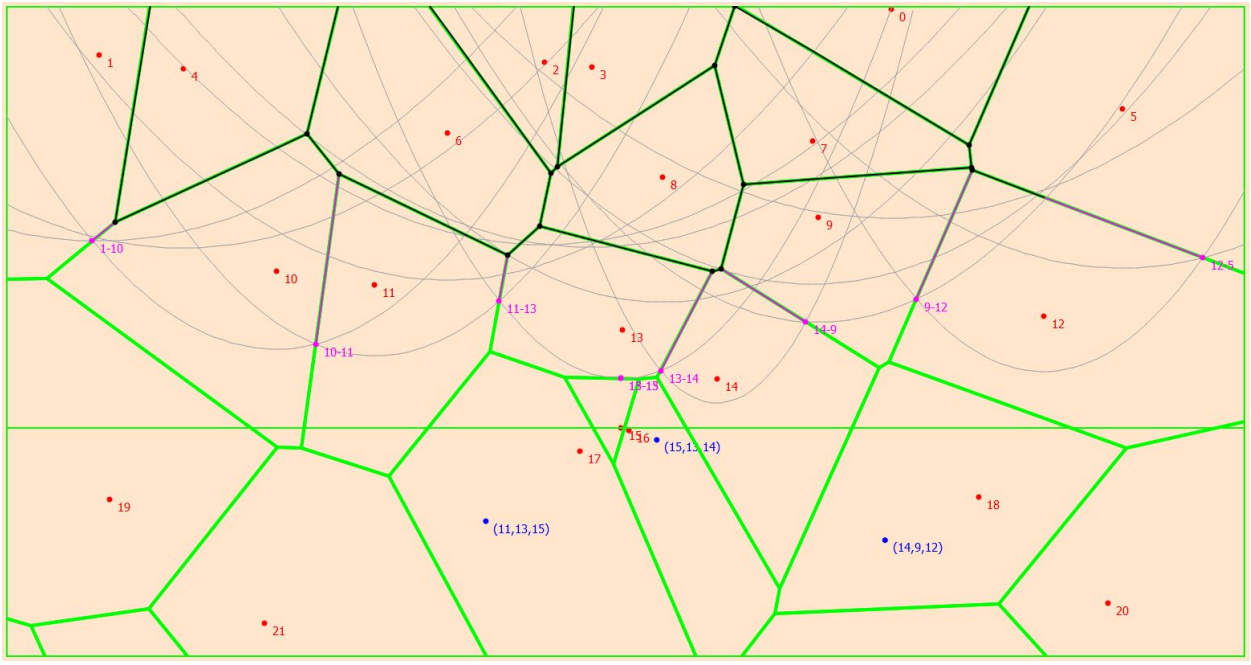
Geometrijski algoritmi @ MATF

Konstrukcija Voronoj dijagrama

Filip Novović



Opis problema

Voronoj dijagram Vor(P) skupa tačaka P={$P_1, P_2, \ldots, P_n$ } je razlaganje ravni na oblasti takve da tačka X pripada oblasti tačke P_i ako i samo ako važi:

$$d(X, P_i) < d(X, P_j)\text{, za svako } i \neq j$$

Oblast tačke P_i naziva se Voronoj ćelija tačke P_i i označava se sa $v(P_i)$

Ulaz: *skup od n tačaka u ravni*

Izlaz: *podela ravni na regione najbližih tačaka zadatim tačkama*

Naivno rešenje problema

Najjednostavnije rešenje za konstrukciju Voronoj dijagrama zasnovano je na gruboj sili. Za određivanje oblasti $v(P_i)$ računa se presek svih poluravni $h(P_i, P_j)$ za svako j takvo da je $j \neq i$. Navedeni pristup je jednostavan, ali nedovoljno efikasan jer uključuje mnogo nepotrebnog rada: sedišta koja su daleko od P_i nemaju uticaj na ćeliju $v(P_i)$.

Opis implementacije naivnog algoritma

Za svako teme region tražimo tako što počnemo od poligona koji je dimenzija površine na kojoj se pokreće algoritam. Teme za koje tražimo tražimo region, nazvaćemo aktivnim temenom. Poligon "seckamo" simetralama koje čini aktivno teme sa ostalim temenima. Kada završimo proveru simetrala sa svim ostalim temenima i ažuriramo poligon koji sadrži aktivno teme, rezultat (taj poligon) proglašavamo regionom aktivnog temena.

Pseudo-kod implementacije naivnog algoritma

```
temeRegion = {} //kljuc je teme, vrednost je region (poligon) tog temena
foreach A in temena
    trPoligon = pravougaonik ekrana
    poligonA = null
    poligonB = null
    foreach B in temena
        if A != B
            p = simetrala duzi A B
            poligonA = poligon koji nastaje podelom trPoligon simetralom p i sadrzi A
            poligonB = poligon koji nastaje podelom trPoligon simetralom p i sadrzi B
            trPoligon = poligonA

    temeRegion[A] = trPoligon
```

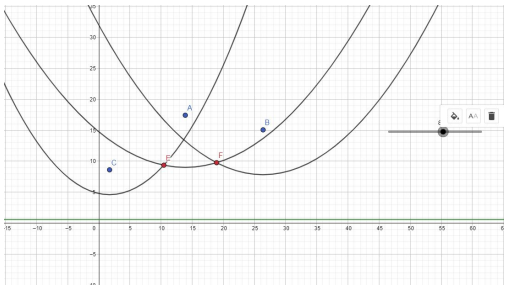
Forčenov algoritam

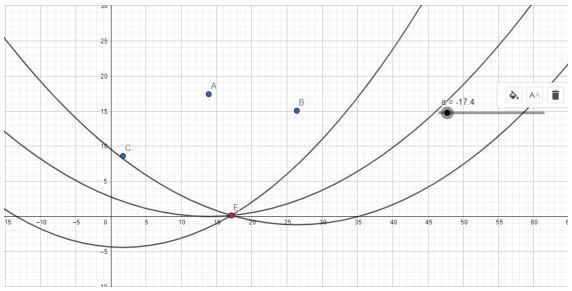
Forčenov algoritam zasnovan je na metodi "brišuće prave". Horizontalna prava l pomera se odozgo nadole i usput konstruiše Voronoj dijagram. Ideja se zasniva na tome da kako l prođe neko teme, mi tom temenu pridružimo parabolu koja se definiše fokusom (temenom) i direktrisom (linijom l). Kada imamo ovako konstruisane parabole za temena koja su iznad prave l grane Voronoj dijagrama tražimo preko preseka susednih parabola. Bitne izmene u stanju pri izvršavanju algoritma se dešavaju samo u određenim tačama pa nije potrebno da se prava l kreće po svim y koordinatama površine na kojoj su temena. Te

tačke kategorišemo kao događaj sedišta i događaj kruga. *Događaji sedišta* su sva temena koja predstavljaju ulaz u algoritam. *Događaji kruga* su tačke u kojima iniciramo stvaranje novog temena Voronoj dijagrama.

Nailazak na događaj sedišta

Nastaje nova parabola za teme koje obilazimo. Kako bismo je smestili u strukturu podataka koja čuva *liniju fronta* (dosadašnji skup preseka susednih parabola), moramo da nađemo u već postojećem frontu luk koji je direktno iznad novog temena. Kada smo našli gornji luk, delimo ga na tri manja luka (levi deo je deo starog luka, srednji deo pripada novom luku, desni deo je takođe deo starog luka). Nakon ubacivanja novog luka (i preseka sa starim lukom), proveravamo da li se nova parabola svojim razvijanjem (pomeranjem linije l naniže) približava nekoj drugoj čiji luk pripada frontu a koje su na frontu odvojene jednim lukom. Ako se približava, proveravamo u kojoj tački će se susresti.





Primer približavanja parabola kojima su fokus C i B i zaklapanje luka koji pripada paraboli A.

Tačku koja je po y koordinati manja za dužinu poluprečnika kruga koji čine tačke A, B i C od tačke u kojoj će se susresti parabole, a po x-u ista proglašavamo događajem kruga i ubacujemo je u red događaja koje je potrebno obraditi. Potrebno je još i izbaciti iz reda događaja one događaje kruga u kojima je učestvovao luk na koji smo ubacili novu parabolu.

Nailazak na događaj kruga

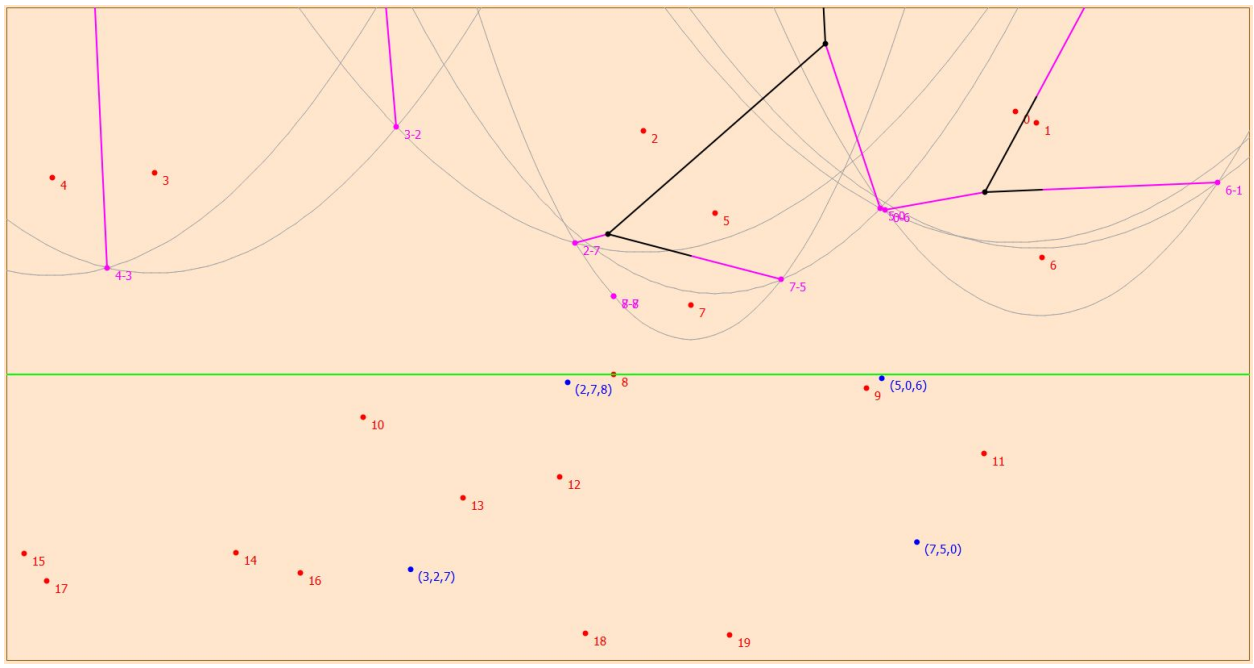
Kada obrađujemo događaj kruga koji smo izračunali preko tačke K gde se spajaju lukovi tri parabole (npr. A, B i C), tu tačku K ubacujemo u strukturu koja nam čuva konstruisani Voronoj dijagram kao teme zajedno sa granama koje su se spojile u tom temenu. Dodatno proveravamo da li postoje događaji kruga za lukove koji su postali susedi nakon izbacivanja luka koji je pripadao srednjoj paraboli (luk koji je nestao).

Opis implementacije Forčenovog algoritma

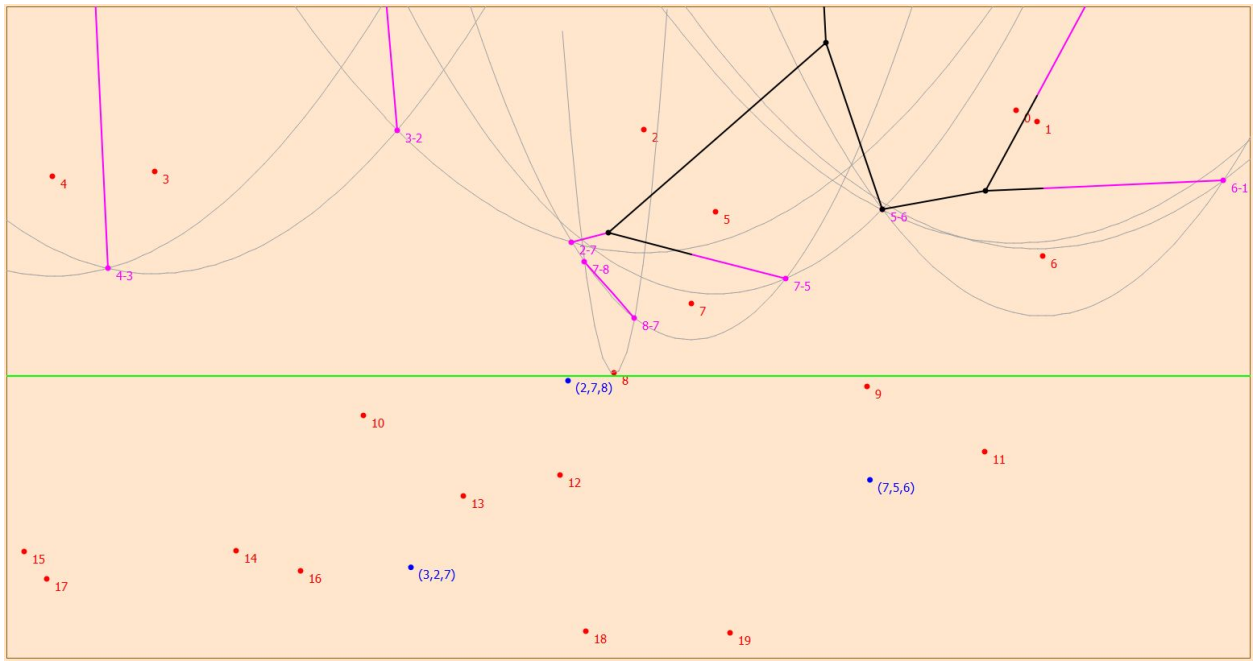
Za red događaja je korišćen vector koji čuva elemente tipa VoronoiEvent (dele se na događaj temena i događaj kruga).

Za opisivanje linije fronta je korišćen rečnik (map u C++-u interno koristi uređeno stablo) koji čuva preseke susednih parabola. Uređenje je obezbeđeno posebno dodatim komparatorom koji vrši sortiranje po x koordinati preseka.

Vizuelizacija algoritma

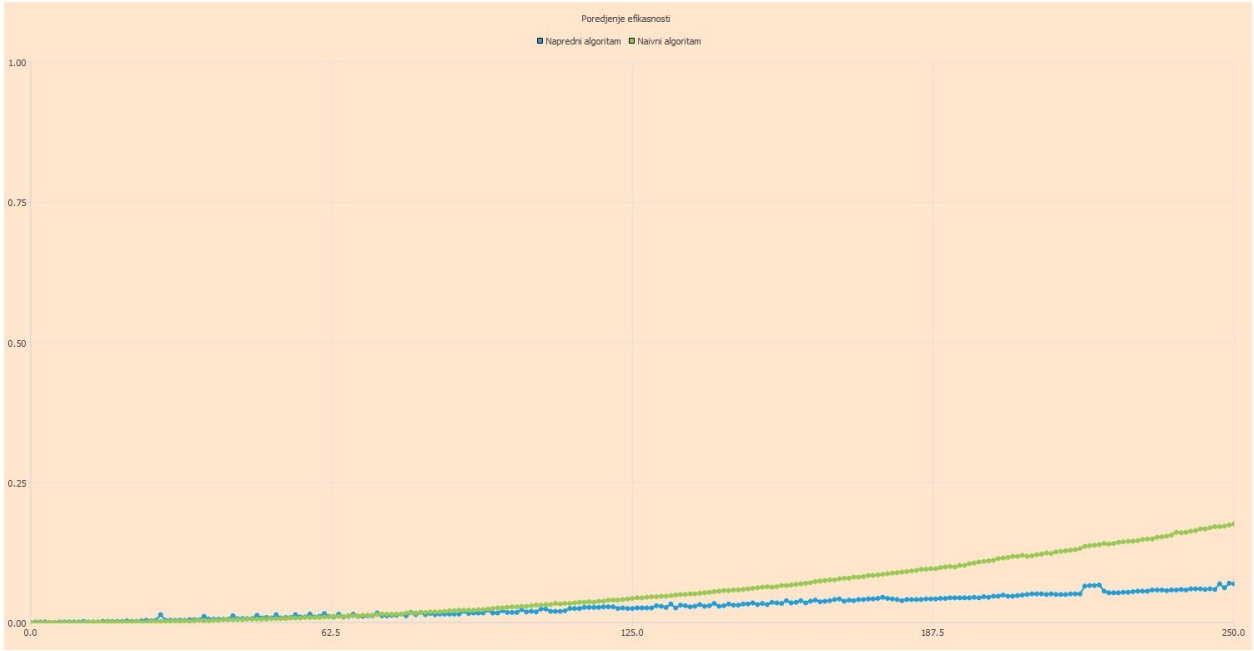


Prikaz nailaska na događaj sedišta (teme 8), primećuje se da su se ubacila dva preseka direktno iznad na luku koji pripada sedištu 7



Prikaz nailaska na događaj kruga (spajanje lukova 5-0-6), uporediti sa prethodnom slikom. Dodatno se može primetiti promena parabole koja pripada tački 8 (dodata na prethodnoj slici)

Poredjenje efikasnosti naivnog i naprednog algoritma



Testiranje ispravnosti algoritma

Provereno je da li rezultat naivnog algoritma ima isti broj temena kao i rezultat Forčenovog algoritma.

Naziv testa	Opis testa	Ulaz	Očekivani izlaz
oneDot	Specijalni slučaj. Očekuje se prazan niz za rezultat	Niz dimenzije 1	[]
twoDots	Specijalni slučaj. Očekuje se prazan niz za rezultat	Niz dimenzije 2	[]
hard1	Provera slučajnih tačaka	Niz dimenzije 50	Poklapanje rezultata naivnog i naprednog algoritma
hard2	Provera slučajnih tačaka	Niz dimenzije 20	Poklapanje rezultata naivnog i naprednog algoritma
hard3	Provera slučajnih tačaka	Niz dimenzije 25	Poklapanje rezultata naivnog i naprednog algoritma
hard4	Provera slučajnih tačaka	Niz dimenzije 70	Poklapanje rezultata naivnog i naprednog algoritma
specHorizontal	Specijalni slučaj	Ulaz koji sadrži i tačke sa istim y	Poklapanje rezultata naivnog i naprednog algoritma
specVertical	Specijalni slučaj	Ulaz koji sadrži i tačke sa istim x	Poklapanje rezultata naivnog i naprednog algoritma