Design and Analysis of Algorithms Assignment - 1

Department of Information Technology

Indian Institute of Information Technology - Allahabad, India

Ritu Kiran Murmu(IIB2019025) Atithi kumari(IIB2019026) Shahid(IIB2019027)

Abstract: ***In this paper ,we have designed an algorithm to find the area of n- sided polygon. We have used the divide and conquer approach to solve the problem and also analysed the time and space complexity of the program.***

Index Terms: ***Arrays, Divide and Conquer***

INTRODUCTION

We have been given set of ordered points in 2D plane.An Array is a linear data structure which stores similar elements in contiguous memory locations. A Shoelace formula has been used on the array.

Divide and Conquer In divide-and-conquer we try to decompose a given problem into two or more similar, but simpler, sub-problems, to solve them in turn, and to compose their solutions to find optimal solution of the problem. So, basic idea here is:

1. **Divide:** This involves dividing the problem into some sub problem.

2. **Conquer:** Sub problem by calling recursively until sub problem solved.

3. **Combine:** The Sub problem Solved so that we will get find problem solution.Let us have a pictorial look of the image which makes the concept of Divide and Conquer Paradigm self explanatory.

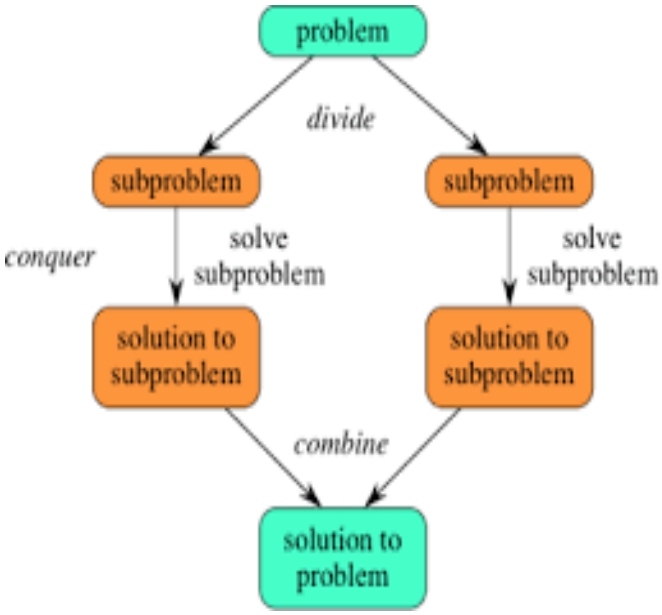

Figure 1: Divide and Conquer

Advantages of Divide and Conquer The Divide and Conquer technique generally has following mentioned advantages over Brute Force Algorithms.

Time Efficiency This approach reduces the running time of the algorithm because of its logarithmic nature. This is because the approach of Divide and Conquer keeps on dividing which makes it logarithmic in nature.

Space Efficiency: As no need to mention the algorithm, but still, the space complexity also reduces up to a much extent due to the nature of dividing and then merging up.

This report further contains:

- Algorithm Designs
- Algorithm Analysis
- Experimental Study and Profiling
- Conclusion
- References
- Appendix

ALGORITHM DESIGN

A general problem based on the divide and conquer approach in which, we divided the problem into sub tasks and then make decisions while capturing the answer and merging. The data structure used in finding the pair of closest points are the standard and primary data structure of the computational domain. Following structures have been taken into account while making the code for the report.

Data Structures Used:

- Structure to store Point
- Array to store the set of Points

Figure 2: Divide and Conquer

Algorithmic Steps: Basically, the approach based on Divide and Conquer paradigm of finding the closest pair points generally has the steps defined in following algorithmic procedure, which have been implemented while solving the problem:

1. Input ordered coordinates i.e. coordinates should be given either in a clockwise manner or anticlockwise from the first vertex to last.
2. Create a function polyArea, which will calculate area of Polygon by dividing them into smaller polygons, until the polygon is reduced to triangle.
3. Recursively, the area of parallelogram of corresponding triangle is founded by taking cross product of vertices.
4. We are storing the area say in variable ans.
5. On returning ans, we are dividing the whole area by 2 to get the area of triangles formed earlier.

The value of ans will give the area of polygon whose vertices are in 2D plane.

```
Double:
Function polyArea(v, int i, int j)
    int length  = j - i + 1
    if length == 2
        double ans = v[i].first*v[j].second -
        v[j].first*v[i].second;
        return ans;

    mid = i+((j-i)/2);
    double area1 = polyArea(v,i,mid);
    double area2 = polyArea(v,mid,j);
    return area1 + area2;



Int:
Function main()
    int n
    Input n

    if n <= 2
        print no polygon exist
        return 0

    points :array

    loop i=0 to n with i++
        Input Points[i]

    To connect last and first point
    push Point[0] to last index
    double ans = polyArea(points,0,n)
    ans = ans/2.0
    print absolute value of ans
    return 0
```

ALGORITHM ANALYSIS

Time complexity Derivation: Let Time complexity of above algorithm be T(n) and it can be expressed as follows:

TIME COMPLEXITY DERIVATION:

T(n) = 2T(n/2) + O(n) + O(n) + O(n)
T(n) = 2T(n/2) + O(n)

Using above relation, we get for T/2, T/8 etc as:
T(n/2) = 2T(n/4) + O(n/2)
T(n/4) = 2T(n/8) + O(n/4)
T(n/8) = 2T(n/16) + O(n/8) and so on.....
.
.
.
.
Thus on combining we get the overall time complexity as:T(n) = T(nlogn)

Time Analysis: Following is the graph representing the time complexity of the algorithm.

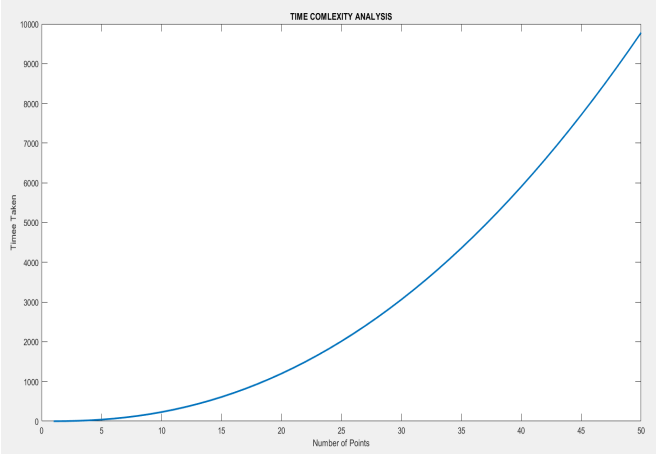

Figure 3: Time Complexity Graph

By the experimental analysis, we found that in case of optimized approach, on increasing the no. of points in the set (increasing the size of set) the graph is strictly increasing with a bend (or concavity) towards horizontal axis. Thus the overall time increases with an increase in no. of points.

Space Analysis: Following is the graph representing the space complexity of the algorithm.

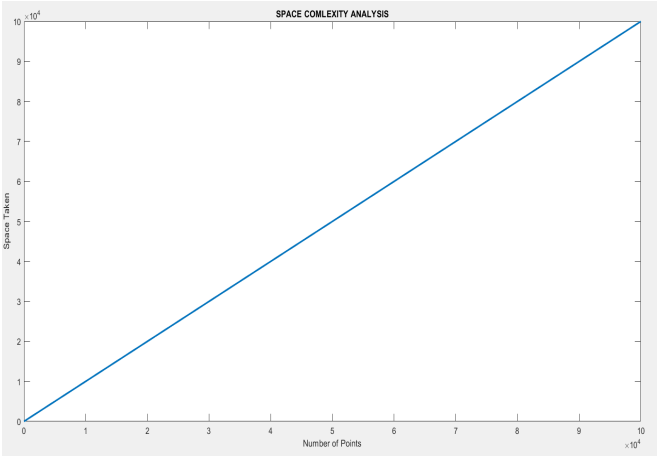

Figure 4: Space Complexity Graph

By the experimental analysis, we found that in case of optimized approach, on increasing the no. of points in the set (increasing the size of set) the graph is strictly increasing Thus the overall space increases with an increase in no. of points.

CONCLUSION: So, with the above mentioned algorithms and their profiling, we come to the conclusion that this classical problem of finding the area of n-sided polygon is achieving its best time complexity of O(nlogn) and space complexity of O(n).

Hence this is one of the most effiicient way to find area of polygon.

REFERENCES

1. GeeksforGeeks
2. Shoelace$_formula$