

Assignment

Name – JIGYA JAIN

UID – 22BCS50101

Section – 609-A

1. Longest Nice Substring –

Code -

```
class Solution {
    public String longestNiceSubstring(String s) {
        if(s.length() < 2) return "";
        HashSet<Character> set = new HashSet<>();
        char[] c = s.toCharArray();
        for(char i : c) set.add(i);
        int i = 0;
        while(i < s.length())
        {
            char x = c[i];
            if(set.contains(Character.toLowerCase(x)) &&
set.contains(Character.toUpperCase(x))) i++;
            else {
                String s1 = longestNiceSubstring(s.substring(0,i));
                String s2 = longestNiceSubstring(s.substring(i+1));
                return s1.length() >= s2.length() ? s1 : s2;
            }
        } return s;
    }
}
```

Submission –

The screenshot displays a coding platform interface. On the left, a table lists three submissions, all marked as 'Accepted' on February 25, 2025, using Java, with a runtime of 2 ms and memory usage around 42.5-42.7 MB. On the right, the 'Code' tab shows the Java implementation for the 'Longest Nice Substring' problem, which uses a recursive approach with a HashSet to check for palindromic substrings.

	Status	Language	Runtime	Memory	Notes
3	Accepted Feb 25, 2025	Java	2 ms	42.6 MB	
2	Accepted Feb 25, 2025	Java	2 ms	42.5 MB	
1	Accepted Feb 25, 2025	Java	2 ms	42.7 MB	

```
1 class Solution {
2     public String longestNiceSubstring(String s) {
3         if(s.length() < 2) return "";
4         HashSet<Character> set = new HashSet<>();
5
6         char[] c = s.toCharArray();
7         for(char i : c)
8         {
9             set.add(i);
10        }
11
12        int i = 0;
13        while(i < s.length())
14        {
15            char x = c[i];
16            if(set.contains(Character.toLowerCase(x)) && set.contains(Character.toUpperCase(x))) i++;
17            else
18            {
19                String s1 = longestNiceSubstring(s.substring(0,i));
20                String s2 = longestNiceSubstring(s.substring(i+1));
21                return s1.length() >= s2.length() ? s1 : s2;
22            }
23        } return s;
24    }
25 }
```

2. Reverse Bits –

Code –

```
public class Solution {  
    public int reverseBits(int n) {  
        n = ((n & 0xffff0000) >>> 16) | ((n & 0x0000ffff) << 16);  
        n = ((n & 0xff00ff00) >>> 8) | ((n & 0x00ff00ff) << 8);  
        n = ((n & 0xf0f0f0f0) >>> 4) | ((n & 0x0f0f0f0f) << 4);  
        n = ((n & 0xcccccccc) >>> 2) | ((n & 0x33333333) << 2);  
        n = ((n & 0xaaaaaaaa) >>> 1) | ((n & 0x55555555) << 1);  
        return n;  
    }  
}
```

Submission –

The screenshot shows a code editor interface with a dark theme. On the left, there's a 'Submissions' tab showing two successful submissions. The first submission is for 'Feb 21, 2025' with a runtime of '0 ms' and memory of '42.1 MB'. The second submission is also for 'Feb 21, 2025' with a runtime of '0 ms' and memory of '42.3 MB'. On the right, the 'Code' tab shows the Java code for the 'reverseBits' method, which is identical to the code provided in the previous block. The code uses bitwise operations to reverse the bits of an integer.

Status	Language	Runtime	Memory	Notes
Accepted	Java	0 ms	42.3 MB	
Accepted	Java	0 ms	42.1 MB	

```
1 public class Solution {  
2     public int reverseBits(int n) {  
3         n = ((n & 0xffff0000) >>> 16) | ((n & 0x0000ffff) << 16);  
4         n = ((n & 0xff00ff00) >>> 8) | ((n & 0x00ff00ff) << 8);  
5         n = ((n & 0xf0f0f0f0) >>> 4) | ((n & 0x0f0f0f0f) << 4);  
6         n = ((n & 0xcccccccc) >>> 2) | ((n & 0x33333333) << 2);  
7         n = ((n & 0xaaaaaaaa) >>> 1) | ((n & 0x55555555) << 1);  
8         return n;  
9     }  
10 }  
11 }
```

3. Number of 1 Bits –

Code –

```
class Solution {  
    public int hammingWeight(int n) {  
        int count = 0;  
        while(n != 0)  
        {  
            if((n & 1) == 1)    count++;  
            n = n>>1;  
        }  
        return count;  
    }  
}
```

Submission –

Problem List < > 🔍

Description | Editorial | Solutions | **Submissions**

	Status	Language	Runtime	Memory	Notes
1	Accepted Feb 21, 2025	Java	0 ms	40.5 MB	

Code

Java Auto

```

1 class Solution {
2     public int hammingWeight(int n) {
3         int count = 0;
4         while(n != 0)
5         {
6             if((n & 1) == 1)
7                 count++;
8             n = n>>1;
9         }
10        return count;
11    }
12 }
13 
```

4. Maximum Subarray –

Code –

```

class Solution {
    public int maxSubArray(int[] nums) {
        int maxSum = nums[0];
        int sum = 0;
        for(int val : nums)
        {
            sum += val;
            maxSum = (maxSum>sum)?maxSum:sum;
            if(sum < 0) sum = 0;
        }
        return maxSum;
    }
}

```

Submission –

Description | Editorial | Solutions | **Submissions**

	Status	Language	Runtime	Memory	Notes
9	Accepted Feb 26, 2025	Java	1 ms	56.9 MB	
8	Accepted Feb 26, 2025	Java	1 ms	56.9 MB	
7	Accepted Feb 26, 2025	Java	1 ms	57.1 MB	
6	Accepted Feb 26, 2025	Java	1 ms	57.1 MB	
5	Accepted Feb 26, 2025	Java	1 ms	56.9 MB	
4	Accepted Feb 26, 2025	Java	1 ms	57 MB	
3	Time Limit Exceeded Feb 26, 2025	Java	N/A	N/A	
2	Time Limit Exceeded Feb 26, 2025	Java	N/A	N/A	

Code

Java Auto

```

1 class Solution {
2     public int maxSubArray(int[] nums) {
3         int maxSum = nums[0];
4         int sum = 0;
5         for(int val : nums)
6         {
7             sum += val;
8             maxSum = (maxSum>sum)?maxSum:sum;
9             if(sum < 0) sum = 0;
10        }
11        return maxSum;
12    }
13 }

```

Saved

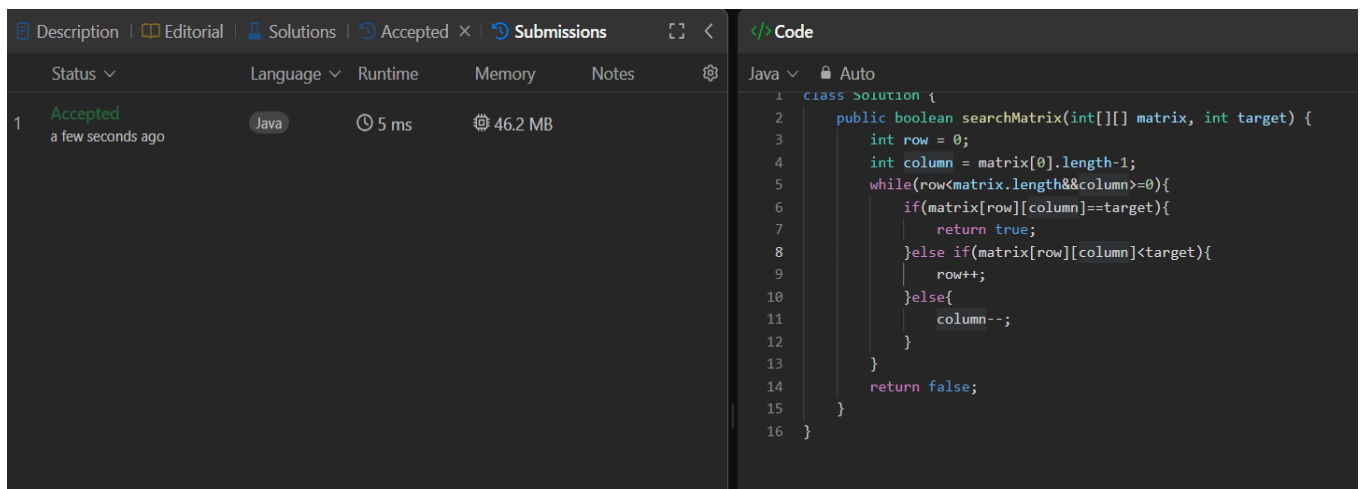
Testcase | Test Result

5. Search a 2D Matrix II –

Code –

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int row = 0;
        int column = matrix[0].length-1;
        while(row<matrix.length&&column>=0){
            if(matrix[row][column]==target){
                return true;
            }else if(matrix[row][column]<target){
                row++;
            }else{
                column--;
            }
        }
        return false;
    }
}
```

Submission –



The screenshot displays a submission interface with two main panels. The left panel shows the submission status: 'Accepted' (indicated by a green checkmark), 'a few seconds ago', 'Java' (language), '5 ms' (runtime), and '46.2 MB' (memory). The right panel, titled 'Code', shows the Java code for the 'searchMatrix' function, which is identical to the code provided in the previous block. The code is as follows:

```
1 class Solution {
2     public boolean searchMatrix(int[][] matrix, int target) {
3         int row = 0;
4         int column = matrix[0].length-1;
5         while(row<matrix.length&&column>=0){
6             if(matrix[row][column]==target){
7                 return true;
8             }else if(matrix[row][column]<target){
9                 row++;
10            }else{
11                column--;
12            }
13        }
14        return false;
15    }
16 }
```

6. Super Pow –

Code –

```
class Solution {
    private static final int MOD = 1337;

    private int pow(int a, int b) {
        int result = 1;
        a %= MOD;
        for (int i = 0; i < b; i++) {
```

```

        result = (result * a) % MOD;
    }
    return result;
}

public int superPow(int a, int[] b) {
    int result = 1;
    for (int i = b.length - 1; i >= 0; i--) {
        result = (result * pow(a, b[i])) % MOD;
        a = pow(a, 10);
    }
    return result;
}
}

```

Submission –

The screenshot shows a submission interface with a table of submissions and a code editor.

	Description	Editorial	Solutions	Accepted	Submissions
	Status	Language	Runtime	Memory	Notes
2	Accepted a few seconds ago	Java	4 ms	44.8 MB	
1	Wrong Answer Mar 17, 2025	Java	N/A	N/A	+ Notes

The code editor shows the following Java code:

```

1 class Solution {
2     private static final int MOD = 1337;
3
4     private int pow(int a, int b) {
5         int result = 1;
6         a %= MOD;
7         for (int i = 0; i < b; i++) {
8             result = (result * a) % MOD;
9         }
10        return result;
11    }
12
13    public int superPow(int a, int[] b) {
14        int result = 1;
15        for (int i = b.length - 1; i >= 0; i--) {
16            result = (result * pow(a, b[i])) % MOD;

```

7. Beautiful Array –

Code –

```

class Solution {
    public ListNode rotateRight(ListNode head, int k) {
        if (head == null || head.next == null) {
            return head;
        }

        int nodes = 1;
        ListNode beg = head;
        ListNode end = head;

        while (end.next != null) {

```

```

        end = end.next;
        nodes++;
    }

    int rotate = k % nodes;
    if (rotate == 0) {
        return head;
    }
    end.next = head;

    ListNode newTail = head;
    for (int i = 0; i < nodes - rotate - 1; i++) {
        newTail = newTail.next;
    }
    ListNode newHead = newTail.next;
    newTail.next = null;
    return newHead;
}
}

```

Submission –

The screenshot displays a submission interface. On the left, the submission is marked as 'Accepted' with 232/232 testcases passed. The runtime is 0 ms, achieving 100.00% beats, and the memory usage is 42.76 MB, achieving 36.07% beats. On the right, the Java code for the 'rotateRight' method is shown, which implements the logic for rotating a linked list by k positions.

```

10  */
11  class Solution {
12      public ListNode rotateRight(ListNode head, int k) {
13          if (head == null || head.next == null) {
14              return head;
15          }
16
17          int nodes = 1;
18          ListNode beg = head;
19          ListNode end = head;
20
21          while (end.next != null) {
22              end = end.next;
23              nodes++;
24          }
25
26          int rotate = k % nodes;
27          if (rotate == 0) {
28              return head;
29          }
30
31          end.next = head;

```

8. The Skyline Problem –

Code –

```

class Solution {
    public ListNode sortList(ListNode head) {
        if (head == null) {
            return null;

```

```
}
```

```
PriorityQueue<ListNode> minHeap = new PriorityQueue<>((a, b) ->  
Integer.compare(a.val, b.val));
```

```
while (head != null) {  
    minHeap.add(head);  
    head = head.next;  
}
```

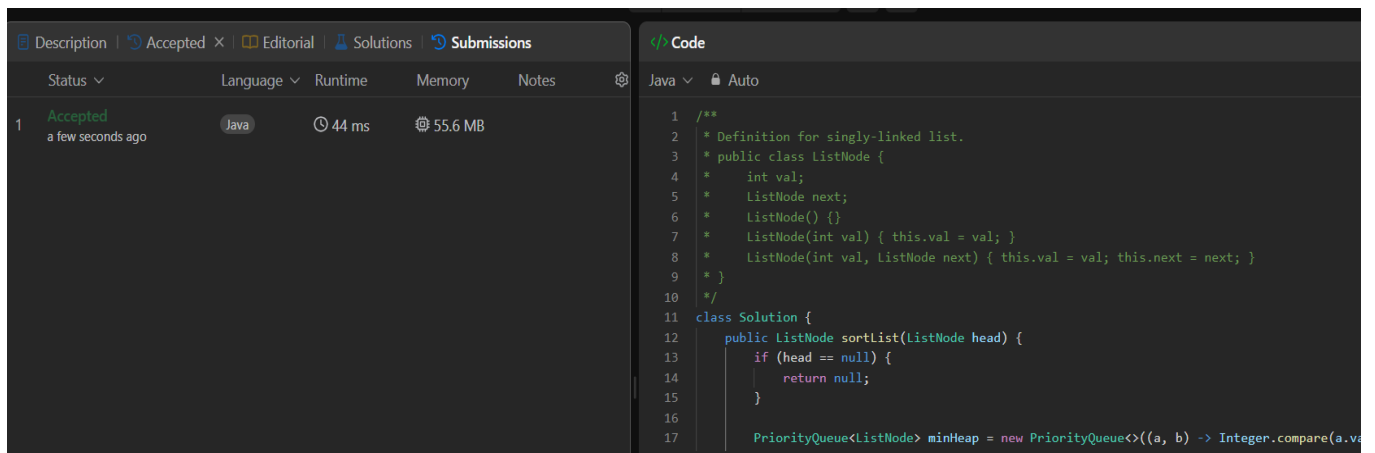
```
ListNode dummy = new ListNode(0);  
ListNode temp = dummy;
```

```
while (!minHeap.isEmpty()) {  
    temp.next = minHeap.poll();  
    temp = temp.next;  
}
```

```
temp.next = null; // Ensure the last node points to null  
return dummy.next;  
}
```

```
}
```

Submission –



```
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 sortList(ListNode head) {  
13         if (head == null) {  
14             return null;  
15         }  
16  
17         PriorityQueue<ListNode> minHeap = new PriorityQueue<>((a, b) -> Integer.compare(a.val,
```

9. Reverse Pairs –

Code –

```
class Solution {
```

```
    public void merge(int[] arr, int low, int mid, int high) {  
        ArrayList<Integer> temp = new ArrayList<>();
```

```

int left = low;
int right = mid+1;

while(left <= mid && right <= high) {
    if(arr[left] <= arr[right]) {
        temp.add(arr[left++]);
    } else {
        temp.add(arr[right++]);
    }
}

while(left <= mid) temp.add(arr[left++]);
while(right <= high) temp.add(arr[right++]);
for(int i=low; i<=high; i++) {
    arr[i] = temp.get(i-low);
}
}

public int countPairs(int[] arr, int low, int mid, int high) {
    int right = mid + 1;
    int cnt = 0;
    for(int i=low; i<=mid; i++) {
        while(right <= high && (long) arr[i] > 2L * arr[right])
            right++;
        cnt += (right - (mid + 1));
    }
    return cnt;
}

public int mergeSort(int[] arr, int low, int high) {
    int cnt = 0;
    if(low >= high) return cnt;
    int mid = (low + high) / 2;
    cnt += mergeSort(arr,low,mid);
    cnt += mergeSort(arr,mid+1,high);
    cnt += countPairs(arr,low,mid,high);
    merge(arr,low,mid,high);
    return cnt;
}

```



```

public int reversePairs(int[] nums) {
    int n = nums.length;
    return mergeSort(nums, 0, n-1);
}
}

```

Submission —

Description
Accepted
Editorial
Solutions
Submissions

	Status	Language	Runtime	Memory	Notes
3	Accepted a minute ago	Java	80 ms	54 MB	
2	Wrong Answer 4 minutes ago	Java	N/A	N/A	
1	Wrong Answer 5 minutes ago	Java	N/A	N/A	

Code

```

5      int left = low,
6      int right = mid+1;
7
8      while(left <= mid && right <= high) {
9          if(arr[left] <= arr[right]) {
10             temp.add(arr[left++]);
11          } else {
12             temp.add(arr[right++]);
13          }
14      }
15
16      while(left <= mid) temp.add(arr[left++]);
17      while(right <= high) temp.add(arr[right++]);
18      for(int i=low; i<=high; i++) {
19          arr[i] = temp.get(i-low);
20      }
21  }

```