UNIVERZITET U SARAJEVU
PRIRODNO - MATEMATIČKI FAKULTET
ODSJEK ZA MATEMATIČKE I KOMPJUTERSKE NAUKE

PROJEKAT IZ PREDMETA
"UVOD U KOMPJUTERSKU GEOMETRIJU"

Tema 5: Pareto

Predmetni profesor: prof. dr. Esmir Pilav

Predmetni asistent: Adisa Bolić, MA

Studentica: Berina Spirjan

Sarajevo, juli 2024. godine

SADRŽAJ

OSNOVNE INFORMACIJE O PROJEKTU

Zadatak ovog projekta jeste da odredimo Pareto figuru na osnovu sljedećeg kriterija: Za skup tačaka P={$p_1, p_2, \ldots, p_n$} u ravni, pri čemu je $p_i = (x_i, y_i)$, definišemo njegov podskup Pareto(P) kao skup kome pripadaju tačke p_i iz P za koje vrijedi da ne postoji tačka p_j $(i \neq j)$ u P takva da je $x_i \geq x_j, y_j \geq y_i$, tj. za svaku tačku u Pareto(P) vrijedi da ne postoji druga tačka u P koja se nalazi iznad ili desno od nje. Implementirati algoritam koji u vremenu $O(n \log n)$ pronalazi Pareto(P). Algoritam sortira tačke po jednoj od osa, kreće od tačke za koju sigurno možemo reći da se nalazi u Pareto(P), te prolazi kroz ostale sortirane tačke odlučujući za svaku da li pripada Pareto(P) ili ne. Potom ćemo odrediti vizualizaciju iscrtavanjem izlomljene duži koja predstavlja Pareto.

Pareto figura predstavlja skup tačaka u dvodimenzionalnom prostoru koje su optimalne u smislu dva kriterija koja smo prethodno naveli. Svaka tačka u Pareto figure je takva da nema druge tačke koja je bolja u oba kriterijuma istovremeno. Odnosno, u kontekstu dvodimenzionalnog prostora, to bi značilo da ne postoji tačka koja je istovremeno desno i iznad bilo koje tačke u Pareto figuri.

U ovom projektu obradit ćemo dva pristupa, budući da se algoritam koji se izvršava u vremenu $O(n \log n)$ može malo optimizirati u kontekstu vremenske složenosti, stoga ćemo predstaviti i to rješenje.

Kod je moguće pronaći in a Github repozitoriju:

https://github.com/berina-spirjan1/optimization-pareto

1. ORGANIZACIJA PROJEKTA

U ovom poglavlju dokumentacije obradit ćemo organizaciju projekta, raspored funkcija, metoda i komponenti koje će biti definisane u zaglavlju 2 ove dokumentacije, a ovdje ćemo definisati njihovu strukturu po fajlovima.

Projekat se sastoji od nekoliko osnovnih fajlova, a to su: **pomocna.cpp**, **pomocna.h**, **prozor.cpp**, **prozor.h**, **Project1.cpp**, **Project1CPH1.h**.

Header fajl **pomocna.h** sadži deklaracije funkcija i struktura koje se nalaze u projektu ciji inicijalni setup je iskorišten sa vježbi iz predmeta "Uvod u kompjutersku geometriju", akademske 2023/2024. godine. On uključuje sve neophodne biblioteke i definiše structure podataka kao što su: **Tačka**, **Duz** i **Pravougaonik,** kao i funkcije za rad sa njima.

Source fajl **pomocna.cpp** sadrži implementacije svih funkcija deklarisanih u **pomocna.h** fajlu. Funkcije uključuju pomoćne funkcije unutar kojih su implementirani algoritmi za poređenje tačaka, crtanje, itd.

Project1.cpp je inicijalni fajl koji služi kao main fajl u aplikaciji, on nam upravlja inicijalizacijom samog programa, kreirat će nam glavnu formu koju ćemo koristi kao GUI naše C++ aplikacije, te pokretanja samog programa. Dolazi zajedno sa **Project1CPH1.h** fajlom koji se pobraja, također u inicijalne fajlove RAD studija.

Fajl **prozor.cpp** služi za implementaciju interaktivnih komponenti unutar forme RAD studija, odnosno komponente koje koristimo za naš GUI. Ovaj fajl povezuje grafičke kontrole sa odgovarajućim funkcijama koje smo deklarisali unutar pomocna.cpp fajla.

2. IMPLEMENTACIJA

U nastavku ćemo detaljno razmotriti ključne aspekte koji su objedinjeni kroz implementaciju odredjenih funkcija, metoda, komponenti i sl.

2.1. vector<Tacka> grahamPareto(vector<Tacka>& tacke)

Funkcija **grahamPareto** koristi modifikovani Graham algoritam da pronađe Pareto figure iz zadatkog skupa tačaka P={p_1, p_2, ..., p_n}. Prije objašnjenja same implementacije razmotrimo koje su to modifikacije u odnosu na Graham algoritam.

Graham algoritam:

Grahamov algoritam se obično koristi za pronalaženje konveksnog omotača za zadani skup tačaka. Pa na osnovu ovoga, algoritam radi sljedeće:

1. Sortira sve tačke u odnosu na tačku sa najmanjom y koordinatom u odnosu na koordinatni početak
2. Potom prolazimo kroz niz sortiranih tačaka i koristimo stek u kojem čuvamo tačke koje čine konveksni omotač, pri čemu je neophodno da osiguramo da tačke ostanu u konveksnom omotaču bez obzira na dodavanje ili uklanjanje tačaka iz steak

Funkcija grahamPareto:

Ideja implementiranog algoritma kroz ovu funkciju je da sortira tačke po x koordinati u opadajućem redoslijedu i zatim prolazimo kroz niz ovih tačaka, dodavajući u niz samo one tačke koje imaju manju y koordinatu od prethodno dodanih tačaka. Na ovaj način, tačke koje pripadaju Pareto figure će ostati u nizu. Funkcija grahamPareto vrši sortiranje tačaka koristeći standardnu funkciju sort koja je ugrađena u C++. Koristi pomoćnu funkciju **porediTacke** koja će biti detaljnije obrađena u sekciji 2.2. Nakon sortiranja, kreiramo prazan vector "paretoFigura" koji će sadržati tačke Pareto figure. Prva tačka u sortiranom nizu se uvijek dodaje u Pareto figure je rima najveću x koordinatu. Nakon toga iteriramo kroz preostale tačke skupa tačaka. Za svaku tačku je neophodno provjeriti da li ima manju y koordinatu od posljednje tačke koja je dodana u niz paretoFigura. Ako tačka zadovoljava ovaj uslov dodaje se u paretoFigura niz, tj. u Pareto figuru. Na kraju funkcija vraća vektor tačaka Pareto figure.

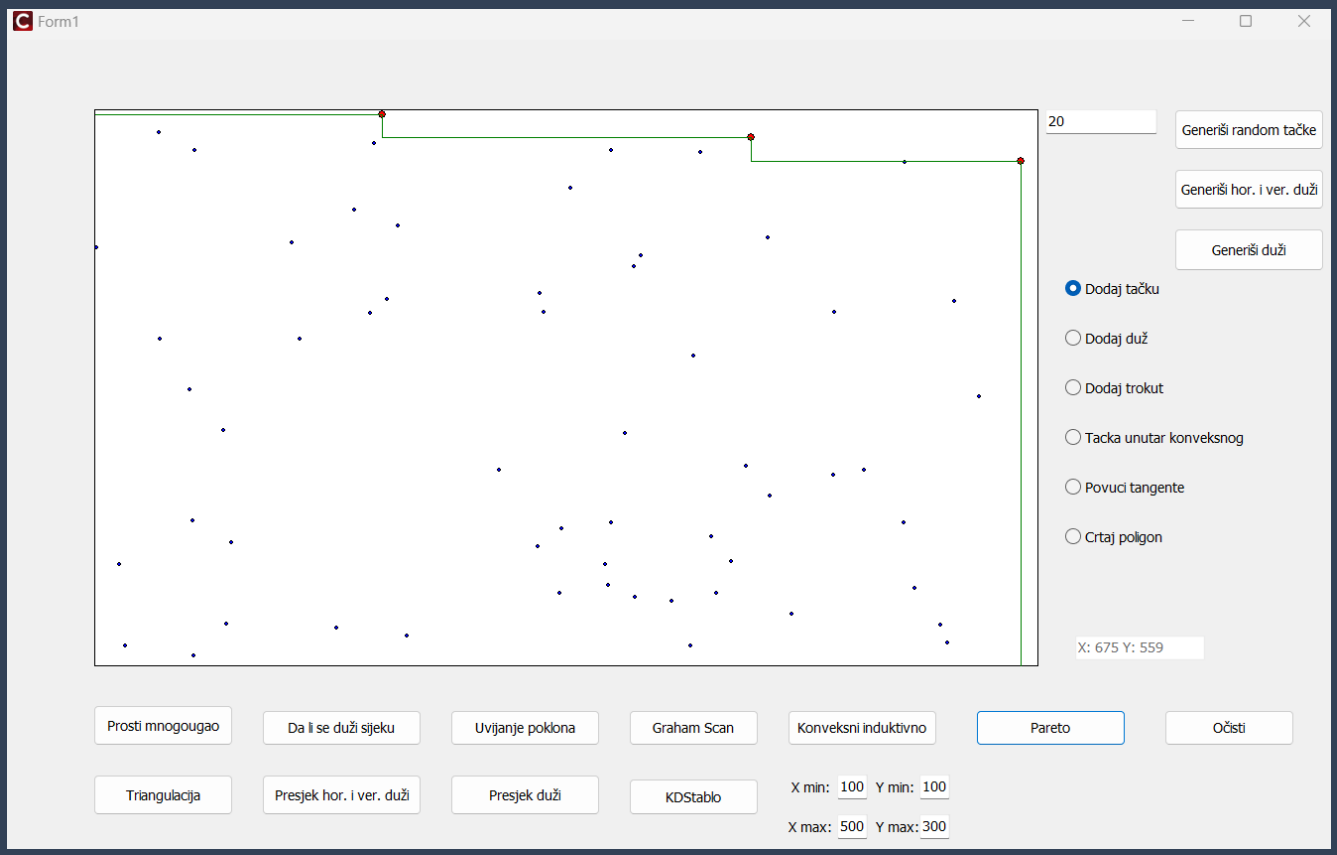
Vremenska složenost:

Razmotrimo vremensku složenost datog algoritma: za sortiranje koristimo $O(n\,logn)$ vremena, jer to je i vremensko izvršavanje sort funkcije ugrađene u C++. Iteracija kroz sve tačke zadanog skupa nam uzima $O(n)$. Sada ukoliko uporedimo ove dvije složenosti imamo $O(n\,logn) > O(n)$, pa zaključujemo da dominira operacija sortiranja tačaka. Dakle, ukupna vremenska složenost grahamPareto algoritma je $O(n\,logn)$.

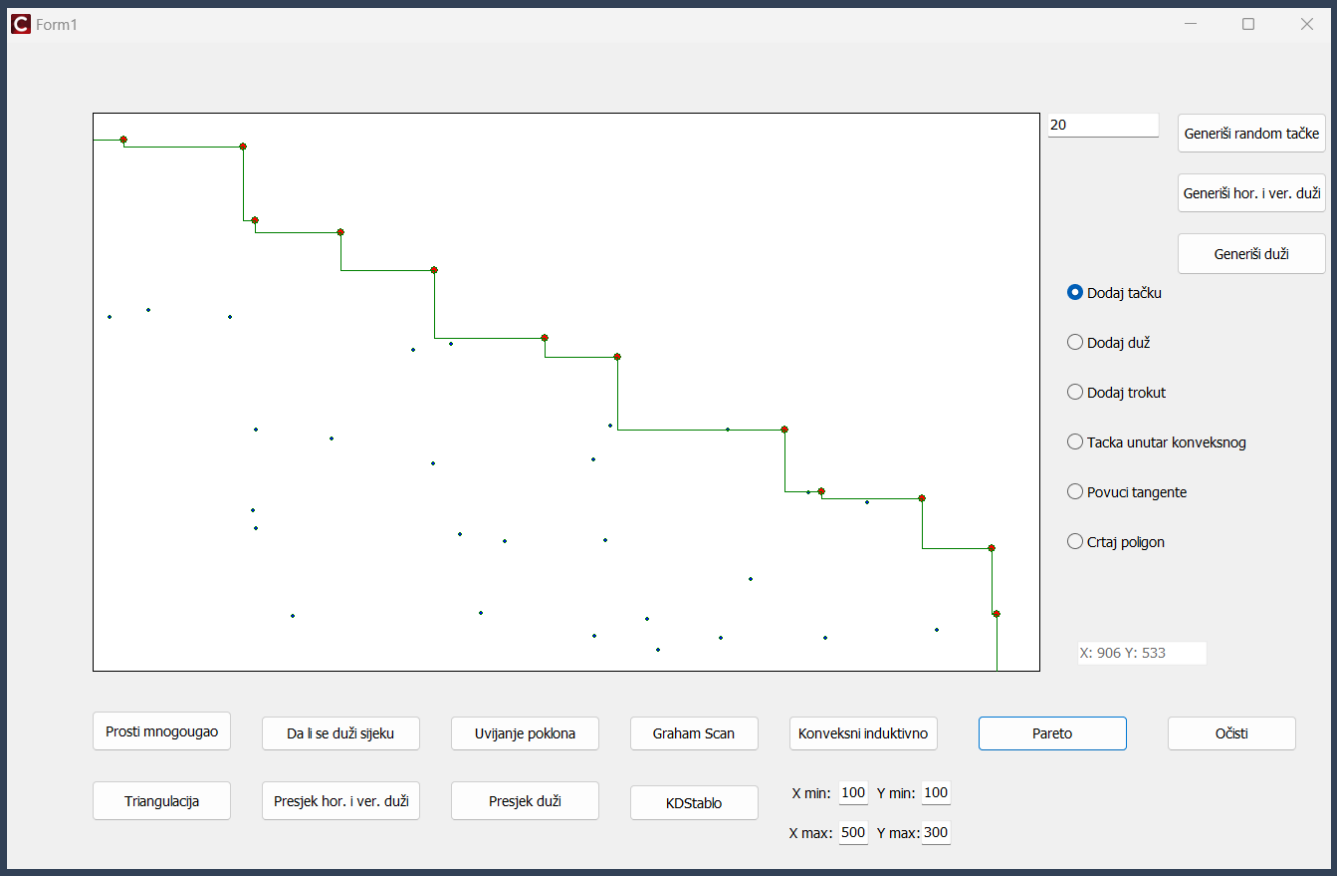
Memorijska složenost:

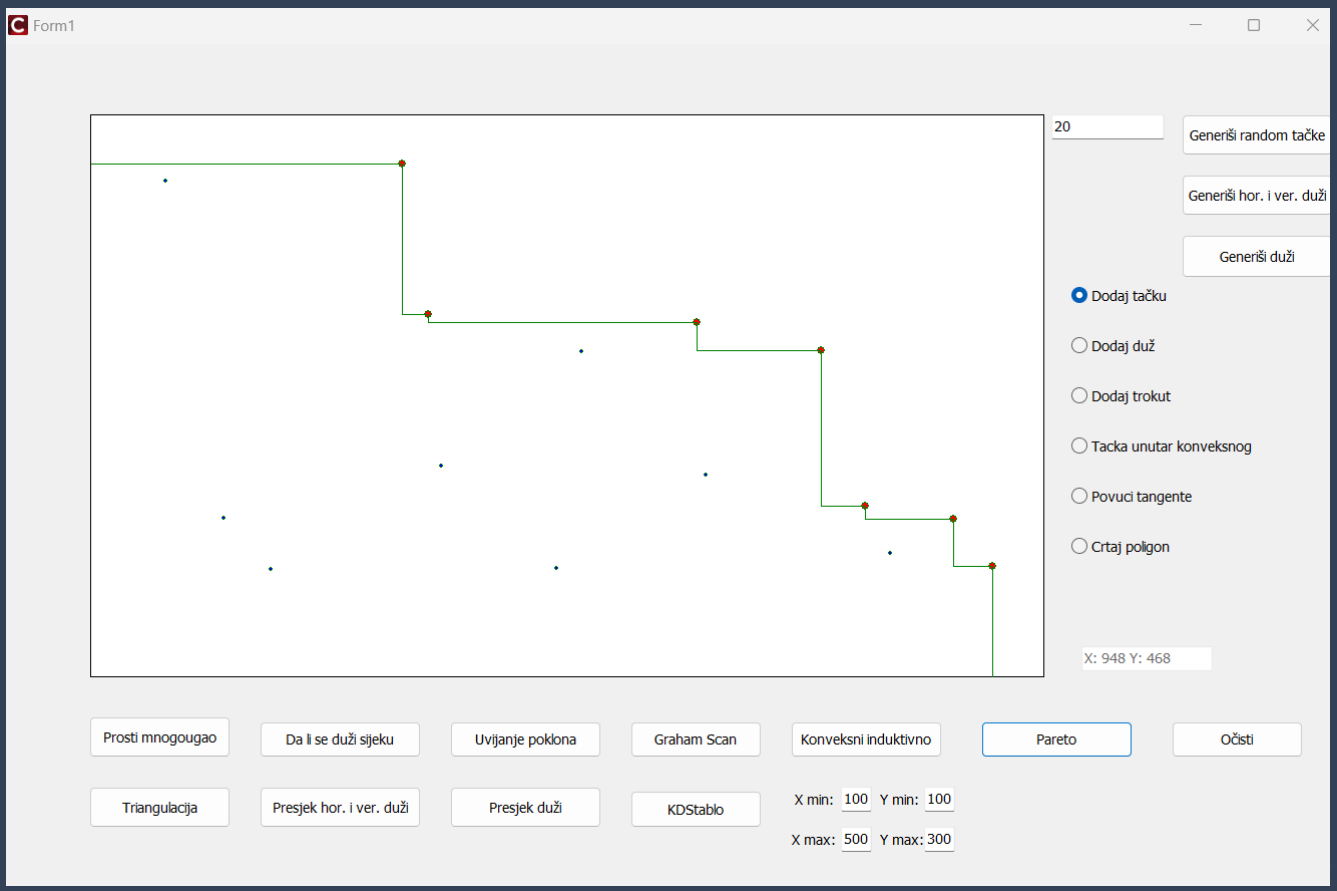
Kada govorimo o memorijskoj složenosti ovog algoritma, ona je jednaka $O(n)$ iz sljedećih razloga: budući da funkcija prihvata kao parametar referencu na niz tačaka, to nam ne zahtjeva dodatni memorijski prostor. Potom kreiramo vektor "paretoFigura", za koji u najgorem slučaju možemo imati da će sadržati n tačaka. Stoga zaključujemo da je memorijska složenost ovog algoritma jednaka $O(n)$.

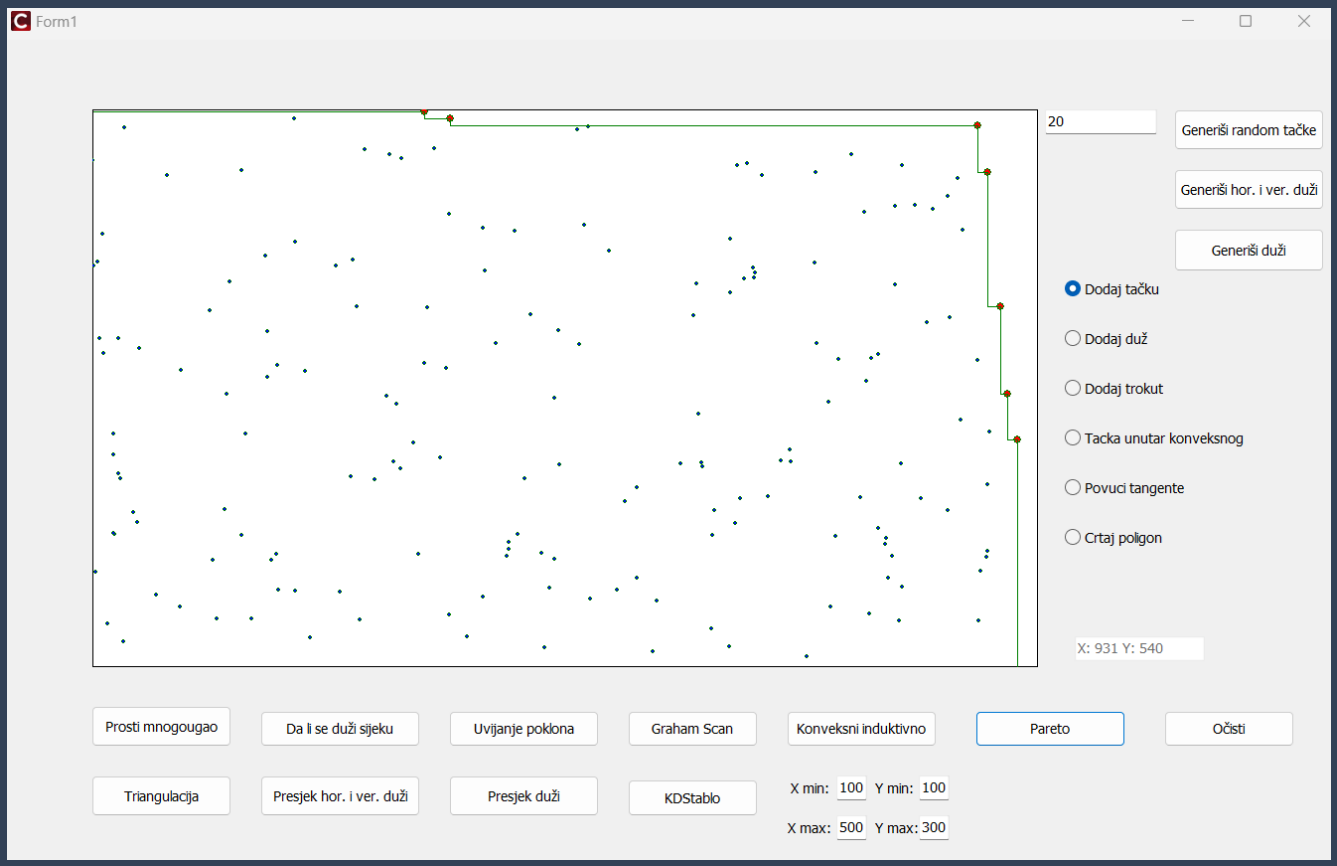
Vizualizacija:

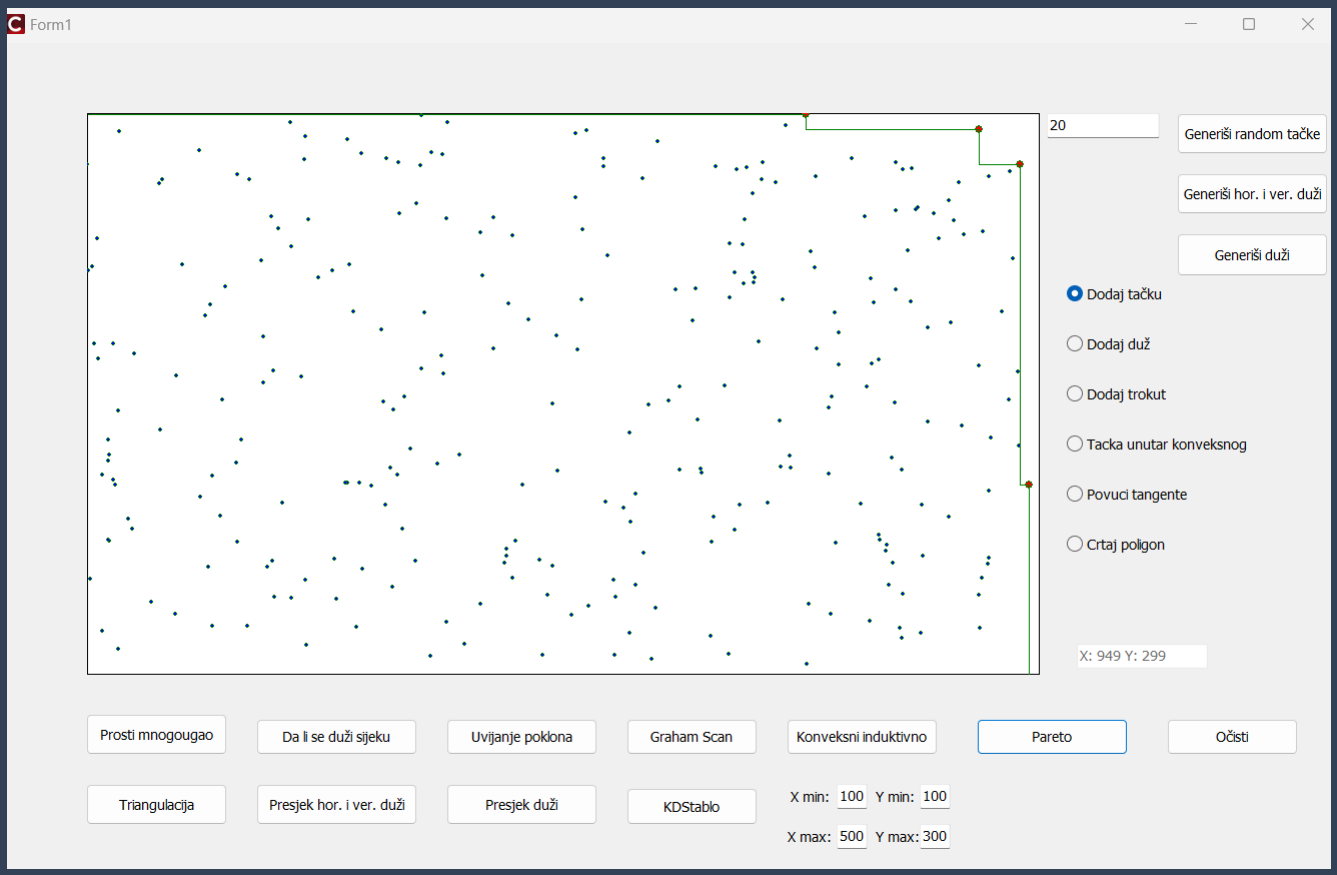


UNIVERZITET U SARAJEVU
PRIRODNO - MATEMATIČKI FAKULTET
ODSJEK ZA MATEMATIČKE I KOMPJUTERSKE NAUKE






Form1
20
Generiši random tačke
Generiši hor. i ver. duži
Generiši duži
Dodaj tačku
Dodaj duž
Dodaj trokut
Tacka unutar konveksnog
Povuci tangente
Crtaj poligon
X: 931 Y: 540
Prosti mnogougao
Da li se duži sijeku
Uvijanje poklona
Graham Scan
Konveksni induktivno
Pareto
Očisti
Triangulacija
Presjek hor. i ver. duži
Presjek duži
KDStablo
X min: 100 Y min: 100
X max: 500 Y max: 300


Form1
20
Generiši random tačke
Generiši hor. i ver. duži
Generiši duži
Dodaj tačku
Dodaj duž
Dodaj trokut
Tacka unutar konveksnog
Povuci tangente
Crtaj poligon
X: 949 Y: 299
Prosti mnogougao
Da li se duži sijeku
Uvijanje poklona
Graham Scan
Konveksni induktivno
Pareto
Očisti
Triangulacija
Presjek hor. i ver. duži
Presjek duži
KDStablo
X min: 100 Y min: 100
X max: 500 Y max: 300

2.2. bool porediTacke(const Tacka& a, const Tacka& b)

Ova funkcija poredi dvije tačke a i b pop njihovim koordinatama, u slučaju da su koordinate x jednake, u tom slučaju poredi tačke po njihovim y koordinatama. Funkcija se koristi kao pomoćna funkcija kod sortiranja tačaka u opadajućem redoslijedu po x koordinati, a u slučaju jednakih x koordinata, tačka sa manjom y koordinatom dolazi prva.

2.3. void __fastcall TForm1::ButtonParetoClick(TObject *Sender)

Funkcija, **ButtonParetoClick**, se koristi za iscrtavanje Pareto figure na kanvasu kada se klikne dugme. Funkcija koristi modifikovani Grahamov algoritam za pronalaženje Pareto figure i iscrtava sve tačke, uključujući i Pareto figuru, na kanvasu. Također, funkcija iscrtava dodatne linije za vizuelno naglašavanje Pareto figure.