**PENGEMBANGAN ALGORITMA *SEQUENTIAL SEARCH* PADA ARRAY DUA DIMENSI**

( Studi kasus : pencarian )

**PROPOSAL**

**Oleh:**

**ZEGA FEBRIANTO**

**NIM. 17104410036**



**UNIVERSITAS ISLAM BALITAR**

**FAKULTAS TEKNOLOGI INFORMASI**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**JANUARI 2021**

**LEMBAR PERSETUJUAN**

Proposal oleh Zega Febrianto ini

Telah diperiksa dan disetujui untuk diuji,

Blitar, 9 Maret 2021

Pembimbing I

Harris Yuana, S.T.,M.T.

NIDN : 0728078602

Blitar, 9 Maret 2021

Pembimbing II

Dimas Fanny Hebrasianto Permadi S.T., M.Kom.

NIDN : 0731039201

**DAFTAR ISI**

Halaman Sampul i

Lembar persetujuan ii

Daftar isi iii

**BAB I PENDAHULUAN** 1

1.1 Latar Belakang Masalah 1

1.2 Rumusan Masalah 4

1.3 Tujuan 4

1.4 Manfaat Penelitian 4

1.4.1 Manfaat Bagi Penulis 4

1.4.2 Manfaat Bagi Universitas 5

1.4.3 Manfaat Bagi Masyarakat (Umum) 5

1.4.3 Manfaat Bagi Pengguna Hasil Penelitian 5

1.5 Ruang Lingkup 5

1.6 Sistematika Penulisan 7

**BAB II TINJAUAN PUSTAKA** 8

2.1 Landasan Teori 8

2.1.1 Mesin Pencarian 8

2.1.2 Algoritma *Sequential search* 10

2.1.3 Indeks 14

2.1.4 Morfologi Helaian Daun 15

2.1.5 Bahasa Pemrograman PHP 19

2.1.6 JSON 19

2.1.7 Array 20

2.2 Kajian Penelitian 20

**BAB III METODE PENELITIAN** 23

3.1 Metode Perancangan 23

3.2 Alur Penelitian 24

3.3 Waktu dan Tempat Penelitian 25

3.4 Obyek Penelitian 26

3.5 Teknik Pengumpulan Data 26

3.6 Metode yang diusulkan 26

3.7 Pengujian sistem 31

**DAFTAR RUJUKAN**

**DAFTAR TABEL**

2.1 Tabel perbandingan sequential search dan binary search 13

2.2 Helai daun 15

2.3 Ujung daun 16

2.4 Pangkal daun 17

2.5 Tulang daun 17

2.6 Permukaan daun 17

2.7 Tepi daun 18

2.8 Torehan daun bertoreh 18

2.9 Torehan yang tidak mempengaruhi bangun daun 18

2.10 Torehan yang mempengaruhi bangun daun 18

2.11 Jumlah helai daun 19

2.12 Kajian penelitian 21

3.1 Jadwal pelaksanaan penelitian 25

3.2 Contoh tabel perbandingan algoritma 32

**DAFTAR GAMBAR**

2.1 Alur *Sequential Search* 10

2.2 *Array* 1 dimensi 20

2.3 *Array* 2 dimensi 21

3.1 *Sequential Search* pada satu larik data 27

3.2 *Sequential Search* pada larik multidimensi 28

**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang Masalah**

Pencarian merupakan kebutuhan utama yang paling sering digunakan oleh seorang yang berhubungan dengan data. Pencarian biasanya dilakukan saat kita ingin menemukan *record* atau rekaman tertentu pada suatu dataset besar, misal data transaksi kita, atau mungkin pada saat menggunakan kamus. Pada penggunaan beberapa alat tersebut, pengguna pasti tidak akan melewatkan fitur pencarian yang disediakan, dikarenakan pengguna tidak mungkin membuang banyak waktu hanya untuk melakukan pencarian secara manual atau membaca satu – persatu. Pada jaringan situs web, mesin pencari web / web search engine adalah program komputer yang dirancang untuk melakukan pencarian atas berkas - berkas yang tersimpan dalam layanan *WWW, FTP*, publikasi milis, ataupun *newsgroup* dalam sebuah ataupun sejumlah komputer dalam suatu jaringan (Salamun dkk., 2017). Namun, pada sistem yang akan dikembangkan peneliti, pencarian hanya akan dilakukan pada suatu data yang tersimpan pada dataset tertentu yang disimpan dalam bentuk .JSON (Javascript Object Notation).

*Sequential* atau *linear search* merupakan algoritma paling sederhana untuk menyelesaikan masalah terkait pencarian data. *Sequential search*ing 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 (Utami & Apridiansyah, 2019). *Sequential search* yang sifatnya merupakan pencarian secara langsung tanpa harus mengurutkan data terlebih dahulu, merupakan solusi termudah untuk permasalahan diatas namun dengan syarat harus diterapkannya sedikit modifikasi agar potensi dari algoritma tersebut dapat digunakan secara maksimal.

Potensi dari Sequential atau linear search mungkin terbatas pada struktur data yang biasa digunakan pada algoritma itu sendiri yaitu larik satu dimensi. 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 larik dua dimensi. Hal tersebut dilakukan atas dasar pertimbangan bahwa data tidak mungkin selalu ideal dan selalu cocok dengan sifat dasar dari suatu algoritma. Algoritma sequential atau linear search masih dapat dikatakan unggul pada beberapa jenis data tertentu, misalnya pada data yang dalam kondisi sedang tidak urut seperti pada yang diungkapkan oleh Utami & Apridiansyah ( 2019) bahwa data yang dicari tersebut tidak perlu diurutkan terlebih dahulu.

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 larik dua dimensi, menurut Sihombing (2020:16) *array* 2 dimensi sering di analogikan 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. Selanjutnya, penyesuaian kedua adalah penambahan *indexing* pada proses algoritma tersebut. 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. *Indexing* 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. 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.

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 larik dua dimensi diharapkan dapat mempermudah dan meningkatkan kecepatan dari proses pencarian. Serta peneliti dapat mengetahui algoritma dan struktur data mana yang mampu melakukan komputasi tercepat dengan kondisi data dengan struktur dan jumlah yang sama.

* 1. **Rumusan Masalah**

1. Bagaimana cara menerapkan algoritma *Sequential Search* pada pencarian dalam lariksatu dimensi dan larik dua dimensi?
2. Bagaimana cara melakukan *indexing* pada penerapan algoritma *Sequential Search* pada pencarian dalam larik satu dimensi dan larik dua dimensi?
3. Bagaimana perbandingan kecepatan dari *Sequential Search* tanpa *indexing* dibandingkan *Sequential Search* dengan *indexing* pada larik satu dimensi dan larik dua dimensi?
   1. **Tujuan**
4. Mengetahui penerapan *Sequential Search* dalam pencarian pada larik satu dimensi dan larik dua dimensi
5. Mengetahui penerapan *Indexing* pada *sequential search* untuk melakukan pencarian pada larik satu dimensi dan larik dua dimensi
6. Mengetahui kecepatan dari *Sequential Search* tanpa *indexing* dibandingkan *Sequential Search* dengan *indexing* pada larik satu dimensi dan larik dua dimensi
   1. **Manfaat Penelitian**
      1. **Manfaat Bagi Penulis**

Meningkatkan kemampuan untuk merancang sistem serta menerapkan keilmuan yang didapat oleh penulis selama masa pendidikan menjadi sebuah produk yang dapat dimanfaatkan bagi penulis maupun orang lain.

* + 1. **Manfaat Bagi Universitas**

1. Sebagai rujukan bagi mahasiswa khususnya mahasiswa Teknik Informatika yang ingin mengembangkan aplikasi serupa.
2. Mengembangkan hasil penelitian baik berupa penambahan, pengurangan, ataupun penyempurnaan.
   * 1. **Manfaat Bagi Masyarakat (Umum)**

Hasil pengujian dari sistem dapat dikembangkan menjadi aplikasi yang siap digunakan oleh masyarakat.

* + 1. **Manfaat Bagi Pengguna Hasil Penelitian**

1. Dataset tanaman yang digunakan peneliti sebagai data uji dapat digunakan oleh peneliti lain untuk melakukan penelitian dengan data serupa.
2. Dipergunakan untuk melakukan analisa keefektifan algoritma yang dipakai oleh penulis dengan algoritma lain dalam bidang algoritma dan struktur data.
   1. **Ruang Lingkup**

Penelitian ini memiliki batasan penelitian sebagai berikut.

1. Data uji menggunakan 1000 dataset tanaman tingkat tinggi yang terdiri dari 50 jenis data yang dilipat gandakan. Data tidak harus valid karena data hanya akan digunakan sebagai data uji dari algoritma.
2. Pengembangan aplikasi hanya berupa prototype namun sudah dapat difungsikan.
3. Prototype terdiri dari 4, yaitu :
   1. Prototype *sequential search* pada larik satu dimensi
   2. Prototype *sequential search* pada larik dua dimensi
   3. Prototype *indexed-sequential search* pada larik dua dimensi
4. Komputasi dilakukan pada perangkat keras dengan spesifikasi berikut :
   1. Laptop merk Lenovo
   2. RAM 4 GB
   3. HDD 1 TB. Digunakannya HDD daripada SDD adalah dengan tujuan waktu komputasi pada HDD lebih lambat sehingga lebih dapat dilihat oleh peneliti seberapa signifikan jalannya suatu algoritma pada perangkat *low-end.*
   4. Processor AMD A9-9425 base clock : 3.0 GHz boost Clock Up to 3.6 GHz 2 CPU Cores 3 GPU Cores 1MB L2 Cache
5. Komputasi dilakukan pada perangkat lunak sebagai berikut
   1. Windows 10
   2. Google Chrome yang disimulasikan pada mode *offline*
   3. Xampp
   4. **Sistematika Penulisan**

Sistematika penulisan yang digunakan dalam penulisan proposal ini dapat diuraikan sebagai berikut:

**BAB I PENDAHULUAN**

Bab ini membahas mengenai latar belakang permasalahan, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, serta sistematika penulisan.

**BAB II TINJAUAN PUSTAKA**

Bab ini membahas landasan teori dan kajian pustaka yang digunakan pada penelitian. Landasan teori membahas seputar teori-teori dan konsep yang dilakukan yang mampu mendukung dalam penyelesaian masalah. Sedangkan kajian pustaka berisi tentang pemaparan beberapa penelitian terdahulu yang berhubungan dengan permasalahan yang diteliti dan menjadi acuan konseptual.

**BAB III METODE PENELITIAN**

Bab ini memaparkan Jenis dan pendekatan penelitian, alur penelitian, waktu penelitian, obyek penelitian, Teknik pengumpulan data serta metode yang diusulkan.

**BAB II**

**TINJAUAN PUSTAKA**

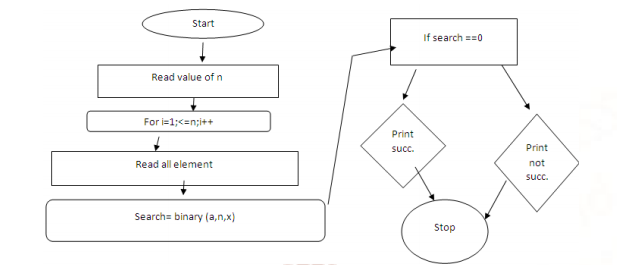
* 1. **Landasan Teori** 
     1. **Mesin pencarian**

Mesin pencari web / web search engine adalah program komputer yang dirancang untuk melakukan pencarian atas berkas-berkas yang tersimpan dalam layanan www, ftp, publikasi milis, ataupun news group dalam sebuah ataupun sejumlah komputer dalam suatu jaringan. Dalam menggunakan fasilitas mesin pencarian seperti google harus dengan beberapa cara agar bisa langsung mendapatkan informasi atau file yang di inginkan.

Ulasan dibawah ini dilakukan oleh peneliti dalam rangka perbandingan fitur dari beberapa mesin pencarain yang sering digunakan baik dalam melakukan pencarian secara umum yang dilakukan pada google.com dan pencarian secara khusus yang dilakukan pada garden.org

1. *Google search engine* ( google.com )
   1. Mesin pencarian google.com hanya menggunakan parameter pencarian berupa kata kunci, gambar ( gambar yang merupakan gambar hasil unggahan seseorang kedalam situs web ), serta suara ( yang akan diubah menjadi text biasa )
   2. Mesin pencarian google.com memang diperuntukkan untuk pencarian bersifat umum, tidak diperuntukkan untuk hal bersifat khusus ( dalam hal ini adalah pencarian tanaman ).
   3. Mesin pencarian serupa, antara lain bing, aol, yahoo dll. merupakan mesin pencarian yang bersifat umum seperti google.com.
2. The National Gardening Association ( [garden.org](https://garden.org/plants/) )
3. Merupakan mesin pencarian secara khusus ditujukan untuk pencarian tanaman
4. Pencarian menggunakan nama tanaman dan tipe tanaman. Pengguna yang tergolong orang awam belum tentu memiliki keilmuan yang cukup untuk menentukan nama dan tipe tanaman yang hendak dicari.
5. Pencarian berdasarkan tipe memiliki parameter pencarian yang relatif luas ( 39 parameter pencarian ). Pencarian berdasarkan tipe tanaman pada situs web tersebut, meliputi daun, buah, bunga, warna bunga, waktu berbuah dll..
   * 1. **Algoritma *Sequential search***

Ketika item data disimpan dalam kumpulan seperti daftar, kami mengatakan bahwa mereka memiliki hubungan linier atau sekuensial. Setiap item data disimpan dalam posisi relatif terhadap yang lain. Dalam daftar Python, posisi relatif ini adalah nilai indeks dari masing-masing item. Karena nilai indeks ini diurutkan, kita bisa mengunjunginya secara berurutan. Proses ini memunculkan teknik pencarian pertama kami, pencarian berurutan. Mulai dari item pertama dalam daftar, kita cukup berpindah dari satu item ke item lain, mengikuti urutan yang mendasari sampai kita menemukan apa yang kita cari atau kehabisan item. Jika kami kehabisan item, kami menemukan bahwa item yang kami cari tidak ada (Raghuvanshi,2018).

 **Gambar 2.1** Alur *Sequential Search*

*Sequential search*ing 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. Terdapat L yang merupakan larik yang berisi n buah data (L[0], L[1],...,L[n-1] dan k adalah data yang hendak dicari. Pencarian dilakukan untuk menemukan L[i]=k Dengan I adalah bilangan index terkecil yang memenuhi kondisi 0 ≤ k ≤ n-1. Tentukan saja ada kemungkinan bahwa data yang dicari tidak ditemukan (Utami & Apridiansyah, 2019).

Kelebihan dari proses pencarian secara sequential ini (Muhazir dkk., 2017).

1. Jika data yang dicari terletak didepan , maka data akan ditemukan dengan cepat. Tetapi dibalik kelebihannya ini, teknik ini juga memiliki kekurangan
2. jika data yang dicari terletak dibelakang atau paling akhir, maka akan membutuhkan waktu yang lama dalam proses pencariannya.
3. beban komputer akan semakin bertambah jika jumlah data dalam array sangat banyak.

Berikut cara kerja dari Metode *Sequential search* (Utami & Apridiansyah, 2019):

1. i ← 0
2. Ditemukan← false
3. Selama (tidak ditemukan) dan (i <= N)kerjakan baris 4
4. Jika (Data[i] = x) maka ditemukan← true, jika tidak i ← i + 1
5. Jika (ditemukan) maka i adalah indeks dari data yang dicari,jika data tidak ditemukan berarti data tidak ada.

Contoh pencarian data Sequential Search (Suhartini dkk., 2018):

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)

1. Bandingkan data ke-2 dengan data yang dicari.

15 == 7 → (False. Bukan data yang dicari)

1. Bandingkan data ke-3 dengan data yang dicari.

7 == 7 → (True. Data yang dicari ditemukan)

1. Proses pencarian berhenti.

Perbandingan antara *linear search/sequential search* dengan *binary search* (Raghuvanshi, 2018)

**Tabel 2.1** Tabel 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 |

Berdasarkan pada tabel diatas, penulis mempertimbangkan bahwa pada masalah yang diangkat oleh peneliti, algoritma yang sesuai adalah *linear*/*sequential search*. Hal tersebut dipertimbangkan berdasarkan dari efisiensi, struktur data yang dimiliki serta proses pencarian yang ingin dilakukan. Utamanya, struktur data yang tidak dalam keadaan urut akan menambah waktu komputasi ketika harus diurutkan terlebih dahulu sebelum proses pencarian dilakukan. Data akan urut pada proses pertama, namun data mungkin saja tidak urut pada proses kedua sehingga tiap proses harus dilakukan proses *sorting* atau pengurutan yang akan meningkatkan waktu komputasi.

* + 1. **Indeks**

Indeks, dalam Kamus Besar Bahasa Indonesia 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 (Kamariah Tambunan, 2012).

Dalam sistem yang akan dikembangkan khususnya pada *prototype* *indexed-sequential search*, index digunakan untuk mengarahkan pencarian menuju kepada *index* terdekat dengan kata kunci yang diberikan kepada sistem, serta sistem akan behenti ketika kata kunci telah melampaui atau lebih besar daripada nilai dari *index* tersebut.

* + 1. **Morfologi Helaian Daun**

Daun tumbuhan memiliki bentuk dan ukuran yang bervariasi, mulai dari yang berbentuk duri kecil pada kaktus hingga yang berbentuk lebar pada palm. Sekalipun bentuk dan ukuran daun tampak bervariasi, pada dasarnya daun terdiri dari tiga bagian, yaitu bagian basal yang berkembang menjadi pelepah (vagina), tangkai daun (petiolus) dan helaian daun (lamina). Daun yang memiliki ketiga bagian tersebut dinamakan daun lengkap. Pada sebagian besar tumbuhan, daun hanya terdiri dari satu atau dua bagian saja, yakni helai daun saja, tangkai dan helai daun, pelepah dan helai daun, atau tangkai daun saja. Daun-daun yang demikian dinamakan sebagai daun tak lengkap. Atas dasar konfigurasi helaiannya, daun dapat dibedakan menjadi daun tunggal dan daun majemuk. Daun tunggal adalah daun yang helaiannya hanya terdiri dari satu helai tanpa adanya persendian di bagian dasar helaian tersebut, sedangkan daun majemuk adalah daun dimana helaiannya disusun oleh sejumlah bagian-bagian terpisah yang berbentuk seperti daun dan disebut anak daun (leaflet) (Latifa, 2015).

Struktur helaian daun antara lain,

**Tabel 2.2** 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 2.3** 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 2.4** Pangkal daun

| Pangkal Daun |
| --- |
| Berlekuk (*emarginatus*) |
| Membulat (*rotundatus*) |
| Runcing (*acutus*) |
| Rompang (*truncatus*), pangkal daun rata |
| Meruncing (*acuminatus*), bentuknya seperti runcing (acutus) tetapi lebih tajam. |
| Tumpul (*obtusus*) |
| Miring atau asimetri (*oblique*) |

**Tabel 2.5** Tulang daun

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

**Tabel 2.6** 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 2.7** Tepi daun

|  |
| --- |
| Tepi daun |
| Rata (Integer) |
| Bertoreh (Divisus) |

**Tabel 2.8** 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 2.9** 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 |

**Tabel 2.10** 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 2.11** 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. |

* + 1. **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.

* + 1. **JSON (Javascript Object Notation)**

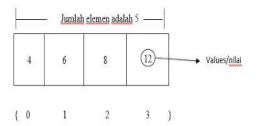
JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON adalah format teks yang tidak bergantung pada bahasa 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 (Rosid, 2017).

* + 1. **Array**

Array merupakan suatu kumpulan data terstruktur yang berupa sejumlah data sejenis (memiliki jenis data yang sama) yang jumlahnya tetap dan diberi suatu nama tertentu. Sebuah array dapat dibayangkan sebagai sekumpulan kotak yang menyimpan sekumpulan elemen bertipe sama secara berurutan (sequential) di dalam memori komputer. Array juga dapat digambarkan sebagai elemen-elemen yang disusun secara vertical sehingga dinamai tabel. Setiap elemen array data diacu melalui indeksnya. Karena elemen disimpan secara berturutan, indeks array haruslah 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. Tiap elemen array langsung diakses dengan menspesifikasikan nama array. Array dapat berupa array 1 dimensi, 2 dimensi, dan bahkan n-dimensi (Sihombing, 2020).

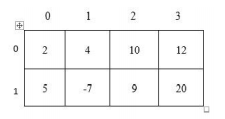
Menurut Sihombing(2020:16), *array* 1 dimensi dan 2 dimensi adalah :

1. Array 1 dimensi merupakan sekumpulan item data yang disusun secara baik menjadi suatu rangkaian dan diacu atau ditunjuk oleh satu identifier.



**Gambar 2.2** *array* 1 dimensi

1. Array 2 dimensi sering di analogikan dan digambarkan sebagai bentuk matriks, dengan indeks pertama berfungsi sebagai baris dan indeks kedua digunakan untuk kolom.



**Gambar 2.3** *array* 2 dimensi

* 1. **Kajian Penelitian**

Berikut beberapa penelitian yang mendasari peneliti dalam melakukan penelitian tentang ketepatan dari Algoritma *Sequential search* untuk mengklasifikasi tanaman berdasarkan bentuk daun tanaman.

**Tabel 2.1**2 Kajian Penelitian tentang Algoritma *Sequential Search,* Taksonomi tanaman, bentuk daun tanaman, JSON, dan *Array*

| **Peneliti** | **Tahun** | **Judul** | **Jurnal** |
| --- | --- | --- | --- |
| Sofyah Andriani | 2017 | Tinjauan Yuridis Sosiologis Perlindungan Hukum Terhadap Pemegang Hak Perlindungan Varietas Tanaman (Pvt). | Penulisan Hukum Fakultas Hukum Universitas Muhammadiyah Malang. |
| Roimil Latifa | 2015 | Karakter Morfologi Daun Beberapa Jenis Pohon Penghijauan Hutan Kota Di Kota Malang. | Prosiding Seminar Nasional Pendidikan Biologi 2015, Prodi Pendidikan Biologi FKIP Universitas Muhammadiyah Malang |
| Muhammad Asyhar Agmalaro, Aziz Kustiyo, Auriza Rahmad Akbar | 2013 | Identifikasi Tanaman Buah Tropika Berdasarkan Tekstur Permukaan Daun Menggunakan Jaringan Syaraf Tiruan Tropical Fruit Plants Identification Based on Leaf Surface Texture Image Using Artificial Neural Network. | Jurnal Ilmu Komputer Agri-informatika Volume 2 Nomor 2 halaman 73 - 82 ISSN: 2089-6026 |
| Abdullah Muhazir, Muhammad Fakhriza,  Eddy Sutejo | 2017 | Implementasi Metode Sequential Dalam Pencarian Pendistribusian Barang pada Cargo Integration Sistem. | *Jurnal Dan Penelitian Teknik Informatika*, *2*(2), 24–30. |
| Durgesh Raghuvanshi | 2018 | Data Structure: Theoretical Approach. | International Journal of Trend in Scientific Research and Development, Volume-3(Issue-1), 268–273. <https://doi.org/10.31142/ijtsrd18977> |
| Mochamad Alfan Rosid. | *2017* | *Implementasi JSON untuk Minimasi Penggunaan Jumlah Kolom Suatu Tabel Pada Database PostgreSQL.* | *JOINCS (Journal of Informatics, Network, and Computer Science), 1(1), 33. https://doi.org/10.21070/joincs.v1i1.802* |
| Johnson Sihombing | 2020 | Penerapan Stack Dan Queue Pada Array Dan Linked List Dalam Java. | June, 15–24. |
| Marissa Utami, Yovi Apridiansyah | 2019 | Implementasi Algoritma *Sequential search*ing Pada Sistem Pelayanan Puskesmas Menggunakan Bootstrap (Studi Kasus Puskesmas Kampung Bali Bengkulu). | JSAI (Journal Scientific and Applied Informatics), 2(1), 81–86. <https://doi.org/10.36085/jsai.v2i1.166> |
| Paramartha, A. G. Y., Suryaningsih, G. K., & Aryanto, K. Y. E. | 2017 | Implementasi Web Service pada Sistem Pengindeksan dan Pencarian Dokumen Tugas Akhir, Skripsi, dan Praktik Kerja Lapangan. | JST (Jurnal Sains dan Teknologi), 5(2). |
| Al-husseini, K. A. O., & Obaid, A. H. | 2018 | Usage of Prototyping in Software Testing. Multi-Knowledge Electronic Comprehensive | Journal For Education And Science Publications, November. |

**BAB III**

**METODE PENELITIAN**

1. **Metode perancangan**

Penelitian ini menggunakan metode perancangan model prototyping. Model prototipe memungkinkan anda membuat prototipe perangkat lunak produk sebelum atau selama tahap penyusunan persyaratan untu perangkat lunak produk. Calon pengguna bekerja dengan prototipe ini mengidentifikasi kekuatannya dan kelemahan, hasilnya 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. Sebagai hasil dari pekerjaan ini, produk akan mencerminkan kebutuhan nyata pengguna (Al-husseini & Obaid, 2018).

Menurut Ogedebe (2012), prototyping dimulai dengan pengumpulan kebutuhan, melibatkan pengembang dan pengguna sistem untuk menentukan tujuan, fungsi dan kebutuhan operasional sistem.

Langkah-langkah dalam prototyping adalah sebagai berikut :

1. Pengumpulan Kebutuhan.
2. Proses desain yang cepat.
3. Membangun prototipe.
4. Evaluasi dan perbaikan.
5. **Alur penelitian**

Alur penelitian pada penelitian ini dibagi menjadi beberapa tahap, yaitu :

1. Pengumpulan data uji

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

* 1. Melakukan pendataan tanaman pada aplikasi Microsoft excel. Mengisikan kolom ciri tanaman yang telah didata dengan ciri – ciri yang terdapat pada buku rujukan. Ciri – ciri helaian daun tumbuhan terdiri dari 10 kolom yang berisi data string.
  2. Dataset tanaman yang telah mencapai batas yang ditentukan lalu disimpan dalam bentuk .csv lalu
  3. ubah menjadi dalam bentuk .json (Javascript Object Notation) untuk selanjutnya dapat diproses oleh sistem.

1. Implementasi Algoritma *Sequential Search*

Pada tahap ini berisi beberapa proses, yaitu :

1. Mengubah file .json menjadi *array*
2. Menyimpan hasil pencarian ke-n (data ke-n)
3. Melakukan pencarian proses selanjutnya menggunakan data ke-(n-1)
4. Melanjutkan proses pencarian hingga kolom ciri – ciri tanaman terakhir
5. Melakukan pengujian terhadap hasil pencarian dengan jumlah pencarian yang dilakukan. Hasil pengujian berupa persentase hasil benar dibagi jumlah berapa kali sistem dijalankan.
6. **Waktu dan Tempat Penelitian**

Penelitian tentang Penerapan Algoritma Pencarian Sequential Search Pada Rancang Sistem Ensiklopedia Tanaman Tingkat Tinggi Berdasarkan Morfologi Daun akan dilaksanakan selama 5 bulan, terhitung dari bulan Januari – Mei 2021.

**Tabel 3.1** Jadwal pelaksanaan penelitian

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No | Kegiatan | Januari | Februari | Maret | April | Mei |
| 1 | Pengumpulan data |  |  |  |  |  |
| 2 | Implementasi algoritma |  |  |  |  |  |
| 3 | Pengujian algoritma |  |  |  |  |  |
| 4 | Pembuatan sistem |  |  |  |  |  |
| 5 | Pengujian sistem |  |  |  |  |  |

Serta peneliti akan melakukan proses pengumpulan data uji pada serangkaian tempat dan lokasi dibawah ini :

1. Toko Bunga dan Bibit Sayuran Pak Nardi, Jl. Mahakam No. 117, Tanjungsari, Kec. Sukorejo, Kota Blitar, Jawa Timur 66122
2. Karilla Florist, Jl. Bengawan solo gg. 6 No. 19, Pakunden, Kec. Sukorejo, Kota Blitar, 66122
3. **Obyek penelitian**

Algoritma *Sequential* atau *Linear search* merupakan obyek utama pada penelitian utama pada penelitian ini, serta *indexing* akan menjadi obyek penelitian kedua yang akan dikombinasikan dengan *Sequential* atau *Linear search* agar algoritma dapat berjalan sesuai apa yang diharapkan oleh peneliti baik dari segi efisiensi waktu dan kompleksitas algoritma.

1. **Teknik pengumpulan data**

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

* 1. Melakukan pendataan tanaman pada aplikasi Microsoft excel.
  2. Mengisikan kolom ciri tanaman yang telah didata dengan ciri – ciri yang terdapat pada buku rujukan. Ciri – ciri helaian daun tumbuhan terdiri dari 10 kolom yang berisi data *string*.
  3. Dataset tanaman yang telah mencapai batas yang ditentukan lalu disimpan dalam bentuk .csv lalu
  4. ubah menjadi dalam bentuk .json (*Javascript Object Notation*) untuk selanjutnya dapat diproses oleh sistem.

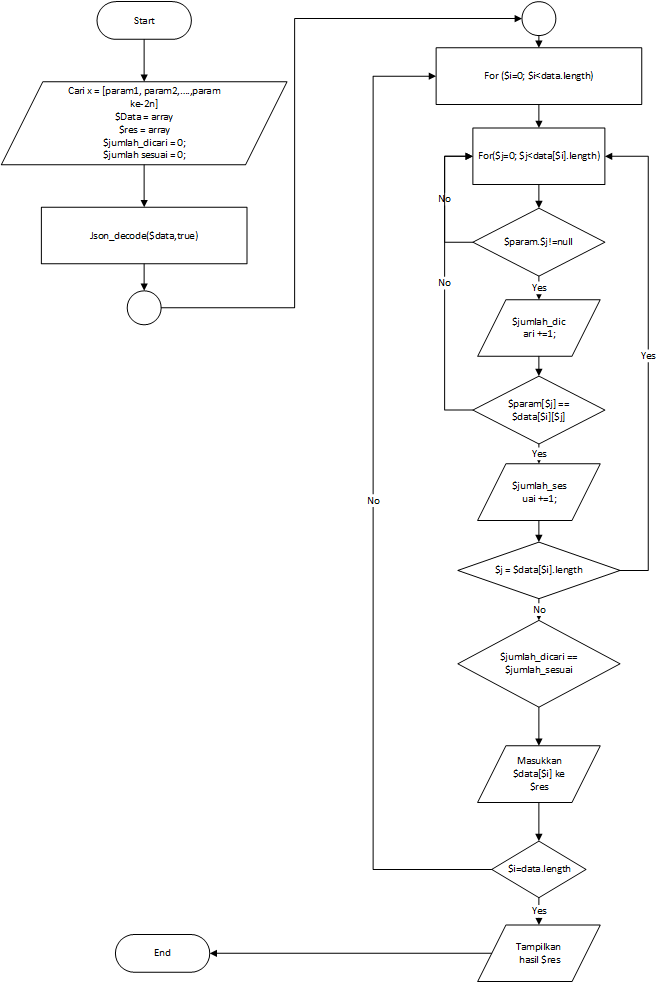
1. **Metode yang diusulkan**

Pada penelitian ini, metode yang diusulkan yaitu metode pencarian data *Sequential Search*. Metode Sequential Search atau disebut pencarian beruntun yang dapat digunakan untuk melakukan pencarian data, baik pada array yang sudah terurut maupun belum terurut. Proses pencarian Sequential Search adalah dengan cara mencocokkan data yang akan dicari dengan semua data yang ada dalam kelompok data secara beruntun mulai dari data pertama sampai data yang dicari ditemukan (Suhartini dkk., 2018).



**Gambar 3.1** Diagram Alur *Sequential Search* pada satu deret data

**Gambar 3.2** Diagram Alur*Sequential Search* pada pencarian tiap kolom



**Gambar 3.2** Diagram Alur*Sequential Search* pada pencarian tiap baris

Diagram alur pada gambar 3.1 menerangkan alur linear search pada keadaan data dalam bentuk larik satu dimensi dengan jumlah data sejumlah n data dikali 10 kolom. Alur yang dijalankan adalah sebagai berikut :

1. Data ke-1 merupakan data tanaman secara keseluruhan yang diproses sedemikian rupa sehingga berbentuk larik satu dimensi dengan indeks sebanyak *n* x10 kolom. Contoh : 100 data memiliki indeks larik sejumlah 100x10 yaitu 999 (karena larik dimulai dari index 0).
2. Pencarian dilakukan dengan batasan sejumlah 10 kolom yang dibatasi atau diatur oleh sistem, bukan dibatasi oleh baris atau kolom dari suatu larik itu sendiri.
3. Proses yang dilakukan adalah minimal sebanyak 10 x *n*. Proses diatur sehingga tidak akan berhenti sebelum semua indeks dari larik terlampaui, dengan dasar bahwa terdapat kemungkinan bahwa di akhir larik bisa saja terdapat data yang dicari oleh sistem.
4. Tidak terdapat eliminasi data seperti pada flowchart pada gambar 3.2.

Diagram alur pada gambar 3.2 menerangkan alur dari sistem yang akan dikembangkan oleh peneliti. Alur yang dijalankan adalah sebagai berikut.

1. Data ke-1 merupakan data tanaman secara keseluruhan(*n*).
2. Proses *indexing* dapat diletakkan pada urutan kedua, atau ketika data sedang pada tahap *preprocessing*. Proses index hanya dapat dilakukan pada satu kolom dari suatu dataset, karena dataset hanya dapat diurutkan berdasarkan satu urutan saja.
3. Proses pencarian melibatkan kolom ke 1 hingga 10 dari dataset tanaman. Jadi jumlah pencarian data maksimal adalah jumlah dataset dikalikan 10 kolom serta proses maksimal yang dilakukan oleh sistem adalah 10 kali proses pencarian yang melibatkan (*n*) data.
4. Hasil pencarian ke-*n* disimpan dalam bentuk deret(*array*) data.
5. Pada proses pencarian ke-(*n*+1), data yang digunakan adalah data ke-(*n*-1). Tiap proses menggunakan data yang dihasilkan dari proses pencarian sebelumnya. Proses ke-*n* juga akan mengambil data hasil dari proses sebelumnya ketika parameter pencarian berisi *null* atau tidak diisi.
6. Hasil dari pencarian ke-10 merupakan hasil akhir dari sistem
7. **Pengujian sistem**

Pengujian sistem menggunakan metode *white box* untuk mengetahui kesesuaian sistem yang telah dibuat dengan diagram alur yang telah didesain sebelumnya tanpa mempertimbangkan *UI* dan *UX* dari sistem tersebut. Menurut Pressman (2010:485), white-box testing, atau yang disebut glass-box testing, adalah metode perancangan test case yang menggunakan penjelasan struktur kontrol sebagai bagian dari component-level design untuk membuat test cases. Dengan menggunakan metode white-box testing, dengan melihat internal kode program tersebut *software engineer* dapat membuat test case yang

1. menjamin semua jalur indepeden di dalam modul telah dieksekusi sekurangnya sekali,
2. menguji semua keputusan logikal pada nilai true dan false,
3. menjalankan semua perulangan pada batasannya dan dalam batas operasionalnya, dan
4. menguji struktur data internal untuk memastikan kebenarannya.

Sedangkan pengujian dari beberapa algoritma pencarian yang terdapat dalam penelitian akan menggunakan perbandingan yang meliputi waktu komputasi ( tahap *preprocessing* data dan proses pencarian data ) dan tingkat kerumitan algoritma (jumlah variabel dan proses yang terdapat pada komputasi ).

**Tabel 3.1** Kategori keberhasilan system

|  |  |
| --- | --- |
| no | keberhasilan |
| 1 | System mampu membaca file .json |
| 2 | System mampu mengkonversi file .json menjadi array |
| 3 | System mampu membaca semua baris dan kolom data pada array |
| 4 | System mampu membaca kolom dan menyimpannya sebagai baris |
| 5 | System mampu memberikan kondisi bahwa data sesuai atau tidak |
| 6 | Sistem mampu menyimpan baris yang benar kedalam array |
| 7 | System mampu menampilkan hasil akhir pencarian |

**Tabel 3.2** contoh tabel perbandingan algoritma

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritma | Preprocessing (ms) | Pencarian (ms) | variable | proses |
| Sequential search larik 1 dimensi | *n* ms | *n* ms | *x* variabel | *x* proses |
| Sequential search larik 2 dimensi | *n* ms | *n* ms | *x* variabel | *x* proses |
| *Indexed-sequential search* pada larik 2 dimensi | *n* ms | *n* ms | *x* variabel | *x* proses |

**BAB IV**

**PEMBAHASAN**

1. *Sequential search*

Seperti sequential search pada umumnya, proses pencarian dilakukan pada satu deret array 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 system adalah sebagai berikut,

|  |  |  |
| --- | --- | --- |
| Kolom 1 | Kolom n | Kolom n+1 |
| 1,1 | 1,2 | 1,3 |
| 2,1, | 2,2 | 2,2 |
| 3,1 | 3,2 | 3,3 |

Keterangan :

1. Simbol panah biru melambangkan proses dilakukan terlebih dahulu
2. Simbol panah merah melambangkan proses dilakukan terlebih setelah panah biru

Mekanisme yang dilakukan :

1. Menyimpan data secara keseluruhan pada variable tertentu.
2. Mendefinisikan variable :
   1. *x*, adalah integer bernilai awal nol. Ketika perbandingan bernilai benar maka *x+=1*.
   2. *y*, adalah integer bernilai awal nol. Ketika parameter tidak null atau kosong *y+=1*.
   3. Melakukan perulangan dengan variable *q*. *q* bernilai awal nol, *q* kurang dari jumlah baris seluruh data, *q+=1*.
   4. *z*, adalah variable dengan type array. Variabel ini akan digunakan untuk menyimpan hasil dari pencarian.
3. Mengambil deret pertama dari suatu dataset dan menyimpan key/index dari deret tersebut kedalam suatu variable dinamis.
4. Melakukan perulangan sebanyak jumlah data yang terdapat pada proses ke-1.
5. Didalam perulangan juga akan dilakukan perulangan kedua untuk melakukan pencarian tiap – tiap kolom dari suatu baris data dari proses perulangan diatasnya.
6. Proses pencarian dilakukan dengan cara membandingkan tiap – tiap nilai dari index suatu deret dengan parameter pencarian yang diurutkan dengan menggunakan perulangan, misal index ke – i akan dibandingkan dengan parameter ke – i.
7. Ketika nilai *x*==*y,* maka seluruh data dari array ke – i akan disimpan pada variabel z.
8. *z* merupakan variable terakhir yang menjadi hasil akhir dari proses diatas
9. Jumlah data dari proses pencarian ini akan semakin melebar, misal proses ke-i+1 akan memiliki data yang lebih banyak dari proses ke-i.

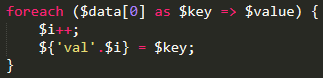
Proses yang dilakukan pada penerapan algoritma diatas :

1. Mengambil data secara keseluruhan



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

1. Mengambil nama kolom dari data



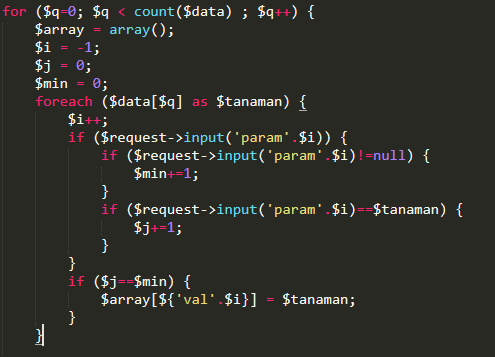
Pada proses ini, kita akan membuat variable yang berisi key dari suatu baris data. Key tersebut akan digunakan sebagai parameter pencarian pada proses selanjutnya.

1. Mendefinisikan *array* untuk menyimpan hasil pencarian



Variable dengan type 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.

1. Melakukan perulangan sebagai proses pencarian tiap baris dari suatu dataset



Keterangan :

* 1. Perulangan *for* digunakan untuk membaca semua baris dari dataset yang didefinisikan pada poin 1.
  2. *Foreach* digunakan untuk membaca tiap kolom dari suatu baris data pada proses poin a.
  3. $min adalah variable yang digunakan sebagai batas minimal baris data dapat ditentukan sebagai benar atau tidak. Nilai $min akan ditambah 1 ketika parameter tidak kosong atau bernilai.
  4. $j adalah variable yang digunakan sebagai nilai benar suatu baris data terhadap parameter yang dicari. Apabila *value* dari suatu kolom sesuai dengan parameter yang sedang dicari, maka $j+=1.
  5. Ketika $j==$min atau $j senilai dengan $min, maka seluruh kolom dari suatu baris data akan dimasukkan pada $final\_array

1. Menampilkan hasil pencarian



Hasil

1. Pencarian pada baris ke – 1



1. Pencarian pada baris ke – n



Keterangan :

Baris dengan warna hitam merupakan hasil yang didapat pada proses ke – n yang ditambahkan kedalam dataset hasil proses ke – n-1.

1. *Sequential search* dua dimensi

Sequential search yang mulanya dilakukan secara horizontal, pada modifikasi pertama dari sequential search ini, pencarian dilakukan secara vertikal, disebut vertikal karena pencarian ini apabila diilustrasikan akan berpola seperti pencarian dari atas ke bawah pada satu persatu kolom.

|  |  |  |
| --- | --- | --- |
| Kolom 1 | Kolom n | Kolom n+1 |
| 1,1 | 1,2 | 1,3 |
| 2,1, | 2,2 | 2,2 |
| 3,1 | 3,2 | 3,3 |

Keterangan :

1. Simbol panah biru melambangkan proses dilakukan terlebih dahulu
2. Simbol panah merah melambangkan proses dilakukan terlebih setelah panah biru

Mekanisme yang dilakukan :

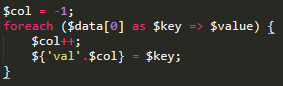
1. Meyimpan data keseluruhan pada suatu variable a
2. Mendefinisikan variabel :
   1. Max = jumlah kolom dari baris pertama variabel pada poin 1
3. Melalukan perulangan sebanyak variabel max.
4. Membuat variable b untuk menyimpan hasil pencarian.
5. Ketika iterasi bernilai 0, atau iterasi pertama, data yang digunakan adalah data dari poin 1.
6. Sistem akan menerapkan kondisi ketika parameter pencarian bernilai kosong atau null, maka sistem akan menyimpan data dari hasil sebelumnya atau iterasi i-1 pada variabel b, namun ketika parameter tidak kosong atau null maka proses berlanjut
7. Proses pencarian dilakukan dengan membandingkan semua kolom ke-I dengan parameter pencarian ke-I dan diulangi hingga iterasi I mencapai max. contoh : pada proses pertama, pencarian hanya akan melakukan pencarian pada kolom ke-i saja dengan parameter pencarian ke-i, tanpa melakukan pencarian pada kolom dan parameter pencarian lainnya.
8. Ketika nilai dari kolom senilai dengan parameter pencarian maka satu deret data akan disimpan pada array hasil.
9. Proses pencarian berlanjut ke kolom selanjutnya atau kolom ke - i+1.
10. Proses pencarian ke – i akan menggunakan data dari proses ke – i-1.
11. Jumlah data dari proses pencarian ini akan semakin mengerucut, misal proses ke-i akan memiliki data yang lebih banyak dari proses ke-i+1.

Proses yang dilakukan pada penerapan algoritma diatas :

1. Mengambil data secara keseluruhan



1. Mengambil nama kolom dari data pertama



1. 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.



Notasi pada suatu pemanggilan index suatu array



Keterangan :

* 1. ${‘a’.$i} adalah value dari suatu proses foreach, dan *i* adalah nama kolom yang sedang dicari. Contoh kasus, Ketika *i* = 3, maka notasi diatas dapat dibaca $a3 atau variable dengan nama a3.
  2. [${‘val’}.$i] adalah nama index atau key dari suatu *array* asosiatif, dan *i* adalah nama kolom yang sedang dicari.

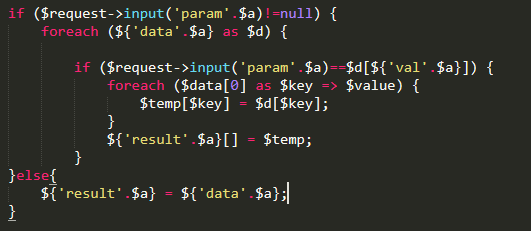
1. Mendefinisikan array untuk proses ke-n



1. 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.



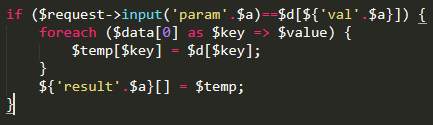
1. Perkondisan untuk menentukkan nilai parameter pencarian



Keterangan :

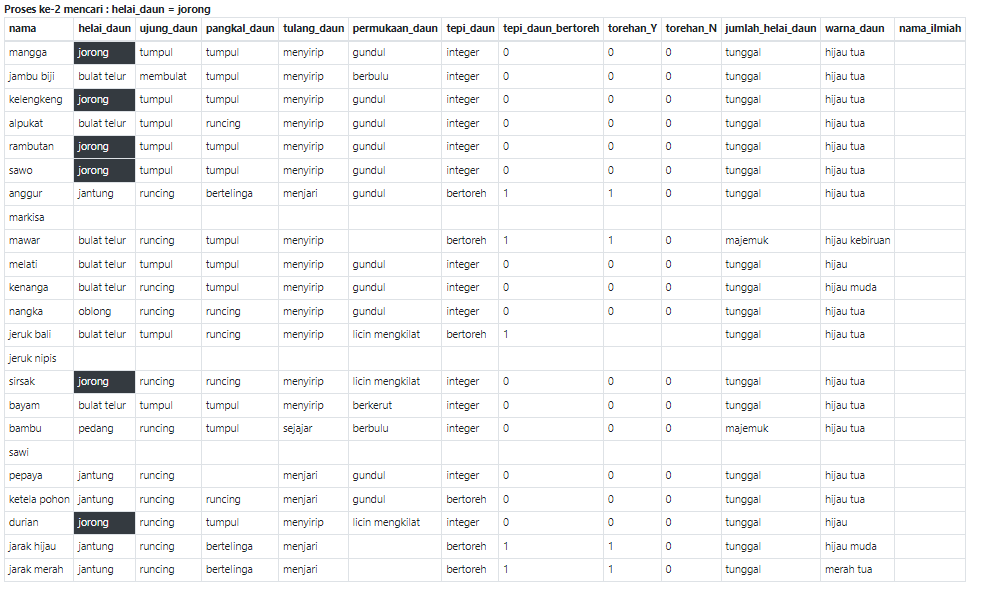
* 1. Perkondisian if untuk menentukan apakah parameter pencarian tidak kosong atau tidak samadengan *null*. Jika *null*, maka pada proses hasil dari proses ke n adalah data ke – n.
  2. Jika parameter pencarian tidak samadengan null, maka terjadilah proses pencarian semua baris dari kolom ke – n.

1. Jika nilai pada kolom ke – n samadengan nilai parameter pencarian, maka satu baris data akan disimpan pada variable *temp* sebelum disimpan kedalam variable dengan tipe data *array* pada poin 4.

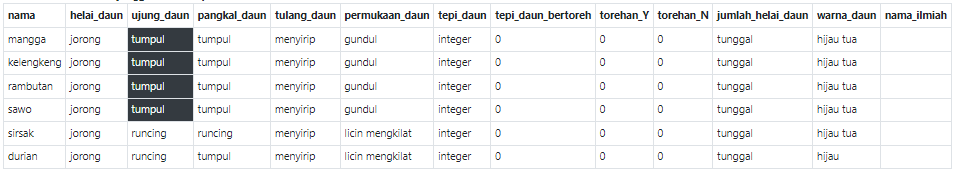


Hasil

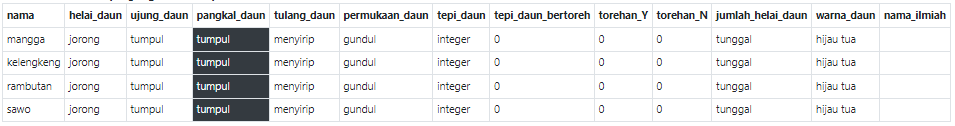
1. Proses pencarian ke – n



1. Proses pencarian ke – n+1



1. Proses pencarian ke – n+2

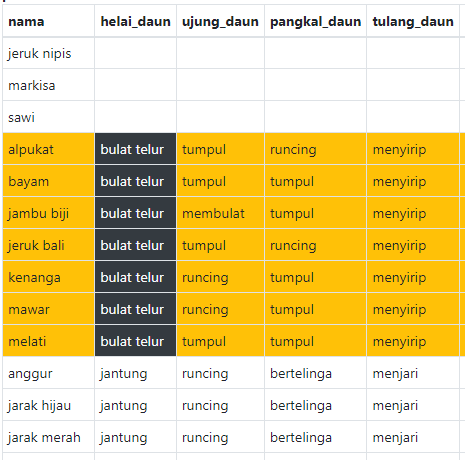


1. *Sequential search* dua dimensi dengan *indexing*

Konsep dari sequential search dengan index adalah sama dengan poin 4.3 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. Contoh : proses ke – 1 adalah pencarian berdasarkan nama, maka dataset tersebut akan diurutkan berdasarkan nama dengan urutan *ascending* atau dari besar ke kecil.

Mekanisme yang dilakukan

1. Meyimpan data keseluruhan pada suatu variable a
2. Mendefinisikan variabel :
   1. Max = jumlah kolom dari baris pertama variabel pada poin 1
3. Melalukan perulangan sebanyak variabel max.
4. Membuat variable b untuk menyimpan hasil pencarian.
5. Ketika iterasi bernilai 0, atau iterasi pertama, data yang digunakan adalah data dari poin 1.
6. Sistem akan menerapkan kondisi ketika parameter pencarian bernilai kosong atau null, maka sistem akan menyimpan data dari hasil sebelumnya atau iterasi i-1 pada variabel b, namun ketika parameter tidak kosong atau null maka proses berlanjut
7. Melakukan sorting berdasarkan value dari array. Kolom yang dijadikan patokan sorting adalah kolom ke – i atau nama kolom yang sedang berjalan pada iterasi perulangan for pada poin 3.
8. Merubah semua value dan parameter pencarian menjadi lowercase.
9. Proses pencarian dilakukan dengan membandingkan semua kolom ke-I dengan parameter pencarian ke-I dan diulangi hingga value dari kolom yang dicari bernilai lebih dari parameter pencarian. Pencarian akan berhenti ketika nilai ascii suatu value dari array telah melebihi nilai ascii dari parameter pencarian.



Keterangan :

1. Pada baris berwarna putih yang berada sebelum baris warna oranye, nilai ascii dari karakter ‘b’ pada string ‘bulat telur’ adalah masih lebih besar daripada null.
2. Pada baris berwarna putih yang berada setelah baris warna oranye, nilai ascii dari huruf ‘j’ pada string ‘jantung’ adalah bernilai lebih tinggi daripada dari karakter ‘b’ pada string ‘bulat telur’
3. Ketika nilai dari kolom senilai dengan parameter pencarian maka satu deret data akan disimpan pada array hasil.
4. Proses pencarian berlanjut ke kolom selanjutnya atau kolom ke - i+1.
5. Proses pencarian ke – i akan menggunakan data dari proses ke – i-1 yang diurutkan berdasarkan kolom ke - i+1.
6. Jumlah data dari proses pencarian ini akan semakin mengerucut, misal proses ke-i akan memiliki data yang lebih banyak dari proses ke-i+1.

**DAFTAR RUJUKAN**

Andriani, Sofyah. (2017). *Tinjauan Yuridis Sosiologis Perlindungan Hukum Terhadap Pemegang Hak Perlindungan Varietas Tanaman (Pvt)*. *Penulisan Hukum Fakultas Hukum Universitas Muhammadiyah Malang.*

Latifa, Roimil. (2015). *Karakter Morfologi Daun Beberapa Jenis Pohon Penghijauan Hutan Kota Di Kota Malang.* *Prosiding Seminar Nasional Pendidikan Biologi 2015, yang diselenggarakan oleh Prodi Pendidikan Biologi FKIP Universitas Muhammadiyah Malang, tema: “Peran Biologi dan Pendidikan Biologi dalam Menyiapkan Generasi Unggul dan Berdaya Saing Global”, Malang, 21 Maret 2015.*

*Paramartha, A. G. Y., Suryaningsih, G. K., & Aryanto, K. Y. E. (2017). Implementasi Web Service pada Sistem Pengindeksan dan Pencarian Dokumen Tugas Akhir, Skripsi, dan Praktik Kerja Lapangan. JST (Jurnal Sains dan Teknologi), 5(2).*

Agmalaro, M. A., Kustiyo, A., & Akbar, A. R. (2013). *Identifikasi Tanaman Buah Tropika Berdasarkan Tekstur Permukaan Daun Menggunakan Jaringan Syaraf Tiruan Tropical Fruit Plants Identification Based on Leaf Surface Texture Image Using Artificial Neural Network*. *2*(Ashari 2006).

Al-husseini, K. A. O., & Obaid, A. H. (2018). Usage of Prototyping in Software Testing. *Multi-Knowledge Electronic Comprehensive Journal For Education And Science Publications*, *November*.

Kamariah Tambunan. (2012). Tesaurus Bioteknologi: Sebagai Alat Bantu Pengindeksan Dokumen. *Baca: Jurnal Dokumentasi Dan Informasi*, *33*(2). https://doi.org/http://dx.doi.org/10.14203/j.baca.v33i2.99

Muhazir, A., Fakhriza, M., & Sutejo, E. (2017). Implementasi Metode Sequential Dalam Pencarian Pendistribusian Barang pada Cargo Integration Sistem. *Jurnal Dan Penelitian Teknik Informatika*, *2*(2), 24–30.

Raghuvanshi, D. (2018). Data Structure: Theoretical Approach. *International Journal of Trend in Scientific Research and Development*, *Volume*-*3*(Issue-1), 268–273. https://doi.org/10.31142/ijtsrd18977

Rosid, M. A. (2017). Implementasi JSON untuk Minimasi Penggunaan Jumlah Kolom Suatu Tabel Pada Database PostgreSQL. *JOINCS (Journal of Informatics, Network, and Computer Science)*, *1*(1), 33. https://doi.org/10.21070/joincs.v1i1.802

Salamun, Rahmalina, W., & Jusman, Y. (2017). POWER SEARCH DALAM MENGGUNAKAN SEARCH ENGINE. *Jurnal Pengabdian Masyarakat Vol.1 No.1 Oktober 2017*, *1*(1), 65–71.

Sihombing, J. (2020). *Penerapan Stack Dan Queue Pada Array Dan Linked List Dalam Java*. *June*, 15–24.

Suhartini, Muchlis, & Lestari, R. P. (2018). Implementation of Sequential Search Method on Android-based Jakabaring Dictionary. *Jurnal Transformatika*, *16*(1), 74. https://doi.org/10.26623/transformatika.v16i1.830

Utami, M., & Apridiansyah, Y. (2019). Implementasi Algoritma Sequential Searching Pada Sistem Pelayanan Puskesmas Menggunakan Bootstrap (Studi Kasus Puskesmas Kampung Bali Bengkulu). *JSAI (Journal Scientific and Applied Informatics)*, *2*(1), 81–86. https://doi.org/10.36085/jsai.v2i1.166

Department of Paleobiology Smithsonian Institution. (1999). Manual of Leaf Architecture – morphological description and category of dicotyledonous and net-veined monocotyledonous angiosperm by Leaf Architecture Working Group 65p.

Wikipedia contributors. (2021a, January 5). *Glossary of leaf morphology*. Wikipedia. https://en.wikipedia.org/wiki/Glossary\_of\_leaf\_morphology

Wikipedia contributors. (2021b, March 1). *Leaf*. Wikipedia. https://en.wikipedia.org/wiki/Leaf