

**if(x <= 0 ){**

**return false;**

**}**

**while(tempx > 0){**

**temp = temp \* 10 + (tempx % 10);**

**tempx /= 10;**

**}**

**return x==temp;**

**}**

**}**

**11. Container With Most Water**

**class Solution {**

**public int maxArea(int[] height) {**

**int maxarea = 0;**

**int subarea = 0;**

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

**for(int j = i+1; j < height.length; j++){**

**int hei = Math.min(height[i],height[j]);**

**int width = j-i;**

**subarea = hei \* width;**

**maxarea = Math.max(maxarea,subarea);**

**}**

**}**

**return maxarea;**

**}**

**}**

**12. Integer to Roman**

**class Solution {**

**public String intToRoman(int num) {**

**String[] key = {"M","CM","D","CD","C","XC","L","XL","X","IX","V","IV","I"};**

**int[] value = {1000,900,500,400,100,90,50,40,10,9,5,4,1};**

**String output = "";**

**int i = 0;**

**while(num > 0){**

**if( num - value[i] < 0){**

**i++;**

**}else{**

**num -= value[i];**

**output += key[i];**

**}**

**}**

**return output;**

**}**

**}**

**13. Roman to Integer**

**class Solution {**

**public int romanToInt(String s) {**

**Map<Character , Integer> map = Map.of('I', 1, 'V', 5, 'X', 10, 'L', 50, 'C', 100, 'D', 500, 'M', 1000);**

**int ans = 0;**

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

**if(i == 0){**

**ans += map.get(s.charAt(i));**

**}else if(map.get(s.charAt(i)) > map.get(s.charAt(i-1))){**

**ans += map.get(s.charAt(i)) - 2 \* map.get(s.charAt(i-1));**

**}else{**

**ans += map.get(s.charAt(i));**

**}**

**}**

**return ans;**

**}**

**}**

**15. 3Sum**

**class Solution {**

**public List<List<Integer>> threeSum(int[] nums) {**

**List<List<Integer>> temp = new ArrayList<Integer>();**

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

**for(int j=0;j<nums.length;j++){**

**for(int k=0;k<nums.length;k++){**

**if(i!=j && j!=k && i!=k){**

**if(nums[i]+nums[j]+nums[k]==0){**

**temp.add(nums[i]);**

**temp.add(nums[j]);**

**temp.add(nums[k]);**

**}**

**break;**

**}**

**continue;**

**}**

**}**

**}**

**return temp;**

**}**

**}**

**16. 3Sum Closest**

**import java.util.Arrays;**

**class Solution {**

**public int threeSumClosest(int[] nums, int target) {**

**Arrays.sort(nums);**

**int sum = 0, left,right;**

**int min = Integer.MAX\_VALUE;**

**for(int i = 1; i < nums.length; i++){**

**left = i -1;**

**right = i =1;**

**sum = nums[i] + nums[left] + nums[right];**

**if(sum == target){**

**return target;**

**}else if(sum > target){**

**min = min > (sum - target)? sum - target : min;**

**right--;**

**}else if(sum < target){**

**min = min > Math.abs(sum - target)? sum - target : min;**

**left++;**

**}**

**}**

**return target + min;**

**}**

**}**

**17. Letter Combinations of a Phone Number**

**18. 4Sum**

**class Solution {**

**public List<List<Integer>> fourSum(int[] nums, int target) {**

**Arrays.sort(nums);**

**Set<List<Integer>> set = new HashSet<>();**

**for(int i = 0; i < nums.length-3; i++){**

**for(int j = i+1; j< nums.length-2; j++){**

**for(int k = j+1; k <nums.length-1; k++){**

**for(int n = k+1; n < nums.length; n++){**

**if (nums[i] + nums[j] + nums[k] + nums[n] == target){**

**set.add(Arrays.asList(nums[i], nums[j], nums[k], nums[n]));**

**}**

**}**

**}**

**}**

**}**

**List<List<Integer>> ans = new ArrayList<>();**

**for (List<Integer> summation : set)**

**ans.add(summation);**

**return ans;**

**}**

**}**

**19. Remove Nth Node From End of List**

**class Solution {**

**public ListNode removeNthFromEnd(ListNode head, int n) {**

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

**ListNode slow = start, fast = start;**

**start.next = head;**

**for(int i = 1; i <= n+1; i++){**

**fast = fast.next;**

**}**

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

**slow = slow.next;**

**fast = fast.next;**

**}**

**slow.next = slow.next.next;**

**return start.next;**

**}**

**}**

**20. Valid Parentheses**

**class Solution {**

**public boolean isValid(String s) {**

**boolean ans = false;**

**Map<String, Integer> map = Map.of("(", 1, "）", 2, "[", 3, "]", 4, "{", 5, "}", 6);**

**int counter = 0;**

**// int p1 = 0; p2 = 0; p3 = 0;**

**if(s.length() < 1) return false;**

**for(int i = 0; i < s.length()-1; ){**

**if(map.get(s[i+1])-map.get(s[i]) != 1){**

**int right = map.get(s[i]);**

**i++;**

**if(map.get(s[i]) % 2 ==1){**

**while(map.get(s[i]) != right ){**

**if(map.get(s[i+1])-map.get(s[i]) == 1){**

**i += 2;**

**}else{**

**return false;**

**}**

**}**

**}else{ return false;}**

**}else{**

**i += 2;**

**ans = true;**

**}**

**}**

**return ans;**

**}**

**}**

**21. Merge Two Sorted Lists**

**class Solution {**

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

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

**ListNode ans = dummy;**

**while(l1 != null && l2 != null){**

**if(l1.val <= l2.val){**

**dummy.next = l1;**

**l1 = l1.next;**

**}else if(l1.val > l2.val){**

**dummy.next = l2;**

**l2 = l2.next;**

**}**

**dummy = dummy.next;**

**}**

**if(l1 == null) dummy.next = l2;**

**else if(l2 == null) dummy.next = l1;**

**return ans.next;**

**}**

**}**

**22. Generate Parentheses**

**24. Swap Nodes in Pairs(ListNoded的结构？递归的过程？)**

**26. Remove Duplicates from Sorted Array**

class Solution {

public int removeDuplicates(int[] nums) {

int n = 1;

for(int i = 1; i <= nums.length - 1; i++){

if(nums[i] != nums[i - 1]){

nums[n++] = nums[i];

}

}

return n;

}

}

**27. Remove Element**

class Solution {

public int removeElement(int[] A, int elem) {

int m = 0;

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

if(A[i] != elem){

A[m++] = A[i];

}

}

return m;

}

}

**28. Implement strStr()**

**class Solution {**

**public int strStr(String haystack, String needle) {**

**int l1 = haystack.length();**

**int l2 = needle.length();**

**if( l1 < l2) return -1;**

**if( l1 == 0 || l2 == 0) return 0;**

**for(int i =0; i < l1 - l2; i++){**

**if(haystack.substring(i, i+l2).equals(needle))**

**return i;**

**}**

**return -1;**

**}**

**}**

**29. Divide Two Integers**

**32. Longest Valid Parentheses**

**33. Search in Rotated Sorted Array**

class Solution {

public int search(int[] nums, int target) {

int res = -1, j = nums.length -1, pivot = nums[j] + 1;

if(pivot == target) return res;

if(target < pivot) res = nums.length >= (pivot - target)? nums.length - (pivot - target) : -1;

else res = target - (pivot + 1) >= 0? target - (pivot + 1) : -1;

return res;

}

}

**34. Find First and Last Position of Element in Sorted Array**

**35. Search Insert Position**

class Solution {

public int searchInsert(int[] nums, int target) {

int start = 0, end = nums.length -1;

if(target == 0 || target < nums[0]) return 0;

if(target > nums[end]) return end + 1;

while (start <= end){

int mid = (start + end) / 2;

if(nums[mid] == target) return mid;

else if(target < nums[mid])

end = mid - 1;

else start = mid + 1;

}

return start;

}

}

**36. Valid Sudoku**

class Solution {

public boolean isValidSudoku(char[][] board) {

for(int i = 0; i < 9; i++){

for(int j = 0; j < 9; j++){

if(board[i][j] == '.') continue;

int n = 0;

while(n < 9){

if (n == j || n == i){

n++;

continue;

}

if (board[i][j] == board[i][n] || board[i][j] == board[n][j]) return false;

else n++;

}

for(int a = i / 3 \* 3, row = a + 3; a < row; a++){

for(int b = j / 3 \* 3, col = b + 3; b < col; b++){

if ( a == i && b == j) continue;

if (board[i][j] == board[a][b]) return false;

}

}

}

}

return true;

}

}