WORKSHEET 3

Student Name: Parth Arora **UID:** 22BCS16661

Branch: BE-CSE **Section/Group:** 22BCS_NTPP-602-A

Semester: 6th **Date of Performance:** 13/01/2025

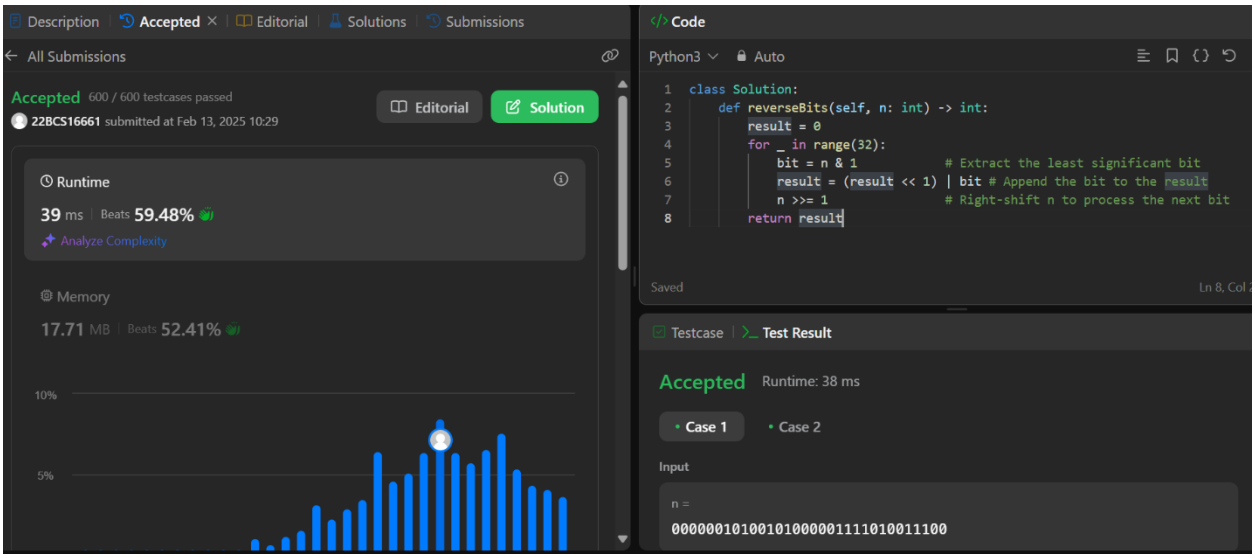
Subject Name: AP LAB - II **Subject Code:** 22CSP-351

1. Aim: Reverse bits of a given 32 bits unsigned integer.

2. Source Code:

```
class Solution:
    def reverseBits(self, n: int) -> int:
        result = 0
        for _ in range(32):
            bit = n & 1          # Extract the least significant bit
            result = (result << 1) | bit # Append the bit to the result
            n >>= 1              # Right-shift n to process the next bit
        return result
```

3. Screenshots of outputs:



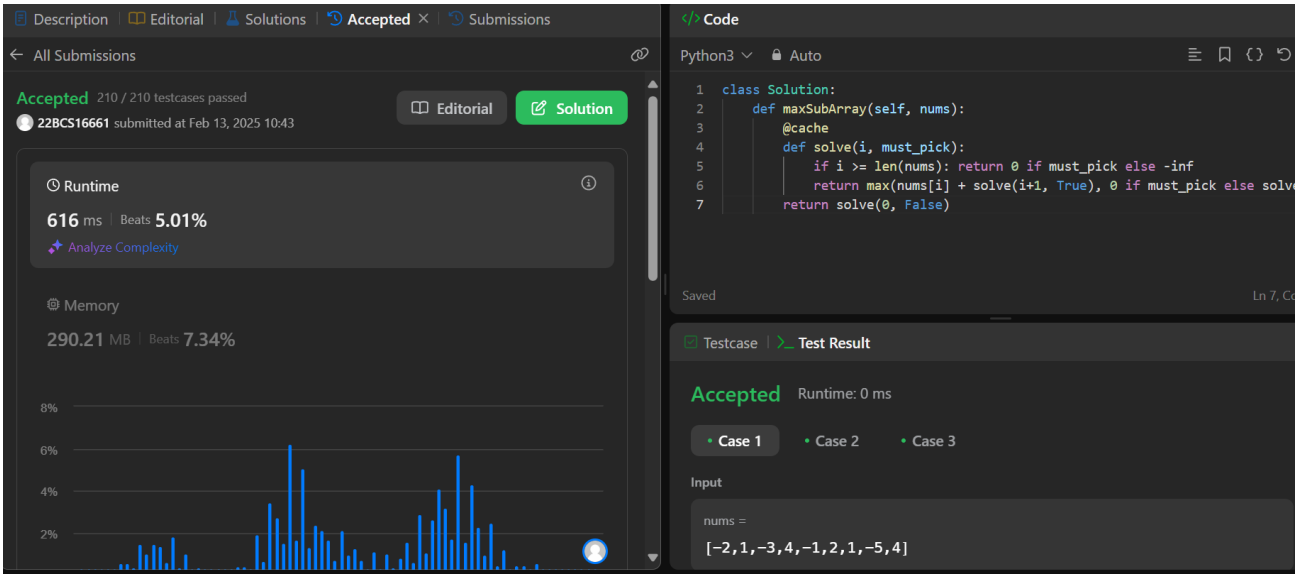
2.

Aim: Given an integer array nums, find the subarray with the largest sum, and return *its sum*.

Source Code:

```
class Solution:
    def maxSubArray(self, nums):
        ans = -inf
        for i in range(len(nums)):
            cur_sum = 0
            for j in range(i, len(nums)):
                cur_sum += nums[j]
                ans = max(ans, cur_sum)
        return ans
```

Screenshots of outputs:



3.

Aim: A city's **skyline** is the outer contour of the silhouette formed by all the buildings in that city when viewed from a distance. Given the locations and heights of all the buildings, return *the **skyline** formed by these buildings collectively*.

The geometric information of each building is given in the array buildings where buildings[i] = [$left_i$, $right_i$, $height_i$]:

- $left_i$ is the x coordinate of the left edge of the i^{th} building.
- $right_i$ is the x coordinate of the right edge of the i^{th} building.
- $height_i$ is the height of the i^{th} building.

You may assume all buildings are perfect rectangles grounded on an absolutely flat surface at height 0.

The **skyline** should be represented as a list of "key points" **sorted by their x-coordinate** in the form [[x_1,y_1],[x_2,y_2],...]. Each key point is the left endpoint of some horizontal segment in the skyline except the last point in the list, which always has a y-coordinate 0 and is used to mark the skyline's termination where the rightmost building ends. Any ground between the leftmost and rightmost buildings should be part of the skyline's contour.

Note: There must be no consecutive horizontal lines of equal height in the output skyline. For instance, [...,[2 3],[4 5],[7 5],[11 5],[12 7],...] is not acceptable; the three lines of height 5 should be merged into one in the final output as such: [...,[2 3],[4 5],[12 7],...]

.

Source Code:

```
class Solution:
    def getSkyline(self, buildings: List[List[int]]) -> List[List[int]]:
        # for the same x, (x, -H) should be in front of (x, 0)
        # For Example 2, we should process (2, -3) then (2, 0), as there's no height change
        x_height_right_tuples = sorted([(L, -H, R) for L, R, H in buildings] + [(R, 0, "doesn't
matter") for _, R, _ in buildings])
        # (0, float('inf')) is always in max_heap, so max_heap[0] is always valid
        result, max_heap = [[0, 0]], [(0, float('inf'))]
        for x, negative_height, R in x_height_right_tuples:
            while x >= max_heap[0][1]:
                # reduce max height up to date, i.e. only consider max height in the right side of line x
                heapq.heappop(max_heap)
            if negative_height:
                # Consider each height, as it may be the potential max height
                heapq.heappush(max_heap, (negative_height, R))
            curr_max_height = -max_heap[0][0]
            if result[-1][1] != curr_max_height:
                result.append([x, curr_max_height])
```

```
        return result[1:]
```

4. Screenshots of outputs: