Walchand College of Engineering, Sangli
Computer Science & Engineering
Third Year
Course: Design and analysis of algorithm Lab
Lab course coordinator:
Dr. B. F. Momin- Batch: - T6, T7, T8
Mr. Kiran P. Kamble- Batch: - T1, T2, T3, T4, T5

Week 5 Assignment

Part: 3

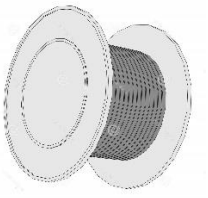
Divide and conquer strategy

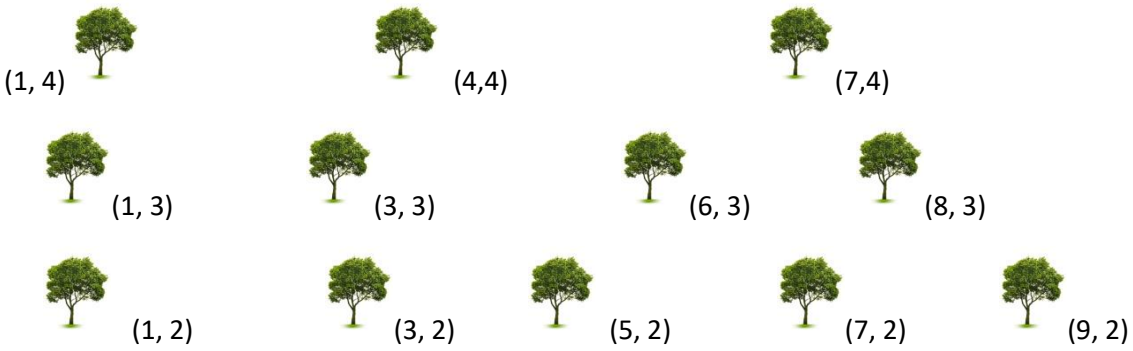
Name: Trupti Rajendra Patil

Prn: 2020BTECS00051

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 a 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.





 (2, 1)  (4, 1)  (6, 1)  (8, 1)

Algorithm:

1. Find the bottom-most point with minimum y coordinate. If y coordinates are same then check x coordinate value.
2. Swap the minimum point with first point of given set of points.
3. Sort remaining n-1 points according to polar angle (in counter clockwise direction) w.r.t first point i.e., Minimum point
   If points have same orientation, then keep farthest point at end.
4. If two or more points make same angle with bottom point, then remove all except farthest point from bottom point.
5. If size of set of points is less than 3 then return.
6. Create a stack and push first 3 points
7. Process the remaining points.
   1) Keep removing points from stack while orientation of following 3 points is not counter clockwise (or they don't make a left turn).
      a) Point next to top in stack
      b) Point at the top of stack
      c) points[i]
   2) Push points[i] to S
8. Print the final points

Code:

```
#include<bits/stdc++.h>
using namespace std;

// vector<pair<int,int>> points;
pair<int,int>  p0;

// function to get next point of top of stack
pair<int,int> nexttoTop(stack<pair<int,int>>& s){
    pair<int,int> pr=s.top();
    s.pop();
    pair<int,int> nxt=s.top();
    s.push(pr);
    return nxt;
}
```

```
// function to calculate distance
int distanceSqr(pair<int,int> p1,pair<int,int> p2){
    return (p1.first - p2.first)*(p1.first - p2.first)+
           (p1.second - p2.second)*(p1.second - p2.second);
}

/*  0 = x, y, z are collinear
    1 = clockwise
    2 = counterclockwise
*/
int orientation(pair<int,int> x,pair<int,int> y,pair<int,int> z){
    int val=(y.second - x.second)*(z.first - y.first)-
            (y.first - x.first)*(z.second - y.second);

    if(val == 0){
        return 0; //collinear
    }

    return (val > 0) ? 1 : 2; //clockwise counterclockwise
}

// function to sort the points
int compare(const void *a,const void *b){
    pair<int,int>* pa = (pair<int,int>*) a;
    pair<int,int>* pb = (pair<int,int>*) b;

    int o = orientation(p0,*pa,*pb);

    if(o == 0){
        return ((distanceSqr(p0,*pb) >= distanceSqr(p0,*pa)) ? -1 :
1);
    }
    return (o == 2) ? -1 : 1;
}

int main(){
    vector<pair<int,int>> points =
{{1,4},{4,4},{7,4},{1,3},{3,3},{6,3},{8,3},{1,2},
                                    {3,2},{5,2},{7,2},{9,2},{2,1},{4,
1},{6,1},{8,1}};
    int n=points.size();
```

```
    // Finding bottom most point
    int ymin=points[0].second,min=0;
    for(int i=0;i<points.size();i++){
        int y=points[i].second;

        if((y < ymin) || (y == ymin && points[i].first <
points[min].first)){
            ymin = points[i].second;
            min = i;
        }
    }

    // Swap bottom point with first point in set
    swap(points[0],points[min]);

    /* Sort the points according to polar angle w.r.t first/bottom
point and
     keep farthest point at end if points have same orientation */
    p0 = points[0];
    // cout<<sizeof(pair<int,int>)<<" "<<n-1<<endl;
    qsort(&points[1],n-1,sizeof(pair<int,int>),compare);

    /*If 2 or more points are collinear than remove all the points
except the furthest point
    from the bottom-most point */
    int m=1;
    for(int i=0;i<n;i++){
        while(i<n-1 &&  orientation(p0,points[i],points[i+1])==0){
            i++;
        }

        points[m++]=points[i];
    }

    // if size is less than 3 then convex polygon is not possible
    if(m<3){
        cout<<"Solution not possible"<<endl;
        return -1;
    }

    // maintain a stack to store points
    stack<pair<int,int>> s;
```

```
    s.push(points[0]);
    s.push(points[1]);
    s.push(points[2]);

    /* remove the point from the stack if polar angle formed by it
is not on left turn
       else push it in stack */
    for(int i=3;i<m;i++){
        while(s.size()>1 &&
orientation(nexttoTop(s),s.top(),points[i])!=2){
            s.pop();
        }
        s.push(points[i]);
    }

    // Print the points of convex hull
    while(!s.empty()){
        pair<int,int> pt=s.top();
        cout<<"("<<s.top().first<<", "<<s.top().second<<")"<<endl;
        s.pop();
    }

    return 0;
}
```

Complexity Analysis:

Time complexity: O(nlogn)

Space complexity: O(n)

Output:

```
PROBLEMS  OUTPUT  TERMINAL  JUPYTER  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ> cd "c:\Users\trupti patil\OneDrive\Desktop
f ($?) { g++ A5Q1.cpp -o A5Q1 } ; if ($?) { .\A5Q1 }
(1, 2)
(1, 4)
(7, 4)
(9, 2)
(8, 1)
(2, 1)
PS C:\Users\trupti patil\OneDrive\Desktop\ACADEMICS\SEM5\DAA\ExpQ>
```