Name: Aditya Singh

UID: 22BCS15063

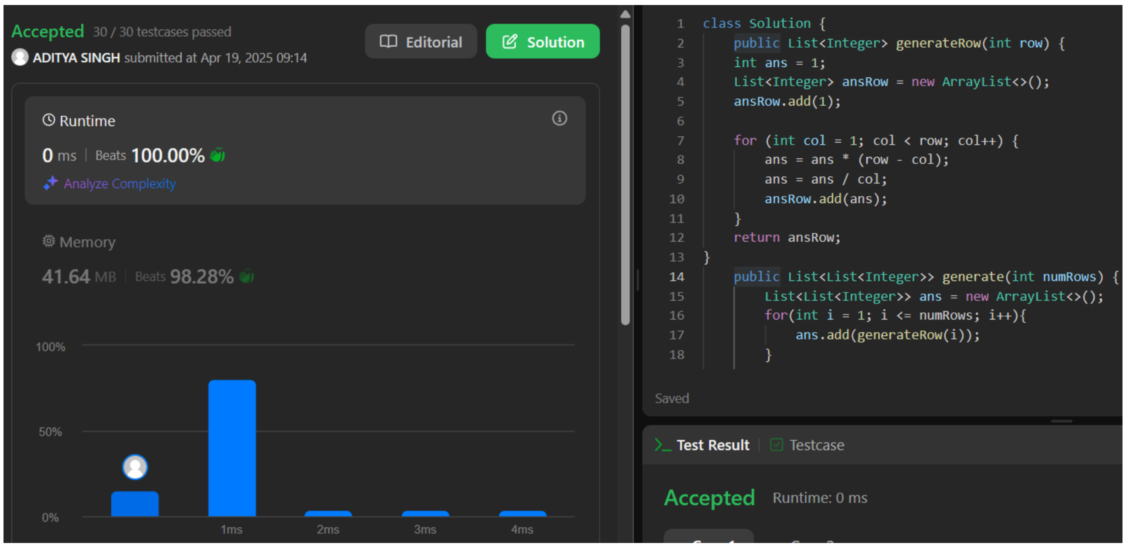
Section: FL_IOT_601 - A

Experiment-10 Solutions:-

1. **Pascal's Triangle:-**

```
class Solution {
    public List<Integer> generateRow(int row) {
        int ans = 1;
        List<Integer> ansRow = new ArrayList<>();
        ansRow.add(1);

        for (int col = 1; col < row; col++) {
            ans = ans * (row - col);
            ans = ans / col;
            ansRow.add(ans);
        }
        return ansRow;
    }

    public List<List<Integer>> generate(int numRows) {
        List<List<Integer>> ans = new ArrayList<>();
        for(int i = 1; i <= numRows; i++){
            ans.add(generateRow(i));
        }
        return ans;
    }
}
```

Result:-



Accepted 30 / 30 testcases passed
ADITYA SINGH submitted at Apr 19, 2025 09:14

Runtime
0 ms | Beats 100.00%
Analyze Complexity

Memory
41.64 MB | Beats 98.28%

```
class Solution {
    public List<Integer> generateRow(int row) {
    int ans = 1;
    List<Integer> ansRow = new ArrayList<>();
    ansRow.add(1);

    for (int col = 1; col < row; col++) {
        ans = ans * (row - col);
        ans = ans / col;
        ansRow.add(ans);
    }
    return ansRow;
}
    public List<List<Integer>> generate(int numRows) {
        List<List<Integer>> ans = new ArrayList<>();
        for(int i = 1; i <= numRows; i++){
            ans.add(generateRow(i));
        }
```

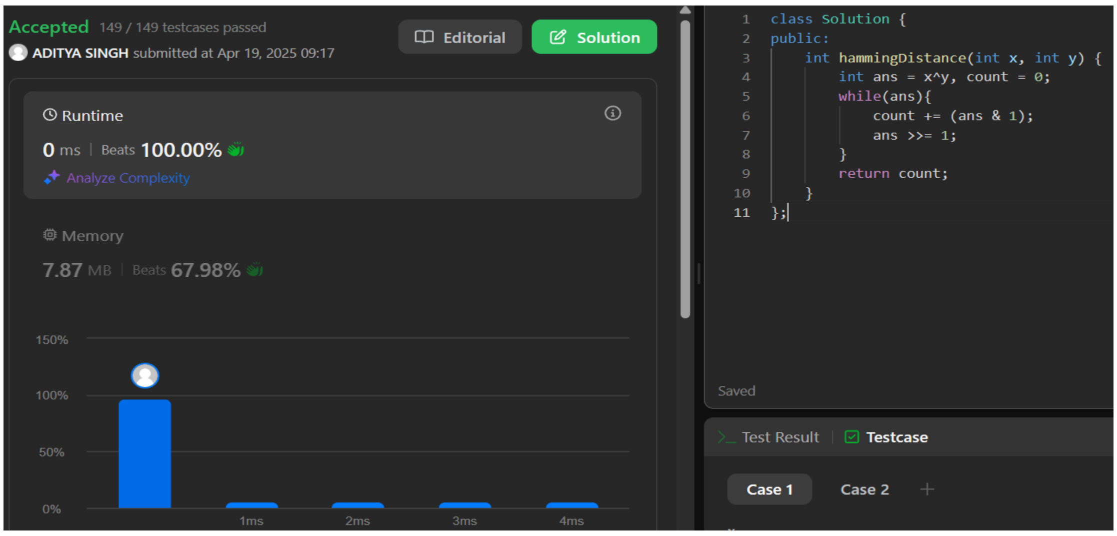
Saved

Accepted Runtime: 0 ms

2. Hamming Distance:-

```
class Solution {
public:
    int hammingDistance(int x, int y) {
        int ans = x^y, count = 0;
        while(ans){
            count += (ans & 1);
            ans >>= 1;
        }
        return count;
    }
};
```

Result:-



Accepted 149 / 149 testcases passed
ADITYA SINGH submitted at Apr 19, 2025 09:17

Runtime
0 ms | Beats 100.00%
Analyze Complexity

Memory
7.87 MB | Beats 67.98%

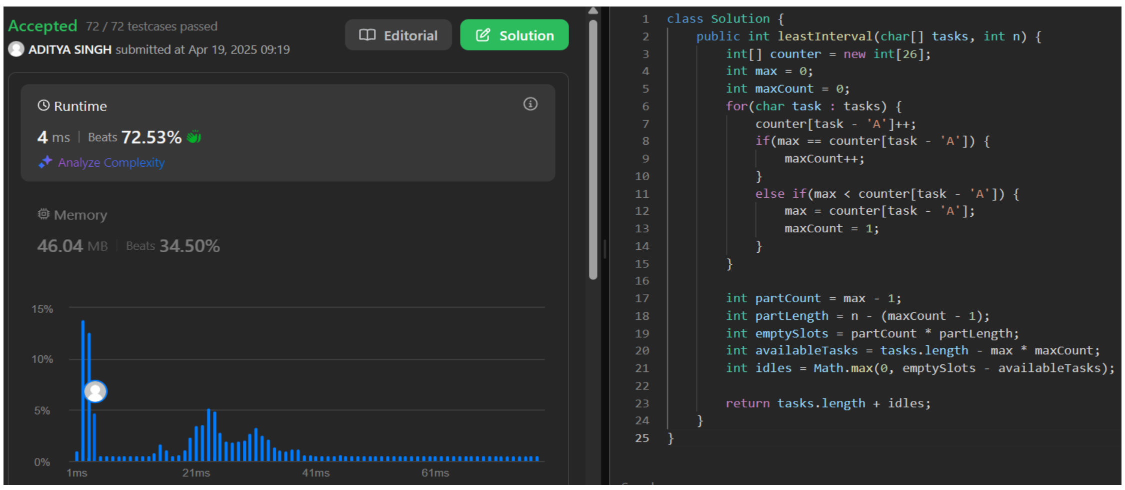
```
class Solution {
public:
    int hammingDistance(int x, int y) {
        int ans = x^y, count = 0;
        while(ans){
            count += (ans & 1);
            ans >>= 1;
        }
        return count;
    }
};
```

Saved

3. Task Scheduler:-

```
class Solution {
    public int leastInterval(char[] tasks, int n) {
        int[] counter = new int[26];
        int max = 0;
        int maxCount = 0;
        for(char task : tasks) {
            counter[task - 'A']++;
            if(max == counter[task - 'A']) {
                maxCount++;
            }
            else if(max < counter[task - 'A']) {
                max = counter[task - 'A'];
                maxCount = 1;
            }
        }

        int partCount = max - 1;
        int partLength = n - (maxCount - 1);
        int emptySlots = partCount * partLength;
        int availableTasks = tasks.length - max * maxCount;
        int idles = Math.max(0, emptySlots - availableTasks);

        return tasks.length + idles;
    }
}
```

Result:-


Accepted 72 / 72 testcases passed
ADITYA SINGH submitted at Apr 19, 2025 09:19

Runtime
4 ms | Beats 72.53%

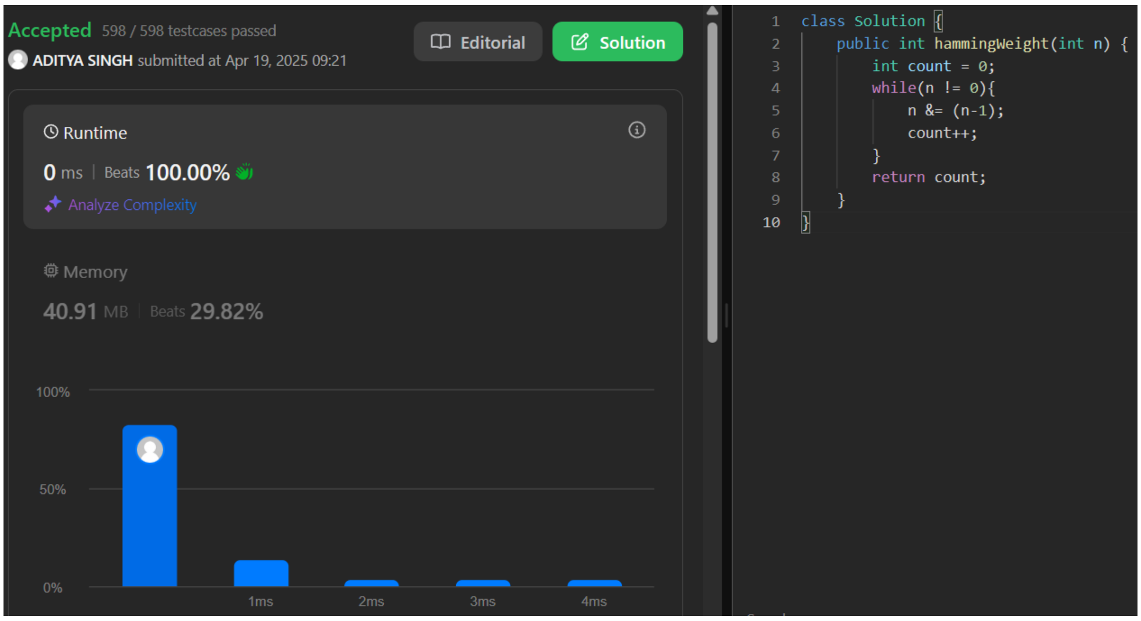
Memory
46.04 MB | Beats 34.50%

```
class Solution {
    public int leastInterval(char[] tasks, int n) {
        int[] counter = new int[26];
        int max = 0;
        int maxCount = 0;
        for(char task : tasks) {
            counter[task - 'A']++;
            if(max == counter[task - 'A']) {
                maxCount++;
            }
            else if(max < counter[task - 'A']) {
                max = counter[task - 'A'];
                maxCount = 1;
            }
        }

        int partCount = max - 1;
        int partLength = n - (maxCount - 1);
        int emptySlots = partCount * partLength;
        int availableTasks = tasks.length - max * maxCount;
        int idles = Math.max(0, emptySlots - availableTasks);

        return tasks.length + idles;
    }
}
```

4. Number of 1 Bits:-

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

Result:-



Accepted 598 / 598 testcases passed

ADITYA SINGH submitted at Apr 19, 2025 09:21

Runtime

0 ms | Beats 100.00%

Memory

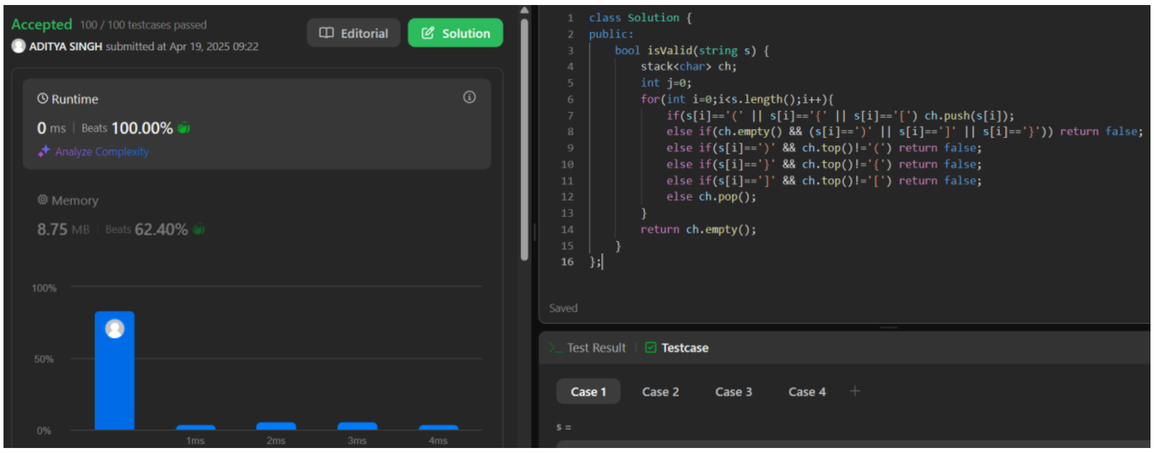
40.91 MB | Beats 29.82%

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

5. Valid Parentheses:-

```
class Solution {
public:
    bool isValid(string s) {
        stack<char> ch;
        int j=0;
        for(int i=0;i<s.length();i++){
            if(s[i]=='(' || s[i]=='{' || s[i]=='[') ch.push(s[i]);
            else if(ch.empty() && (s[i]==')' || s[i]==']' || s[i]=='}')) return false;
            else if(s[i]==')' && ch.top()!='(') return false;
            else if(s[i]=='}' && ch.top()!='{') return false;
            else if(s[i]==']' && ch.top()!='[') return false;
            else ch.pop();
        }
        return ch.empty();
    }
};
```

Result:-


Accepted 100 / 100 testcases passed

ADITYA SINGH submitted at Apr 19, 2025 09:22

Runtime

0 ms | Beats 100.00%

Memory

8.75 MB | Beats 62.40%

```
class Solution {
public:
    bool isValid(string s) {
        stack<char> ch;
        int j=0;
        for(int i=0;i<s.length();i++){
            if(s[i]=='(' || s[i]=='{' || s[i]=='[') ch.push(s[i]);
            else if(ch.empty() && (s[i]==')' || s[i]==']' || s[i]=='}')) return false;
            else if(s[i]==')' && ch.top()!='(') return false;
            else if(s[i]=='}' && ch.top()!='{') return false;
            else if(s[i]==']' && ch.top()!='[') return false;
            else ch.pop();
        }
        return ch.empty();
    }
};
```

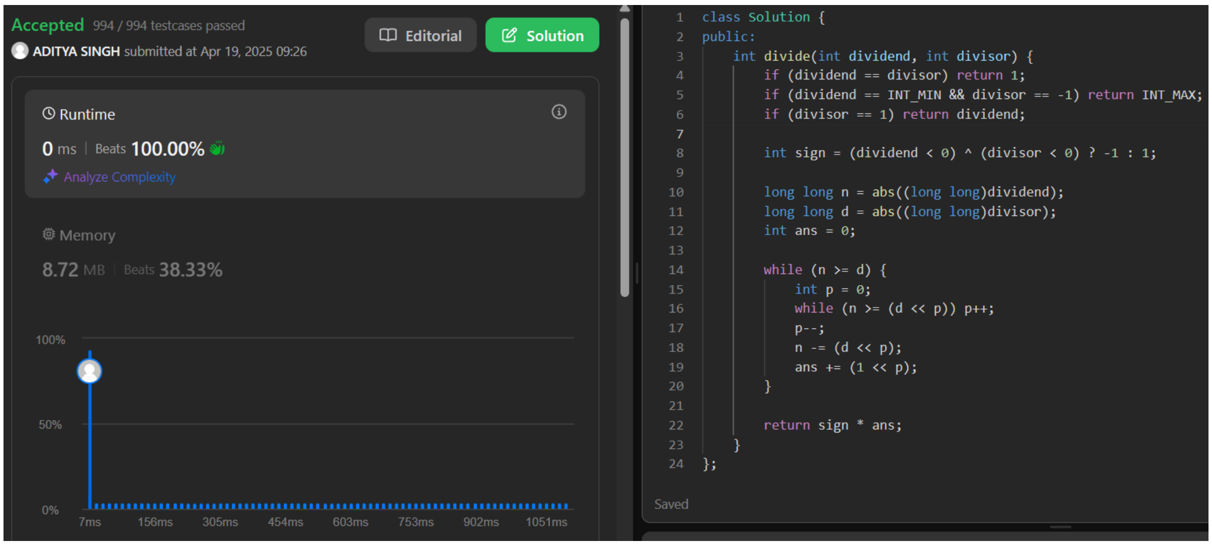
Saved

s =

6. **Divide Two Integers:-**

```
class Solution {
public:
    int divide(int dividend, int divisor) {
        if (dividend == divisor) return 1;
        if (dividend == INT_MIN && divisor == -1) return INT_MAX;
        if (divisor == 1) return dividend;

        int sign = (dividend < 0) ^ (divisor < 0) ? -1 : 1;
        long long n = abs((long long)dividend);
        long long d = abs((long long)divisor);
        int ans = 0;

        while (n >= d) {
            int p = 0;
            while (n >= (d << p)) p++;
            p--;
            n -= (d << p);
            ans += (1 << p);
        }

        return sign * ans;
    }
};
```

Result:-



Accepted 994 / 994 testcases passed

ADITYA SINGH submitted at Apr 19, 2025 09:26

Runtime

0 ms | Beats 100.00%

Memory

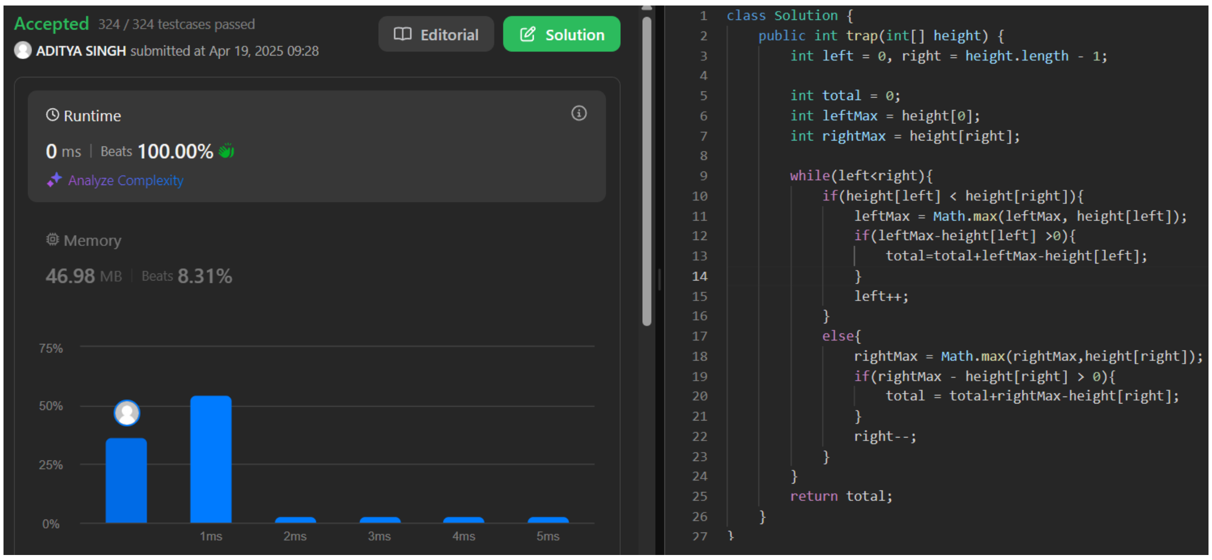
8.72 MB | Beats 38.33%

```
class Solution {
public:
    int divide(int dividend, int divisor) {
        if (dividend == divisor) return 1;
        if (dividend == INT_MIN && divisor == -1) return INT_MAX;
        if (divisor == 1) return dividend;

        int sign = (dividend < 0) ^ (divisor < 0) ? -1 : 1;

        long long n = abs((long long)dividend);
        long long d = abs((long long)divisor);
        int ans = 0;

        while (n >= d) {
            int p = 0;
            while (n >= (d << p)) p++;
            p--;
            n -= (d << p);
            ans += (1 << p);
        }

        return sign * ans;
    }
};
```

7. Trapping Rainwater:-

```
class Solution {
    public int trap(int[] height) {
        int left = 0, right = height.length - 1;

        int total = 0;
        int leftMax = height[0];
        int rightMax = height[right];

        while(left<right){
            if(height[left] < height[right]){
                leftMax = Math.max(leftMax, height[left]);
                if(leftMax-height[left] >0){
                    total=total+leftMax-height[left];
                }
                left++;
            }
            else{
                rightMax = Math.max(rightMax,height[right]);
                if(rightMax - height[right] > 0){
                    total = total+rightMax-height[right];
                }
                right--;
            }
        }
        return total;
    }
}
```

Result:-



Accepted 324 / 324 testcases passed
ADITYA SINGH submitted at Apr 19, 2025 09:28

Runtime
0 ms | Beats 100.00%

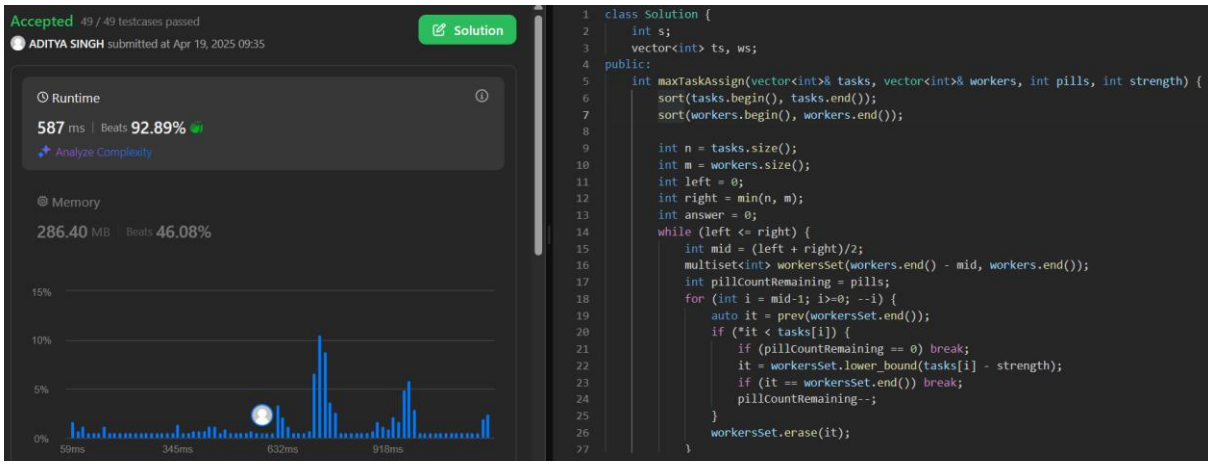
Memory
46.98 MB | Beats 8.31%

```
class Solution {
    public int trap(int[] height) {
        int left = 0, right = height.length - 1;

        int total = 0;
        int leftMax = height[0];
        int rightMax = height[right];

        while(left<right){
            if(height[left] < height[right]){
                leftMax = Math.max(leftMax, height[left]);
                if(leftMax-height[left] >0){
                    total=total+leftMax-height[left];
                }
                left++;
            }
            else{
                rightMax = Math.max(rightMax,height[right]);
                if(rightMax - height[right] > 0){
                    total = total+rightMax-height[right];
                }
                right--;
            }
        }
        return total;
    }
}
```

8. Maximum Number of Tasks You Can Assign:-

```
class Solution {
    int s;
    vector<int> ts, ws;
public:
    int maxTaskAssign(vector<int>& tasks, vector<int>& workers, int pills,
int strength) {
        sort(tasks.begin(), tasks.end());
        sort(workers.begin(), workers.end());
        int n = tasks.size();
        int m = workers.size();
        int left = 0;
        int right = min(n, m);
        int answer = 0;
        while (left <= right) {
            int mid = (left + right)/2;
            multiset<int> workersSet(workers.end() - mid, workers.end());
            int pillCountRemaining = pills;
            for (int i = mid-1; i>=0; --i) {
                auto it = prev(workersSet.end());
                if (*it < tasks[i]) {
                    if (pillCountRemaining == 0) break;
                    it = workersSet.lower_bound(tasks[i] - strength);
                    if (it == workersSet.end()) break;
                    pillCountRemaining--;
                }
                workersSet.erase(it);
            }

            if (workersSet.empty()) {
                answer = mid;
                left = mid + 1;
            }
            else    right = mid - 1;
        }
        return answer;
    }
};
```

Result:-



Accepted 49 / 49 testcases passed
ADITYA SINGH submitted at Apr 19, 2025 09:35

Runtime
587 ms | Beats 92.89%

Memory
286.40 MB | Beats 46.08%

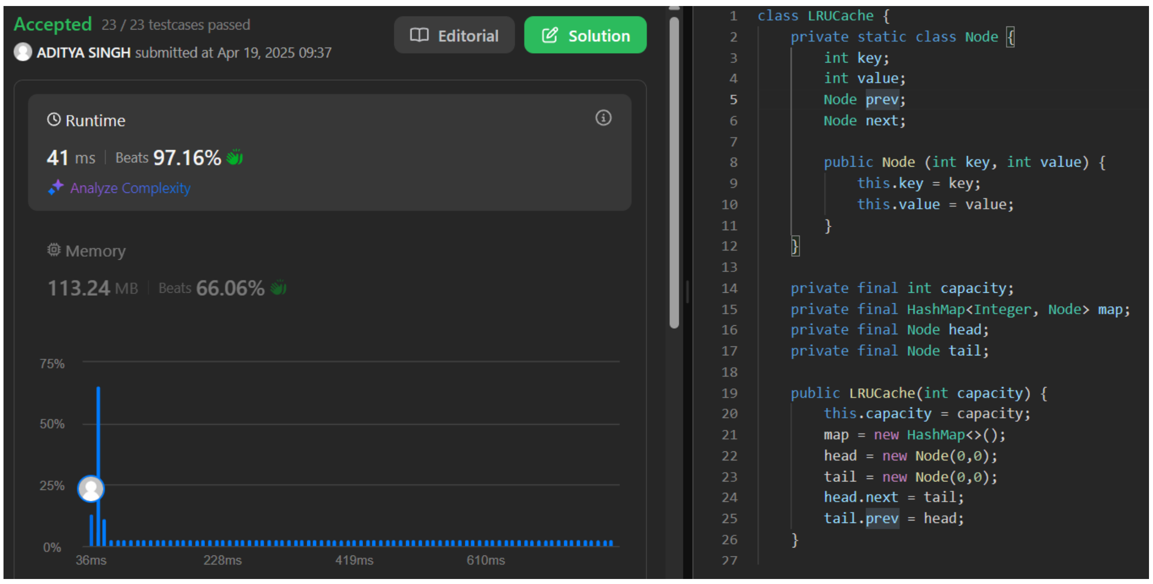
```
class Solution {
    int s;
    vector<int> ts, ws;
public:
    int maxTaskAssign(vector<int>& tasks, vector<int>& workers, int pills, int strength) {
        sort(tasks.begin(), tasks.end());
        sort(workers.begin(), workers.end());

        int n = tasks.size();
        int m = workers.size();
        int left = 0;
        int right = min(n, m);
        int answer = 0;
        while (left <= right) {
            int mid = (left + right)/2;
            multiset<int> workersSet(workers.end() - mid, workers.end());
            int pillCountRemaining = pills;
            for (int i = mid-1; i>=0; --i) {
                auto it = prev(workersSet.end());
                if (*it < tasks[i]) {
                    if (pillCountRemaining == 0) break;
                    it = workersSet.lower_bound(tasks[i] - strength);
                    if (it == workersSet.end()) break;
                    pillCountRemaining--;
                }
                workersSet.erase(it);
            }
```

9. **<u>LRU Cache:-</u>**

```
class LRUCache {
   private static class Node {
      int key;
      int value;
      Node prev;
      Node next;

      public Node (int key, int value) {
         this.key = key;
         this.value = value;
      }
   }

   private final int capacity;
   private final HashMap<Integer, Node> map;
   private final Node head;
   private final Node tail;

   public LRUCache(int capacity) {
      this.capacity = capacity;
      map = new HashMap<>();
      head = new Node(0,0);
      tail = new Node(0,0);
      head.next = tail;
      tail.prev = head;
   }

   public int get(int key) {
      if(!map.containsKey(key)){
         return -1;
      }
      Node node = map.get(key);
      remove(node);
      insertAtHead(node);
      return node.value;
   }
```

```
    public void put(int key, int value) {
        if (map.containsKey(key)){
            Node node = map.get(key);
            node.value = value;
            remove(node);
            insertAtHead(node);
        }
        else{
            if(map.size() == capacity){
                map.remove(tail.prev.key);
                remove(tail.prev);
            }
            Node newNode = new Node(key,value);
            map.put(key, newNode);
            insertAtHead(newNode);
        }
    }
    private void remove(Node node) {
        node.prev.next = node.next;
        node.next.prev = node.prev;
    }
    private void insertAtHead(Node node) {
        node.next = head.next;
        node.next.prev = node;
        head.next = node;
        node.prev = head;
    }
}

/**
 * Your LRUCache object will be instantiated and called as such:
 * LRUCache obj = new LRUCache(capacity);
 * int param_1 = obj.get(key);
 * obj.put(key,value);
 */
```

Result:-



Accepted 23 / 23 testcases passed
ADITYA SINGH submitted at Apr 19, 2025 09:37

Runtime
41 ms | Beats 97.16%

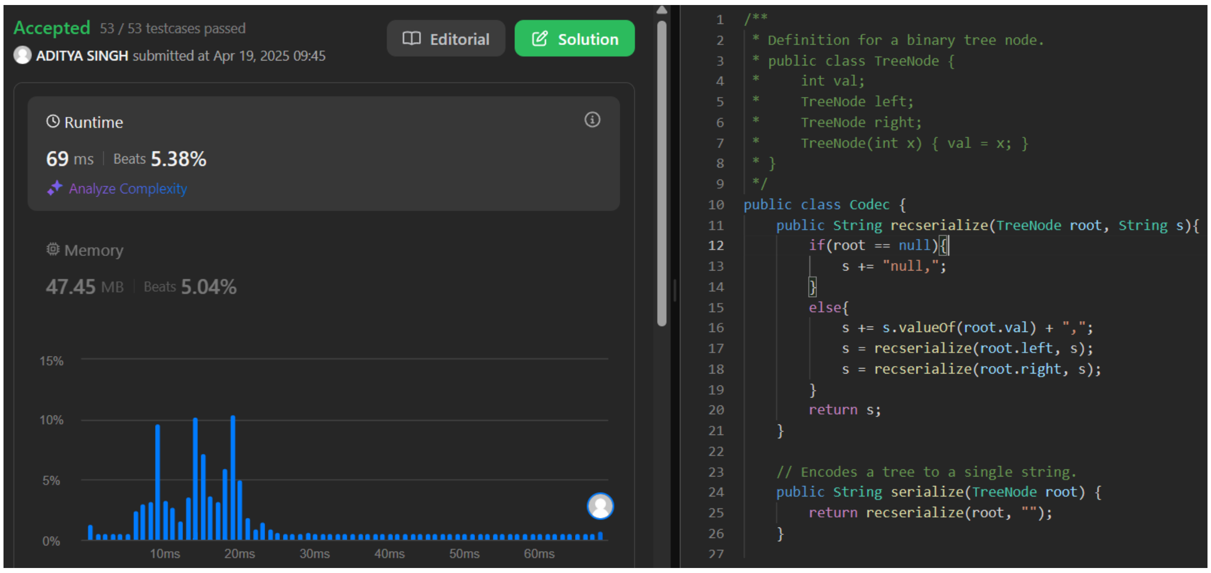
Memory
113.24 MB | Beats 66.06%

```
class LRUCache {
    private static class Node {
        int key;
        int value;
        Node prev;
        Node next;

        public Node (int key, int value) {
            this.key = key;
            this.value = value;
        }
    }

    private final int capacity;
    private final HashMap<Integer, Node> map;
    private final Node head;
    private final Node tail;

    public LRUCache(int capacity) {
        this.capacity = capacity;
        map = new HashMap<>();
        head = new Node(0,0);
        tail = new Node(0,0);
        head.next = tail;
        tail.prev = head;
    }
```

10. Serialize and Deserialize Binary Tree:-

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
public class Codec {
    public String recserialize(TreeNode root, String s){
        if(root == null){
            s += "null,";
        }
        else{
            s += s.valueOf(root.val) + ",";
            s = recserialize(root.left, s);
            s = recserialize(root.right, s);
        }
        return s;
    }

    // Encodes a tree to a single string.
    public String serialize(TreeNode root) {
        return recserialize(root, "");
    }

    public TreeNode recdeserialize(List<String> str){
        if(str.get(0).equals("null")){
            str.remove(0);
            return null;
        }
        TreeNode root = new TreeNode(Integer.valueOf(str.get(0)));
        str.remove(0);
        root.left = recdeserialize(str);
        root.right = recdeserialize(str);

        return root;
    }
    // Decodes your encoded data to tree.
    public TreeNode deserialize(String data) {
        String[] strArray = data.split(",");
        List<String> strList = new
LinkedList<String>(Arrays.asList(strArray));
        return recdeserialize(strList);
    }
}
```

<u>Result:-</u>


Accepted 53 / 53 testcases passed
ADITYA SINGH submitted at Apr 19, 2025 09:45

Runtime
69 ms | Beats 5.38%

Memory
47.45 MB | Beats 5.04%

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
public class Codec {
    public String recserialize(TreeNode root, String s){
        if(root == null){
            s += "null,";
        }
        else{
            s += s.valueOf(root.val) + ",";
            s = recserialize(root.left, s);
            s = recserialize(root.right, s);
        }
        return s;
    }

    // Encodes a tree to a single string.
    public String serialize(TreeNode root) {
        return recserialize(root, "");
    }

```