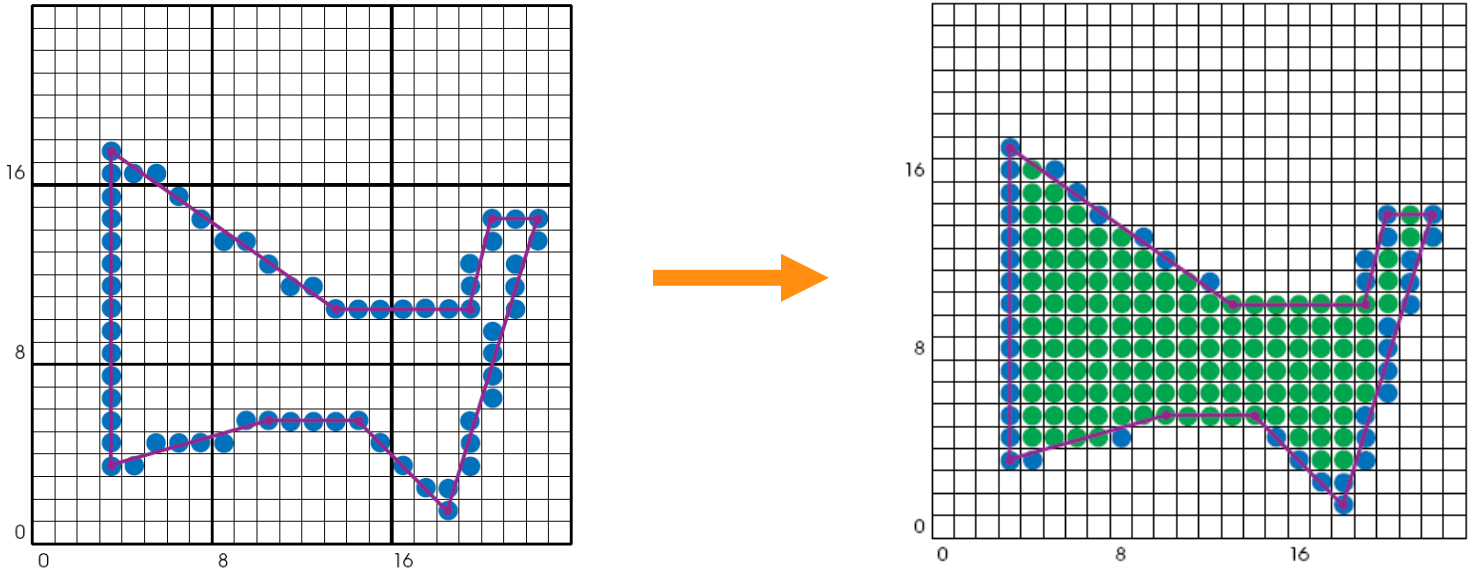
Računarska grafika
(2OER7O02)

Popuna (*filling*) objekata

Predavanja

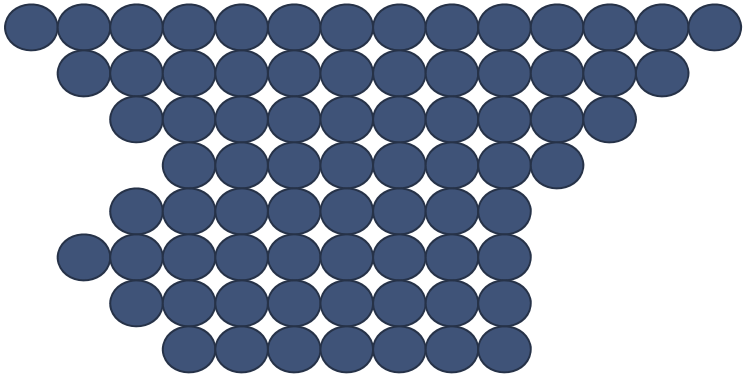
Popunjavanje oblasti (regiona)

- **Oblast** (region) je grupa susednih, povezanih piksela.



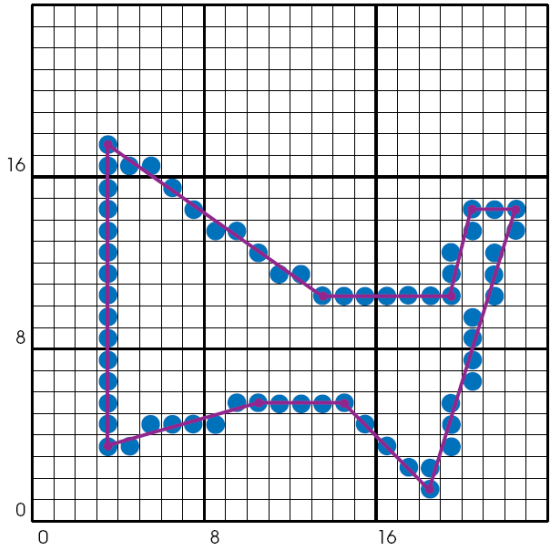
Popunjavanje oblasti (regiona)

- Uobičajeno, oblast se definiše na jedan od dva načina:
  - Svi pikseli koji pripadaju oblasti imaju istu (datu) vrednost;



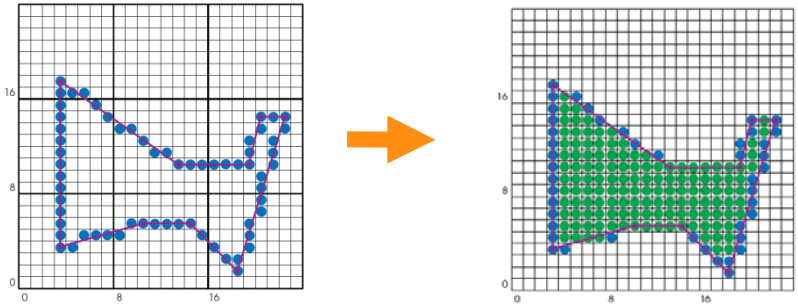
Popunjavanje oblasti (regiona)

- Pikseli koji su na ivici oblasti imaju datu vrednost.



Tipovi oblasti

- Prema načinu definisanja
- Prema povezanosti piksela

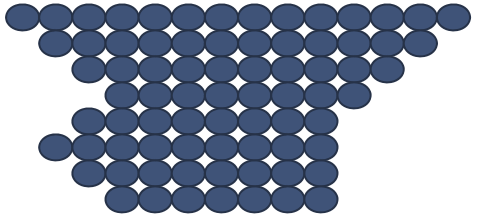


Tipovi oblasti prema načinu definisanja

- **Interior-defined** (oblast definisana unutrašnjošću)
- **Boundary-defined** (oblast definisana granicom)

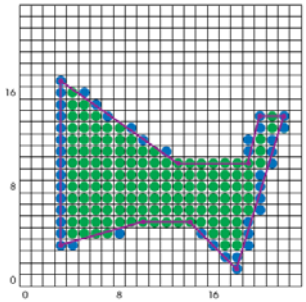
Interior-defined oblasti

- Svi pikseli u oblasti imaju vrednost old_value, i nema piksela na granici oblasti sa tom vrednošću.
- Algoritmi za popunu koji rade nad ovakvim oblastima treba da postave njihove piksele na vrednost new_value.
- Ovakvi algoritmi se nazivaju algoritmima popunjavanja plavljenjem (flood-fill).



Boundary-defined oblasti

- Pikseli na granici oblasti imaju vrednost boundary_value, dok pikseli u oblasti imaju proizvoljnu drugu vrednost.
- Algoritmi koji rade nad ovakvim oblastima treba da postave sve njihove piksele na vrednost new_value
- Ovi algoritmi se nazivaju algoritmima popunjavanja oivičene oblasti (boundary-fill).

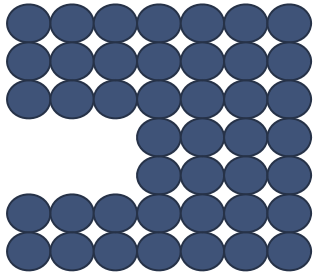


Tipovi oblasti prema povezanosti piksela

- **4-susedne** oblasti (*4-connected*)
- **8-susedne** oblasti (*8-connected*)

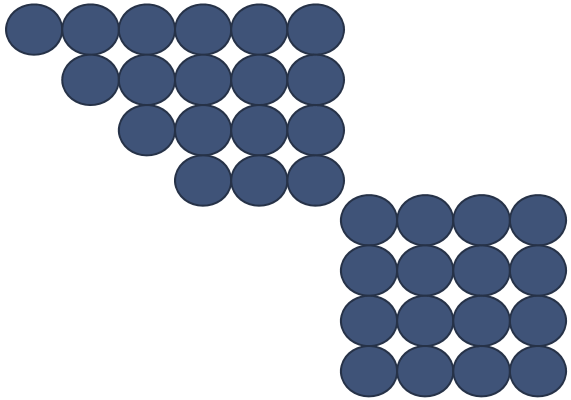
4-susedne oblasti

- Od proizvoljnog piksela u oblasti do svih drugih u oblasti može se stići sekvencom horizontalnih i vertikalnih pomeraja za 1 piksel.



8-susedne oblasti

- Od proizvoljnog piksela u oblasti do svih drugih u oblasti može se stići sekvencom horizontalnih, vertikalnih i dijagonalnih pomeraja za 1 piksel



Algoritmi za popunu

- Flood-fill algoritmi
- Algoritmi za popunu oivičenih oblasti

Rekurzivni Flood-fill algoritam za 4-susedne oblasti

```
void Flood_Fill4(CDC* pDC, int x, int y, COLORREF old_value, COLORREF
new_value)
{
   if (ReadPixel(pDC,x,y) == old_value){
         WritePixel(pDC,x,y,new_value);
         Flood_Fill4(pDC, x, y-1, old_value, new_value);
         Flood_Fill4(pDC, x, y+1, old_value, new_value);
         Flood_Fill4(pDC,x+1, y,  old_value, new_value);
         Flood_Fill4(pDC,x-1, y,  old_value, new_value);
   }
}

COLORREF ReadPixel(CDC* pDC, int x, int y)
{
   return pDC->GetPixel(x,y);
}
```

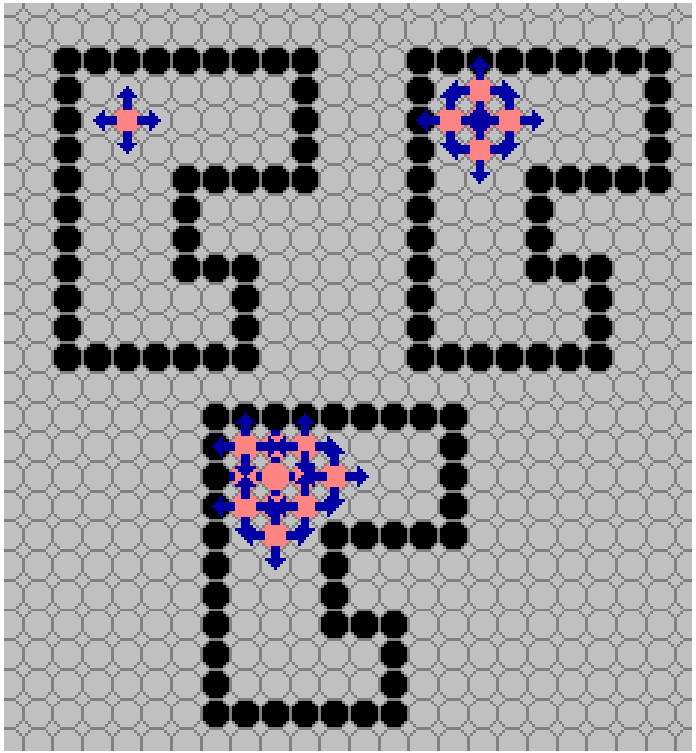
Rekurzivni Flood-fill algoritam za 8-susedne oblasti

```
void Flood_Fill8(CDC* pDC, int x, int y, COLORREF old_value,
COLORREF new_value)
{
   if (ReadPixel(pDC,x,y) == old_value){
         WritePixel(pDC,x,y,new_value);
         Flood_Fill8(pDC, x, y-1, old_value, new_value);
         Flood_Fill8(pDC, x, y+1, old_value, new_value);
         Flood_Fill8(pDC,x+1, y,  old_value, new_value);
         Flood_Fill8(pDC,x-1, y,  old_value, new_value);
         Flood_Fill8(pDC,x-1,y-1, old_value, new_value);
         Flood_fill8(pDC,x-1,y+1, old_value, new_value);
         Flood_fill8(pDC,x+1,y-1, old_value, new_value);
         Flood_fill8(pDC,x+1,y+1, old_value, new_value);
   }
}
```

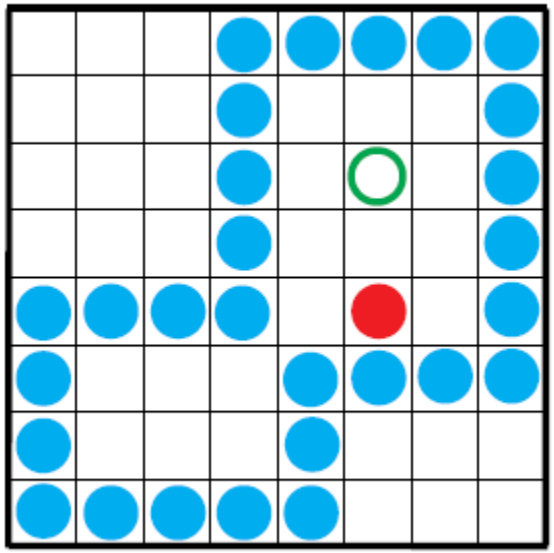
Rekurzivni algoritam za popunu oivičene 4-susedne oblasti

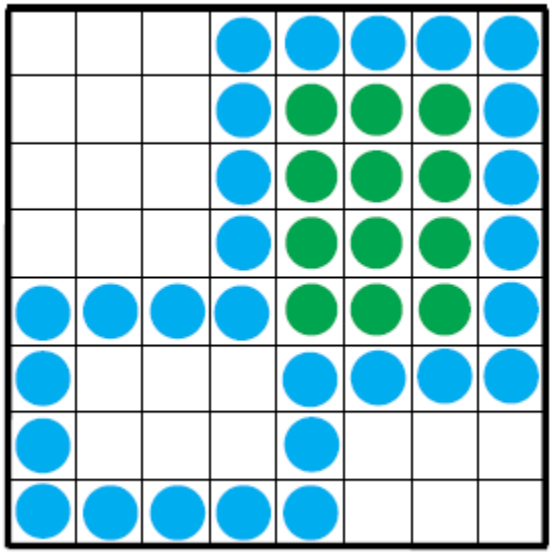
```
void Boundary_Fill4(CDC* pDC, int x, int y, COLORREF boundary_value,
COLORREF new_value)
{
   COLORREF temp_value = ReadPixel(pDC,x,y);
   if (temp_value != boundary_value) && (temp_value != new_value)){
         WritePixel(pDC,x,y,new_value);
         Boundary_Fill4(pDC, x, y-1, boundary_value, new_value);
         Boundary_Fill4(pDC, x, y+1, boundary_value, new_value);
         Boundary_Fill4(pDC,x+1, y,  boundary_value, new_value);
         Boundary_Fill4(pDC,x-1, y,  boundary_value, new_value);
}
```

Rekurzivni algoritam za popunu oivičene 4-susedne oblasti

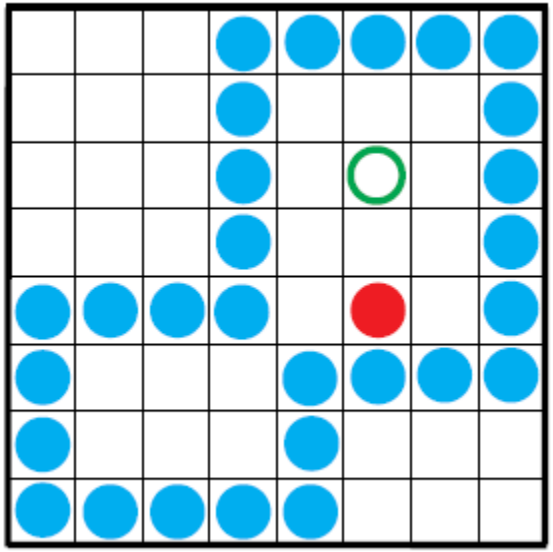


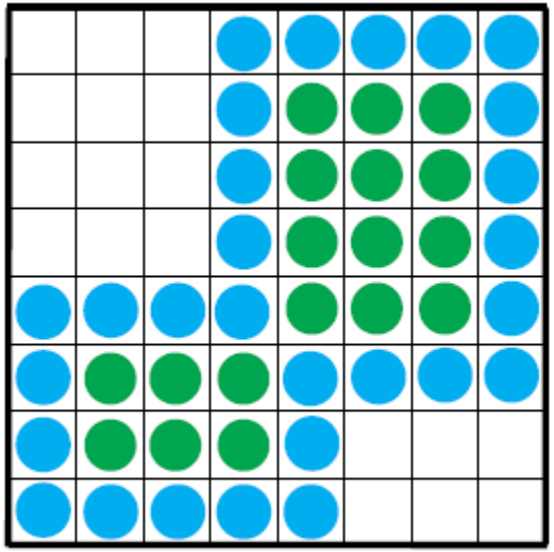
Rekurzivni algoritam za popunu oivičene 4-susedne oblasti





Rekurzivni algoritam za popunu oivičene 8-susedne oblasti

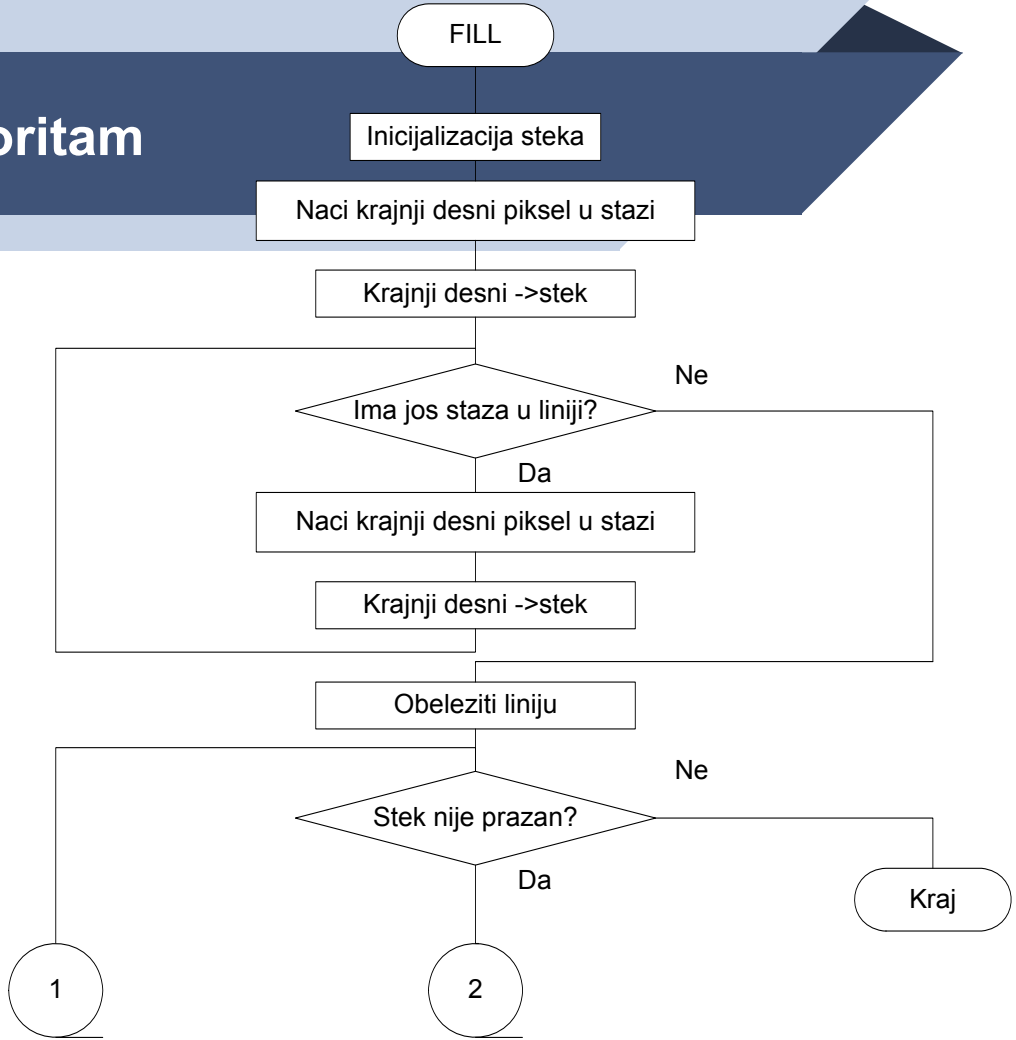




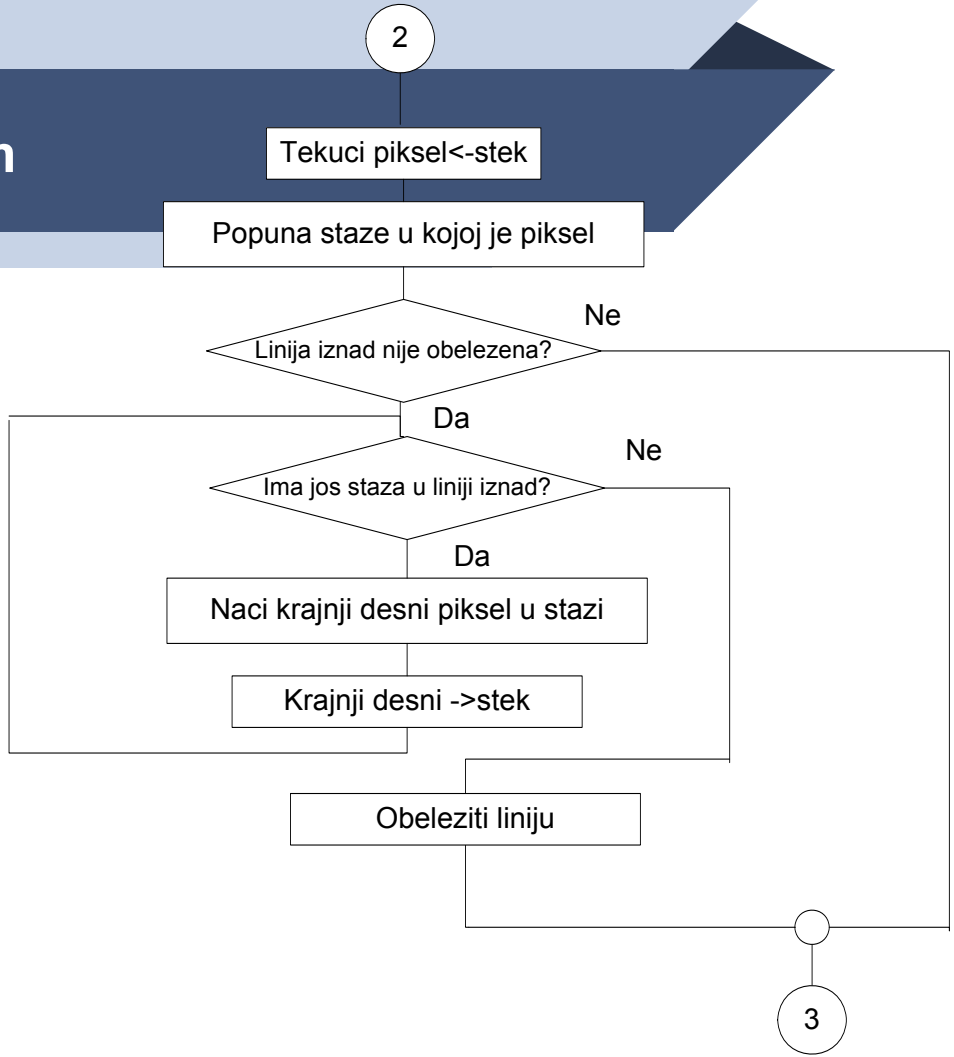
Nedostaci rekurzivnih algoritama za popunu oblasti

- **Brza popuna steka!!!**
- **Zbog toga se koriste iterativni algoritmi.**

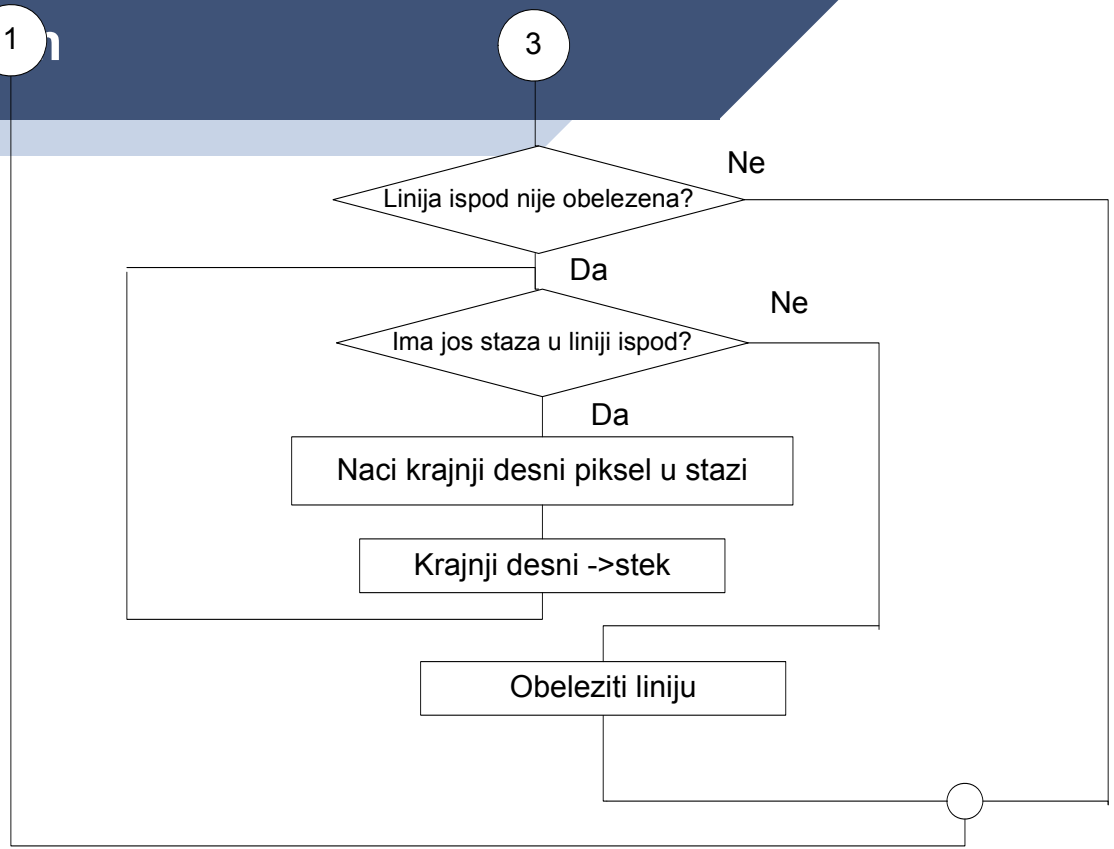
Iterativni algoritam



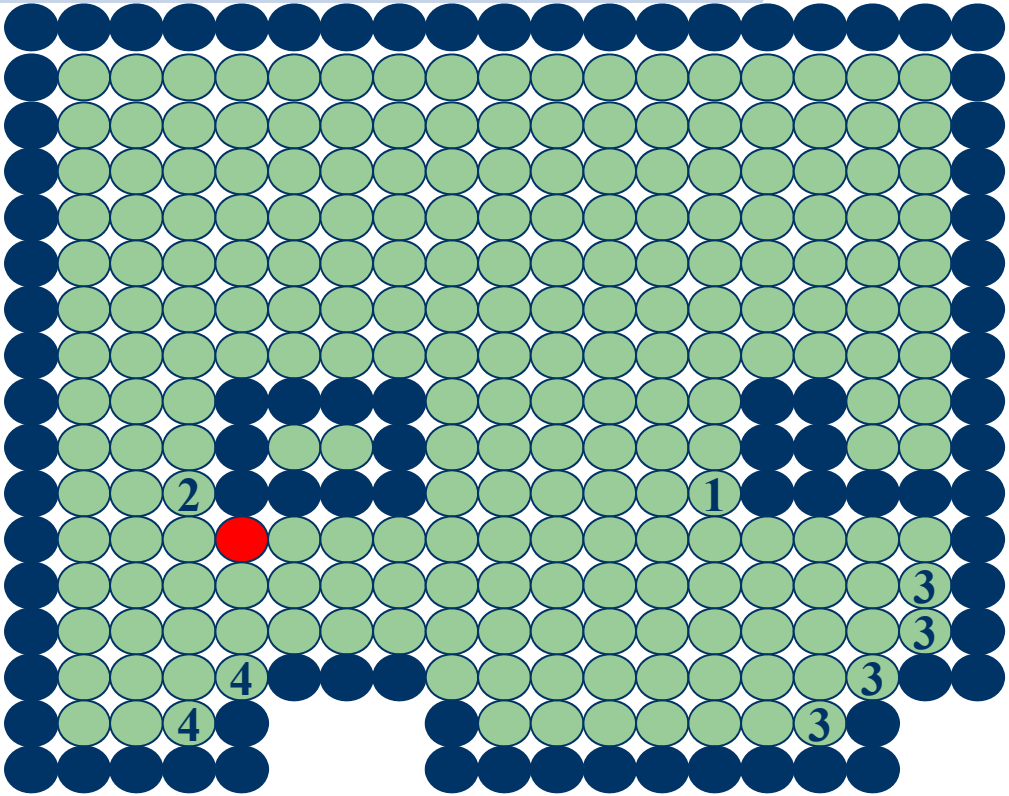
Iterativni algoritam



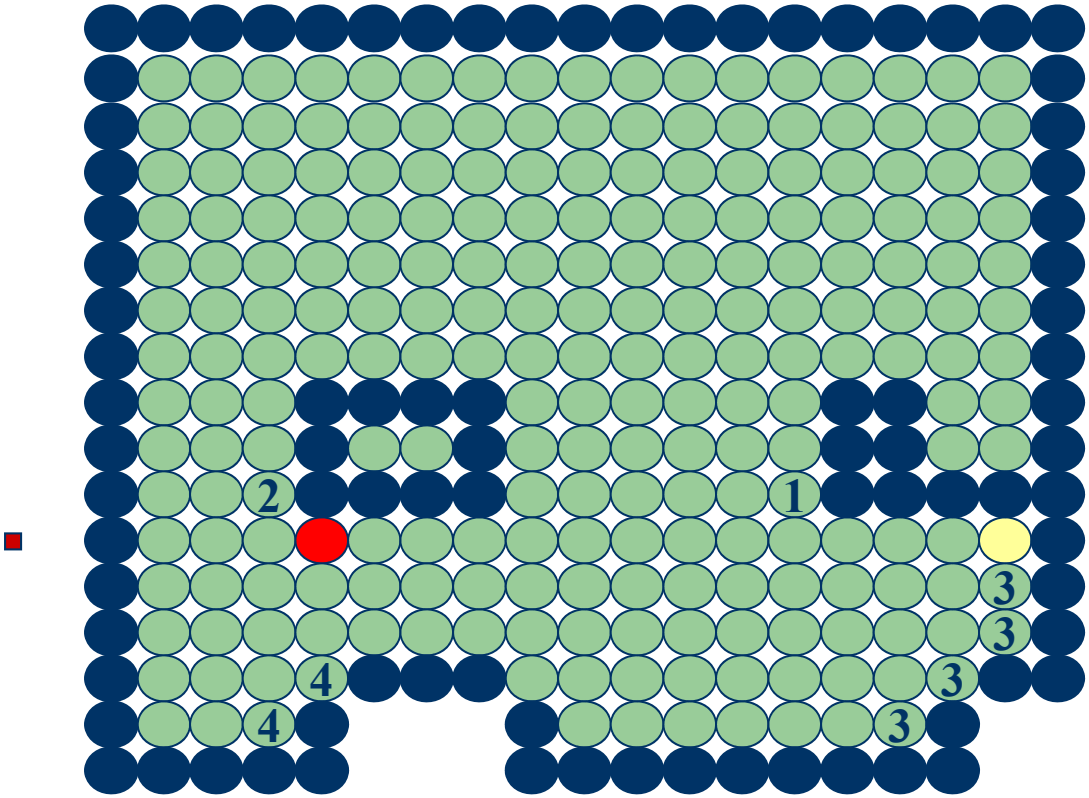
Iterativni algori[illegible]



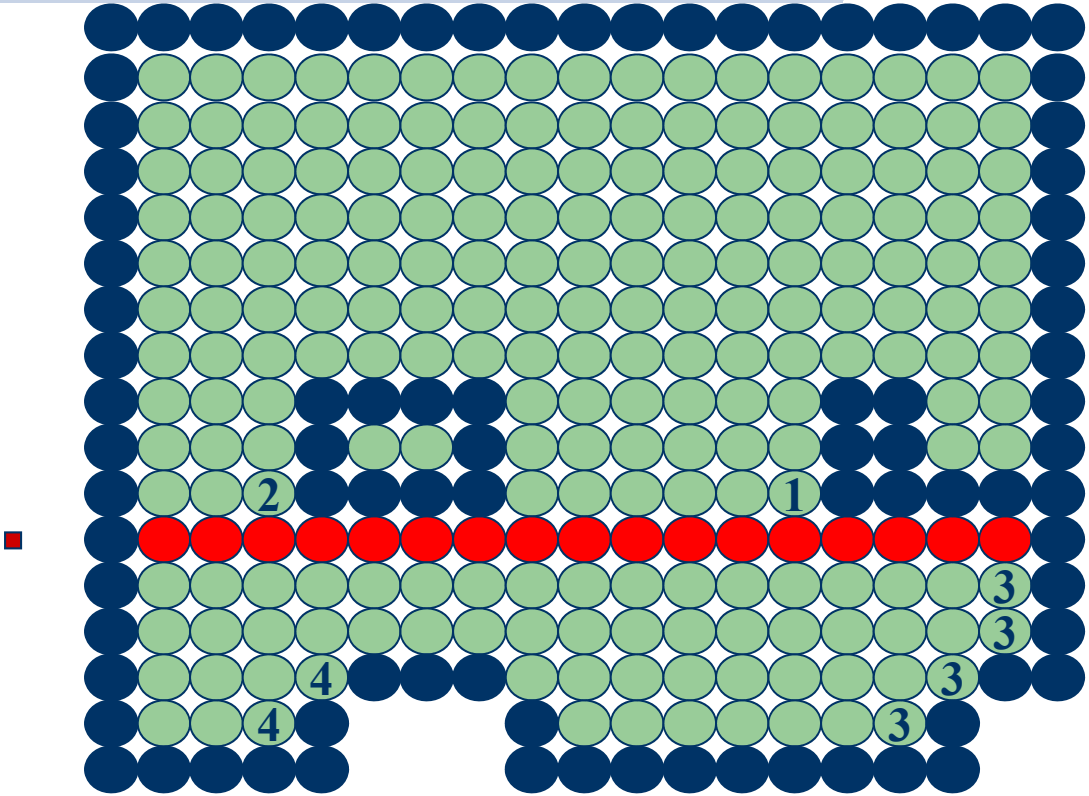
Iterativni algoritam



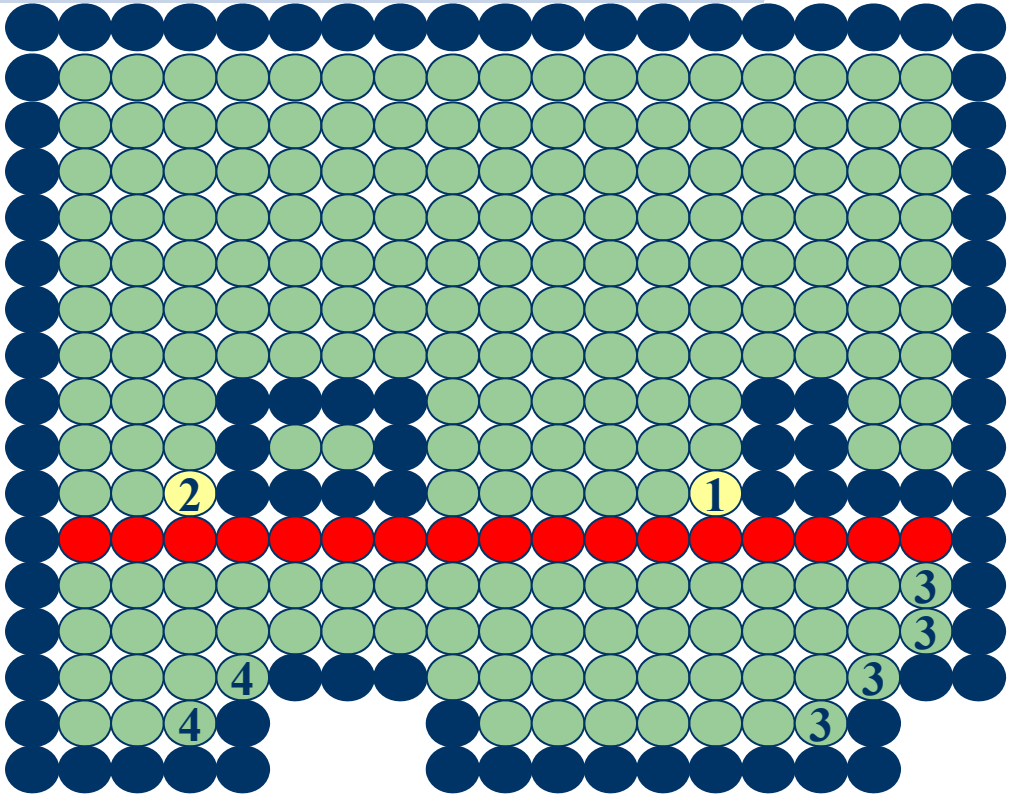
Iterativni algoritam



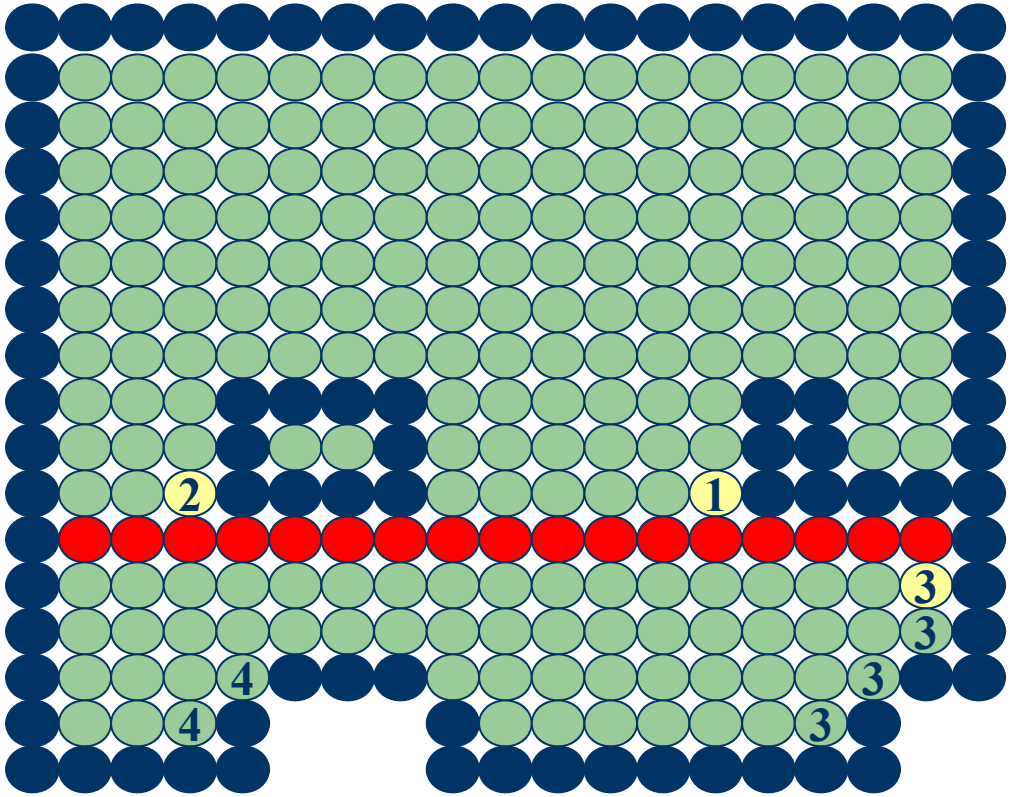
Iterativni algoritam



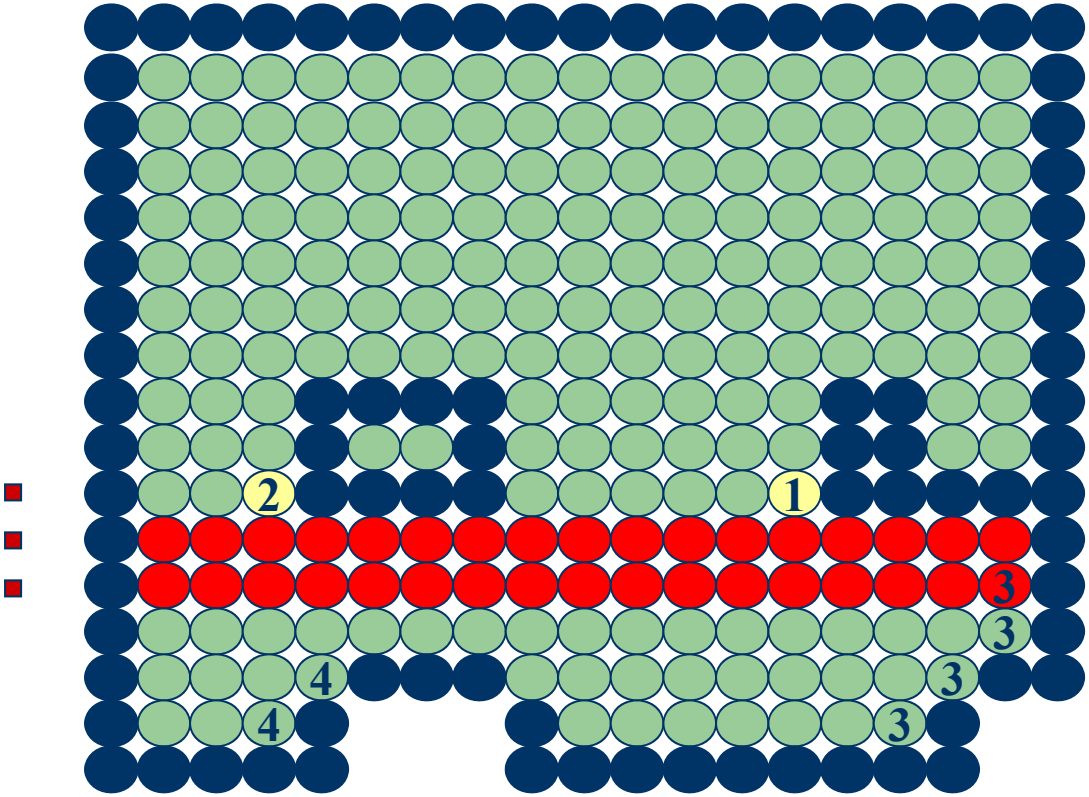
Iterativni algoritam



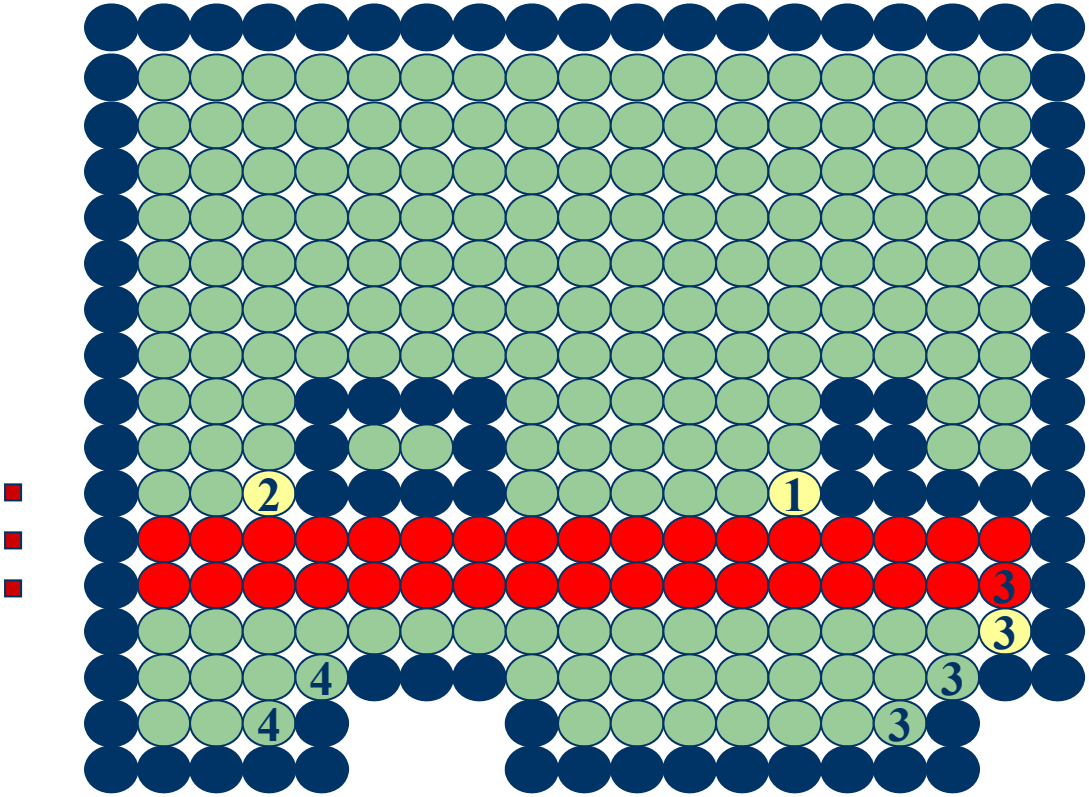
Iterativni algoritam



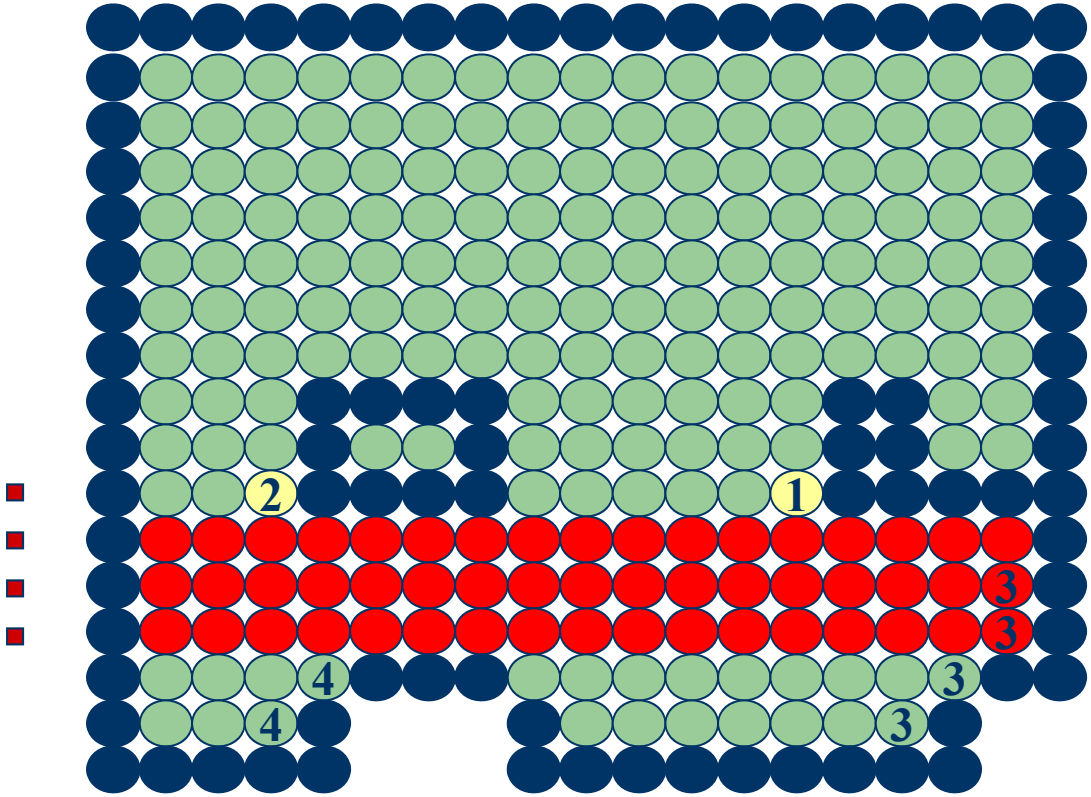
Iterativni algoritam



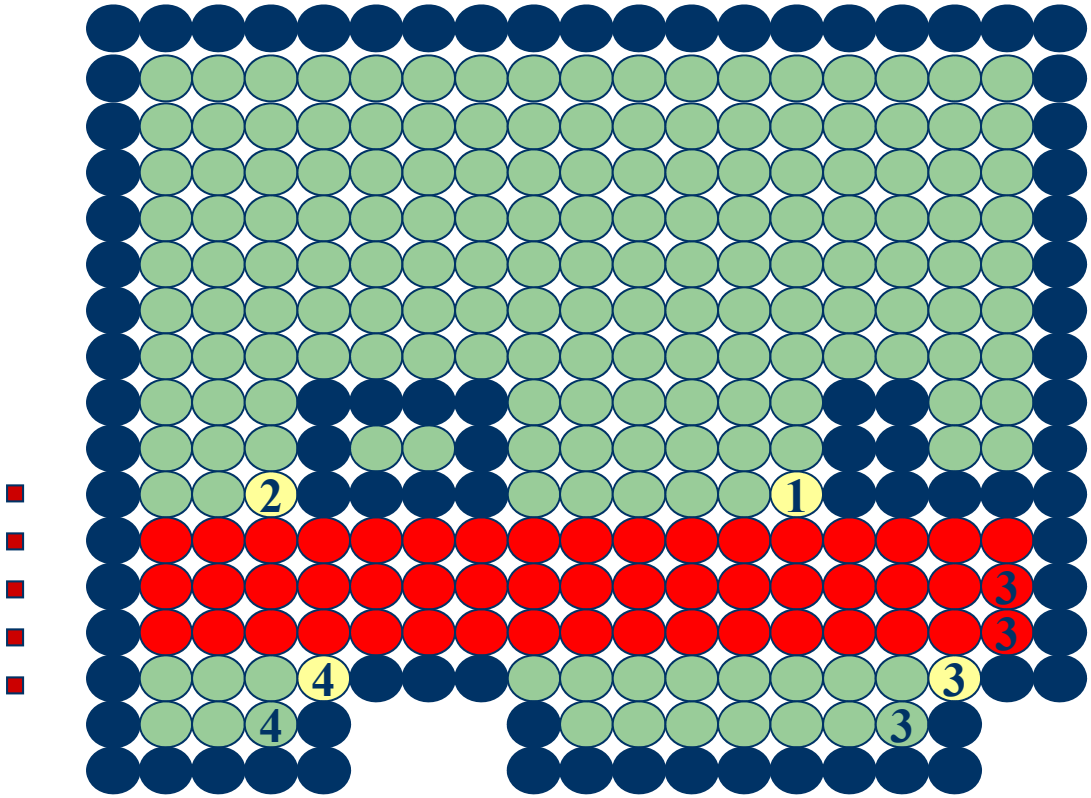
Iterativni algoritam



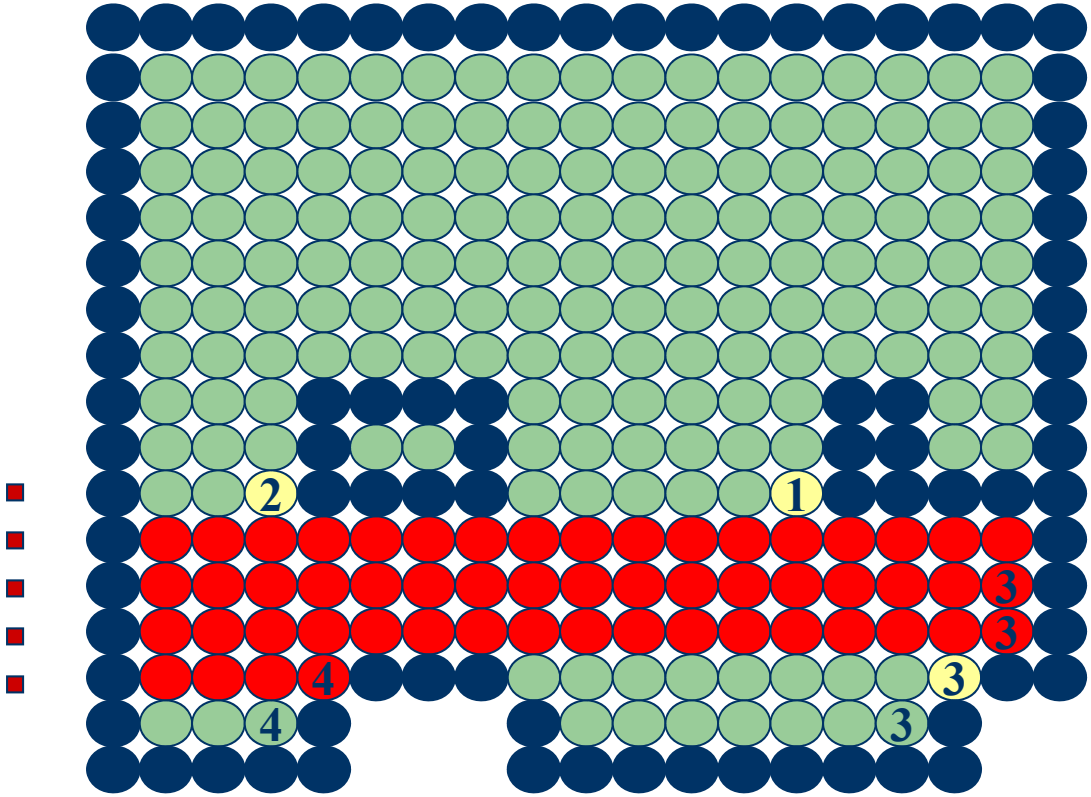
Iterativni algoritam



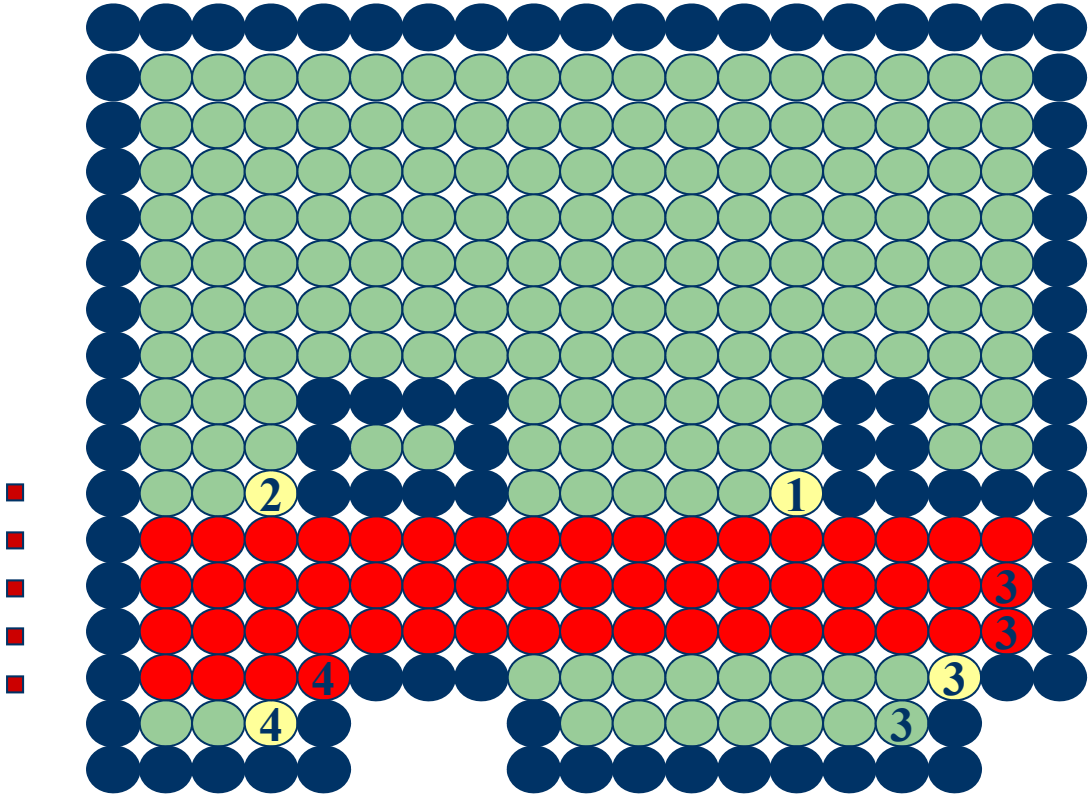
Iterativni algoritam



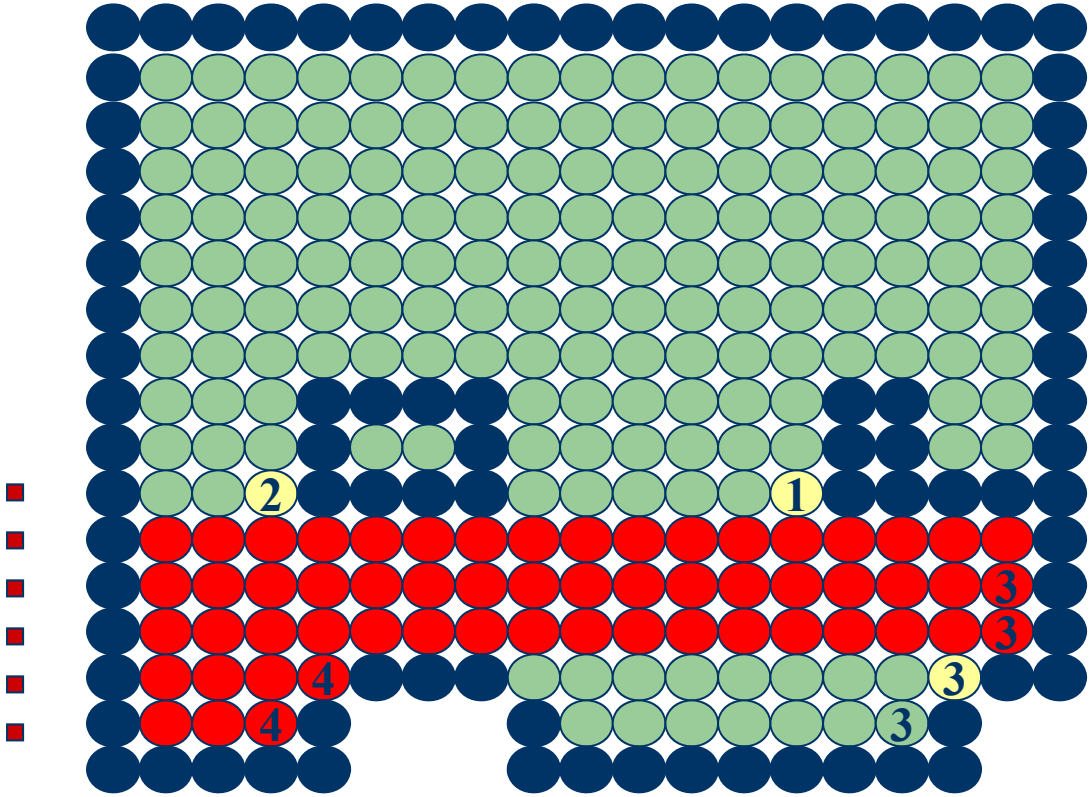
Iterativni algoritam



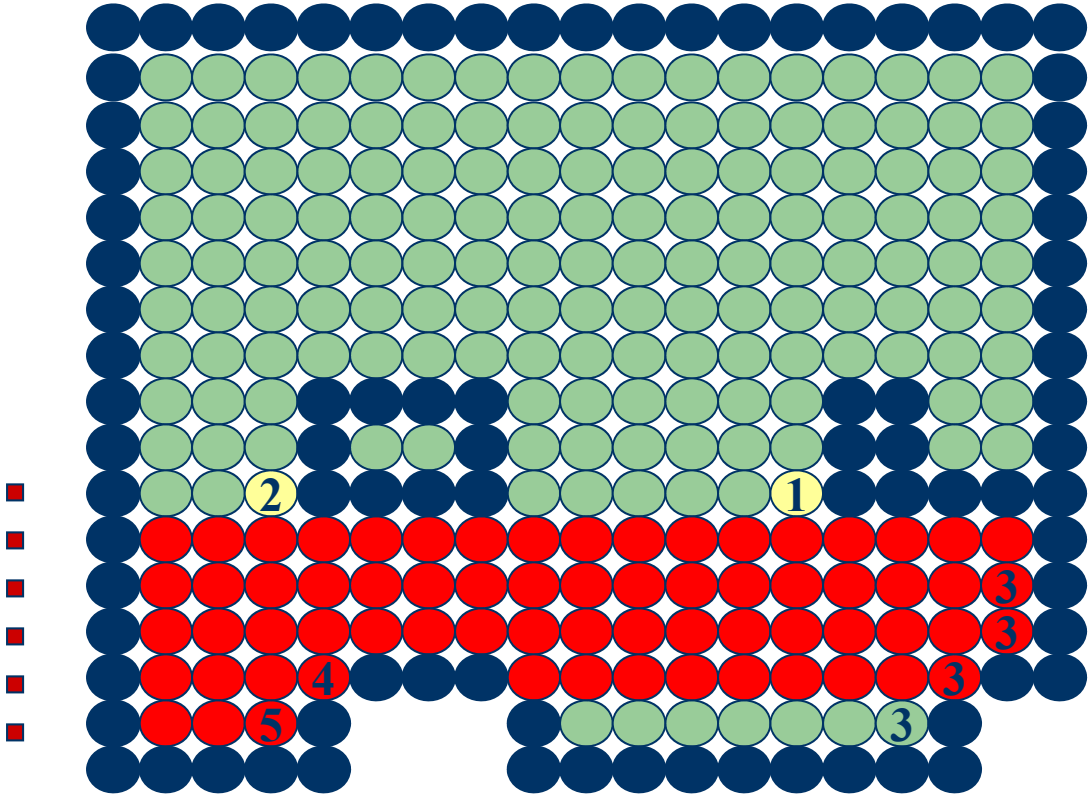
Iterativni algoritam



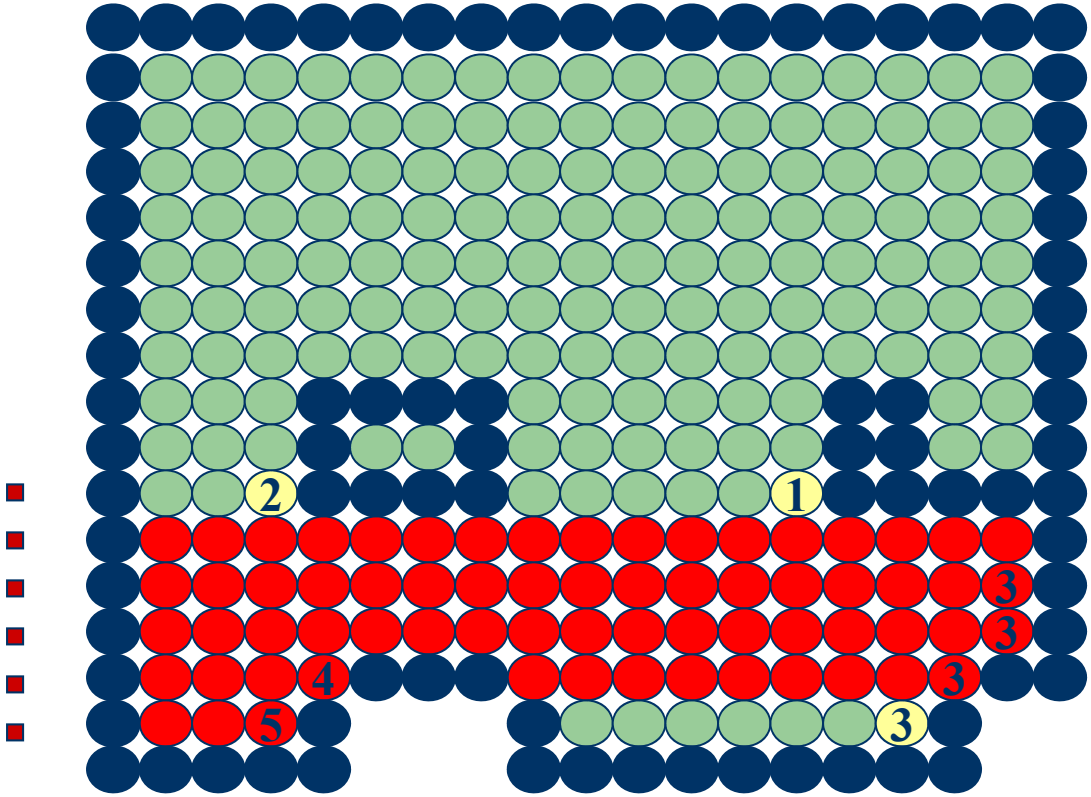
Iterativni algoritam



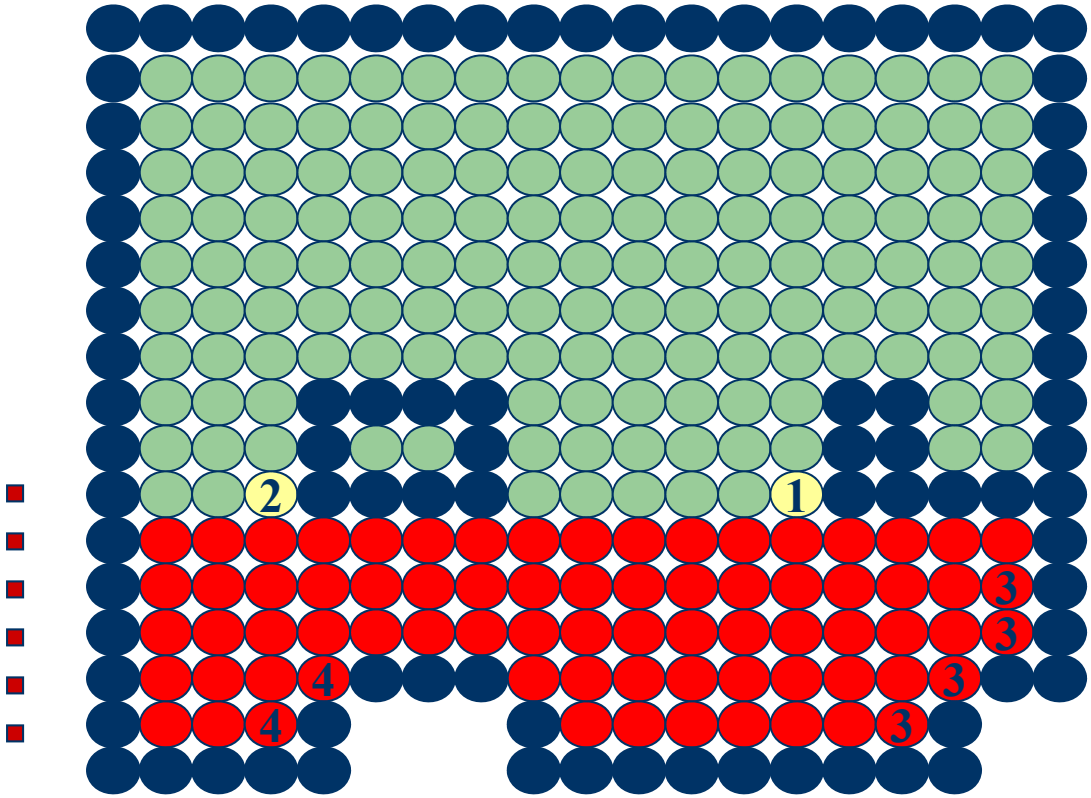
Iterativni algoritam



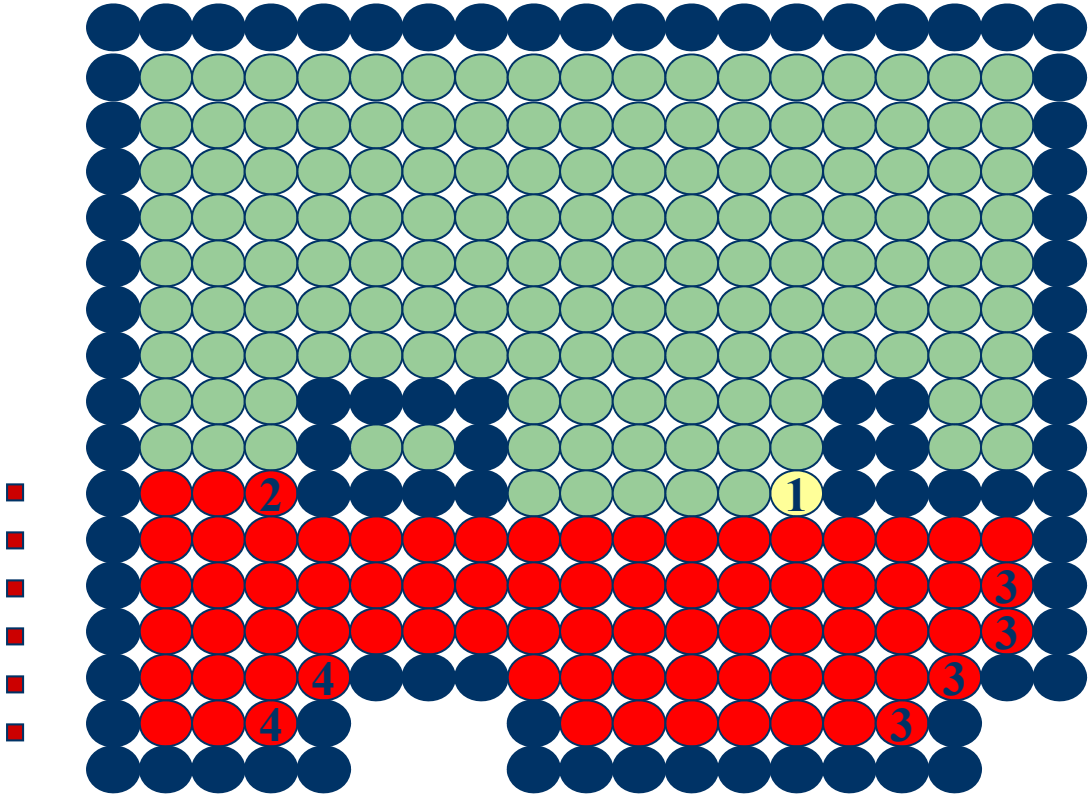
Iterativni algoritam



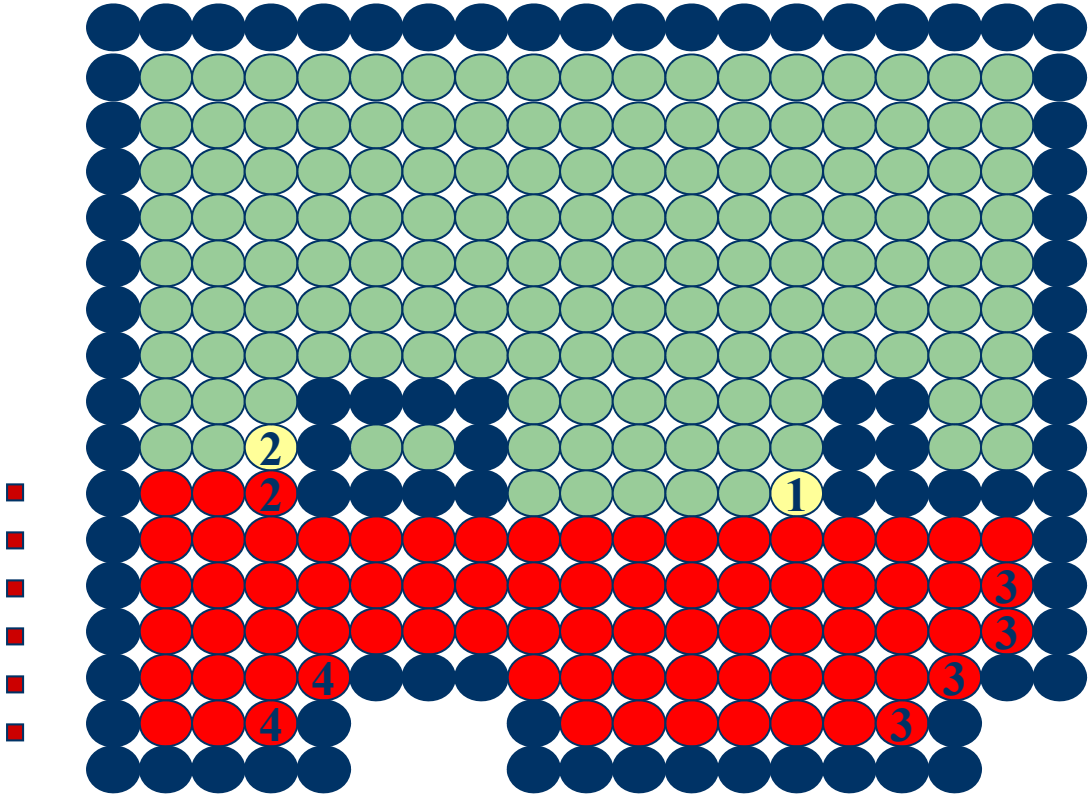
Iterativni algoritam



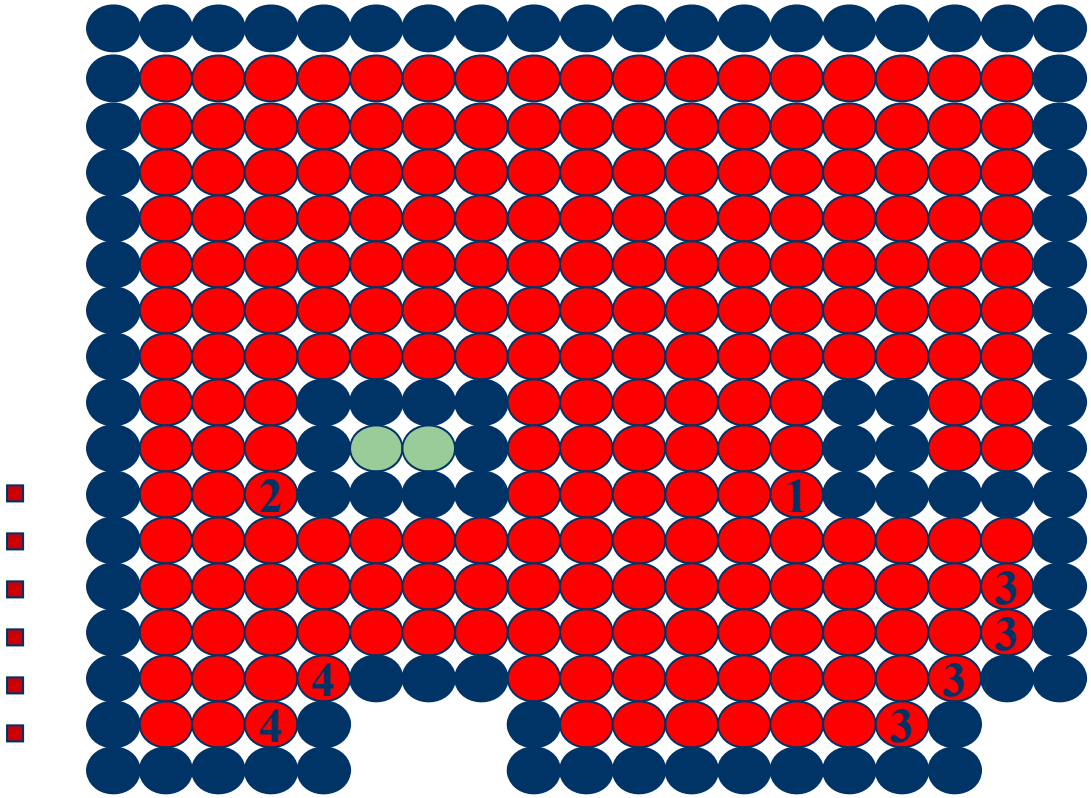
Iterativni algoritam



Iterativni algoritam



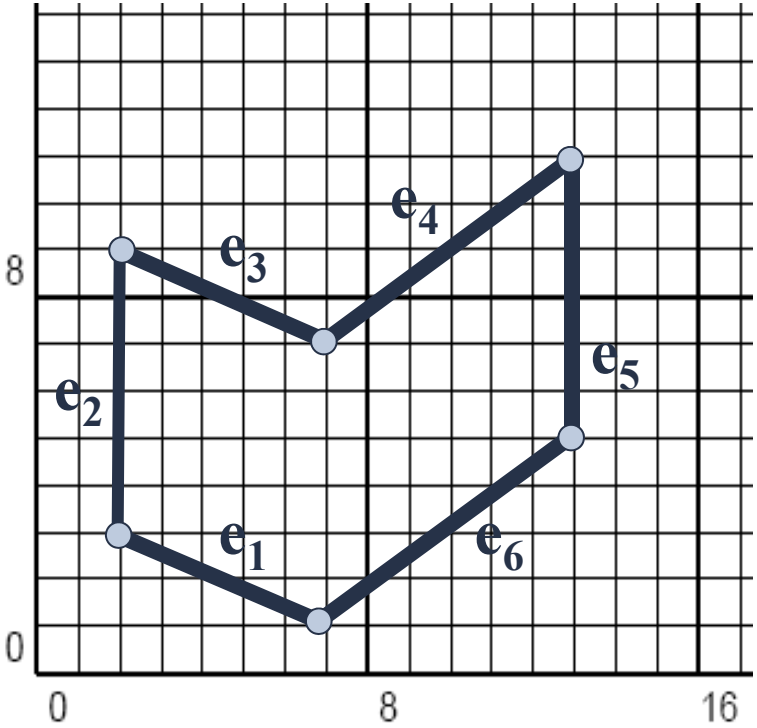
Iterativni algoritam



Popuna grafičkih primitiva

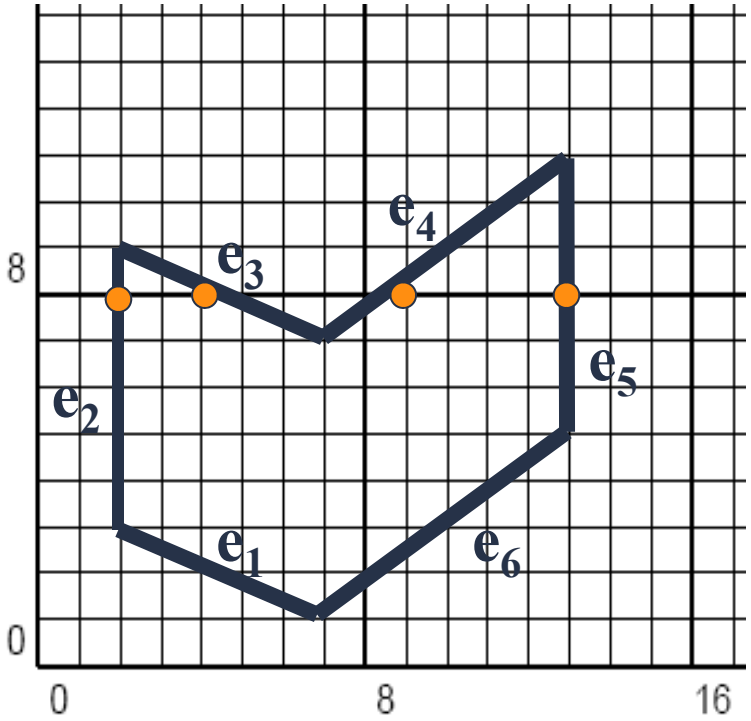
- Popuna poligona
- Popuna kruga
- Popuna elipse

Popuna poligona zadatog listom temena



Popuna poligona zadatog listom temena

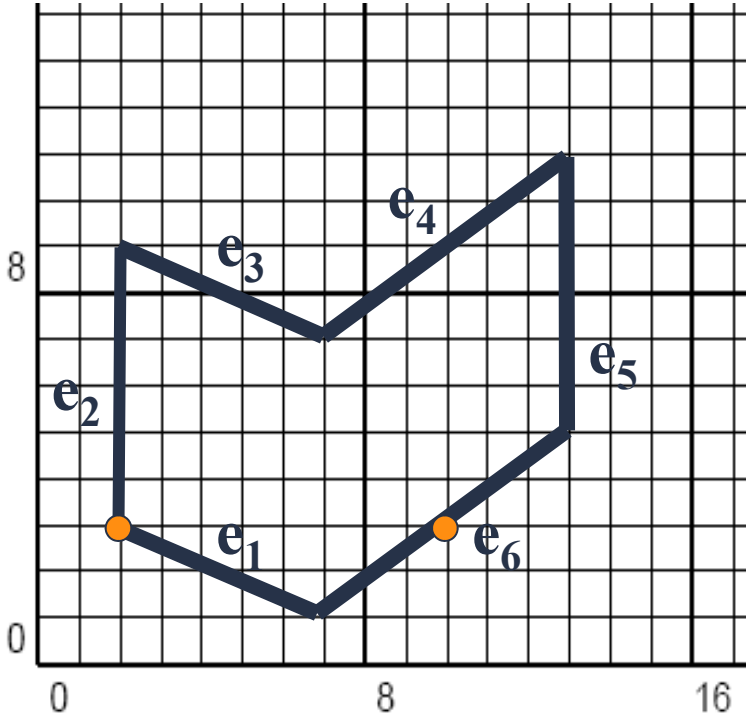
y = 8



x = 2,4,9,13

Popuna poligona zadatog listom temena

y = 3

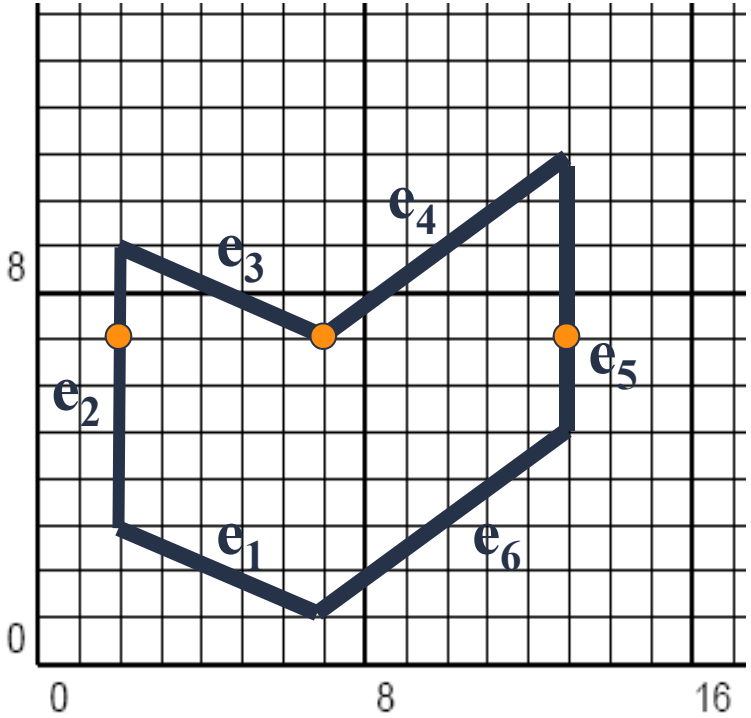


x = 2,2,10

Treba izbaciti jednu 2

Popuna poligona zadatog listom temena

y = 7



x = 2,7,7,13

Ne treba izbaciti jednu 7

Popuna poligona zadatog listom temena

Treba uzeti oba principa ali treba razlikovati ova dva slučaja:

Ako se dvostruki presek nalazi na **lokalnom minimumu** ili **maksimumu**, treba uzeti presek kao **dve tačke**. U ostalim slučajevima treba uzimati presek kao **jednu tačku**.

Popuna poligona zadatog listom temena

$$x_{i+1} = x_i + \frac{1}{n}$$

n – nagib linije

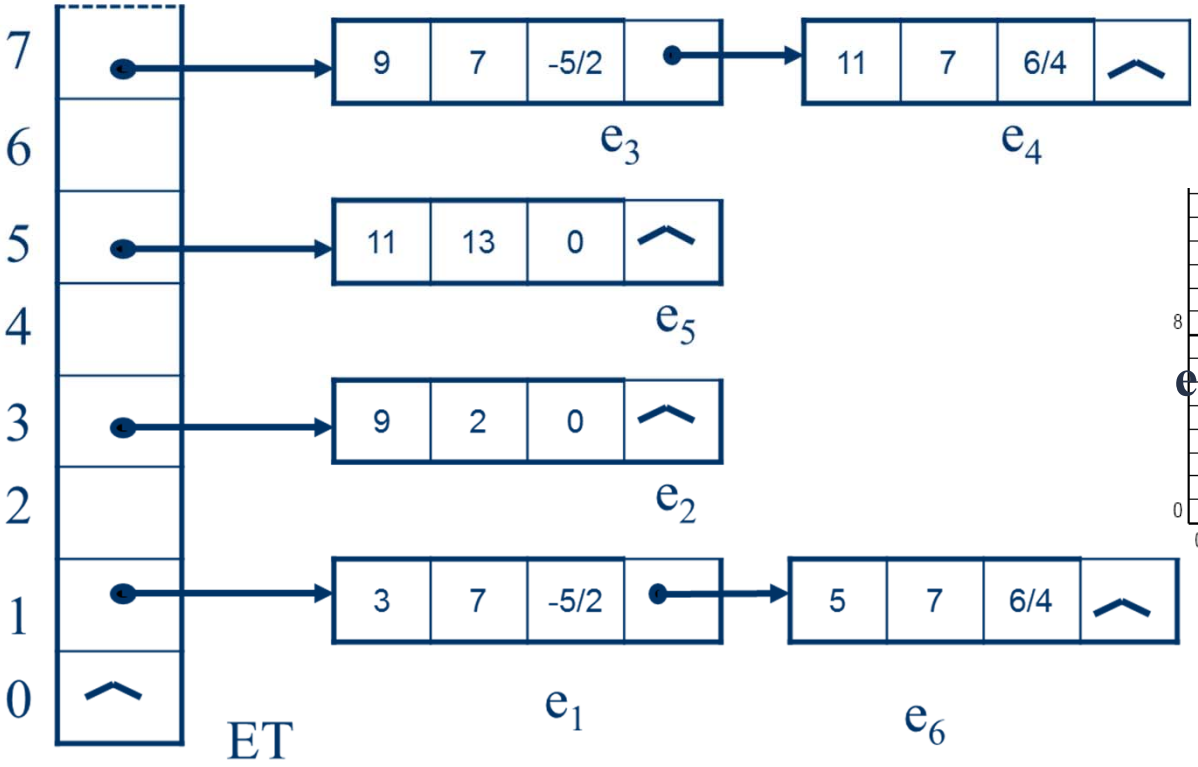
Popuna poligona zadatog listom temena - Algoritam

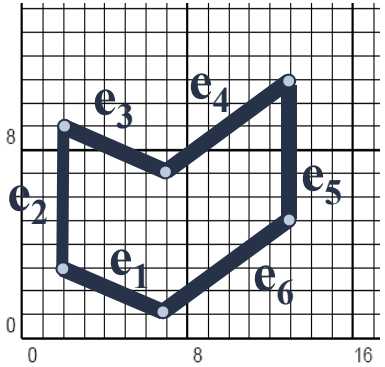
Koriste se dve tabele:

- Tabela ivica – **ET** (*Edge Table*) – tabela pointera na sken linije
- Tabela aktivnih ivica – **AET** (*Active Edge Table*)

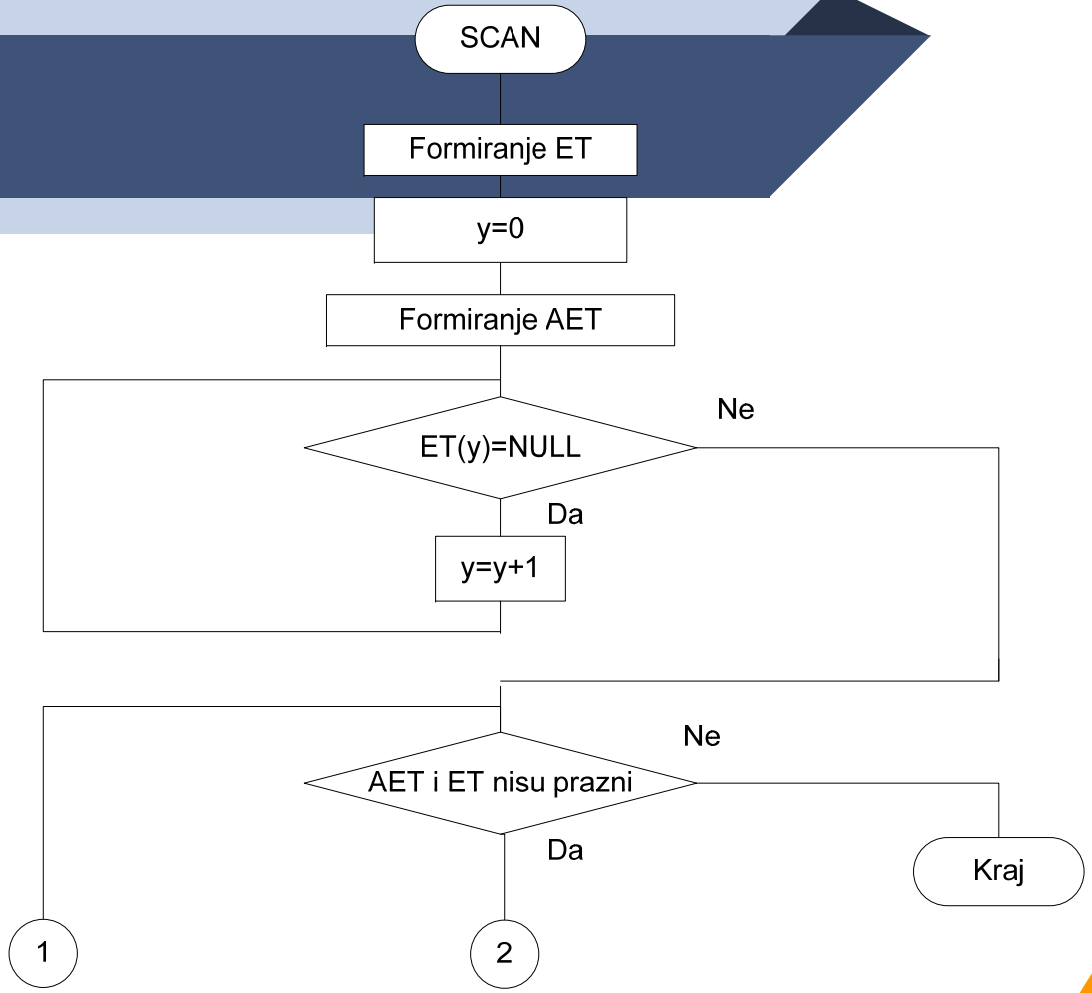
AET

Popuna poligona zadatog listom temena - Algoritam

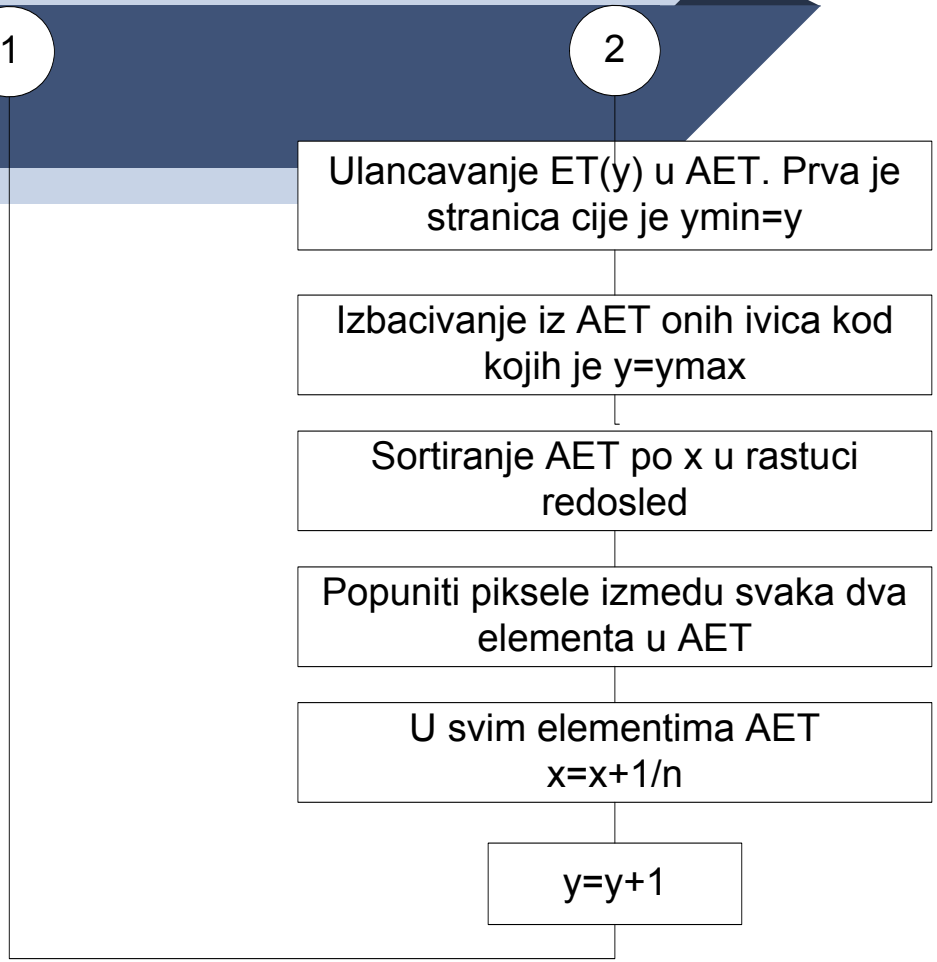




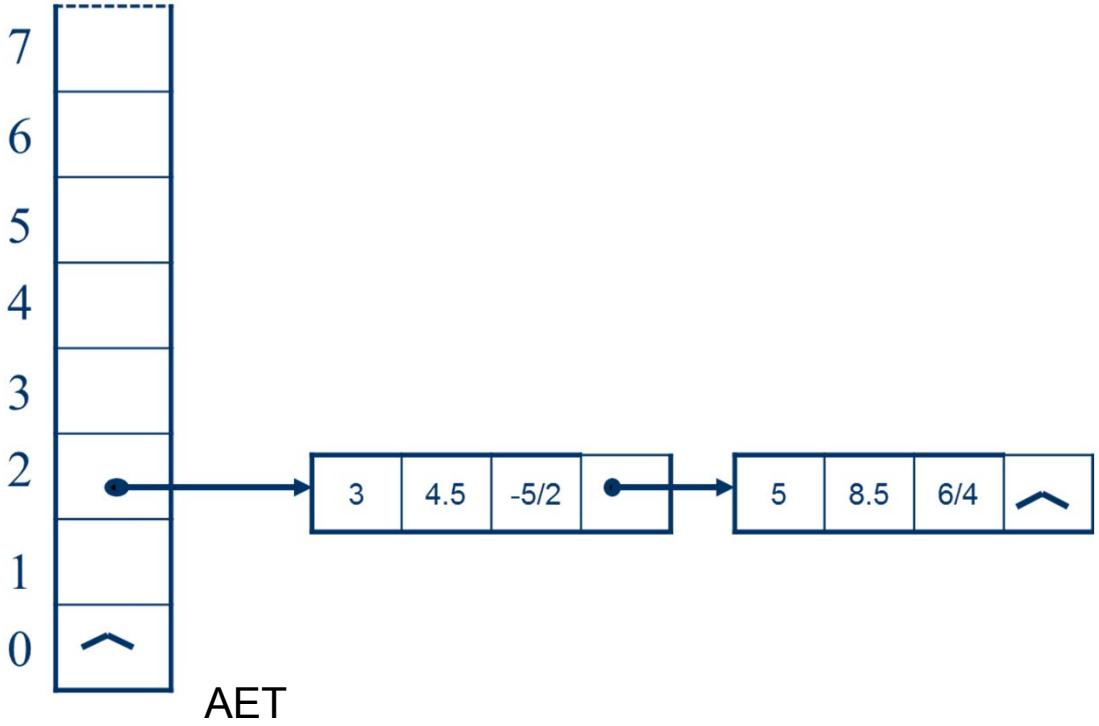
Algoritam

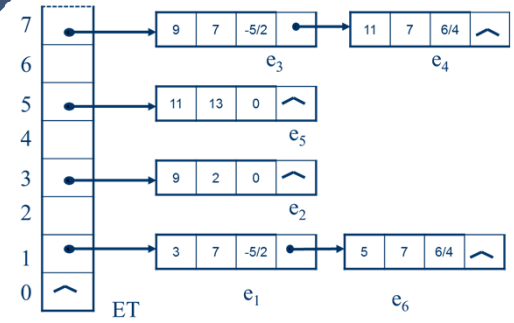

SCAN
Formiranje ET
y=0
Formiranje AET
ET(y)=NULL
Ne
Da
y=y+1
AET i ET nisu prazni
Ne
Da
Kraj
1
2

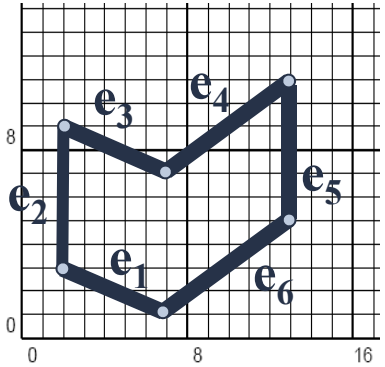
Algoritam


1
2
Ulancavanje ET(y) u AET. Prva je stranica cije je ymin=y
Izbacivanje iz AET onih ivica kod kojih je y=ymax
Sortiranje AET po x u rastuci redosled
Popuniti piksele izmedu svaka dva elementa u AET
U svim elementima AET x=x+1/n
y=y+1

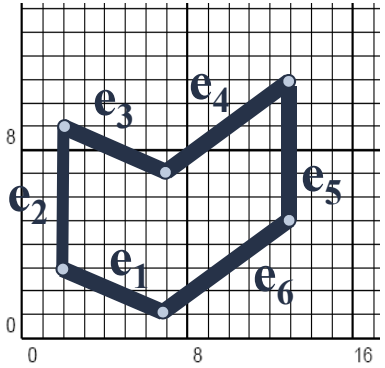
Popuna poligona zadatog listom temena - Algoritam

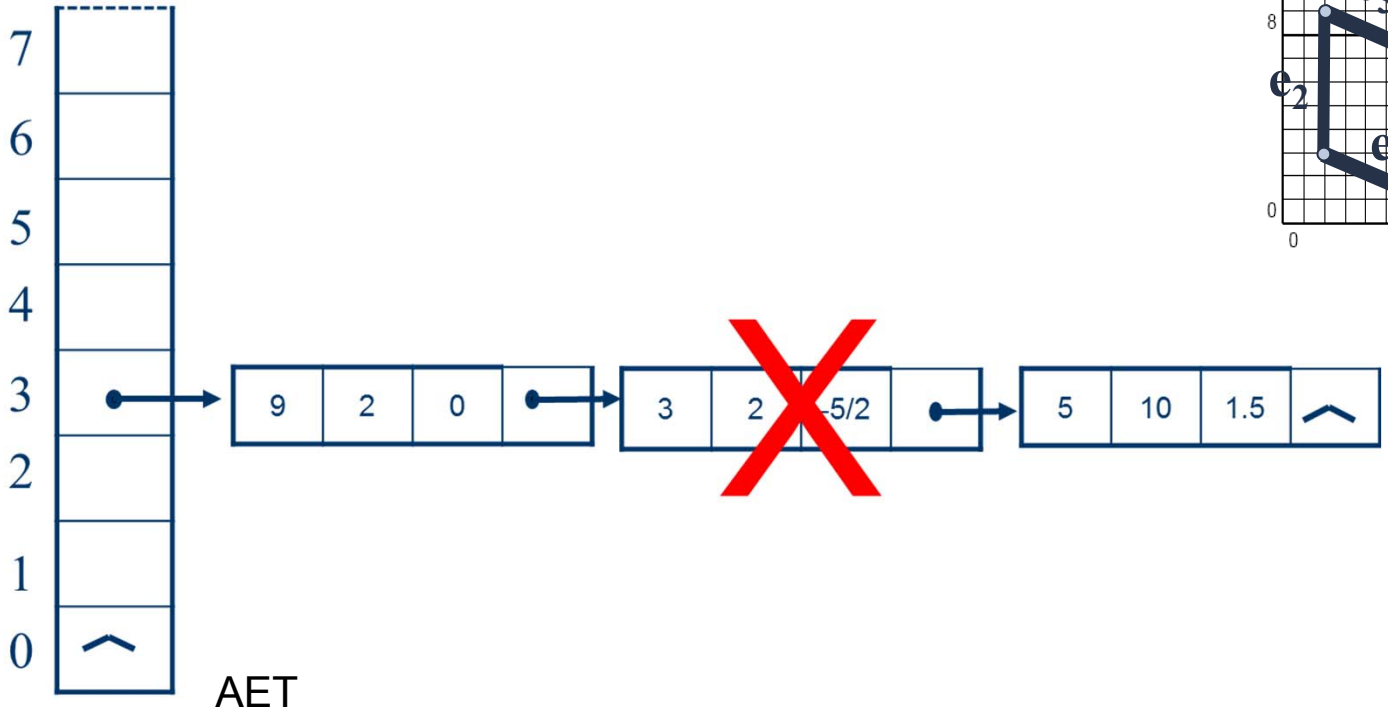




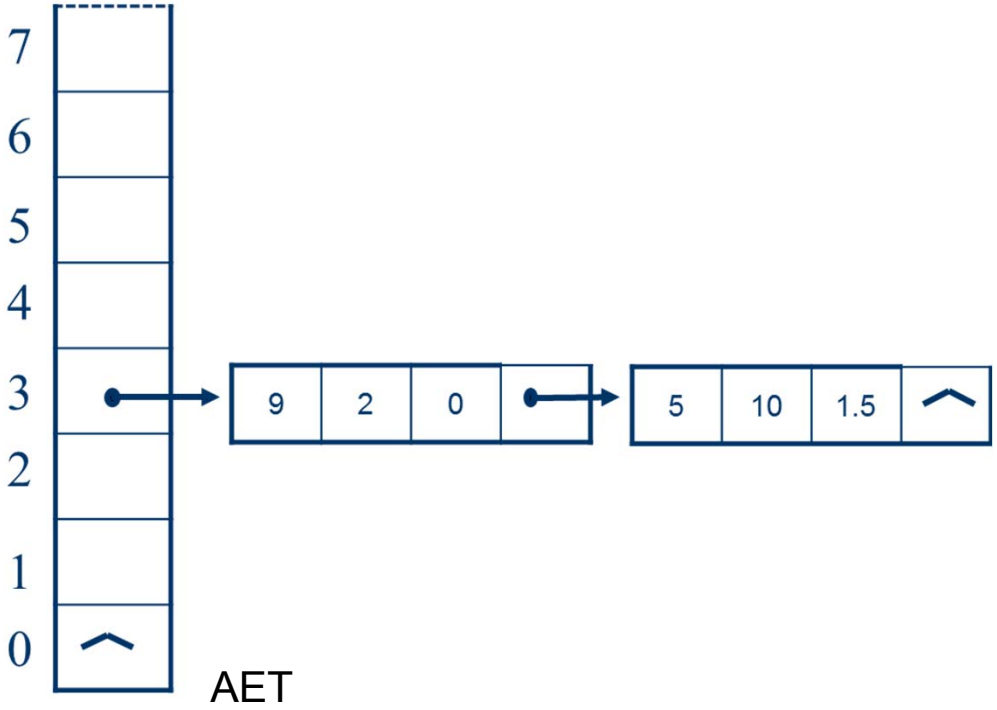


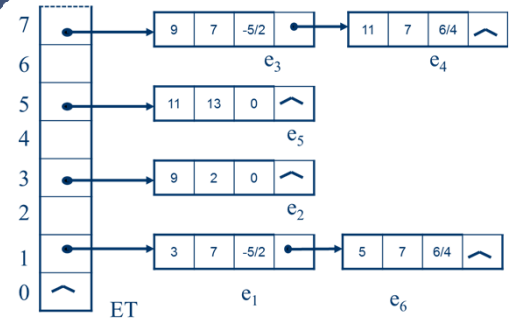
Popuna poligona zadatog listom temena - Algoritam

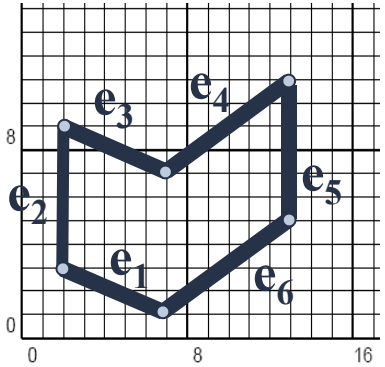




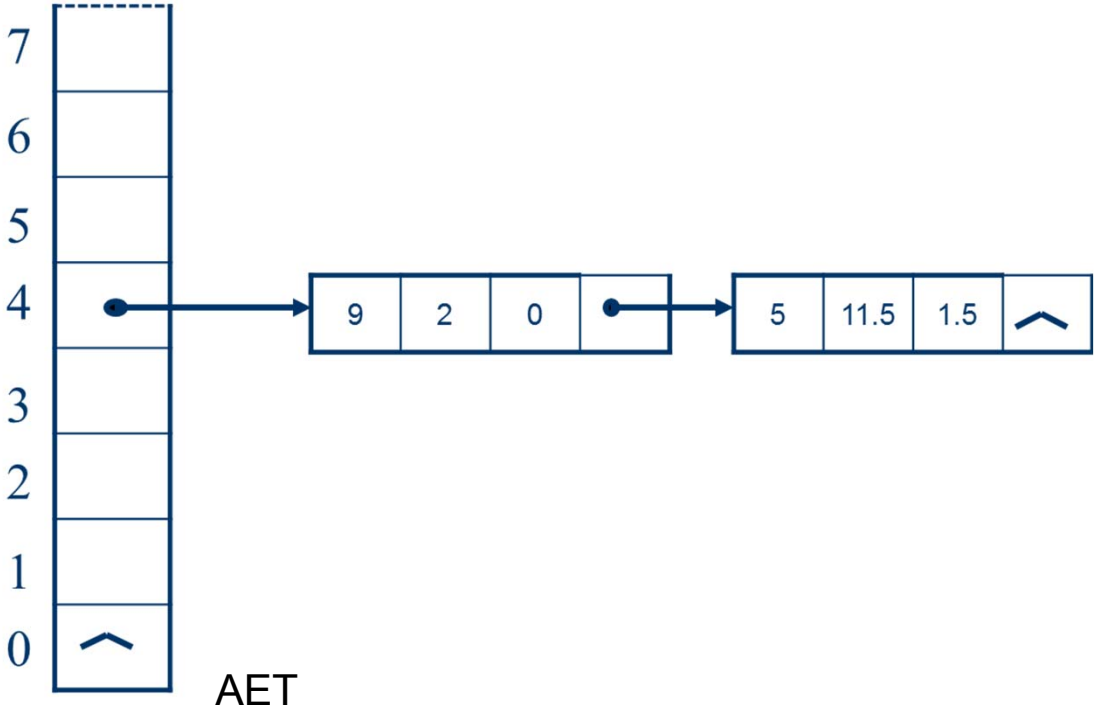
Popuna poligona zadatog listom temena - Algoritam

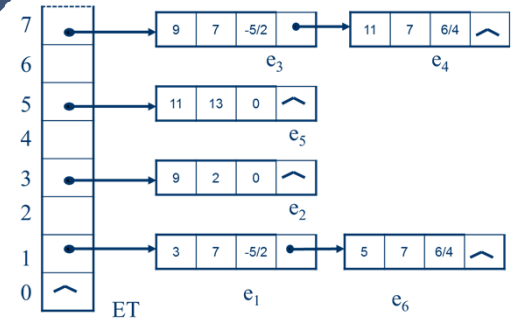


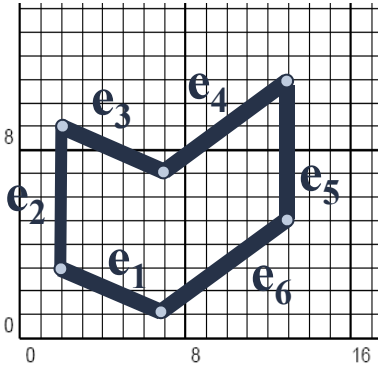




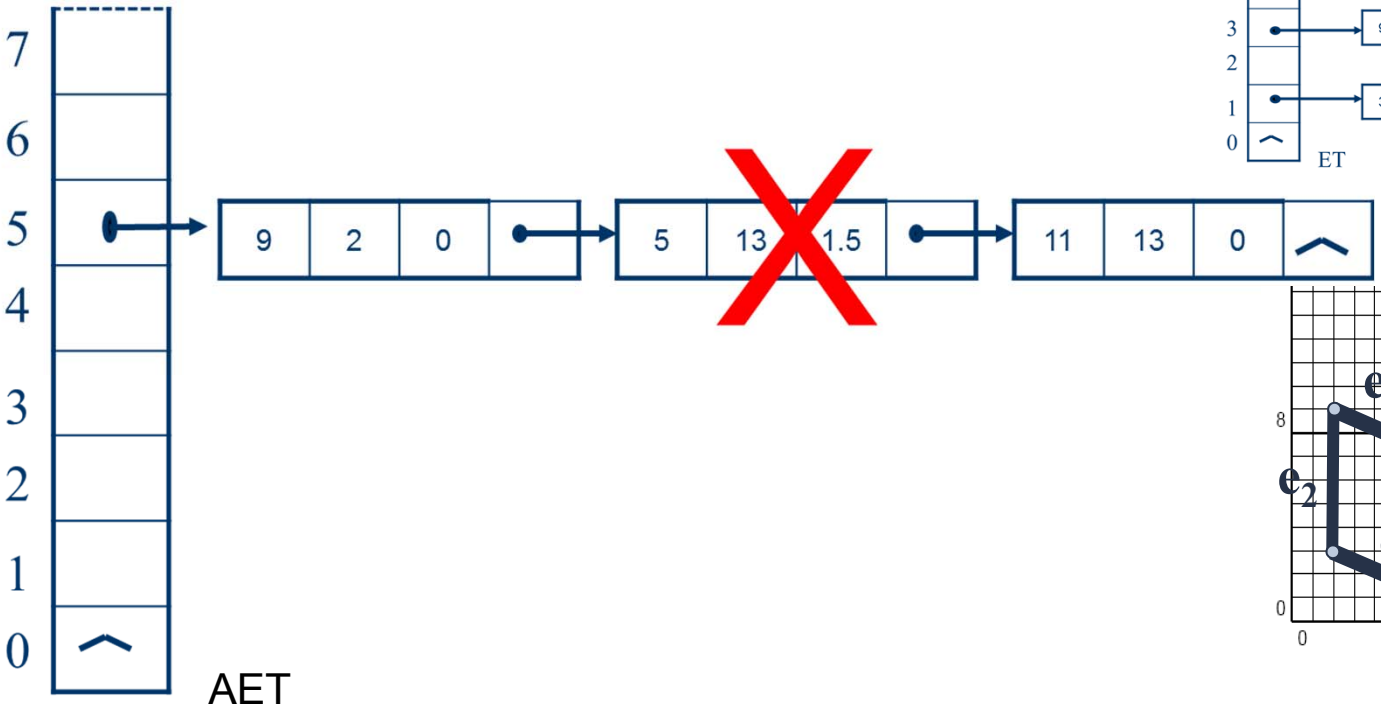
Popuna poligona zadatog listom temena - Algoritam

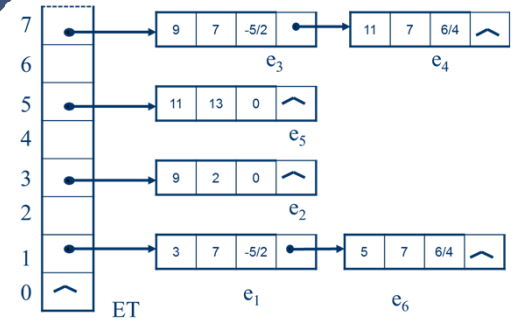


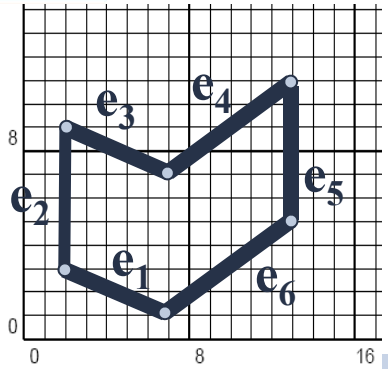




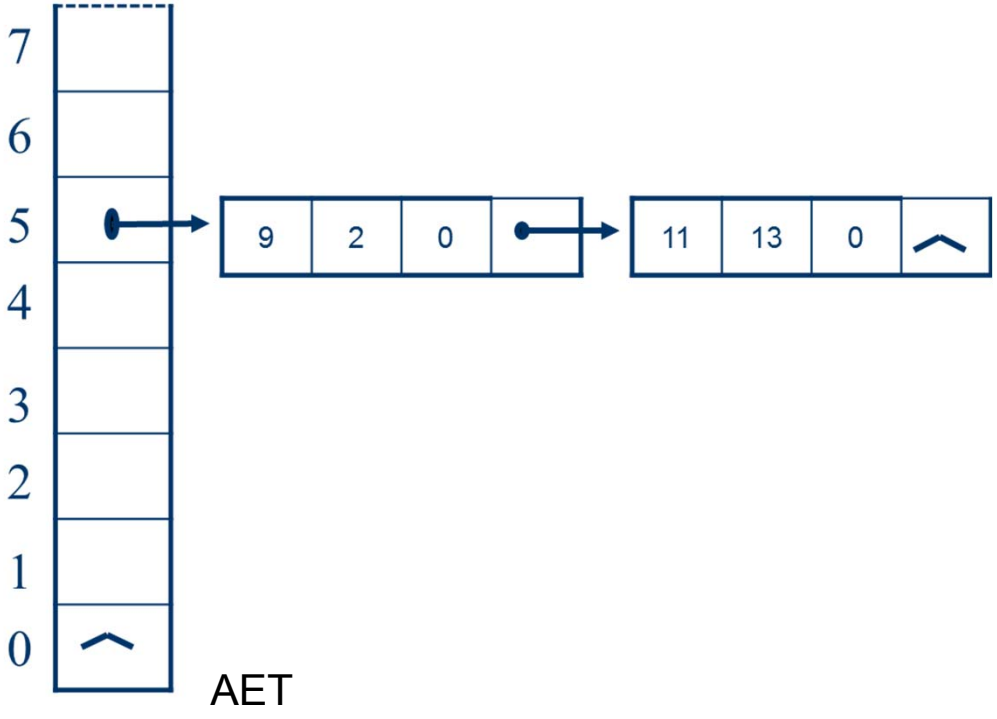
Popuna poligona zadatog listom temena - Algoritam

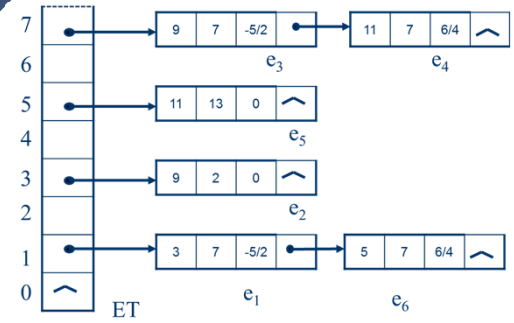


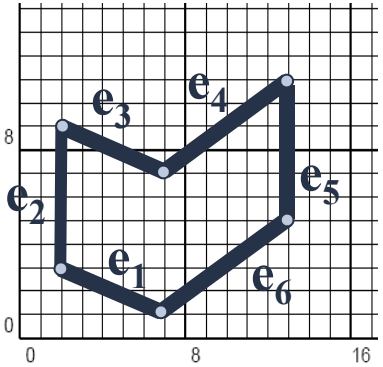




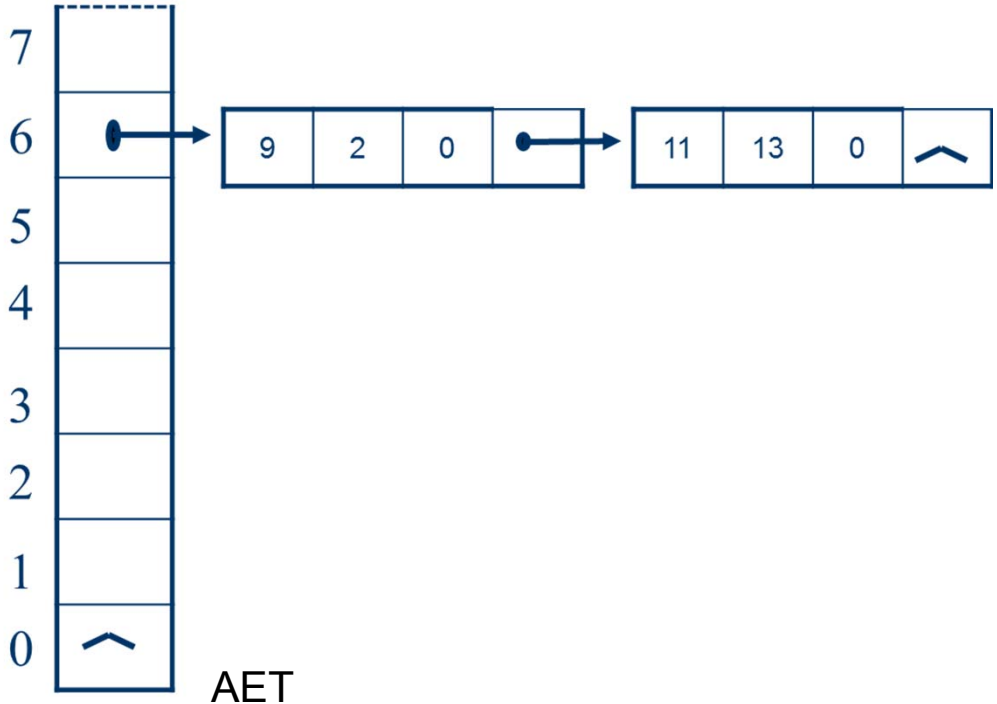
Popuna poligona zadatog listom temena - Algoritam

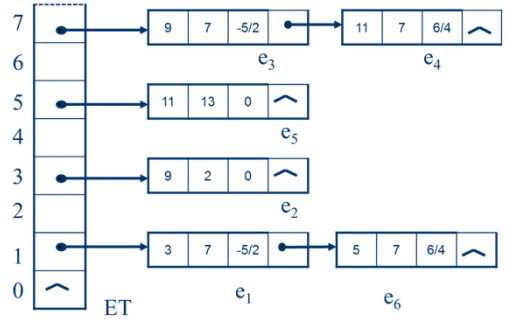


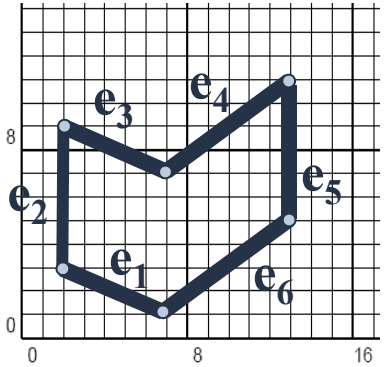




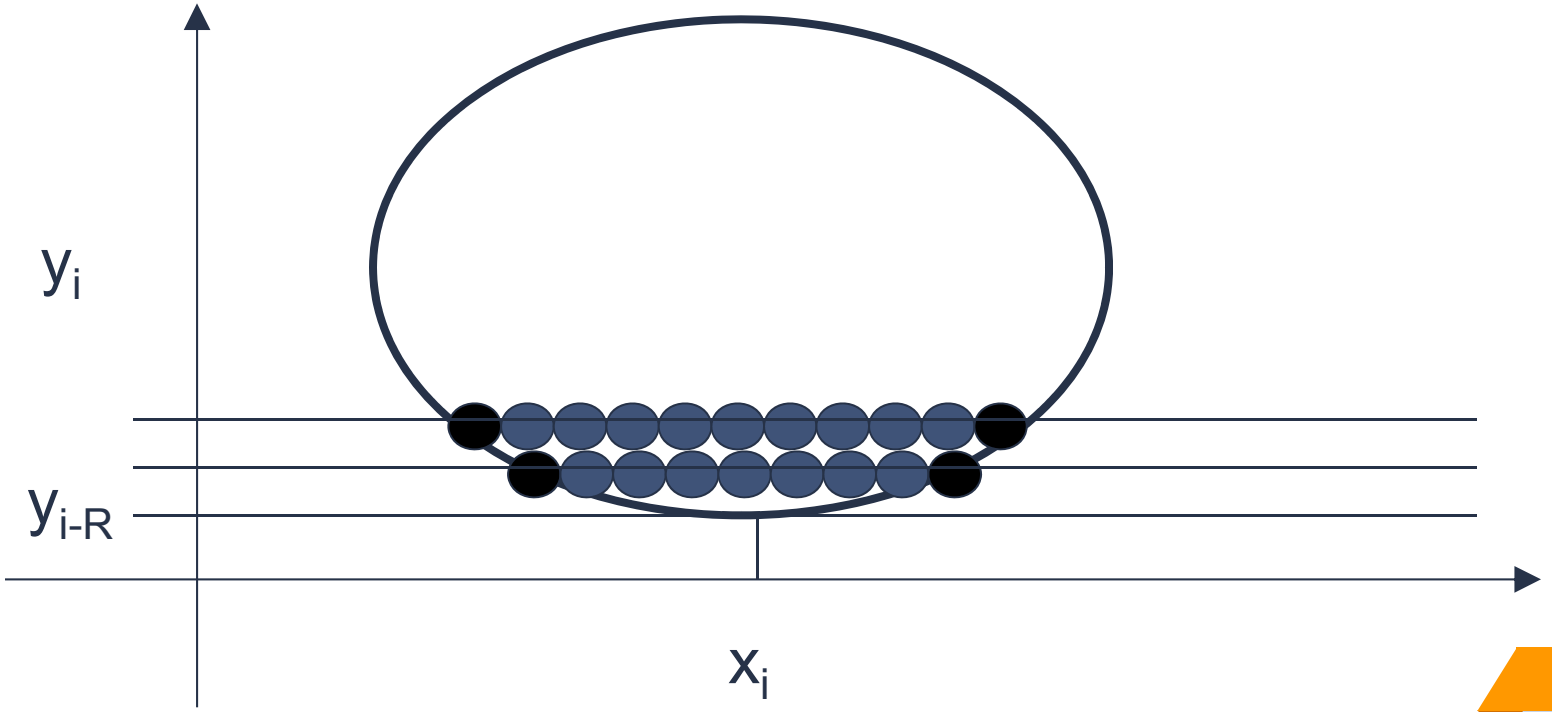
Popuna poligona zadatog listom temena - Algoritam







Popuna elipse i kruga

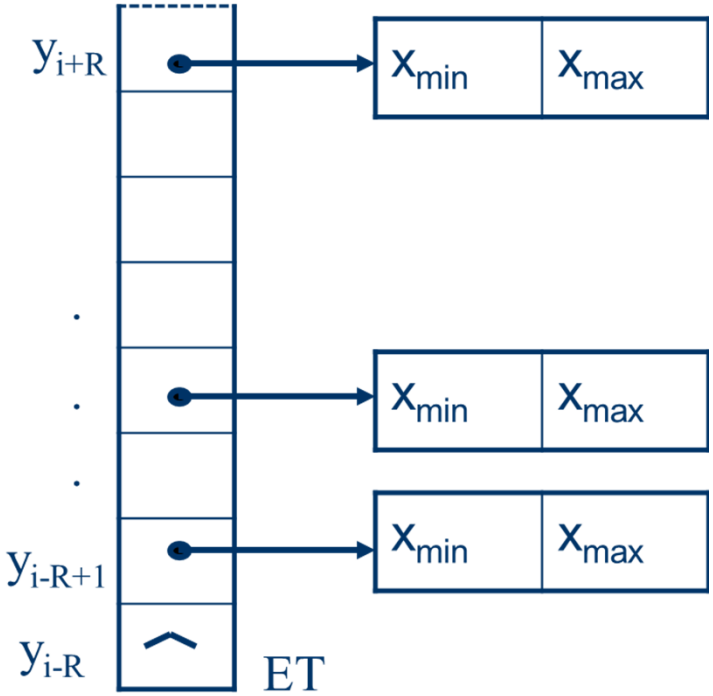


Popuna elipse i kruga

- Obzirom da znamo da svaka sken linija može 2 puta da preseče kružnicu ili elipsu, nema potrebe za AET.

- Obzirom da je reč o simetričnim figurama, možemo to da iskoristimo u algoritmu.

Popuna kruga i elipse – Algoritam



PITANJA