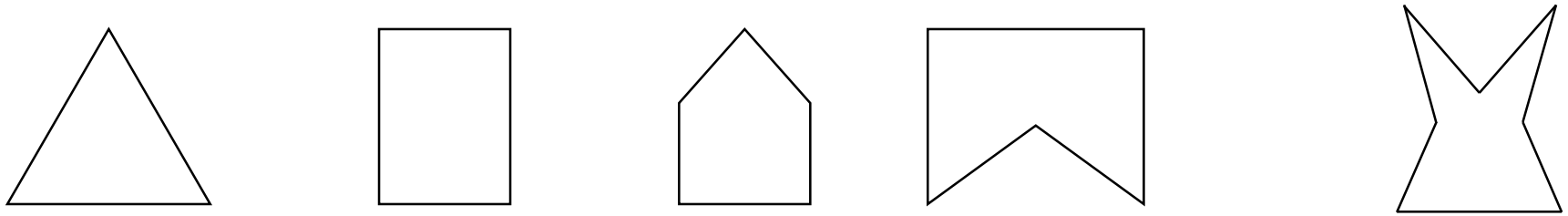
Polygon (Definition)

Polygon is a figure having many slides. It may be represented as a number of line segments end to end to form a closed figure.
The line segments which form the boundary of polygon are called as edges or slides of polygon.
The end of the side are called the polygon vertices.



Triangle is the most simple form of polygon having three side and three vertices.
The polygon may be of any shape.

Types of Polygon

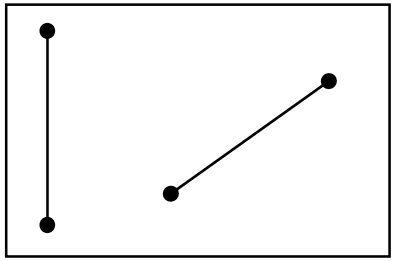
1. Convex

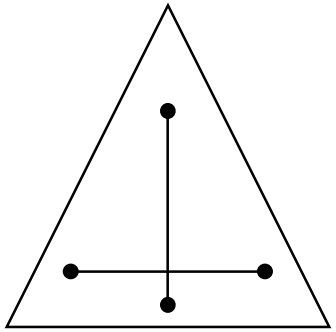
2. Concave

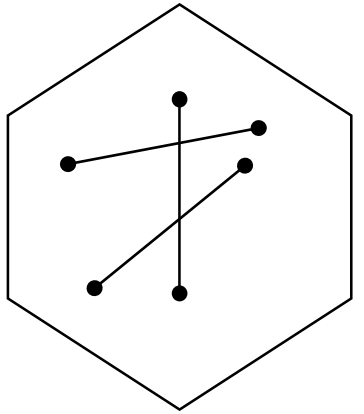
3. Complex

Convex

Convex polygon is a polygon in which if we take any points which are surely inside the polygon and if we draw a joining these two points and if all the points on that line lies into the polygon, then such a polygon is called as convex polygon.

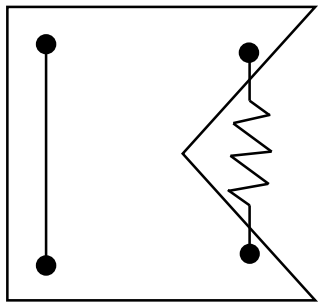


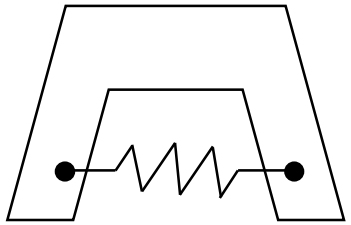


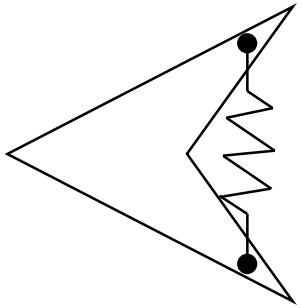


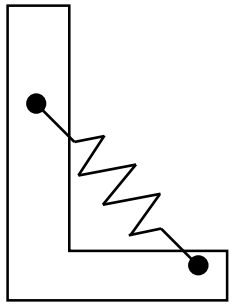
Concave

Concave polygon is a polygon in which if we take any two points which are surely inside the polygon and if we draw a line joining these two points and if all the points on that line are not lying inside the polygon, then such a polygon is called as concave polygon.





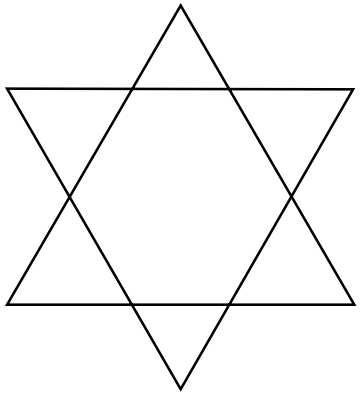


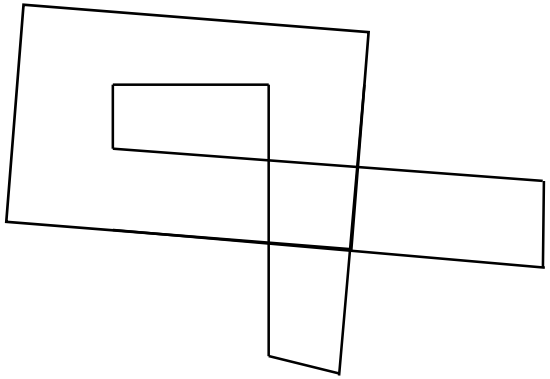


Complex

In a computer graphics, a polygon which is neither convex nor concave is referred as complex polygons.

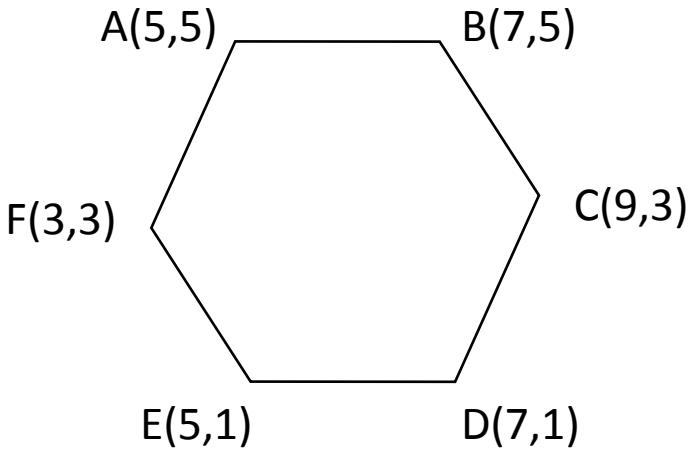
The complex polygon includes any polygon which intersects itself of the polygon which overlaps itself.
The complex polygon has a boundary.





Polygon Representation

OP	X	Y
6	5	5
2	7	5
2	9	3
2	7	1
2	5	1
2	3	3
2	5	5



Inside Test Methods

1. Even-odd Method

2. Winding Number Method

Even - Odd Methods

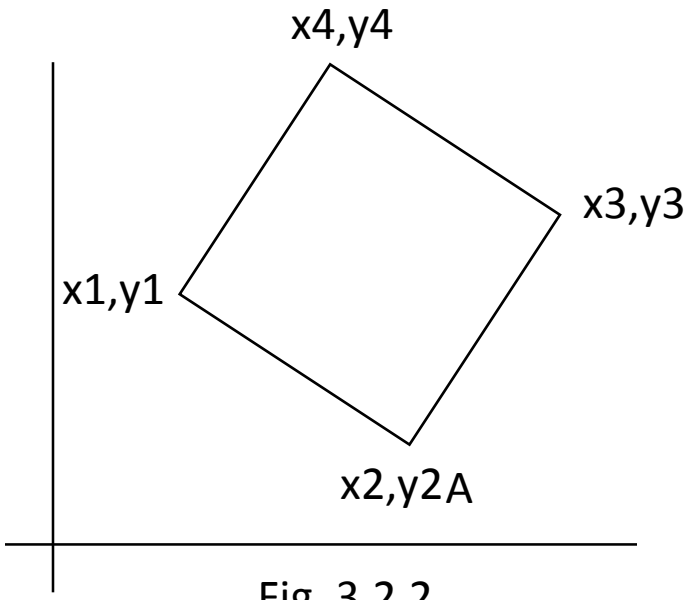


Fig. 3.2.2

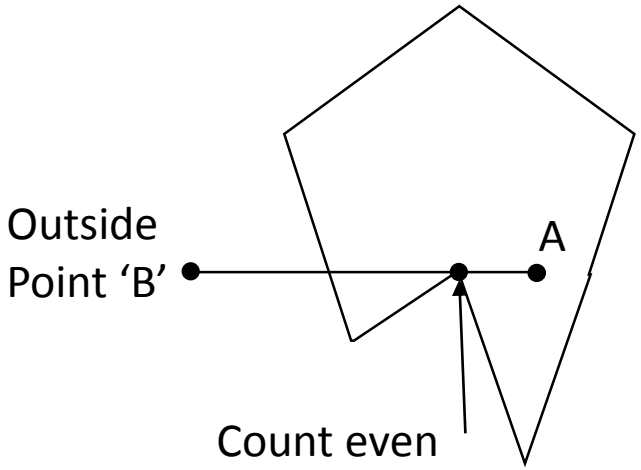


Fig. 3.2.3

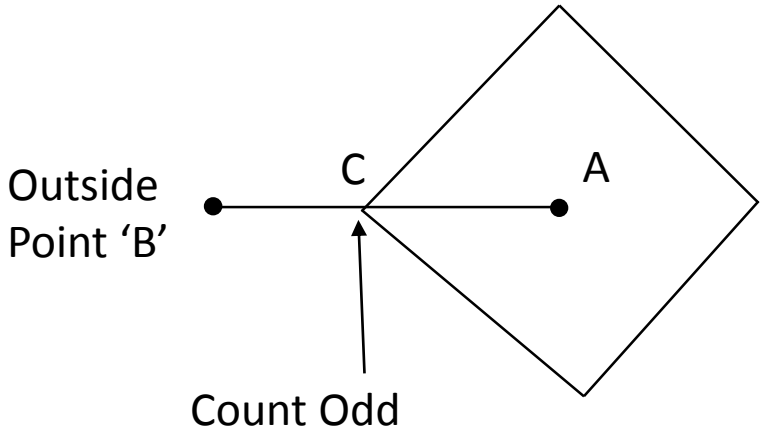
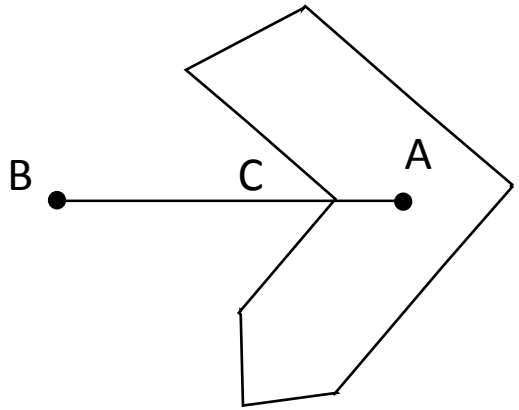


Fig. 3.2.4



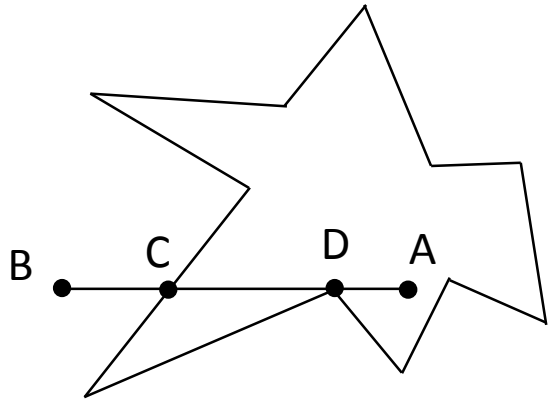


Fig. 3.2.5

Winding Number Methods

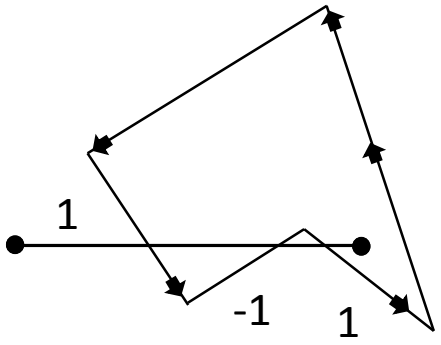


Fig. 3.2.7

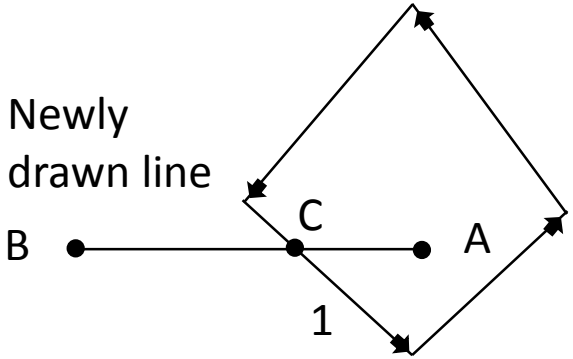


Fig. 3.2.8

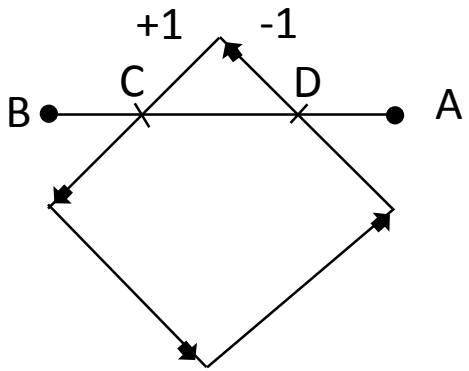


Fig. 3.2.9

Winding Number Methods

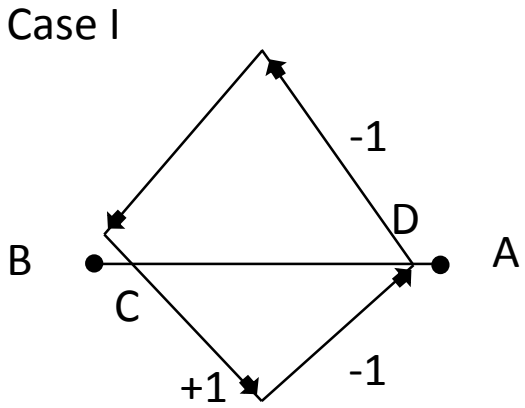


Fig. 3.2.10 (a)

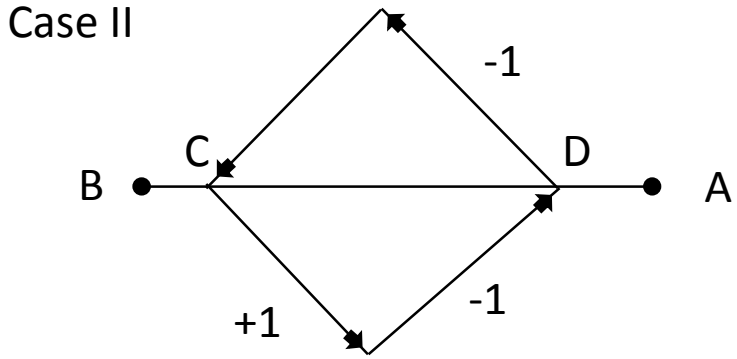


Fig. 3.2.10 (b)

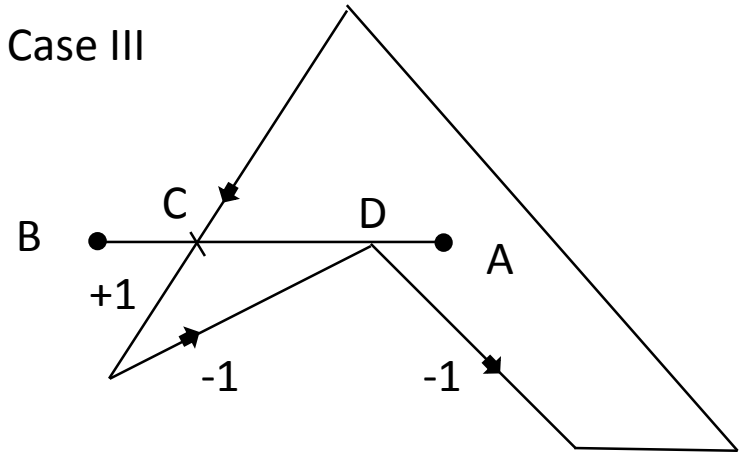
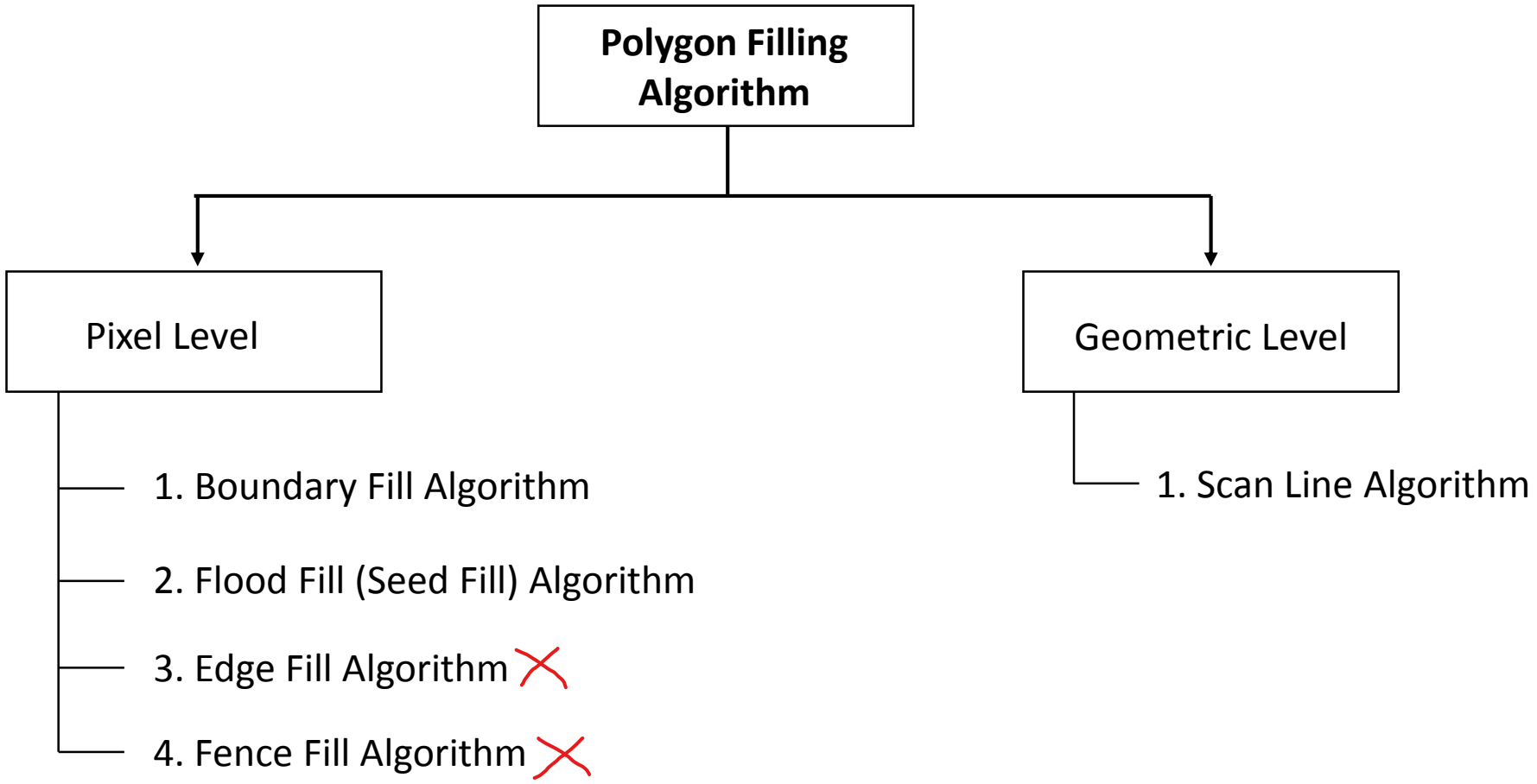


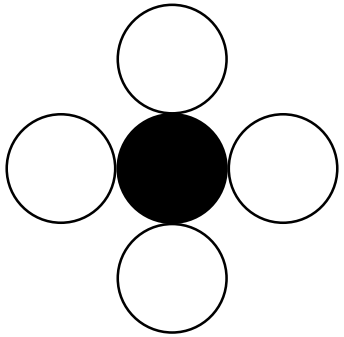
Fig. 3.2.10 (c)

Polygon Filling



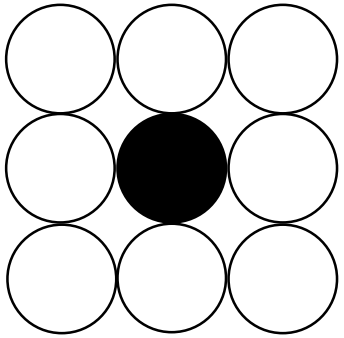
4-connected Method

In this 4 neighboring points of a current test point are tested.
These are pixel positions that are right, left, above and below of current pixels as shown in figure 2.26



8-connected Method

Here 8 neighboring pixels of current test pixel are tested.
These pixel positions are left, right, above and 4 – diagonal positions of current pixel as shown in figure 2.27



Boundary Fill Algorithm

The following procedure illustrate a recursive method for boundary fill by using 4-connected method.

```
bfill (x, y, newcolor)
{
        current = getpixel (x, y);
        if (current != newcolor)) && (current != boundarycolor)

        {
                putpixel (x, y, newcolor);
                bfill (x+1, y, newcolor);
                bfill (x-1, y, newcolor);
                bfill (x, y+1, newcolor);
                bfill (x, y-1, newcolor);
        }

}
```

Flood Fill (Seed fill) Algorithm

The following procedure illustrate a recursive method for flood fill by using 4-connected method.

```
F-fill (x, y, newcolor)
{       current = getpixel (x, y);
        if (current != newcolor)
        {       putpixel (x, y, newcolor);
                f-fill (x-1, y, newcolor);
                 f-fill (x+1, y, newcolor);
                 f-fill (x, y-1, newcolor);
                 f-fill (x, y+1, newcolor);
        }
}
```

Scan line Algorithm

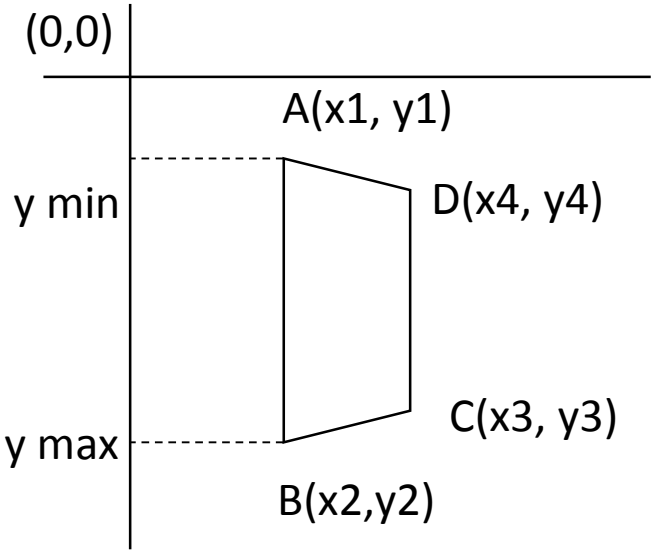


Fig. 3.4.1

Fig. 3.4.1

Edge	y max	x max	y min	x min	Slope
AB	y2	x2	y1	x1	m1
AD	y4	x4	y1	x1	m2
CD	y3	x4	Y4	x4	m3
BC	y2	x2	y3	x3	m4

Scan line Algorithm

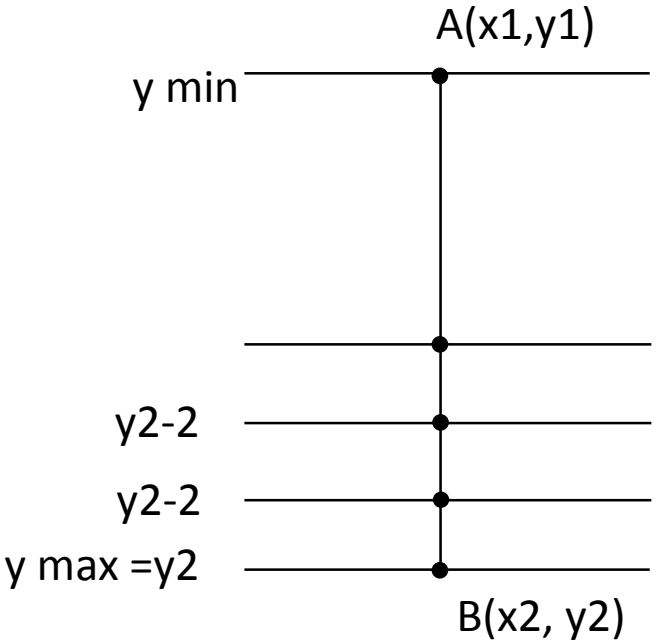


Fig. 3.4.2

Edge	y max	x max	y min	x min	Slope
AB	y2	x2	y1	x1	m1
BC	y2	x2	y3	x3	m4
CD	y3	x3	Y4	x4	m3
AD	y4	x4	y1	x1	m2

Scan line Algorithm

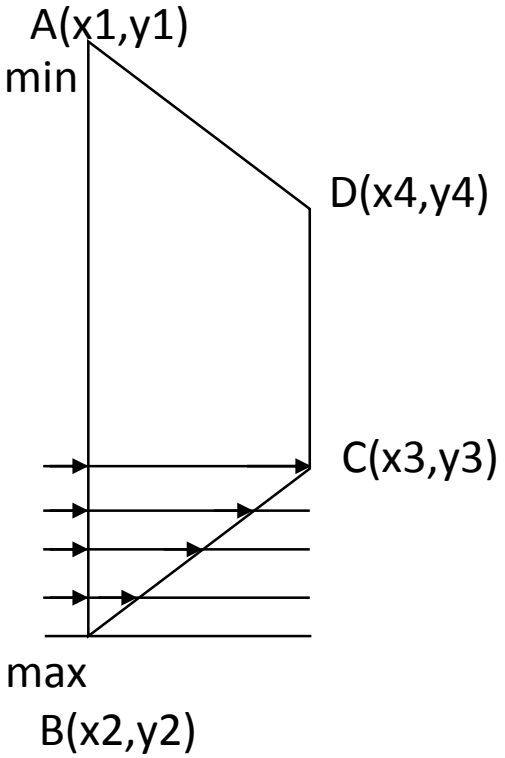


Fig. 3.4.3

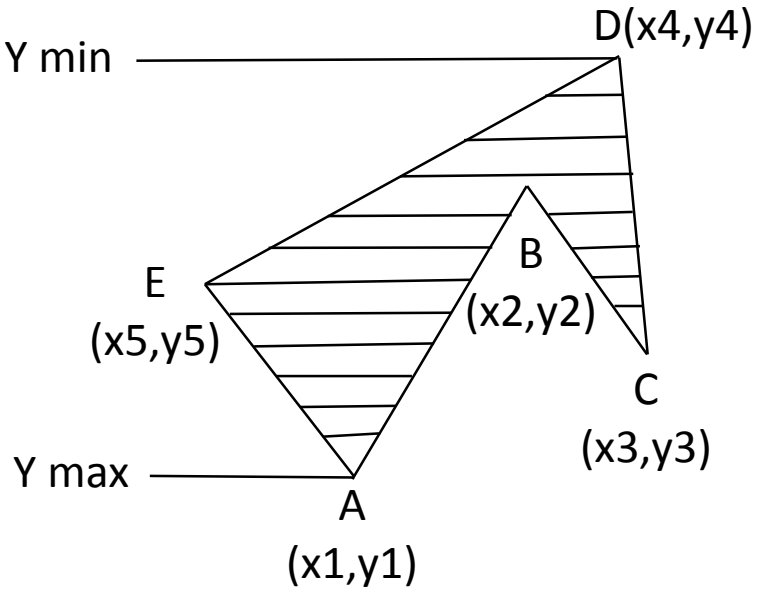


Fig. 3.4.4

Edge	y max	x max	y min	x min	Slope
AB	y1	x1	y2	x2	m1
BC	y1	x1	y5	x3	m2
CD	y3	x3	Y2	x2	m3
AD	y	x	y	x	m4

Scan line Algorithm

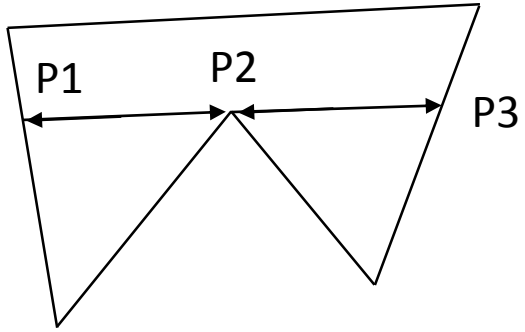


Fig. 3.4.5

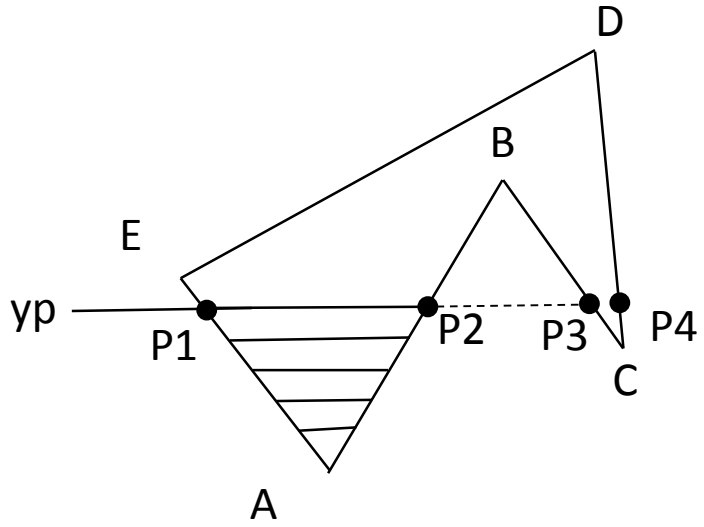


Fig. 3.4.6

Flood fill Vs Boundary fill Vs Edge fill Vs Fence fill Vs Scan line fill

Flood fill	Boundary Fill	Edge fill	Fence Fill	Scan line fill
Need Seed point to start.	Need Seed point to start.	Based on complimenting the pixels.	Based on complimenting the pixels.	Based on line drawing, polygon is filled.
Current pixels color is compared with new pixel color.	Current pixels color is compared with new pixel color and boundary color.	Pixels which are on right side of an edge are getting complemented.	Pixels which are on right side of an edge and to the left of fence as well as left side of edge and right side of fence are getting complemented.	Intersection of polygon edges are found with the scan line and then the solid lines are drawn between two such intersection points.
Useful for polygons having single color boundary.	Useful for polygons having multi color boundary.	More number of pixels are unnecessarily accessed.	Less number of pixels are accessed as compared to edge fill.	Need separate attention to handle concave polygons.

Connected boundary fill Algorithm

To enhance speed of filling colour one may look for alternative of 4-connected.
In this algorithm all adjacent pixels will be consider for filling till the match is true. Algorithm is as follows :

```
boundary_fill (x, y, f_colour, b_colour)
{
        if (getpixel (x, y) != b_colour && getpixel (x, y) != f_colour)
        {
                putpixel (x, y, f_colour);
                boundary_fill (x+1, y, f_colour, b_colour);
                boundary_fill (x-1, y, f_colour, b_colour);
                boundary_fill (x, y+1, f_colour, b_colour);
                boundary_fill (x, y-1, f_colour, b_colour);
                boundary_fill (x+1, y+1, f_colour, b_colour);
                boundary_fill (x-1, y-1, f_colour, b_colour);
                boundary_fill (x+1, y-1, f_colour, b_colour);
                boundary_fill (x-1, y+1, f_colour, b_colour);
        }
}
```