Experiment Number: 02

Comparing Brute Force Approach & Divide & Conquer

Maximum Sub Array Problem

Aim:

Given an array of n numbers includes mix of positive and negative numbers, the task is to calculate the maximum subarray sum. In other words, the goal is to find the largest possible sum of sequence of consecutive values in the array. To apply, implement and compare the results of brute-force and divide & conquer approach in solving maximum sub array problem.

Algorithm(s):

(a) Maximum Sub Array – Brute Force Approach

```
Algorithm MaxSubArray_BruteForce(A[1..n])
    MSum = A[1]
    MRange = (1,1)
    For i←1 to n do
        Sum ← 0
        For j←i to n do
            Sum ← Sum + A[j]
            If Sum > MSum then
                MSum = Sum
                MRange = (i,j)
            End If
        End For
    End For
    Return MSum, MRange
End MaxSubArray_BruteForce
```

(b) Maximum Sub Array – Divide & Conquer Approach

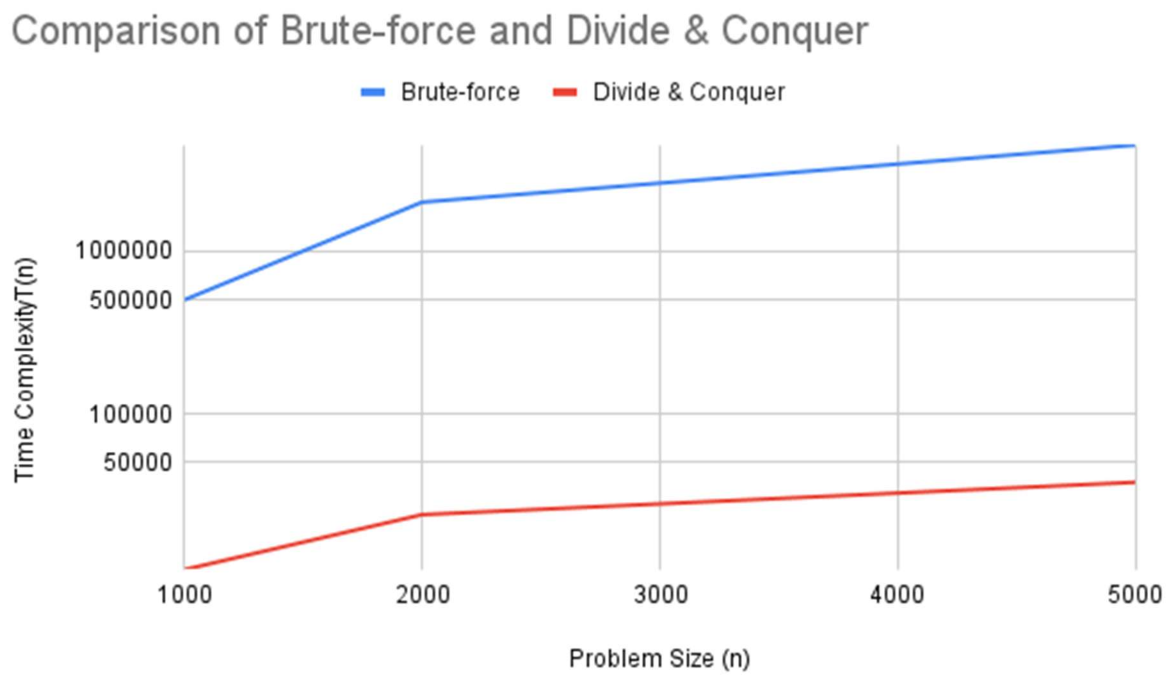
```
Algorithm MaxSubArray_DC(A[1..n], low, high)
    //Base Case
    If low = high then
        Return (low, high, A[low])
    End If

    //Recursive Case
    mid ← ⌊( low + high ) / 2⌋
    (low1, high1, sum1) ← MaxSubArray_DC(A, low, mid)
    (low2, high2, sum2) ← MaxSubArray_DC(A, mid+1, high)
    (low3, high3, sum3) ← MidCrossingSubArray(A, low, mid, high)

    //Returning the sub array with maximum sum.
    If sum1>sum2 and sum1>sum3 then
        Return (low1, high1, sum1)
    Else If sum2>sum3 then
        Return (low2, high2, sum2)
    Else
        Return (low3, high3, sum3)
    End If
End MaxSubArray_DC


Algorithm MidCrossingSubArray(A[1..n], low, mid, high)

    //Finding Left Maximum Sum
    LSum ← A[mid]
    LMaxIndex ← mid

    Sum ← A[mid]
    For i←mid-1 to low do downwards
        Sum ← Sum + A[i]
        If Sum>LSum then
            LSum ← Sum
            LMaxIndex ← i
        End If
    End For

    //Finding Right Maximum Sum
    RSum ← A[mid+1]
    RMaxIndex ← mid+1

    Sum ← A[mid+1]
    For i←mid+2 to high do
        Sum ← Sum + A[i]
        If Sum>RSum then
            RSum ← Sum
            RMaxIndex ← i
        End If
    End For

    //Returning the sub array with mid element and maximum sum
    Return (LMaxIndex, RMaxIndex, LSum+RSum)

End MidCrossingSubArray
```

Results & Discussion:

Comparison Table

Size (n)	Number of Active Operations	
	Bruteforce Method	Divide & Conquer Approach
1000	500500	10975
2000	2001000	23951
3000	4501500	37903

Comparison Chart



Conclusion

1. For the Maximum Sub Array Problem, applied brute-force method and divide & conquer approach.
2. Irrespective of input values, divide & conquer approach provides better performance compared to brute-force method.