VIDYAVARDHAKA COLLEGE OF ENGINEERING

(Autonomous, Affiliated to VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

OPEN ENDED EXPERIMENT ON

“Kadane’s Algorithm for Subarray Sum”

Submitted in partial fulfillment of the requirement for the completion of 4th semester of

‘BACHELOR OF ENGINEERING’

Submitted by:

NAME:Ayush NP’ U4ã 4VV23CS030

NAME:Bandla Lohith’ U4ã 4VV23CS031

Under the guidance of:

Dr. Ayesha T

Associate Professor

Dept of CSE, VVCE

“COMPUTER SCIENCE AND ENGINEERING”

1. Introduction

The Maximum Subarray Problem is a famous problem in computer science, which requires finding the contiguous subarray within a one-dimensional array of numbers that has the largest sum.

This problem is important in various real-world applications like analyzing stock prices, financial modelling, signal processing, and finding patterns in data streams.

Kadane’s Algorithm is a dynamic programming technique that solves this problem efficiently in linear time. It smartly updates the maximum sum encountered while iterating through the array, making it one of the most optimal solutions.

2. Problem Statement

Given an array arr[] of integers (which may include both positive and negative

numbers), the goal is to find the contiguous subarray which has the largest sum

and return that sum.

3.Approach: Kadane’s Algorithm (Dynamic Programming)

Kadane’s algorithm works based on two ideas:

At each position, we either extend the previous subarray or start a new subarray at the current element.We keep track of two variables:current_sum !’ maximum sum ending at the current index.max_sum !’ overall maximum sum found so far.Steps:

1. Initialize current_sum = max_sum = arr[0].

2. Traverse the array from the second element:

Update current_sum as the maximum of the current element andcurrent_sum + current element.Update max_sum if current_sum is greater than max_sum.3. At the end of the traversal, max_sum contains the answer.

This avoids checking all possible subarrays (which would be very slow) and instead solves the problem in O(n) time.

4. Python Code Implementation:

```
def kadane(arr):

n = len(arr)

current_sum = max_sum = arr[0]

for i in range(1, n):

current_sum = max(arr[i], current_sum + arr[i])

max_sum = max(max_sum, current_sum)

return max_sum

# Example usage

arr = [-2, 1, -3, 4, -1, 2, 1, -5, 4]

print("Maximum subarray sum is:", kadane(arr))
```

Output:

```
Maximum subarray sum is: 6
```

5. Performance Graph:

The following conceptual graph illustrates the performance:

Y-axis: Execution Time (arbitrary units)X-axis: Number of elements (n)Algorithm

Growth

Kadane's Algorithm

Linear O(n)

Brute-force Approach

Quadratic $O(n^2)$

Kadane’s algorithm grows linearly with input size.A brute-force method that checks all subarrays would grow quadratically and be much slower for large arrays.

6.Comparison Table:

Factor

Kadane's Algorithm

Brute-force Method

Time Complexity

O(n)

O(n2)

Optimality

Always optimal

Always optimal

Scalability

Highly scalable

Poor for large inputs

Speed

Fast

Very slow for large n

Ease of

Implementation

Simple

Simple but inefficient

7. Analysis:

Kadane’s Algorithm is highly efficient because it reduces the maximum

subarray problem to a simple iteration through the array with constant extraspace. It is a classic example of dynamic programming where we solve acomplex problem by breaking it into simpler subproblems.

While brute-force methods are easy to understand, they are computationally expensive and impractical for large datasets.

Kadane’s Algorithm is not only faster but also memory-efficient, making it idealfor competitive programming and real-world applications.

8. Conclusion:

The Maximum Subarray Problem shows the importance of optimization in

algorithm design. Kadane's Algorithm provides a powerful and efficient solution

with linear time complexity.

Its simplicity and speed make it a preferred choice in real-world scenarios whereperformance matters. Understanding Kadane’s algorithm also builds strong

foundational knowledge about dynamic programming, greedy choices, and

problem-solving strategies.

Thus, Kadane’s Algorithm stands out as one of the most elegant and efficient

solutions for subarray sum problems.