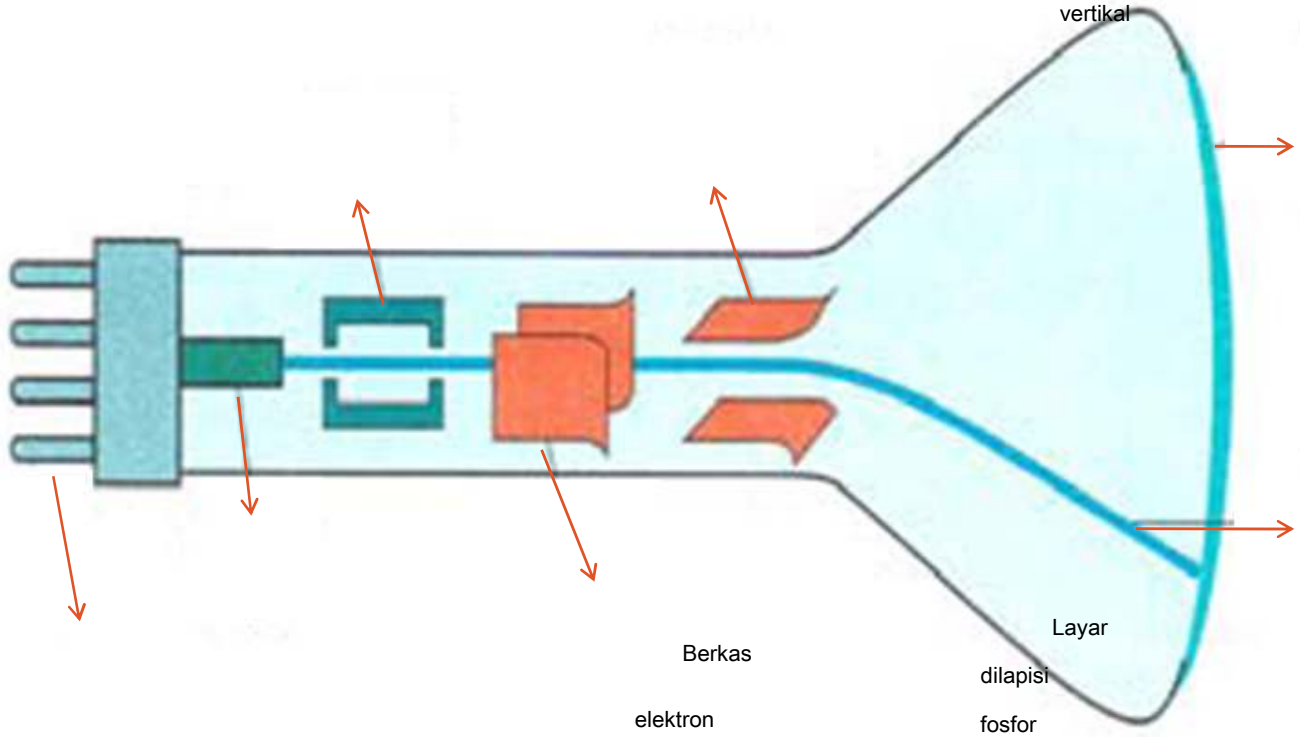
Perangkat Layar

Senjata elektron

Sistem fokus

Pelat membelok horisontal

Pelat membelok vertikal



Segarkan CR

Berkas elektron

Layar dilapisi fosfor

monitor yang paling umum menggunakan C

garis bergerigi yang diplot sebagai titik diskrit

Sistem raster menghasilkan

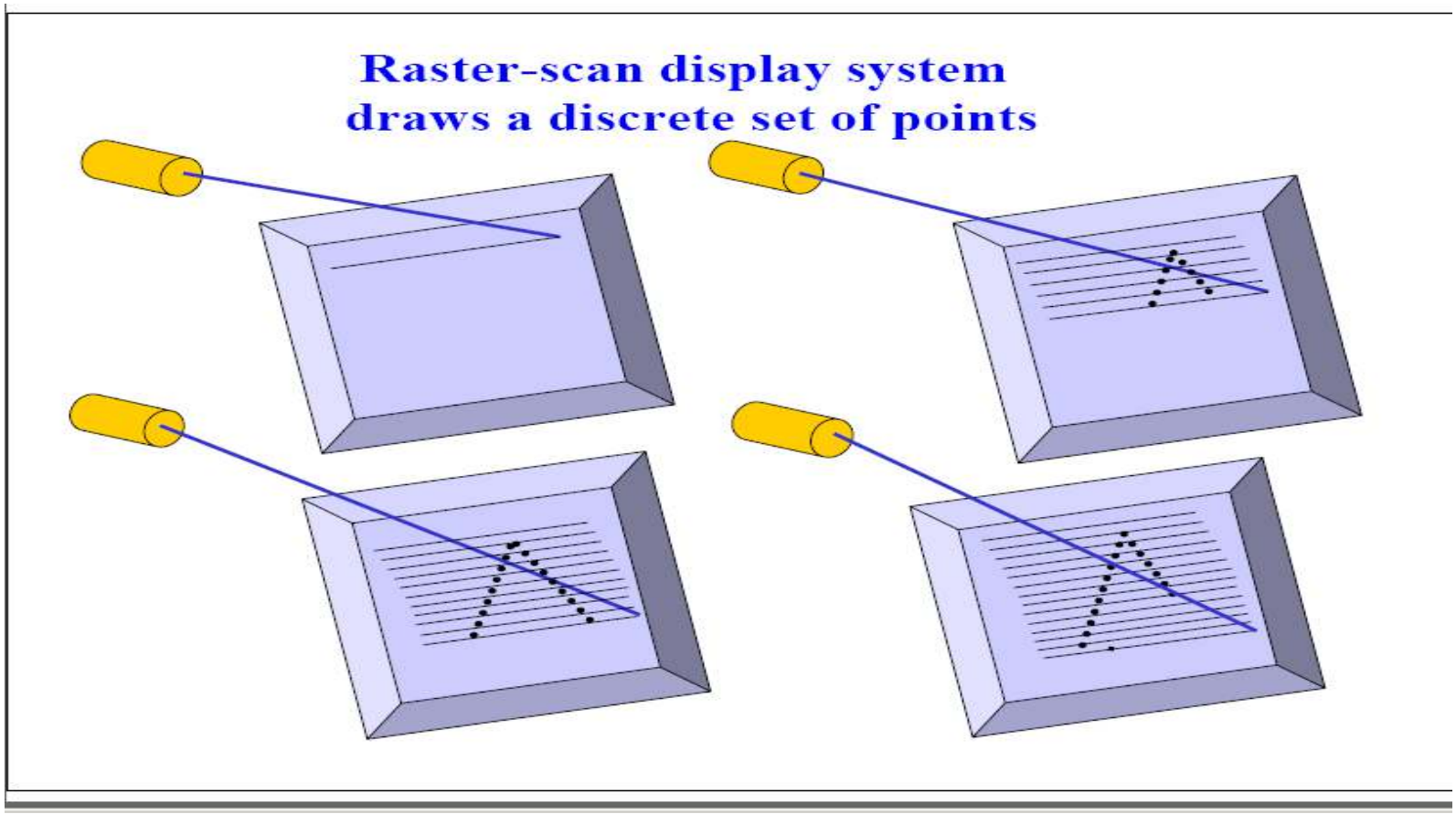
Tampilan Pi

di layar satu baris sekaligus dari atas ke bawah.

disapu

Satu titik atau elemen gambar raster. Rentang intensitasnya u

khusus digunakan untuk menyimpan gambar dengan scan-out yang sinkron ke


Raster-scan display system
draws a discrete set of points

Tampilan Pin

kembali

kembali

ke sudut kiri atas layar untuk memulai bingkai berikutnya.

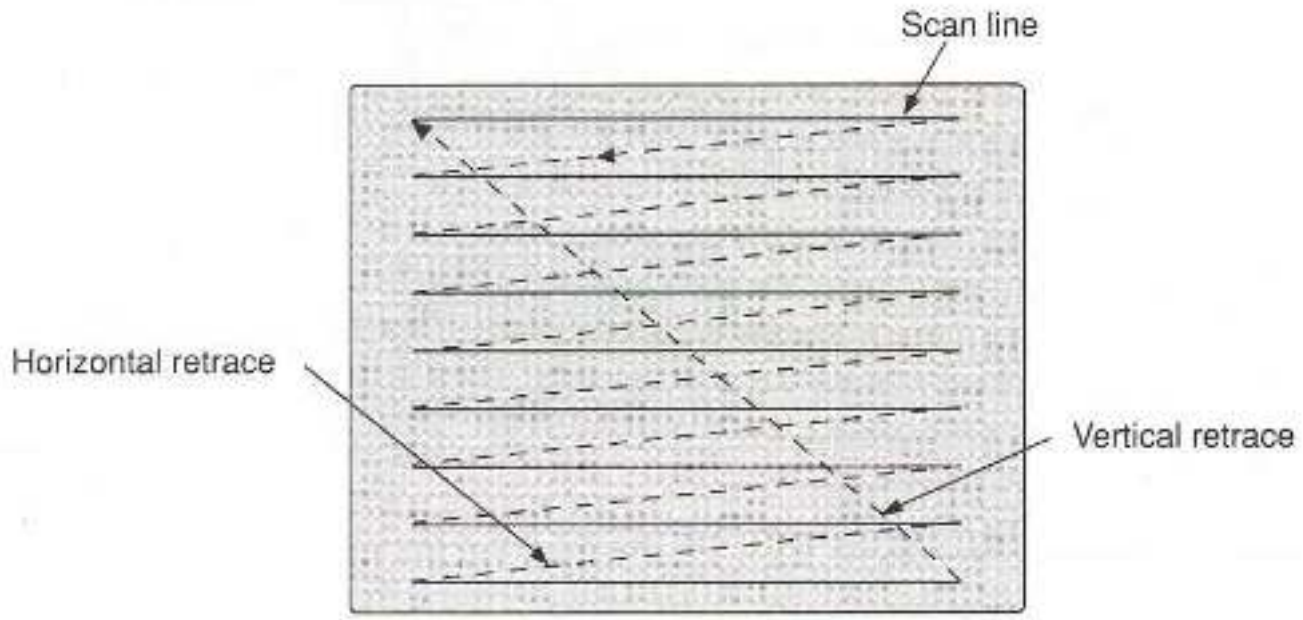
ke sisi kiri

layar

untuk mulai menampilkan garis pemindaian berikutnya. Per

Biasanya 30 ~ 75 Hz

Tampilan Pin



diarahkan

resolusi tinggi karena definisi gambar disimpan sebagai perintah menggambar garis

hanya ke bagian layar di mana gambar akan

kembali ke perintah baris pertama dalam daftar.

Acak

siklus

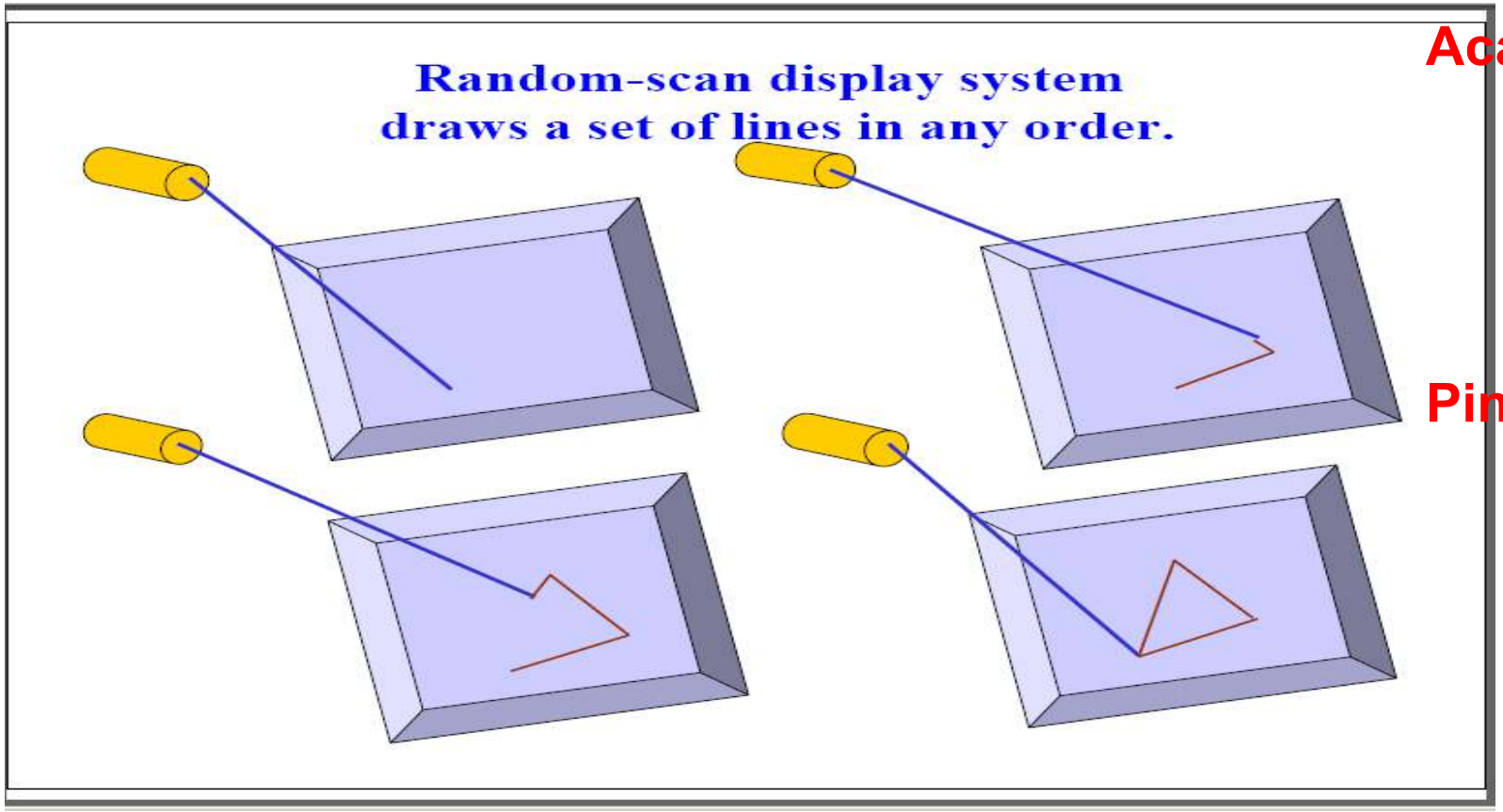
melalui

serangkaian perintah dalam file tampilan, menggambar setiap baris komp

siklus

gambar disimpan sebagai satu set perintah menggambar

Pindai Tampil



Acak

Pindai Tamp

cont ...

penentuan piksel yang sesuai untuk mewakili gambar atau grafik obyek

Primitif Kelua

Titik, segmen garis lurus, lingkaran, dan bagian kerucut lainnya,

adalah langkah terakhir rasterisasi. Ini mengubah definisi gambar menjadi satu set nilai intensitas pik

Garis

Lingkaran

Poligon

ALGORITMA

Grafis komputer

Dapatkaninput dari2 titikyang ingindihubungkan (titikawalP0

berdasarperbedaanyang dihitungpadalangkah2, ditentukanjumlah langkahuntukpixe

menentukanpixel denganmennambahnilaix dany sesuairumusyang telah

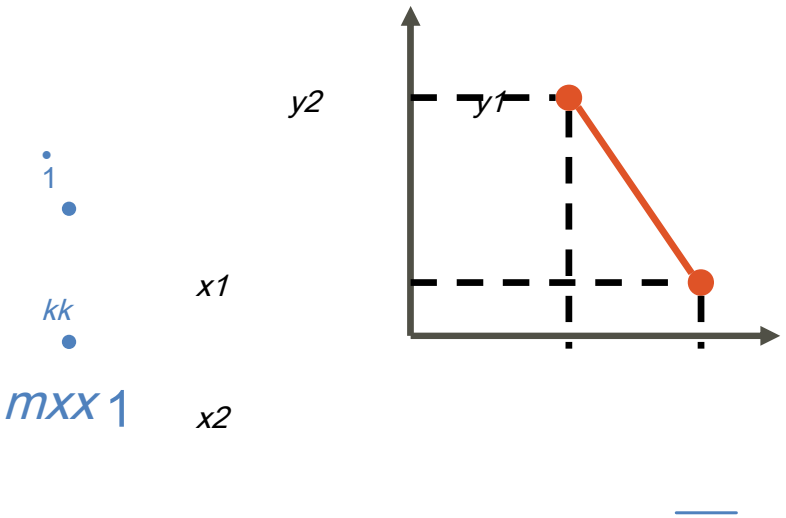
```
Xincrement = dx / (float) steps;
Yincrement = dy / (float) steps;
```

```
for(int v=0; v < Steps; v++)
    {
        x = x + Xincrement;
        y = y + Yincrement;
        putpixel(x,y);
    }
```

- Kemiringan <-1
  - Interval satuan = 1
- -1 <= Kemiringan <0
  - Satuan x interval = 1
- 0 <Kemiringan <= 1
  - Interval satuan = 1
- Kemiringan> 1
  - Satuan x interval = 1

DDA

Algoritma



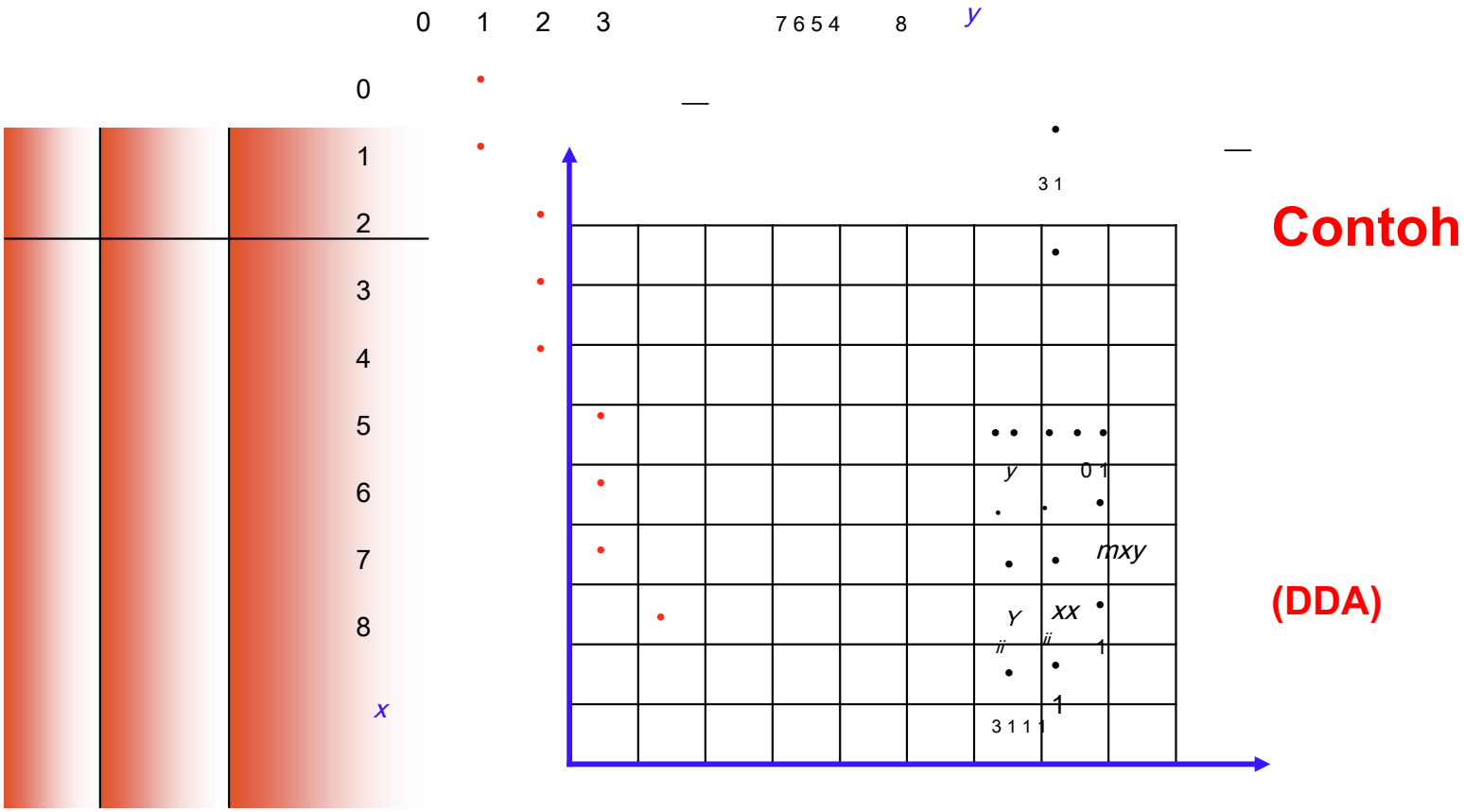
lanjut ..

11/3 10/3 8/3 7/3 5/3 4/3 y

3 2 1

bulat (y

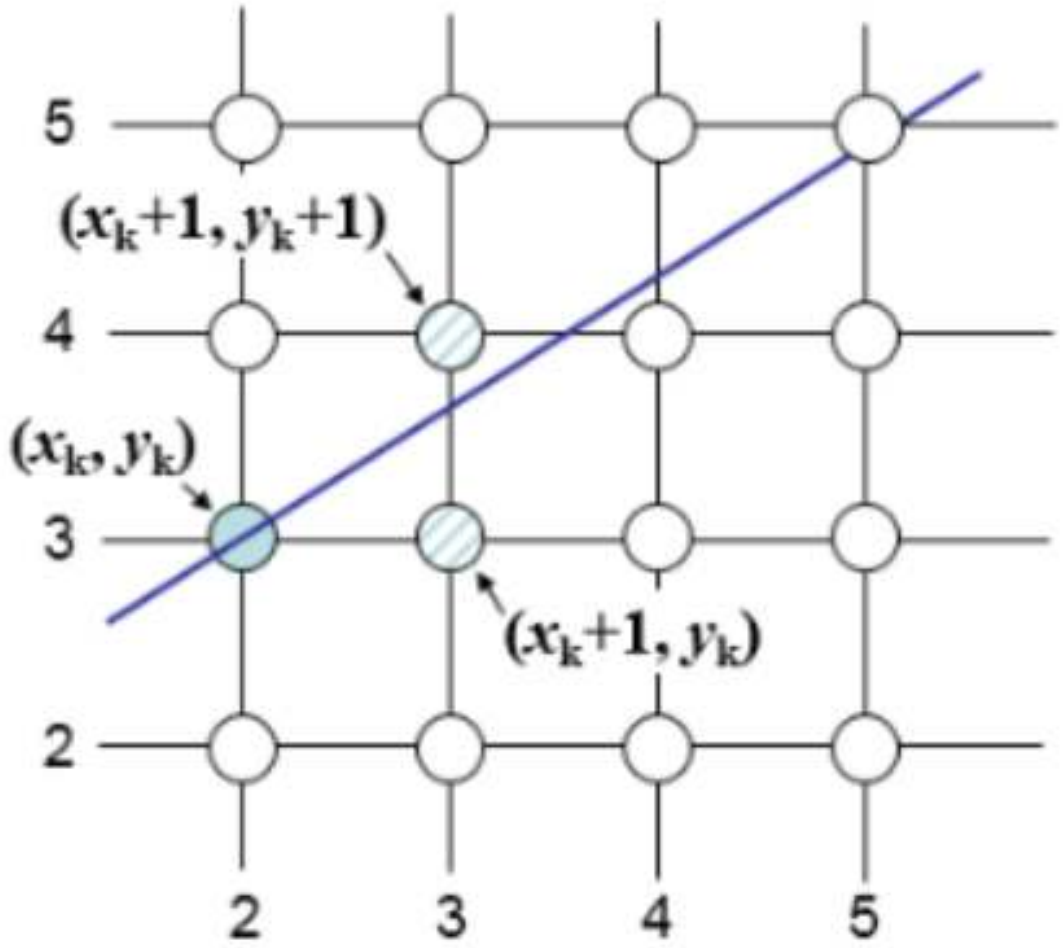
4 3 3 3 2 2 2 1 1)

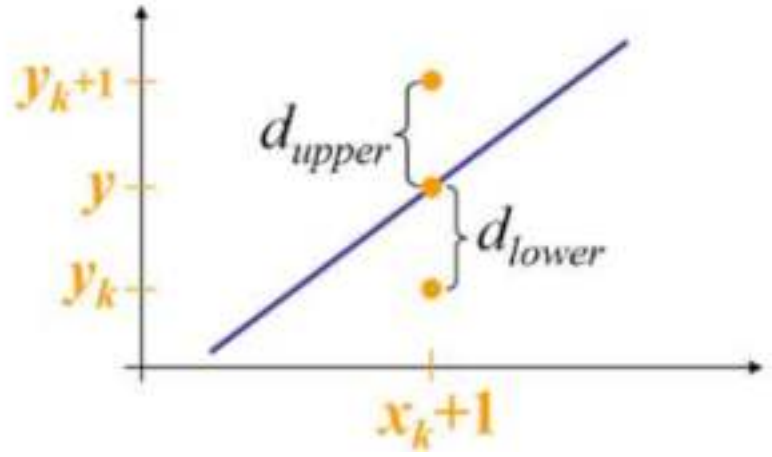


Contoh

(DDA)

dengansuatuinterval tertentu, danpadamasing-masing langkahmemilih


5
4
3
2
2
3
4
5
(x_k+1, y_k+1)
(x_k, y_k)
(x_k+1, y_k)


y_k+1
y
y_k
d_{upper}
d_{lower}
x_k+1

$$Y = m(X_k + 1) + b$$

$$\begin{aligned} d_{lower} &= y - y_k \\ &= m(X_k + 1) + b - Y_k \end{aligned}$$

$$\begin{aligned} d_{upper} &= (y_k + 1) - y \\ &= Y_k + 1 - m(X_k + 1) - b \end{aligned}$$

$$d_{lower} - d_{upper} = 2m(x_k + 1) - 2y_k + 2b - 1$$

Parameter keputusan

$$dx(d_{lower} - d_{upper}) = dx(2\frac{dy}{dx}(x_k + 1) - 2y_k + 2b - 1)$$

$$= 2dy \cdot x_k - 2dx \cdot y_k + 2dy + dx(2b - 1)$$

$$= 2dy \cdot x_k - 2dx \cdot y_k + C$$

$$p_k = dx(d_{lower} - d_{upper})$$

$$= 2dy \cdot x_k - 2dx \cdot y_k + C$$

Step 1: Input the two end-points of line, storing the left end-point in (x_0, y_0).

Step 2: Plot the point (x_0, y_0).

Step 3: Calculate the constants dx, dy, 2dy, and (2dy – 2dx) and get the first value for the decision parameter as:

$$p_0 = 2dy - dx$$

Step 4: At each X_k along the line, starting at $k = 0$, perform the following test:

If $p_k < 0$, the next point to plot is (x_k+1, y_k) and

$$p_{k+1} = p_k + 2dy\text{Otherwise,}$$
$$p_{k+1} = p_k + 2dy - 2dx$$

Step 5: Repeat step 4 (dx – 1) times.

For m > 1, find out whether you need to increment x while incrementing y each time.

After solving, the equation for decision parameter p_k will be very similar, just the x and y in the equation gets interchanged.

menggambarkan algoritma, kami mendigitalkan ga

10 14 - 2 2 6 *hal*
k

(25,14)(24,13)(23,12)(22,12)(21,11)(*x*
k +1, *y*

k +1)

0, *y*

0)

= (-20,10) dan tentukan posisi piksel be
parameter keputusan sebagai

9 8 7 6 5 **K**

10 14 - 2 2 6 *hal*
k

		(30,18)(29,17)(28,16)(27,16)(26,15)(*x*			
			k +1, *y*		
			k +1)		

A plot of the pixels generated along this line path is shown in Fig.

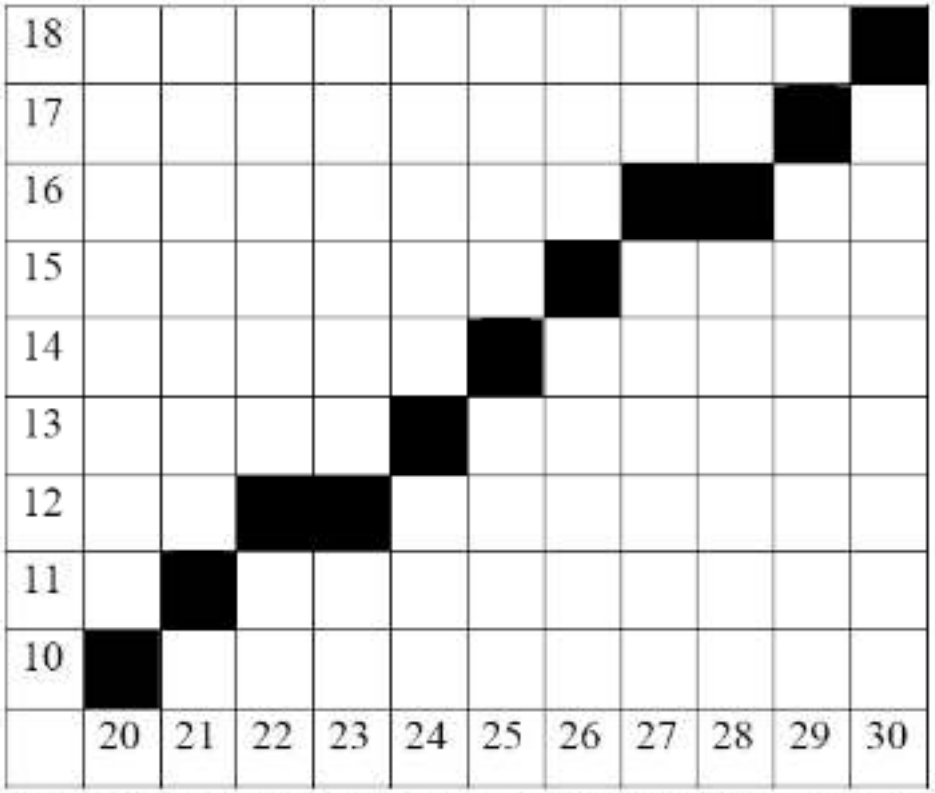


Figure: The Bresenham line from point (20,10) to point (30,18)

CIRCLE

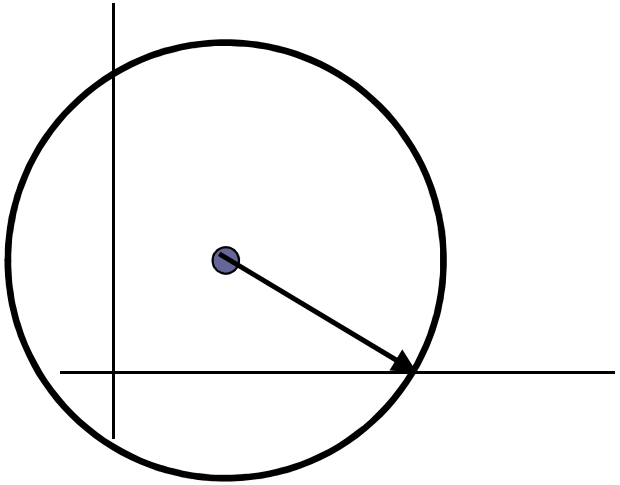
(x

c, y

r

c)

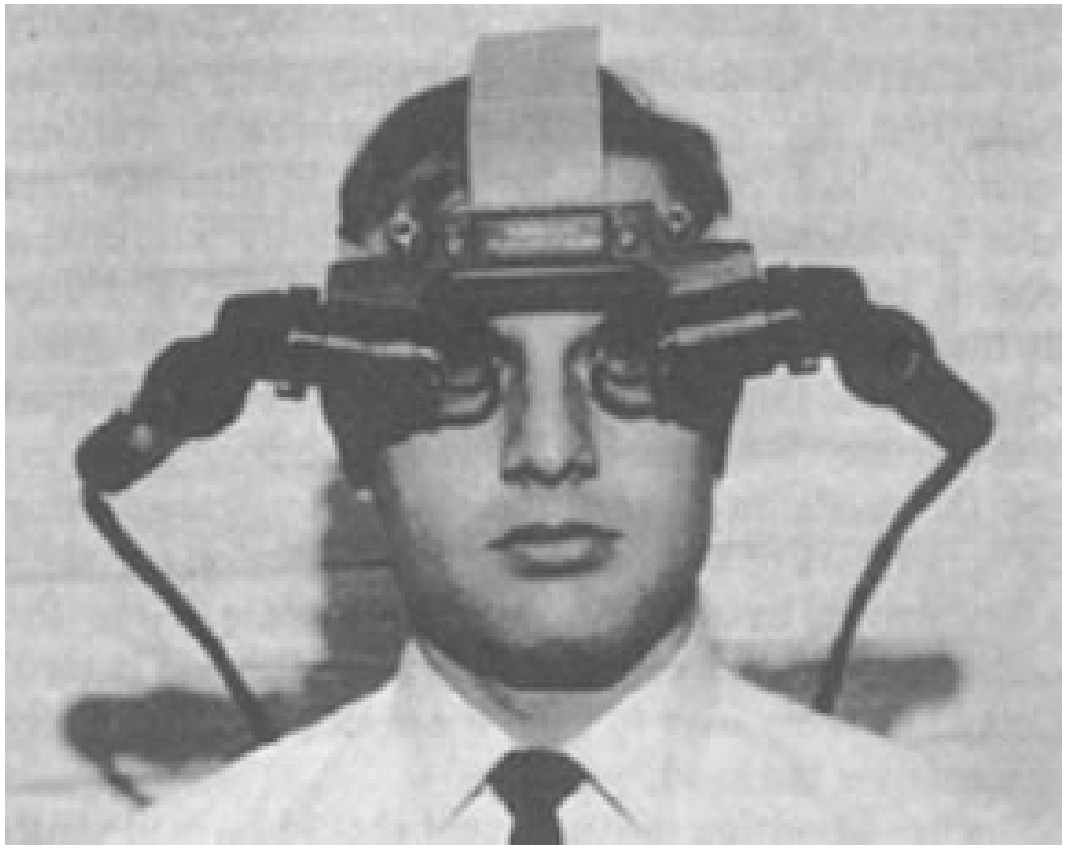
Bukan metode yang sangat bagus. Mengapa?

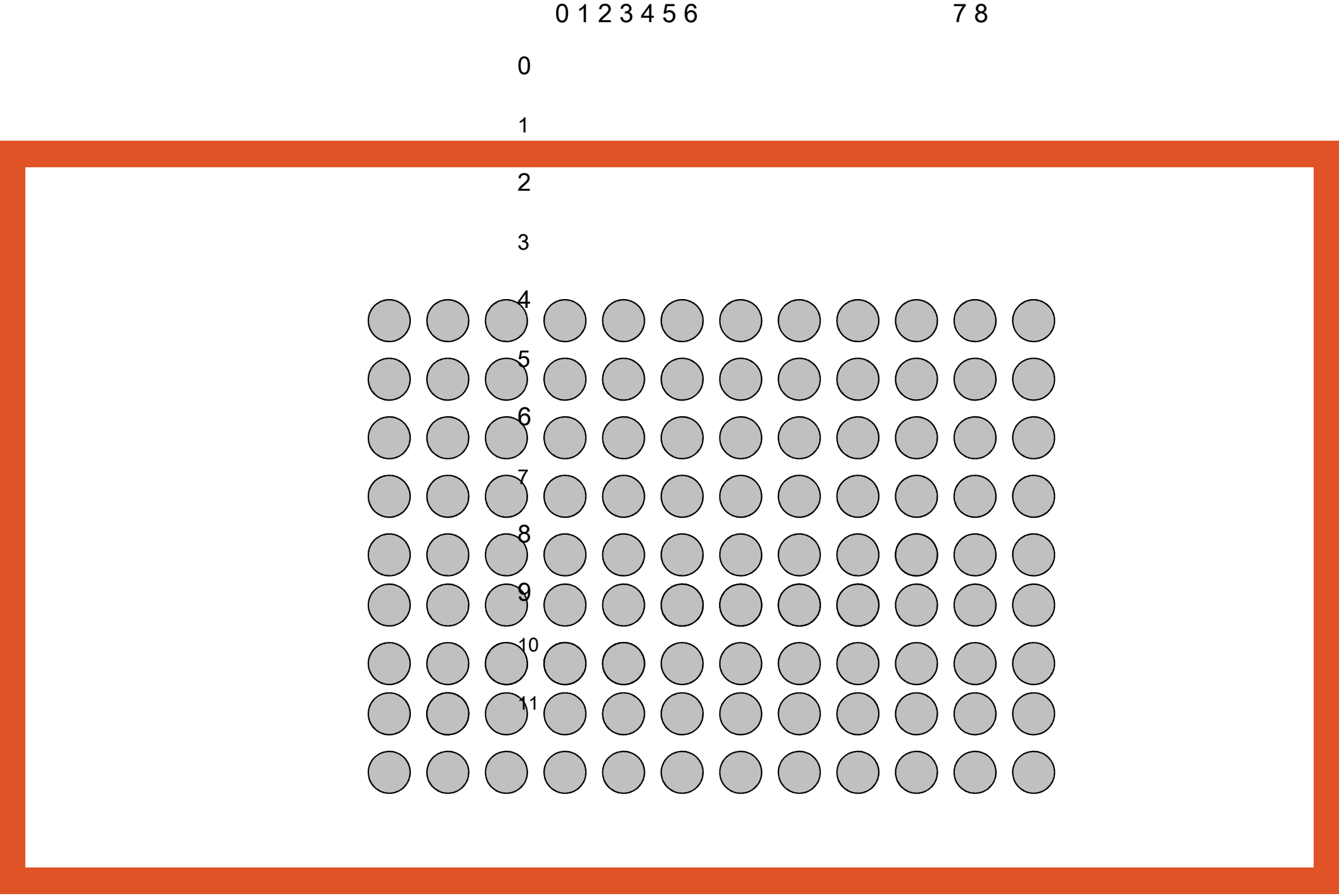


$$y = \pm\sqrt{r^2 - (x - c_x)^2} + c_y$$

$$(y - c_y)^2 = r^2 - (x - c_x)^2$$

$$(y - c_y)^2 + (x - c_x)^2 = r^2$$



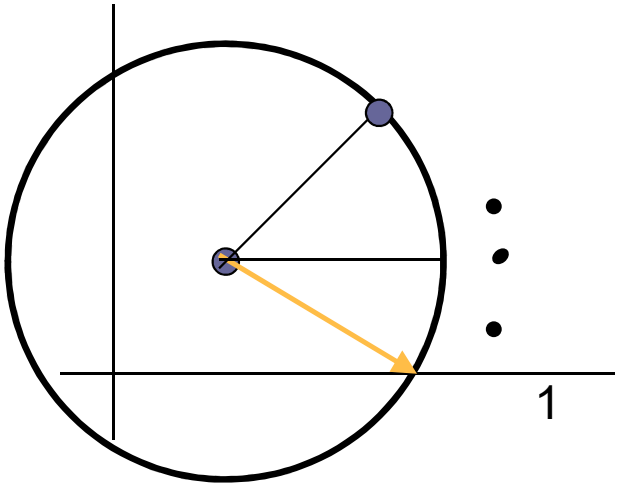

0 1 2 3 4 5 6
7 8
0
1
2
3
4
5
6
7
8
9
10
11

sederhana: plot langsung dari persamaan parametrik

r

θ

(x, y)



xx

cc

r *ryy*

karena

srsr

dosa

1

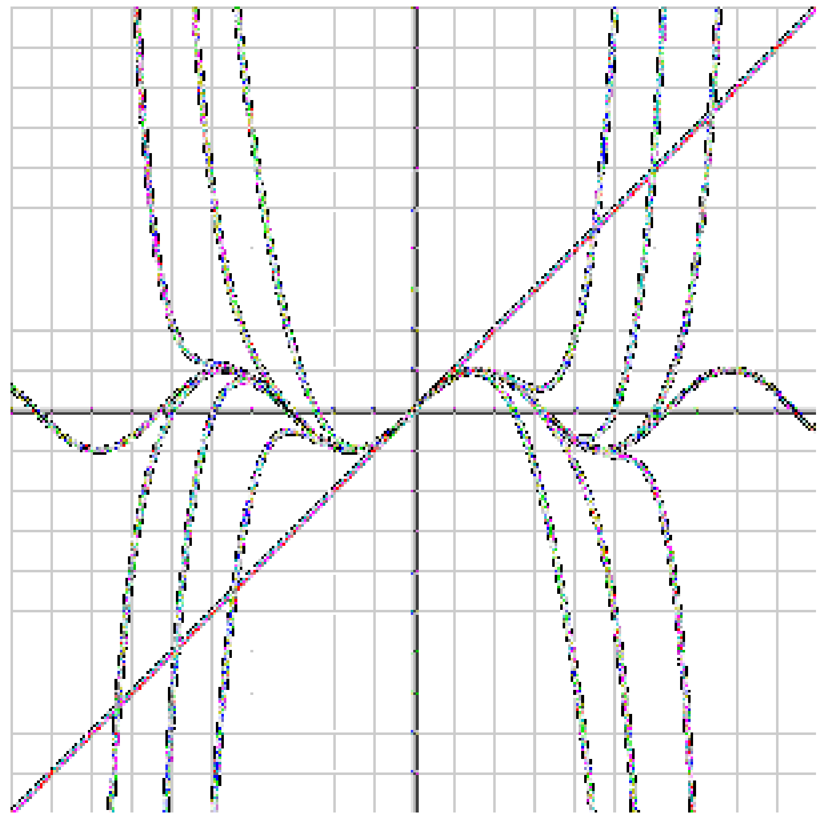
0

0 (2 (1))! n

(2 (1) 1)! n

2 nnn

2 nnn 1

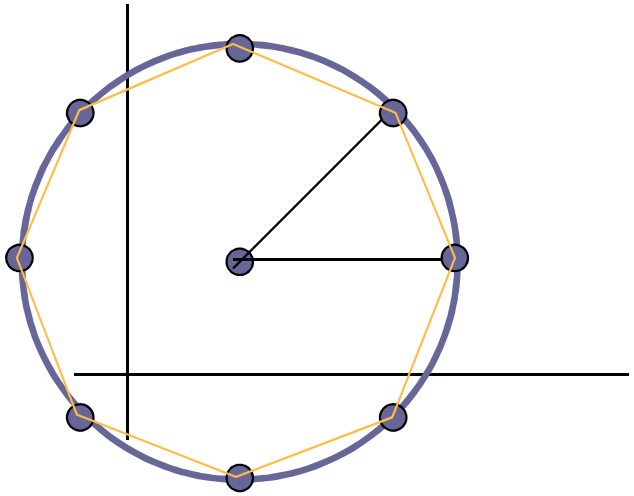


(x

saya, y

i) Hitung

simpul poligon dari persamaan



kutub; terhubung dengan algoritma garis

Algoritma

untuk

kontrol

untuk komputer dari plotter digital, *Jurnal Sistem IBM,* 4 (1), 1965

tampilan digital tambahan dari busur lingkaran. *Komunikasi ACM,* 20

xy

x^2

y^2

r^2

$f_{lingkaran}$

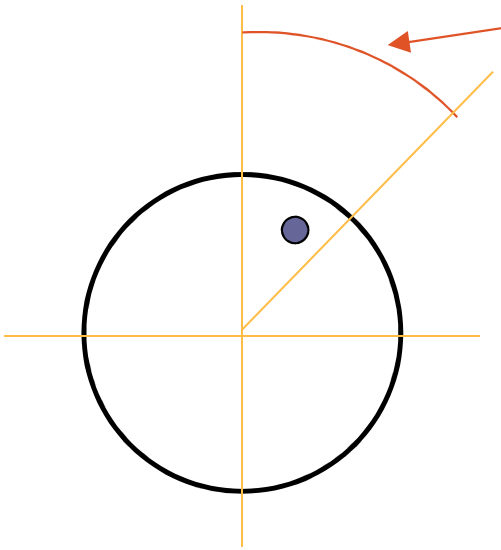
xy

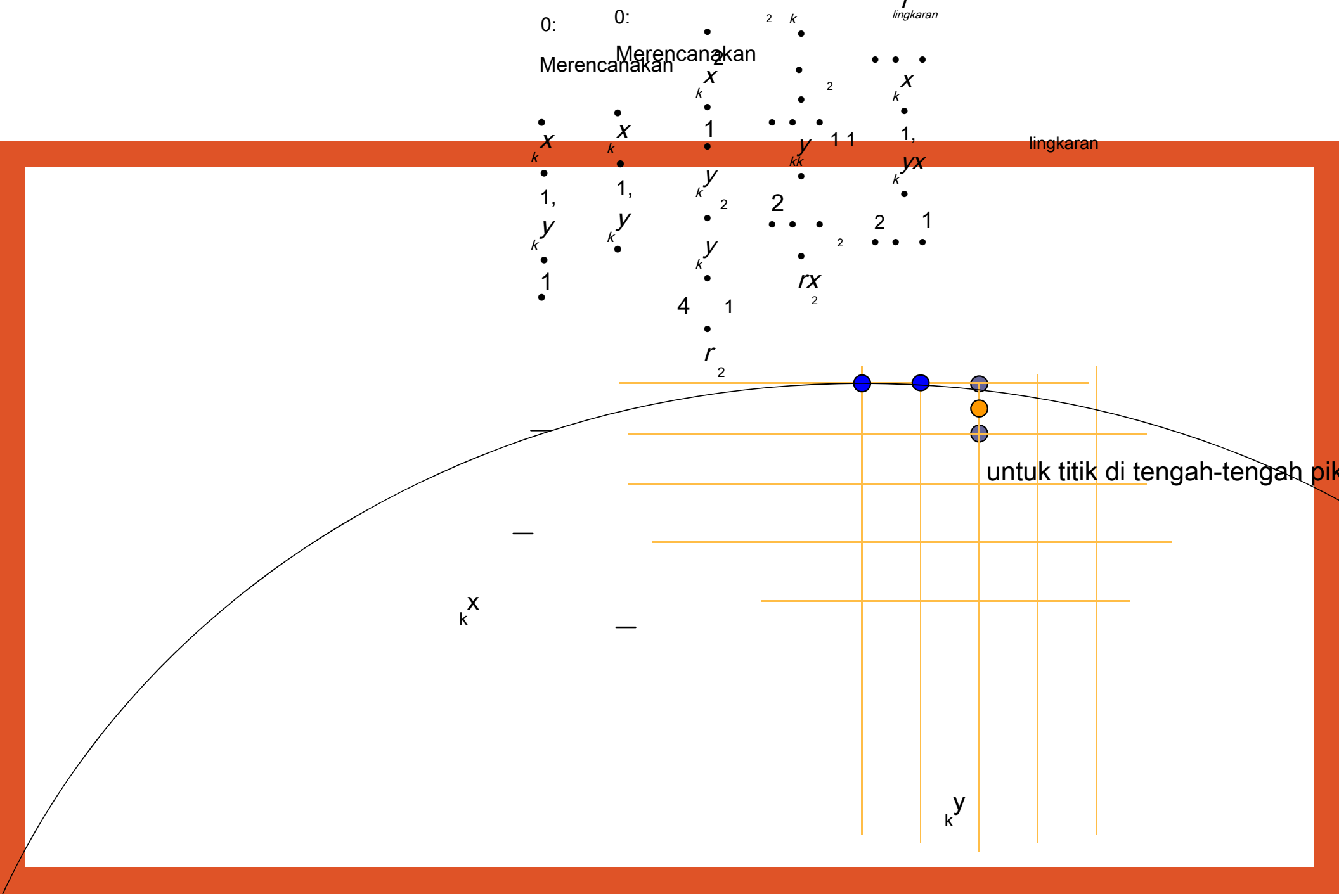
0, 0, 0,

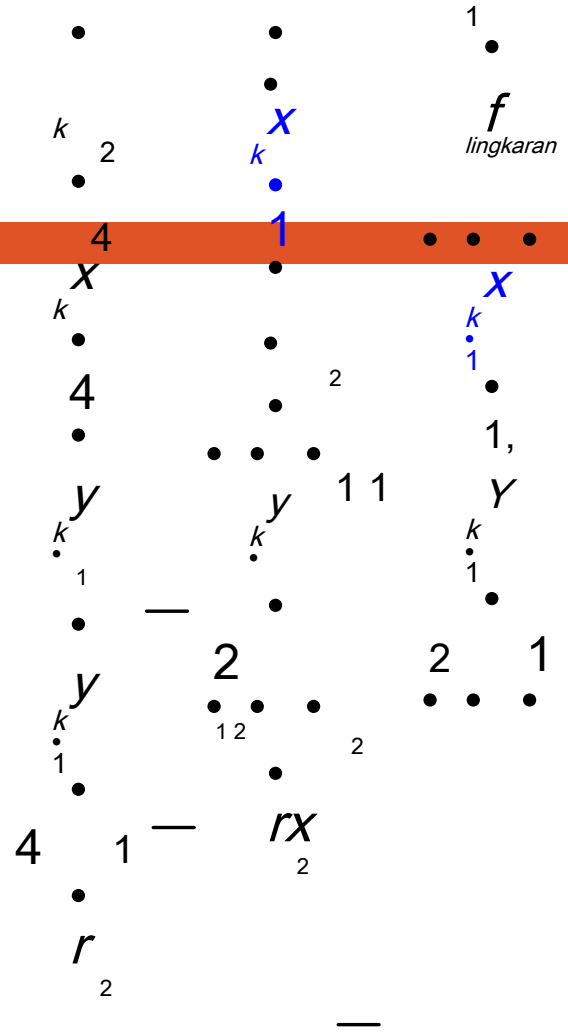
jika (xi, yi) di dalam lingkaran

0

<| m | <1 di wilayah ini




Merencanakan
lingkaran
untuk titik di tengah-tengah pik


k
2
x
k
1
f
lingkaran
4
x
k
4
y
k
1
y
k
1
r
2
2
1 1
2
1 2
rx
2
1
x
k
1
1,
y
k
1
2
1

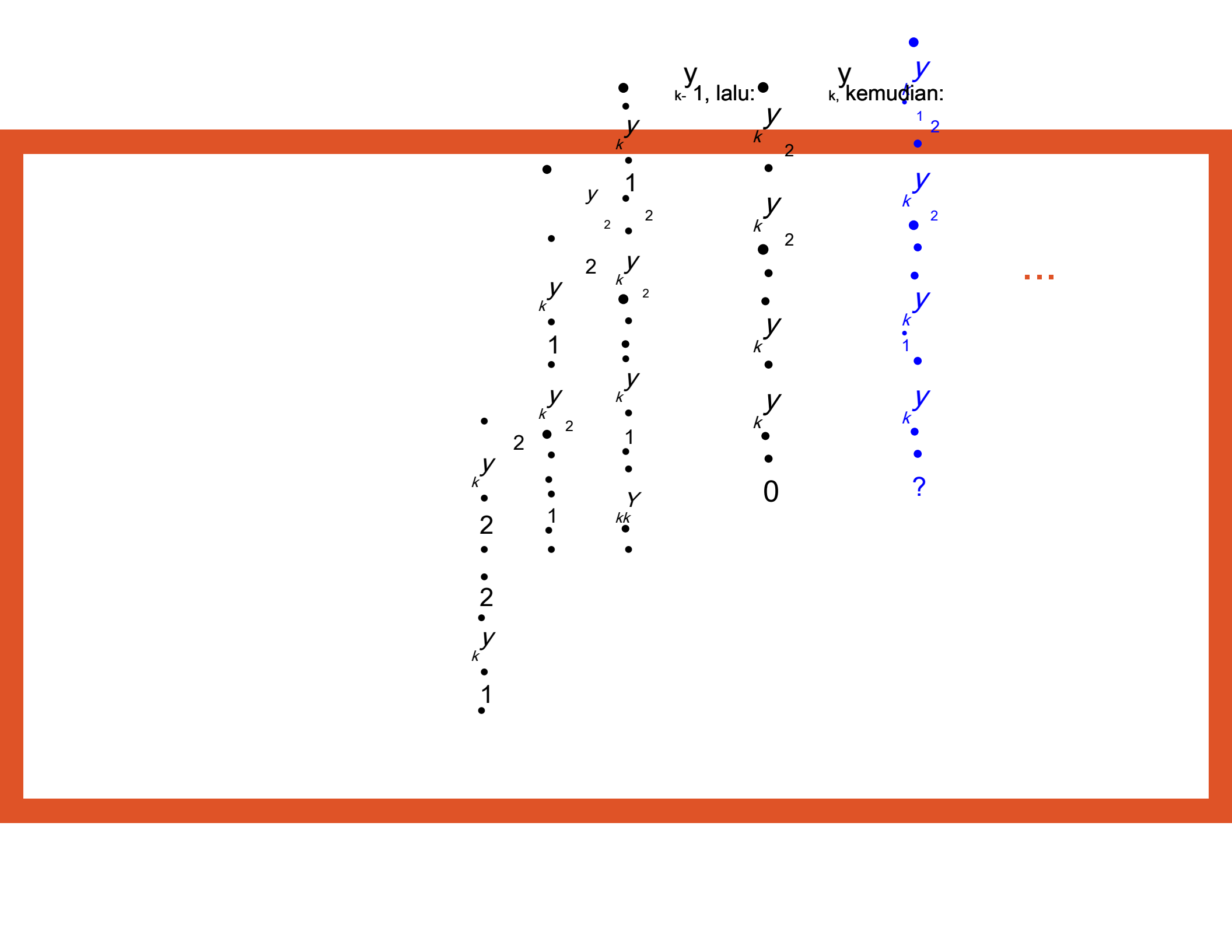
• x_k • 1 • 1 • • y_k • 1 2 • y_k 2 • • • y_k • 1 • y_k •

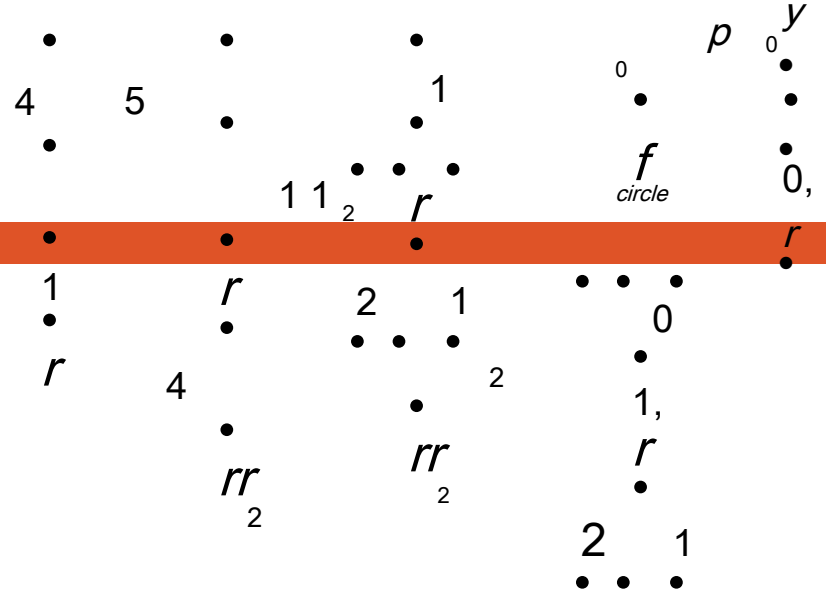
hal k • (2 1 2 x_k • 2) • • • y_k • 1 2 • y_k 2 • • • y_k • 1 • yp_{kk} •

1 • x_k 2 • 4 x_k • 4 • y_k • 1 2 • y_k • 1 • 4 1 • r 2

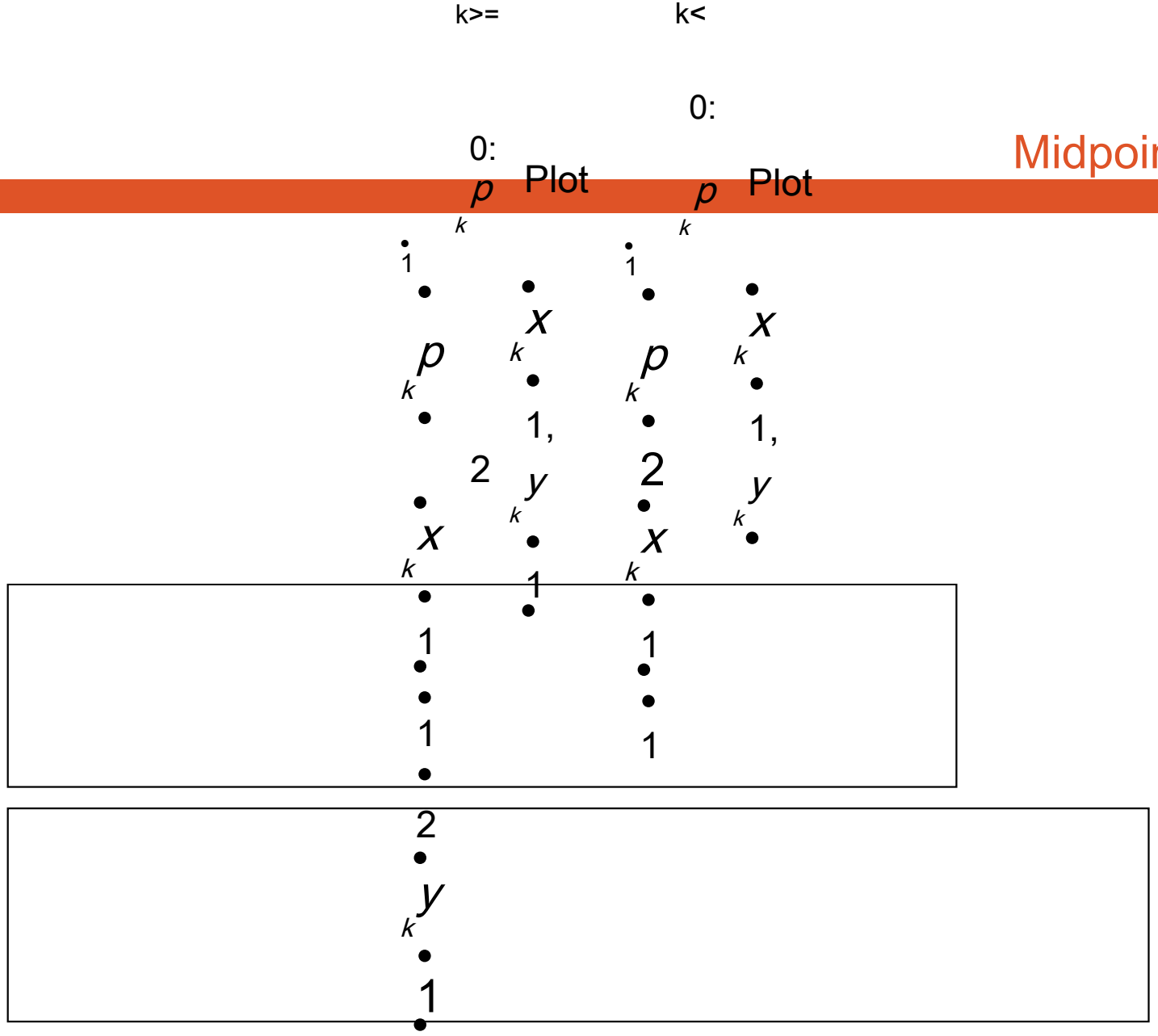
k • x_k 2 • 2 x_k • 1 • — y_k 2 • y_k • 4 1 — • r 2

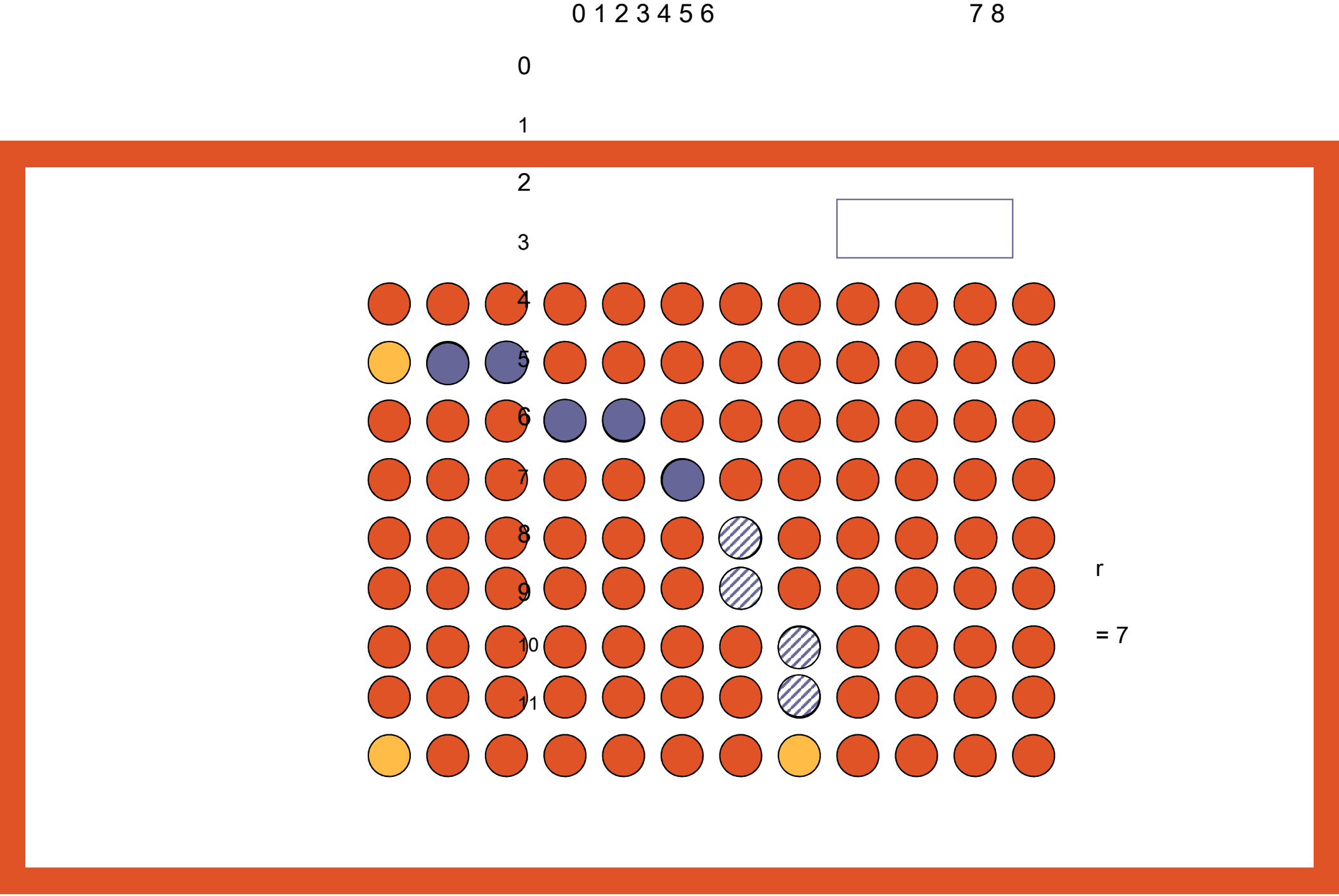
k


y k- 1, lalu:
y k, kemudian:
?
0
...


p
y
0
0
4
5
1
f
circle
0,
r
1 1
2
r
1
r
2
1
0
r
4
2
1,
r
rr
2
rr
2
2
1

Midpoint Circle Alg




0 1 2 3 4 5 6
7 8
0
1
2
3
4
5
6
7
8
9
10
11
r
= 7

b) (-a, -b)

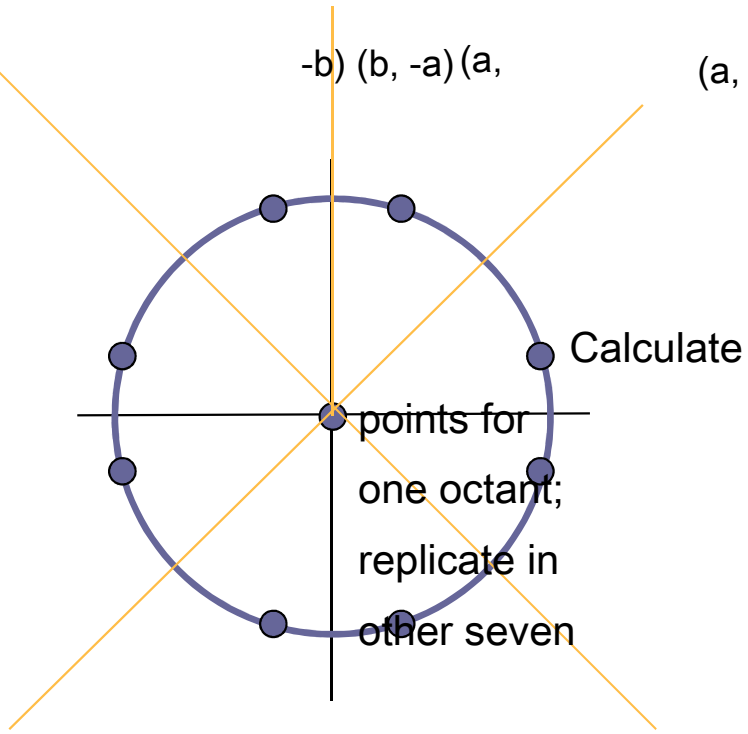
(-b, -a)

(-b, a)

(b, a)

-b) (b, -a) (a,

(a, b)



Calculate points for one octant; replicate in other seven

Midpoint Circle Algorithm

1. Input radius r and circle center (x_c, y_c), and obtain the first point on the circumference of a circle centered on the origin as

$$(x_0, y_0) = (0, r)$$

2. Calculate the initial value of the decision parameter as

$$p_0 = \frac{5}{4} - r$$

3. At each x_k position, starting at $k = 0$, perform the following test: If $p_k < 0$, the next point along the circle centered on $(0, 0)$ is (x_{k+1}, y_k) and

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

   Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and

$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$

   where $2x_{k+1} = 2x_k + 2$ and $2y_{k+1} = 2y_k - 2$.

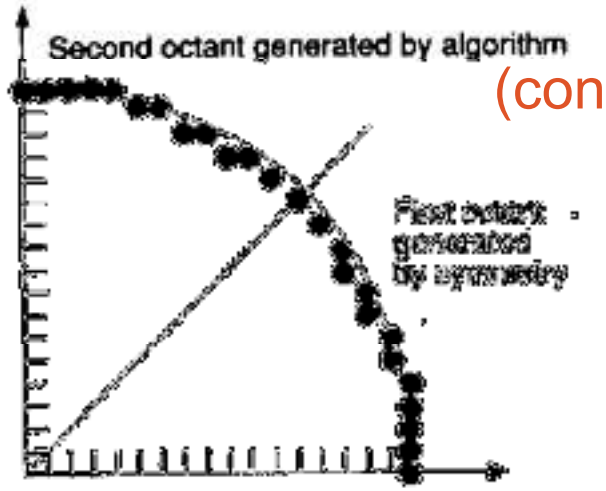
4. Determine symmetry points in the other seven octants.
5. Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot the coordinate values:

$$x = x + x_c, \qquad y = y + y_c$$

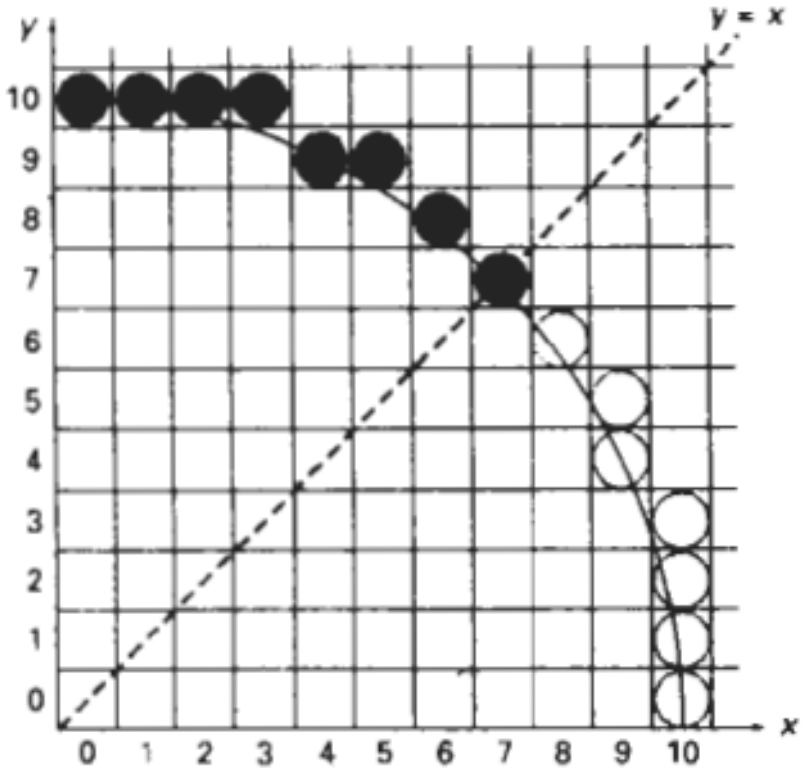
6. Repeat steps 3 through 5 until $x \geq y$.

```
void MidpointCircle (int radius, int value)
/* Assumes center of circle is at origin. Integer arithmetic only */
{
    int x = 0;
    int y = radius;
    int d = 1 - radius;
    CircleFcns (x, y, value);

    while (y > x) {
        if (d < 0)              /* Select E */
            d += 2 * x + 3;
        else {                  /* Select SE */
            d += 2 * (x - y) + 5;
            y--;
        }
        x++;
        CircleFcns (x, y, value);
    }                           /* while */
}                               /* MidpointCircle */
```



(cont.)


y
y = x
10
9
8
7
6
5
4
3
2
1
0
0 1 2 3 4 5 6 7 8 9 10
x

(7, 7) (6, 8) (5, 9) (4, 9) (3, 10) (2, 10) (1, 10) x_{i+1}, y

$i+1$

14 12 10 8 6 4 2 1 $2x_{i+}$ 0, y – r to integer) Initial point (x

0) = (0, 10)

14 16 18 18 20 20 20 1 $2y_{i+}$

0 1 2 3 4 5 6 7 8 9 10

Example(M

0 1 2 3 4 5 6 7 8 9 10

49

0= 5/4

id

Point Circle Algorithm

it using a polyline.

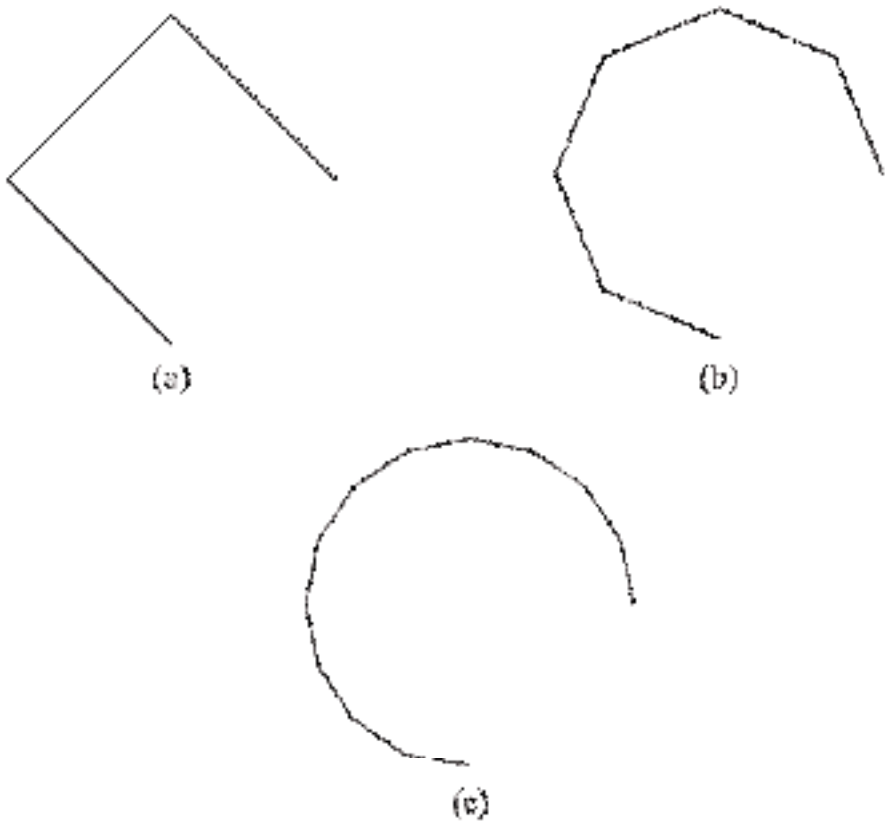


Figure 3-15

A circular arc approximated with (a) three straight-line segments, (b) six line segments, and (c) twelve line segments.

POLYGON

point and terminal point of any polyline is same, i,e

starting

- A chain of connected line segments.

hen

Filled Area Pr

Flood Fill

Boundary Fill

we encounter the specified boundary conditions. Useful with more complex boundaries a

Method: Start from

a given interior position and paint outward from

the

Filled Area

overlap intervals for scan lines that cross the area. It is typi

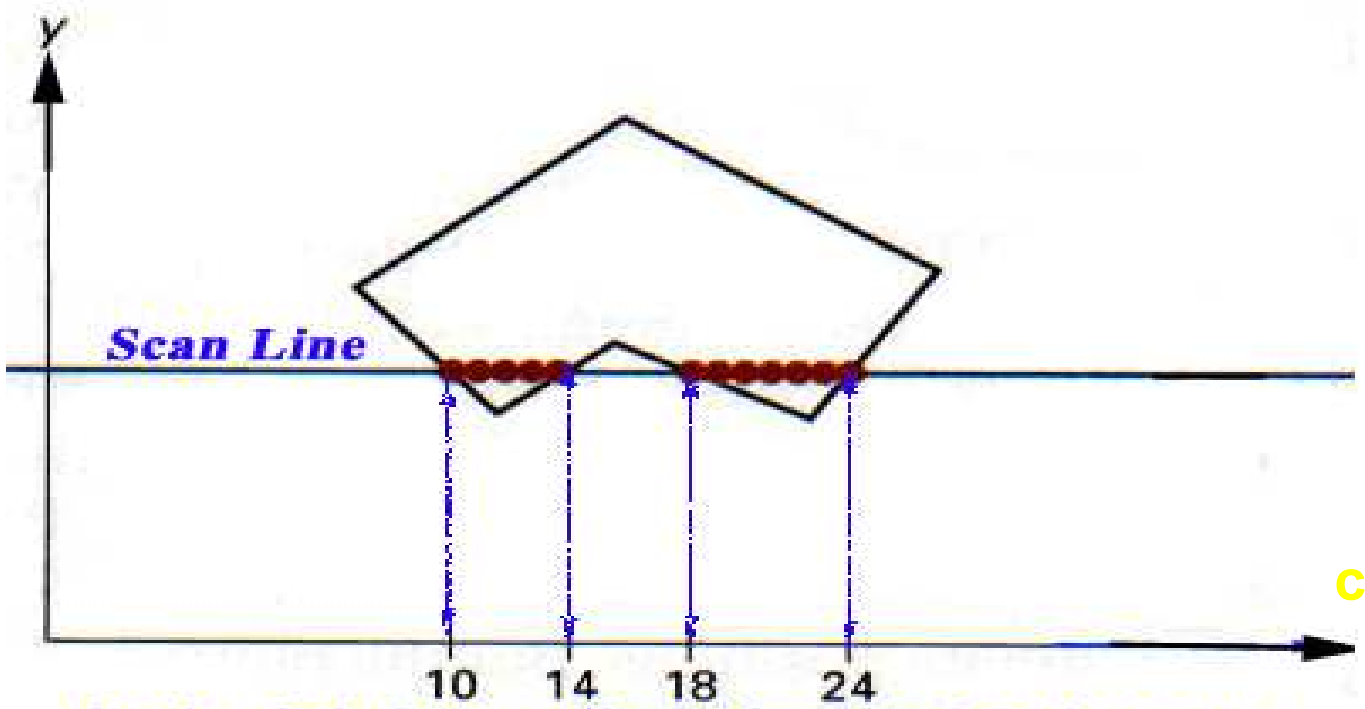
until

point

cont..

this

Filled Area



Interior pixels along scan line passing through a polygon area

cont..

at a

Filled Area

is useful in interactive painting packages, where interior points are easily se

point inside a region and paint the interior outward towa

proceeds

are:

outward pixel by pixel until the boundary color is encountered.

cont..

color value.

we want to

This approach is called a flood-fill algorithm. We start from

a

specified

fill in (or

Filled Area

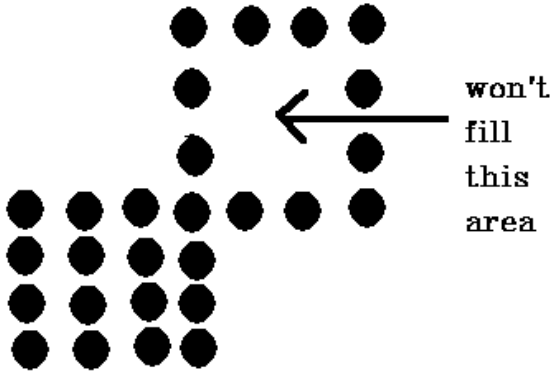
interior point (*x,y)* and reassign all pixel values that are currently set to a given interior co

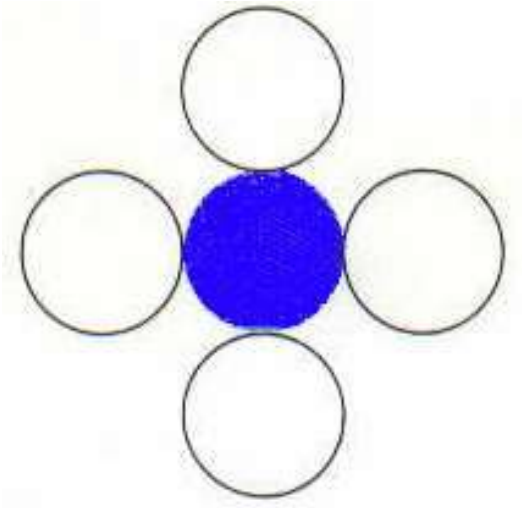
recolor) an

area that is not defined within a single color boundary

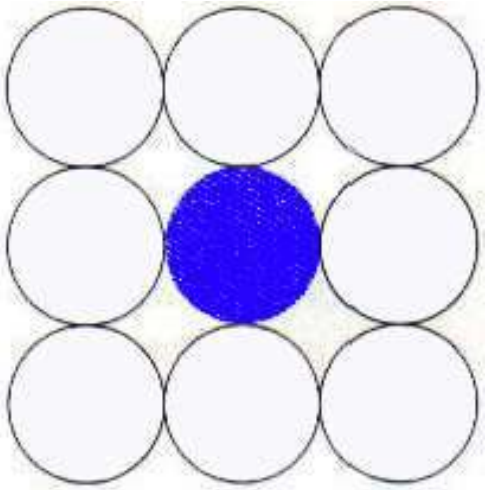
cont..

this





technique 4-connected pixels are used as shown in



Sometimes we come across an object where we want to fill the area and its boundary with different colors. We can paint such objects with a specified interior color instead of searching for particular boundary color as in boundary filling algorithm.

Instead of relying on the boundary of the object, it relies on the fill color. In other words, it replaces the interior color of the object with the fill color. When no more pixels of the original interior color exist, the algorithm is completed.

Once again, this algorithm relies on the Four-connect or Eight-connect method of filling in the pixels. But instead of looking for the boundary color, it is looking for all adjacent pixels that are a part of the interior.