

Brža pretraga lokacija: Linearna pretraga vs KD-stablo

AJLA HODŽIĆ and ENSAR HODŽIĆ, Elektrotehnički fakultet u Sarajevu, BiH

U ovom projektu razvijena je demonstrativna web stranica za interaktivnu pretragu tačaka interesa na karti (npr. restorani) unutar zadanog radijusa oko odabrane lokacije. Glavna ideja je uporediti naivni linearni obilazak svih tačaka sa pretragom zasnovanom na *K-D stablu* (k-d tree), uz vizualizaciju procesa pretrage. Sistem koristi javne geo-servise (Nominatim za geokodiranje i Overpass API za dohvat OSM čvorova) i prikazuje rezultate na Leaflet mapi. Ovaj izvještaj daje pregled literature, teorijsku osnovu, opis arhitekture, detalje implementacije i opis korištenog skupa podataka/primjera, kao i finalnu diskusiju o rezultatima testiranja i evaluaciji performansi algoritama.

Additional Key Words and Phrases: K-D stablo, pretraga u radijusu, računaska geometrija, Web Mercator, OpenStreetMap, Leaflet

1 Uvod

Prostorne (geometrijske) pretrage tačaka u ravni pojavljuju se u velikom broju primjena: geolokacijske aplikacije, navigacija, analiza podataka, sistemi preporuka u prostoru, kao i računaska geometrija općenito. Jedna od najčešćih operacija je *range query* – pronalazak svih tačaka unutar kruga (radijusa) oko zadate pozicije.

Najjednostavniji pristup je linearno proći kroz sve tačke i izračunati udaljenost do centra. Međutim, kako broj tačaka raste, taj pristup postaje skup (vremenska složenost $O(n)$ po upitu). Zbog toga se u praksi koriste prostorne strukture podataka, npr. K-D stablo, koje omogućava bržu pretragu uz *pruning* (odsijecanje) dijelova prostora koji sigurno ne mogu sadržavati rješenje [1, 2].

Cilj projekta je implementacija i analiza K-D stabla u dvodimenzionalnom prostoru za potrebe pretrage tačaka unutar zadanog radijusa. Pored toga, razvijen je web-interfejs koji omogućava interaktivni izbor lokacije i radijusa pretrage na karti. Kao referentna metoda implementirana je i linearna pretraga, kako bi se omogućilo poređenje sa K-D pristupom. Poseban akcenat stavljen je na vizualizaciju toka pretrage, odnosno redoslijeda obilaska tačaka i čvorova, s ciljem boljeg razumijevanja ponašanja algoritama i razlika u njihovoj efikasnosti.

2 Pregled literature

K-D stablo je standardna struktura podataka u računskoj geometriji za indeksiranje tačaka u k -dimenzionalnom prostoru. U literaturi se najčešće opisuje kao binarno stablo koje na svakom nivou dijeli prostor hiper-ravni normalnom na jednu koordinatnu osu. U knjizi *Computational Geometry: Algorithms and Applications* detaljno se obrađuju prostorne strukture, pretrage i složenosti, te se K-D stablo navodi kao praktična metoda za upite tipa pretrage po opsegu (range searching) i najbližeg susjeda [1].

U predavanjima Goodrich–Wenk naglasak je na intuitivnom shvatanju pretrage: prvo se ide u “glavnu” granu u smjeru gdje leži ciljna tačka, a zatim se druga grana obilazi samo ako sfera pretrage presijeca razdjelnu ravan (tj. ako je apsolutna vrijednost razlike na osi manja ili jednaka radijusu). Time se postiže značajno smanjenje broja posjećenih čvorova u prosjeku [2].

Dodatno, bilješke sa predavanja (Haris Šupić) su korištene kao pomoć za terminologiju, očekivane cjeline izvještaja i praktične napomene oko implementacije i demonstracije algoritma [3].

Authors' Contact Information: Ajla Hodžić, ahrustic4@etf.unsa.ba; Ensar Hodžić, ehodzic8@etf.unsa.ba, Elektrotehnički fakultet u Sarajevu, BiH.

3 Teorijska osnova

3.1 Problem: pretraga tačaka u radijusu

Neka je dato n tačaka $P = \{p_i\}$ u ravni i upitna tačka q . Za dati radijus r želimo pronaći skup

$$\{p \in P \mid \text{dist}(p, q) \leq r\}.$$

3.2 Linearno pretraživanje

Linearno pretraživanje predstavlja najjednostavniji i konceptualno najdirektniji pristup rješavanju problema pretrage tačaka u prostoru. U ovom pristupu, za svaki upit sistem sekvencijalno prolazi kroz cjelokupan skup tačaka i za svaku pojedinačno izračunava udaljenost do upitne tačke. Tačke čija je udaljenost manja ili jednaka zadanom radijusu uključuju se u rezultat upita.

Vremenska složenost linearnog pretraživanja iznosi $O(n)$ po upitu, gdje je n ukupan broj tačaka, nezavisno od njihove prostorne raspodjele ili veličine radijusa. Iako je implementacija jednostavna i ne zahtijeva dodatne strukture podataka niti pripremne korake, ovakav pristup postaje neefikasan za veće skupove podataka. Zbog potrebe da se svaka tačka ispita pri svakom upitu, linearno pretraživanje ne omogućava nikakav oblik *pruning*-a prostora, što ga čini pogodnim isključivo kao referentnu ili baznu metodu za poređenje sa naprednijim prostornim strukturama, poput K-D stabla.

3.3 K-D stablo

K-D stablo (u 2D slučaju) je binarno stablo gdje svaki čvor sadrži jednu tačku i oznaku osi $axis \in \{0, 1\}$ na osnovu koje se vrši podjela. Ako je $axis = 0$, čvor dijeli prostor vertikalnom linijom $x = \text{const}$; ako je $axis = 1$, dijeli horizontalnom linijom $y = \text{const}$. U klasičnoj konstrukciji koristi se *medijan* po odgovarajućoj osi, čime se dobija relativno balansirano stablo [1].

3.4 Pseudokod K-D algoritma

Algorithm 1 Izgradnja K-D stabla

```

1: function BUILDKDTree(points, depth)
2:   if points je prazan then
3:     return null
4:   end if
5:    $axis \leftarrow \text{depthmod}k$ 
6:   sortiraj points po axis
7:    $median \leftarrow \text{srednjielement}$ 
8:    $node \leftarrow \text{novivorsamedian}$ 
9:    $node.left \leftarrow \text{BuildKDTree}(\text{lijevidio}, \text{depth} + 1)$ 
10:   $node.right \leftarrow \text{BuildKDTree}(\text{desnidio}, \text{depth} + 1)$ 
11:  return node
12: end function
```

3.5 Range search uz pruning

Za upit u radijusu, u svakom čvoru provjeravamo da li je tačka u čvoru unutar kruga. Zatim biramo “glavnu” granu prema znaku $\text{diff} = q[\text{axis}] - p[\text{axis}]$. Druga grana se pretražuje samo ako je $|\text{diff}| \leq r$, jer u tom slučaju krug može presijecati razdjelnu liniju/hyper-ravan i u suprotnom području mogu postojati validne tačke.

3.6 Koordinatni sistem i projekcija

Pošto se radi sa geografskim koordinatama (lat/lon), u implementaciji je korištena Web Mercator projekcija da bi se metrička udaljenost aproksimirala u ravni. Transformacija koristi radijus Zemlje $R = 6378137$ i preslikava (λ, φ) u metre (x, y) :

$$x = R\lambda, \quad y = R \ln \tan \left(\frac{\pi}{4} + \frac{\varphi}{2} \right),$$

što odgovara standardnoj projekciji korištenoj u web mapama. U kodu je ovo implementirano funkcijom `lonLatToMercator` (vidi Sekciju 5).

4 Opis sistema i arhitektura

Sistem je implementiran kao klijentska web aplikacija sa interaktivnom Leaflet mapom. Korisnik najprije unosi naziv željene lokacije, nakon čega aplikacija putem Nominatim servisa dobija odgovarajuće geografske koordinate. Na osnovu izabrane lokacije formira se upit prema Overpass API-ju, kojim se dohvaćaju OpenStreetMap čvorovi određene kategorije unutar šireg inicijalnog radijusa. Dobijene tačke se zatim projektuju u ravninski koordinatni sistem i koriste za izgradnju K-D stabla. Pretraga unutar korisnički definisanog radijusa vrši se nad tom strukturom, a rezultati se prikazuju na mapi. Sistem dodatno omogućava vizualizaciju toka pretrage kroz animirani prikaz posjećenih tačaka i odluka o grananju, čime se olakšava razumijevanje algoritma.

5 Implementacijski detalji

Cjelokupna implementacija razvijenog sistema, uključujući izgradnju K-D stabla, algoritme pretrage i vizualizaciju procesa, dostupna je u javnom GitHub repozitoriju <https://github.com/ensar-hodzic/NASP-Project>.

5.1 Geo pomoćne funkcije (projekcija)

U modulu `geo.ts` definisan je konstantni radijus $R = 6378137$ i funkcija `lonLatToMercator` koja prevodi (lon, lat) u (x, y) u metrima. Kod koristi standardnu Web Mercator formulu (log-tan za y). Ova projekcija omogućava da se pretraga u ravni implementira jednostavno, jer K-D stablo očekuje tačke u Euklidskom prostoru.

5.2 K-D stablo: konstrukcija i pretraga

K-D stablo je implementirano u `KdTree.ts` kao klasa `KDNode` sa poljima `point`, `axis`, `left` i `right`. Konstrukcija K-D stabla realizovana je rekurzivno. Na osnovu dubine čvora određuje se osa particije kao `axis = depth mod k`. Skup tačaka se zatim sortira po toj osi, a srednji element se bira kao tačka trenutnog čvora. Tačke sa manjim koordinatama formiraju lijevo podstablo, dok tačke sa većim koordinatama čine desno podstablo. Ovakav pristup rezultira relativno balansiranim stablom pogodnim za efikasnu pretragu.

Takav pristup daje balansirano stablo, ali zbog sortiranja na svakom nivou ima veću konstantu (u najjednostavnijoj varijanti približno $O(n \log^2 n)$), što je prihvatljivo za demonstraciju i umjerene veličine skupa.

Range pretraga `rangeSearch` koristi kvadrat udaljenosti `distanceSquared` (izbjegava korijen):

- (1) provjeri da li je trenutna tačka unutar radijusa ($d^2 \leq r^2$),
- (2) odredi granu koju treba prvu posjetiti na osnovu znaka `diff`,
- (3) eventualno posjeti i drugu granu ako $|diff| \leq r$ (krug presijeca razdjelnu ravan).

Pseudokod pretrage (u 2D, ali radi i za k dimenzija):

```
function RangeSearch(node, target, r, depth):
```

```

148 if node == null: return
149 axis = depth mod k
150 if dist2(node.point, target) <= r*r:
151     output.add(node)
152 diff = target[axis] - node.point[axis]
153 main = (diff < 0) ? node.left : node.right
154 other = (diff < 0) ? node.right : node.left
155 RangeSearch(main, target, r, depth+1)
156 if abs(diff) <= r:
157     RangeSearch(other, target, r, depth+1)

```

5.3 Dohvat podataka: Nominatim i Overpass

Dohvat podataka u sistemu oslanja se na javno dostupne geo-servise zasnovane na OpenStreetMap infrastrukturi. Nakon što korisnik unese naziv lokacije, aplikacija koristi Nominatim servis za geokodiranje tekstualnog upita i dobijanje geografskih koordinata centra pretrage. Ovaj korak omogućava fleksibilan unos lokacija bez potrebe za ručnim navođenjem koordinata.

Na osnovu dobijenog centra, sistem formira upit prema Overpass API-ju kojim se dohvaćaju OpenStreetMap čvorovi određene kategorije, konkretno objekti označeni kao `amenity=restaurant`. Upit se izvršava unutar unaprijed definisanog šireg radijusa kako bi se obezbijedio dovoljan skup tačaka za izgradnju prostorne strukture podataka. Dobijeni rezultati se filtriraju na čvorove tipa `node` i mapiraju u internu strukturu podataka koja sadrži identifikator, koordinate i osnovne oznake objekta. [5–7]

5.4 MapWithSearch: filtriranje putem K–D stabla

U komponenti `MapWithSearch.tsx` koristi se optimizacija putem `useMemo` kako bi se izbjegla nepotrebna ponovna izgradnja struktura. Svaka dohvaćena tačka se najprije projektuje u ravninske koordinate u metrima korištenjem Web Mercator projekcije. Nad tim projektovanim tačkama gradi se K–D stablo, nakon čega se za trenutno izabrani centar i radijus izvršava pretraga unutar opsega. Dobijeni rezultati se zatim mapiraju nazad na originalne OSM objekte i prikazuju na mapi.

Pošto `rangeSearch` vraća čvorove K–D stabla (samo projektovane koordinate), uvedena je mapa `projectedToRestaurant` koja spaja projektovanu tačku (ključ formiran sa `toFixed(3)`) sa originalnim OSM čvorom. Ovaj “lookup” omogućava da se rezultati pretrage prikazuju kao markeri na mapi sa nazivom restorana i dodatnim tagovima (npr. kuhinja, web stranica). [5–7]

5.5 LinearSearch: bazna metoda

Komponenta `LinearSearch.tsx` implementira linearnu pretragu: prolazi kroz sve tačke i za svaku računa Haversine udaljenost¹ u metrima (na sferi) u odnosu na centar. Time se dobija korektna geo-udaljenost bez projekcije, što je korisno kao referenca. Pretraga takođe generiše listu “koraka” (steps) koja se koristi za animaciju (posjećeno / u radijusu), ali bez grana i osi (u step-u `axis=-1`).

5.6 Vizualizacija procesa pretrage (trace/animacija)

Radi didaktičkog prikaza rada algoritma, implementirana je vizualizacija procesa pretrage koja omogućava praćenje redoslijeda obilaska tačaka. Tokom izvršavanja pretrage, sistem bilježi svaku posjetu čvora K–D stabla i čuva informacije o osi particije, udaljenosti od centra pretrage i odluci o eventualnom prelasku u drugu granu stabla.

¹Haversine udaljenost predstavlja standardnu formulu za izračunavanje najkraće udaljenosti između dvije tačke na površini sfere na osnovu njihovih geografskih koordinata.

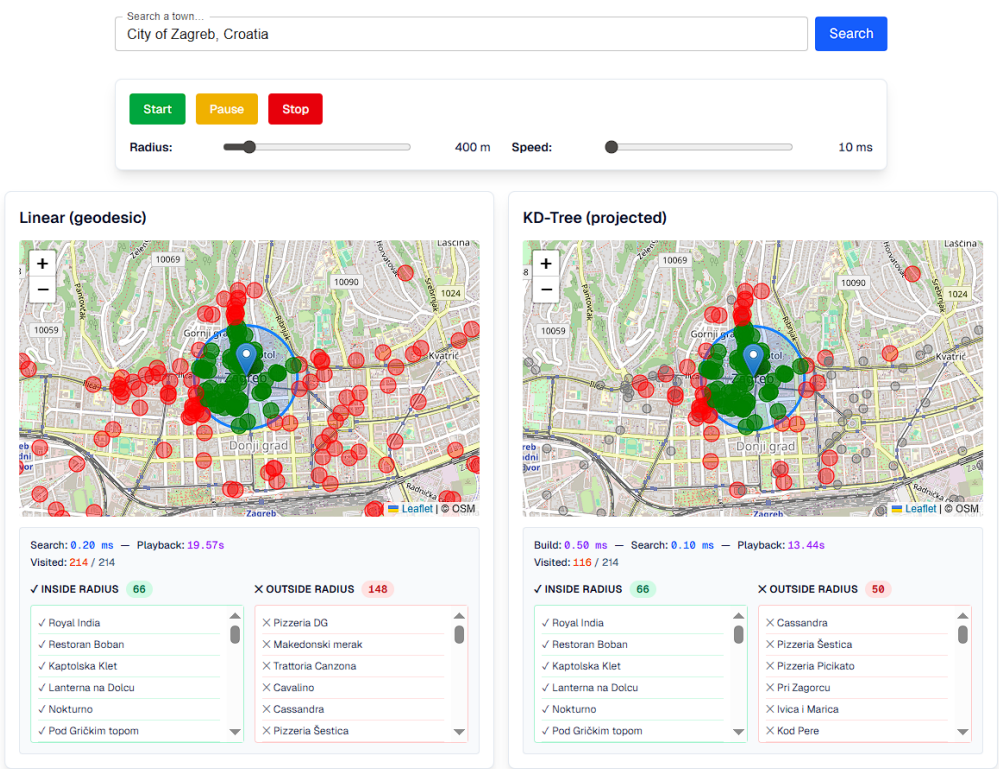


Fig. 1. Izgled gotove implementacije web stranice

Vizualizacija se realizuje kroz animirani prikaz na mapi, gdje se tačke koje su posjećene tokom pretrage grafički ističu. Tačke koje se nalaze unutar zadanog radijusa dodatno se označavaju kako bi se jasno razlikovale od ostalih posjećenih tačaka. Ovakav prikaz omogućava intuitivno razumijevanje principa *pruning*-a, odnosno razloga zbog kojih se određene grane stabla uopće ne obilaze.

6 Skup podataka i odabrani primjeri

U projektu se ne koristi unaprijed definisan statički skup podataka, već se ulazni podaci dinamički generišu u zavisnosti od korisničkog upita. Skup tačaka čine objekti preuzeti iz OpenStreetMap baze podataka, pri čemu se njihova brojnost i prostorna raspodjela razlikuju u zavisnosti od izabrane lokacije i veličine inicijalnog radijusa.

Ovakav pristup omogućava rad sa realnim podacima i realističnim prostornim raspodjelama, što doprinosi praktičnoj vrijednosti demonstracije algoritma. Istovremeno, promjenjiva veličina i gustoća skupa tačaka omogućavaju posmatranje ponašanja algoritma u različitim scenarijima, od rijetko naseljenih područja do urbanih sredina sa velikim brojem objekata.

7 Rezultati testiranja algoritama

Testiranje je provedeno nad skupovima tačaka interesa (POI) dobijenim iz OpenStreetMap-a za više gradova. Za svaki grad mjerena su vremena izvršavanja upita pretrage u radijusu za (i) linearnu pretragu (geodetski pristup) i (ii) pretragu zasnovanu na K-D stablu (projektovane koordinate). U skladu sa dizajnom aplikacije, u rezultatima je analizirano *isključivo vrijeme pretrage*, dok vrijeme izgradnje K-D stabla nije uključeno (pretpostavlja se da se struktura izgradi jednom nad dohvaćenim skupom tačaka, a zatim služi za više upita).

Pored vremena, zabilježen je ukupan broj POI (n), broj POI unutar radijusa (rezultati upita), te broj posjećenih čvorova/tačaka tokom pretrage. U eksperimentu sa više gradova, radijus pretrage je fiksiran na 600 m za sve gradove, kako bi se osigurala konzistentnost upita i omogućilo direktno poređenje performansi algoritama pri različitim veličinama i gustoćama skupa tačaka.

Tablica 1. Poređenje linearne pretrage i K-D pretrage (vrijeme pretrage u ms; n = ukupan broj POI).

Grad	n	POI_{in}	Lin (ms)	KD (ms)	KD_{vis}	KD_{nvis}	$KD_{vis}\%$	Speedup
Zagreb (CRO)	215	102	0.2384	0.1486	142	73	66.05	1.6
Barcelona (SPA)	1572	366	1.0728	0.1076	500	1072	31.81	10.0
New York (USA)	850	83	0.4416	0.0353	194	656	22.82	12.5
London (ENG)	614	65	0.2197	0.0335	154	460	25.08	6.6
Tokyo (JAP)	1176	187	0.1701	0.0598	297	879	25.26	2.8
Amsterdam (NED)	1049	262	0.3922	0.0644	400	649	38.13	6.1
Berlin (GER)	652	61	0.1053	0.0077	136	516	20.86	14.0
Sarajevo (BIH)	133	29	0.0255	0.0036	48	85	36.09	7.0
Paris (FRA)	1804	44	0.2289	0.0070	118	1686	6.54	33.0
Rome (ITA)	1086	80	0.2070	0.0110	180	906	16.57	19.0

Rezultati u Tabeli 1 pokazuju da K-D pristup u prosjeku ostvaruje značajno kraće vrijeme pretrage u odnosu na linearnu metodu. Dobitak je posebno izražen u slučajevima gdje se tokom K-D pretrage posjeti manji procenat čvorova (npr. Paris, gdje je posjećeno oko 6.5% tačaka), što direktno ilustrira efekat *pruning*-a. Nasuprot tome, kada je procenat posjećenih čvorova veći (npr. Zagreb), relativni dobitak je manji.

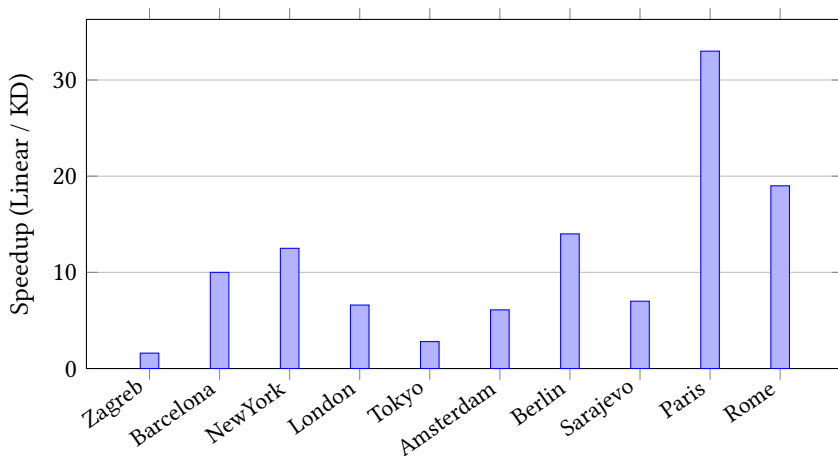


Fig. 2. Ubrzanje (speedup) K-D pretrage u odnosu na linearnu pretragu po gradovima

7.1 Analiza uticaja radijusa pretrage

Pored poređenja performansi algoritama na različitim gradovima, provedena je dodatna analiza nad jednim fiksnim skupom podataka kako bi se ispitao uticaj promjene radijusa pretrage na efikasnost algoritama. Za ovu studiju slučaja odabran je grad Prag, pri čemu je skup tačaka interesa (POI) ostao nepromijenjen, dok je radijus pretrage variran u opsegu od 200 m do 2000 m.

U ovom eksperimentu, K-D stablo je izgrađeno jednom nad kompletnim skupom tačaka, a zatim su izvršeni upiti sa različitim vrijednostima radijusa. Na taj način se izoluje uticaj samog upita pretrage i omogućava analiza ponašanja algoritma u zavisnosti od obima prostora koji se pretražuje.

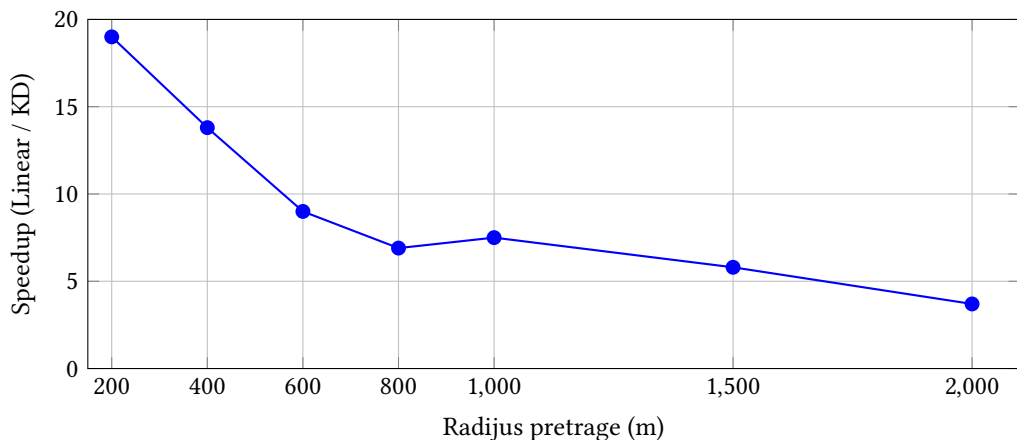


Fig. 3. Ubrzanje K-D pretrage u odnosu na linearnu pretragu u zavisnosti od radijusa pretrage (grad Prag).

Rezultati prikazani na Slici 3 jasno pokazuju da efikasnost K-D pretrage zavisi od veličine radijusa pretrage. Za male radijuse, K-D stablo ostvaruje vrlo značajno ubrzanje u odnosu na linearnu pretragu, jer se tokom upita obilazi samo mali dio stabla, dok se veliki dijelovi prostora efikasno odbacuju putem *pruning*-a. Kako se radijus povećava, sve veći dio prostora ulazi u oblast pretrage, što dovodi do povećanja broja posjećenih čvorova i smanjenja relativne prednosti K-D pristupa.

Tablica 2. Poređenje performansi linearne pretrage i K-D pretrage za različite vrijednosti radijusa (grad Prag, $n = 1041$ POI).

Radijus (m)	POI _{in}	Lin (ms)	KD (ms)	KD _{vis}	KD _{nvis}	KD _{vis} %	Speedup
200	50	0.1389	0.0073	110	931	10.57	19.0
400	166	0.1453	0.0105	297	744	28.53	13.8
600	286	0.1834	0.0203	413	628	39.67	9.0
800	402	0.1483	0.0215	560	481	53.79	6.9
1000	527	0.1429	0.0190	678	363	65.13	7.5
1500	778	0.1452	0.0251	973	68	93.47	5.8
2000	1041	0.1923	0.0518	1041	0	100.0	3.7

U graničnom slučaju, kada radijus obuhvata gotovo sve tačke u skupu, K-D pretraga se po ponašanju približava linearnoj metodi, jer je neophodno obići gotovo cijelo stablo. Ovaj eksperiment potvrđuje teorijska očekivanja i ilustruje da je K-D stablo naročito pogodno za upite sa manjim radijusima i lokalizovane pretrage, dok njegova prednost opada kako se obim pretrage povećava.

8 Zaključak

U ovom radu razmatran je problem prostorne pretrage tačaka interesa unutar zadanog radijusa, koji se često javlja u savremenim geolokacijskim aplikacijama, sistemima preporuka i drugim oblastima gdje se radi sa geografskim podacima. Poseban fokus stavljen je na poređenje dva različita pristupa rješavanju ovog problema: jednostavnog linearnog pretraživanja i pretrage zasnovane na K-D stablu kao prostornoj strukturi podataka.

Kroz teorijski dio rada prikazani su osnovni principi rada oba algoritma, njihove vremenske složenosti i karakteristična ponašanja. Linearno pretraživanje, iako veoma jednostavno za implementaciju i lako razumljivo, zahtijeva obilazak svih tačaka pri svakom upitu, što ga čini neefikasnim za veće skupove podataka. Sa druge strane, K-D stablo omogućava hijerarhijsku organizaciju prostora i selektivno obilazak samo onih dijelova prostora koji potencijalno mogu sadržavati rješenje upita, čime se postiže značajno smanjenje broja posjećenih tačaka.

Praktična implementacija i eksperimenti sprovedeni u ovom radu dodatno potvrđuju teorijska očekivanja. U eksperimentu koji je obuhvatio više različitih gradova, pri čemu je radijus pretrage bio fiksiran, pokazano je da K-D stablo u većini slučajeva ostvaruje značajno kraće vrijeme pretrage u odnosu na linearnu metodu. Razlika je naročito izražena u gradovima sa većim brojem tačaka interesa, gdje linearni algoritam nužno mora obraditi cjelokupan skup podataka, dok K-D pristup uspijeva izbjeći obilazak velikog dijela prostora.

Dodatna analiza provedena nad jednim gradom sa promjenjivim vrijednostima radijusa omogućila je dublje razumijevanje ponašanja algoritama. Rezultati jasno pokazuju da je prednost K-D stabla najveća kod manjih radijusa pretrage, kada se pretražuje relativno mali dio prostora. Kako se radijus povećava, oblast pretrage obuhvata sve veći dio skupa tačaka, što dovodi do smanjenja efekta *pruning*-a i postepenog približavanja ponašanja K-D algoritma linearnom pretraživanju. U graničnom slučaju, kada radijus obuhvati gotovo sve tačke, razlika između algoritama postaje znatno manja, što je u skladu sa teorijskim analizama.

Važan doprinos rada predstavlja i vizualizacija procesa pretrage. Prikaz posjećenih i neposjećenih tačaka, kao i redoslijeda obilaska čvorova u K-D stablu, omogućava intuitivno razumijevanje načina na koji algoritmi funkcionišu. Ovakav vizualni pristup pokazao se posebno korisnim za uočavanje efekta prostornog odsijecanja i za objašnjenje zašto K-D stablo u prosjeku posjećuje znatno manji broj tačaka u odnosu na linearni algoritam.

Na osnovu dobijenih rezultata može se zaključiti da K–D stablo predstavlja efikasno rješenje za prostorne upite u situacijama gdje se vrši veliki broj lokalizovanih pretraga nad istim skupom podataka. Ipak, važno je naglasiti da ono nije univerzalno optimalno rješenje za sve scenarije, te da njegova prednost zavisi od veličine skupa podataka, prostorne raspodjele tačaka i samog opsega pretrage.

Kao mogući pravci budućeg rada nameće se proširenje sistema na druge prostorne strukture podataka, kao što su R-stabla, kao i analiza ponašanja algoritama u višedimenzionalnom prostoru. Također, dodatna optimizacija i keširanje podataka, kao i testiranje sistema u realnim aplikacijama sa velikim brojem uzastopnih upita, moglo bi dodatno istaći praktičnu vrijednost i primjenjivost pristupa predstavljenog u ovom radu.

Literatura

[1] Mark de Berg, Otfried Cheong, Marc van Kreveld, i Mark Overmars. *Computational Geometry: Algorithms and Applications*. 3. izdanje. Springer, 2008.

[2] Michael T. Goodrich i Carola Wenk. Predavanja/skripte: *K-D Trees*. (note: materijali sa predavanja), 2025.

[3] Haris Šupić. Bilješke sa predavanja iz predmeta NASP. Interni materijal, 2025.

[4] Leaflet. *Leaflet: An Open-Source JavaScript Library for Mobile-Friendly Interactive Maps*. Projekat i dokumentacija (web resurs), pristupljeno 2025.

[5] OpenStreetMap contributors. *OpenStreetMap*. Otvorena baza geografskih podataka (web resurs), pristupljeno 2025.

[6] Nominatim. *Nominatim: OpenStreetMap Search*. Geokodiranje i pretraga lokacija (web resurs), pristupljeno 2025.

[7] Overpass API. *Overpass API*. API za upite nad OpenStreetMap podacima (web resurs), pristupljeno 2025.