1. *What is point clipping and line clipping ?*
2. *How will you clip a point ?*

AU : May-13, Marks 2

5.8 Cohen-Sutherland Line Subdivision Clipping Algorithm

AU : May-12, Dec.-12

- This is one of the oldest and most popular line clipping algorithm developed by Dan Cohen and Ivan Sutherland.
- To speed up the processing this algorithm performs initial tests that reduce the number of intersections that must be calculated.
- This algorithm uses a four digit (bit) code to indicate which of nine regions contain the end point of line.
- The four bit codes are called **region codes** or **outcodes.** These codes identify the location of the point relative to the boundaries of the clipping rectangle as shown in the Fig. 5.8.1.

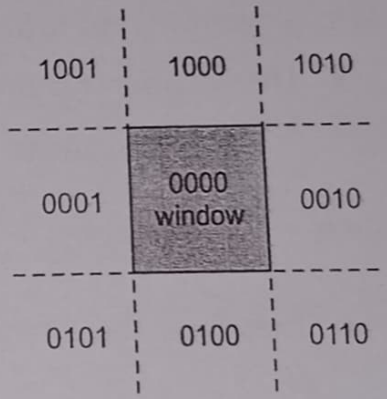


Fig. 5.8.1 Four-bit codes for nine regions

- Each bit position in the region code is used to indicate one of the four relative co-ordinate positions of the point with respect to the clipping window : To the

left, right, top or bottom. The rightmost bit is the first bit and the bits are set to 1 based on the following scheme :

- Set Bit 1 - If the end point is to the **left** of the window
- Set Bit 2 - If the end point is to the **right** of the window
- Set Bit 3 - If the end point is **below** the window
- Set Bit 4 - If the end point is **above** the window

Otherwise, the bit is set to zero.

- Once we have established region codes for all the line endpoints, we can determine which lines are completely inside the clipping window and which are clearly outside.
- Any lines that are completely inside the window boundaries have a region code of 0000 for both endpoints and we trivially accept these lines.
- Any lines that have a 1 in the same bit position in the region codes for each endpoint are completely outside the clipping rectangle and we trivially reject these lines.
- A method used to test lines for total clipping is equivalent to the logical **AND** operator.
- If the result of the logical AND operation with two end point codes is not 0000, the line is completely outside the clipping region.
- The lines that cannot be identified as completely inside or completely outside a clipping window by these tests are checked for intersection with the window boundaries.

Sutherland and Cohen subdivision line clipping algorithm :

1. Read two end points of the line say $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$.
2. Read two corners (left-top and right-bottom) of the window, say (W_{x1}, W_{y1}) and (W_{x2}, W_{y2}).
3. Assign the region codes for two endpoints P_1 and P_2 using following steps :

   Initialize code with bits 0000

   Set Bit 1 - if($x < W_{x1}$)

   Set Bit 2 - if ($x > W_{x2}$)

   Set Bit 3 - if ($y < W_{y2}$)

   Set Bit 4 - if ($y > W_{y1}$)

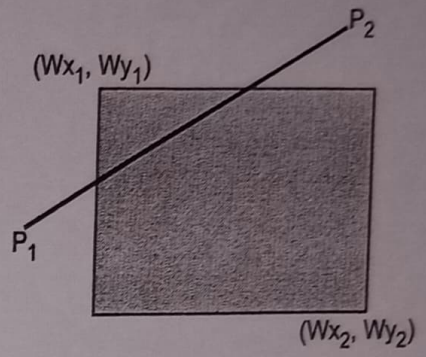


Fig. 5.8.5

4. Check for visibility of line P_1P_2
   a) If region codes for both endpoints P_1 and P_2 are zero then the line is completely visible. Hence draw the line and go to step 9.
   b) If region codes for endpoints are not zero and the logical ANDing of them is also non-zero then the line is completely invisible, so reject the line and go to step 9.
   c) If region codes for two endpoints do not satisfy the conditions in (4a) and (4b) the line is partially visible.
5. Determine the intersecting edge of the clipping window by inspecting the region codes of two endpoints.
   a) If region codes for both the end points are non-zero, find intersection points P_1' and P_2' with boundary edges of clipping window with respect to point P_1 and point P_2, respectively
   b) If region code for any one end point is non-zero then find intersection point P_1' or P_2' with the boundary edge of the clipping window with respect to it.
6. Divide the line segments considering intersection points.
7. Reject the line segment if any one end point of it appears outsides the clipping window.
8. Draw the remaining line segments.
9. Stop.

Program 5.8.1 *C' code for sutherland and Cohen subdivision line clipping algorithm.*

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<dos.h>
#include<math.h>
#include<graphics.h>
/* Defining structure for end point of line */
typedef struct co-ordinate
{
int x,y;
char code[4];
}PT;
void drawwindow();
void drawline (PT p1,PT p2,int cl);
PT setcode(PT p);
int visibility (PT p1,PT p2);
PT resetendpt (PT p1,PT p2);
main()
{
int gd=DETECT, gm,v;
PT p1,p2,ptemp;
initgraph(&gd,&gm," ");
cleardevice();
printf("\n\n\t\tENTER END-POINT 1 (x,y): ");
scanf("%d,%d",&p1.x,&p1.y);
printf("\n\n\t\tENTER END-POINT 2 (x,y): ");
scanf("%d,%d",&p2.x,&p2.y);
cleardevice();
drawwindow();
getch();
drawline(p1,p2,15);
getch();
p1=setcode(p1);
p2=setcode(p2);
v=visibility(p1,p2);
switch(v)
        {
        case 0: cleardevice(); /* Line completely visible*/
                drawwindow();
                drawline(p1,p2,15);
                break;
        case 1: cleardevice(); /* Line completely invisible */
                drawwindow();
                break;
        case 2: cleardevice();  /* line partly visible */
```

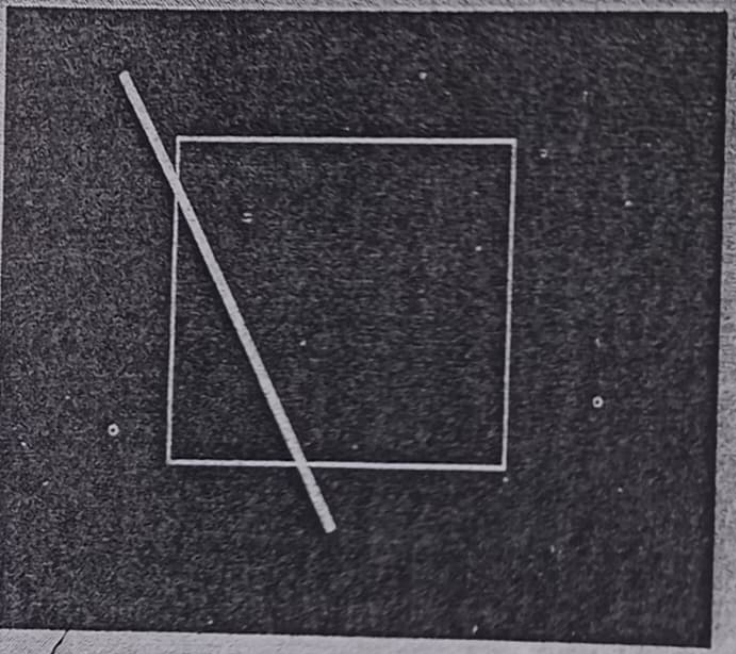
```
                p1=resetendpt (p1,p2);
                p2=resetendpt(p2,p1);
                drawwindow();
                drawline(p1,p2,15);
                break;
        }
        getch();
        closegraph();
        return(0);
        }
        /* Function to draw window */
        void drawwindow()
        {
        setcolor(RED);
        line(150,100,450,100);
        line(450,100,450,350);
        line(450,350,150,350);
        line(150,350,150,100);
        }
        /* Function to draw line between two points
        ----------------------------------------*/
        void drawline (PT p1,PT p2,int cl)
        {
        setcolor(cl);
        line(p1.x,p1.y,p2.x,p2.y);
        }
        /* Function to set code of the co-ordinates
        ----------------------------------------*/
        PT setcode(PT p)
        {
        PT ptemp;
        if(p.y<100)
        ptemp.code[0]='1'; /* TOP */
        else
        ptemp.code[0]='0';
        if(p.y>350)
        ptemp.code[1]='1'; /* BOTTOM */
        else
        ptemp.code[1]='0';
        if (p.x>450)
        ptemp.code[2]='1'; /* RIGHT */
        else
        ptemp.code[2]='0';
        if (p.x<150) /* LEFT */
        ptemp.code[3]='1';
        else
        ptemp.code[3]='0';
```

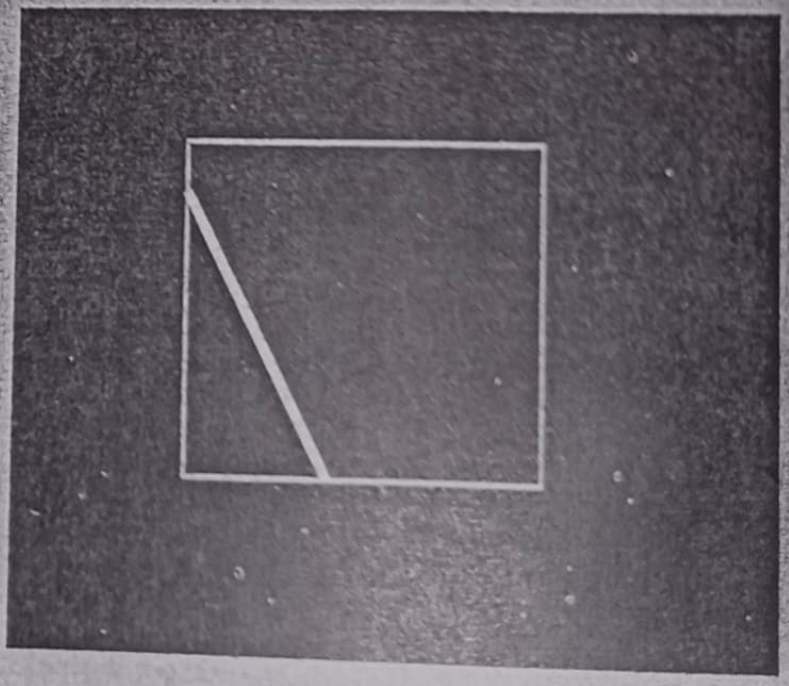
```
ptemp.x=p.x;
ptemp.y=p.y;
return(ptemp);
}
/* Function to determine visibility of line
-----------------------------------------*/
int visibility (PT p1,PT p2)
{
int i,flag=0;
for(i=0;i<4;i++)
{
if((p1.code[i]!='0')||(p2.code[i]!='0'))
flag=1;
}
if(flag==0)
return(0);
for(i=0;i<4;i++)
{
if((p1.code[i]==p2.code[i]) &&(p1.code[i]=='1'))
flag=0;
}
if(flag==0)
return(1);
return(2);
}
/* Function to find new end points
-----------------------------------*/
PT resetendpt (PT p1,PT p2)
{
PT temp;
int x,y,i;
float m,k;
if( p1.code[3]=='1') /* Cutting LEFT Edge */
x=150;
if(p1.code[2]=='1') /* Cutting RIGHT Edge */
x=450;
if((p1.code[3]=='1')||(p1.code[2]=='1'))
{
m=(float) (p2.y-p1.y)/(p2.x-p1.x);
k=(p1.y+(m*(x-p1.x)));
temp.y=k;
temp.x=x;
for(i=0;i<4;i++)
temp.code[i]=p1.code[i];
if(temp.y<=350&&temp.y>=100)
```

```
return(temp);
}
if(p1.code[0]=='1') /* Cutting TOP Edge */
y=100;
if(p1.code[1]=='1') /* Cutting BOTTOM Edge */
y=350;
if((p1.code[0]=='1')||(p1.code[1]=='1'))
{
m=(float)(p2.y-p1.y)/(p2.x-p1.x);
k=(float)p1.x+(float)(y-p1.y)/m;
temp.x=k;
temp.y=y;
for(i=0;i<4;i++)
temp.code[i]=p1.code[i];
return(temp);
}
else
return(p1);
}
```

Output

```
Enter END-POINT 1(x, y) : 50, 100
Enter END-POINT 2(x, y) : 300, 400
```





Example 5.8.2 Use the Cohen-Sutherland outcode

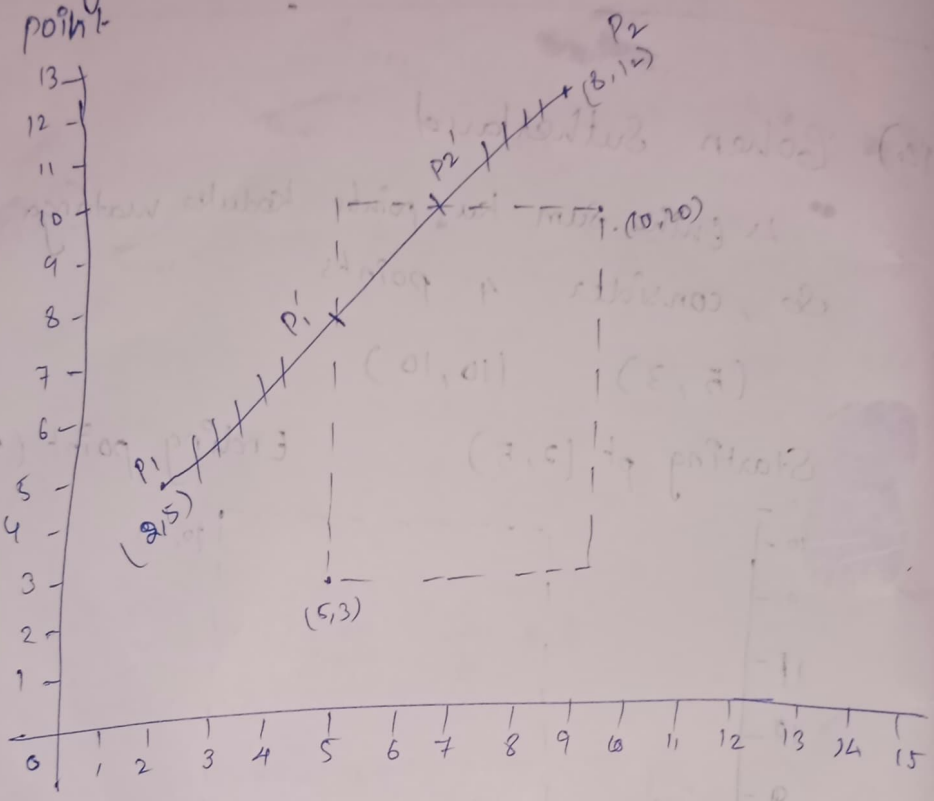
14) Cohen Sutherland

Sum la points kodukka matang

So consider 4 points

(5,3) (10,10)

Starting point (2,5) Ending point (8,12)



y Intersection (x constant)

formula $y = y_1 + m(x - x_1)$

x_1, y_1 x_2, y_2

(2,5) (8,12)

$m = \frac{y_2 - y_1}{x_2 - x_1}$

$= \frac{12 - 5}{8 - 2}$

$m = 7/6$

(5,3) (2,5)

x y x_1, y_1

(constant)

$y = 5 + \frac{7}{6}(5 - 2)$

$= 5 + \frac{7}{6_2} \times 3$

$y = 17/2 = 8.5$

$P_1' = (5, 8.5)$

X intersection : (y constant)

$$x = x_1 + \frac{(y - y_1)}{m}$$

(10, 10) (8, 12)
x, y x_1 y_1

$$= 8 + \frac{10 - 12}{7/6}$$

$$= 8 + \frac{-2 \times 6}{7}$$

$$= \frac{56 - 12}{7}$$

$$x = \frac{44}{7} = 6.28$$

$P_2' = (6.28, 10)$