Računarska grafika
Popunjavanje regiona i poligona

Vesna Marinković

Rasterizacija duži – obnavljanje

1. Kako bi izgledala vektorska, a kako rasterska slika dva pravougaonika sa stranicama paralelnim koordinatnim osama? Koja od ove dve slike bi zauzimala manje memorije? Zašto?
2. Navesti primer gde se u praksi koristi vektorska grafika, a gde rasterska.
3. Da li crtamo po jedan piksel u svakoj koloni ili vrsti ako je koeficijent prave (a) $m = 1/2$ (b) $m = 3$ (c) $m = -1/2$?
4. Koliko piksela (računajući i krajnje tačke) treba da bude uključeno ukoliko na rasterskom uređaju treba linijom spojiti tačke:
(a) (1,2) i (53,30) (b) (1,2) i (22,50) (c) (1,2) i (31,32)

Rasterizacija duži – obnavljanje

5. Koji su nedostaci algoritma grube sile za crtanje duži? Na koji način se oni mogu eliminisati?
6. Koja je ključna ideja midpoint algoritma za crtanje duži?
7. Koju vrednost u algoritmu rasterizacije duži zovemo promenljiva odlučivanja i zašto?
8. Zašto prvu vrednost promenljive odlučivanja množimo sa 2 u realizaciji midpoint algoritma?
9. Ako se midpoint algoritmom iscrtava duž između tačaka (2,4) i (12,9), čemu je jednaka početna vrednost promenljive odlučivanja?
10. Koja je vremenska složenost midpoint algoritma za crtanje duži?

Rasterizacija kruga – obnavljanje

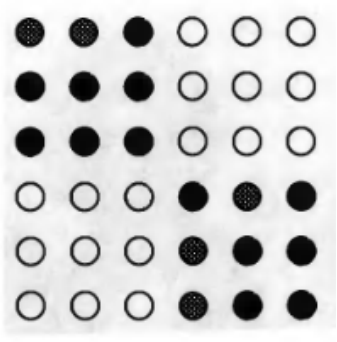
11. Kako funkcioniše algoritam grube sile za iscrtavanje kruga? Koji su njegovi nedostaci?
12. Zašto se u midpoint algoritmu za crtanje kruga razmatra samo osmina kruga?
13. Koju vrednost u algoritmu rasterizacije kruga zovemo promenljiva odlučivanja?
14. Koja je razlika u inkrementalnom računanju vrednosti d u odnosu na algoritam za crtanje duži?
15. Čemu je jednaka prva vrednost promenljive d? Kako se borimo sa tim što početna vrednost za d nije celobrojna?

Rasterizacija kruga – obnavljanje

16. U unapređenoj verziji midpoint algoritma za crtanje kruga, zašto je neophodno da svaki put ažuriramo obe promenljive Δ_E i Δ_{SE}?
17. Koje veličine nazivamo razlikama drugog reda?
18. Koja je vremenska složenost midpoint algoritma za crtanje kruga?
19. Kako izmeniti midpoint algoritam ako je potrebno nacrtati krug sa centrom u tački (c_1, c_2)?

Povezanost regiona

- Nakon iscrtavanja primitiva, nekad ih je potrebno i obojiti
- Razmatramo prvo opšti slučaj, kada je dat *proizvoljni* region
- Region je kolekcija piksela; mora biti povezan
  - Region je 4-povezan ako se svaka dva piksela tog regiona mogu povezati nizom piksela iz tog regiona do kojih se stiže kretanjem nagore, nadole, ulevo i udesno
  - Region je 8-povezan ako se svaka dva piksela tog regiona mogu povezati nizom piksela iz tog regiona do kojih se stiže kretanjem nagore, nadole, ulevo, udesno, gore-levo, gore-desno, dole-levo, dole-desno

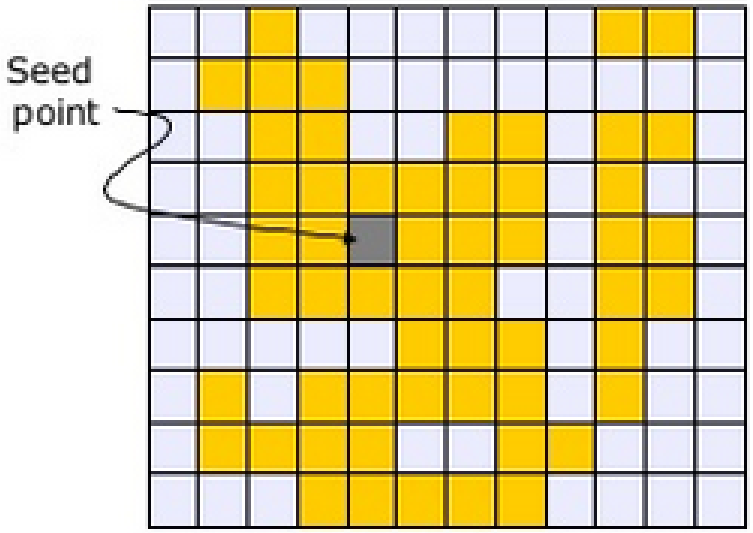


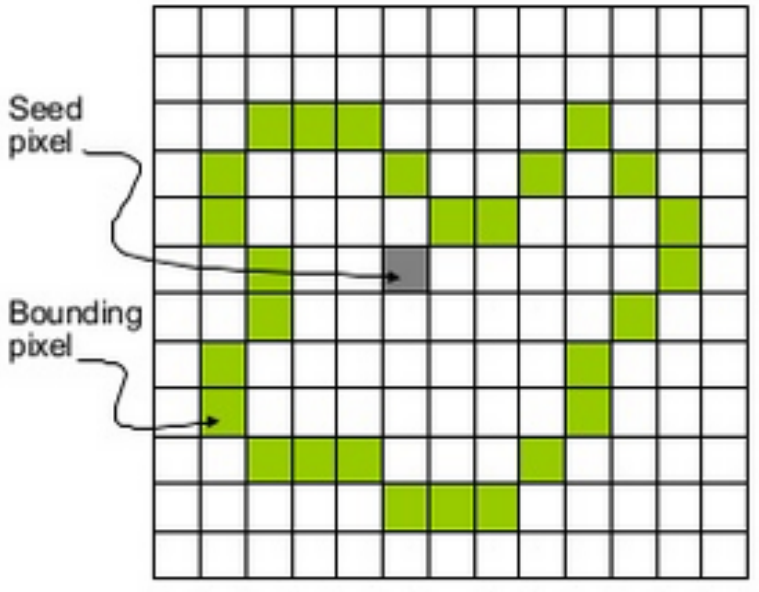
Vrste algoritama za popunjavanje regiona

- seed algoritmi
  - kreću od neke početne vrednosti (seed-a) unutar regiona
  - ako piksel nije već postavljen, postavljamo ga i posećujemo susede rekurzivno
  - nije neophodno imati matematički opis regiona – može biti slobodnoručni crtež
  - jednostavni algoritmi, ali su memorijski neefikasni i spori
- algoritmi koji koriste scan linije
  - koriste osobine koherentnosti scan linija, ivica i opsega scan linija
- kombinovani algoritmi
  - imaju prednosti obe vrste algoritama
  - umesto rekurzivnih provera za susede, traži se najduži niz piksela u scan liniji koji se može popuniti

Definisanje regiona

- Za dati početni piksel P region se može definisati na dva načina:
  - region definisan unutrašnjošću je *najveći povezani* region piksela čija je boja ista kao boja piksela P
  - region definisan granicom je *najveći povezani* region piksela čija boja nije jednaka nekoj graničnoj vrednosti boje





Algoritmi za popunjavanje regiona definisanog unutrašnjošću

- Flood-fill algoritam
- Kreće se od početnog piksela u unutrašnjosti (seed-a), proverava se da li je boja piksela stara, ako jeste piksel se boji novom bojom i pomeramo se iz ovog piksela u 4 (ili 8) smerova
- Granica oblasti se može sastojati od različitih boja
- Ovakva implementacija je praktično neupotrebljiva (zbog potencijalno velike dubine rekurzije)

```
procedure FloodFill4(x, y, old_color, new_color : integer);
begin
 if (getPixel(x,y) == old_color) then
   begin
     setPixel(x,y,new_color);
     FloodFill4(x-1,y,old_color,new_color);
     FloodFill4(x+1,y,old_color,new_color);
     FloodFill4(x,y-1,old_color,new_color);
     FloodFill4(x,y+1,old_color,new_color);
   end
```

Algoritmi za popunjavanje regiona definisanog granicom

- Boundary-fill algoritam
- Kreće se od početnog piksela u unutrašnjosti, proverava se da li je boja piksela različita od granične vrednosti boje i od nove vrednosti boje, ako jeste piksel se boji novom bojom i pomeramo se iz ovog piksela u 4 (ili 8) smerova
- Rekurzivna priroda algoritma može prouzrokovati prerani prekid rada ako se nova boja pojavi u regionu koji je definisan granicom

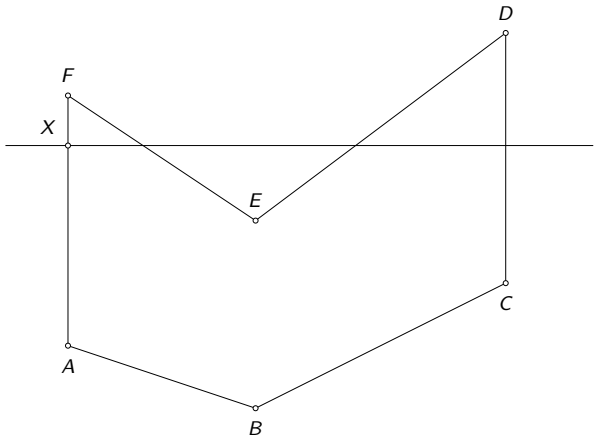
```
procedure BoundaryFill4(x, y, boundary_color, new_color : integer);
begin
   color c := getPixel(x,y);
   if (c != boundary_color && c != new_color) then
     begin
       setPixel(x,y,new_color);
       BoundaryFill4(x-1,y,boundary_color,new_color);
       BoundaryFill4(x+1,y,boundary_color,new_color);
       BoundaryFill4(x,y-1,boundary_color,new_color);
       BoundaryFill4(x,y+1,boundary_color,new_color);
     end
end.
```

Naivni algoritmi za popunjavanje poligona

- Zadatak je obojiti sve piksele koji se nalaze u unutrašnjosti datog poligona (istom bojom)
- Za svaku tačku na slici možemo da utvrdimo da li pripada unutrašnjosti poligona
- Ideja naivnog algoritma:
  - Odrediti *najmanji* pravougaonik sa stranicama paralelnim koordinatnim osama koji sadrži dati poligon
  - Za svaku tačku tog pravougaonika proverava se da li pripada unutrašnjosti datog poligona
  - Provera pripadnosti vrši se na osnovu parnosti broja preseka proizvoljne poluprave iz te tačke sa stranicama poligona

Scan popunjavanje poligona

- Scan linija je horizontalna linija na rasterskom sistemu
- Ideja algoritma: za svaku scan liniju odrediti piksele koji se nalaze u unutrašnjosti poligona i obojiti ih



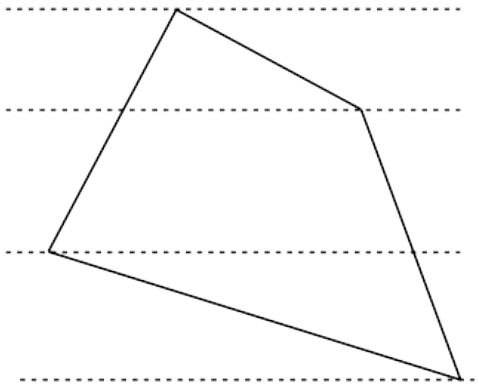
- Algoritam korektno radi i za konkavne poligone, čak i ako poligon ima samopresecanja ili rupe (kako se za njih definiše unutrašnjost?)

Scan popunjavanje poligona

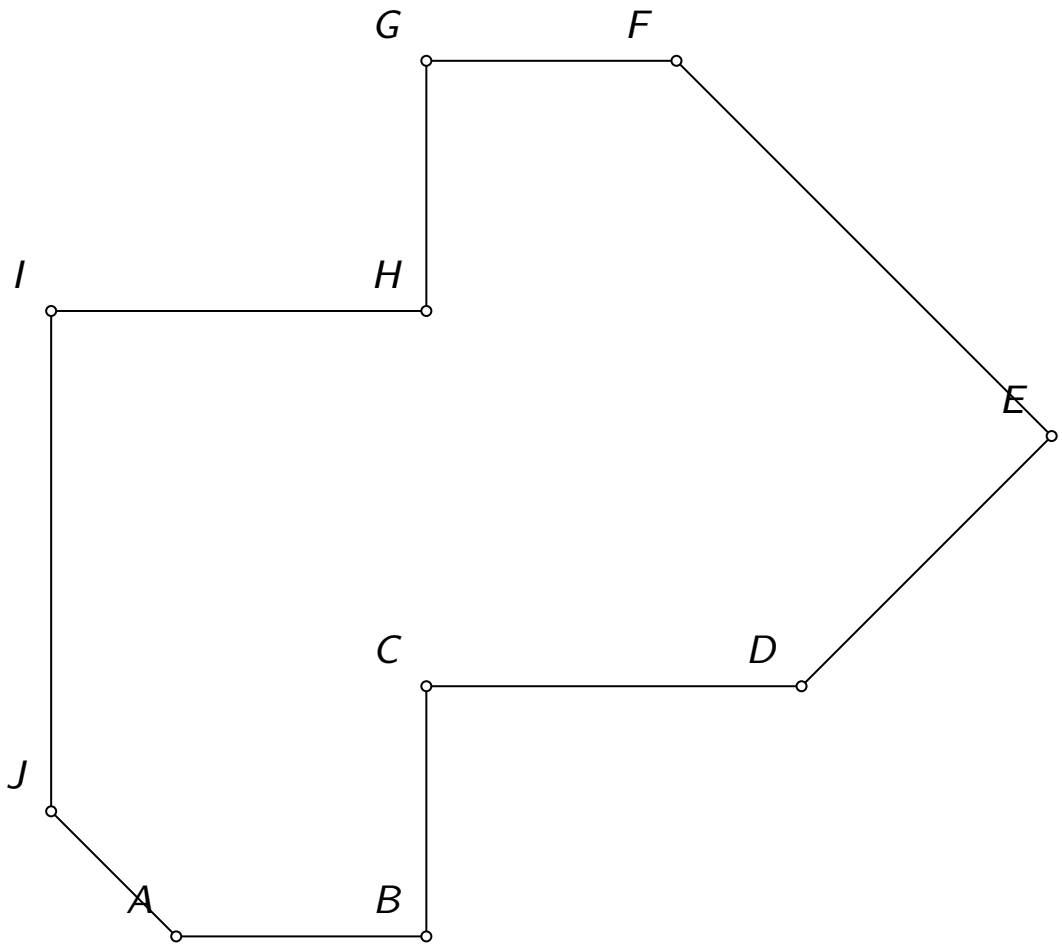
- Na svakoj scan liniji određujemo tačke koje pripadaju stranicama poligona
- Za svaki presek scan linije sa stranicom poligona inkrementiramo vrednost namenskog brojača (čija je inicijalna vrednost 0)
- Važna je parnost brojača:
  - kod tačaka sa neparnom vrednošću treba započeti bojenje
  - kod tačaka sa parnom vrednošću treba prekinuti bojenje

Scan popunjavanje poligona – specijalni slučajevi

- Šta ako scan linija seče stranicu u temenu?
  - za stranicu poligona računamo tačku preseka sa najmanjom y koordinatom, a ne računamo tačku preseka sa najvećom y koordinatom
  - parnost se menja osim ako je u pitanju lokalni minimum/maksimum
- Šta ako poligon ima horizontalnu stranicu?
  - kod horizontalnih stranica ne brojimo nijednu presečnu tačku
  - efekat ovog pravila je da se donje horizontalne stranice boje, a gornje ne; takođe ne boje se i desne tačke stranica

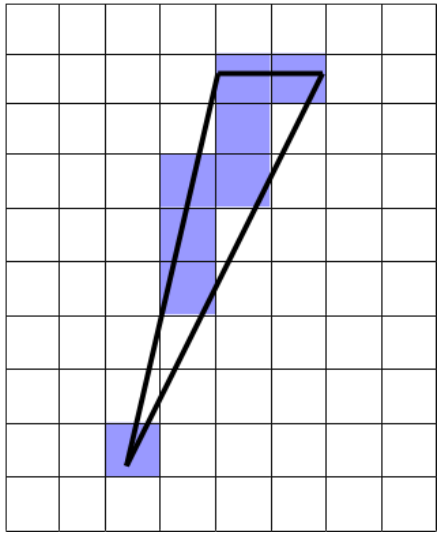


Scan popunjavanje poligona – primer



Scan popunjavanje poligona – sliver

- Sliver je uska poligonalna oblast koja sadrži 0 ili 1 piksel za neke scan linije
- Ovo je posledica toga da se boje samo pikseli koji su u unutrašnjosti ili na levim ili donjim stranicama poligona
- Antialiasing tehnika se može koristiti za rešavanje ovog problema



Određivanje preseka scan linije sa stranicama poligona

- Najčešće scan linija seče samo nekoliko stranica poligona, dakle nije isplativo da određujemo presek scan linije sa svim stranicama poligona
- Najčešće stranice koje seče n-ta scan linija seče i $(n+1)$-a scan linija
- Koristi se inkrementalni pristup da bi se izračunali preseci jedne scan linije na osnovu prethodne scan linije, bez toga da analitički računamo presek scan linije sa svakom stranicom poligona
- Ideja nalik midpoint algoritmu, samo je potrebno uvek birati tačke u unutrašnjosti poligona

Određivanje preseka scan linije sa stranicama poligona – inkrementalni pristup

- Označimo sa (x_i, y_i) koordinate preseka i-te scan linije sa stranicom poligona
- Prilikom prelaska sa tekuće na narednu scan liniju, inkrementira se y koordinata: $y_{i+1} = y_i + 1$
- Ako je jednačina prave koja sadrži stranicu $y = mx + B$ onda važi: $y_i = mx_i + B$ i $y_{i+1} = mx_{i+1} + B$
- Odavde sledi:

$$x_{i+1} = \frac{1}{m} \cdot (y_{i+1} - B) = \frac{1}{m}(y_i + 1 - B) = \frac{1}{m} \cdot (y_i - B) + \frac{1}{m} = x_i + \frac{1}{m}$$

- Potrebno je izvršiti zaokruživanje za računanje najbližeg piksela u unutrašnjosti
- Želimo da modifikujemo navedenu vezu tako da koristi samo celobrojnu aritmetiku

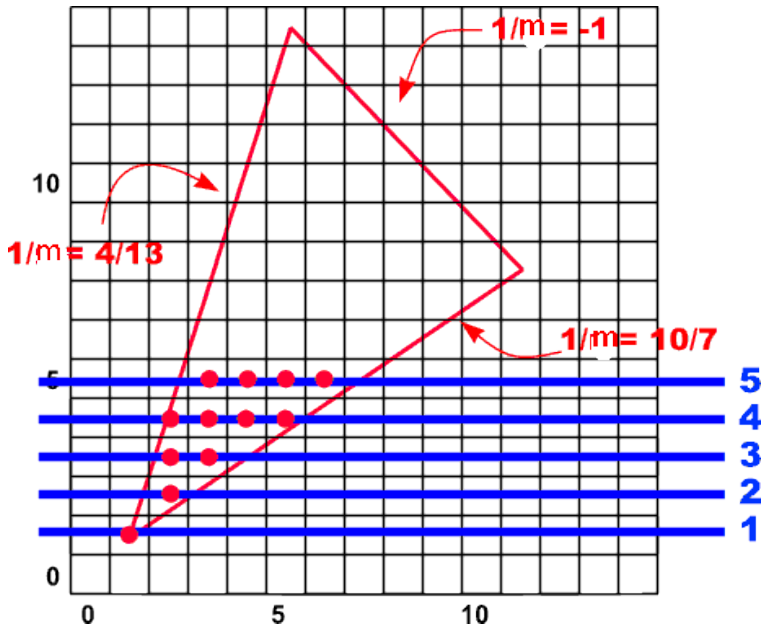
Određivanje preseka scan linije sa levom stranicama poligona

- Razmatramo slučaj kada je $m > 1$ i kada je u pitanju leva stranica (parnost pre ovog preseka je 0)
- Važi:

$$x_{i+1} = x_i + \frac{1}{m} = x_i + \frac{x_{max} - x_{min}}{y_{max} - y_{min}}$$

- Razmatramo posebno celi i razlomljeni deo vrednosti $\frac{x_{max} - x_{min}}{y_{max} - y_{min}}$
  - kada razlomljeni deo pređe 1 inkrementiramo celi deo a dekrementiramo razlomljeni deo
  - kada je razlomljeni deo jednak 0 možemo da obojimo piksel (x, y)
  - kada je razlomljeni deo različit od 0 treba zaokružiti tako da x pripada unutrašnjosti poligona

Izračunavanje preseka sa levom stranicom – primer



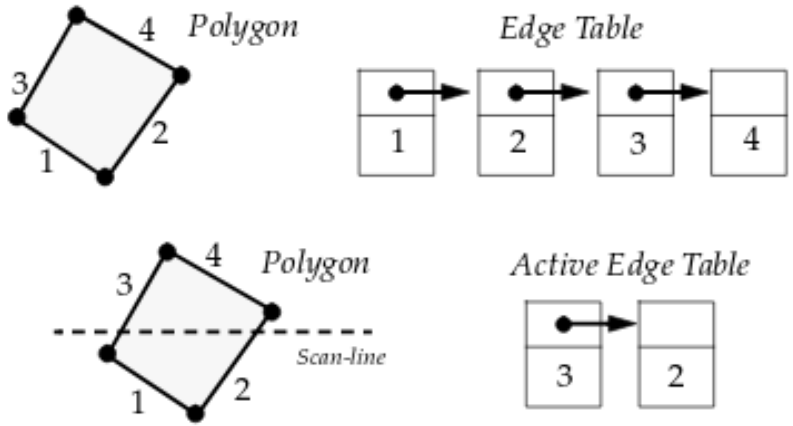
- $x_{min} = 1, x_{max} = 5, y_{min} = 1, y_{max} = 14$, $(x_{max} - x_{min})/(y_{max} - y_{min}) = 4/13$
- $x_1 = 1$ (razlomljeni deo je jednak 0),
- $x_2 = x_3 = x_4 = 2$ (razlomljeni deo je > 0, zaokružujemo tako da pripada unutrašnjosti poligona)
- $x_5 = 3$ (razlomljeni deo je prešao 1, pa smo uvećali x za 1) , . . .

Algoritmi za određivanje preseka sa levom stranicom

```
procedure LeftEdgeScan(xmin, ymin, xmax, ymax : integer);
begin
   x := xmin;
   y := ymin;
   brojilac := xmax - xmin;
   imenilac := ymax - ymin;
   inkrement := imenilac; { vazno za zaokruzivanje na unutrasnji piksel }
   for y := ymin to ymax do
      begin
         SetPixel(x,y);
         inkrement := inkrement + brojilac;
         if inkrement > imenilac then
            begin  { prekoracenje, zaokruzi na sledeci }
                   { piksel i smanji inkrement }
               x := x + 1;
               inkrement := inkrement - imenilac;
            end;
      end
end.
```

Scan popunjavanje poligona – praktična implementacija

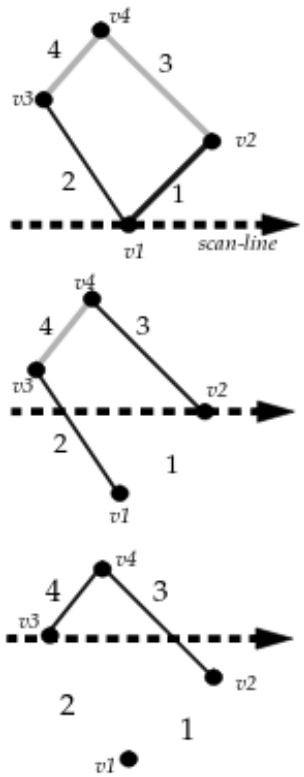
- Namenska struktura podataka čuva podatke o aktivnim stranicama A – stranicama koje se seku sa tekućom scan linijom
- Prva scan linija je $y = y_{min}$, gde je y_{min} najmanja vrednost y koordinate svih temena poligona
- Skup S čuva podatke o još uvek neobrađenim stranicama poligona
- Inicijalno je $A = \emptyset$, a S sadrži sve stranice poligona (sortirane po vrednosti najmanje y koordinate temena)



Scan popunjavanje poligona – algoritam

1. Prebaci iz skupa S u skup A one stranice čija je vrednost najmanje y koordinate jednaka y vrednosti scan linije
2. Sortiraj stranice u skupu A rastuće prema x koordinati tačke preseka sa scan linijom
3. Oboji sve piksele na scan liniji između neparnih i parnih preseka
4. Izbaci iz skupa A one stranice čija je vrednost maksimalne y koordinate jednaka y vrednosti scan linije
5. Uvećaj y vrednost scan linije za 1
6. Ažuriraj tačke preseka (kao u algoritmu `LeftEdgeScan`)

Scan popunjavanje poligona – algoritam



1. Scan linija prolazi kroz prvo teme v_1, $E = \{3, 4\}$, $A = \{2, 1\}$
2. Scan linija dostiže drugo teme v_2, $E = \{4\}$, $A = \{2, 3\}$
3. Scan linija prolazi kroz treće teme v_3, $E = \emptyset$, $A = \{4, 3\}$