

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

EXPERIMENT 10

STUDENT NAME: PRANJAL SINGH

UID: 22BCS13041

BRANCH: CSE

SECTION: 22BCS_FL_IOT_601A

SEMESTER: 6

DATE OF PERFORMANCE: 17/4/25

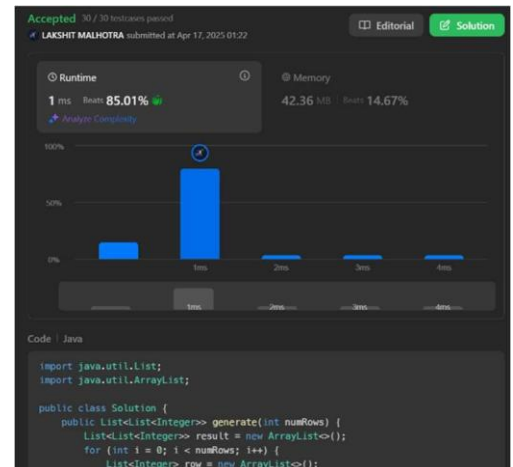
SUBJECT NAME: AP LAB -2

SUBJECT CODE: 22CSP-351

118. Pascal's Triangle

<https://leetcode.com/problems/pascals-triangle/>

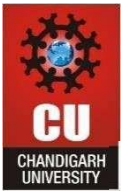
```
import java.util.List;
import java.util.ArrayList;
public class Solution {
    public List<List<Integer>> generate(int numRows) {
        List<List<Integer>> result = new ArrayList<>();
        for (int i = 0; i < numRows; i++) {
            List<Integer> row = new ArrayList<>();
            row.add(1);
            for (int j = 1; j < i; j++) {
                row.add(result.get(i - 1).get(j - 1) + result.get(i - 1).get(j));
            }
            if (i > 0) {
                row.add(1);
            }
            result.add(row);
        }
        return result;
    }
}
```



461. Hamming Distance

<https://leetcode.com/problems/hamming-distance/>

```
public class Solution {
    public int hammingDistance(int x, int y) {
        int xor = x ^ y;
        int distance = 0;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
while (xor != 0) {
    distance += (xor & 1);
    xor >>= 1;
}
return distance;
}
```

621. Task Scheduler

<https://leetcode.com/problems/task-scheduler/>

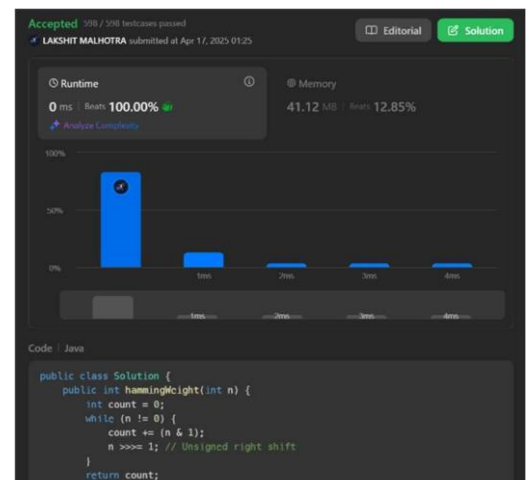
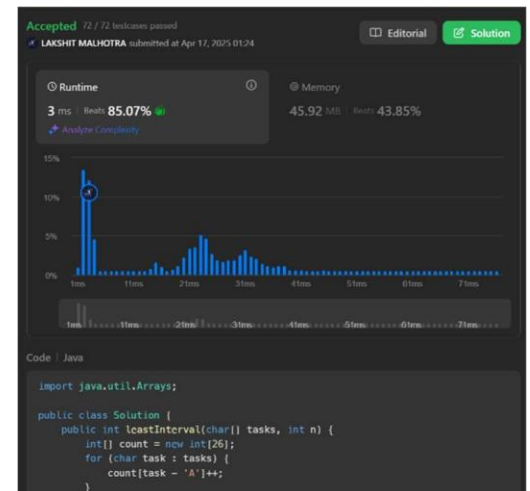
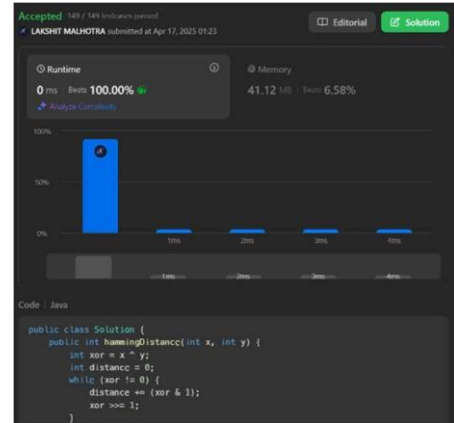
```
import java.util.Arrays;
public class Solution {
    public int leastInterval(char[] tasks, int n) {
        int[] count = new int[26];
        for (char task : tasks) {
            count[task - 'A']++;
        }
        Arrays.sort(count);
        int maxCount = count[25] - 1;
        int idleSlots = maxCount * n;

        for (int i = 24; i >= 0 && count[i] > 0; i--) {
            idleSlots -= Math.min(count[i], maxCount);
        }
        return idleSlots > 0 ? tasks.length + idleSlots : tasks.length;
    }
}
```

191. Number of 1 Bits

<https://leetcode.com/problems/number-of-1-bits/>

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



```

    }
    return count;
  }
}

```

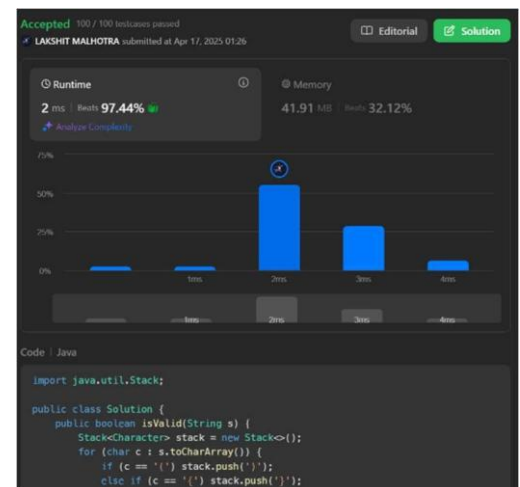
20. Valid Parentheses

<https://leetcode.com/problems/valid-parentheses/>

```

import java.util.Stack;
public class Solution {
    public boolean isValid(String s) {
        Stack<Character> stack = new Stack<>();
        for (char c : s.toCharArray()) {
            if (c == '(') stack.push('(');
            else if (c == '{') stack.push('{');
            else if (c == '[') stack.push('[');
            else if (stack.isEmpty() || stack.pop() != c) return
false;
        }
        return stack.isEmpty();
    }
}

```



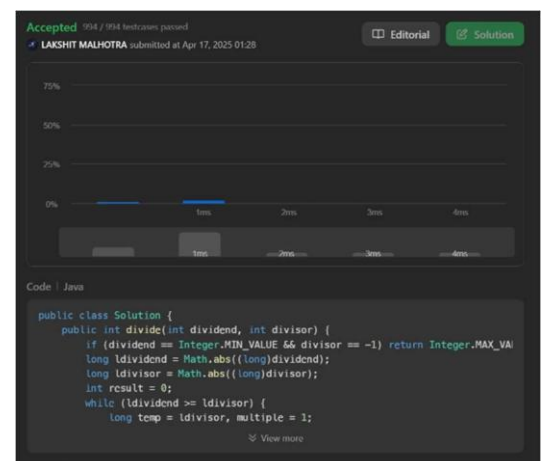
29. Divide Two Integers

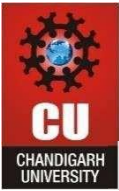
<https://leetcode.com/problems/divide-two-integers/>

```

public class Solution {
    public int divide(int dividend, int divisor) {
        if (dividend == Integer.MIN_VALUE && divisor
== -1) return Integer.MAX_VALUE;
        long ldividend = Math.abs((long) dividend);
        long ldivisor = Math.abs((long) divisor);
        int result = 0;
        while (ldividend >= ldivisor) {
            long temp = ldivisor, multiple = 1;
            while (ldividend >= (temp << 1)) {
                temp <<= 1;
                multiple <<= 1;

```





DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

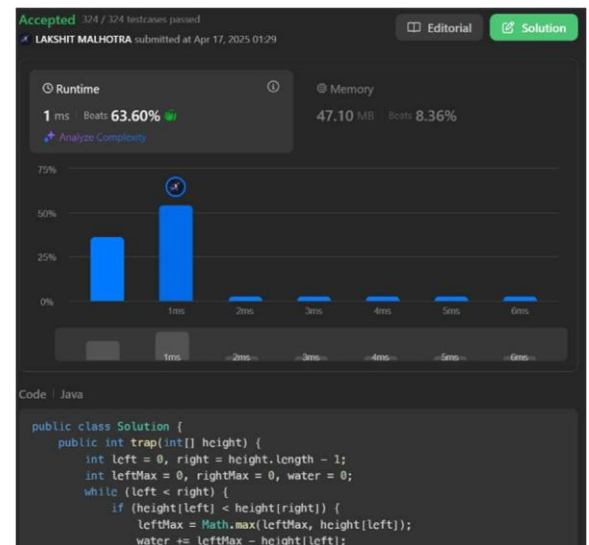
Discover. Learn. Empower.

```
}  
    dividend -= temp;  
    result += multiple;  
}  
return ((dividend > 0) == (divisor > 0)) ? result : -result;  
}  
}
```

42. Trapping Rain Water

<https://leetcode.com/problems/trapping-rain-water/>

```
public class Solution {  
    public int trap(int[] height) {  
        int left = 0, right = height.length - 1;  
        int leftMax = 0, rightMax = 0, water = 0;  
        while (left < right) {  
            if (height[left] < height[right]) {  
                leftMax = Math.max(leftMax,  
height[left]);  
                water += leftMax - height[left];  
                left++;  
            } else {  
                rightMax = Math.max(rightMax,  
height[right]);  
                water += rightMax - height[right];  
                right--;  
            }  
        }  
        return water;  
    }  
}
```



2071. Maximum Number of Tasks You Can Assign

<https://leetcode.com/problems/maximum-number-of-tasks-you-can-assign/>

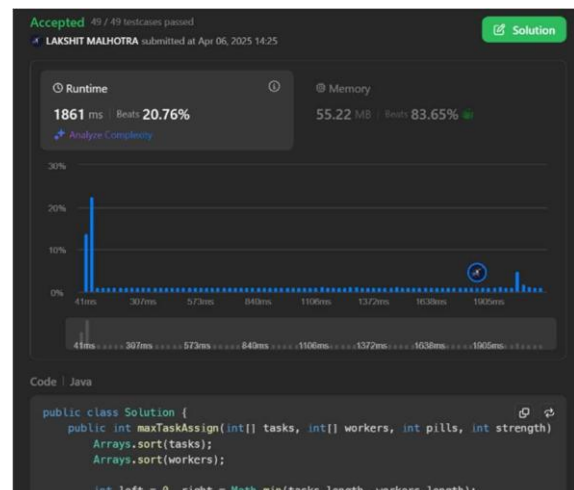
```
import java.util.Arrays;  
import java.util.PriorityQueue;  
public class Solution {  
    public int maxTaskAssign(int[] tasks, int[] workers, int pills, int strength) {  
        Arrays.sort(tasks);
```

```

Arrays.sort(workers);
int left = 0, right = Math.min(tasks.length, workers.length), res = 0;
while (left <= right) {
    int mid = (left + right) / 2;
    if (canAssign(mid, tasks, workers, pills, strength)) {
res = mid;
        left = mid + 1;
    } else {
        right = mid - 1;
    }
}
return res;
}

private boolean canAssign(int k, int[] tasks, int[] workers, int pills, int strength) {
    PriorityQueue<Integer> pq = new PriorityQueue<>();
    int j = workers.length - k;
    for (int i = 0; i < k; i++) pq.add(tasks[i]);
    int remainingPills = pills;
    for (int i = j; i < workers.length; i++) {
        if (!pq.isEmpty() && workers[i] >= pq.peek()) {
            pq.poll();
        } else if (remainingPills > 0) {
            Integer task = null;
            for (int t : pq) {
                if (workers[i] + strength >= t) {
                    task = t;
                    break;
                }
            }
            if (task != null) {
                pq.remove(task);
                remainingPills--;
            } else {
                return false;
            }
        } else return false;
    }
    return pq.isEmpty();
}
}

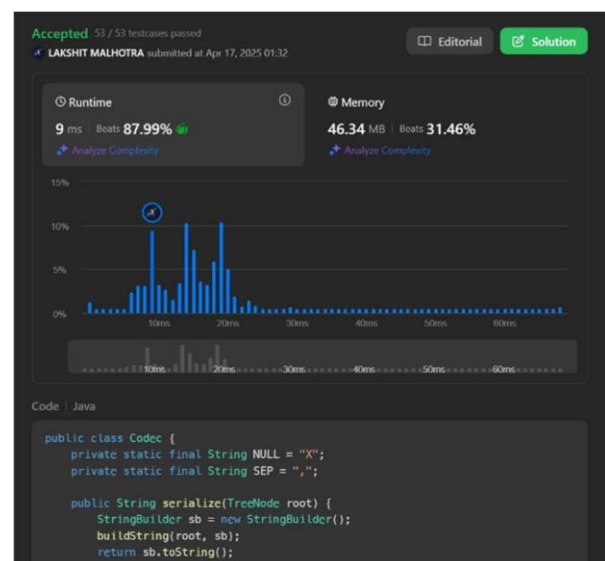
```



297. Serialize and Deserialize Binary Tree

<https://leetcode.com/problems/serialize-and-deserialize-binary-tree/>

```
public class Codec {
    private static final String NULL = "X";
    private static final String SEP = ",";
    public String serialize(TreeNode root) {
        StringBuilder sb = new StringBuilder();
        buildString(root, sb);
        return sb.toString();
    }
    private void buildString(TreeNode node, StringBuilder sb) {
        if (node == null) {
            sb.append(NULL).append(SEP);
            return;
        }
        sb.append(node.val).append(SEP);
        buildString(node.left, sb);
        buildString(node.right, sb);
    }
    public TreeNode deserialize(String data) {
        LinkedList<String> nodes = new LinkedList<>(Arrays.asList(data.split(SEP)));
        return buildTree(nodes);
    }
    private TreeNode
    buildTree(LinkedList<String> nodes) {
        String val = nodes.removeFirst();
        if (val.equals(NULL)) return null;
        TreeNode node = new
        TreeNode(Integer.parseInt(val));
        node.left = buildTree(nodes);
        node.right = buildTree(nodes);
        return node;
    }
}
```





DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

146. LRU Cache

<https://leetcode.com/problems/lru-cache/>

```
import java.util.LinkedHashMap;
import java.util.Map;
public class LRUCache extends LinkedHashMap<Integer, Integer> {
    private int capacity;
    public LRUCache(int capacity) {
        super(capacity, 0.75f, true);
        this.capacity = capacity;
    }
    public int get(int key) {
        return super.getOrDefault(key, -1);
    }
    public void put(int key, int value) {
        super.put(key, value);
    }
    @Override
    protected boolean
removeEldestEntry(Map.Entry<Integer,
Integer> eldest) {
        return size() > capacity;
    }
}
```

