

Third Year B. Tech., Sem V 2022-23

Design and Analysis of Algorithm Lab

Assignment / Journal submission

PRN/ Roll No: 21520010

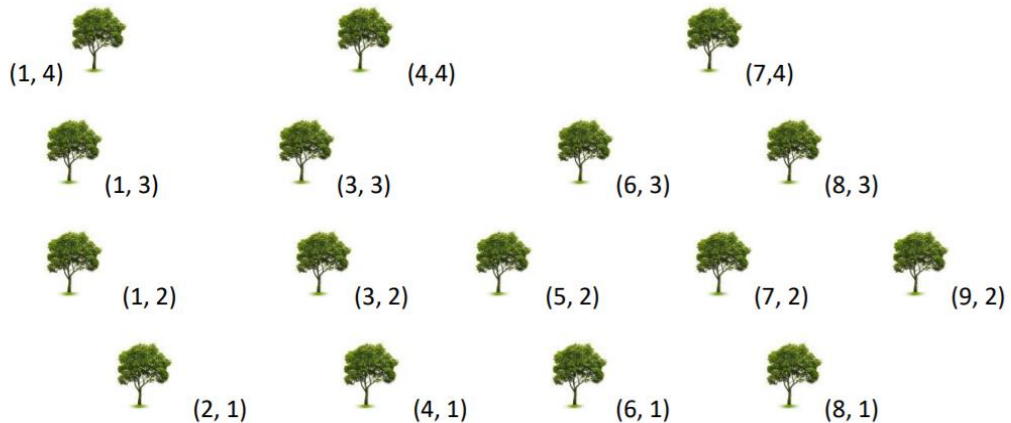
Full name: Mayur Sunil Savale

Batch: T8

Assignment: Week 5

Title of assignment: Divide and Conquer Strategy

1. Farmer has planted only mango trees in his farm as shown in figure (indicates position of each tree with x, y coordinates) It is not fixed size farm. He wants to protect those mango trees from a thief, for that he decided to round up a tree with electric wire. But as the farm is not of fixed size, he decided to find all set of mango trees which will cover all the remaining mango trees and then he will wind all tree with electric wire.



Implement an algorithm to find set of mango trees which cover all remaining mango trees. Also find perimeter of rounded wire.

Ans:

a) Algorithm: (Pseudocode)

- Add points.
- Sort points lexicographically.
- Build lower hull.
- If the point at K-1 position is not a part of hull as vector from $ans[k-2]$ to $ans[k-1]$ and $ans[k-2]$ to $A[i]$ has a clockwise turn.
- Build upper hull.
- If the point at K-1 position is not a part of hull as vector from $ans[k-2]$ to $ans[k-1]$ and $ans[k-2]$ to $A[i]$ has a clockwise turn.
- Return a list of points on the convex hull in counter-clockwise order.
- Return the points and distance between the points.

b) Code snapshots of implementation

```
#include <bits/stdc++.h>
#define ll long long int
using namespace std;

struct Point
{
    ll x, y;

    bool operator<(Point p)
    {
        return x < p.x || (x == p.x && y < p.y);
    }
};

ll cross_product(Point O, Point A, Point B)
{
    return (A.x - O.x) * (B.y - O.y) - (A.y - O.y) * (B.x - O.x);
}

vector<Point> convex_hull(vector<Point> A)
{
    int n = A.size(), k = 0;

    if (n <= 3)
        return A;

    vector<Point> ans(2 * n);

    sort(A.begin(), A.end());

    for (int i = 0; i < n; ++i)
    {
        while (k >= 2 && cross_product(ans[k - 2],
```

```

        ans[k - 1], A[i]) <= 0)
        k--;
        ans[k++] = A[i];
    }

    for (size_t i = n - 1, t = k + 1; i > 0; --i)
    {
        while (k >= t && cross_product(ans[k - 2],
            ans[k - 1], A[i - 1]) <= 0)
            k--;
        ans[k++] = A[i - 1];
    }

    ans.resize(k - 1);

    return ans;
}

double dist(Point a, Point b)
{
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}

double perimeter(vector<Point> ans)
{
    double perimeter = 0.0;

    for (int i = 0; i < ans.size() - 1; i++)
    {
        perimeter += dist(ans[i], ans[i + 1]);
    }

    perimeter += dist(ans[0], ans[ans.size() - 1]);

    return perimeter;
}

```

```

}

int main()
{
    vector<Point> points;

    points.push_back({1, 2});
    points.push_back({1, 3});
    points.push_back({1, 4});
    points.push_back({2, 1});
    points.push_back({3, 2});
    points.push_back({3, 3});
    points.push_back({4, 1});
    points.push_back({4, 4});
    points.push_back({5, 2});
    points.push_back({6, 1});
    points.push_back({6, 3});
    points.push_back({7, 2});
    points.push_back({7, 4});
    points.push_back({8, 1});
    points.push_back({8, 3});
    points.push_back({9, 2});

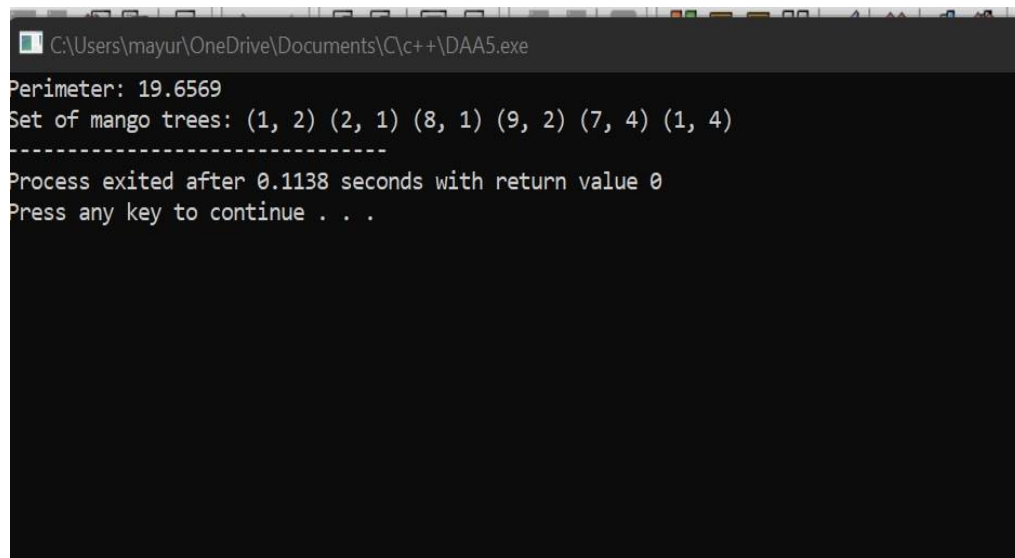
    vector<Point> ans = convex_hull(points);

    cout << "Perimeter: " << perimeter(ans) << endl;

    cout << "Set of mango trees: ";
    for (int i = 0; i < ans.size(); i++)
    {
        cout << "(" << ans[i].x << ", " << ans[i].y << ")"
            << " ";
    }
    return 0;
}

```

Output:



```
C:\Users\mayur\OneDrive\Documents\C\c++\DAA5.exe
Perimeter: 19.6569
Set of mango trees: (1, 2) (2, 1) (8, 1) (9, 2) (7, 4) (1, 4)
-----
Process exited after 0.1138 seconds with return value 0
Press any key to continue . . .
```

c) Complexity of proposed algorithm (Time & Space)

- Time Complexity: $O(n \log n)$

d) Your comment (How your solution is optimal?)

- We construct the convex hull in $O(n \log n)$ time. We have to sort the points first and then calculate the upper and lower hulls in $O(n)$ time. We will then find the perimeter of the convex hull using the points on the convex hull which can be done in $O(n)$ time as the points are already sorted in clockwise order.