Kegunaan BFS dan DFS dalam Mendiagnosa Pasien

Dicky Adrian - 13516050[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13516050@std.stei.itb.ac.id

***Abstrak*—Banyaknya kebutuhan mengenai kesehatan masyarakat menimbulkan kebutuhan pengambilan keputusan yang cepat dan tepat dari pihak yang bersangkutan. Dengan banyaknya data yang sudah dikumpulkan oleh instansi-instansi, pengambilan keputusan bisa dilakukan dengan lebih mangkus dan sangkil. Salah satu contohnya adalah penentuan obat-obatan dengan menggunakan struktur data pohon, yaitu dengan melihat gejala-gejala yang ada pada tubuh seseorang dan membuat pohon baik secara *Breadth Firsth Search* (BFS) ataupun dengan *Depth First Search* (DFS). Dengan menggunakan algoritma yang sesuai, kita dapat membuat pengambilan keputusan mengenai obat-obatan lebih tepat berdasarkan kasus-kasus yang sudah terjadi, tentu saja jika didasari dengan data-data yang sudah pasti benar juga. Hal ini juga bisa mengurangi kemungkinan dari pihak-pihak yang kurang bertanggung jawab terhadap pekerjaannya dan melakukan malapraktik dalam menjalankan pekerjaannya sebagai dokter atau juga memberikan obat-obatan yang tidak diperlukan hanya untuk menambah keuntungan yang didapat oleh satu pihak.**

Kata Kunci— Pohon, BFS, DFS, obat.

I. Introduction

Perkembangan data yang cukup cepat dalam berbagai bidang di dunia ini menarik perhatian bagi pengembang untuk mengembangkan kakas yang bisa membantu. Dengan semakin besarnya data, semakin banyak keputusan yang bisa diambil dalam kejadian-kejadian yang terjadi. Karena itu salah satu cara yang bisa digunakan adalah dengan membuat sebuah struktur data pohon untuk kemudian mengambil keputusan yang tepat dan juga cepat.

Contoh pembuatan pohon sederhana dari berbagai kondisi yang ada adalah dengan menyimpan sebuah data bagaimana merencanakan suatu liburan dari data-data yang ada. Misalkan data yang kita punya negara mana saja yang memiliki tingkat hujan tinggi dan rendah. Selain itu juga kita memiliki data mengenai negara mana saja yang memiliki pantai dan negara mana yang tidak memiliki pantai. Jika seseorang merencanakan liburan di suatu tempat yang cukup hangat dan memiliki pantai, kita bisa membuatkan suatu pilihan dari negara-negara yang memungkinkan untuk dijadikan tempat liburan orang itu.

Contoh sederhana di atas bisa dijadikan sebuah kakas yang lebih bermanfaat lagi, contohnya adalah dengan menggunakannya di bidang kesehatan. Kesehatan adalah salah satu kebutuhan yang paling utama manusia. Untuk menjaga kesehatan, tubuh memerlukan gizi-gizi dan vitamin-vitamin yang cukup.

Tetapi dengan berkembangnya penyakit yang ada di dunia, kita tidak bisa selalu menghindari penyakit yang ada. Oleh karena itu terkadang satu-satunya cara adalah mengunjungi pihak yang bisa membantu dalam bidang kesehatan, yaitu dokter. Tetapi, masalah yang terjadi adalah ketika dokter memberikan diagnosa penyakit yang salah kepada pasien yang bersangkiutan, atau ternyata dokter tersebut melakukan malapraktik dan menyebabkan penyakit pasien bertambah parah.

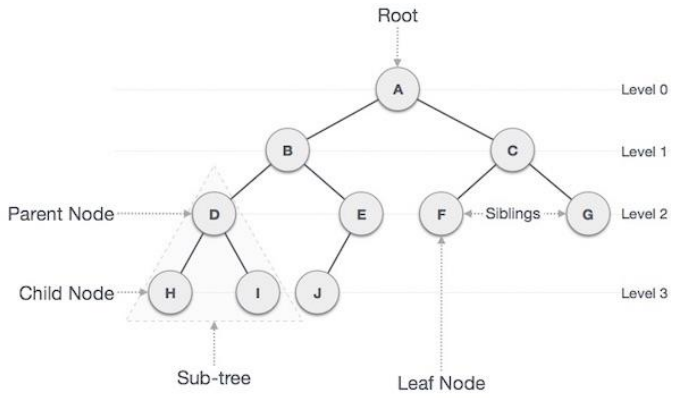
Oleh karena itu, dengan menggunakan data-data yang sudah dimiliki, kita bisa menggunakan struktur data pohon untuk membuat diagnosa yang lebih tepat agar tidak terjadi kesalahan yang terlalu fatal.

II. Pohon

A. Definisi

Sebuah pohon dalam konteks struktur data adalah salah satu cara untuk merepresentasikan sebuah data dalam bentuk hierarki. Pohon banyak digunakan dalam struktur data karena memiliki keunikan-keunikan tertentu dalam penyimpanan sebuah data. Misalnya, bentuknya yang hierarki akan memudahkan seseorang untuk memahami permasalahan tertentu. Selain itu juga pohon memiliki kegunaan-kegunaan lainnya, misalnya bisa menjadi sebuah pohon pencarian, atau bisa juga menjadi pohon untuk membantu memutuskan sesuatu. Sebuah pohon bisa dibentuk dengan menggunakan pendekatan rekrusif maupun dengan pendekatan traversal. Sebuah pohon juga bisa didefinisikan sebagai sebuah graf tidak berarah yang tidak memiliki sirkuit di dalamnya. Dalam sebuah pohon terdapat beberapa istilah seperti akar, simpul, daun, dan lain-lain. Istilah-istilah tersebut akan dijelaskan dalam upa-bab selanjutnya.

B. Terminologi



sumber: https://www.tutorialspoint.com/data_structures_algorithms/tree_data_structure.htm

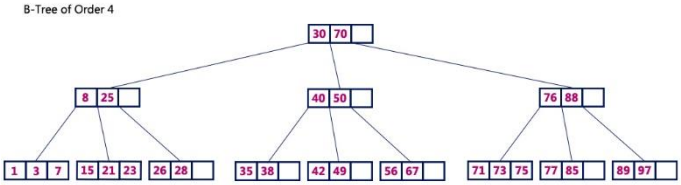
Dari gambar di atas, dapat diuraikan beberapa istilah dalam pohon sebagai berikut:

1. Simpul
   Simpul adalah data yang direpresentasikan dalam sebuah pohon. Pada gambar di atas, terdapat beberapa Simpul yaitu: A, B, C, D, E, F, G, H, I, dan J
2. Akar (*Root*)
   Sebuah akar adalah simpul paling atas dari sebuah pohon. Pada gambar di atas, *simpul* yang menjadi akar dari pohon tersebut adalah *simpul* A.
3. Anak dan Orang Tua (*Child and Parent*)
   Dalam sebuah pohon hubungan yang merepresentasikan anak dan orang tua adalah hubungan antara satu *simpul* dengan simpul lainnya. Contoh pada gambar di atas adalah simpul D adalah anak dari simpul B, yang berarti simpul B adalah orang tua dari simpul D.
4. *Descendant* dan *Ancestor*
   Hubungan *descendant* dan *ancestor* mirip dengan hubungan anak dan orang tua. Definisi formal dari istilah ini adalah jika ada suatu jalan dari satu simpul ke simpul yang lain melalui simpul yang lain lagi, maka bisa dibilang salah satu simpul tersebut adalah *descendant* dan yang lainnya adalah *ancestor*.
   Contohnya pada gambar di atas adalah simpul H adalah *descendant* dari simpul *B* yang berarti simpul B adalah *ancestor* dari simpul H.
5. Saudara
   Istilah saudara dalam struktur data pohon adalah jika suatu simpul mempunyai lebih dari satu anak, maka kedua simpul tersebut adalah saudara.
   Contoh dari gambar di atas adalah simpul B mempunyai dua anak yaitu D dan E, maka D dan E adalah saudara.
6. Daun
   Dalam struktur data pohon, daun diartikan sebagai simpul yang tidak memiliki anak lagi, atau sudah berada pada akhir dari struktur data.
   Contoh simpul daun dari gambar di atas adalah H, I, J, F, G.
7. *Internal Node*
   Kebalikan dari istilah daun, *internal node* adalah simpul yang setidaknya masih memiliki satu anak dalam sebuah pohon.
   Contoh pada gambar di atas adalah A, B, C, D, dan E.
8. Level
   Istilah level digunakan untuk merepresentasikan generasi ke-berapa sebuah simpul. Dari gambar di atas bisa dilihat bahwa simpul D memiliki level 2, simpul H memiliki level 3, dan seterusnya.
9. Tinggi
   Ketinggian adalah level maksimum dari sebuah pohon. Pada gambar di atas, tinggi dari pohon adalah 3.
10. Upa-Pohon (*Sub-Tree*)
    Upa-Pohon adalah sebuah pohon yang masih terdapat dalam sebuah pohon. Pada gambar di atas dapat dilihat bagian yang diarsir adalah upa-pohon yang dimaksud.
11. Hutan
    Kumpulan dari pohon-pohon

C. Jenis Pohon

Terdapat beberapa jenis pohon, yang dibagi-bagi berdasarkan keunikan nilai setiap simpulnya, atau peletakan simpulnya. Beberapa jenis pohon tersebut adalah sebagai berikut:

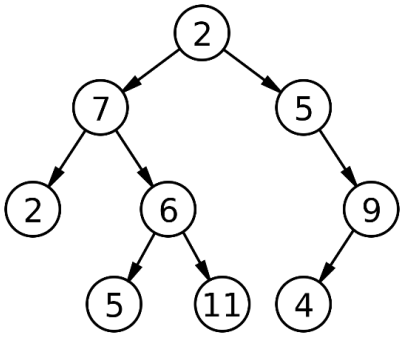
1. *B-Tree*
   Istilah *B-Tree* Bisa disamakan dengan sebuah pohon pencarian yang juga seimbang. Setiap simpul memiliki antara *m* sampai *m/2* buah anak. Dimana *m* adalah sebuah bilangan bulat positif yang memiliki nilai lebih dari satu. Angka yang dipilih sebagai *m* biasa disebut juga *order* dari pohon tersebut.



Sumber:http://btechsmartclass.com/DS/images/B-Tree%20Example.jpg

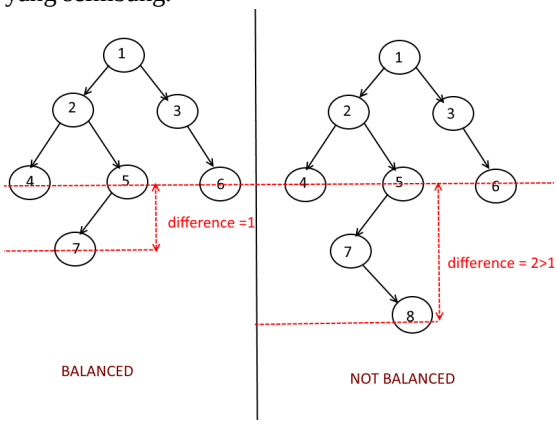
Dapat dilihat pada gambar bahwa setiap simpul dalam pohon tersebut memiliki 3 anak. Hal ini memenuhi syarat karena 3 berada di antara 2 dan 4.
Pohon ini biasa digunakan jika kecepatan memory yang dipakai cukup lamban. Sehingga dengan memperbesar *m*, tinggi dari pohon akan semakin rendah dan mempercepat akses data dari struktur data yang sudah dibaut.

2. Pohon Biner
   Pohon biner merupakan jenis pohon yang cukup banyak digunakan dalam struktur data. Pohon ini memiliki ciri-ciri hanya memiliki maksimal dua buah simpul anak pada setiap simpulnya.



Sumber:
https://upload.wikimedia.org/wikipedia/commons/thumb/f/f7/Binary_tree.svg/1200px-Binary_tree.svg.png
Dapat dilihat pada gambar di atas setiap simpul dari pohon tersebut memiliki simpul anak <= 2. Sehingga pohon tersebut memenuhi syarat sebagai pohon biner.

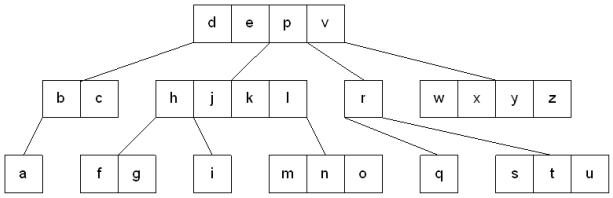
3. Pohon Seimbang (*Balanced Tree*)
Pohon dikatakan seimbang ketika di dalam satu pohon beda ketinggian antara upa-pohon kanan dan upa-pohon kiri maksimbal adalah satu. Selain itu, setiap upa-pohon juga harus merupakan pohon yang seimbang.
Seperti yang sudah dijelaskan sebelumnya, B-Tree juga merupakan salah satu yang termasuk pohon seimbang. Selain B-tree, beberapa jenis lainnya yang termasuk pohon seimbang adalah BB(*a*) *Tree*, AVL *Tree*, dan *red-black tree*. Berikut ini adalah contoh dari sebuah pohon yang seimbang.



Sumber:
https://i1.wp.com/algorithms.tutorialhorizon.com/files/2014/09/BalancedTree-Example.png

Seperti yang bisa dilihat pada gambar di atas, gambar kiri merupakan pohon yang seimbang karena perbedaan ketinggiannya hanya 1, sedangkan pada gambar yang kanan terdapat perbedaan ketinggian sejumlah 2, yang berarti pohon tersebut tidak termasuk dalam pohon yang seimbang.

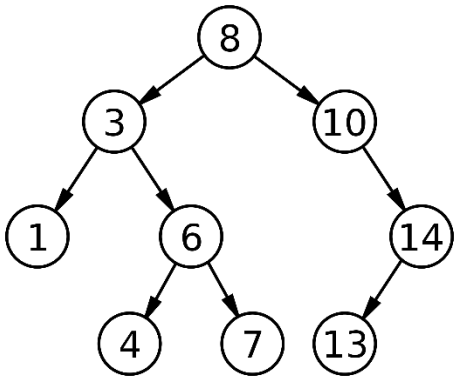
4. *Multiway Tree*
Sebuah pohon dikatakan sebagai *multiway tree* jika setiap simpulnya tidak diharuskan memiliki satu nilai. Biasanya setiap simpulnya dinyatakan dalam bentuk sebuah *list* yang menyatakan nilai-nilai yang termasuk dalam simpul tersebut. Berikut ini adalah contoh pohon yang termasuk *multiway tree*



Sumber:
http://faculty.cs.niu.edu/~freedman/340/340notes/gifImages/340multi1.gif

Pada Gambar di atas bisa dilihat bahwa setiap simpul tidak lagi dinytakan dengan satu huruf atau satu angka, tetapi dengan menggunakan kumpulan huruf-huruf dalam satu simpul.

5. Pohon Pencarian
Sebuah pohon bisa digunakan untuk melakukan pencarian tertentu. Cara untuk melakukannya adalah dengan menggunakan pohon pencarian. Pohon pencarian dibuat dengan menyusun setiap simpulnya mengikuti suatu aturan tertentu. Aturan yang biasanya dipakai adalah semua simpul yang ada di kiri akar memiliki nilai leibh kecil dari akarnya, dan semua simpul yang berada di kanan akar memiliki nilai yang lebih besar daripada akarnya. Salah satu jenis pohon pencarian yang sering dipakai adalah pohon pencarian biner yang pada setiap simpulnya memiliki dua simpul, simpul yang kiri akan bernilai lebih kecil dari pada nilai akarnya, dan simpul kanan akan bernilai lebih besar dari akarnya. Berikut adalah contoh pohon pencarian.

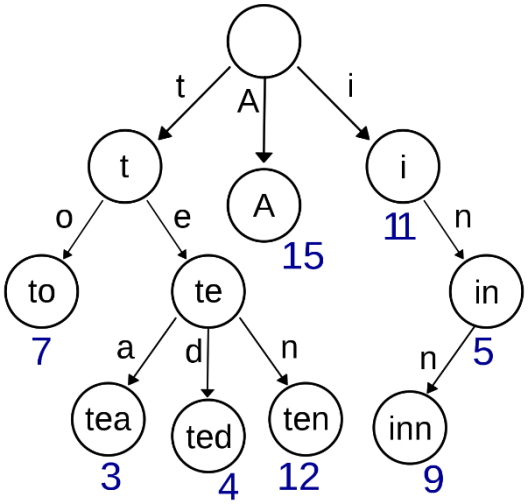


Sumber:
https://upload.wikimedia.org/wikipedia/commons/thumb/d/da/Binary_search_tree.svg/2000px-Binary_search_tree.svg.png

Pada gambar yang diberikan di atas, pohon pencarian tersebut termasuk pohon pencarian biner. Bisa dilihat bahwa setiap nilai pada upa-pohon kiri dari simpul 8 memiliki nilai yang lebih kecil dari 8, sedangkan upa-pohon kanan dari simpul 8 memiliki nilai yang lebih besar dari 8.

6. *Digital Tree*

Jenis pohon yang ini memiliki keunikan karena setiap simpulnya menyimpan nilai sebuah *string*. Nilai yang disimpan tersebut dibentuk dari upa-*string* yang didefinisikan sebelumnya. Berikut adalah contoh dari pohon jenis ini.

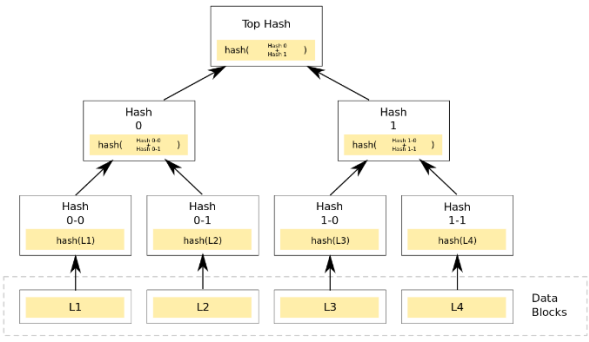


Sumber: https://upload.wikimedia.org/wikipedia/commons/thumb/b/be/Trie_example.svg/1200px-Trie_example.svg.png

Pada contoh di atas bisa dilihat bahwa setiap simpul memiliki nilai berupa *string* tertentu yang dibentuk dari upa-*string* simpul sebelumnya. Beberapa jenis dari *digital tree* adalah pohon pencarian *digital*, *trie*, *Patricia tree*, dan pohon sufiks.

7. *Merkle Tree*

Pohon dengan jenis *merkle tree* memiliki keunikan yaitu setiap simpul internalnya memiliki fungsi hash yang akan memberikan informasi semua daun yang ada di bawah simpul internal tersebut. Selain itu, pada setiap daun yang ada di pohon tersebut akan menyimpan fungsi hash yang menunjukan data yang ditunjuk oleh daun tersebut. Setiap daun memiliki ketinggian yang sama dan setiap simpul didahulukan ke kiri.



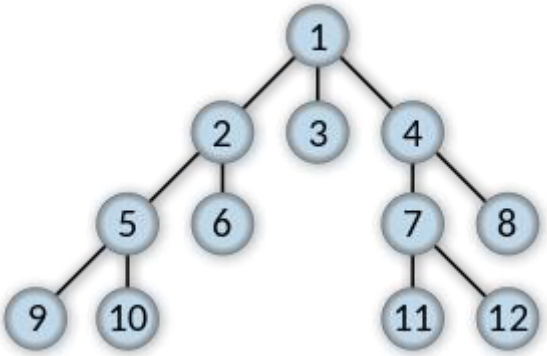
Sumber: https://upload.wikimedia.org/wikipedia/commons/thumb/9/95/Hash_Tree.svg/1200px-Hash_Tree.svg.png

III. Algoritma Pembentukan

Ada Beberapa algoritma untuk membentuk pohon, tetapi pada bab ini hanya akan dibahas dua algoritma yaitu BFS (*Breadth First Search*) dan DFS (*Depth First Search*).

A. BFS

BFS atau *Breadth First Search* adalah salah satu algoritma pencarian dalam pohon yang akan membangkitkan semua anak dari suatu simpul terlebih dahulu. Contohnya adalah sebagai berikut.



Sumber: https://upload.wikimedia.org/wikipedia/commons/thumb/3/33/Breadth-first-tree.svg/300px-Breadth-first-tree.svg.png

Pada gambar di atas bisa dilihat urutan pembangkitan anak yang dilakukan oleh pohon tersebut. Pertama-tama simpul 1 sebagai akar dari pohon tersebut akan membangkitkan tiga anaknya yaitu simpul 2, 3, dan 4. Setelah itu setiap anak-anaknya akan membangkitkan simpul-simpul anaknya, misalnya simpul 2 akan membangkitkan simpul 5 dan simpul 6, dan seterusnya.

Metode BFS bisa diterapkan dalam pencarian di pohon ruang status. Berikut ini adalah langkah-langkah pencarian dalam pohon ruang status dengan menggunakan BFS:

1. Buat sebuah antrian baru misalkan *Q*.
2. Masukkan sebuah simpul ke dalam antrian *Q*.
3. Ambil sebuah elemen dari *Q*, misalkan *v*, jika *v* adalah solusi, maka solusi sudah ditemukan, proses pencarian dihentikan.
4. Jika *v* bukan solusi, maka bangkitkan semua anak dari *v* jika ada, dan masukkan semua anak ke belakang antrian *Q*.
5. Kembali ke langkah 3.

Dengan menggunakan algoritma di atas, metode BFS akan selalu mendapatkan sebuah solusi jika terdapat solusi. Solusi yang didapatkan dengan metode BFS juga selalu merupakan solusi yang optimal karena akan dicek dari level terendah pohon tersebut.

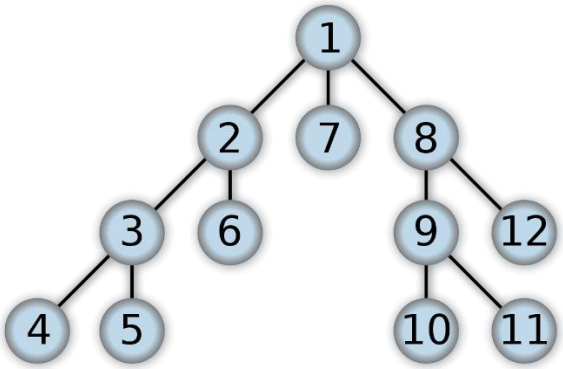
Tetapi, BFS memiliki kelemahan juga dalam pencarian pohon ruang status. Kelemahan dari metode ini yaitu memiliki kompleksitas yang cukup tinggi. Misalkan suatu simpul membangkitkan *b* buah simpul anak, dan setiap simpul anaknya akan membangkitkan *b* buah simpul anak lainnya, maka kompleksitas yang didapat dari algoritma ini bersifat eksponensial, karena jumlah seluruh simpul yang dibangkitkan adalah:

$$1 + b + b^2 + b^3 + \cdots + b^d = \frac{b^{d+1} - 1}{b - 1}$$

Sehingga kompleksitas waktu yang didapatkan dari algoritma BFS adalah O(b^d).

B. DFS

Metode DFS adalah metode yang membangkitkan anak sampai ke kedalaman yang paling dalam terlebih dahulu dalam melakukan pencarian. Metode ini digunakan untuk memperbaiki kelemahan yang dimiliki metode BFS. Hal ini karena dengan menggunakan DFS, tidak semua anak harus dibangkitkan terlebih dahulu sebelum mencapai solusi. Berikut ini adalah urutan pembangkitan simpul dengan menggunakan metode DFS.



Sumber: https://upload.wikimedia.org/wikipedia/commons/thumb/1/1f/Depth-first-tree.svg/1200px-Depth-first-tree.svg.png

Bisa dilihat pada gambar di atas setiap angka merepresentasikan urutan pembangkitan simpul. Seperti yang dijelaskan sebelumnya dengan menggunakan metode DFS, anak yang dibangkitkan akan mencapai ke kedalaman paling dalam terlebih dahulu. Sehingga simpul 1 yang merupakan simpul akar akan membangkitkan simpul 2, simpul 2 akan membangkitkan simpul 3, simpul 3 akan membangkitkan simpul 4, dank arena simpul 4 merupakan daun, maka akan dilakukan runut balik ke simpul sebelumnya yaitu simpul 3. Simpul 3 kemudian akan membangkitkan simpul anak lainnya yaitu simpul 5.

Sama seperti menggunakan BFS, metode DFS juga bisa digunakan dalam pencarian pohon ruang status. Untuk metode DFS bisa dilakukan dengan algoritma sebagai berikut:

1. Buat sebuah antrian misalkan *Q*.
2. Masukan sebuah simpul akar ke dalam antrian tersebut.
3. Jika simpul tersebut merupakan solusi, maka solusi ditemukan.
4. Bangkitkan semua anak dari simpul akar, tempatkan simpul anak di depan antrian
5. Ambil satu simpul dari kepala antrian
6. Kembali ke langkah 3.
7. Jika antrian sudah kosong dan belum ditemukan solusi, maka tidak ada solusi.

Dengan menggunakan metode DFS ini, kompleksitas waktu yang didapatkan biasanya akan lebih baik daripada dengan menggunakan metode BFS. Pada metode ini kemungkinan yang terburuk didapat adalah bahwa semua simpul akan dibangkitkan yang berarti kompleksitas waktunya sama dengan metode BFS, yaitu O(b^m). Tetapi, kompleksitas ruang yang dimiliki metode ini lebih baik jika dibandingkan dengan BFS, yaitu O(bm), karena kita tidak perlu menyimpan semua nilai dari semua simpul, melainkan hanya menyimpan satu buah lintasan tunggal dari akar sampai daun ditambah dengan simpul-simpul saudara yang belum dibangkitkan.

Metode DFS biasanya digunakan ketika suatu pohon memiliki banyak solusi. Metode DFS akan menghasilkan solusi lebih cepat jika dibandingkan dengan metode BFS. Tetapi solusi yang dihasilkan metode DFS belum tentu merupakan solusi yang paling optimal. Hal ini karena jika ditemukan solusi di kedalaman terdalam suatu pohon, dan pengecekan dihentikan, belum bisa menjamin bahwa tidak ada solusi di kedalaman yang lebih rendah pada pohon tersebut.

IV. DATA DAN KALKULASI

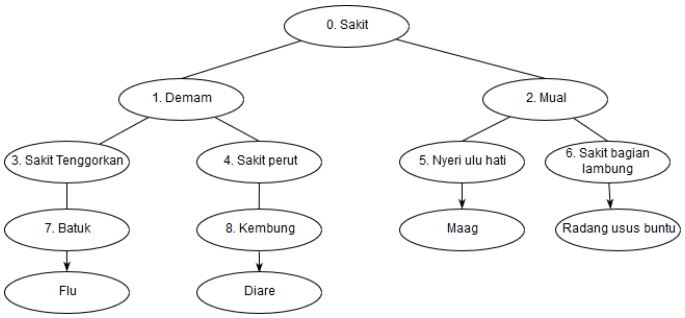
Misalkan data dari penyakit dan gejala-gejala yang kita miliki adalah sebagai berikut ini:

1. Flu
   Gejala: Demam, sakit kepala, batuk
2. Diare
   Gejala: Demam, sakit perut, kembung
3. Maag
   Gejala: Mual, nyeri ulu hati
4. Radang usus buntu
   Gejala: Mual, sakit pada bagian lambung

Maka, dengan menggunakan kedua metode yang sudah dijelaskan di atas, kita bisa memberikan dua jenis solusi yang akan memudahkan bagi pasien untuk mengetahui diagnosa sementara dan juga bagi dokter untuk membantu dalam diagnosa penyakit yang diderita pasien.

A. Metode BFS

Dengan menggunakan metode BFS, pohon yang dihasilkan dari data yang kita miliki adalah sebagai berikut:



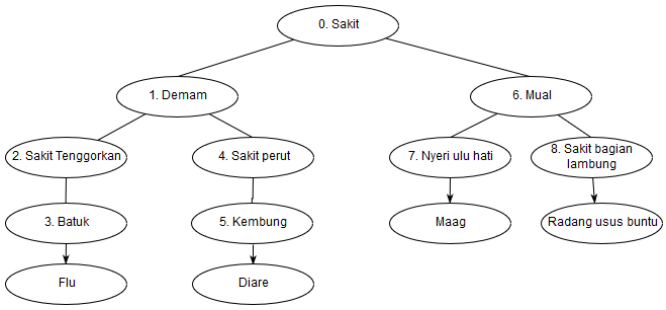
Sumber: Dokumen pribadi

Pohon tersebut terbentuk dengan urutan sesuai nomor yang tertera di gambar. Misalkan pengguna memasukkan gejala dengan urutan demam, sakit perut, dan kembung, maka akan dilakukan pengecekan ruang status sampai didapatkan status yang sama dengan gejala yang didapat. Setelah mencapai daun, maka daun tersebut akan dihubungkan ke sebuah sistem data yang memberitahu pengguna menderita gejala-gejala penyakit diare.

Setelah mendapatkan diagnosa sementara dari pengecekan pohon ruang status, maka pengguna bisa berkonsultasi lebih lanjut lagi dengan dokter yang berwenang memberikan obat-obatan tertentu.

A. Metode DFS

Dengan menggunakan metode DFS, pohon yang dihasilkan dari data yang kita miliki adalah sebagai berikut:



Sumber: Dokumen pribadi

Dengan menggunakan metode ini, tentu saja belum tentu semua simpul terbentuk dengan sempurna, misalkan jika pengguna mencari status demam, sakit tenggorokan dan batuk, maka pengguna akan langsung menemukannya di pencarian mendalam pertama dan simpul yang terbentuk hanya mencapai simpul nomor 3. Sehingga hasil yang diperoleh bisa menjadi lebih cepat dibandingkan dengan menggunakan metode BFS, dan kompleksitas ruang yang dihasilkan seperti yang sudah dijelaskan sebelumnya juga lebih baik daripada dengan menggunakan BFS.

V. KESIMPULAN

Berdasarkan hasil dari data dan kalkulasi yang sudah dijelaskan pada bab sebelumnya, penggunaan struktur data pohon dan pencarian ruang status dengan kedua metode yaitu BFS dan DFS bisa berguna di banyak bidang kehidupan. Setelah menggunakan kedua metode tersebut diperoleh bahwa kita bisa mempercepat pencarian diagnosa terhadap gejala-gejala yang ada jika kita memiliki cukup data mengenai penyakit dan gejala-gejalanya.

Berdasarkan teori dan penelitian yang ada, akan lebih baik jika digunakan metode DFS, karena pada kasus ini, pengguna membutuhkan kecepatan dalam pencarian status. Seperti yang sudah dijelaskan sebelumnya bahwa dengan algoritma DFS, kita bisa mendapatkan hasil pencarian ruang status lebih cepat daripada dengan menggunakan metode BFS. Selain itu juga pada kasus ini, sangat kecil kemungkinan ada dua solusi yang berbeda, jika pohon yang dibentuk sudah benar dan sesuai dengan data-data yang benar juga.

Selanjutnya dengan dibuatnya struktur hierarki seperti yang telah dijelaskan di atas, harapannya adalah untuk membuat sebuah aplikasi interaktif. Misalnya dengan memberikan pilihan pada setiap tahapan keputusan. Sehingga pengguna bisa lebih mudah menggunakannya.

VI. SARAN

Untuk hasil yang lebih baik tentu saja diperlukan data yang lebih banyak lagi daripada hanya mendata empat buah penyakit. Selain itu juga dibandingkan dengan menggunakan algoritma yang ada pada bab III, lebih baik digunakan algoritma runut balik pada pencarian ruang status dengan menggunakan metode DFS. Dengan menggunakan algoritma runut balik, ruang yang dibutuhkan akan menjadi lebih kecil yang berarti kompleksitas ruang yang didapat bisa menjadi lebih baik lagi. Tentu saja hasil diagnosis yang didapat dari pencarian ruang status ini tidak bisa benar-benar akurat dan 100% karena banyak variabilitas di bidang kesehatan itu sendiri. Sehingga pengguna tetap harus berkonsultasi dengan dokter untuk pengobatan yang sesuai dengan penyakitnya. Pencarian ruang status ini hanyalah digunakan untuk meminimalisir terjadinya malapraktik dan juga kesalahan diagnosa.

DAFTAR PUSTAKA

[1] Dr. Rinaldi Munir “Diktat Kuliah Strategi Algoritma” Tahun 2018
[2] https://xlinux.nist.gov/dads/HTML/tree.html diakses pada 12 Mei 2018

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Mei 2018

Dicky Adrian – 13516050