**DAA LAB 6**

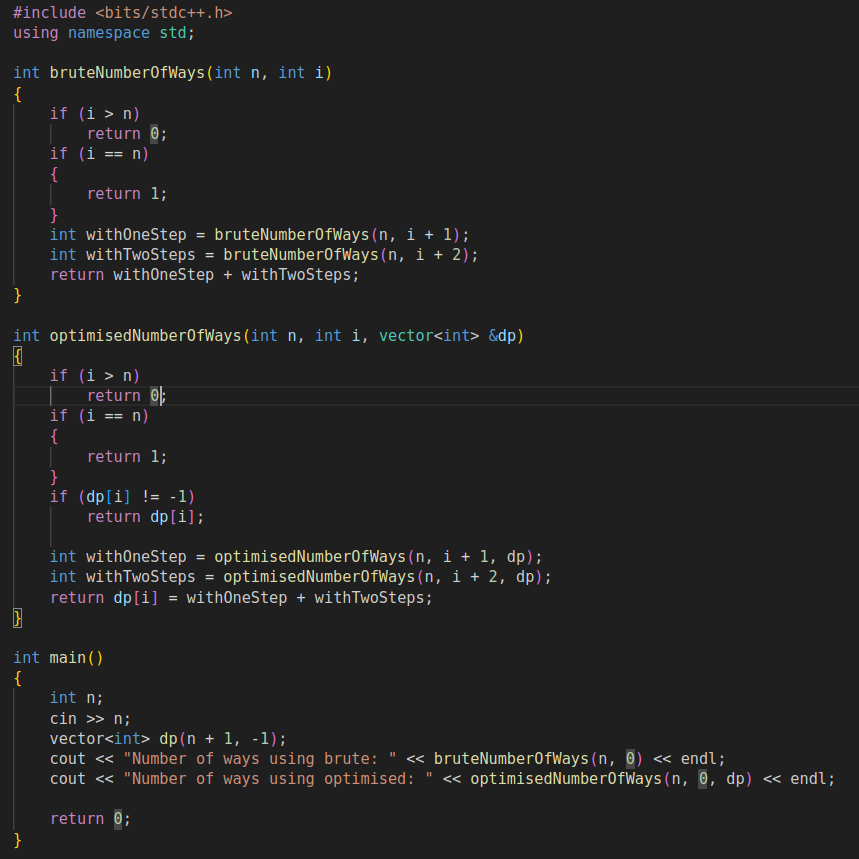
**Name: Vatsal Bhuva**

**Roll No.: IIT2022004**

**Section: A**

Q1) Imagine you are standing at the bottom of a staircase with n steps. You want to reach the top of the staircase. However, you can only climb the stairs by taking steps of size 1 or 2 at a time. The goal is to find out in how many distinct ways you can climb to the top of the staircase.

**Brute force and optimised approach:**

****

**Time Complexity Analysis:**

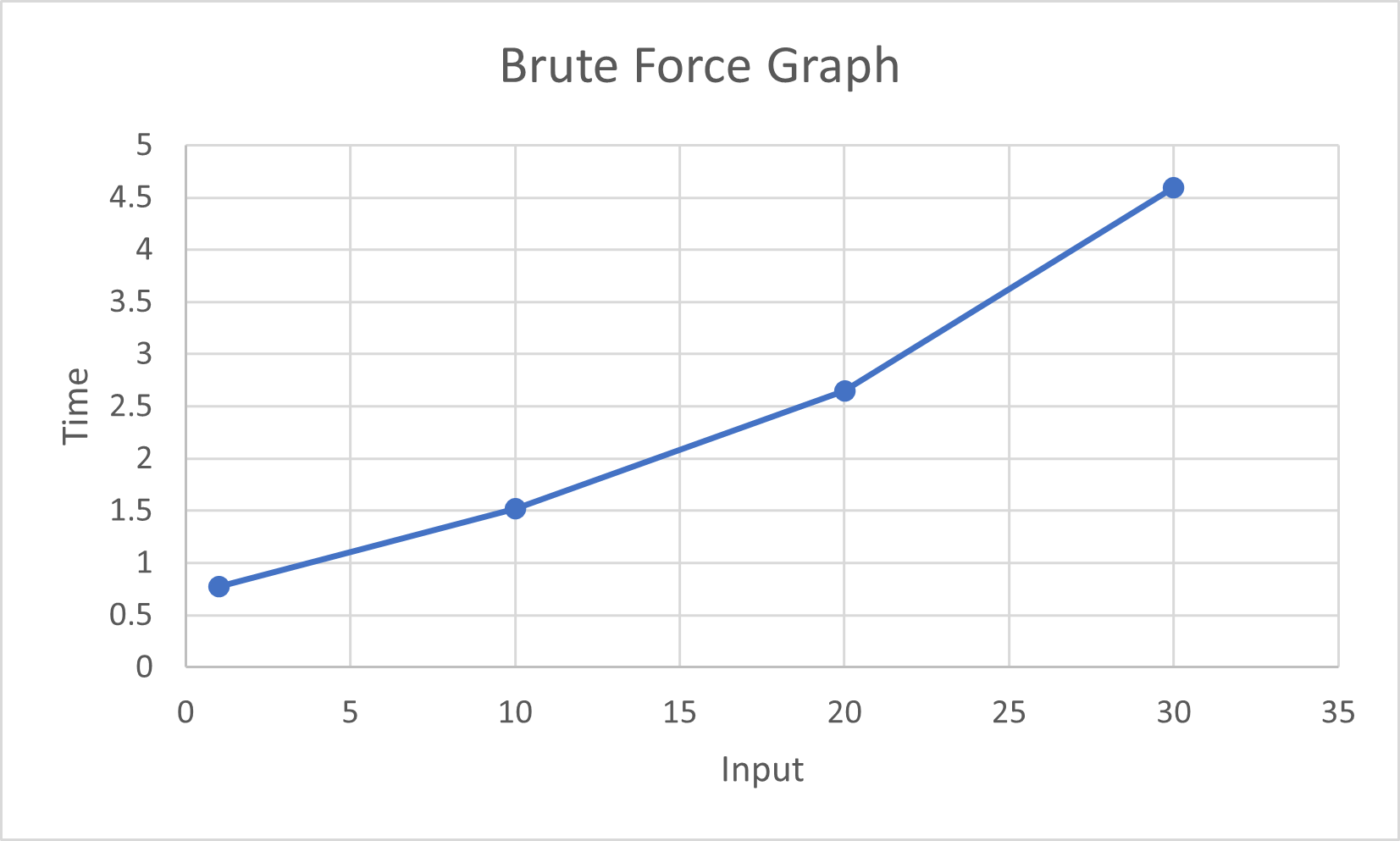
1. Brute Force Approach

Time complexity of brute force approach is O(2^n). This is because the given code on recursion runs as Fibonacci sequence. As Fibonacci sequence is increasing function exponentially closely nearer to the power of 2. Hence by master theorem, the complexity of this brute force recursion comes out to be 2^n.

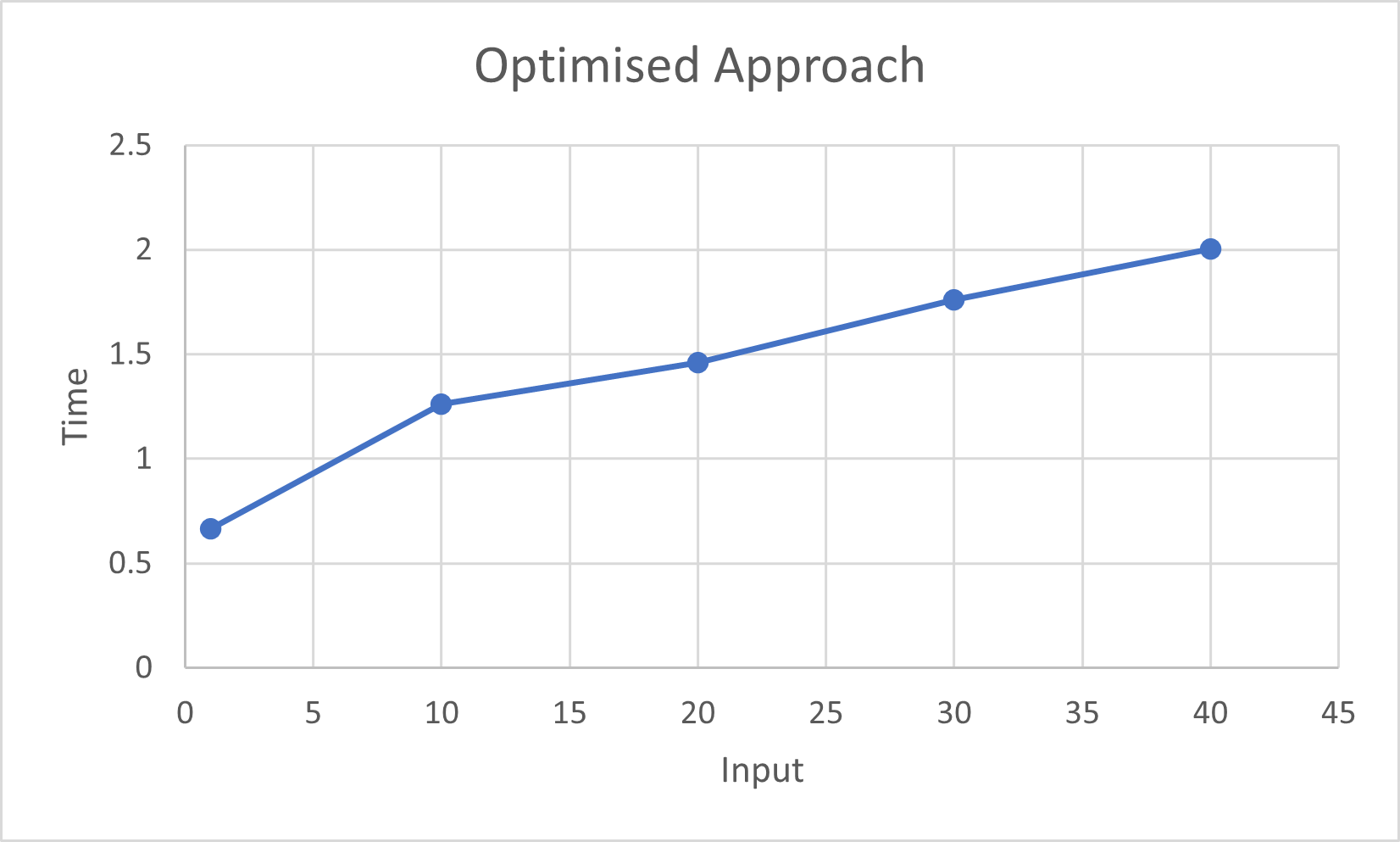
1. Optimised Approach

Time complexity of optimised approach O(n). The reason for this is the usage of dynamic programming approach. We create an array of size(n+1) and store the Fibonacci number of each number. This is calculated in O(1) time and the results are stored. This justifies a time complexity of O(n).

Brute Force Approach

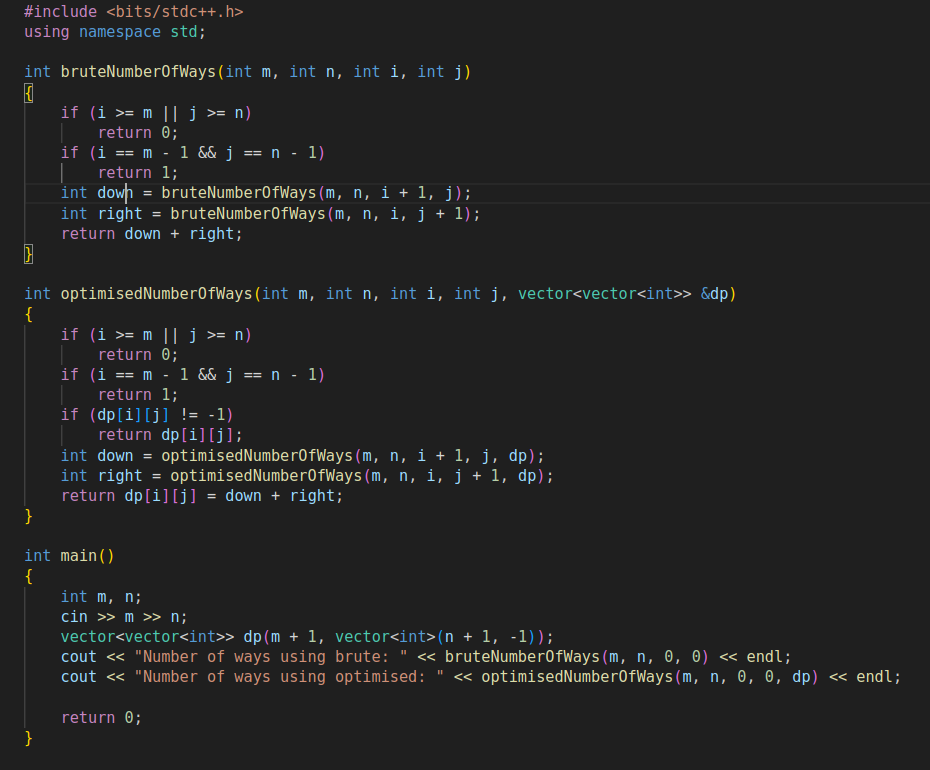


Optimised Approach



Q2 There is a bot on an m x n matrix. The bot is initially located at the top-left corner (i.e., matrix[0][0]) and tries to move to the bottom-right corner (i.e., matrix[m -1][n - 1]). The bot can only move either down or right at any point in time. Given the two integers m and n, return the number of possible unique paths that thebot can take to reach the bottom-right corner.

**Brute Force and Optimised Approach:**

****

**Time Complexity Analysis:**

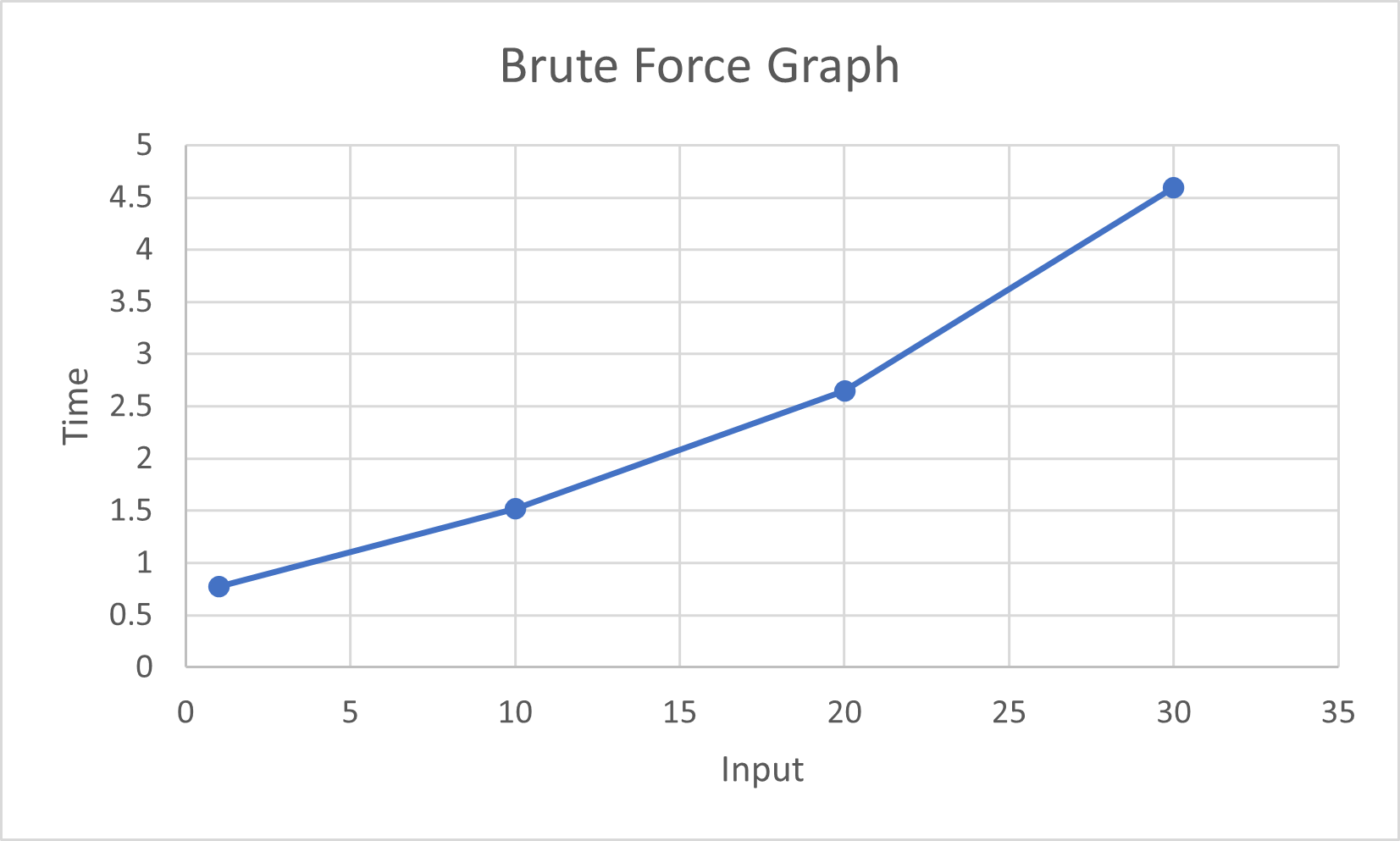
1. Brute Force Approach:

Time complexity of brute force approach is O(2^n). This is because for every cell in the matrix, bot has two options to continue his journey. He can either go to the bottom or go to the right. Thus at every cell he has two choices and for a size n , he would have 2^n choices thus making the complexity exponential.

1. Optimised Approach

The approach for this is O(n\*m) .This is because we are using an 2D array to store the no of possible paths for a particular cell. As we reach that cell , we simply access the ans stored thus making very less computations and giving it a complexity of O(n\*m).

Brute Force Approach:



Optimised Approach:

