Optimasi Pencarian Berbasis Inverted Index dan Binary Search pada Aplikasi Desktop Free Pascal

Ade Hastuty[1], Annisa Nur Hikmah[2], Achmad Zacky Dhienata Putra[3], Arpan Maulana[4]
[1,2,3,4] Program Studi Teknik Informatika, Universitas Muhammadiyah Parepare, Indonesia
Email: adehastuty@iainpare.ac.id, annisanurhikmah531@gmail.com,
achmadzacky923@gmail.com, arpan6154@gmail.com

Abstrak

Performa pencarian pada aplikasi desktop cenderung menurun seiring bertambahnya volume data, terutama pada pendekatan Linear Search yang berkompleksitas waktu O(N), sementara aplikasi desktop skala kecil-menengah umumnya tidak memiliki mekanisme indexing bawaan sebagaimana tersedia pada sistem basis data modern. Penelitian ini mengoptimalkan performa pencarian pada aplikasi desktop berbasis Free Pascal melalui penerapan Inverted Index yang dipadukan dengan Binary Search, menggunakan desain eksperimen kuantitatif komparatif pada lima ukuran dataset sintetis (100–10.000 baris), masing-masing diuji sepuluh kali dan dilaporkan dalam nilai rata-rata, minimum, maksimum, dan standar deviasi waktu eksekusi. Hasil pengujian menunjukkan peningkatan performa yang konsisten, dari 88,13% pada dataset 100 baris hingga 99,82% pada dataset 10.000 baris, sejalan dengan penurunan kompleksitas teoretis dari O(N) menjadi O(log K). Kebaruan penelitian ini terletak pada evaluasi kuantitatif langsung atas kombinasi Inverted Index dan Binary Search pada aplikasi desktop native berbasis Free Pascal, konteks yang belum banyak didokumentasikan dalam literatur information retrieval yang selama ini didominasi kajian pada sistem basis data server dan mesin pencari berskala besar. Penelitian ini memberikan kontribusi konseptual, empiris, dan praktis berupa bukti kuantitatif serta panduan implementasi bagi developer yang menghadapi kendala performa pencarian serupa tanpa bergantung pada mesin basis data eksternal.

Kata Kunci: Inverted Index; Binary Search; performa pencarian; Free Pascal; pengujian benchmark

Abstract

Search performance in desktop applications tends to decline as data volume grows, particularly when using a Linear Search with O(N) time complexity, since small-to-medium desktop applications typically lack the built-in indexing mechanisms found in modern database systems. This study optimizes search performance in a Free Pascal desktop application by combining an Inverted Index with Binary Search, using a quantitative comparative experimental design across five synthetic dataset sizes (100–10,000 rows), each tested ten times and reported as mean, minimum, maximum, and standard deviation execution time. Results show a consistent performance improvement, from 88.13% for the 100-row dataset to 99.82% for the 10,000-row dataset, consistent with the theoretical complexity reduction from O(N) to O(log K). The novelty of this study lies in the direct quantitative evaluation of the Inverted Index–Binary Search combination on a native Free Pascal desktop application, a context rarely documented in information retrieval literature dominated by studies on server-based databases and large-scale search engines. The study offers conceptual, empirical, and practical contributions, providing quantitative evidence and implementation guidance for developers facing similar search-performance constraints without relying on external database engines.

Keywords: Inverted Index; Binary Search; search performance; Free Pascal; benchmark testing

PENDAHULUAN

Aplikasi desktop berbasis Free Pascal masih banyak digunakan pada lingkungan bisnis kecil dan menengah karena ringan, cepat dieksekusi secara native, dan tidak memerlukan dependensi runtime yang berat. Salah satu fitur inti yang hampir selalu dibutuhkan pada aplikasi semacam ini adalah pencarian data, misalnya pencarian nama barang, pelanggan, atau transaksi. Pada implementasi sederhana, pencarian umumnya dilakukan dengan Linear Search, yaitu memeriksa setiap baris data satu per satu hingga ditemukan kecocokan. Pendekatan ini mudah diimplementasikan, namun berkompleksitas waktu O(N) sehingga performanya menurun signifikan seiring bertambahnya volume data (Amir, 2024). Persoalan ini menjadi lebih mendesak karena sebagian besar aplikasi desktop skala kecil-menengah, termasuk yang dibangun dengan Free Pascal, tidak memiliki mekanisme indexing bawaan sebagaimana tersedia pada sistem manajemen basis data (DBMS) modern, sementara migrasi ke basis data server sering kali tidak sebanding dengan skala kebutuhan aplikasi yang relatif kecil.

Permasalahan performa pencarian pada aplikasi desktop skala kecil-menengah relatif jarang dibahas secara mendalam dalam literatur information retrieval (IR), karena sebagian besar kajian optimasi pencarian data lebih berfokus pada sistem basis data relasional berskala besar, aplikasi berbasis web, atau mesin pencari IR berskala besar (Asrul dkk., 2023; Thakur dkk., 2021). Padahal, banyak aplikasi desktop, termasuk yang dikembangkan dengan Free Pascal, tidak selalu terhubung dengan mesin basis data yang menyediakan fitur indexing bawaan, sehingga penerapan teknik IR klasik seperti Inverted Index dan Binary Search sepenuhnya menjadi tanggung jawab pengembang pada level struktur data dan algoritma aplikasi itu sendiri. Kesenjangan (research gap) inilah yang mendasari penelitian ini: minimnya bukti kuantitatif yang mengukur secara langsung besaran peningkatan performa pencarian ketika teknik IR klasik tersebut diterapkan pada aplikasi desktop native, khususnya berbasis Free Pascal, sementara pengujian benchmark yang terstandardisasi justru berkembang pesat pada skala sistem IR besar (Thakur dkk., 2021).

Perlu ditegaskan, penelitian ini tidak mengklaim Inverted Index maupun Binary Search sebagai algoritma baru; keduanya adalah teknik information retrieval klasik yang telah mapan (Manning dkk., 2008). Kebaruan (novelty) penelitian ini terletak pada implementasi dan evaluasi kuantitatif langsung atas kombinasi kedua teknik tersebut pada konteks yang belum banyak didokumentasikan, yaitu aplikasi desktop native berbasis Free Pascal, melalui pengujian benchmark bertingkat pada berbagai ukuran dataset. Posisi ini berbeda dari penelitian sebelumnya yang membahas optimasi pencarian pada basis data server atau aplikasi berbasis web, maupun IR berskala besar yang berfokus pada mesin pencari (Lin dkk., 2019; Mallia dkk., 2019; Mackenzie dkk., 2021; Moffat & Mackenzie, 2023). Penelitian ini secara spesifik mengangkat konteks aplikasi desktop native berbasis Free Pascal yang tidak memanfaatkan mesin basis data pihak ketiga, sehingga seluruh mekanisme indexing dan pencarian diimplementasikan langsung pada level struktur data aplikasi menggunakan struktur bawaan bahasa seperti TStringList.

Berdasarkan uraian tersebut, rumusan masalah penelitian ini adalah: (1) bagaimana penerapan algoritma Inverted Index yang dipadukan dengan Binary Search sebagai solusi optimasi performa pencarian data pada aplikasi desktop berbasis Free Pascal; dan (2) seberapa besar

peningkatan performa pencarian yang dapat diukur secara kuantitatif melalui pengujian benchmark, dibandingkan dengan Linear Search pada berbagai ukuran dataset. Tujuan penelitian ini adalah mendeskripsikan tahapan teknis penerapan algoritma tersebut, menganalisis perbandingan kompleksitasnya secara teoretis, serta mengukur secara empiris kuantitatif waktu eksekusi pencarian pada kedua pendekatan menggunakan dataset sintetis yang dirancang untuk mengisolasi pengaruh algoritma dari variabel data produksi yang tidak terkontrol, seperti duplikasi kata atau distribusi frekuensi kata yang tidak merata.

Secara akademis, penelitian ini memperkaya literatur tentang penerapan teknik information retrieval pada lingkungan pemrograman desktop native yang masih jarang dikaji, sekaligus melengkapinya dengan bukti kuantitatif berbasis pengujian benchmark. Secara praktis, temuan ini mendokumentasikan strategi optimasi pencarian yang dapat dipertimbangkan oleh developer lain yang menghadapi permasalahan serupa tanpa bergantung pada mesin basis data eksternal. Secara spesifik, penelitian ini memberikan tiga kontribusi utama: (1) kontribusi konseptual berupa penjelasan sistematis mengenai penurunan kompleksitas pencarian dari O(N) menjadi O(log K) pada penerapan kombinasi Inverted Index dan Binary Search; (2) kontribusi empiris berupa kerangka pengujian benchmark kuantitatif yang mencakup desain eksperimen, dataset, dan parameter pengujian sehingga dapat direplikasi pada studi sejenis; dan (3) kontribusi praktis berupa dokumentasi teknis penerapan yang dapat menjadi bahan pertimbangan bagi pengembangan aplikasi desktop lain, khususnya yang menggunakan Free Pascal. Penelitian ini dibatasi pada eksperimen terhadap satu aplikasi desktop berbasis Free Pascal, dengan fokus pada algoritma pencarian data dan pengukuran performanya, tanpa membahas antarmuka pengguna, keamanan data, atau arsitektur perangkat lunak secara menyeluruh.

METODE

Penelitian ini menggunakan pendekatan kuantitatif dengan desain eksperimen komparatif (comparative experimental design) pada modul pencarian data, yang mengukur secara terkontrol perbedaan waktu eksekusi pencarian sebelum (Linear Search) dan sesudah (Inverted Index dipadukan Binary Search) penerapan optimasi pada berbagai ukuran dataset. Objek penelitian adalah modul pencarian data pada satu aplikasi desktop berbasis Free Pascal, dan seluruh pengukuran dilakukan pada lingkungan perangkat keras dan perangkat lunak yang seragam sebagaimana dirangkum pada Tabel 1, agar hasil antar skenario pengujian dapat dibandingkan secara konsisten.

a. Lingkungan dan Spesifikasi Pengujian

Pengujian performa dilakukan pada satu unit perangkat keras dan lingkungan pengembangan yang tetap agar hasil pengukuran waktu eksekusi antar skenario dapat dibandingkan secara konsisten. Spesifikasi perangkat keras, sistem operasi, compiler, serta metode pengukuran waktu yang digunakan dirangkum pada Tabel 1.

Tabel 1. Spesifikasi perangkat keras dan perangkat lunak pengujian.

Komponen	Spesifikasi
Prosesor (CPU)	Ryzen 5 7000 series
Memori (RAM)	16 GB
Penyimpanan	512 GB
Sistem Operasi	Windows 11
Compiler / IDE	Free Pascal Compiler (FPC) 3.2.2 / Lazarus 4.2

Mode kompilasi	ObjFPC ({$mode objfpc}{$H+})
Bahasa pemrograman	Object Pascal (Free Pascal)
Metode pengukuran waktu	Fungsi penghitung waktu presisi tinggi (mis. GetTickCount64 / TStopwatch), resolusi milidetik

b. Dataset Pengujian

Pengujian dilakukan pada beberapa variasi ukuran dataset untuk melihat konsistensi peningkatan performa seiring bertambahnya volume data. Dataset yang digunakan bersifat sintetis dan dibangkitkan secara otomatis, bukan diambil dari data produksi riil, dengan pertimbangan bahwa data sintetis memungkinkan kontrol penuh atas ukuran dataset dan estimasi jumlah kata unik pada setiap skenario pengujian, sehingga perbandingan waktu eksekusi antar ukuran dataset dapat dilakukan secara konsisten dan bebas dari variabel pengganggu seperti duplikasi kata atau distribusi frekuensi kata yang tidak merata pada data riil. Pendekatan ini memungkinkan isolasi pengaruh algoritma terhadap performa pencarian, sekalipun konsekuensinya, yaitu keterbatasan representativitas terhadap karakteristik data produksi, tetap diakui sebagai batasan penelitian (lihat bagian Keterbatasan Penelitian). Rincian ukuran dataset, estimasi jumlah kata unik, sumber data, dan format penyimpanan dirangkum pada Tabel 2.

Tabel 2. Komposisi dataset pengujian

Ukuran Dataset (baris)	Estimasi Kata Unik (K)	Sumber Data	Format Penyimpanan
100	±125	Data sintetis dibangkitkan otomatis	TStringList
500	±525	Data sintetis dibangkitkan otomatis	TStringList
1.000	±1.025	Data sintetis dibangkitkan otomatis	TStringList
5.000	±5.025	Data sintetis dibangkitkan otomatis	TStringList
10.000	±10.025	Data sintetis dibangkitkan otomatis	TStringList

c. Prosedur Benchmark

Prosedur pengujian benchmark dilakukan melalui lima tahap. Pertama, dataset pada setiap ukuran dimuat ke dalam aplikasi, kemudian dilakukan pembentukan indeks (tokenizing, case folding, dan penyimpanan ke struktur data terurut) sebelum pengujian pencarian dimulai. Kedua, pada Linear Search, waktu diukur dari pemeriksaan seluruh baris data hingga ditemukan kecocokan terhadap kata kunci uji. Ketiga, pada Inverted Index dan Binary Search, waktu diukur mulai dari proses Binary Search pada indeks kata hingga daftar ID dokumen relevan diperoleh, tanpa menyertakan waktu pembuatan indeks yang hanya dilakukan sekali di awal. Keempat, setiap skenario diulang sepuluh kali menggunakan kata kunci uji yang sama untuk memperoleh nilai rata-rata, minimum, maksimum, dan standar deviasi waktu eksekusi yang

representatif. Kelima, waktu eksekusi diukur menggunakan fungsi penghitung waktu presisi tinggi sebagaimana dirangkum pada Tabel 1, agar galat pengukuran akibat overhead sistem dapat diminimalkan.

d. Instrumen dan Teknik Analisis Data

Instrumen yang digunakan adalah instrumen pengujian benchmark berbasis perangkat lunak, yaitu rutin pengukuran waktu eksekusi (timer routine) yang disisipkan pada kode program untuk mencatat waktu proses pencarian pada kedua pendekatan di setiap ukuran dataset, mengikuti parameter pada Tabel 1 dan Tabel 2. Teknik analisis data dilakukan secara kuantitatif deskriptif-komparatif, membandingkan Linear Search dengan Inverted Index yang dipadukan Binary Search berdasarkan tiga dimensi: kompleksitas waktu (Big-O notation), struktur data yang digunakan, dan hasil pengukuran performa kuantitatif. Data waktu eksekusi dari sepuluh ulangan pada setiap ukuran dataset dan setiap pendekatan diringkas menggunakan statistik deskriptif berupa nilai rata-rata, minimum, maksimum, dan standar deviasi, kemudian dihitung persentase peningkatan performa antara kondisi sebelum dan sesudah optimasi.

HASIL DAN PEMBAHASAN

Hasil

a. Identifikasi Masalah pada Pendekatan Linear Search

Pada implementasi awal, pencarian data dilakukan dengan memeriksa setiap baris data menggunakan fungsi pencocokan string secara berurutan, seperti pola berikut.

```
for i := 1 to JumlahData
do if Pos(Kata, Data[i]) >
0 then
```

Pendekatan ini berkompleksitas waktu O(N), dengan N adalah jumlah data, sehingga waktu pencarian meningkat seiring bertambahnya volume data (lihat Tabel 4).

b. Penerapan Algoritma Inverted Index

Untuk mengatasi permasalahan tersebut, diterapkan algoritma Inverted Index, yaitu struktur indeks yang menyimpan setiap kata unik beserta daftar ID baris atau dokumen tempat kata tersebut muncul, sehingga pencarian tidak perlu membaca seluruh data setiap kali dilakukan. Pembentukan indeks meliputi empat langkah: (1) membaca seluruh data sumber; (2) tokenizing (memecah kalimat menjadi kata-kata); (3) case folding (mengubah kata menjadi huruf kecil agar pencarian tidak case-sensitive); dan (4) menyimpan hasilnya ke struktur data terurut, misalnya TStringList dengan properti Sorted := True dan Duplicates := dupIgnore, atau TFPGMap pada Free Pascal.

Sebagai ilustrasi, Tabel 3 menunjukkan contoh data sumber dan hasil pembentukan indeks dari empat baris data yang berisi nama produk.

Tabel 3. Contoh hasil pembentukan Inverted Index

Kata	ID Dokumen
Asus	1
Gaming	1, 3, 4
Keyboard	4
Laptop	1, 2
Lenovo	2
Logitech	3
mechanical	4
Mouse	3
Thinkpad	2

Karena kata pada indeks disimpan dalam kondisi terurut, proses pencarian selanjutnya dapat memanfaatkan algoritma Binary Search, yang jauh

lebih efisien dibandingkan pemeriksaan satu per satu.

c. Proses Pencarian dengan Binary Search

Ketika pengguna memasukkan kata kunci pencarian, misalnya "gaming", aplikasi tidak lagi membaca seluruh baris data. Kata kunci diubah menjadi huruf kecil, kemudian dicari melalui Binary Search pada daftar kata di dalam indeks, yang membagi ruang pencarian menjadi dua bagian secara berulang hingga kata ditemukan. Indeks kemudian langsung mengembalikan daftar ID dokumen yang relevan, sehingga aplikasi hanya menampilkan baris data pada ID tersebut tanpa memindai ulang seluruh dataset. Jumlah langkah pencarian menjadi jauh lebih sedikit dibandingkan pemeriksaan satu per satu, khususnya ketika jumlah kata unik pada indeks berjumlah besar.

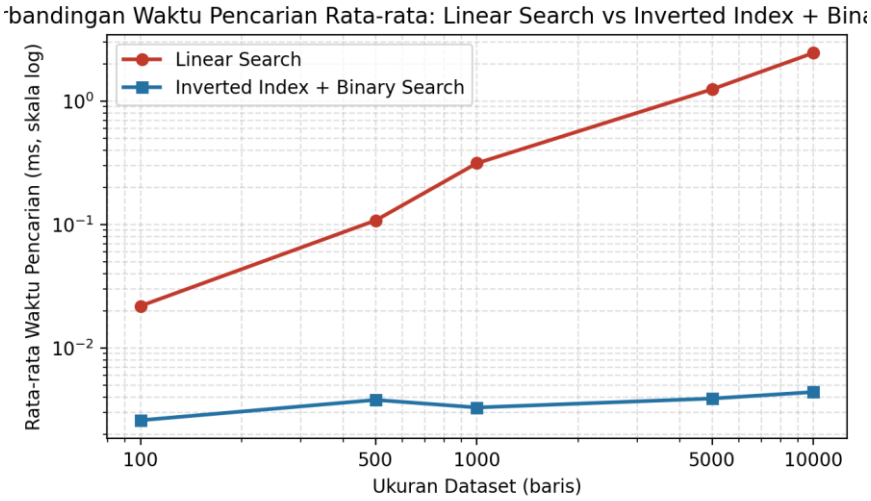
d. Hasil Pengujian Benchmark

Hasil pengujian benchmark waktu pencarian pada kedua pendekatan, untuk setiap ukuran dataset dengan sepuluh kali ulangan, dirangkum pada Tabel 4. Pada dataset 100 baris, rata-rata waktu pencarian Linear Search sebesar 0,0219 ms turun menjadi 0,0026 ms setelah penerapan Inverted Index dan Binary Search, atau setara peningkatan 88,13%. Selisih ini melebar secara konsisten seiring bertambahnya ukuran dataset: peningkatan sebesar 96,48% pada 500 baris, 98,95% pada 1.000 baris, 99,69% pada 5.000 baris, hingga 99,82% pada 10.000 baris. Rentang nilai minimum-maksimum dan standar deviasi pada Tabel 4 juga menunjukkan bahwa waktu eksekusi Inverted Index dan Binary Search jauh lebih stabil (standar deviasi maksimum 0,0006 ms) dibandingkan Linear Search yang variasinya melebar seiring bertambahnya volume data (standar deviasi hingga 0,2227 ms pada 10.000 baris).

Tabel 4. Hasil pengujian benchmark waktu pencarian, n = 10 ulangan

Ukuran Dataset	Kondisi	Rata-rata (ms)	Min (ms)	Maks (ms)	Std. Dev (ms)	Peningkatan
100	Sebelum (Linear Search)	0,0219	0,0211	0,0270	0,0017	-
100	Sesudah (Inverted Index + Binary Search)	0,0026	0,0025	0,0033	0,0002	88,13%
500	Sebelum (Linear Search)	0,1079	0,1068	0,1149	0,0024	-
500	Sesudah (Inverted Index + Binary Search)	0,0038	0,0029	0,0050	0,0006	96,48%
1.000	Sebelum (Linear Search)	0,3134	0,2383	0,3429	0,0341	-
1.000	Sesudah (Inverted Index + Binary Search)	0,0033	0,0031	0,0045	0,0004	98,95%
5.000	Sebelum (Linear Search)	1,2441	1,1357	1,5200	0,1351	-

5.000	Sesudah (Inverted Index + Binary Search)	0,0039	0,0037	0,0050	0,0004	99,69%
10.000	Sebelum (Linear Search)	2,4479	2,2304	2,8868	0,2227	-
10.000	Sesudah (Inverted Index + Binary Search)	0,0044	0,0042	0,0054	0,0003	99,82%



Gambar 1. Perbandingan waktu pencarian rata-rata Linear Search dan Inverted Index + Binary Search pada berbagai ukuran dataset (skala logaritmik), disusun dari data Tabel 4.

Pembahasan

Perbandingan kompleksitas antara pendekatan sebelum dan sesudah optimasi dirangkum pada Tabel 6.

Tabel 6. Perbandingan kompleksitas Linear Search dan Inverted Index + Binary Search

Aspek	**Sebelum Optimasi (Linear Search)**	**Setelah Optimasi (Inverted Index + Binary Search)**
Struktur data	Tidak ada indeks; data dibaca apa adanya	Indeks kata terurut (TStringList Sorted)
Kompleksitas pencarian	O(N)	O(log K)
Estimasi langkah (1 juta data/kata unik)	± 1.000.000 langkah	± 20 langkah
Waktu pembuatan struktur	Tidak diperlukan	O(N × M), dilakukan sekali saat data dimuat
Skalabilitas terhadap volume data	Menurun signifikan	Relatif stabil

Pembentukan indeks memiliki kompleksitas O(N × M), dengan N adalah jumlah dokumen atau baris data dan M adalah rata-rata jumlah kata per dokumen,

namun biaya ini hanya ditanggung sekali saat data dimuat atau diperbarui. Pencarian dengan Binary Search kemudian berkompleksitas O(log K), dengan K adalah jumlah kata unik pada indeks; sebagaimana diilustrasikan pada Tabel 6, pada estimasi satu juta kata unik hal ini setara dengan penurunan dari sekitar satu juta langkah pemeriksaan pada Linear Search menjadi hanya sekitar dua puluh langkah.

Temuan ini sejalan dengan teori kompleksitas algoritma klasik yang menyatakan bahwa struktur data terurut memungkinkan pencarian logaritmik, sebagaimana banyak dibahas dalam literatur information retrieval (Manning dkk., 2008). Perbedaan penelitian ini terletak pada konteks penerapannya, yaitu pada aplikasi desktop native berbasis Free Pascal yang tidak memanfaatkan mesin basis data pihak ketiga, sehingga seluruh mekanisme indexing dan pencarian diimplementasikan langsung pada level struktur data aplikasi. Dari sisi implementasi, penerapan pendekatan ini tidak memerlukan perubahan besar pada arsitektur aplikasi, karena struktur indeks dapat dibangun sebagai lapisan tambahan di atas struktur data yang sudah ada tanpa mengganti mekanisme penyimpanan data utama. Perlu dicatat, riset IR skala besar terus berkembang, baik melalui pengindeksan berbasis impact/learned sparse representation (Lin & Ma, 2021; Mallia dkk., 2021) maupun optimasi pemrosesan kueri pada Inverted Index untuk perangkat keras modern (Zhao dkk., 2025), namun Inverted Index eksak yang dipadukan Binary Search tetap proporsional untuk aplikasi desktop skala kecil-menengah tanpa kompleksitas implementasi tambahan.

Selain waktu eksekusi, penggunaan memori dan biaya pembentukan indeks turut relevan dipertimbangkan. Pengukuran profiler memori aktual (mis. FastMM) belum dilakukan pada penelitian ini, sehingga uraian berikut bersifat estimasi teoretis (lihat Keterbatasan Penelitian). Konsumsi memori Inverted Index diperkirakan tumbuh linear terhadap jumlah kata unik terindeks, yaitu O(K); mengacu Tabel 2, K berkisar ±125 (100 baris) hingga ±10.025 (10.000 baris), sehingga kebutuhan memori tambahan relatif menedibanding volume datanya—sejalan dengan kajian konstruksi dan pemeliharaan Inverted Index berskala besar (Mackenzie dkk., 2021; Moffat & Mackenzie, 2023). Trade-off yang dihasilkan: Inverted Index dan Binary Search unggul saat data besar dan sering dicari, karena biaya indeks teramortisasi oleh banyaknya pencarian cepat; sebaliknya Linear Search tetap lebih sesuai pada volume data kecil, data write-heavy, memori sangat terbatas, atau pencarian yang hanya sesekali dilakukan.

Tabel 5. Perbandingan penelitian ini dengan penelitian terdahulu

Penulis (Tahun)	**Metode & Konteks**	**Hasil Utama**	**Posisi terhadap Penelitian Ini**
Amir (2024)	Optimasi struktur data (Pascal), aplikasi desktop	Struktur data lebih efisien untuk aplikasi Pascal	Belum menyajikan pengujian benchmark kuantitatif berulang; penelitian ini melengkapi dengan pengujian benchmark bertingkat (5 ukuran dataset, n=10)
Asrul dkk. (2023)	Analisis kompleksitas algoritma, aplikasi native (bahasa umum)	Deskripsi kompleksitas algoritma	Kajian bersifat umum, tidak spesifik pada Free Pascal; penelitian ini spesifik pada

		pencarian pada aplikasi native	konteks Free Pascal dan struktur TStringList
Elizabeth (2024)	Sequential Search vs Binary Search, aplikasi pencarian data faktur	Binary Search lebih cepat dibanding Sequential Search pada data terurut	Tidak menggabungkan Binary Search dengan Inverted Index; penelitian ini memadukan keduanya untuk pencarian berbasis kata/token
Lin dkk. (2019); Mallia dkk. (2019)	Evaluasi & framework pengindeksan performant (C++)	Karakterisasi performa indexing dan kerangka pengindeksan efisien untuk riset IR skala besar	Fokus pada skala mesin pencari dan kompleksitas implementasi tinggi, bukan aplikasi bisnis kecil-menengah
Penelitian ini (2026)	Inverted Index + Binary Search (Free Pascal), aplikasi desktop bisnis kecil-menengah	Penurunan waktu pencarian 88,13%–99,82% tergantung ukuran dataset	Cakupan aplikasi tunggal; dataset benchmark bersifat sintetis (lihat Keterbatasan Penelitian)

Tabel 5 merangkum posisi penelitian ini terhadap penelitian-penelitian terdahulu yang membahas optimasi algoritma pencarian, baik pada konteks aplikasi native maupun sistem information retrieval berskala besar. Penelitian ini melengkapi kajian sebelumnya (Amir, 2024; Asrul dkk., 2023; Elizabeth, 2024) dengan pengujian benchmark bertingkat pada aplikasi Free Pascal, sekaligus menerapkan prinsip pengindeksan IR berskala besar (Lin dkk., 2019; Mallia dkk., 2019) pada konteks sumber daya yang jauh lebih terbatas.

KETERBATASAN PENELITIAN

Penelitian ini memiliki beberapa keterbatasan yang perlu diperhatikan dalam menginterpretasikan temuan. Pertama, penelitian ini menggunakan cakupan eksperimen tunggal (single-application experiment) pada satu aplikasi desktop berbasis Free Pascal, sehingga generalisasi temuan terhadap aplikasi lain dengan karakteristik data dan pola pencarian berbeda perlu dilakukan secara hati-hati. Kedua, pengujian benchmark dirancang menggunakan dataset sintetis (lihat Tabel 2), sehingga karakteristik data belum sepenuhnya merepresentasikan variasi data produksi yang sesungguhnya, seperti duplikasi kata, kesalahan ejaan, atau distribusi frekuensi kata yang tidak merata. Ketiga, pengujian dilakukan hanya pada satu unit perangkat keras dan satu konfigurasi compiler (Tabel 1), sehingga hasil pengukuran waktu eksekusi berpotensi berbeda pada spesifikasi perangkat keras atau versi compiler yang berbeda. Keempat, penelitian ini belum melakukan pengujian pada struktur indeks alternatif seperti Trie atau Hash-based Index, sehingga perbandingan performa antar struktur indeks lanjutan belum dapat disimpulkan. Kelima, penelitian ini belum melakukan uji statistik inferensial atas perbedaan waktu eksekusi antar kedua pendekatan, serta analisis penggunaan memori dan biaya pembentukan indeks masih bersifat estimasi teoretis berbasis kompleksitas O(K) dan O(N × M), karena belum didukung pengukuran profiler memori aktual, sehingga signifikansi statistik dan trade-off ruang-waktu antara Inverted Index dan Linear Search belum terukur secara empiris secara menyeluruh dan perlu divalidasi pada penelitian lanjutan.

SIMPULAN

Penelitian ini menunjukkan bahwa penurunan performa pencarian data pada aplikasi desktop berbasis Free Pascal, yang semula menggunakan Linear Search berkompleksitas O(N), dapat diatasi melalui penerapan algoritma Inverted Index yang dipadukan dengan Binary Search sehingga kompleksitas pencarian turun menjadi O(log K). Pengujian benchmark kuantitatif pada berbagai ukuran dataset mengonfirmasi konsistensi peningkatan performa tersebut sejalan dengan proyeksi teoretisnya (lihat Tabel 4 dan Tabel 6). Meskipun demikian, penerapan pendekatan ini menanggung biaya pembentukan indeks dan konsumsi memori tambahan, sehingga Linear Search tetap relevan pada kondisi data kecil, data yang sangat sering berubah, atau lingkungan dengan keterbatasan memori yang ketat.

Secara akademis, penelitian ini memberikan kontribusi berupa kerangka pengujian kuantitatif atas penerapan teknik information retrieval klasik pada lingkungan pemrograman desktop native yang masih relatif terbatas dibahas dalam literatur, khususnya pada aplikasi desktop berbasis Free Pascal. Implikasi praktisnya, temuan ini menunjukkan bahwa optimasi pencarian pada aplikasi desktop, khususnya berbasis Free Pascal, dapat dicapai tanpa bergantung pada mesin basis data eksternal, sehingga dapat dipertimbangkan sebagai salah satu opsi solusi oleh pengembang yang menghadapi kendala performa serupa.

Saran penelitian selanjutnya adalah mengkaji penerapan struktur indeks yang lebih kompleks, seperti Trie atau Hash-based Index, melengkapi pengujian benchmark dengan dataset produksi riil selain data sintetis, serta melakukan pengukuran penggunaan memori secara empiris guna memperkuat generalisasi temuan.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Program Studi Teknik Informatika, Universitas Muhammadiyah Parepare, atas dukungan fasilitas dan lingkungan akademik yang mendukung pelaksanaan penelitian ini. Penulis juga menyampaikan apresiasi kepada seluruh pihak yang telah memberikan masukan dan dukungan selama proses penelitian hingga penyusunan artikel ini.

DAFTAR PUSTAKA

Amir, R. (2024). Optimasi struktur data pada aplikasi desktop berbasis Pascal. Jurnal Teknik Informatika, 12(1), 45–56.

Asrul, M., dkk. (2023). Studi kompleksitas algoritma pencarian pada aplikasi native. Jurnal Sains dan Teknologi, 9(2), 88–97.

Elizabeth, T. (2024). Implementasi Algoritma Sequential Search dan Binary Search dalam Pencarian Data Faktur. Jurnal Teknik Informatika dan Sistem Informasi, 11(2).

Lin, J., & Ma, X. (2021). A few brief notes on DeepImpact, COIL, and a conceptual framework for information retrieval techniques. arXiv preprint arXiv:2106.14807.

Lin, J., Paniak, L., & Boerke, G. (2019). The performance envelope of inverted indexing on modern hardware. arXiv preprint arXiv:1910.11028.

Mackenzie, J., Petri, M., & Moffat, A. (2021). Anytime ranking on document-ordered indexes. ACM Transactions on Information Systems, 40(1), Article 13.

Mallia, A., Siedlaczek, M., Mackenzie, J., & Suel, T. (2019). PISA: Performant indexes and search for academia. In Proceedings of the Open-Source IR Replicability Challenge (OSIRRC) at SIGIR 2019 (pp. 50–56).

Mallia, A., Khattab, O., Suel, T., & Tonellotto, N. (2021). Learning passage impacts for inverted indexes. In Proceedings of the 44th International ACM SIGIR Conference

on Research and Development in Information Retrieval (pp. 1723-1727).

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge University Press.

Moffat, A., & Mackenzie, J. (2023). Efficient immediate-access dynamic indexing. Information Processing & Management, 60(3), 103248.

Thakur, N., Reimers, N., Rückle, A., Srivastava, A., & Gurevych, I. (2021). BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In 35th Conference on Neural Information Processing Systems (NeurIPS 2021) Datasets and Benchmarks Track.

Zhao, W., Zhao, M., Hu, H., & Qian, W. (2025). Columnar formatted inverted index for highly-paralleled, vectorized query processing. In 2025 IEEE 41st International Conference on Data Engineering (ICDE) (pp. 1800-1813). https://doi.org/10.1109/ICDE65448.2025.00138