ISSN (Online) 2747-0563
Volume 4 Tahun 2024

Pengembangan Algoritma *Sequential Search* pada Dataset Helaian Daun Tanaman

Zega Febrianto[1], Haris Yuana[2], Dimas Fanny Hebrasianto Permadi[3], Muhamad Azrino Gustalika[4*]

[1,2] Fakultas Teknologi Informasi Universitas Islam Blitar, Kota Blitar
[3,4] Teknik Informatika, Fakultas Informatika, Institut Teknologi Telkom Purwokerto
[1]randombox@gmail.com
[2]harisyuana2010@gmail.com
[3]dimas@ittelkom-pwt.ac.id
[4]azrino@ittelkom-pwt.ac.id
**Corresponding author email*: azrino@ittelkom-pwt.ac.id

***Abstrak*— Dalam proses pencarian, data berupa larik akan diperlakukan sebagai larik satu dimensi maupun dua dimensi serta akan diterapkan pencarian baik secara vertikal (pencarian tiap baris) maupun secara horizontal (pencarian tiap kolom). Terdapat tiga pengembangan algoritma Sequential Search, yaitu: Sequential Search dasar, Sequential Search pada kolom, dan Sequential Search pada kolom dengan indeks. Ketiga algoritma ini bekerja pada array dua dimensi dengan tingkat ketepatan 100%, namun memiliki waktu pencarian dan jumlah proses yang berbeda-beda. Sequential Search dasar menghasilkan waktu pencarian rata-rata 198,4 ms dengan 10 proses. Algoritma Sequential Search pada kolom menunjukkan peningkatan efisiensi dengan waktu pencarian rata-rata 198,3 ms dan hanya memerlukan 9 proses. Sementara itu, Sequential Search pada kolom dengan indeks menghasilkan waktu pencarian rata-rata 252,2 ms dengan 12 proses. Meski waktu pencarian pada algoritma terakhir lebih lama, ketiga metode tersebut menunjukkan kemampuan untuk melakukan pencarian data dengan akurat dalam array dua dimensi. Implementasi dari berbagai pengembangan Sequential Search ini memberikan fleksibilitas dalam memilih metode yang sesuai berdasarkan kebutuhan spesifik dari waktu dan efisiensi proses yang diinginkan. Analisis ini penting untuk memahami kelebihan dan kekurangan setiap metode pencarian dalam berbagai situasi, serta memberikan wawasan yang berguna untuk aplikasi nyata di bidang pemrosesan data dan analisis algoritma.**

Kata Kunci— **pencarian data, index kolom dan baris, sequential search, dataset, daun tanaman.**

I. PENDAHULUAN

Sequential atau *linear search* merupakan algoritma paling sederhana untuk menyelesaikan masalah terkait pencarian data. *Sequential searching* merupakan teknik pencarian data secara urut dari depan ke belakang atau dari awal sampai akhir berdasarkan *key* yang dicari dalam *array* 1 dimensi. Data yang akan dicari nanti akan ditelusuri dalam semua elemen-elemen *array* dari awal sampai akhir, dan data yang dicari tersebut tidak perlu diurutkan terlebih dahulu [1].

Pada penelitian ini, peneliti ingin melakukan modifikasi *sequential* atau *linear search* agar dapat dipergunakan pada struktur data yang berbeda, yaitu pada struktur data yang berupa *array* dua dimensi. Hal tersebut dilakukan atas dasar pertimbangan bahwa data tidak mungkin selalu ideal dan selalu cocok dengan sifat dasar dari suatu algoritma. Seperti diungkapkan pada paragraf sebelumnya, *Sequential Search* umumnya dilakukan pada *array* 1 dimensi, sedangkan pada umumnya, data seringkali berbentu *array* 2 dimensi, atau serupa dengan matriks, atau tabel.

Menurut Kamus Besar Bahasa Indonesia (KBBI) *index* atau indeks adalah daftar kata atau istilah yang dianggap penting yang terdapat dalam buku cetakan (biasanya pada bagian akhir buku) yang disusun menurut abjad dan memberikan informasi mengenai halaman tempat kata atau istilah itu ditentukan. Pengindeksan dilakukan agar sistem dapat melakukan pencarian berdasarkan *index* dimana data tersebut berada, sehingga sistem tidak harus melakukan pencarian hingga elemen ke-n atau dengan kata lain sistem harus melakukan pencarian hingga semua elemen telah ditelusuri. Kode ASCII dari karakter pertama dari suatu string menjadi penentu dimana algoritma harus tetap berjalan ataupun berhenti, atau dapat disebut sebagai batas dimana algoritma akan melakukan pencarian. Hal tersebut diharapkan dapat meningkatkan efisiensi waktu dari proses pencarian yang dijalankan oleh sistem. Pada penelitian ini, peneliti juga akan tetap menyertakan algoritma *Sequential* dan *linear search* pada larik satu dimensi sebagai pembanding efisiensi waktu daripada algoritma hasil modifikasi.

Dengan jumlah *dataset* yang relatif banyak, serta kondisi dari dataset yang tidak dalam keadaan urut serta tidak memungkinkan untuk diurutkan secara keseluruhan maka algoritma yang diterapkan harus disesuaikan dengan keadaan tersebut. Penyesuaian pertama adalah, algoritma *Sequential* atau *linear search* akan digunakan untuk melakukan pencarian pada *array* dua dimensi. *Array* 2 dimensi sering dianalogikan dan digambarkan sebagai bentuk matriks, dengan indeks pertama berfungsi sebagai baris dan indeks kedua digunakan untuk kolom, atau biasanya dapat juga disebut dan mempunyai kemiripan dengan tabel [2]. Selanjutnya, penyesuaian kedua adalah penambahan pengindeksan pada proses algoritma tersebut.

Pada proses pencarian, data berupa larik tersebut akan diperlakukan sebagai larik satu dimensi maupun dua dimensi serta akan diterapkan pencarian baik secara vertikal (pencarian tiap baris) maupun secara horizontal (pencarian tiap kolom). Dengan bantuan algoritma *Sequential search* dan modifikasinya serta tipe data berupa *array* dua dimensi diharapkan dapat mempermudah dan meningkatkan kecepatan dari proses pencarian. Serta dari hasil yang didapat, peneliti

ISSN (Online) 2747-0563
Volume 4 Tahun 2024

mampu mengetahui algoritma dan struktur data mana yang dapat melakukan komputasi tercepat dengan kondisi data dengan struktur dan jumlah yang sama.

II. TINJAUAN PUSTAKA

A. Helaian Daun

Tabel I hingga Tabel X merupakan kategori-kategori dari penyusun helaian daun tanaman. Dataset helaian daun tanaman disusun dengan merujuk kepada Tabel I hingga Tabel X dengan pertimbangan kemiripan ciri helaian daun dengan keterangan yang terdapat pada Tabel I.

TABEL I
HELAI DAUN

Helai Daun
Bulat atau bundar (*orbicularis*)
Bulat telur (*ovatus*)
Bulat telur sungsang (*obovatus*)
Delta (*deltoid*)
Garis (*linearis*)
Ginjal (*reniformis*)
Jarum (*acerosus* atau *acicular*)
Jantung (*cordatus*)
Jantung sungsang (*obcordatus*)
Jorong (*ovalis* atau *ellipticus*)
Lanset (*lanceolatus*)
Memanjang (*oblongus*)
Paku atau dabus (*subulatus*)
Pedang (*ensiformis*)
Perisai (*peltatus*)
Pita (*ligulatus*)
Segitiga (*triangularis*)
Segitiga terbalik atau pasak (*cuneatus*)

TABEL II
UJUNG DAUN

Ujung Daun
Runcing (*acutus*), bila kedua tepi kiri dan kanan ibu tulang daun bertemu membentuk sudut lancip.
Meruncing (*acuminatus*), seperti pada bentuk runcing tetapi pertemuan tepi daun lebih panjang ke depan.
Tumpul (*obtusus*)
Membulat (*rotundatus*), bila ujung daun membentuk seperti busur
Rompang (*truncatus*), bila ujung daun membentuk garis rata
Terbelah (*retusus*), bila ujung daun membentuk lekukan
Berlekuk (emarginatus)
Berduri (*mucronatus*), bila ujung daun merupakan suatu duri

TABEL III
PANGKAL DAUN

Pangkal Daun
Berlekuk (*emarginatus*)
Membulat (*rotundatus*)
Runcing (*acutus*)
Rompang (*truncatus*), pangkal daun rata

Pangkal Daun
Meruncing (*acuminatus*), bentuknya seperti runcing (acutus) tetapi lebih tajam.
Tumpul (*obtusus*)
Miring atau asimetri (*oblique*)

TABEL IV
TULANG DAUN

Tulang daun
Menyirip (*penninervis*)
Menjari (*palminervis*)
Melengkung (*cervinervis*)
Sejajar (*rectinervis*)

TABEL V
PERMUKAAN DAUN

Permukaan Daun
Berbingkul-bingkul (*bullatus*). Berkerut tetapi kerutannya lebih besar
Berbulu (*pilosus*)
Berbulu halus dan rapat (*villosus*)
Berbulu kasar (*hispidus*)
Berkerut (*rugosus*)
Bersisik (*Lepidus*)
Gundul (*glaber*)
Kasap (*scaber*)
Licin (*laevis*) - Mengkilat (*nitidus*)
Licin (*laevis*) - Suram (*opacus*)
Licin (*laevis*) - Berselaput lilin (*pruinosus*)

TABEL VI
TEPI DAUN

Tepi daun
Rata (Integer)
Bertoreh (Divisus)

TABEL VII
TEPI DAUN BERTOREH

Tepi daun bertoreh
Torehan yang tidak mempengaruhi bentuk asli daun (toreh yang merdeka)
Torehan yang amat dalam sehingga mengubah bentuk daun

TABEL VIII
TOREHAN YANG TIDAK MEMPENGARUHI BANGUN DAUN

Torehan yang tidak mempengaruhi bangun daun
Bergerigi (*serratus*), apabila sinus dan angulus sama lancipnya
Bergerigi ganda/rangkap (*biserratus*), apabila angulus cukup besar dan tepinya bergerigi lagi
Bergigi (*dentatus*), apabila sinus tumpul dan angulus lancip
Beringgit (*crenatus*), apabila sinus tajam dan angulus tumpul
Berombak (*repandus*), apabila sinus dan angulusnya sama-sama tumpul
Berlekuk (*lobatus*), tepi daun memiliki lekukan tetapi tidak sampai ke tengah daun
Undulate, tepi daun berombak tetapi lebih dangkal

ISSN (Online) 2747-0563
Volume 4 Tahun 2024

TABEL IX
TOREHAN YANG MEMPENGARUHI BANGUN DAUN

Torehan yang mempengaruhi bangun daun
Berlekuk (lobatus), jika kedalaman toreh kurang daripada setengah panjang tulang-tulang yang terdapat di kiri kanannya.
Bercangap (fissus), jika kedalaman toreh kurang lebih sampai ditengah panjang tulang-tulang daun di kiri kanannya.
Berbagi (partitus), jika kedalaman toreh melebihi setengah panjang tulang daun di kanan kirinya.

TABEL X
JUMLAH HELAI DAUN

Jumlah helai daun
Daun tunggal, bila pada tangkai daun memiliki satu helai daun saja
Daun majemuk, bila tangkai daun bercabang-cabang dan masing-masing memiliki helaian daun.

B. Algoritma Sequantial Search

Sequential searching merupakan teknik pencarian data secara urut dari depan ke belakang atau dari awal sampai akhir berdasarkan key yang dicari dalam array 1 dimensi. Data yang akan dicari nanti akan ditelusuri dalam semua elemen-elemen array dari awal sampai akhir, dan data yang dicari tersebut tidak perlu diurutkan terlebih dahulu apabila sampai akhir pengulangan tidak ada data yang sama, berarti data yang dimaksud tidak ada [1].

Proses yang terjadi pada metode pencarian Sequential Search adalah sebagai berikut:

1. Membaca array data
2. Menentukan data yang dicari
3. Mulai dari data pertama sampai dengan data terakhir, data yang dicari dibandingkan dengan masing-masing data di dalam array. Jika data yang dicari tidak ditemukan maka semua data atau elemen array dibandingkan sampai selesai. Jika data yang dicari ditemukan maka perbandingan akan dihentikan [3]

Contoh pencarian data Sequential Search :
Data : 12, 15, 7, 9, 1, 20
Data yang dicari : 7
Proses Sequential Search :

1. Pencarian dilakukan dari data pertama sampai data terakhir
2. Bandingkan data ke-1 dengan data yang dicari.
   12 == 7 → (False. Bukan data yang dicari)
3. Bandingkan data ke-2 dengan data yang dicari.
   15 == 7 → (False. Bukan data yang dicari)
4. Bandingkan data ke-3 dengan data yang dicari.
   7 == 7 → (True. Data yang dicari ditemukan)
5. Proses pencarian berhenti [4]

Perbandingan antara *linear search/sequential search* dengan *binary search* [5]. Hasil dari perbandingannya dapat dilihat pada Tabel XI

TABEL XI
PERBANDINGAN SEQUENTIAL SEARH DAN BINARY SEARCH

Dasar perbandingan	*Linear/sequential search*	*Binary search*
Kompleksitas waktu	O(n)	O(log2n)
Waktu kasus terbaik	Element pertama O(1)	Element tengah
Prasyarat untuk *array*	Tidak dibutuhkan	*Array* harus diurutkan
Kasus terburuk untuk n jumlah elemen	Membutuhkan perbandingan sejumlah N	Dapat disimpulkan setelah (logN)
Dapat diimplementasikan pada	*Array* dan *linked list*	Tidak dapat di implementasikan langsung ke *linked list*
Tipe algoritma	Bersifat perulangan	Bersifat *divide and conquer*
Operasi Insert	Dapat dilakukan *insert* di akhir deret	Memerlukan pemrosesan untuk melakukan *insert* di tempat yang tepat untuk mempertahankan daftar yang diurutkan
Penggunaan	Mudah digunakan	Algoritma yang rumit
Baris kode	Sedikit	Lebih banyak

C. Indeksi

Index adalah bahasa yang biasanya dipergunakan di dalam sebuah buku untuk mencari informasi berdasarkan kata kunci yang mengarah ke dalam suatu halaman [6]. Dalam Kamus Besar Bahasa Indonesia, indeks diartikan sebagai daftar kata atau istilah penting yang terdapat dalam buku cetakan (biasanya pada bagian akhir buku) tersusun menurut abjad yang memberikan informasi mengenai halaman tempat kata atau istilah itu ditemukan. Indeks tidak sekedar melengkapi suatu penerbitan seperti apa yang dikenal masyarakat pada umumnya. Akan tetapi, indeks telah berkembang menjadi suatu sistem tatanan informasi tersendiri. Dalam tatanan tersebut dikenal istilah mengindeks, yaitu suatu proses mendeskripsikan suatu dokumen atau informasi ke dalam suatu bentuk tertentu dengan tujuan agar semua artikel atau dokumen yang telah diolah dan disimpan dapat ditemukan kembali [7]. Sedangkan pada array, indeks *array* seharusnya adalah suatu tipe yang juga mempunyai keterurutan (memiliki suksesor dan ada predesesor), misalnya tipe integer atau karakter. Jika indeksnya adalah integer maka keterurutan indeks sesuai dengan urutan integer. Jika indeks array adalah karakter maka keterurutan indeks sesuai dengan urutan karakter [2].

D. Pre Processing Data

Dokumen pada umumnya mempunyai struktur yang sembarangan atau tidak terstruktur. Oleh karena itu, diperlukan suatu proses yang dapat mengubah bentuk data yang sebelumnya tidak terstruktur ke dalam bentuk data yang

ISSN (Online) 2747-0563
Volume 4 Tahun 2024

terstruktur. Proses pengubahan ini dikenal dengan istilah text *preprocessing* [8]. Tahapan *preprocessing* data pada penelitian ini terletak pada proses konversi dari data berupa json menjadi data dengan tipe *array* atau larik, serta proses konversi beberapa *string* yang tidak *lowercase* menjadi *lowercase* secara keseluruhan. Mengubah string menjadi lowercase diperlukan ketika melakukan perbandingan antara dua *string* dikarenakan huruf kapital atau *uppercase* nilai ASCII nya berbeda dengan huruf kecil atau *lowercase*.

E. Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah proses untuk mencari kesalahan pada setiap item perangkat lunak, mencatat hasilnya, mengevaluasi setiap aspek pada setiap komponen (sistem) dan mengevaluasi fasilitas-fasilitas dari perangkat lunak yang akan dikembangkan [9]. Jenis pengujian dalam penelitian ini menggunakan metode *whitebox*. Pengujian metode *whitebox* dikenal dengan pengujian yang terstruktur, pengujian transparent box, pengujian berdasarkan logika atau pengujian berdasarkan kode [10]. Pengujian unit atau unit testing yang akan menjadi metode pengujian utama pada penelitian ini termasuk dalam kategori pengujian dengan metode *whitebox*. Unit testing adalah teknik untuk menguji apakah kode program perangkat lunak sudah efektif dan bebas dari kesalahan. Pada beberapa kasus pengembangan perangkat lunak, sering terjadi bahwa kode program yang dikembangkan ternyata kurang efektif atau bahkan tidak pernah digunakan [11]. Pengujian unit pada penelitian ini diperlukan untuk menguji apakah masing-masing modul dari program dapat berjalan sebagai mana mestinya, serta luaran yang dihasilkan oleh modul bernilai valid ketika akan digunakan sebagai masukan untuk modul-modul lain. Kebenaran nilai dari tiap-tiap modul dapat berpengaruh dari luaran terakhir atau hasil akhir dari sistem atau program secara keseluruhan

F. Kode ASCII

Kode Standar Amerika untuk pertukaran informasi atau ASCII (American Standard Code for Information Interchange) merupakan suatu standar internasional dalam kode huruf dan 397ahasa seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|". Kode ini selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks [12]. Kode ASCII digunakan sebagai pembanding apakah suatu string bernilai lebih besar atau lebih kecil dari string pembanding. Perbandingan ini tidak diterapkan pada semua karakter dari suatu string namun hanya diterapkan pada karakter pertama dari suatu string

G. Bahasa Pemrograman PHP

PHP (akronim rekursif untuk PHP: Hypertext Preprocessor) adalah bahasa skrip untuk keperluan umum open source yang banyak digunakan dan sangat cocok untuk pengembangan web dan dapat disematkan ke dalam HTML

H. JSON (Javascript Object Notation)

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999. JSON adalah format teks yang tidak bergantung pada 397ahasa pemprograman apapun sehingga dapat digunakan di Bahasa pemrograman C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON sangat ideal sebagai bahasa pertukaran data [13]. Karena setiap objek JSON dalam dokumen JSON adalah kumpulan pasangan Key-Value, dokumen JSON secara alami direpresentasikan sebagai struktur pohon data yang disebut JSON tree. Suatu nilai dapat berupa nilai atomik seperti string, integer, angka, array, atau null. Untuk menangkap struktur komposisi data JSON, setiap nilai dapat menjadi satu set objek JSON lain [14].

Pada penelitian ini, file dengan tipe .json digunakan sebagai struktur data utama yang akan digunakan pada proses pencarian. Dengan strukturnya yang mirip dengan hasil *query* dari *database* pada umumnya, maka *programmer* sudah tidak asing lagi dengan tipe data .json ini. JSON terbentuk dari dua struktur diantaranya :

1. *Object* dimana data disimpan dengan pasangan name:*value*. *Object* dimulai dengan kurung kurawal buka ({) dan diakhiri dengan kurung kurawal tutup (}). Setiap nama diikuti dengan titik dua (:) dan setiap pasangan nama dan nilai dipisahkan oleh tanda koma (,) [6].
2. *Array* adalah kumpulan data yang diserialisasikan di satu tempat. *Array* dimulai dengan kurung siku buka ([) dan diakhiri dengan kurung kotak tutup (]), dan setiap nilai dipisahkan oleh tanda koma (,) [6].

I. Array

Array atau larik adalah tipe terstruktur yang terdiri dari sejumlah komponen-komponen yang mempunyai tipe yang sama. Menurut definisinya, array adalah sebuah variabel yang dapat menyimpan lebih dari satu nilai sejenis dengan memiliki tipe data yang sama. Suatu array mempunyai jumlah komponen yang banyaknya tetap. Banyaknya komponen dalam suatu ditunjukan oleh suatu indeks yang disebut tipe indeks [13]. Pada penelitian ini, tipe data array digunakan untuk mempermudah proses ketika program ingin mengakses data-data tertentu atau mengambil nilai tertentu yang terletak pada suatu indeks, baik indeks berupa angka atau indeks asosiatif

III. METODE PENELITIAN

a. Pengumpulan Data Uji

Data uji pada penelitian ini menggunakan dataset helaian daun tanaman, serta data ini bukan merupakan data yang valid karena fungsinya hanya digunakan sebagai data uji. Data uji

ISSN (Online) 2747-0563
Volume 4 Tahun 2024

merupakan sekumpulan data yang disimpan pada *Microsoft Excel* atau aplikasi pengolahan data lainnya

TABEL XII
CONTOH DATASET HELAIAN DAUN TANAMAN

nama	**helai_daun**	**ujung_daun**	**pangkal_daun**	**tulang_daun**
mangga	jorong	tumpul	tumpul	menyirip
jambu biji	bulat telur	membulat	tumpul	menyirip
kelengkeng	jorong	tumpul	tumpul	menyirip
alpukat	bulat telur	tumpul	runcing	menyirip
rambutan	jorong	tumpul	tumpul	menyirip
sawo	jorong	tumpul	tumpul	menyirip
anggur	jantung	runcing	bertelinga	menjari

Tabel XII berisikan contoh dari *dataset* yang digunakan pada penelitian ini. *Dataset* yang akan digunakan pada penelitian akan diubah terlebih dahulu menjadi .csv atau *comma-separated value*, setelah itu data diubah menjadi .json.
Proses implementasi algoritma *Sequential Search* beserta alur proses pencarian menjadi aplikasi yang dibuat menggunakan Bahasa pemrograman PHP
Melakukan pengujian waktu komputasi, validitas hasil pencarian, melakukan pengujian unit atau unit testing untuk tiap-tiap modul penerapan algoritma, dan kompleksitas alur pemrograman

b. Perancangan Sistem

Penelitian ini menggunakan metode perancangan model *prototyping*. Model *prototype* adalah proses pembuatan perangkat lunak sebelum atau selama tahap penyusunan persyaratan hingga aplikasi sampai pada produk akhir. Calon pengguna bekerja dengan *prototype* ini mengidentifikasi kelebihan dan kekurangan dari hasil yang sedang dibuat. Hasil dari proses identifikasi dilaporkan ke pengembang produk perangkat lunak. Dengan demikian, umpan balik diberikan antara pengguna dan pengembang, yang digunakan untuk berubah atau perbaiki spesifikasi persyaratan untuk produk perangkat lunak [15]. Singkatnya, program akan dikembangkan secara berkelanjutan mulai dari pengembangan, evaluasi dan revisi setelah evaluasi. Proses tersebut akan terus dilakukan hingga semua kebutuhan dari program tersebut sudah tercapai

c. Objek Penelitian

Algoritma *Sequential Search* merupakan algoritma dasar sistem pencarian pada penelitian ini. Pada penelitian ini juga melakukan pengembangan dari Algoritma *Sequential Search* dengan menambahkan proses pengindeksan pada kolom data di sistem pencarian ini

d. Proses Pengolahan Data

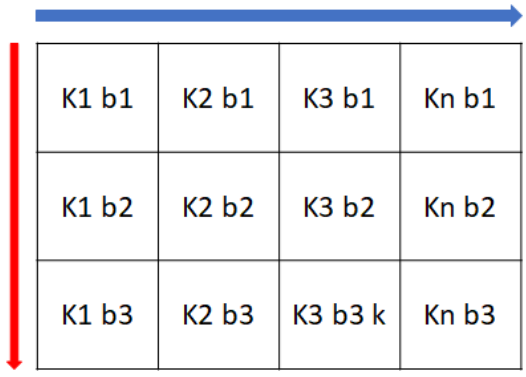
Proses pengumpulan data pada penelitian ini akan dibagi menjadi beberapa tahap, yaitu :

1. Melakukan pendataan data testing pada aplikasi *Microsoft excel*. Kriteria yang wajib dipenuhi antara lain :
   a. Nama kolom wajib *lowercase* atau huruf kecil
   b. Nama kolom tidak mengandung spasi namun dapat diganti dengan garis bawah atau "_".
2. Data uji yang telah mencapai batas yang ditentukan lalu disimpan dalam bentuk .csv. String dibawah ini merupakan salah satu contoh isi dari file berformat .csv dengan koma sebagai pembatas dari tiap nilai dari satu baris data. Baris data pertama dari file tersebut merupakan *table header* dari suatu *dataset*. *Table header* tersebut dapat digunakan sebagai *key* dari suatu *array* ketika .csv sudah dirubah menjadi *array*
   Contoh baris file csv :
   1. mangga,jorong,tumpul,tumpul,menyirip,gundul,integer,0,0,0,tunggal,hijau tua
   2. Contoh table header dari file .csv
   3. nama,helai_daun,ujung_daun,pangkal_daun,tulang_daun,permkaan_daun,tepi_daun,tepi_daun_bertoreh,torehan_Y,torehan_N,jumlah_helai_daun,warna_daun
   4. Lalu ubah menjadi dalam bentuk ,json untuk selanjutnya dapat diproses oleh system

IV. HASIL DAN PEMBAHASAN

a. Sequential Search

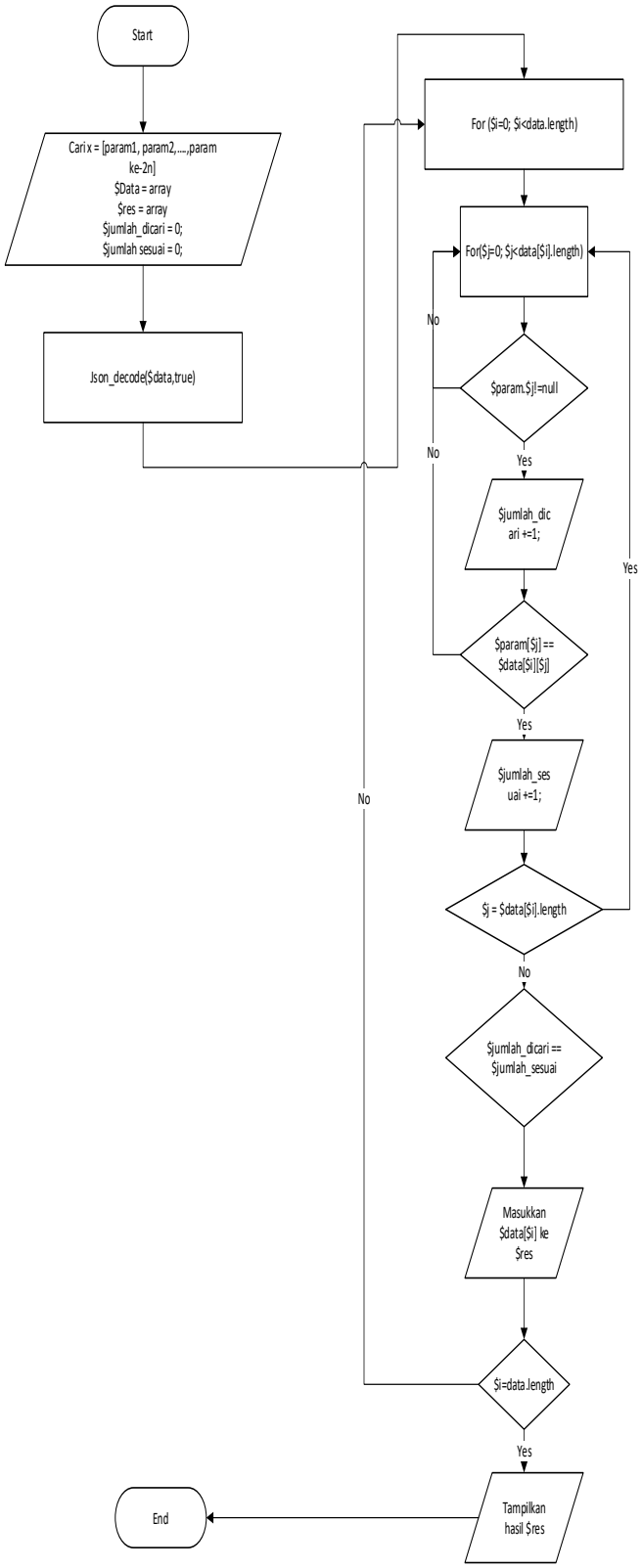
Seperti sequential search pada umumnya, proses pencarian dilakukan pada satu deret yang berisi beberapa kolom dari suatu baris data. Proses pencarian yang dilakukan memiliki arah horizontal, yaitu membandingkan antara *value* dari *index* yang sedang berjalan dengan parameter pencarian. Pola pencarian yang berjalan pada algoritma *Sequential Search* pada Gbr. 1.



Gbr. 1 ilustrasi proses pencarian pada Sequential Search pada 2 dimensi

Pada Gbr. 1, simbol panah biru melambangkan proses dilakukan terlebih dahulum sedangkan simbol panah merah melambangkan proses dilakukan setelah panah biru selesai dilakukan. Dengan *k* mewakili kolom, dan *b* mewakili baris

ISSN (Online) 2747-0563
Volume 4 Tahun 2024



Gbr. 2 Diagram Alur *Sequential Search* pada pencarian tiap baris

Implemetasi tiap tahap dari algoritma *Sequential Search* yang diterapkan pada 2 dimensi yang telah digambarkan pada Gbr. 2:

1. Mengambil data secara keseluruhan

Pada proses ini, data yang awalnya berbentuk json akan diubah strukturnya menjadi *array*.

2. Mengambil nama kolom dari data

Pada proses ini, kita akan membuat *variable* yang berisi *key* dari suatu baris data. *Key* tersesbut akan digunakan sebagai parameter pencarian pada proses selanjutnya.

3. Mendefinisikan *array* untuk menyimpan hasil pencarian
4. *Variable* dengan tipe data *array* diatas akan digunakan untuk menyimpan hasil pencarian. Data akan diisikan kedalam *array* secara berulang–ulang hingga seluruh data telah berhasil dibaca dan diseleksi/dicari.
5. Melakukan perulangan sebagai proses pencarian tiap baris dari suatu dataset
6. Menampilkan hasil pencarian

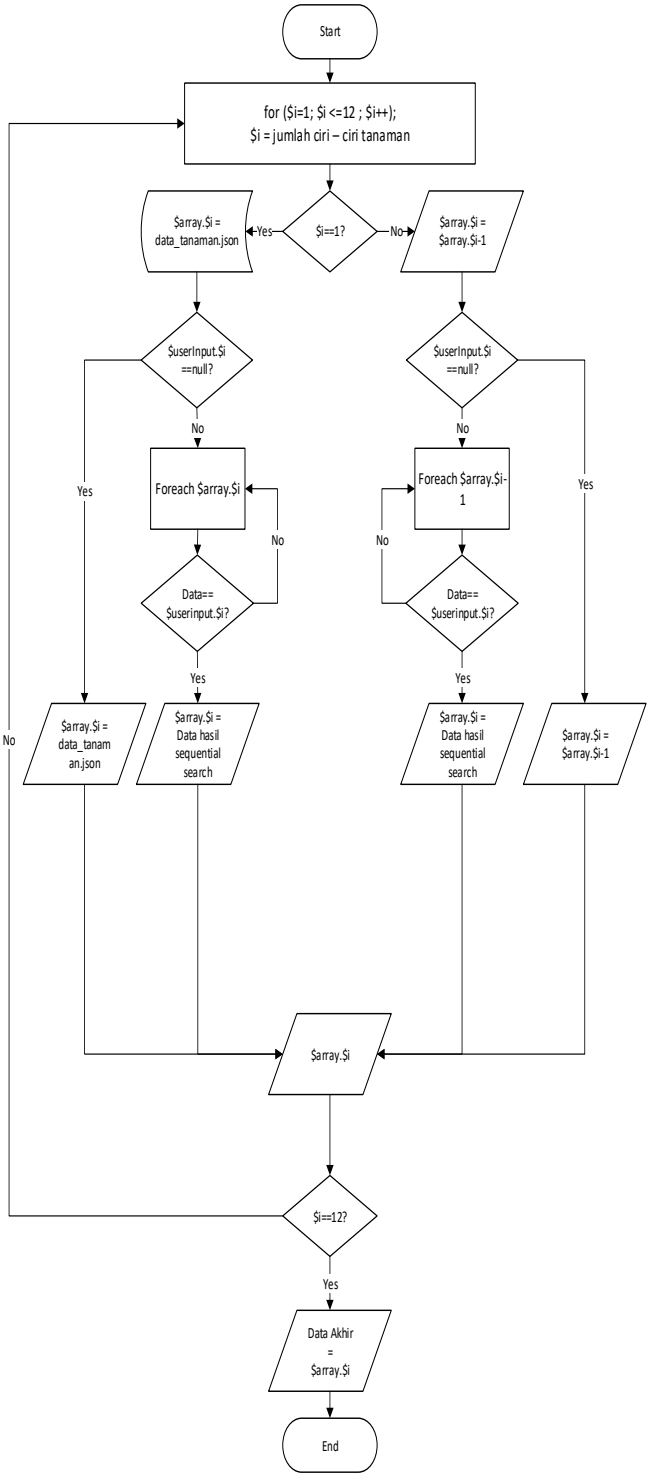
b. Sequntial Search pada Kolom

Sequential Search pada kolom ini dimodifikasi dengan memulai pencarian secara vertikal. Hal ini sedikit berbeda dengan yang *Sequential Search* dasar yang pencariannya dengan alur secara horizontal. Sehingga ilustrasi proses jalannya modifikasi algoritma ini dapat dilihat pada Gbr. 4

Gbr 3 Ilustrasi jalannya proses pencarian pada *sequential search* yang diterapkan pada kolom dari 2 dimensi

Pada Gbr 3 simbol panah biru melambangkan proses dilakukan terlebih dahulum sedangkan simbol panah merah melambangkan proses dilakukan setelah panah biru selesai dilakukan. Dengan *k* mewakili kolom, dan *b* mewakili baris.

ISSN (Online) 2747-0563
Volume 4 Tahun 2024



Gbr. 4 Diagram Alur Sequential Search pada pencarian tiap kolom

Implemetasi tiap tahap pada penerapan algoritma *Sequential Search* pada kolom yang diterapkan pada *array* 2 dimensi yang digambarkan pada Gbr. 4:

1. Mengambil data secara keseluruhan
2. Mengambil nama kolom dari data pertama
3. Melakukan perulangan for untuk melakukan pencarian berdasarkan kolom. Pencarian dilakukan dari kolom pertama hingga kolom ke–*n*, *n* adalah jumlah kolom dari satu baris data.
4. Mendefinisikan *array* untuk proses ke-*n*
5. Data yang digunakan pada proses ke-1 adalah data yang diambil dari poin 1. Selebihnya data yang digunakan pada proses ke-2 hingga akhir adalah data hasil dari proses ke-(*n*-1).
6. Perkondisan untuk menentukkan nilai parameter pencarian hasil dengan penjelasan sebagai berikut :
   a. Perkondisian *if* untuk menentukan apakah parameter pencarian tidak kosong atau tidak sama dengan *null*. Jika parameter pencarian bernilai *null*, maka hasil dari proses ke-n adalah data ke-(*n*-1).
   b. Jika parameter pencarian tidak sama dengan *null*, maka terjadilah proses pencarian semua baris dari kolom ke-n
   c. Melakukan *foreach* semua kolom ke-*n*
   d. Melakukan perbandingan nilai antara nilai dari kolom dengan parameter pencarian menggunakan sintaks php *stripos*. *Stripos* - Temukan posisi kemunculan pertama dari *substring case-insensitive* dalam sebuah *string* (php.net). Sintaks *stripos* digunakan untuk mencocokan pola suatu *string* dengan *string* utama. Apabila terdapat kecocokan pola, maka kembalian dari perkondisian tersebut adalah indeks dimana pola itu berada, ketika tidak terdapat kecocokan pola maka data yang dicari tidak cocok
   e. Jika sesuai, *array* akan menyimpan satu baris data kedalam *variable* $temp, yang setelahnya akan disimpan pada *variable*

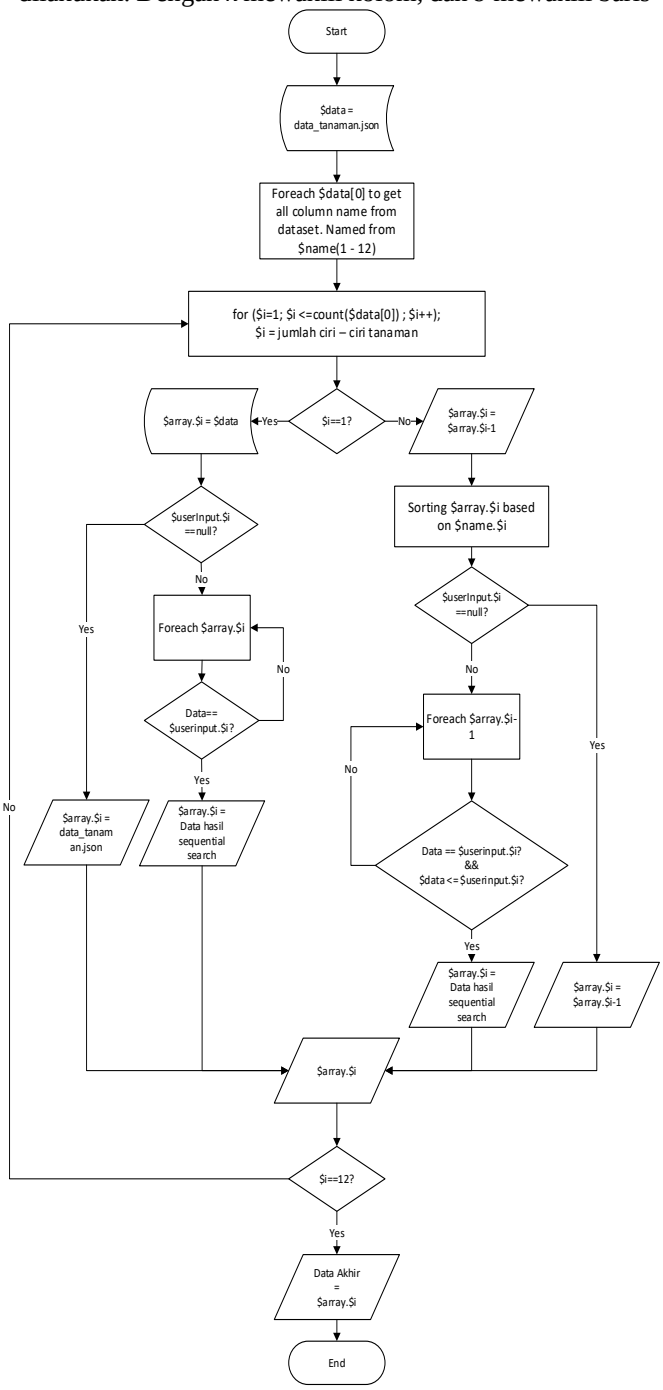
c. Sequential Search pada Kolom dengan Indeks

Konsep dari Sequential Search pada pencarian tiap kolom dengan pengindeksan adalah sama dengan *Sequential Search* pada kolom namun sebelum digunakan, data akan terlebih dahulu diproses yaitu diurutkan berdasarkan *index* ke-*i* atau diurutkan berdasarkan nama kolom yang sedang berjalan pada proses tersebut.

Gbr. 5 Ilustrasi jalannya proses pencarian pada *sequential search* yang diterapkan pada kolom dari 2 dimensi

ISSN (Online) 2747-0563
Volume 4 Tahun 2024

Pada Gbr. 5, simbol panah biru melambangkan proses dilakukan terlebih dahulum sedangkan simbol panah merah melambangkan proses dilakukan setelah panah biru selesai dilakukan. Dengan *k* mewakili kolom, dan *b* mewakili baris



Gbr. 6 Diagram Alur *Sequential Search* pada pencarian tiap kolom dengan indeks

Implemetasi tiap tahap pada penerapan algoritma *Sequential search* pada kolom dengan indeks yang digambarkan pada Gbr. 6:

1. Mengambil data secara keseluruhan dan mengubah struktur data tersebut menjadi *array*
2. Mengambil nama kolom dan menyimpannya dalam *variable*
3. Melakukan perulangan sebanyak kolom yang terdapat pada proses pada poin 2
4. Setiap iterasi $a, data keseluruhan adalah data yang diambil dari proses sebelumnya, namun pada proses ke-1, data yang digunakan adalah data pada poin 1
5. Setiap iterasi $a, data keseluruhan akan diurutkan berdasarkan nama kolom, atau *key* pada *array* yang sedang berjalan pada iterasi $a.
6. Berikut adalah perkondisian yang digunakan untuk menentukan apakah data sesuai dengan kriteria yang diinginkan.

```
if (stripos($result[${'val'.$a}], $request->input(${'val'.$a}))!==FALSE && strtolower
    ($result[${'val'.$a}])<=strtolower($request->input(${'val'.$a}))) {
```

Gbr. 7 perkondisian untuk menentukan nilai benar atau salah dari suatu data

Keterangan dari perkondisian pada Gbr. 7 adalah sebagai berikut :

a. *Stripos*

Stripos - Temukan posisi kemunculan pertama dari *substring case-insensitive* dalam sebuah *string* (php.net). Sintaks *stripos* digunakan untuk mencocokan pola suatu *string* dengan *string* utama. Apabila terdapat kecocokan pola, maka kembalian dari perkondisian tersebut adalah indeks dimana pola itu berada, ketika tidak terdapat kecocokan pola maka data yang dicari tidak cocok.

b.Strtolower($result[${‘val’.$a}])<=strtolower($request->input(${‘val’.$a})

Digunakan untuk menentukan apakah nilai ASCII dari *value* tersebut lebih kecil atau sama dengan parameter pencarian. Sintaks *strtolower* sendiri digunakan untuk mengubah semua string menjadi huruf kecil atau *lowercase*, sehingga lebih mudah pada saat dibandingkan nilai ASCII nya. Ketika nilai ASCII dari *value* lebih kecil maka proses akan terus berjalan, namun ketika nilai ascii sudah lebih besar, dan perkondisian sudah tidak terpenuhi, maka proses akan berhenti.

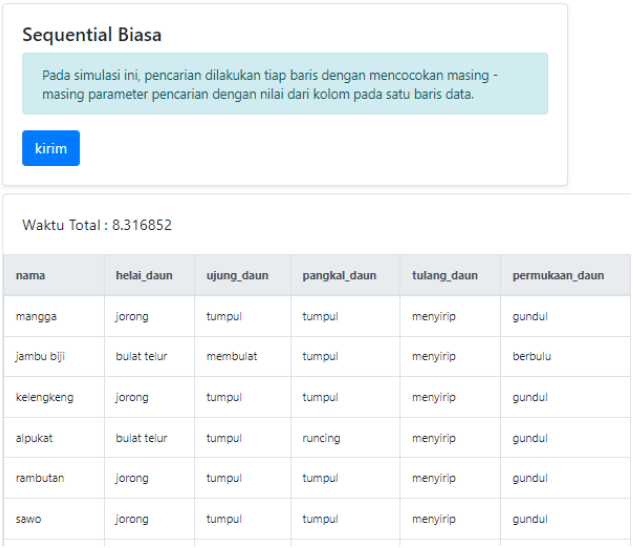
7. Jika perkondisian pada poin 6 keduanya bernilai *true* atau *benar*, maka satu baris data akan disimpan pada *variable* hasil

d. Implementasi pada Aplikasi

Tahapan yang dilalui dalam penelitian, pembangunan konsep, atau penyelesaian kasus, dituliskan pada bagian metodologi. Algoritma yang dibahas pada penelitian ini diterapkan pada aplikasi berbasis *website* dengan tampilan yang terdiri dari *form* pencarian dan kolom atau tampilan hasil pencarian. Algoritma *Sequential Search,, Sequential Search* pada kolom, *Sequential Search* pada kolom dengan indeks diterapkan pada Bahasa pemrograman PHP dengan penyimpanan data uji

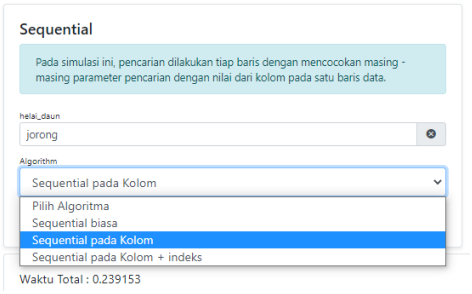
ISSN (Online) 2747-0563
Volume 4 Tahun 2024

menggunakan data dengan tipe .json. Hasil implementasinya dapat dilihat pada Gbr. 8 hingga Gbr. 10.



nama	helai_daun	ujung_daun	pangkal_daun	tulang_daun	permukaan_daun
mangga	jorong	tumpul	tumpul	menyirip	gundul
jambu biji	bulat telur	membulat	tumpul	menyirip	berbulu
kelengkeng	jorong	tumpul	tumpul	menyirip	gundul
alpukat	bulat telur	tumpul	runcing	menyirip	gundul
rambutan	jorong	tumpul	tumpul	menyirip	gundul
sawo	jorong	tumpul	tumpul	menyirip	gundul

Gbr. 8 Tampilan utama aplikasi



Gbr. 9 Pilihan jenis algoritma pencarian

nama	helai_daun	ujung_daun	pangkal_daun	tulang_daun	permukaan_daun
mangga	jorong	tumpul	tumpul	menyirip	gundul
kelengkeng	jorong	tumpul	tumpul	menyirip	gundul
rambutan	jorong	tumpul	tumpul	menyirip	gundul
sawo	jorong	tumpul	tumpul	menyirip	gundul
sirsak	jorong	runcing	runcing	menyirip	licin mengkilat
durian	jorong	runcing	tumpul	menyirip	licin mengkilat

Gbr 10 Hasil pencarian dari implementasi algoritma

e. Pengujian Sistem

Pada pengujian penerapan algoritma ini, data tanaman dipilih untuk digunakan sebagai data testing

Kriteria dan unit testing

Unit testing atau unit pengujian akan diterapkan pada pengujian kriteria yang telah disampaikan pada bab sebelumnya. *Unit testing* ini yang akan menentukan apakah unit–unit/komponen dari perangkat lunak yang dibuat telah mencapai keberhasilan atau kegagalan

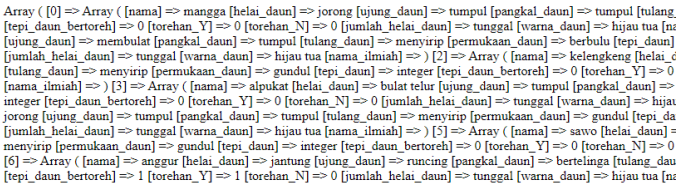
1. Membaca *file*

Pada pengujian ini, sistem diharapkan dapat melakukan proses baca keseluruhan data yang terdapat pada file yang dituju. Gbr. 11 merupakan hasil dari proses baca yang telah dilakukan oleh sistem.

Gbr. 11 pengujian pengambilan *file* dan melakukan proses baca

2. Konversi struktur data menjadi *array*

Hasil dari poin A akan diubah strukturnya dari yang bertipe data *json* menjadi *array*. Bahasa pemrograman php sendiri telah menyediakan sintaks untuk mengubah data bertipe *json* menjadi *array*. Hasilnya dapat dilihat pada Gbr. 12.



Gbr. 12 konversi data pada pengujian poin A kedalam

3. Melakukan sorting

Proses *sorting* yang dilakukan tiap kali perulangan berlangsung menggunakan *value* atau nilai dari masing – masing kolom yang akan diurutkan secara *ascending* atau dari besar ke kecil. Gbr. 13 menunjukkan bahwa *value* yang diurutkan adalah *value* dari kolom nama

nama	helai_daun	ujung_daun	pangkal_daun	tulang_daun
alpukat	bulat telur	tumpul	runcing	menyirip
anggur	jantung	runcing	bertelinga	menjari
bambu	pedang	runcing	tumpul	sejajar
bayam	bulat telur	tumpul	tumpul	menyirip
durian	jorong	runcing	tumpul	menyirip

Gbr. 13 proses pengurutan berdasarkan nama

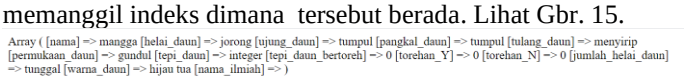
Gbr. 14 menunjukkan bahwa *value* yang diurutkan adalah *value* dari kolom nama

mawar	bulat telur	runcing	tumpul	menyirip
melati	bulat telur	tumpul	tumpul	menyirip
anggur	jantung	runcing	bertelinga	menjari
jarak hijau	jantung	runcing	bertelinga	menjari

Gbr.14 proses pengurutan berdasarkan helai daun

4. Membaca baris dan kolom pada

Proses membaca baris pada *array* dapat dilakukan dengan memanggil indeks dimana tersebut berada. Lihat Gbr. 15.



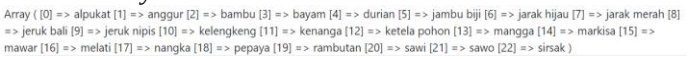
Gbr.15 proses mebaca baris data pada

5. Membaca kolom dan menyimpan sebagai baris

Proses menyimpan kolom dilakukan dengan membaca semua baris namun hanya mengambil *key* tertentu dari suatu *dataset*.

ISSN (Online) 2747-0563
Volume 4 Tahun 2024

Pada Gbr. 16, *key* yang diambil hanya "nama", sehingga didapat data berupa nama saja, tanpa menyimpan *key* lain didalam *array*.



Gbr. 16 kolom pertama atau key pertama dari data yang disimpan pada

6. Membandingkan data

Proses ini merupakan proses utama pada pembahasan algoritma ini. Pada proses ini, data akan dibandingkan dengan parameter–parameter pencarian tertentu. Perbandingan ini hanya dapat menghasilkan 2 kondisi yaitu benar dan salah atau *true* dan *false*, sehingga tidak akan terdapat persentase kebenaran selain 0 (salah) atau 1 (benar).

7. Perbandingan ASCII

Perbandingan ini digunakan ketika ingin menjalankan suatu perulangan, atau ketika ingin menghentikan suatu perulangan. Perulangan akan berjalan ketika nilai ASCII masih lebih kecil daripada batas yang ditentukan. Pada perbandingan ini, yang akan dibandingkan adalah *char* atau karakter pertama dari suatu string saja. Lalu satu *char* tersebut akan dibandingkan nilainya. Ketika tidak sebanding atau tidak sama dengan, maka perulangan akan terus berjalan, namun ketika sebanding atau sama dengan makan perulangan akan dihentikan

8. Menyimpan baris kedalam

Pada *Sequential Search* pada *array* dua dimensi, suatu variabel *array* didefinisikan pada keadaan *null* atau kosong. Ketika proses berjalan dan ditemukan data yang sesuai maka data tersebut akan dimasukkan kedalam *array* tersebut.

9. Menampilkan hasil pencarian

Terdapat perbedaan mekanisme penyimpanan hasil akhir pada *Sequential Search* dan *Sequential Search* pada kolom pada array dua dimensi. Perbedaan tersebut terdapat pada variabel yang didefinisikan untuk menyimpan hasil, pada *Sequential Search* pada *array* dua dimensi, *array* yang mulanya kosong, akan terus diisi hingga proses berakhir. Pada *Sequential Search* pada kolom dan *Sequential Search* pada kolom dengan pengindeksan, *array* akan didefinisikan tiap kali perulangan berlangsung, jadi apabila terdapat 10 macam kolom pada suatu *dataset*, maka jumlah *array* yang didefinisikan juga berjumlah 10, dengan *array* terakhir merupakan hasil akhir dari pencarian tersebut.

TABEL XIII
KRITERIA KEBERHASILAN SISTEM

no	Kriteria	Keberhasilan
1	Mampu membaca file .json	Ya
2	Mampu mengkonversi file .json menjadi *array*	Ya
3	Mampu melakukan pengurutan atau sorting sesuai dengan urutan – urutan tertentu	Ya
4	Mampu membaca semua baris dan kolom data pada *array*	Ya
5	Mampu membaca kolom dan menyimpannya sebagai baris	Ya
6	Mampu memberikan kondisi bahwa data sesuai atau tidak	Ya
7	Mampu melakukan perbandingan nilai ASCII suatu string dengan string lainnya	Ya
8	Sistem mampu menyimpan baris yang benar kedalam *array*	Ya
9	Mampu menampilkan hasil akhir pencarian	Ya

Pada Tabel XIII, nilai keberhasilan didapat berdasarkan pengujian yang telah dilakukan pada pengujian 1 – 9. Ketika pengujian unit dapat berjalan sesuai kriteria pengujian maka keberhasilan dinilai 'ya'.

Perbandingan algoritma

1. Waktu pengujian

Waktu pengujian diambil dari waktu yang sudah didefinisikan dari awal dari tiap–tiap proses dan dihentikan ketika proses telah selesai. Waktu pengujian tidak termasuk proses menampilkan hasil pengujian karena waktu telah dihentikan sebelum proses penampilan hasil pengujian. Waktu pengujian dibagi menjadi 3, antara lain waktu untuk membaca data keseluruhan, waktu untuk mengambil data kolom, dan waktu pencarian, serta total waktu pencarian yang merupakan akumulasi dari ketiga waktu tersebut. Pengujian waktu bagi tiap algoritma dilakukan sebanyak 10 kali. Teknis pengujian ini dilakukan dengan mengirim *request* kepada sistem sebanyak 10 kali agar sistem selalu mengulangi proses dari awal serta tidak terjadi *caching* dari beberapa data yang biasa dimuat diawal dari serangkaian proses. Proses caching dapat mempengaruhi waktu pengujian karena sistem tidak perlu membuka file yang sama ketika *cache* sudah tersimpan pada memori sistem.

a. *Sequential search*

No	Load data (ms)	Get Column (ms)	Proses pencarian (ms)	Total (ms)
1	4.4779	0.0064	2.3076	6.7919
2	5.8819	0.0065	2.2991	8.1875
3	4.1680	0.0053	2.1771	6.3504
4	4.2863	0.0054	2.1747	6.4664
5	4.2963	0.0053	2.1794	6.4810
6	4.3835	0.0056	2.1689	6.5580
7	4.2562	0.0052	2.1643	6.4257
8	4.2376	0.0055	2.1763	6.4194
9	4.2181	0.0052	2.1758	6.3991
10	4.5106	0.0056	2.3110	6.8272

Gbr.17 Hasil pengujian pertama Sequential Search

Pada pengujian *Sequential* Search, rata – rata *load* data membutuhkan waktu 4,07164ms, rata – rata waktu pengambilan kolom data membutuhkan 0,0056ms, rata – rata waktu pencarian 2,21342ms serta rata – rata waktu total pencarian (*load data*, pengambilan kolom, waktu pencarian) membutuhkan waktu 6,69336ms.

ISSN (Online) 2747-0563
Volume 4 Tahun 2024

b. *Sequential search pada kolom*

No	Load data (ms)	Get Column (ms)	Proses pencarian (ms)	Total (ms)
1	4.8828	0.0071	0.1004	4.9903
2	4.1684	0.0054	0.0855	4.2593
3	4.1746	0.0054	0.0843	4.2643
4	4.1143	0.0056	0.0841	4.2040
5	4.1804	0.0053	0.0840	4.2697
6	5.6642	0.0069	0.0943	5.7654
7	4.2093	0.0052	0.0855	4.3000
8	5.8935	0.0071	0.1000	6.0006
9	4.5558	0.0066	0.0905	4.6529
10	4.2065	0.0053	0.0858	4.2976

Gbr.18 Hasil pengujian Sequential Search pada kolom

Pada pengujian *Sequential* Search, rata – rata *load* data membutuhkan waktu 4,60498ms, rata – rata waktu pengambilan kolom data membutuhkan 0,0059ms, rata – rata waktu pencarian 0,08944ms serta rata – rata waktu total pencarian (*load data*, pengambilan kolom, waktu pencarian) membutuhkan waktu 4,70041ms.

c. *Sequential search pada kolom dengan indeks*

No	Load data (ms)	Get Column (ms)	Proses pencarian (ms)	Total (ms)
1	4.3748	0.0054	0.0937	4.4739
2	4.1493	0.0051	0.0923	4.2467
3	4.5897	0.0057	0.1001	4.6955
4	6.0499	0.0066	0.1023	6.1588
5	4.1698	0.0053	0.0917	4.2668
6	4.4558	0.0053	0.0980	4.5591
7	4.2049	0.0050	0.0928	4.3027
8	4.2369	0.0051	0.0924	4.3344
9	4.1334	0.0050	0.0940	4.2324
10	4.7295	0.0058	0.1006	4.8359

Gbr.19 Hasil pengujian Sequential Search pada kolom dengan indeks

Pada pengujian *Sequential* Search, rata – rata *load* data membutuhkan waktu 4,50940ms, rata – rata waktu pengambilan kolom data membutuhkan 0,0053ms, rata – rata waktu pencarian 0,09479ms serta rata – rata waktu total pencarian (*load data*, pengambilan kolom, waktu pencarian) membutuhkan waktu 4,61062ms.

2. Proses pencarian

Terdiri dari proses–proses yang dihitung berdasarkan proses yang berdiri sendiri, serta proses yang memiliki kondisi dan fungsi yang sama akan dihitung sebanyak satu kali. Proses yang dihitung antara lain variabel statis, *array*, perulangan *foreach*, perulangan *for*, serta perkondisian *if*.

a. Sequential search

Terdapat 2 tahap *preprocessing* atau pengambilan data dan pengambilan nama kolom, serta 7 tahap pada proses pencarian dan penyimpanan hasil antara lain perulangan for, perulangan *foreach* untuk membaca semua kolom dari satu baris data, perkondisian untuk mengetahui terdapat parameter pencarian atau tidak, perkondisian untuk mengetahui nilai dari parameter apakah *null* atau *!null*, perkondisian untuk mengetahui apakah nilai dari suatu kolom data bernilai sama dengan parameter pencarian atau tidak, perkondisian untuk mengetahui apakah jumlah yang sesuai bernilai sama dengan jumlah yang dicari, menyimpan hasil kedalam ketika poin 6 bernilai true atau benar.

b. *Sequential search* pada kolom

Terdapat 2 tahap *preprocessing* atau pengambilan data dan pengambilan nama kolom, serta 8 tahap pada proses pencarian dan penyimpanan hasil antara lain perulangan *for*, Perkondisian untuk mengetahui apakah perulangan pada poin 1 bernilai 0 atau lebih, mengambil data dari proses sebelumnya, menyimpan data dari proses sebelumnya kedalam suatu *array*, perkondisian untuk mengetahui parameter pencarian bernilai *null* atau *!null*, perkondisian untuk mengetahui apakah data bernilai sama dengan parameter pencarian, proses menyimpan satu baris pada suatu *temporary*, dan menyimpan semua *array* dari *temporary* kedalam *array* hasil.

c. *Sequential search* pada kolom dengan indeks

Terdapat 2 tahap *preprocessing* atau pengambilan data dan pengambilan nama kolom, serta 8 tahap pada proses pencarian dan penyimpanan hasil antara lain perulangan *for*, perkondisian untuk mengetahui apakah perulangan pada poin 1 bernilai 0 atau lebih, mengambil data dari proses sebelumnya, melakukan *sorting* atau mengurutkan data, menyimpan data dari proses sebelumnya kedalam suatu *array*, perkondisian untuk mengetahui parameter pencarian bernilai *null* atau *!null*, perkondisian untuk mengetahui apakah data bernilai sama dengan parameter pencarian, ketika poin 6 bernilai true atau benar, maka satu baris akan disimpan pada suatu *array temporary*, menyimpan semua dari *array temporary* kedalam hasil.

3. Hasil perbandingan

Tahap *preprocessing* pada ketiga algoritma pada tabel dibawah menunjukkan angka yang berbeda namun perbedaan yang dihasilkan tidak signifikan. Namun pada rata – rata proses pencarian, *Sequential Search* biasa menunjukkan waktu pencarian yang lebih lambat, dengan kedua algoritma lain yang memiliki waktu rata – rata pencarian dibawah 1 *ms* sedangkan *Sequential Search* biasa menghasilkan waktu diatas 1 *ms*.

TABEL XIII
HASIL PERBANDINGAN ALGORITMA

Algoritma	**Preprocessing (ms)**	**Rata – rata proses pencarian (ms)**	**proses**	**Akurasi**
1	4,07164	4,4401912	9	100%

ISSN (Online) 2747-0563
Volume 4 Tahun 2024

Algoritma	Preprocessing (ms)	Rata – rata proses pencarian (ms)	proses	Akurasi
2	4,60498	0,3372588	10	100%
3	4,5094	0,5668663	11	100%

Algoritma nomor 1 merupakan *Sequential Search* pada *array* dua dimensi. Memiliki waktu preprocessing lebih cepat daripada 2 algoritma lainnya yaitu sebesar 4,07164ms, namun memiliki rata-rata waktu pencarian yang lebih tinggi yaitu sebesar 4,4401912 ms dengan 9 proses. Algoritma nomor 2 merupakan *Sequential Search* kolom pada *array* dua dimensi memiliki waktu preprocessing tertinggi yaitu 4,60498ms namun dengan rata-rata waktu pencarian yang paling rendah sebesar 0,3372588ms dengan 10 proses. Sedangkan algoritma nomor 3 merupakan *Sequential Search* kolom pada *array* dua dimensi dengan pengindeksan yang memiliki waktu preprocessing nomor 2 terendah dengan 4,5094ms dengan rata-rata waktu pencarian sebesar 0,5668663ms dengan 11 proses. Dapat diambil kesimpulan bahwa algoritma *Sequential Search* kolom pada *array* dua dimensi memiliki waktu pencarian tercepat dibandingkan ketiga algoritma lainnya dengan kompleksitas nomor 2 setelah *Sequential Search* pada array dua dimensi.

V. KESIMPULAN

Sequential search dapat diterapkan pada 2 dimensi dengan cara memanipulasi struktur dari suatu data yang awalnya berbentuk 2 dimensi, diubah menjadi satu dimensi dengan mengambil satu baris dari suatu dataset secara berulang – ulang. Penerapan pengindeksan pada *sequential search* dapat dilakukan dengan cara mengurutkan suatu *string* berdasarkan suatu urutan – urutan tertentu dengan memanfaatkan nilai ASCII dari karakter pertama dari *string* utama. Serta memberikan perkondisian bahwa pencarian akan terus berjalan ketika nilai ASCII dari *string* a lebih kecil daripada *string* b dan pencarian akan berhenti ketika nilai *string* a lebih besar daripada nilai ASCII daripada *string* b, dengan *string* a adalah kata kunci dan *string* b adalah *string* utama yang dicari. *Sequential Search* dan *Sequential Search* pada kolom dapat melakukan pencarian serta menampilkan hasil lebih cepat daripada *Sequential Search* yang harus menerapkan proses *sorting* terlebih dahulu. *Sequential search* pada kolom mampu melakukan pencarian dengan waktu tercepat dengan rata – rata waktu pencarian 0,08944 *ms*, disusul oleh sequential search pada kolom dengan indeks dengan rata – rata waktu pencarian 0,09479 *ms* lalu *sequential search* pada urutan paling akhir dengan rata – rata waktu pencarian 2,21342 *ms*.

REFERENSI

[1] M. Utami and Y. Apridiansyah, "Implementasi Algoritma Sequential Searching Pada Sistem Pelayanan Puskesmas Menggunakan Bootstrap (Studi Kasus Puskesmas Kampung Bali Bengkulu)," *JSAI (Journal Sci. Appl. Informatics)*, vol. 2, no. 1, pp. 81–86, 2019, doi: 10.36085/jsai.v2i1.166.

[2] J. Sihombing, "Penerapan Stack Dan Queue Pada Array Dan Linked List Dalam Java," no. June, pp. 15–24, 2020.

[3] A. Akbar, "Algoritma Sequential Searching Implementasi Algoritma Sequential Searching Pada Aplikasi E-Office," *Naratif(Jurnal Nasional, Riset, Apl. dan Tek. Inform.*, vol. 1, no. 1, pp. 29–34, 2019.

[4] Suhartini, Muchlis, and R. P. Lestari, "Implementation of Sequential Search Method on Android-based Jakabaring Dictionary," *J. Transform.*, vol. 16, no. 1, p. 74, 2018, doi: 10.26623/transformatika.v16i1.830.

[5] D. Raghuvanshi, "Data Structure: Theoretical Approach," *Int. J. Trend Sci. Res. Dev.*, vol. Volume-3, no. Issue-1, pp. 268–273, 2018, doi: 10.31142/ijtsrd18977.

[6] A. A. G. Y. Paramartha, G. K. Suryaningsih, and K. Y. E. Aryanto, "Implementasi Web Service Pada Sistem Pengindeksan Dan Pencarian Dokumen Tugas Akhir, Skripsi, Dan Praktik Kerja Lapangan," *JST (Jurnal Sains dan Teknol.*, vol. 5, no. 2, p. 818, 2017, doi: 10.23887/jst-undiksha.v5i2.8813.

[7] Kamariah Tambunan, "Tesaurus Bioteknologi: Sebagai Alat Bantu Pengindeksan Dokumen," *Baca J. Dokumentasi Dan Inf.*, vol. 33, no. 2, 2012, doi: http://dx.doi.org/10.14203/j.baca.v33i2.99.

[8] F. S. Jumeilah, "Penerapan Support Vector Machine (SVM) untuk Pengkategorian Penelitian," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 1, no. 1, pp. 19–25, 2017, doi: 10.29207/resti.v1i1.11.

[9] W. Wibisono and F. Baskoro, "Pengujian Perangkat Lunak Dengan Menggunakan Model Behaviour Uml," *JUTI J. Ilm. Teknol. Inf.*, vol. 1, no. 1, p. 43, 2002, doi: 10.12962/j24068535.v1i1.a95.

[10] E. S. Eriana and G. Saputri, "Analisis white box testing pada aplikasi koperasi simpan pinjam berbasis web," *J. Ilmu Komput.*, vol. III, no. 03, pp. 2–7, 2020.

[11] A. N. Hasibuan and T. Dirgahayu, "Pengujian dengan Unit Testing dan Test case pada Proyek Pengembangan Modul Manajemen Pengguna," 2020.

[12] R. Irawan, Ilhamsyah, and Y. Brianorman, "APLIKASI ENKRIPSI DAN DEKRIPSI PESAN SINGKAT MENGGUNAKAN ALGORITMA KNAPSACK BERBASIS ANDROID," *J. Coding Sist. Komput. Untan Vol. 03, No. 3 (2015), hal 57-66*, vol. 29, no. 6, pp. 58–59, 2015.

[13] N. Hindriani, Narwen, and H. Yozza, "Implementasi Antrian Dengan menggunakan array," *J. Mat. UNAND Vol.3 No. 4 Hal. 147 - 151*, vol. 3, no. 4, pp. 147–151, 2014.

[14] T. Lv, P. Yan, and W. He, "Survey on JSON Data Modelling," *J. Phys. Conf. Ser.*, vol. 1069, no. 1, 2018, doi: 10.1088/1742-6596/1069/1/012101.

[15] K. A. O. Al-husseini and A. H. Obaid, "Usage of Prototyping in Software Testing," *Multi-Knowledge Electron. Compr. J. Educ. Sci. Publ.*, no. November, 2018.