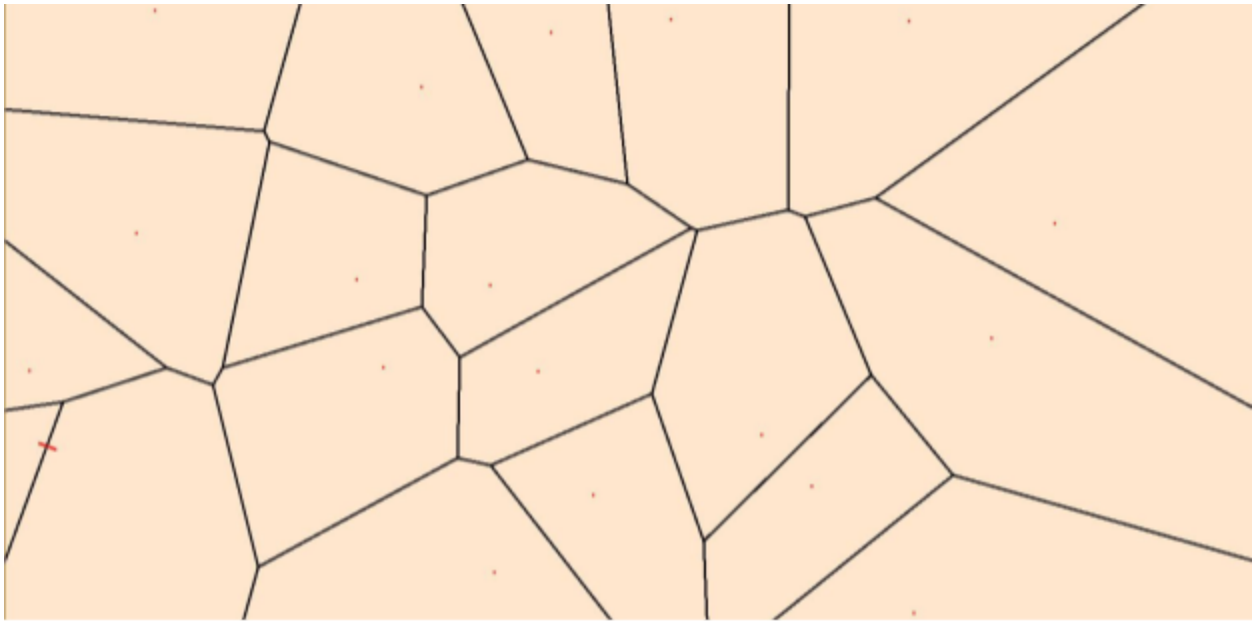
Algoritam za određivanje najbližeg para tačaka pomoću Voronoj dijagrama

Sreten Kovačević 1081/16



Slika 1: Prikaz vizuelizacije algoritma na kraju njegovog izvršavanja

Opis problema

Za skup tačaka X u ravni je potrebno odrediti takav par koji se nalazi na najmanjem međusobnom rastojanju. Za merenje rastojanja koristi se euklidska metrika.

Ulaz: *skup od n tačaka u ravni*

Izlaz: *Par koji predstavlja dve najbliže tačke*

Da bi ulaz bio validan, potrebno je da važi $n > 1$.

Naivno rešenje problema

Algorithm 1 Naive Algorithm

```
1: procedure NearestPair(X)
2:     minDistance = MAX_DOUBLE
3:     nearestPair = ()
4:     for i = 0; i < X.size(); i++ do
5:         for j = i + 1; j < X.size(); j++; do
6:             tmpDistance = distance(X[i], X[j])
7:             if tmpDistance < minDistance then
8:                 minDistance = tmpDistance
9:                 nearestPair = (X[i], X[j])
10:            end if
11:        end for
12:    end for
13:    Return nearestPair
14: end procedure
```

Algoritam 1: Pseudo-kod naivnog algoritma

Naivni pristup rešenju ovog problema predstavlja poređenje svakog para tačaka u potrazi za rešenjem. Ovakav pristup daje složenost $O(n^2)$.

Algoritam sa Voronoj dijagramom

Najpre je potrebno na osnovu ulaznog skupa tačaka konstruisati Voronoj dijagram. Ukoliko je $n = 2$, tada nije moguće konstruisati Voronoj dijagram (minimum 3 tačke) i rešenje je upravo taj par tačaka. Ukoliko je dijagram uspešno konstruisan, sledeća faza je poređenje relevantnih parova. Tačku *p* je dovoljno uporediti sa svim tačkama *q*, za koje važi da ćelije kojima one pripadaju imaju zajedničku granicu u dijagramu. Odnosno, proveravamo one tačke čije ćelije su susedne. Složenost konstrukcije dijagrama je $O(nlogn)$, dok je složenost druge faze, faze poređenja $O(n)$. To nas dovodi do ukupne složenosti ovakvog algoritma, koja iznosi $O(nlogn)$.

Ispravnost algoritma se zasniva na činjenici da Delone graf (dualni graf Voronoj dijagrama) predstavlja skup svih potencijalnih najkraćih rastojanja između tačaka. Kako su ivice Delone grafa upravo duži koje spajaju susedne ćelije, algoritam naveden narednim pseudo-kodom predstavlja upravo razmatranje koje od tih rastojanja je najkraće.

Algorithm 2 Efficient Algorithm

```
1: procedure NearestPairVoronoi(X)
2:     diagram = constructVoronoi(X)
3:     minDistance = MAX_DOUBLE
4:     nearestPair = ()
5:     for each cell cell in diagram do
6:         for each cell neighbor in cell.neighbors do
7:             tmpDistance = distance(cell.point, neighbor.point
8:             if tmpDistance < minDistance then
9:                 minDistance = tmpDistance
10:                nearestPair = (cell.point, neighbor.point)
11:            end if
12:        end for
13:    end for
14:    Return nearestPair
15: end procedure
```

Algoritam 2: Pesudo-kod algoritma sa Voronoj dijagramom

Boost Voronoj biblioteka

Za konstrukciju Voronoj dijagrama korišćena je C++ boost biblioteka. Za njeno ispravno funkcionisanje potrebno joj je proslediti podatke o strukturi podataka koja se koristi za reprezentaciju tačaka, kao i upoznati je sa metodama koje se koriste za dobijanje odgovarajućih koordinata. Voronoj dijagram ima razne mogućnosti iteriranja kroz dijagram. U okviru ovog projekta korišćena je iteracija kroz ćelije i iteracija kroz ivice odgovarajuće ćelije pri proveri suseda, a iteracija kroz ivice pri vizuelizaciji.

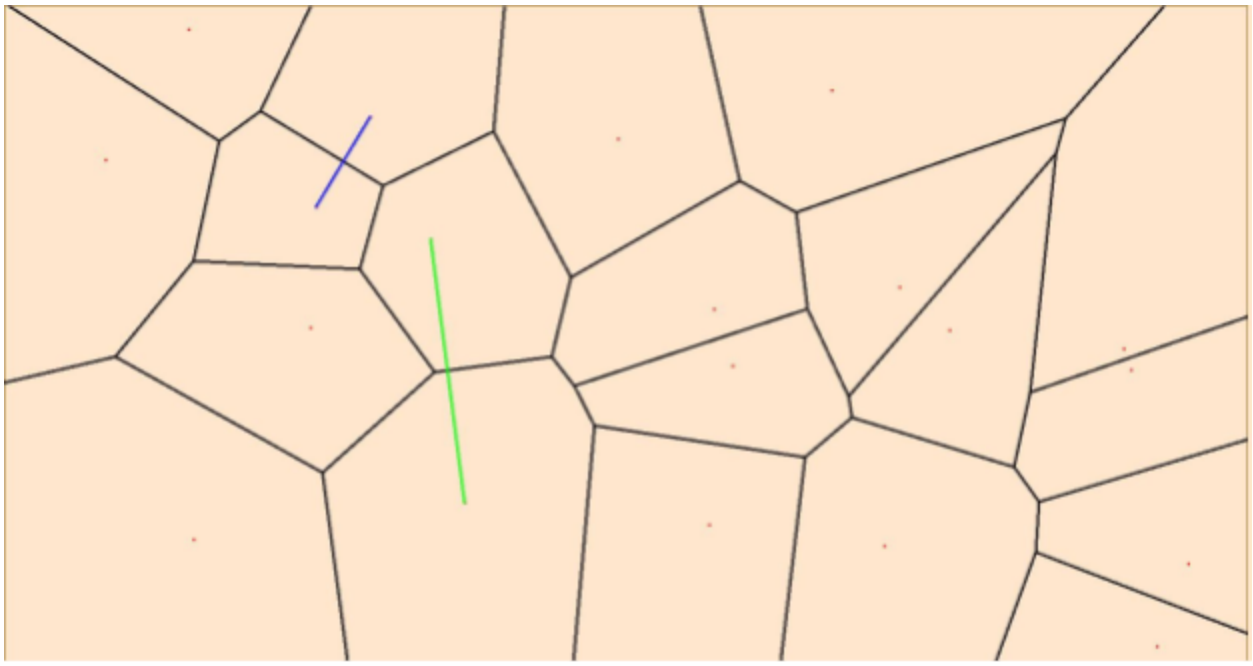
Osim navedene, biblioteka pruža mnogo širu podršku radu sa Voronoj dijagramom (na primer, generisanje na osnovu segmenta, konstrukcija dijagrama u prostoru, itd.). Dosta jednostavan uvod u rad sa ovom bibliotekom može se naći na adresi https://www.boost.org/doc/libs/1_67_0/libs/polygon/doc/voronoi_main.htm.

```
// Link QPoint with boost library, so it can be used for
// construction of Voronoi diagram.
namespace boost {
    namespace polygon {

        template <>
        struct geometry_concept<QPoint> { typedef point_concept type; };

        template <>
        struct point_traits<QPoint> {
            typedef int coordinate_type;

            static inline coordinate_type get(const QPoint& point,
                                              orientation_2d orient) {
                return (orient == HORIZONTAL) ? point.x() : point.y();
            }
        };
    }; // namespace polygon
}; // namespace boost
```

Slika 2: Kod koji povezuje Boost biblioteku sa QPoint klasom

Vizuelizacija algoritma

U okviru vizuelizacije algoritma crvenim tačkama su predstavljene tačke koje su ulaz algoritma. Crne linije predstavljaju ivice Voronoj dijagrama. U toku izvršavanja algoritma plavom linijom je povezan tekući par najbližih tačaka. Zelena linija povezuje par tačaka koji se trenutno razmatra kao potencijalni najbliži. Na kraju izvršavanja algoritma najbliži par se povezuje crvenom linijom (prikazano na naslovnoj slici).

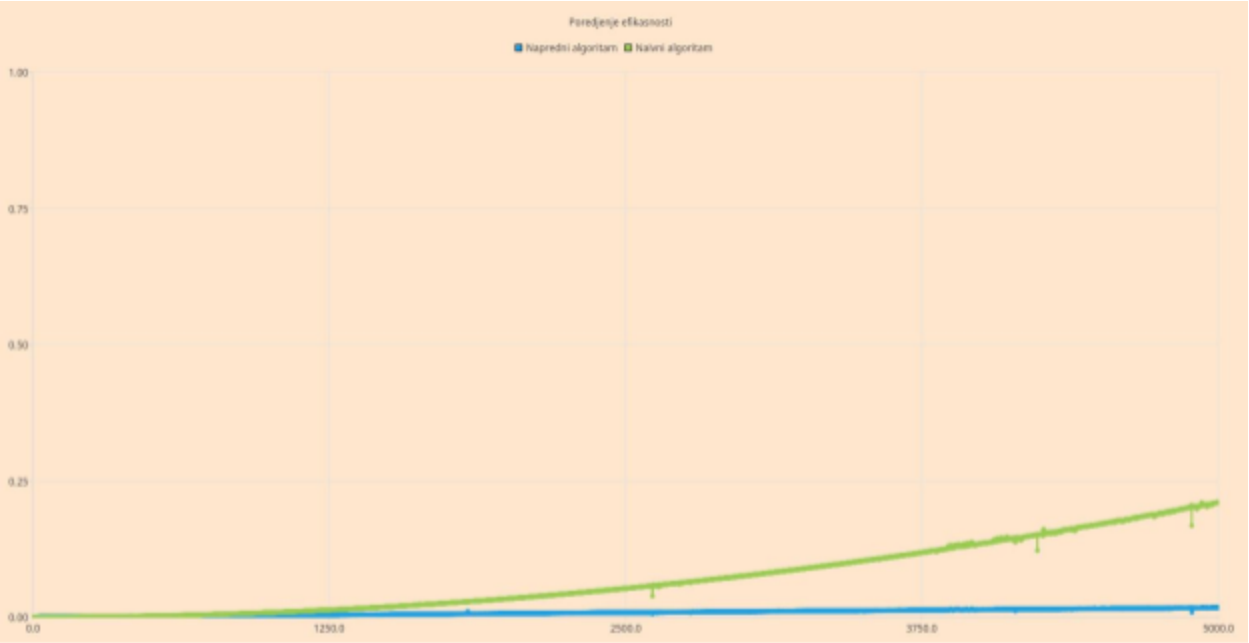


Slika 3: Prikaz vizuelizacije algoritma u toku rada

Poredjenje efikasnosti naivnog i naprednog algoritma

U tabeli je dat prikaz vremena (u sekundama) izvršavanja algoritama u odnosu na veličinu ulaza. Uneti rezultati predstavljaju srednju vrednost na 5 merenja.

alg\dim	100	500	1000	10000	20000	50000
Naivni	$8,6 * 10^{-5}$	$1,08 * 10^{-3}$	$4,44 * 10^{-3}$	$4,24 * 10^{-1}$	$1,69$	$10,5$
Optimalni	$3,69 * 10^{-4}$	$1,28 * 10^{-3}$	$1,84 * 10^{-3}$	$1,83 * 10^{-2}$	$3,78 * 10^{-2}$	$9,21 * 10^{-2}$



Slika 4: Grafički prikaz odnosa efikasnosti algoritama

Da bi jasnije prikazali razliku između ova dva algoritma, pokrenut je test nad $1.000.000$ tačaka. Naivnom algoritmu je trebalo 4214 sekundi, dok je optimalnom trebalo manje od sekunde, odnosno 0,9 sekundi.

Testiranje ispravnosti algoritma

Testiranje ispravnosti algoritama izvedeno je unit testovima (Google Test).

Naziv testa	Opis testa	Ulaz	Očekivani izlaz
WrongInput1	Zadavanje ulaza koji nije ispravan.	[]	INVALID INPUT
WrongInput2	Zadavanje ulaza koji nije ispravan.	[{0, 0}]	INVALID INPUT
CornerInput	Zadavanje ulaza koji ne generiše dijagram.	[{0, 0}, {20, 0}]	{{0, 0}, {20, 0}}

ThreePoints	Prvi regularan ulaza za konstrukciju dijagrama	[{0, 0}, {5, 10}, {50, 50}]	{{0, 0}, {5, 10}}
ThreeCollinearPoints	Regularan ulaz koji konstruiše atipičan dijagram	[{50, 50}, {50, 40}, {50, 80}]	{{50, 50}, {50, 40}}
MultipleSolution	Ulaz u kom postoji više parova na jednakom rastojanju	[{0, 0}, {50, 50}, {3, 4}, {53, 54}]	distance(output) = 5
RandomInput20	Nasumični ulaz	Niz dimenzije 20	Poklapanje rezultata sa naivnim algoritmom
RandomInput100	Nasumični ulaz	Niz dimenzije 100	Poklapanje rezultata sa naivnim algoritmom
RandomInput1000	Nasumični ulaz	Niz dimenzije 1000	Poklapanje rezultata sa naivnim algoritmom