

AP Assignment 10

Name: Sushil Kumar

UID: 22BCS16771

Section: 22BCS_IOT614-B

Ques 1 <https://leetcode.com/problems/pascals-triangle/description/>

Code:

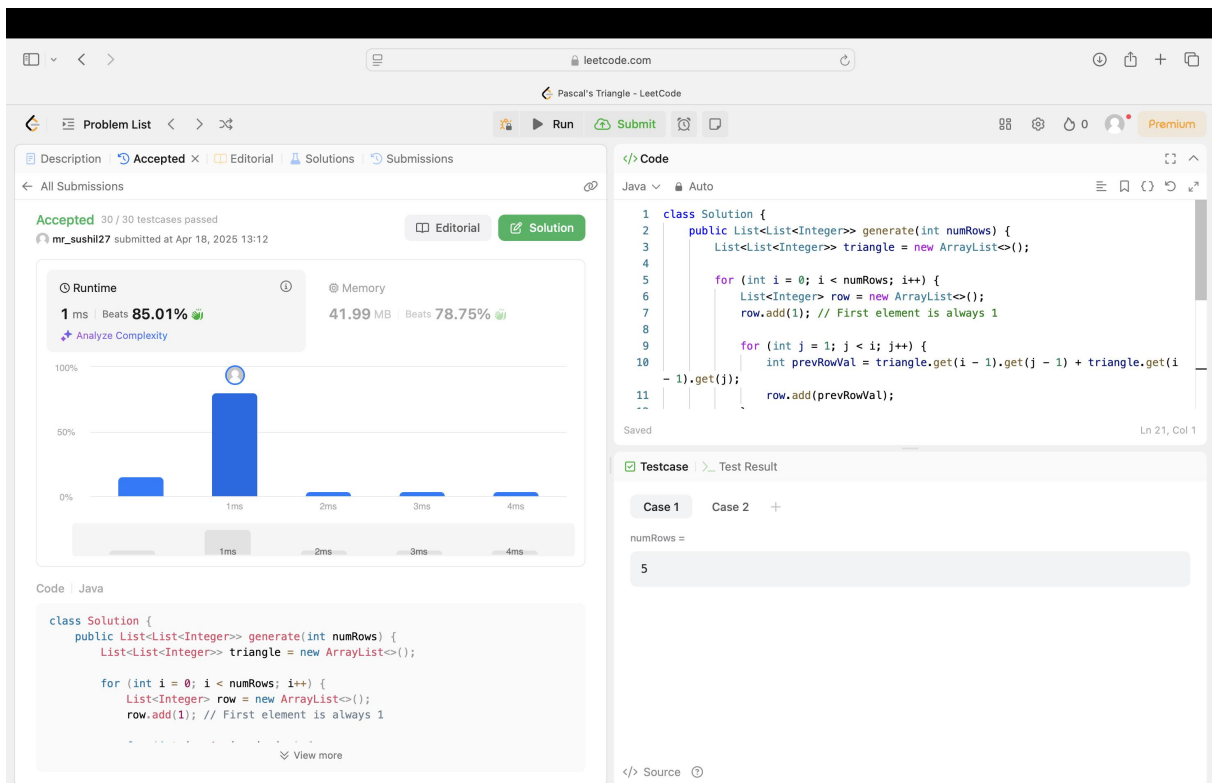
```
class Solution {
    public List<List<Integer>> generate(int numRows) {
        List<List<Integer>> triangle = new ArrayList<>();

        for (int i = 0; i < numRows; i++) {
            List<Integer> row = new ArrayList<>();
            row.add(1); // First element is always 1

            for (int j = 1; j < i; j++) {
                int prevRowVal = triangle.get(i - 1).get(j - 1) + triangle.get(i - 1).get(j);
                row.add(prevRowVal);
            }

            if (i > 0) row.add(1); // Last element is 1 (for rows > 1)
            triangle.add(row);
        }

        return triangle;
    }
}
```



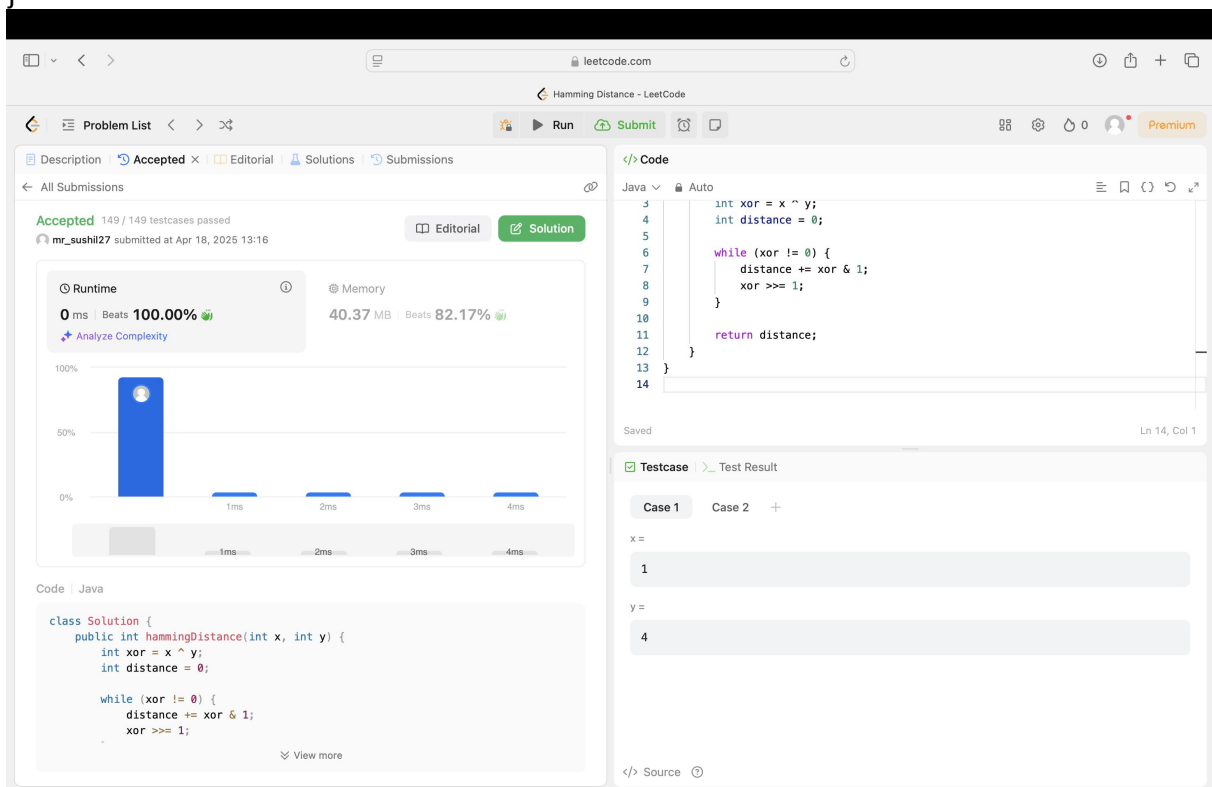
Q2. <https://leetcode.com/problems/hamming-distance/>

CODE:

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

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

        return distance;
    }
}
```



Q3. <https://leetcode.com/problems/task-scheduler/description/>

Code:

```
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 maxFreq = count[25] - 1;
        int idleSlots = maxFreq * n;

        for (int i = 24; i >= 0 && count[i] > 0; i--) {
```

```

        idleSlots -= Math.min(count[i], maxFreq);
    }

    return idleSlots > 0 ? idleSlots + tasks.length : tasks.length;
}
}

```

The screenshot displays the LeetCode 'Task Scheduler' problem page. On the left, the 'Accepted' status is confirmed with 72/72 testcases passed. The runtime is 4 ms (72.27% beats) and memory is 46.23 MB (18.99% beats). A bar chart visualizes the runtime performance across various test cases. The code editor on the right shows a Java solution that counts the frequency of each task, sorts them, and calculates the minimum idle slots required. The test case section shows an input of tasks = ["A","A","A","B","B"] and n = 2, with the expected output being the same sequence of tasks.

Q4. <https://leetcode.com/problems/number-of-1-bits/description/>

Code:

```

public class Solution {
    // you need to treat n as an unsigned value
    public int hammingWeight(int n) {
        int count = 0;
        while (n != 0) {
            count += n & 1;
            n >>= 1; // use unsigned right shift
        }
        return count;
    }
}

```

The screenshot shows a LeetCode submission for the problem "Number of 1 Bits". The submission is accepted, with a runtime of 0ms and memory usage of 40.81 MB. The code is in Java and implements a function to count the number of 1 bits in an integer using a while loop and bit shifting.

```

public class Solution {
    // you need to treat n as an unsigned value
    public int hammingWeight(int n) {
        int count = 0;
        while (n != 0) {
            count += n & 1;
            n >>= 1; // use unsigned right shift
        }
        return count;
    }
}

```

Q5. <https://leetcode.com/problems/divide-two-integers/description/>

CODE:

```

class Solution {
    public int divide(int dividend, int divisor) {
        // Edge case: overflow
        if (dividend == Integer.MIN_VALUE && divisor == -1)
            return Integer.MAX_VALUE;

        // Convert both numbers to long and take absolute values
        long a = Math.abs((long) dividend);
        long b = Math.abs((long) divisor);
        int result = 0;

        while (a >= b) {
            long temp = b, multiple = 1;
            while (a >= (temp << 1)) {
                temp <<= 1;
                multiple <<= 1;
            }
            a -= temp;
            result += multiple;
        }

        // Apply sign
        return (dividend > 0) == (divisor > 0) ? result : -result;
    }
}

```

The screenshot displays the LeetCode submission page for the 'Divide Two Integers' problem. The submission is accepted, with 994/994 test cases passed. The runtime is 1 ms, beating 74.35% of other submissions, and the memory usage is 41.20 MB, beating 34.98%. The Java code is shown on the right, and the test case on the left shows dividend = 10 and divisor = 3, resulting in a quotient of 3.

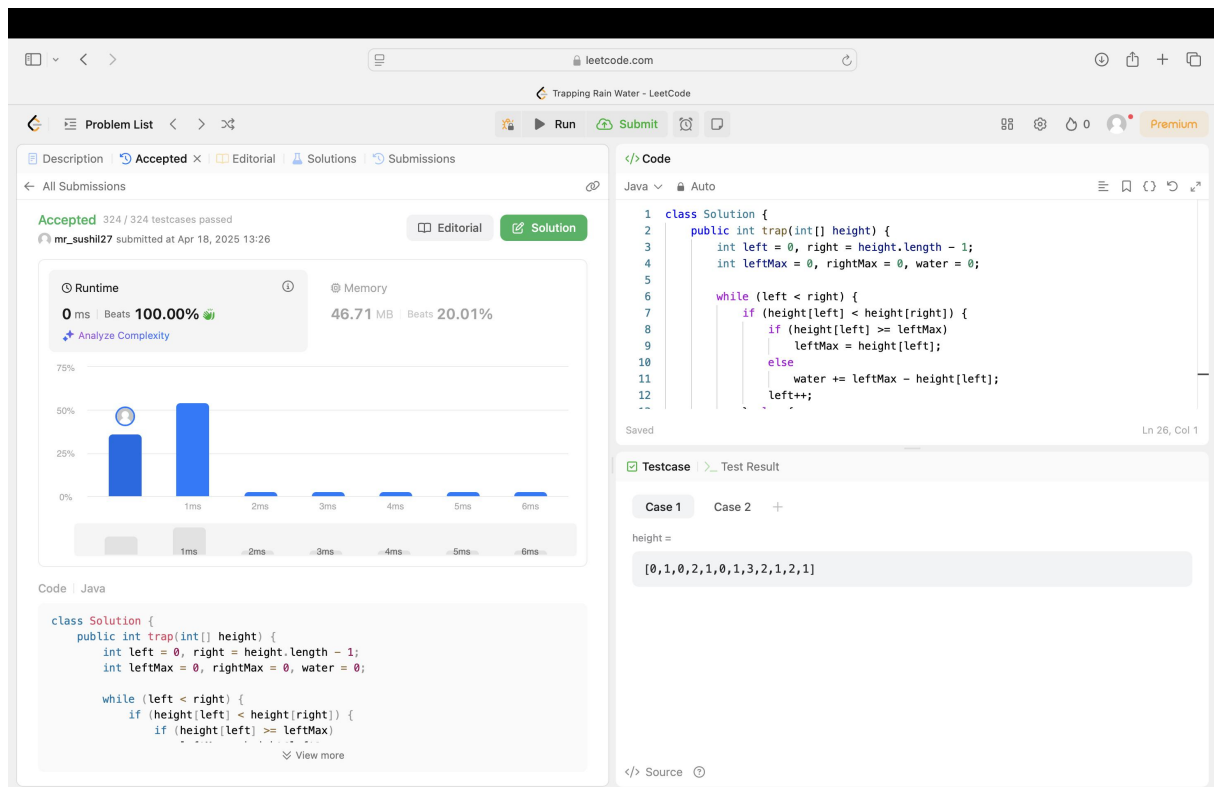
Q6. <https://leetcode.com/problems/trapping-rain-water/description/>

CODE:-

```
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]) {
                if (height[left] >= leftMax)
                    leftMax = height[left];
                else
                    water += leftMax - height[left];
                left++;
            } else {
                if (height[right] >= rightMax)
                    rightMax = height[right];
                else
                    water += rightMax - height[right];
                right--;
            }
        }

        return water;
    }
}
```



Q7. <https://leetcode.com/problems/maximum-number-of-tasks-you-can-assign/description/?envType=problem-list-v2&envId=greedy>

CODE:-

```

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

    public boolean canAssign(int count, int[] tasks, int[] workers, int pills, int strength){
        Deque<Integer> dq = new ArrayDeque<>();
        int ind = workers.length - 1;
    }
}

```

```

for (int i = count - 1; i >= 0; i--) {
    while(ind >= workers.length - count && workers[ind] + strength >= tasks[i])
    {
        dq.offerLast(workers[ind]);
        ind--;
    }
    if(dq.isEmpty()) return false;
    if(dq.peekFirst() >= tasks[i])
    {
        dq.pollFirst();
    }
    else
    {
        dq.pollLast();
        pills--;
        if(pills < 0) return false;
    }
}
return true;
}
}

```

Maximum Number of Tasks You Can Assign - LeetCode

Greedy

Accepted 49 / 49 testcases passed
mr_sushil27 submitted at Apr 18, 2025 13:28

Runtime: 64 ms, Beats 73.94%
Memory: 55.04 MB, Beats 96.36%

Code

```

class Solution {
    public int maxTaskAssign(int[] tasks, int[] workers, int pills, int strength)
    {
        int left = 0, right = Math.min(tasks.length, workers.length);
        Arrays.sort(tasks);
        Arrays.sort(workers);
        while(left < right)
        {
            int mid = left + (right - left) / 2;
            if(canAssign(mid, tasks, workers, pills, strength))
            {
                left = mid;
            }
        }
    }
}

```

Testcase

Case 1 Case 2 Case 3 +

tasks =
[3, 2, 1]

workers =
[0, 3, 3]

pills =
1

strength =

Source

Q8. <https://leetcode.com/problems/serialize-and-deserialize-binary-tree/description/>

CODE:-

```

public class Codec {

    // Encodes a tree to a single string.
    public String serialize(TreeNode root) {
        StringBuilder sb = new StringBuilder();
    }
}

```

```

        serializeHelper(root, sb);
        return sb.toString();
    }

    private void serializeHelper(TreeNode root, StringBuilder sb) {
        if (root == null) {
            sb.append("#,");
            return;
        }
        sb.append(root.val).append(",");
        serializeHelper(root.left, sb);
        serializeHelper(root.right, sb);
    }

    // Decodes your encoded data to tree.
    public TreeNode deserialize(String data) {
        Queue<String> nodes = new LinkedList<>(Arrays.asList(data.split(", ")));
        return deserializeHelper(nodes);
    }

    private TreeNode deserializeHelper(Queue<String> nodes) {
        String val = nodes.poll();
        if (val.equals("#")) return null;

        TreeNode root = new TreeNode(Integer.parseInt(val));
        root.left = deserializeHelper(nodes);
        root.right = deserializeHelper(nodes);
        return root;
    }
}

```

The screenshot displays the LeetCode submission interface for the problem "Serialize and Deserialize Binary Tree". The left sidebar shows the problem status as "Accepted" with 53/53 test cases passed, submitted by "mr_sushil27" on April 18, 2025. The performance metrics show a runtime of 9 ms (beating 87.99%) and memory usage of 46.45 MB (beating 23.57%). A bar chart compares the user's performance against other submissions. The main area shows the Java code for the "Codec" class, which implements the serialize and deserialize methods. The right sidebar shows the test results for Case 1 with the input "[1, 2, 3, null, null, 4, 5]", which corresponds to the binary tree diagram shown below the input.

```

public class Codec {
    // Encodes a tree to a single string.
    public String serialize(TreeNode root) {
        StringBuilder sb = new StringBuilder();
        serializeHelper(root, sb);
        return sb.toString();
    }

    // Decodes your encoded data to tree.
    private void serializeHelper(TreeNode root, StringBuilder sb) {
        if (root == null) {
            sb.append("#,");
            return;
        }
        sb.append(root.val).append(",");
        serializeHelper(root.left, sb);
        serializeHelper(root.right, sb);
    }

    public TreeNode deserialize(String data) {
        Queue<String> nodes = new LinkedList<>(Arrays.asList(data.split(", ")));
        return deserializeHelper(nodes);
    }

    private TreeNode deserializeHelper(Queue<String> nodes) {
        String val = nodes.poll();
        if (val.equals("#")) return null;

        TreeNode root = new TreeNode(Integer.parseInt(val));
        root.left = deserializeHelper(nodes);
        root.right = deserializeHelper(nodes);
        return root;
    }
}

```

Testcase 1: [1, 2, 3, null, null, 4, 5]

```

graph TD
    1((1)) --- 2((2))
    1 --- 3((3))
    3 --- 4((4))
    3 --- 5((5))

```


Q9. <https://leetcode.com/problems/lru-cache/description/>

CODE:-

```
class LRUCache extends LinkedHashMap<Integer, Integer> {
    private int capacity;

    public LRUCache(int capacity) {
        super(capacity, 0.75f, true); // accessOrder = 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;
    }
}
```

The screenshot displays the LeetCode interface for the LRU Cache problem. The top section shows the problem title and a link to the description. Below this, the 'Accepted' status is confirmed with 23/23 test cases passed. A runtime graph shows a single bar at 41ms, indicating a fast solution. The submitted code is shown in the 'Code' tab, which is a Java implementation of the LRU Cache using LinkedHashMap. The code is as follows:

```
class LRUCache extends LinkedHashMap<Integer, Integer> {
    private int capacity;

    public LRUCache(int capacity) {
        super(capacity, 0.75f, true); // accessOrder = 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;
    }
}
```

The bottom section shows the 'Testcase' tab with a single test case passing. The test case input is ["LRUCache", "put", "put", "get", "put", "get", "put", "get", "get", "get"] and the output is [[2], [1, 1], [2, 2], [1], [3, 3], [2], [4, 4], [1], [3], [4]].