Rancangan Aplikasi Pencarian File pada Direktori Windows Dengan Menggunakan Metode Breadth First Search

Krisman Sitompul [1], Andy Paul Harinja[2]

[1,2,] *Fakultas Ilmu Komputer Universitas Katolik Santo Thomas Medan, Indonesia*

ARTICLE INFORMATION

Received: April, 19, 2021
Revised: April 22, 2021
Available online: April,28,2021

KEYWORDS

Pencarian, Search Engine, BreadthFirst Search (BFS), File Direktori

CORRESPONDENCE

E-mail: krisman_tompul@gmail.com[1]
apharianja@gmail.com[2]

A B S T R A C T

The current technological developments with the facilities provided are sufficient to motivate many parties to be able to find and obtain the maximum possible information, even as short as possible. This is what makes computer users to search or search the data / documents needed by using search engine facilities. However, very often the search results are unrelated, making it difficult to find the information needed. Searching is a process that is often used in data management. The search process is to find a certain value (data) in a set of data of the same type (either the base type or the form type). Data can be stored temporarily in main memory or stored permanently in secondary memory (tape or disk). In main memory, the common data storage structure is in the form of an array or table (array), while in secondary memory it is an archive (file). The search algorithm that will be discussed starts with one of the search algorithms that can be applied in a search in a directory, namely Breadth First Search (BFS).

PENDAHULUAN

Kemajuan komputer saat ini dilengkapi dengan fasilitas yang diberikan cukup memotivasi banyak pihak untuk dapat mengolah dan mendapatkan File semaksimal mungkin bahkan sesingkat mungkin. Hal inilah yang membuat para pengguna komputer untuk melakukan searching atau pencarian terhadap data/dokumen yang diperlukan dengan menggunakan fasilitas search engine. Namun sering sekali hasil pencariannya tidak memiliki hubungan sehingga sulit untuk menemukan File yang diperlukan.

Dokumen apapun bentuknya tentu disimpan dengan sebuah metode tertentu, dengan harapan jika dikemudian hari data/dokumen yang tekandung di dalamnya diperlukan maka cukup dengan melakukan pencarian, data yang diinginkan akan didapatkan dengan cepat. Namun semakin banyak dokumen yang disimpan maka waktu pencarian juga akan meningkat ditambah lagi dengan hasil pencarian yang tidak tepat. Hal ini disebabkan oleh banyaknya dokumen yang harus dipilih dan diteliti relevansinya dengan subjek yang dicari.

Algoritma pencarian (searching algorithm) adalah algoritma yang menerima sebuah argumen kunci dan dengan langkah-langkah tertentu akan mencari rekaman dengan kunci tersebut. Setelah proses pencarian dilaksanakan, akan diperoleh salah satu dari dua kemungkinan, yaitu data yang dicari ditemukan (successful) atau tidak ditemukan (unsuccessful). Metode pencarian data dapat dilakukan dengan dua cara yaitu, pencarian internal(internal searching) dan pencarian eksternal (external searching). Pada pencarian internal, semua rekaman yang diketahui berada dalam pengingat komputer sedangkan pada pencarian eksternal, tidak semua rekaman yang diketahui berada dalam pengingat komputer, tetapi ada sejumlah rekaman yang tersimpan dalam penyimpan luar misalnya pita atau cakram magnetis[1].

Breadth First Search(BFS) adalah suatu teknik yang digunakan untuk mencari data dalam sebuah berkas tertentu dalam sebuah File. Dengan adanya teknik searching ini maka jika ingin mencari salah satu data dari data yang sangat banyak dengan cara manual akan membutuhkan waktu yang sangat lama, tetapi dengan menggunakan metode Breadth First Search(BFS) maka pencarian tersebut akan lebih cepat[2] .

Berdasarkan keterangan di atas maka penulis tertarik untuk membuat sebuah program yang dapat membantu pengguna melakukan pencarian File tertentu tanpa harus mencari dengan cara manual yang mana dalam pencarian File tersebut menggunakan metode Breadth First Search(BFS) untuk mencari data yang akan dicari sehingga akan lebih muda dan lebih cepat.

BAHAN DAN METODE

Dalam penulisan penelitian ini, penulis melakukan beberapa hal untuk mendapatkan data yang diperlukan, antara lain:

1. Metode Pengumpulan Data
   Beberapa metode pengumpulan data yang dilakukan oleh penulis yaitu:
   a. Studi kepustakaan (*library search*)
      Untuk mendapatkan hasil teori yang valid untuk dijadikan sebuah landasan, penulis mencari beberapa buku referensi dari beberapa perpustakaan seperti mencari pembahasan penulis.
   b. Pengumpulan data melalui *surfing* (*field research*)

Pencarian atau penjelajahan untuk mencari data yang dapat dijadikan landasan penulis yang sesuai melalui internet, seperti mencari File artikel yang membahas masalah Breadth First Search(BFS).

2. Metode Perancangan Sistem
   a. Analisis Kebutuhan
      Analisis kebutuhan adalah yaitu analisa Metode *Breadth First Search(BFS)* yang dilakukan untuk menentukan input dan output yang diinginkan berdasarkan rumus yang di ada.
   b. Analisa dan Perancangan Sistem
      Perancangan sistem merupakan tahapan yang dilakukan untuk membuat sebuah rancangan program berdasarkan input dan output yang diinginkan.
   c. Implementasi Sistem
      Setelah pembuatan perancangan sistem maka langkah selanjutnya adalah mengimplementasi hasil perancangan ke dalam program.
   d. Evaluasi Sistem
      Evaluasi merupakan langkah setelah *Breadth First Search(BFS)* dimplementasikan untuk mengetahui kesalahan atau trouble yang mungkin terjadi, sampai dipastikan sistem dapat berjalan dengan sempurna.
   e. Penulisan laporan penelitian
      Ini adalah tahap akhir dari penelitian .

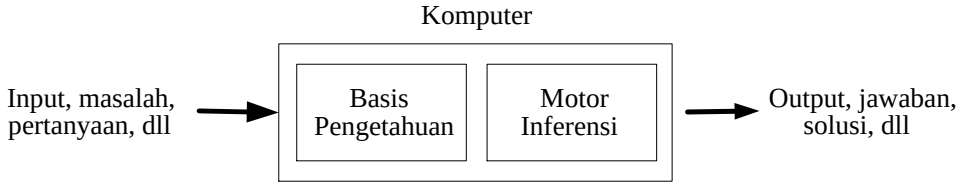
2.1. Pengertian AI

Artificial Intelligence (AI) disebut juga dengan kecerdasan buatan. AI merupakan salah satu bagian ilmu komputer yang mempelajari tentang bagaimana cara membuat agar komputer dapat melakukan pekerjaan seperti yang dilakukan oleh manusia[3]. Tujuan dari AI adalah untuk memecahkan persoalan dunia nyata (bersifat praktis) dan memahami intelijensia (bersifat memahami). AI merupakan salah satu bagian ilmu komputer yang mempelajari tentang bagaimana cara membuat agar komputer dapat melakukan pekerjaan seperti yang dilakukan oleh manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan zaman, maka peran komputer semakin mendominasi kehidupan umat manusia. Komputer tidak lagi hanya digunakan sebagai alat hitung, lebih dari itu, komputer diharapkan untuk dapat diberdayakan untuk mengerjakan segala sesuatu yang bisa dikerjakan oleh manusia. Manusia bisa menjadi pandai dalam menyelesaikan segala permasalahan di dunia ini karena manusia mempunyai pengetahuan dan pengalaman. Pengetahuan diperoleh dari belajar. Semakin banyak bekal pengetahuan yang dimiliki oleh seseorang tentu saja diharapkan akan lebih mampu dalam menyelesaikan permasalahan. Namun bekal pengetahuan saja tidak cukup, manusia juga diberi akal untuk melakukan penalaran, mengambil kesimpulan berdasarkan pengetahuan dan pengalaman yang mereka miliki. Tanpa memiliki kemampuan menalar yang baik, manusia dengan segudang pengalaman dan pengetahuan tidak akan dapat menyelesaikan masalah dengan baik. Demikian pula, dengan kemampuan menalar yang sangat baik, namun tanpa bekal pengetahuan dan pengalaman yang memadai, manusia juga tidak akan bisa menyelesaikan masalah dengan baik[3].

Komputer juga harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar, agar komputer bisa bertindak seperti dan sebaik manusia. Untuk itu pada *artificial intelligence*, akan mencoba untuk memberikan beberapa metode untuk membekali komputer dengan kedua komponen tersebut agar komputer bisa menjadi mesin yang pintar. Untuk menciptakan aplikasi kecerdasan buatan ada 2 bagian utama yang sangat dibutuhkan [4], yaitu:

1. Basis Pengetahuan (*Knowledge Base*)
2. Motor Inferensi (*Inference Engine*)

Basis pengetahuan merupakan inti dari suatu sistem pakar, yaitu berupa representasi pengetahuan dari pakar. Basis pengetahuan tersusun atas fakta dan kaidah. Fakta adalah informasi tentang objek, peristiwa, atau situasi. Kaidah adalah cara untuk membangkitkan suatu fakta baru dari fakta yang sudah diketahui.Dalam prosesnya,seperti yang terlihat pada gambar II.1, mesin inferensi menggunakan strategi penalaran dan strategi pengendalian. Strategi penalaran terdiri dari strategi penalaran pasti (Exact Reasoning) dan strategi penalaran tak pasti (Inexact Reasoning). Exact reasoning akan dilakukan jika semua data yang dibutuhkan untuk menarik suatu kesimpulan tersedia, sedangkan inexact reasoning dilakukan pada keadaan sebaliknya.Strategi pengendalian berfungsi sebagai panduan arah dalam melakukan prose penalaran.



Gambar 1. Sistem yang menggunakan AI [4]

2.2. Soft Computing

Soft computing adalah koleksi dari beberapa metodologi yang bertujuan untuk mengeksploitasi adanya toleransi terhadap ketidaktepatan, ketidakpastian dan kebenaran parsial untuk dapat diselesaikan dengan mudah[5], *robustness*, dan biaya

penyelesaiannya murah. Definisi ini pertama kali diungkapkan oleh Prof. Lotfi A. Zadeh pada tahun 1992. *Soft computing* merupakan inovasi baru dalam membangun sistem cerdas. Sistem cerdas ini merupakan sistem yang memiliki keahlian seperti manusia pada domain tertentu, mampu beradaptasi dan belajar agar dapat bekerja lebih baik jika terjadi perubahan lingkungan. Unsur-unsur pokok dalam *Soft Computing*, adalah:

1. *Sistem Fuzzy* (mengakomodasi ketidaktepatan).
2. Jaringan Syaraf (menggunakan pembelajaran).
3. *Probabilistic Reasoning* (mengakomodasi ketidakpastian).
4. *Evolutionary Computing* (optimasi).

Keempat unsur tersebut bukan merupakan pesaing antara satu dengan lainnya, namun diantaranya bisa saling melengkapi. Bahkan, pada kenyataannya, biasanya unsur-unsur pokok tersebut akan digunakan secara sinergis ketimbang dikerjakan secara sendiri-sendiri. Sehingga, Zadeh juga mendefinisikan bahwa *soft computing* itu merupakan hubungan antara logika *fuzzy*, *neuro-computing*, *probabilistic reasoning*, dan algoritma genetik. Tabel 1 menunjukkan hubungan antara *fuzzy*/*probabilistic-reasoning*, jaringan syaraf tiruan dan AI konvensional.

Tabel 1. Hubungan Antara Unsur-Unsur Pokok dalam Soft Computing[6]

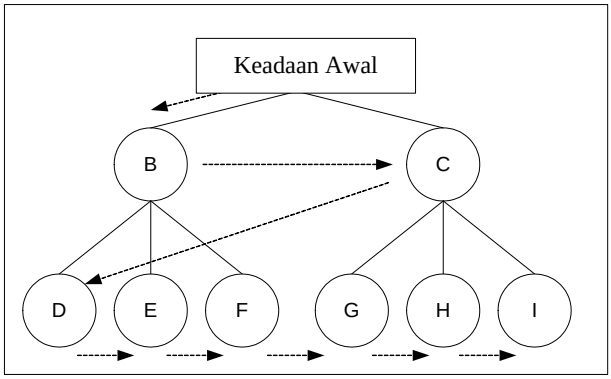
	Pembelajaran	**Ekstraksi Pengetahuan**	**Operasi real-time**	**Representasi Pengetahuan**	**Optimasi**
Fuzzy/probabilistic reasoning	Tidak	Ya	Ya	simbolik/ numerik	Tidak
Jaringan Syaraf Tiruan	Ya	Tidak	Ya	numerik	Tidak
Sistem Evolusioner	Ya	Tidak	Tidak	numerik	Ya
Sistem AI Konvensional	Tidak	Ya	Tidak	simbolik/ numerik	Tidak

Karakteristik *Soft Computing*:

a. *Soft computing* memerlukan keahlian manusia, apabila direpresentasikan dalam bentuk aturan (IF-THEN).
b. Mode komputasinya diilhami oleh proses biologis.
c. *Soft computing* merupakan teknik optimasi baru.
d. *Soft computing* menggunakan komputasi numeris.
e. *Soft computing* memiliki toleransi kegagalan (meskipun kualitasnya berangsur-angsur memburuk).

2.3. Pencarian Melebar Pertama (Breadth-First Search)

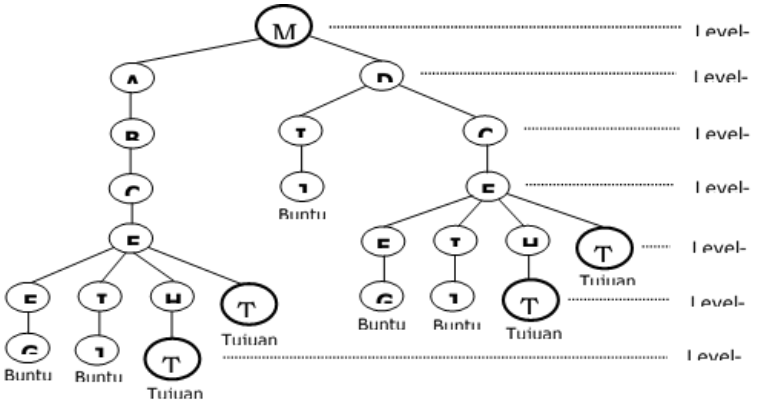
Pada metode ini, semua *node* pada level *n* akan dikunjungi terlebih dahulu sebelum mengunjungi *node-node* pada level *n*+1 (level berikutnya). Pencarian dimulai dari *node* akar terus ke level ke-1 dari kiri ke kanan, kemudian berpindah ke level-2 dan seterusnya. Pohon dikembangkan hingga ditemukannya solusi[2].



Gambar 2. Prosedur Pencarian Melebar Pertama[3], [7]

2.4. Pohon Pelacakan

Untuk menghindari kemungkinan adanya proses pelacakan suatu node secara berulang, maka digunakan struktur pohon.



Gambar 3. Struktur Pohon [3], [4]

Struktur pohon digunakan untuk menggambarkan keadaan secara hirarkis. Pohon juga terdiri-dari beberapa node. Node yang terletak pada level-0 disebut dengan nama "akar". Node akar menunjukkan keadaan awal yang biasanya merupakan topik atau obyek. Node akar ini terletak pada level ke nol. Node akar memiliki beberapa percabangan yang terdiri-atas beberapa node successor yang sering disebut dengan nama "anak" dan merupakan node-node perantara. Namun jika dilakukan pencarian mundur, maka dapat dikatakan bahwa node tersebut memiliki predecessor[8]. Node-node yang tidak memiliki anak sering disebut dengan nama node "daun" yang menunjukkan akhir dari suatu pencarian, dapat berupa tujuan yang diharapkan (*goal*) atau jalan buntu (*dead end*).

HASIL DAN PEMBAHASAN

3.1. Analisa Algoritma Pencarian

Solusi pencarian file menggunakan metode pencarian melebar pertama (*breadth-first search* / BFS), karena solusi dari permasalahan bisa lebih dari satu. Dimulai dari posisi awal sebagai *node* akar, selanjutnya metode BFS mencari solusi dengan mengembangkan *node* akar ke level-level berikutnya, semua pergerakan yang mungkin, tidak melanggar ketentuan dan menghasilkan kondisi baru dikembangkan semaksimal mungkin. Pencarian berakhir apabila tidak ada lagi *node* atau kondisi baru yang dapat dikembangkan. Semua *node* yang merupakan posisi tujuan merupakan solusi.

Tahap analisa sistem dilakukan setelah tahap perencanaan sistem dan sebelum tahap desain. Tahap analisa merupakan tahap yang kritis dan sangat penting, karena kesalahan didalam tahap ini akan menyebabkan juga kesalahan ditahap selanjutnya

BFS Mengubah graph awal menjadi suatu pohon dengan cari sebagai berikut :

1. Pilih induk dari pohon (misal f) pilih yang terhubung dengan sirkuit
2. Pilih titik-titik yang mnghasilkan jalur terpanjang dengan mengabaikan adanya sirkuit
3. (setelah mengambil f kita bisa beralih ke titik yang berada di kanannya dahulu atau di kirinya, missal hubungkan titik f-g-h-k-j)
4. Setelah mencapai ujung jalur, lakukan pengecekan dengan kembali ke titik sebelumnya. Mengecek apakah pada titik tersebut masih mempunyai cabang atau tidak, Jika ada cabang tuliskan titik tersebut sebagai cabang. Kemudian cek lagi cabang tersebut apakah bercabang lagi atau tidak.
5. (setelah mencapai ujung j, kembali lagi ke titik sebelumnya k, lakukan pengecekan, lakukan langkah ini hingga semua titik terdapat dalam satu pohon)

Dan Algoritma dari BFS adalah sebagai berikut ini :

1. Algoritma
2. Step 1: letakkan root parent di tempat yang paling atas
3. Step 2: Lakukan looping hingga titik terakhir
4. Step 3: letakkan titik yang terhubung dengan root parent pada satu garis lurus
5. Step 4: lakukan pengecekan, jika titik tidak memiliki child letakkan pada garis, pindah ke titik selanjutnya
6. Step 5: jika titik yang ditemui memiliki child hubungkan dengan titik sebelumnya

3.2. Contoh Kasus Analisa Pencarian File Dengan Algoritma BFS

Agar lebih jelas, akan digunakan contoh untuk menentukan Graf Ruang Solusinya

Table 2. Contoh Direktori

Contoh Posisi File di Direktori
DIR1 - File-1 - File-2 - File-3 - File-4 - File-5 - File-6

- File-7
- File-8
- File-9
- File-10

DIR2

- File-11
- File-12
- File-13
- File-14
- File-15
- File-16
- File-17

DIR3

- File-21
- File-22
- File-23
- File-24
- File-25
- File-26
- File-27

DIR4

- File-31
- File-32
- File-33
- File-34
- File-35
- File-36
- File-37

DIR5

- File-41
- File-42
- File-43
- File-44
- File-45
- File-46
- File-47

DIR6

- File-51
- File-52
- File-53
- File-54
- File-55
- File-56
- File-57

DIR7

- File-61
- File-62
- File-63
- File-64
- File-65
- File-66
- File-67

Dalam prosedur ini kami mengambil asumsi bahwa setiap node telah merepresentasikan karakter yang diwakilinya, Perubahan dari status ke status (dari node ke anaknya) diasumsikan sebagai operasi string untuk membentuk kata. Kemudian kata tebakan harus lebih kecil dari kata awal.

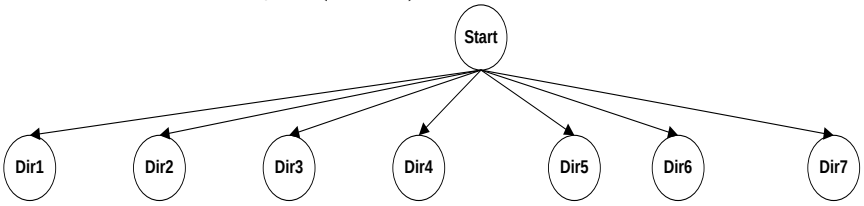
1. Masukkan node awal ke Queue
2. Set Kedalaman dengan 0
3. Ambil kepala Queue
4. Bangkitkan semua anak
5. Jika anak memenuhi syarat push ke Queue
6. Jika memenuhi solusi stop, jika tidak ulangi langkah 3 sampai kedalaman maks
7. Cetak solusi dan perhitungan skor berdasarkan kedalaman

Contoh pencarian

File yang akan dicari = “File44”

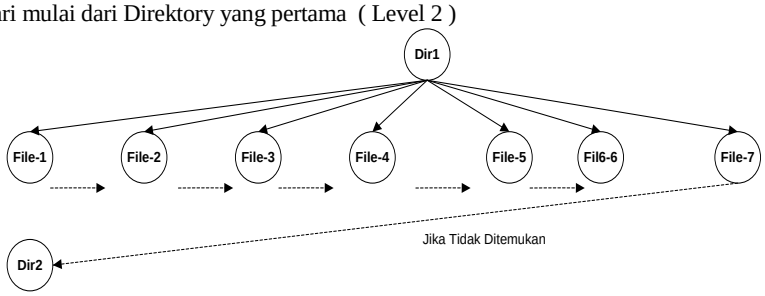
Maka tahap yang dilakukan adalah sebagai berikut :

a. Menset Semua Direktori ke dalam sebuah Queue (Level-1)



Gambar 4. Set Semua Direktory dalam Queue

a. Cek file yang akan dicari mulai dari Direktory yang pertama (Level 2)

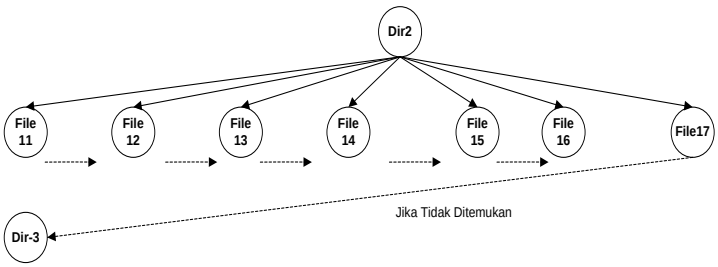


Gambar 5. Proses Pencarian Level 2

Pada tahap level-2 maka dilakukan pencarian pada direktory “DIR1” dengan cara membandingkan apa yang dicari dengan file yang ada didalam direktory tersebut, dan prosesnya dilakukan seperti langkah berikut ini :

- Cek apakah “File-1” = “File yang dicari” ?
  Jika “YA” maka “File-1” adalah “GOL” atau data ditemukan
  Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah “File-2” = “File yang dicari” ?
  Jika “YA” maka “File-2” adalah “GOL” atau data ditemukan
  Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah “File-3” = “File yang dicari” ?
  Jika “YA” maka “File-3” adalah “GOL” atau data ditemukan
  Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah “File-4” = “File yang dicari” ?
  Jika “YA” maka “File-5” adalah “GOL” atau data ditemukan
  Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah “File-6” = “File yang dicari” ?
  Jika “YA” maka “File-6” adalah “GOL” atau data ditemukan
  Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah “File-7” = “File yang dicari” ?
  Jika “YA” maka “File-7” adalah “GOL” atau data ditemukan
  Jika Tidak maka Lanjutkan ke **direktory** yang lainnya karena

b. Cek file yang akan dicari mulai dari Direktori Level -3



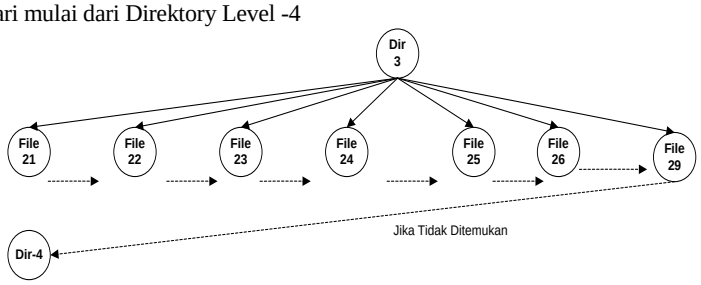
Gambar 6. Proses Pencarian Level 3

Pada tahap level-3 maka dilakukan pencarian pada direktory “DIR-2” dengan cara membandingkan apa yang dicari dengan file yang ada didalam direktory tersebut, dan prosesnya dilakukan seperti langkah berikut ini :

- Cek apakah “File-11” = “File yang dicari” ?
  - Jika “YA” maka “File-11” adalah “GOL” atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah “File-12” = “File yang dicari” ?

- Jika "YA" maka "File-12" adalah "GOL" atau data ditemukan
  - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-13" = "File yang dicari" ?
  - Jika "YA" maka "File-13" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-14" = "File yang dicari" ?
  - Jika "YA" maka "File-4" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-15" = "File yang dicari" ?
  - Jika "YA" maka "File-15" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-16" = "File yang dicari" ?
  - Jika "YA" maka "File-16" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-17" = "File yang dicari" ?
  - Jika "YA" maka "File-17" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke **direktory** "DIR-3"

c. Cek file yang akan dicari mulai dari Direktory Level -4

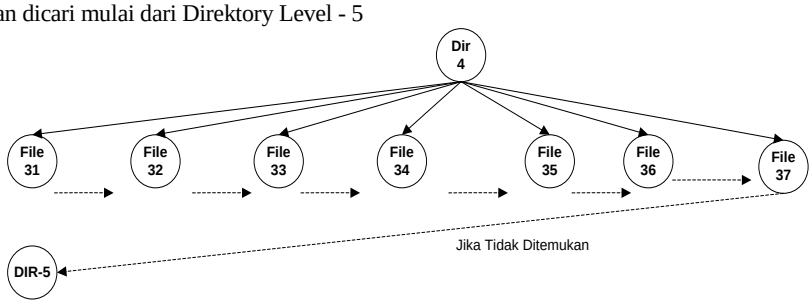


Gambar 7. Proses Pencarian Level 4

Pada tahap level-4 maka dilakukan pencarian pada direktory "DIR-3" dengan cara membandingkan apa yang dicari dengan file yang ada didalam direktory tersebut, dan prosesnya dilakukan seperti langkah berikut ini :

- Cek apakah "File-21" = "File yang dicari" ?
  - Jika "YA" maka "File-21" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-22" = "File yang dicari" ?
  - Jika "YA" maka "File-22" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-23" = "File yang dicari" ?
  - Jika "YA" maka "File-23" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-24" = "File yang dicari" ?
  - Jika "YA" maka "File-24" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-25" = "File yang dicari" ?
  - Jika "YA" maka "File-25" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-26" = "File yang dicari" ?
  - Jika "YA" maka "File-26" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-27" = "File yang dicari" ?
  - Jika "YA" maka "File-27" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke **direktory** "DIR-4"

d. Cek file yang akan dicari mulai dari Direktory Level - 5

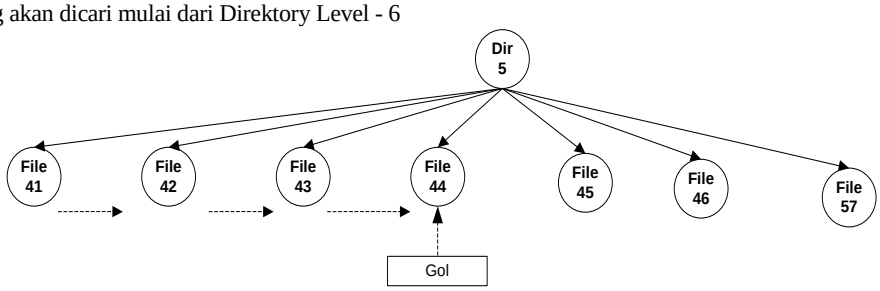


Gambar 8. Proses Pencarian Level 5

Pada tahap level-5 maka dilakukan pencarian pada direktory "DIR-4" dengan cara membandingkan apa yang dicari dengan file yang ada didalam direktory tersebut, dan prosesnya dilakukan seperti langkah berikut ini :

- Cek apakah "File-31" = "File yang dicari" ?
  - Jika "YA" maka "File-31" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-32" = "File yang dicari" ?
  - Jika "YA" maka "File-32" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-33" = "File yang dicari" ?
  - Jika "YA" maka "File-33" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-34" = "File yang dicari" ?
  - Jika "YA" maka "File-34" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-35" = "File yang dicari" ?
  - Jika "YA" maka "File-35" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-36" = "File yang dicari" ?
  - Jika "YA" maka "File-36" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-37" = "File yang dicari" ?
  - Jika "YA" maka "File-37" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke **direktory** "DIR-5"

e. Cek file yang akan dicari mulai dari Direktory Level - 6



Gambar 9. Proses Pencarian Level 6

Pada tahap level-6 maka dilakukan pencarian pada direktory "DIR-5" dengan cara membandingkan apa yang dicari dengan file yang ada didalam direktory tersebut, dan prosesnya dilakukan seperti langkah berikut ini :

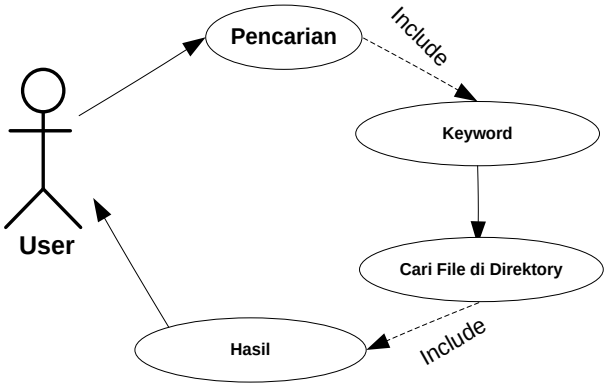
- Cek apakah "File-41" = "File yang dicari" ?
  - Jika "YA" maka "File-41" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-42" = "File yang dicari" ?
  - Jika "YA" maka "File-42" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-43" = "File yang dicari" ?
  - Jika "YA" maka "File-43" adalah "GOL" atau data ditemukan
    - Jika Tidak maka Lanjutkan ke File selanjutnya
- Cek apakah "File-44" = "File yang dicari" ?
  - Jika "YA" maka "File-34" adalah "GOL" ➔ GOL

Pada kedalaman 1, anak yang memenuhi kriteria akan dimasukan ke dalam queue. Lalu dari anak yang masuk queue tadi dibangkitkan anaknya. Dalam gambar di atas status 2 akan dimasukkan ke queue karena memenuhi criteria. Namun sebenarnya yang dibangkitkan anaknya bukan cuma status 2, namun juga status 4,6,8. Namun karena statusstatus itu merepresentasikan hal yang sama, maka tidak saya gambarkan, karena saya anggap hasilnya akan sama dengan anakanak yang dibangkitkan oleh status ke 2.Setiap kedalaman akan dilakukan operasi yang sama, sampai kedalaman maksimal atau saat CurrentNode adalah solusi.

Dengan melihat gambar di atas terlihat jelas keuntungan BFS dalam pencarian solusi ini, karena BFS mengunjungi anak secara melebar. Sehingga untuk kasus dimana pencarian yang lebih sedikit dimasukkan pohon status akan berhenti pada node tersebut tanpa harus melangkah lebih dalam.

a. Use Case Diagram

Menggambarkan sejumlah *external actors* dan hubungannya ke *use case* yang diberikan oleh sistem. *Use case* adalah depenelitian fungsi yang disediakan oleh system dalam bentuk teks sebagai dokumentasi dari *use case symbol* namun dapat juga dilakukan dalam *activity diagrams*. *Use case* digambarkan hanya yang dilihat dari luar oleh *actor* (keadaan lingkungan sistem yang dilihat user) dan bukan bagaimana fungsi yang ada di dalam sistem.



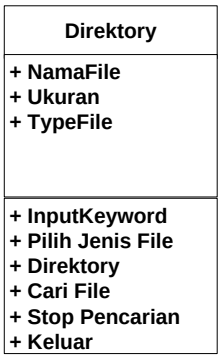
Gambar 10. UML Use Case Diagram Pencarian

Untuk memudahkan kita dalam menganalisa skenario yang akan kita gunakan pada fase-fase selanjutnya maka kita dapat melakukan pemilahan terhadap skenario tersebut, antara lain:
Nama use case : Pencarian informasi
Actor : user pengujung, system
Type : Primary
Tujuan : mencari informasi dengan metode *brute force*

ACTOR	SYSTEM
1. User membuka Program	
	3. Sistem menampilkan Tampilan Program di komputer user
4. User memasukkan Keyword yang akan dicari	
	5. Sistem Melakukan Pencarian Pada database dengan menggunakan metode *BFS* 6. Jika informasi=”ADA” maka tampilkan informasi yang dicari ke user 7. Jika informasi =”TIDAK ADA” maka tampilkan “informasi tidak ditemukan”
	8. Selesai

b. Class Diagram

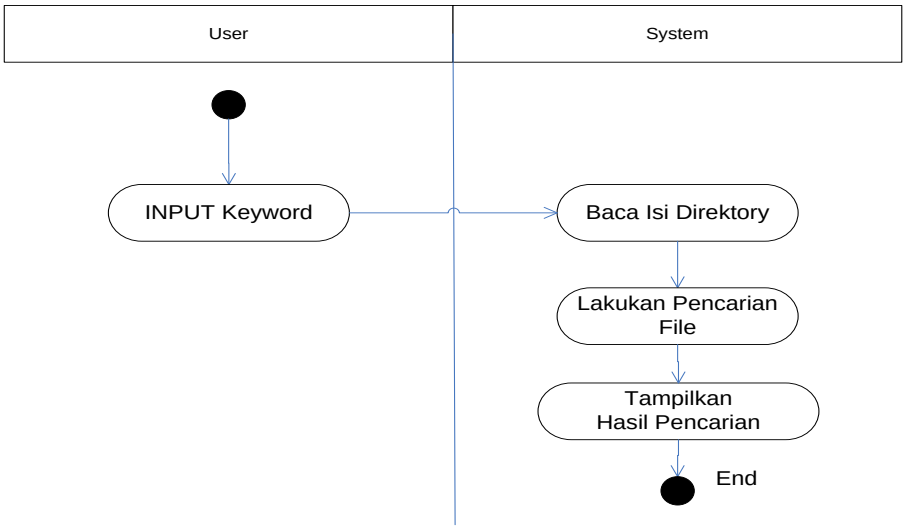
Diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem / perangkat lunak yang sedang kita kembangkan. Diagram kelas (Class Diagram) memberi kita gambaran (diagram statis) tentang sistem / perangkat lunak dan relasi-relasi yang ada di dalamnya. Bentuk Class Diagram dari sistem yang dibangun dapat dilihat pada gambar dibawah ini.



Gambar 11. UML Class Diagram Pencarian

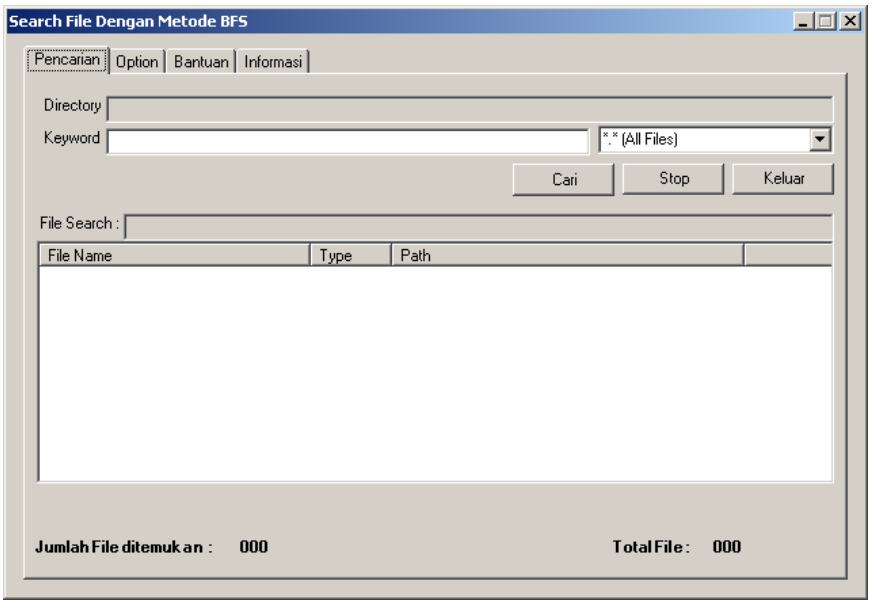
c. Activity Diagram

Activity diagram adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja. Dalam beberapa hal, diagram ini memainkan peran mirip sebuah diagram alir, tetapi perbedaan prinsip antara diagram ini dan notasi diagram alir adalah diagram ini mendukung behavior paralel.



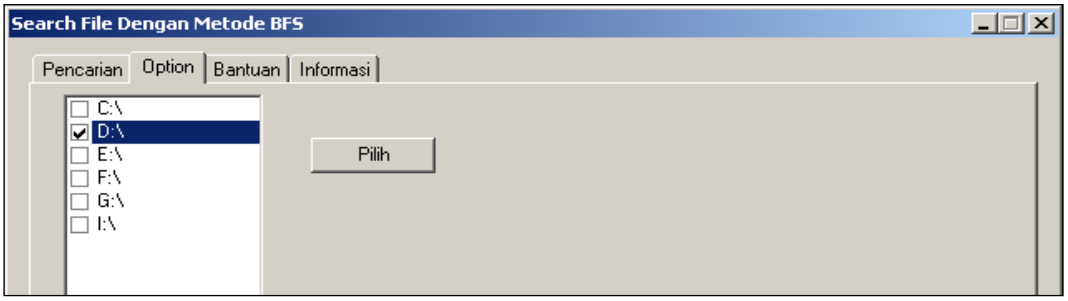
Gambar 12. UML *Activity* diagram

Untuk menjalankan perangkat lunak pencarian file ini cukup dengan menjalankan file BFS.exe, setelah dijalankan maka akan muncul tampilan flash sebagai berikut :



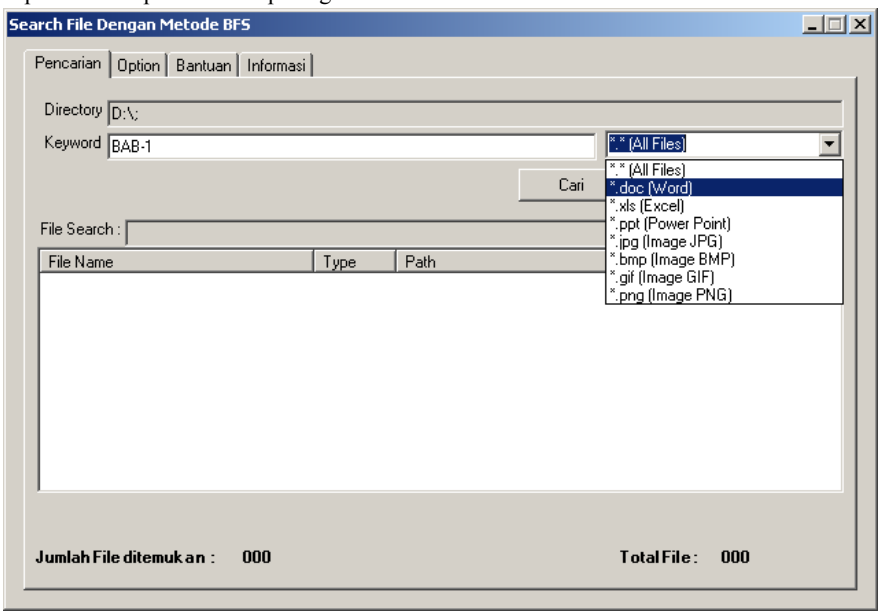
Gambar 13. Tampilan Utama

Proses ini digunakan untuk menampilkan bagaimana memilih sebuah *drive* yang didalamnya dilakukan proses pencarian, dan memilih tombol "Pilih" untuk mengaktifkan drive yang akan dipilih seperti terlihat pada gambar 14.



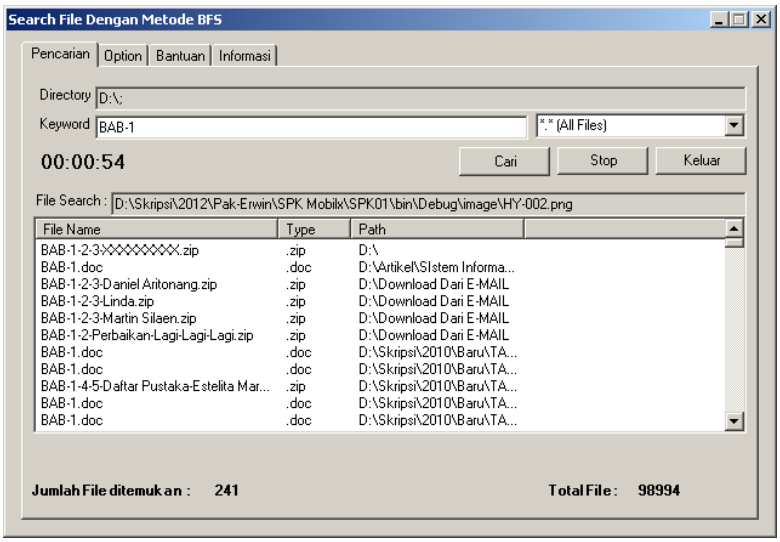
Gambar 14. Tampilan Pemilihan Drive

Proses ini digunakan untuk menampilkan bagaimana memasukkan sebuah keyword kedalam proses pencarian, dan memilih tombol "cari" untuk memulai pencarian seperti terlihat pada gambar 15.



Gambar 15.. Tampilan Input Keyword

Proses ini digunakan untuk memulai bagaimana proses pencarian, dan memilih tombol "cari" untuk memulai pencarian seperti terlihat pada gambar 16



Gambar 16. Tampilan Input Keyword

KESIMPULAN DAN SARAN

Sebagai penutup pembahasan dalam penulisan penelitian ini penulis mengambil kesimpulan-kesimpulan sekaligus memberikan saran kepada user yang menggunakan aplikasi ini. Dengan adanya kesimpulan dan saran ini dapatlah diambil suatu perbandingan yang akhirnya dapat memberikan perbaikan-perbaikan pada masa yang akan datang.

Adapun kesimpulan yang penulis peroleh adalah sebagai berikut:

1. Jenis File yang dapat diload oleh program adalah semua jenis file
2. Algoritma *BFS* sangat baik digunakan dalam pencarian kata dalam sebuah direktory
3. Program dapat mencari file yang nama filenya mengandung huruf sesuai dengan keyword yang dimasukkan
4. Perangkat lunak merupakan implementasi nyata dalam penerapan pohon pelacakan dalam mencari solusi.
5. Dalam proses pembuatan sistem yang baru dapat diketahui bahwa untuk menyusun suatu program aplikasi yang baik, tahap-tahap yang perlu dilakukan adalah dengan mempelajari sistem yang ada, kemudian mendesain suatu sistem yang dapat mengatasi masalah serta mengimplementasikan sistem yang didesain.
6. Dengan menerapkan sistem BFS pada sebuah dokumen maka proses pencarian dan pencocokan kata akan semakin cepat dan akurat

DAFTAR PUSTAKA

[1] "Algoritma Zone: Pengertian Algoritma Pencarian (searching algorithm)." http://algoritmazone.blogspot.com/2012/05/pengertian-algoritma-pencarian.html (accessed Apr. 19, 2021).

[2] Santi, "Penerapan Algoritma Best First Search (BFS) dalam Pencarian Lokasi Apotek K-24 Berbasis Android di Kota Makassar," *SNIf*, pp. 288–291, 2015.

[3] M. Arhami, "Konsep dasar sistem pakar," *Yogyakarta Andi*, vol. 206, 2005.

[4] S. Kusrini, "Sistem Pakar Teori dan Aplikasi," *Penerbit Andi Yogyakarta*, 2006.

[5] P. S. Ramadhan, "Sistem Pakar Pendiagnosaan Dermatitis Imun Menggunakan Teorema Bayes," *InfoTekJar (Jurnal Nas. Inform. dan Teknol. Jaringan)*, 2018, doi: 10.30743/infotekjar.v3i1.643.

[6] N. E. Putri, "SISTEM PAKAR KERUSAKAN HARDWARE KOMPUTER DENGAN METODE FORWARD CHAINING (Studi Kasus: Benhur Sungai Penuh) | Putri | Jurnal Momentum," 2016. https://ejournal.itp.ac.id/index.php/momentum/article/view/374 (accessed Dec. 03, 2019).

[7] R. Ambarita, "Sistem Pakar Diagnosa Kerusakan Mainboard Komputer," *IJIS - Indones. J. Inf. Syst.*, vol. 2, no. 1, 2017, doi: 10.36549/ijis.v2i1.20.

[8] B. Prasetiyo and M. R. Hidayah, "Penggunaan Metode Depth First Search (DFS) dan Breadth First Search (BFS) pada Strategi Game Kamen Rider Decade Versi 0.3," *Sci. J. Informatics*, vol. 1, no. 2, pp. 161–167, 2015, doi: 10.15294/sji.v1i2.4022.