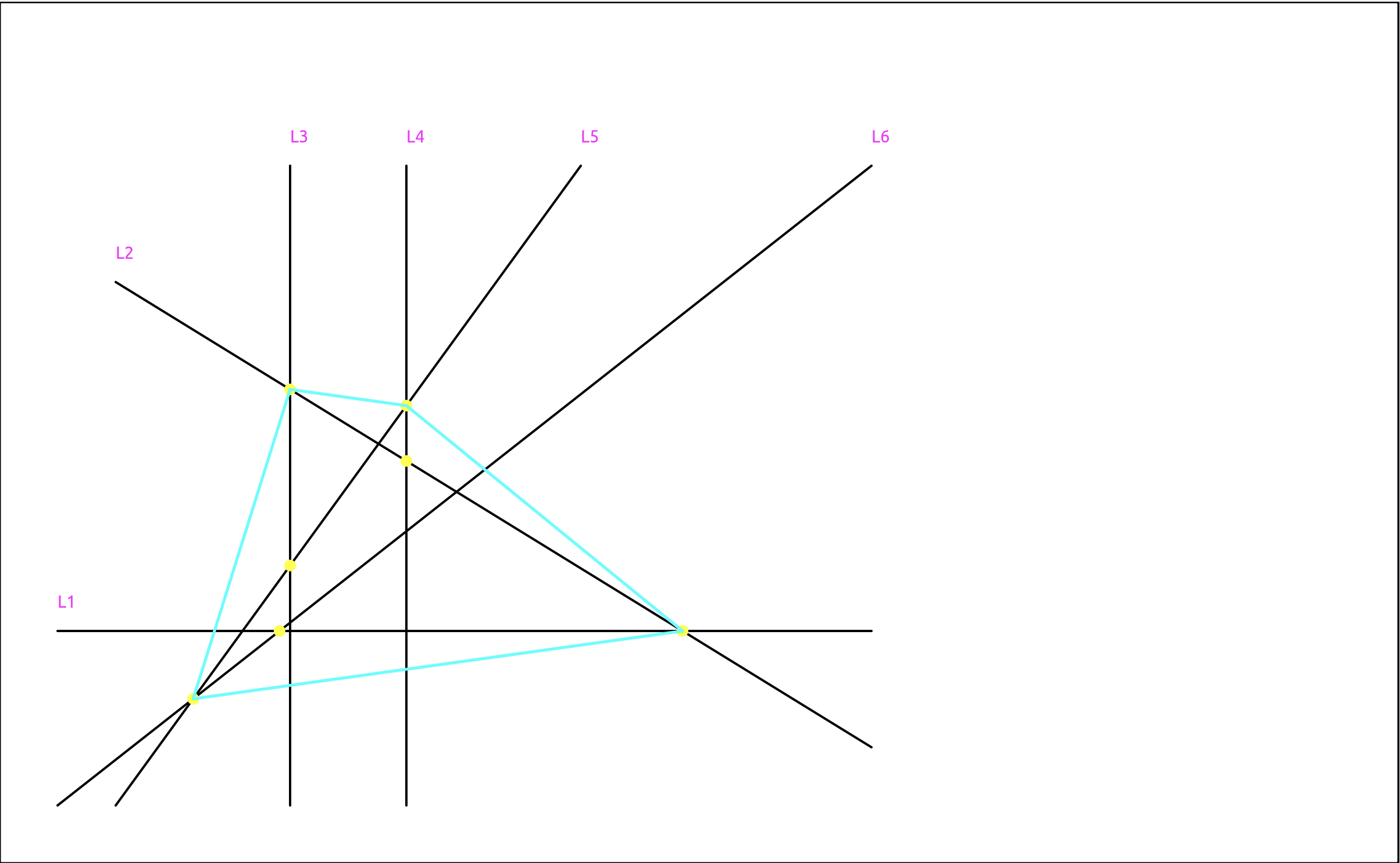
Geometrijski algoritmi @ MATF

**Algoritam za izračunavanje konveksnog omotača preseka linija**

Anđelka Milovanović

horizontal line



# **Opis problema**

**[\*Tekst je preuzet iz knjige profesora Janičića]** Skup tačaka 𝑋 (ravni ili prostora) je konveksan ako za svake dve tačke 𝐴 i 𝐵 skupa 𝑋 svaka tačka duži 𝐴𝐵 pripada skupu 𝑋. Za poligon (tj. za skup tačaka poligona) važi: poligon je konveksan ako su za svaku njegovu stranicu sve njegove tačke sa iste njene strane. Važi i sledeće: poligon je konveksan, ako su mu svi unutrašnji uglovi manji od opruženog. Konveksni omotač skupa tačaka je najmanji konveksan skup tačaka koji sadrži 𝑋. Za konačan skup tačaka u ravni, konveksni omotač je (konveksni) poligon. Za fiksiran skup tačaka, konveksni omotač je određen jednoznačno.

**[\*Atallah, M.J., 1986. Computing the convex hull of line intersections. *Journal of Algorithms*, *7*(2), pp.285-288.]** Na osnovu algoritma predloženog u ovom radu, potrebno je pronaći konveksni omotač preseka svih linija, ali tako da vremenska složenost bude O(nlogn). Prema tome, nije opcija da pronađemo sve preseke (njih može biti O(n2) među n linija), nego da redukujemo broj pronađenih presečnih tačaka, od kojih se sigurno može napraviti konveksni omotač preseka svih linija.

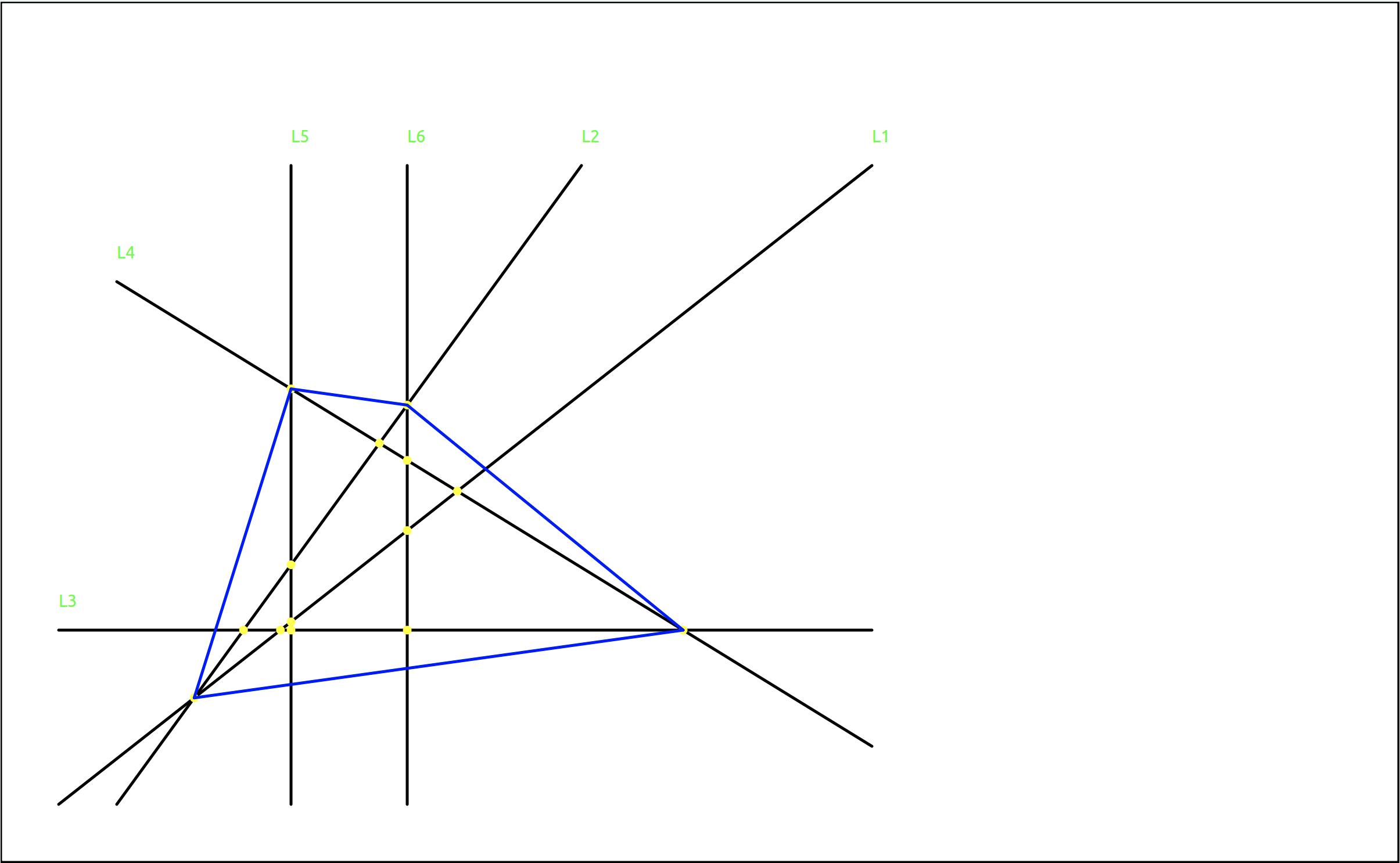
**Ulaz**: *skup od 2\*n tačaka u ravni, gde 2 tačke predstavljaju 1 liniju*

**Izlaz**: *skup tačaka koje predstavljaju konveksan omotač*

### **Naivno rešenje problema**

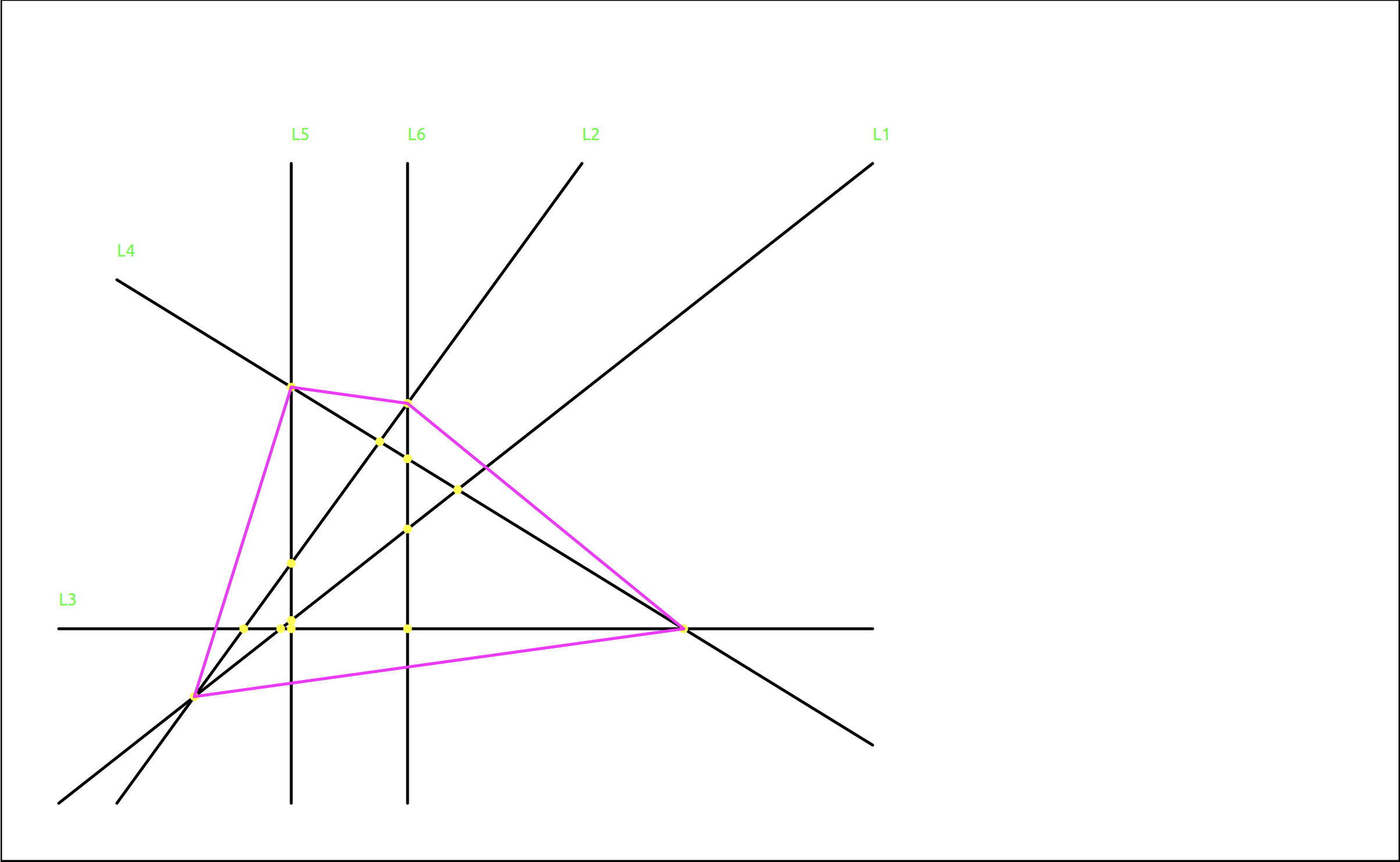
Naivno rešenje ovog algoritma je implementirano na dva načina.

Prvi primarni način je da se pronađu svi preseci linija u složenosti O(n2), gde se gledaju preseci svake sa svakom, a zatim da se pozove Gremov algoritam koji pronalazi konveksni omotač od formiranih tačaka u složenosti O(nlogn). Primer tako formiranog konveksnog omotača nad jednim veštački generisanim ulazom linija prikazan je na slici 1.



Slika 1: Konveksni omotač svih preseka linija (gruba sila + Gremov algoritam)

Drugi način je da se pronađu svi preseci linija u složenosti O(n2), gde se gledaju preseci svake sa svakom, a zatim da se algoritmom grube sile pronađe konveksni omotač od svih tih preseka u složenosti O(n3). Primer tako formiranog konveksnog omotača nad jednim veštački generisanim ulazom linija prikazan je na slici 2.



Slika 2: Konveksni omotač svih preseka linija (gruba sila preseka + gruba sila omotača)

Gremov algoritam je rađen u okviru kursa i detalji implementacije se mogu pogledati u literaturi predmeta i knjizi Computational Geometry. Napomena je da ulazne tačke algoritma mogu biti zadate i tipom QPointF (float vrednosti) i obrađen je slučaj kolinearnih i paralelnih pravih.

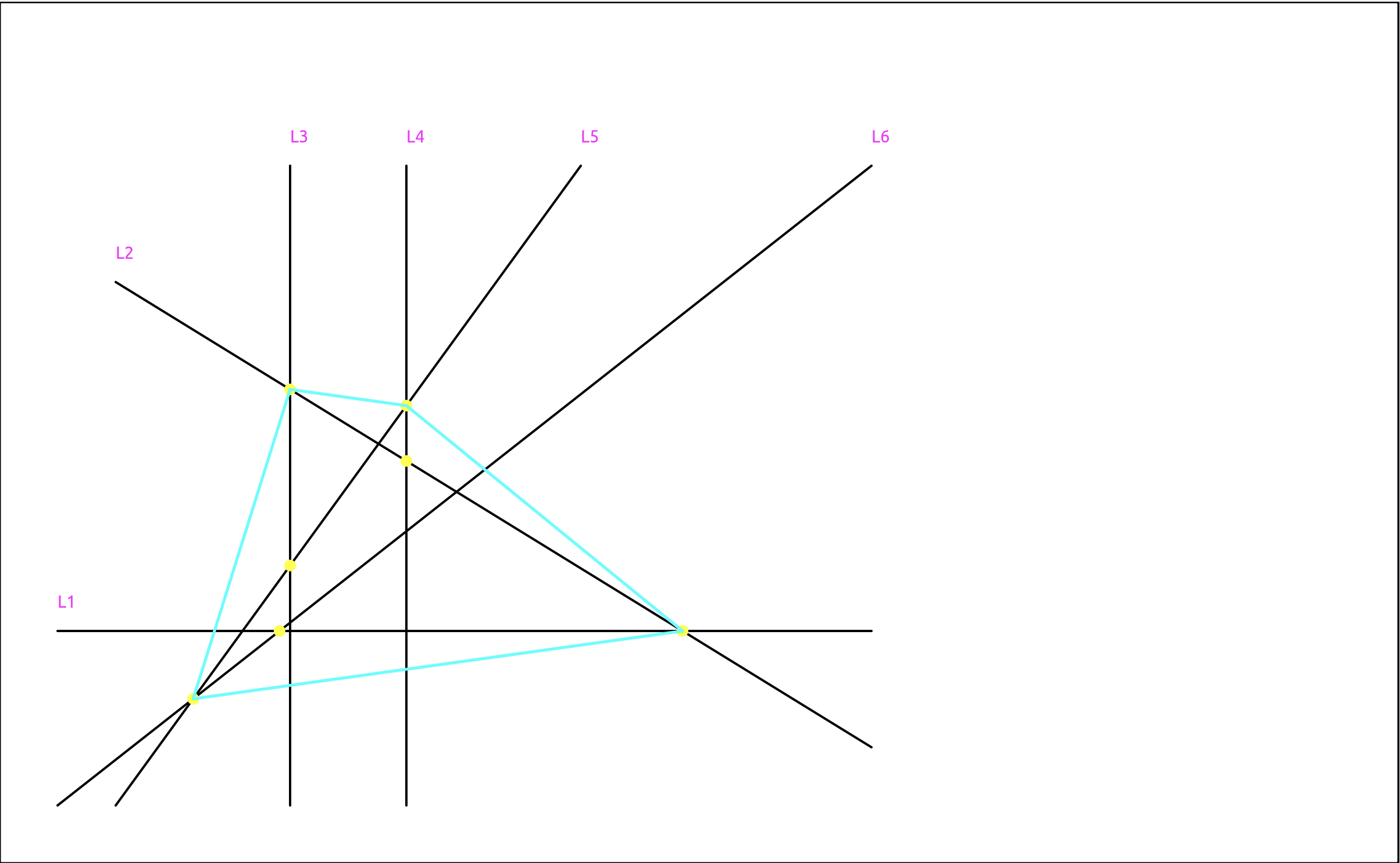
**Optimalni algoritam**

Implementiran algoritam u okviru ovog seminarskog rada je predstavljen u pomenutom radu: Computing the convex hull of line intersections (Atallah). Poenta ovog algoritma je vremenska složenost O(nlogn), gde n predstavlja broj ulaznih linija. Algoritam se sastoji iz 3 glavna koraka:

1. [O(nlogn)] Potrebno je sortirati sve linije prema nagibu ugla koji zaklapaju sa x-osom
2. [O(n)] Potrebno je proći kroz sortirane linije i pronaći preseke svake 2 susedne (prema uglu) - ukoliko je linija poslednja, onda se gleda njen presek sa prvom (preseka ima n i presek 2 se registruje u linearnom vremenu, tako da ovaj korak nosi O(n) vremena)
3. [O(nlogn)] Pozvati neki algoritam (u ovom slučaju Gremov algoritam) koji pronalazi konveksni omotač od pronađenih preseka

Ono što je zanimljivost je da u radu nije objašnjeno na koji način treba da se obrađuju specijalni slučajevi (kada postoje kolinearne tačke, kada je jedan presek dobijen od 3 ili više linija, kada imamo paralelne prave). Slučaj sa kolinearnim tačkama je rešen izbacivanjem svih središnjih tačaka u složenosti O(n), na kraju algoritma. Slučaj paralelnih linija je obrađivan tako što je korišćena struktura mape, gde ključ predstavlja vrednost ugla (koeficijent pravca), a vrednost je vektor linija koje imaju taj koeficijent pravca:

***std::map<double, std::vector<QLineF>> \_mapaUgaoDuzi;***

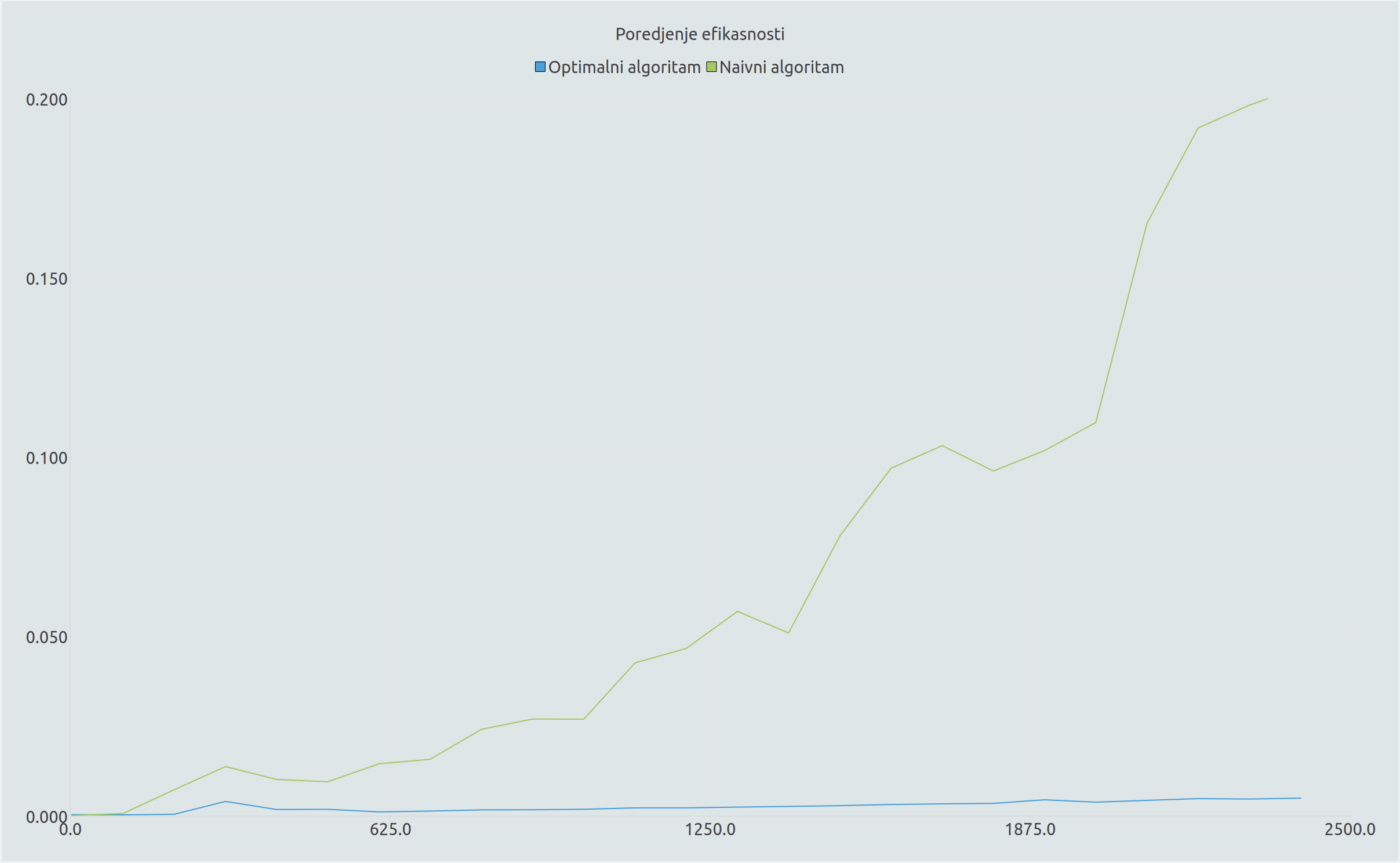
Ono što se radi je da se gledaju susedne linije prema uglu (odnosno 2 susedna ključa mape) i registruju se preseci prve i poslednje linije tekućeg ugla, sa prvom i poslednjom linijom susednog ugla. Na kraju se pozove Gremov algoritam za pronađene preseke. Primer ovako formiranog konveksnog omotača nad jednim veštački generisanim ulazom linija prikazan je na slici 3:

Slika 3: Konveksni omotač svih preseka linija (optimalni algoritam)

## **Poređenje efikasnosti naivnog i optimalnog algoritma**

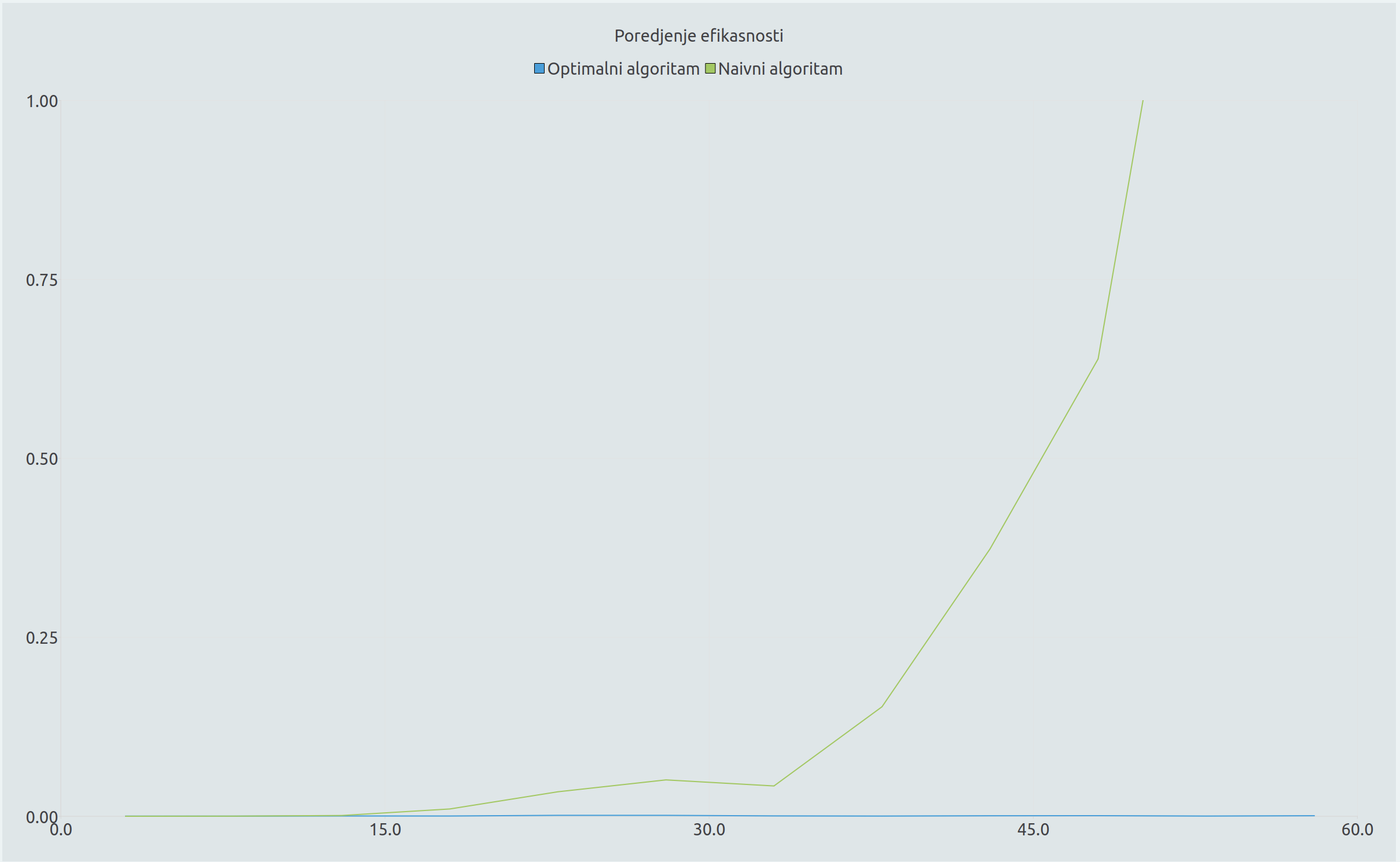
Poređenje je rađeno za optimalni algoritam i obe verzije naivnog algoritma.

Na slici 4 prikazan je grafik vremenskog izvršavanja u zavisnosti od veličine ulaza za optimalni algoritam (plavo) i naivni algoritam gde je Gremov algoritam korišćen za određivanje konveksnog omotača. Korišćeni parametri u config.h datoteci su MIN\_DIM=3, step=100 , MAX\_DIM=2500, Y\_MAX\_VAL=0.2.



Slika 4: Poređenje efikasnosti optimalnog algoritma i primarne verzije naivnog algoritma

Na slici 5 je prikazano isto to samo za drugu verziju naivnog algoritma (sve gruba sila). Korišćeni parametri u config.h datoteci su MIN\_DIM=3, step=5 , MAX\_DIM=50, Y\_MAX\_VAL=1.



Slika 5: Poređenje efikasnosti optimalnog algoritma i druge verzije naivnog algoritma

## **Testiranje ispravnosti algoritma**

Testiranje algoritma je rađeno pomoću Google Test okruženja unutar QtCreatora. Testirano je 13 slučajeva sa različitim zadatim ulaznim konfiguracijama, kao i sa nasumičnim brojem željenih linija koje treba da se generišu i za koje se testiraju algoritmi. Praćeno je poklapanje konveksnih omotača.

|  |  |  |  |
| --- | --- | --- | --- |
| **Naziv testa** | **Opis testa** | **Ulaz** | **Očekivani izlaz** |
| input10 | Zadavanje vrednosti 10 za generisanje nasumičnih linija u programu i testiranje poklapanja konveksnih omotača | brojLinija1=10 | Poklopljene tačke u konveksnim omotačima optimalnog i naivnog algoritma |
| input100 | Zadavanje vrednosti 100 za generisanje nasumičnih linija u programu i testiranje poklapanja konveksnih omotača | brojLinija2=100 | Poklopljene tačke u konveksnim omotačima optimalnog i naivnog algoritma |
| input1000 | Zadavanje vrednosti 1000 za generisanje nasumičnih linija u programu i testiranje poklapanja konveksnih omotača | brojLinija3=1000 | Poklopljene tačke u konveksnim omotačima optimalnog i naivnog algoritma |
| input10000 | Zadavanje vrednosti 10000 za generisanje nasumičnih linija u programu i testiranje poklapanja konveksnih omotača | brojLinija4=10000 | Poklopljene tačke u konveksnim omotačima optimalnog i naivnog algoritma |
| inputFileBasic | Zadavanje tačaka iz datoteke, manuelan ulaz | ulaz1 tj. datoteka input1.txt | Poklopljene tačke u konveksnim omotačima optimalnog i naivnog algoritma |
| inputFileCollinear | Zadavanje tačaka iz datoteke, postoje kolinearne tačke | ulaz2 tj. datoteka input2.txt | Poklopljene tačke u konveksnim omotačima optimalnog i naivnog algoritma |
| inputFileGridParallel | Zadavanje grida paralelnih linija iz datoteke | ulaz3 tj. datoteka grid\_parallel.txt | Poklopljene tačke u konveksnim omotačima optimalnog i naivnog algoritma |
| inputFileEmpty | Zadavanje prazne datoteke | ulaz4 tj. datoteka empty.txt | Poklopljene tačke u konveksnim omotačima optimalnog i naivnog algoritma |
| inputFile2Parallel | Zadavanje 2 paralelne prave | ulaz5 tj. datoteka  2\_parallel.txt | Poklopljene tačke u konveksnim omotačima optimalnog i naivnog algoritma |
| inputFileRandom | Nasumičan ulaz koji je u nekom trenutku pravio problem | ulaz6 tj. datoteka random\_generated.txt | Poklopljene tačke u konveksnim omotačima optimalnog i naivnog algoritma |
| testGenerisiLinije | Testiranje funkcije za nasumično generisanje linija | brojLinija1=10 | Poklapanje vraćenih linija sa brojLinija1 |
| testGenerisiLinijeInput0 | Testiranje funkcije za nasumično generisanje linija kad je uneta 0 | brojLinija=0 | Poklapanje sa unetim brojem linija |
| testGenerisiUglove | Testiranje broja vraćenih uglova | brojLinija1=10 | Broj vraćenih uglova treba da se poklopi sa brojem linija |