Leetcode Problems DSA :

STACK:

150. Evaluate Reverse Polish Notation

import java.util.Stack;

public class Solution {

public int evalRPN(String[] tokens) {

int a,b;

Stack<Integer> S = new Stack<Integer>();

for (String s : tokens) {

if(s.equals("+")) {

S.add(S.pop()+S.pop());

}

else if(s.equals("/")) {

b = S.pop();

a = S.pop();

S.add(a / b);

}

else if(s.equals("\*")) {

S.add(S.pop() \* S.pop());

}

else if(s.equals("-")) {

b = S.pop();

a = S.pop();

S.add(a - b);

}

else {

S.add(Integer.parseInt(s));

}

}

return S.pop();

}

}

84. Largest Rectangle in Histogram

class Solution {

    public int largestRectangleArea(int[] arr) {

        int area = 0;

        Stack<Integer> st = new Stack<>();

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

            while(!st.isEmpty() && arr[i]<arr[st.peek()]){

                int h = arr[st.pop()];

                int r = i;

                if(st.isEmpty()){

                    area = Math.max(area, h\*r);

                }else{

                    int l = st.peek();

                    area = Math.max(area, h\*(r-l-1));

                }

            }

            st.push(i);

        }

            int r = arr.length;

            while(!st.isEmpty()){

                int h = arr[st.pop()];

                if(st.isEmpty()){

                    area = Math.max(area, h\*r);

                }else{

                    int l = st.peek();

                    area = Math.max(area, h\*(r-l-1));

                }

        }

        return area;

    }

}

LINKEDLIST:

25. Reverse Nodes in k-Group

class Solution {

public ListNode reverseKGroup(ListNode head, int k) {

ListNode t = head;

ListNode curr = head;

ListNode prev = null;

int cnt = 0;

while(t != null && cnt < k)

{

t = t.next;

cnt++;

}

if(cnt < k)

{

return head;

}

for(int i=k; i>0; i--)

{

t = curr.next;

curr.next = prev;

prev = curr;

curr = t;

}

head.next = reverseKGroup(curr,k);

return prev;

}

}

328. Odd Even Linked List

class Solution {

    public ListNode oddEvenList(ListNode head) {

          if (head == null || head.next == null) return head;

        ListNode odd = head;

        ListNode even = head.next;

        ListNode evenHead = even;

        while (even != null && even.next != null) {

            odd.next = even.next;

            odd = odd.next;

            even.next = odd.next;

            even = even.next;

        }

        odd.next = evenHead;

        return head;

    }

}

3507. Minimum Pair Removal to Sort Array I

class Solution {

public int minimumPairRemoval(int[] nums) {

int length = nums.length;

int count = 0;

while (length > 1) {

boolean nonDecreasing = true;

int maxSum = Integer.MAX\_VALUE;

int maxIndex = -1;

for (int i = 0; i < length - 1; i++) {

if (nums[i] > nums[i + 1]) {

nonDecreasing = false;

}

if (nums[i] + nums[i + 1] < maxSum) {

maxSum = nums[i] + nums[i + 1];

maxIndex = i;

}

}

if (nonDecreasing) {

break;

}

nums[maxIndex] = maxSum;

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

nums[i] = nums[i + 1];

}

length--;

count++;

}

return count;

}

}

1475. Final Prices With a Special Discount in a Shop

class Solution {

    public int[] finalPrices(int[] prices) {

        int n = prices.length;

        int[] result = new int[n];

        for (int i = 0; i < n; i++) {

            result[i] = prices[i];

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

                if (prices[j] <= prices[i]) {

                    result[i] -= prices[j];

                    break;

                }

            }

        }

        return result;

    }

}

1171. Remove Zero Sum Consecutive Nodes from LinkedList

class Solution

{

    public ListNode removeZeroSumSublists(ListNode head)

    {

        ListNode h = new ListNode(0);

        h.next = head;

        ListNode temp = h;

        while(temp != null)

        {

            ListNode t1 = temp.next;

            int sum =0;

            while(t1 != null)

            {

                sum += t1.val;

                if(sum == 0 )

                {

                    temp.next = t1.next;

                }

                t1 = t1.next;

            }

            if(temp != null)

                temp = temp.next;

        }

        return h.next;

    }

}