TRIBHUVAN UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY

Central Department of Computer Science and Information Technology
Kirtipur, Kathmandu
2022

Lab Report: V

"Implementation of Convex Hull"

Submitted By:
Sudhan Kandel
Semester: 2nd
Roll no: 2

Submitted To:
Asst.Prof. Jagdish Bhatta
_____

1. Write a program for finding convex hull using extreme point elimination

```
import matplotlib.pyplot as plt
import numpy as np
class Node:
    def __init__(self,data):
        self.data = data;
        self.previous = None;
        self.next = None;

class DoublyLinkedList:
    def __init__(self):
        self.head = None;
        self.tail = None;
    def addNode(self, data):
        newNode = Node(data);
        if(self.head == None):
            self.head = self.tail = newNode;
            self.head.previous = None;
            self.tail.next = None;
        else:
            self.tail.next = newNode;
            newNode.previous = self.tail;
            self.tail = newNode;
            self.tail.next = None;
    def deleteAllNodes(self):
         while (self.head != None):
            temp = self.head
            self.head = self.head.next
```

```
            temp = None
class ExtremePoint(DoublyLinkedList):
    vertices_set=[]
    def __init__(self):
        super(ExtremePoint, self).__init__()
        a=int(input("Enter Number of Vertices"))
        for i in range(a):
            x,y=input("Please Enter X and Y Cordinates").split(',')
            x=float(x)
            y=float(y)
            self.vertices_set.append([x,y])
    def turntest(self,points):
        area=(points[1][0]-points[0][0])*(points[2][1]-points[0][1])-(points[2][0]-
points[0][0])*(points[1][1]-points[0][1])
        if area>0:
            return "Left"
        elif area<0:
            return "Right"
    def pointinclusion(self,l):
        cur=self.head
        turn=[]
        while cur:
            a=cur.data
            cur=cur.next
            if cur==None:
                cur1=self.head
                while cur1:
```

```
                b=cur1.data
                break
        else:
            b=cur.data
        points=[a,b,l]
        turn.append(self.turntest(points))
    return turn
def pointelimination(self):
    vertices=[]
    n=len(self.vertices_set)
    for i in range(n-1):
        for j in range(n-2):
            if j != i:
                for k in range(n-3):
                    if k != i and k != j:
                        for l in range(n-4):
                            if l != i and l != j and l != k:
                                self.addNode(self.vertices_set[i])
                                self.addNode(self.vertices_set[j])
                                self.addNode(self.vertices_set[k])
                                turn=self.pointinclusion(self.vertices_set[l])
                                result = all(element == turn[0] for element in turn)
                                if (result):
                                    if self.vertices_set[l] in vertices:
                                        pass
                                    else:
                                        vertices.append(self.vertices_set[l])
```

```
                        self.deleteAllNodes()
        nonextreme=vertices
        points=list(self.vertices_set)
        for i in nonextreme:
            points.remove(i)
        return points
    def sorting(self):
        Point=list(self.pointelimination())
        n=len(Point)
        x=[]
        y=[]
        for i in Point:
            x.append(i[0])
            y.append(i[1])
        centroid=[sum(x)/n,sum(y)/n]
        angle=[]
        for i in Point:
            ag=np.degrees(np.arctan2(i[1]-centroid[1], i[0]-centroid[0]))
            angle.append(ag)
        list1, list2 = zip(*sorted(zip(angle, Point)))
        return list2
    def dislpay(self):
        print("\n.....")
        print("Name: Sudhan Kandel","\nRoll No: 2","\nSection: A")
        print("\n.....")
        print("Given random points of the polygon is:\n",self.vertices_set)
        print("\n.....")
```

```
    print("Extreme points are:\n",self. pointelimination())
    print("\n.....")
    print("Sorted vertices are:\n",self.sorting())
def visualization(self):
    x=[]
    y=[]
    points=self.sorting()
    a=[]
    b=[]
    for i in range(len(self.vertices_set)):
        x.append(self.vertices_set[i][0])
        y.append(self.vertices_set[i][1])
    for i in points:
        a.append(i[0])
        b.append(i[1])
    a.append(a[0])
    b.append(b[0])
    fig, axes = plt.subplots(1,3, figsize = (12,4))
    axes[0].scatter(x,y)
    axes[0].grid(True)
    axes[0].set_title("Random points",fontdict={'fontsize':20})
    axes[0].set_xlabel("X-axis")
    axes[0].set_ylabel("Y-axis")
    axes[1].scatter(a,b)
    axes[1].grid(True)
    axes[1].set_title("Extreme Points",fontdict={'fontsize':20})
    axes[1].set_xlabel("X-axis")
```

```
    axes[1].set_ylabel("Y-axis")
    axes[2].plot(a,b,linestyle="--")
    axes[2].scatter(x,y)
    axes[2].grid(True)
    axes[2].set_title("Convex-Hull",fontdict={'fontsize':20})
    axes[2].set_xlabel("X-axis")
    axes[2].set_ylabel("Y-axis")
    extremepoint=ExtremePoint()
    extremepoint.dislpay()
    extremepoint.visualization()
```

OUTPUT:

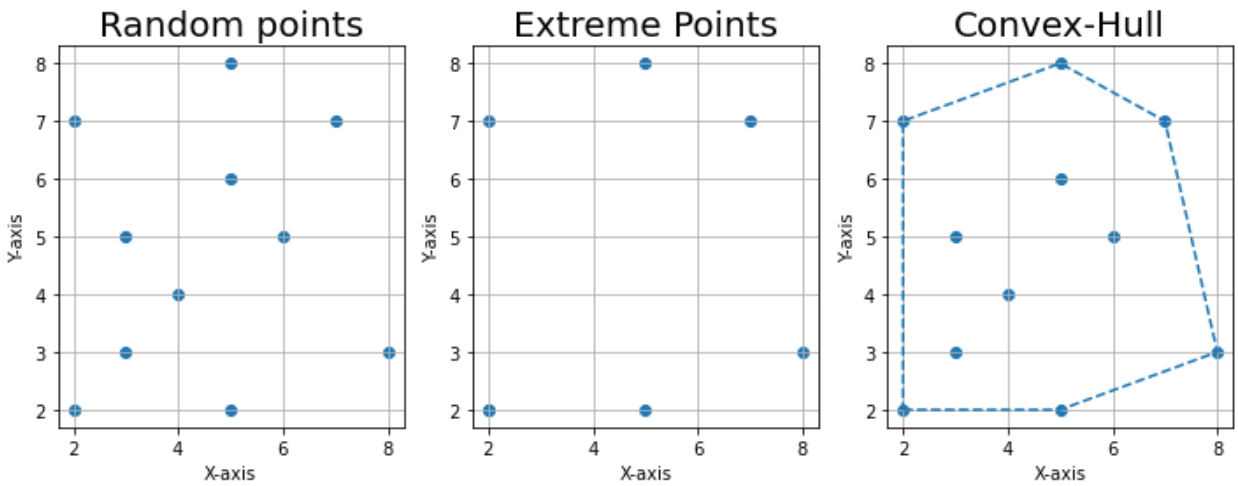
```
Enter Number of Vertices11
Please Enter X and Y Cordinates5,8
Please Enter X and Y Cordinates2,7
Please Enter X and Y Cordinates5,6
Please Enter X and Y Cordinates3,5
Please Enter X and Y Cordinates6,5
Please Enter X and Y Cordinates4,4
Please Enter X and Y Cordinates3,3
Please Enter X and Y Cordinates2,2
Please Enter X and Y Cordinates5,2
Please Enter X and Y Cordinates8,3
Please Enter X and Y Cordinates7,7

..............................
Name: Sudhan Kandel
Roll No: 2
Section: A

..............................
Given random points of the polygon is:
 [[5.0, 8.0], [2.0, 7.0], [5.0, 6.0], [3.0, 5.0], [6.0, 5.0], [4.0, 4.0],
[3.0, 3.0], [2.0, 2.0], [5.0, 2.0], [8.0, 3.0], [7.0, 7.0]]

..............................
Extreme points are:
 [[5.0, 8.0], [2.0, 7.0], [2.0, 2.0], [5.0, 2.0], [8.0, 3.0], [7.0, 7.0]]

..............................
Sorted vertices are:
 ([2.0, 2.0], [5.0, 2.0], [8.0, 3.0], [7.0, 7.0], [5.0, 8.0], [2.0, 7.0])
```



2. Write a program for finding convex hull using extreme edge elimination.

```
import matplotlib.pyplot as plt
import numpy as np
class Node:
    def __init__(self,data):
        self.data = data;
        self.previous = None;
        self.next = None;

class DoublyLinkedList:
    def __init__(self):
        self.head = None;
        self.tail = None;
    def addNode(self, data):
        newNode = Node(data);
        if(self.head == None):
            self.head = self.tail = newNode;
            self.head.previous = None;
            self.tail.next = None;
        else:
            self.tail.next = newNode;
            newNode.previous = self.tail;
            self.tail = newNode;
            self.tail.next = None;
    def deleteAllNodes(self):
        while (self.head != None):
```

```
            temp = self.head
            self.head = self.head.next
            temp = None
class ExtremeEdge(DoublyLinkedList):
    vertices_set=[]
    def __init__(self):
        super(ExtremeEdge, self).__init__()
        a=int(input("Enter Number of Vertices"))
        for i in range(a):
            x,y=input("Please Enter X and Y Cordinates").split(',')
            x=float(x)
            y=float(y)
            self.vertices_set.append([x,y])
    def turntest(self,points):
        area=(points[1][0]-points[0][0])*(points[2][1]-points[0][1])-(points[2][0]-points[0][0])*(points[1][1]-points[0][1])
        if area>0:
            return "Left"
        elif area<0:
            return "Right"
        else:
            return "Colinear"
    def extremedge(self):
        extreme_edges=[]
        n=len(self.vertices_set)
        for i in range(n):
            for j in range(n):
                if j != i:
                    res = [None] * n
                    line = [self.vertices_set[i], self.vertices_set[j]]
                    for k in range(n):
                        p=[self.vertices_set[i], self.vertices_set[j], self.vertices_set[k]]
                        res[k] = self.turntest(p)=='Left' or self.turntest(p)=="Colinear"
                    if set(res) == {True}:
                        extreme_edges.append(line)
        return extreme_edges
    def extremvertex(self):
        extreme_vertex=[]
```

```
        extreme_edges=list(self.extremedge())
        for i in extreme_edges:
            if i[0] in extreme_vertex:
                pass
            elif i[1] in extreme_vertex:
                pass
            else:
                extreme_vertex.append(i[0])
                extreme_vertex.append(i[1])
        return extreme_vertex
    def sorting(self):
        extreme_vertex=self.extremvertex()
        n=len(extreme_vertex)
        x=[]
        y=[]
        for i in extreme_vertex:
            x.append(i[0])
            y.append(i[1])
        centroid=[sum(x)/n,sum(y)/n]
        angle=[]
        for i in extreme_vertex:
            ag=np.degrees(np.arctan2(i[1]-centroid[1], i[0]-centroid[0]))
            angle.append(ag)
        list1, list2 = zip(*sorted(zip(angle, extreme_vertex)))
        return list2
    def dislpay(self):
        print("\n.....")
        print("Name: Sudhan Kandel","\nRoll No: 2","\nSection: A")
        print("\n.....")
        print("Given random points of the polygon is:\n",self.vertices_set)
        print("\n.....")
        print("Extreme points are:\n",self. extremvertex())
        print("\n.....")
        print("Sorted vertices are:\n",self.sorting())
    def visualization(self):
        x=[]
        y=[]
```

```
        points=self.sorting()
        a=[]
        b=[]
        for i in range(len(self.vertices_set)):
            x.append(self.vertices_set[i][0])
            y.append(self.vertices_set[i][1])
        for i in points:
            a.append(i[0])
            b.append(i[1])
        a.append(a[0])
        b.append(b[0])
        fig, axes = plt.subplots(1,3, figsize = (12,4))
        axes[0].scatter(x,y)
        axes[0].grid(True)
        axes[0].set_title("Random points",fontdict={'fontsize':20})
        axes[0].set_xlabel("X-axis")
        axes[0].set_ylabel("Y-axis")
        axes[1].scatter(a,b)
        axes[1].grid(True)
        axes[1].set_title("Extreme Points",fontdict={'fontsize':20})
        axes[1].set_xlabel("X-axis")
        axes[1].set_ylabel("Y-axis")
        axes[2].plot(a,b,linestyle="--")
        axes[2].scatter(x,y)
        axes[2].grid(True)
        axes[2].set_title("Convex-Hull",fontdict={'fontsize':20})
        axes[2].set_xlabel("X-axis")
        axes[2].set_ylabel("Y-axis")
extremeedge=ExtremeEdge()
extremeedge.visualization()
extremeedge.dislpay()
```

OUTPUT:

```
Enter Number of Vertices11
Please Enter X and Y Cordinates5,8
Please Enter X and Y Cordinates7,7
Please Enter X and Y Cordinates2,7
Please Enter X and Y Cordinates5,6
Please Enter X and Y Cordinates3,5
Please Enter X and Y Cordinates6,5
```

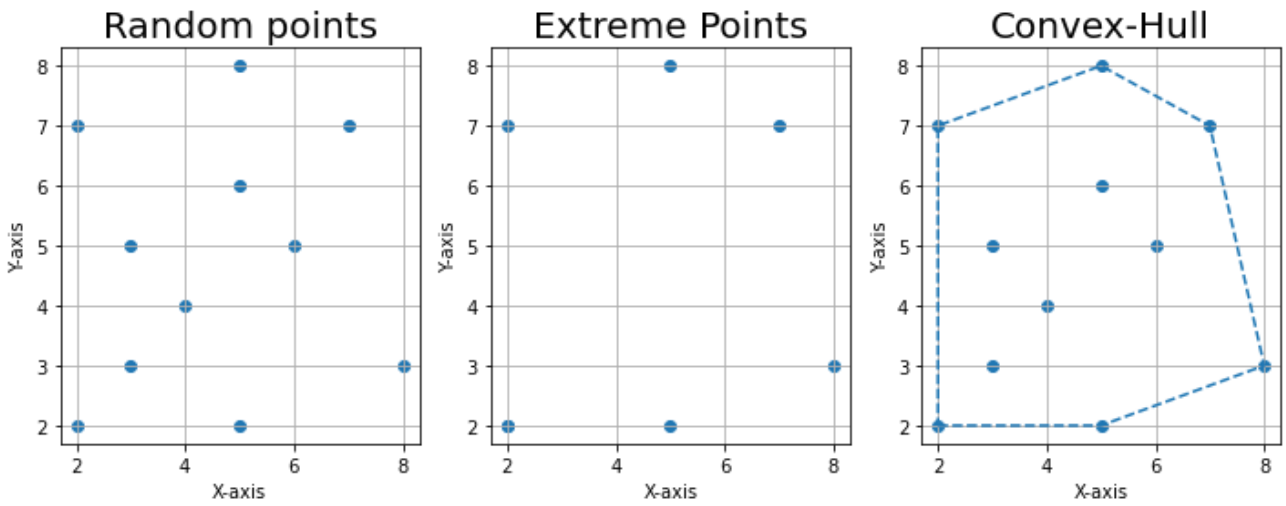
```
Please Enter X and Y Cordinates4,4
Please Enter X and Y Cordinates3,3
Please Enter X and Y Cordinates2,2
Please Enter X and Y Cordinates5,2
Please Enter X and Y Cordinates8,3

.................................
Name: Sudhan Kandel
Roll No: 2
Section: A

.................................
Given random points of the polygon is:
 [[5.0, 8.0], [7.0, 7.0], [2.0, 7.0], [5.0, 6.0], [3.0, 5.0], [6.0, 5.0],
[4.0, 4.0], [3.0, 3.0], [2.0, 2.0], [5.0, 2.0], [8.0, 3.0]]

.................................
Extreme points are:
 [[5.0, 8.0], [2.0, 7.0], [2.0, 2.0], [5.0, 2.0], [8.0, 3.0], [7.0, 7.0]]

.................................
Sorted vertices are:
 ([2.0, 2.0], [5.0, 2.0], [8.0, 3.0], [7.0, 7.0], [5.0, 8.0], [2.0, 7.0])
```



3. Write a program for finding convex hull using Graham Scan Algorithm.

```
import matplotlib.pyplot as plt
import numpy as np
class GrahmScan:
    vertices_set=[]
    def __init__(self):
        a=int(input("Enter Number of Vertices"))
        for i in range(a):
            x,y=input("Please Enter X and Y Cordinates").split(',')
```

```
        x=float(x)
        y=float(y)
        self.vertices_set.append([x,y])
    def sorting(self):
        min_xy=min(self.vertices_set,key=lambda x:(x[1]))
        angle=[]
        for i in self.vertices_set:
            ag=np.degrees(np.arctan2(i[1]-min_xy[1], i[0]-min_xy[0]))
            angle.append(ag)
        list1, list2 = zip(*sorted(zip(angle, self.vertices_set)))
        return list2
    def turntest(self,points):
        area=(points[1][0]-points[0][0])*(points[2][1]-points[0][1])-(points[2][0]-points[0][0])*(points[1][1]-points[0][1])
        if area>0:
            return "Left"
        elif area<0:
            return "Right"
        else:
            return "Colinear"
    def extrem(self):
        stack=[]
        list2=self.sorting()
        stack.append(list2[0])
        stack.append(list2[1])
        i=2
        n=len(self.vertices_set)
        a=len(stack)
        while(i<n):
            points=[stack[-2],stack[-1],list2[i]]
            if self.turntest(points)=="Left":
                stack.append(list2[i])
                i+=1
            else:
                stack.pop()
                points=[stack[-2],stack[-1],list2[i]]
                turn=self.turntest(points)
                while turn=="left":
```

```
                stack.append(list2[i])
    return stack
def dislpay(self):
    print("\n.....")
    print("Name: Sudhan Kandel","\nRoll No: 2","\nSection: A")
    print("\n.....")
    print("Given random points of the polygon is:\n",self.vertices_set)
    print("\n.....")
    print("Sorted vertices are:\n",self.sorting())
    print("\n.....")
    print("Extreme points are:\n",self.extrem())
def visualization(self):
    x=[]
    y=[]
    points=self.extrem()
    a=[]
    b=[]
    for i in range(len(self.vertices_set)):
        x.append(self.vertices_set[i][0])
        y.append(self.vertices_set[i][1])
    for i in points:
        a.append(i[0])
        b.append(i[1])
    a.append(a[0])
    b.append(b[0])

    fig, axes = plt.subplots(1,3, figsize = (12,4))
    axes[0].scatter(x,y)
    axes[0].grid(True)
    axes[0].set_title("Random points",fontdict={'fontsize':20})
    axes[0].set_xlabel("X-axis")
    axes[0].set_ylabel("Y-axis")
    axes[1].scatter(a,b)
    axes[1].grid(True)
    axes[1].set_title("Extreme Points",fontdict={'fontsize':20})
    axes[1].set_xlabel("X-axis")
    axes[1].set_ylabel("Y-axis")
```

```
        axes[2].plot(a,b,linestyle="--")
        axes[2].scatter(x,y)
        axes[2].grid(True)
        axes[2].set_title("Convex-Hull",fontdict={'fontsize':20})
        axes[2].set_xlabel("X-axis")
        axes[2].set_ylabel("Y-axis")
        fig.tight_layout()
        plt.show()
grahmscan=GrahmScan()
grahmscan.dislpay()
grahmscan.visualization()
```

OUTPUT:

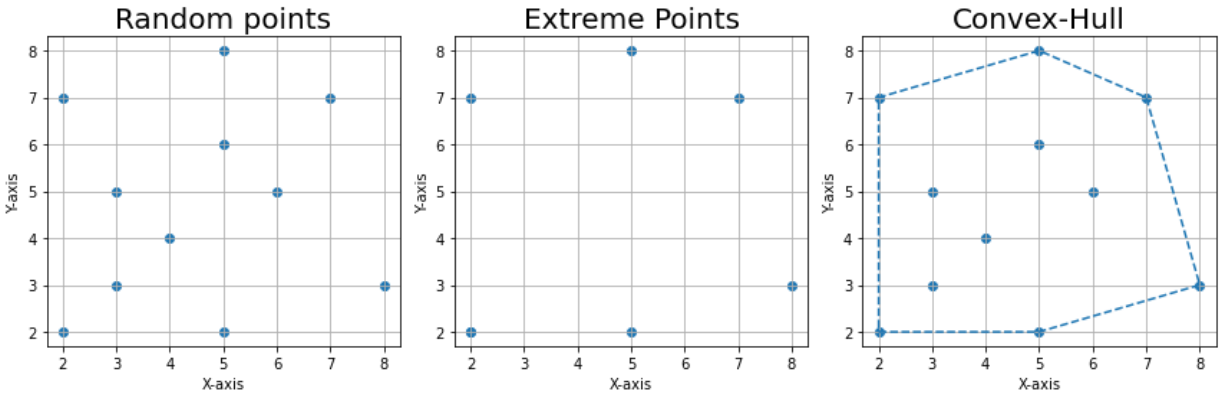
```
Enter Number of Vertices11
Please Enter X and Y Cordinates5,8
Please Enter X and Y Cordinates2,7
Please Enter X and Y Cordinates7,7
Please Enter X and Y Cordinates5,6
Please Enter X and Y Cordinates3,5
Please Enter X and Y Cordinates6,5
Please Enter X and Y Cordinates4,4
Please Enter X and Y Cordinates3,3
Please Enter X and Y Cordinates2,2
Please Enter X and Y Cordinates5,2
Please Enter X and Y Cordinates8,3

..............................
Name: Sudhan Kandel
Roll No: 2
Section: A

..............................
Given random points of the polygon is:
 [[5.0, 8.0], [2.0, 7.0], [7.0, 7.0], [5.0, 6.0], [3.0, 5.0], [6.0, 5.0],
[4.0, 4.0], [3.0, 3.0], [2.0, 2.0], [5.0, 2.0], [8.0, 3.0]]

..............................
Sorted vertices are:
 ([2.0, 2.0], [5.0, 2.0], [8.0, 3.0], [6.0, 5.0], [3.0, 3.0], [4.0, 4.0],
[7.0, 7.0], [5.0, 6.0], [5.0, 8.0], [3.0, 5.0], [2.0, 7.0])

..............................
Extreme points are:
 [[2.0, 2.0], [5.0, 2.0], [8.0, 3.0], [7.0, 7.0], [5.0, 8.0], [2.0, 7.0]]
```



Code Link:
https://github.com/sudhankandel/CG-lab/blob/main/SUDHAN_KANDEL_2_Lab5.ipynb