Course Name:	**Competitive Programming Laboratory (216U01L401)**	**Semester:**	**IV**
Date of Performance:	**06 / 02 / 2025**	**DIV/ Batch No:**	**A1**
Student Name:	**Aaryan Sharma**	**Roll No:**	**16010123012**

Experiment No: 4

Title: To implement a competitive programming problem on a platform (eg. Leetcode) optimized using a greedy approach.

Aim and Objective of the Experiment:

1. **Understand** the concepts of Greedy Approach.
2. **Apply** the concepts to solve the problem.
3. **Implement** the solution to given problem statement.
4. **Create** test cases for testing the solution.
5. **Analyze** the result for efficiency of the solution.

COs to be achieved:

CO3: Solve intricate problems involving graphs, tree structures, and algorithms.

Books/ Journals/ Websites referred:

1. https://leetcode.com/problems/minimum-number-of-coins-to-be-added/description/

Theory:

The Greedy Algorithm is an optimization strategy used to solve problems by making a sequence of choices, each of which is the best at that moment. This approach does not always guarantee an optimal solution. The key idea is to make local choices that lead to a globally optimal solution.

Problem statement

You are given a 0-indexed integer array coins, representing the values of the coins available, and an integer target.
An integer x is obtainable if there exists a subsequence of coins that sums to x.
Return [1, target].
A subsequence of an array is a new non-empty array that is formed from the original array by deleting some (possibly none) of the elements without disturbing the relative positions of the remaining elements.

Example 1:Input: coins = [1,4,10], target = 19

Output: 2
Explanation: We need to add coins 2 and 8. The resulting array will be [1,2,4,8,10].
It can be shown that all integers from 1 to 19 are obtainable from the resulting array, and that 2 is the minimum number of coins that need to be added to the array.

Code:

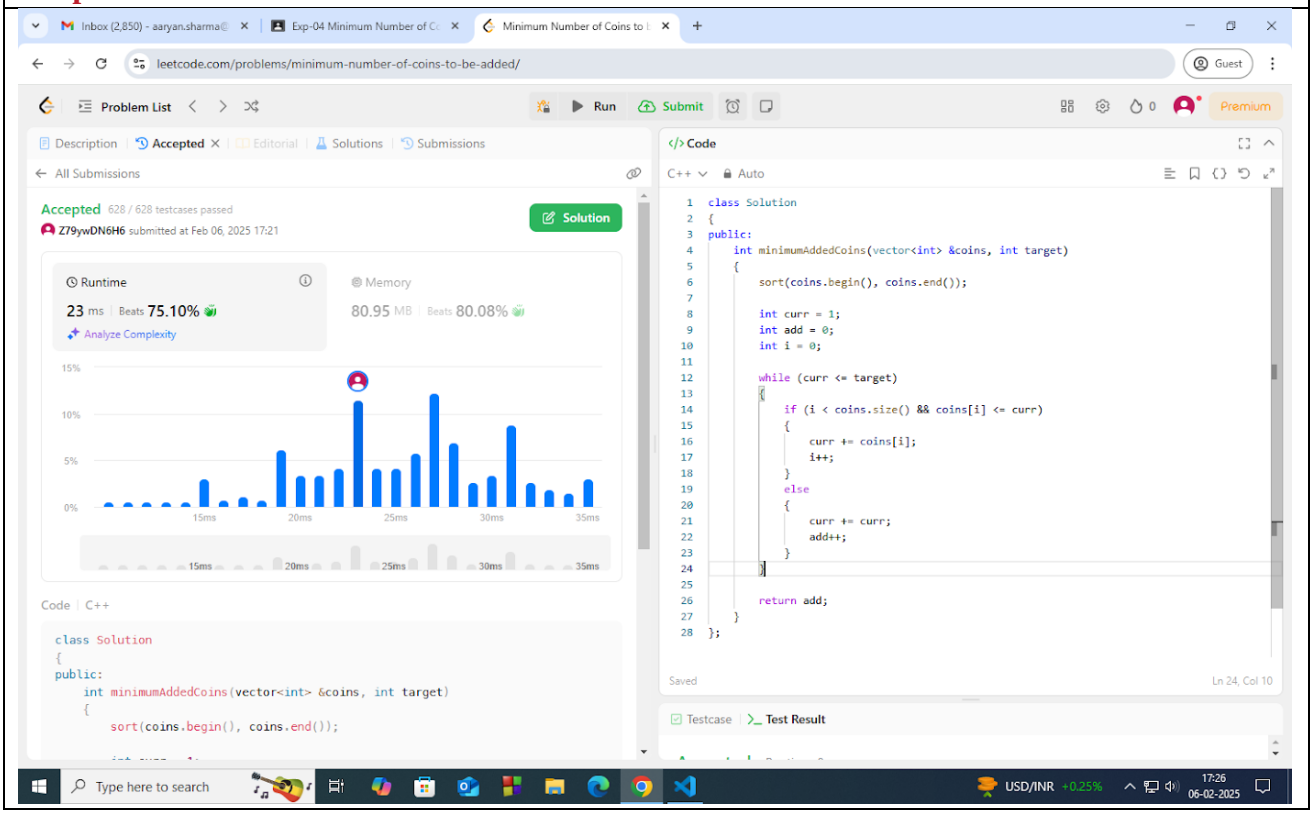
```
class Solution
{
public:
   int minimumAddedCoins(vector<int> &coins, int target)
   {
      sort(coins.begin(), coins.end());

      int curr = 1;
      int add = 0;
      int i = 0;

      while (curr <= target)
      {
         if (i < coins.size() && coins[i] <= curr)
         {
            curr += coins[i];
            i++;
         }
         else
         {
            curr += curr;
            add++;
         }
      }

      return add;
   }
};
```

Output:



Conclusion:

I implemented the greedy approach to optimize coin selection, ensuring all values up to the target were obtainable. This experiment enhanced my understanding of greedy algorithms and their efficiency in problem-solving.