Univerzitet u Sarajevu

Prirodno-matematički fakultet

Odsjek za matematiku

Uvod u kompjutersku geometriju

Tema

Implementirati algoritam koji koristi strategiju „Divide and Conquer“ za pronalaženje svih presječnih tačaka 𝑛 duži koje se nalaze u ravni. Ideja algoritma je sljedeća: odabirese proizvoljna (random) duž te se pronalaze svi presjeci prave koju određuje ta duž sa preostalim dužima. Ukoliko presjek pripada odabranoj duži, to je jedna od traženih tačaka presjeka. Svi presjeci prave sa dužima dijele svaku od tih duži na dvije nove duži (koje su sa različitih strana prave). Nakon toga algoritam se dvaput rekurzivno poziva, jednom na duži koje su sa jedne strane prave, a drugi put na one koje su sa druge strane. Obavezno izvršiti vizualizaciju te odrediti asimptotsku složenost algoritma.

Nazivi varijabli I funkcija korištenih u algoritmu

vector<Duz> duzi; //generisane duži

vector<Tacka> tacke; //vektor tačaka u kojima se desio presjek duži

void algoritam(); // dijeli duži na one iznad I ispod, I poziva algoritam\_rek nad njima

void algoritam\_rek(const vector<Duz>& duzi);

Algoritam je zasnovan na strategiji “Divide and Conquer” u tri koraka:

1. Divide

Biram random duž p, tako da dijelimo sve duži u ravni na one koje su iznad I one koje su ispod te duži p. U ovom projektu prvi-plave su ispod, a drugi-zelene su iznad. Pravu p dodajem u zelene.

Podjela duži se vrši tako što idemo petljom kroz duži,

* Prvi slučaj koji provjeravamo je slučaj kada prava kroz duž p, siječe neku duž vektora duzi[I].

Ako je zadovoljen uslov (Orijentacija(p.A,p.B,duzi[i].A)) != (Orijentacija(p.A,p.B,duzi[i].B)

&& Orijentacija(p.A,p.B,duzi[i].A) !=0 && Orijentacija(p.A,p.B,duzi[i].B)!=0) to znači da je položaj duži p I duzi[I] , kao da se sijeku. Krajnje tačke trenutne duži nalaze se s različitih strana duži p. U ovom slučaju zahtjevamo da se duži ne sijeku, tj. prava koja prolazi kroz duž p bi sjekla duzi[I]. Dakle potrebno je ispuniti I uslov (!daLiSeSijeku(p,duzi[i])) kada smo našli takve duži pronalazimo presjek pomoću funckije Tacka tackapresjeka(Tacka A, Tacka B, Tacka C, Tacka D), zatim provjeravamo da li je (Orijentacija(p.A,p.B,duzi[i].A)>0) ako je početak duži duzi[I] iznad prave p, onda pravimo novu duz nazvanu zelenaa koja ce pripadati vektoru drugi, sa početkom u s tački presjeka, I krajem u tački A trenutne duzi, dok će duž plava imati početak I kraj u tačkama s I krajnjoj tački trenutne duži, u drugom slučaju imali bismo obrnutu situaciju da je predhodno plava duž zelena, a zelena da je plava.

* Drugi slučaj koji provjeravamo je kada su duži u potpunosti iznad ili ispod duži p

Dakle ako uslov (Orijentacija(p.A,p.B,duzi[i].A)) != (Orijentacija(p.A,p.B,duzi[i].B)

&& Orijentacija(p.A,p.B,duzi[i].A) !=0 && Orijentacija(p.A,p.B,duzi[i].B)!=0) nije ispunjen, tada ćemo su obje krajnje tačke trenutne duži ili iznad ili ispod prave p. Dovoljno je samo provjeriti da li je jedna od tačaka pravi negativnu ili pozitivnu orijentaciju sa krajnjim tačkama duži p.

Ako je orijentacija negativna onda se ta duž dodaje u vektor prvi (kojeg čine duži ispod prave p), inače dodamo duž u vektor drugi (kojeg čine duži iznad prave p).

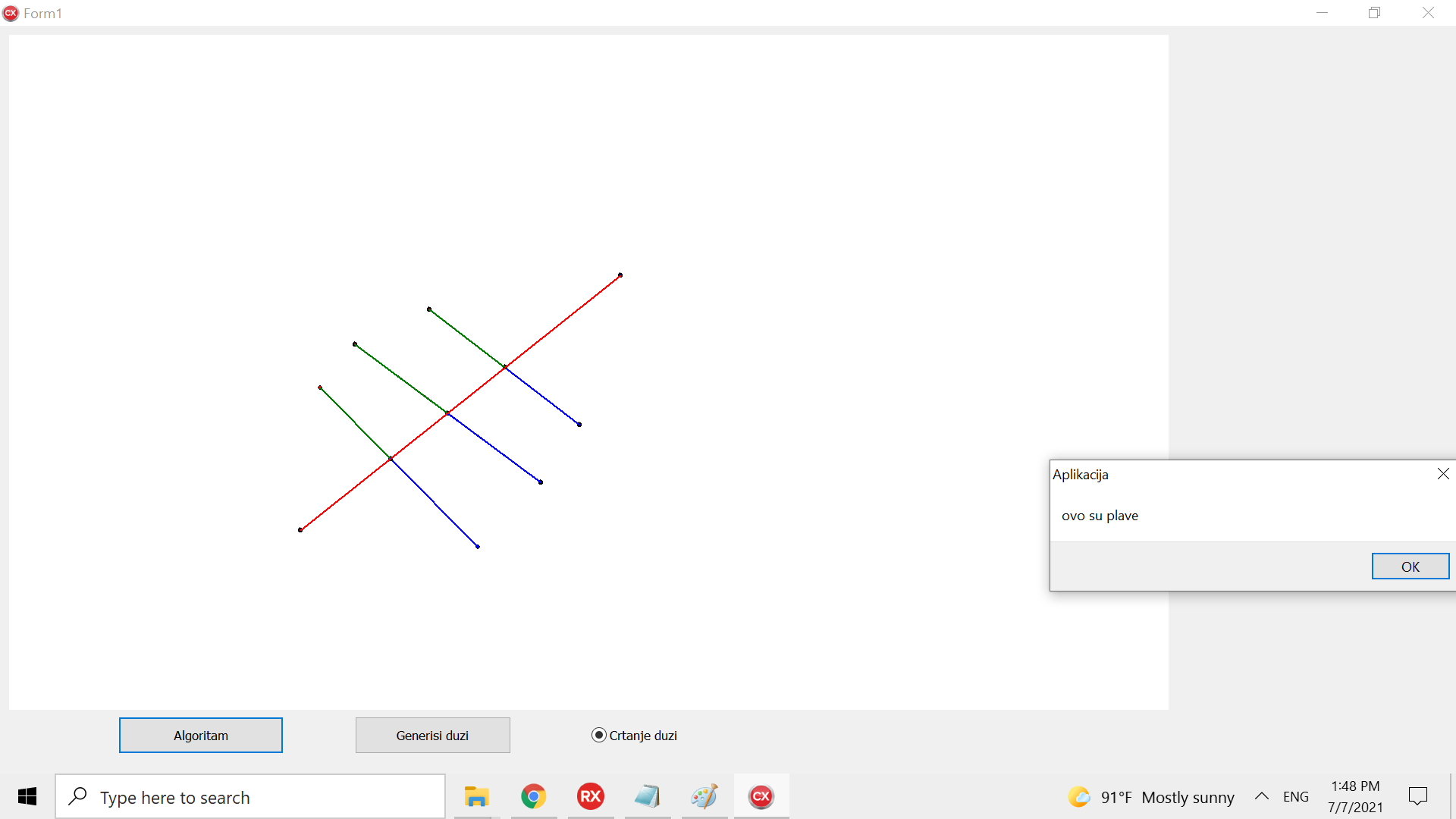
* Treći slučaj je ako se duž p siječe sa trenutnom duži

U tom slučaju pronalazimo samo tačku presjeka duži, kreiramo nove dvije duži sa krajevima u s I A ili sa krajevima B I s, zatim ih dodamo u odgovarajući vektor na osnovu toga da li su iznad ili ispod prave p. Dodam tačku presjeka u vektor tačke.

1. Conquer

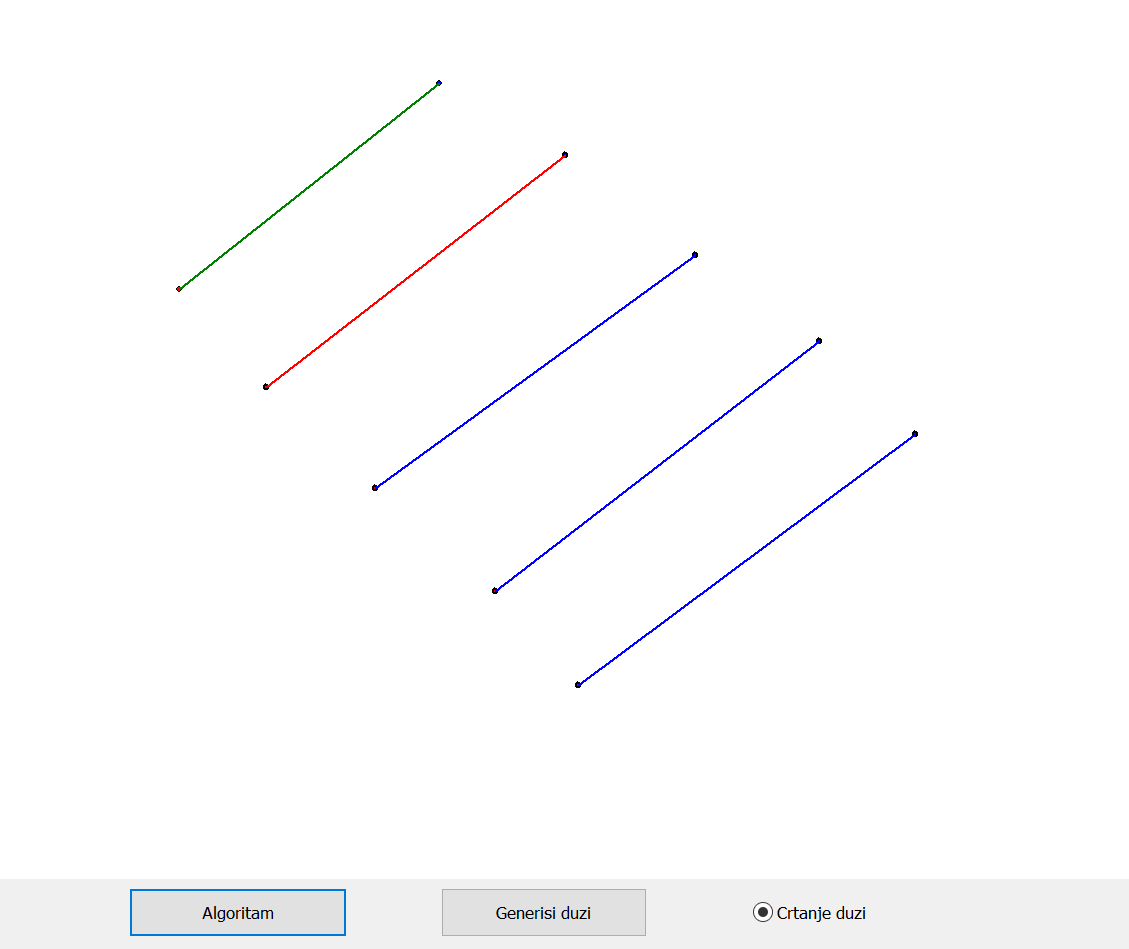
Pozivam rekurzivan algoritam na vektor plavih I vektor zelenih duži,,uslov kraja rekurzije da se proslijeđen vektor s nula,jednim ili dva elementa tj. duži ali koje se ne sijeku.

U najgorem slučaju mogu se generisati duži na sljedeći način,



Gdje se n duži dijeli na 2n-1 novih duži, tada bi imali rekurziju oblika T(n)= T(n) + T(n-1) + k, k broj presjeka najveći koji može biti je O(n^2). Vidimo da već u sljedećem koraku imamo situaciju ispod, tj. Najbolju situaciju gdje nema presjeka uopšte na gornjoj niti na donjoj strani.

Najbolja situacija je sljedeća,



Gdje prava p ne dijeli , tj. ne formira nove duži. Tada bismo imali rekurziju oblika T(n)=2\*T(n/2) + 1 to je vrijeme O(log n) . U ovom slučaju nemamo presjeka među dužima.

Vrijeme izvršavanja algoritma zavisi od broja novih presjeka,onda je vrijeme izvršavanja algoritma dato sa rekurzijom

T(n)=2T(n/2)+k , za k broj presjeka.

1. Combine

U vektor tačke dodajem sve presjeke duži s trenutnom duži p.

Strukture podataka I funkcije korištene u projektu

struct Duz {

Tacka A,B;

Duz():A(),B(){}

Duz(Tacka a, Tacka b)

void Crtaj(TImage \*Slika, TColor boja = clBlack) const;

}; //po x koordinati stavljakrajnje tačke

bool operator!=(Duz aA,Duz bB)

bool daLiSeSijeku(Duz prva, Duz druga)

bool Duzi\_iznad\_ispod (Duz prva,Duz druga)

Tacka presjekduzi(Duz pq, Duz ab)

struct Tacka {

double x,y;

Tacka(double X, double Y):x(X),y(Y) {}

Tacka ():x(),y(){}

void Crtaj(TImage \*Slika, TColor boja = clWhite) const;

};

bool operator<(Tacka A, Tacka B)

bool operator != (Tacka A,Tacka B)

Tacka tackapresjeka(Tacka A, Tacka B, Tacka C, Tacka D) ;

int Orijentacija(Tacka A, Tacka B, Tacka C);