Aplikasi Pohon Keputusan dalam Pemilihan Struktur Data pada Pembuatan Program

Mohamad Daffa Argakoesoemah 13520118[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13520118@stei.itb.ac.id

***Abstract*—Struktur data merupakan salah satu konsep yang paling dasar pada pemrograman. Pemilihan struktur data saat pembuatan program menjadi sebuah persoalan yang penting karena struktur data yang cocok untuk sebuah program akan membuat program tersebut lebih efisien. Namun, ada banyak struktur data yang dapat dipakai sehingga terkadang membuat seseorang bingung saat membuat program. Hal ini bisa diselesaikan menggunakan bantuan ilmu dalam matematika diskrit, yaitu pohon. Salah satu aplikasi pohon adalah pohon keputusan. Pohon keputusan dapat menjadi salah satu alat yang dapat membantu memilih struktur data yang cocok sesuai kebutuhan.**

***Kata kunci*—pohon keputusan, struktur data, karakteristik, program**

I. Pendahuluan

Salah satu kemampuan penting yang harus dimiliki seorang *programmer*, yaitu dapat menentukan struktur data yang cocok untuk program yang dibuatnya. Struktur data yang cocok akan memberikan banyak manfaat bagi sebuah program. Salah satu pengaruh struktur data, yaitu akan memberikan dampak yang cukup signifikan pada kecepatan sebuah kode program saat dijalankan.

Struktur data	Add	Find	Delete	Get-by-index
Array	O(n)	O(n)	O(n)	O(1)
Linked list	O(1)	O(n)	O(n)	O(n)
Stack	O(1)	-	O(1)	-
Queue	O(1)	-	O(1)	-
Hash table	O(1)	O(1)	O(1)	-
Balanced Search Tree	O(log n)	O(log n)	O(log n)	O(log n)

Gambar 1.1 Tabel perbandingan kompleksitas operasi dalam berbagai struktur data

Sumber: Salindia Kuliah IF2110 – Algoritma & Struktur Data

Pada Gambar 1.1 dapat dilihat bahwa masing-masing struktur data memiliki kompleksitas waktu yang berbeda-beda untuk masing-masing operasi dasarnya, seperti operasi penambahan, pencarian, penghapusan, dan pengambilan elemen.

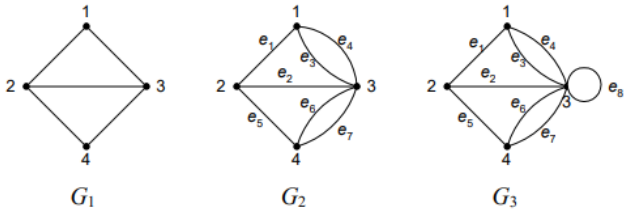
Ada banyak jenis struktur data yang dapat digunakan pada sebuah program, seperti pada Gambar 1.1. Struktur data yang digunakan bergantung kepada kebutuhan dan tujuan sebuah program. Hal ini disebabkan masing-masing struktur data memiliki karakteristik yang berbeda-beda. Misalnya, struktur data yang digunakan pada program antrian pesanan makanan tentu akan berbeda dengan program yang menyimpan daftar nilai ujian mahasiswa.

Berdasarkan pemaparan di atas, persoalan pemilihan struktur data pada saat pembuatan program dapat dipermudah dengan bantuan pohon keputusan. Dengan pohon keputusan, *programmer* dapat memilih struktur data yang cocok sesuai kebutuhannya.

II. Landasan Teori

A. Graf

Sebuah graf G didefinisikan sebagai sepasang nilai (V, E). V adalah himpunan tidak kosong dari simpul-simpul atau *vertices* dan E adalah himpunan sisi atau atau *edges* yang menghubungkan sepasang simpul.

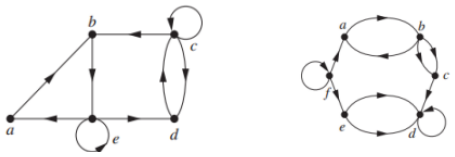


Gambar 2.1 G1 graf sederhana, sedangkan G2 dan G3 adalah graf tak sederhana

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, graf dibedakan menjadi graf sederhana dan graf tak sederhana. Graf sederhana adalah graf yang tidak mempunyai gelang maupun sisi ganda. Gelang adalah sisi yang menghubungkan simpul yang sama. Graf tak sederhana adalah graf yang mengandung sisi ganda atau gelang.

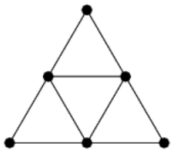
Berdasarkan orientasi arah pada sisi graf, graf dibedakan menjadi graf berarah dan tak berarah. Graf berarah adalah graf yang mempunyai orientasi arah.



Gambar 2.2 Contoh dua graf berarah

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

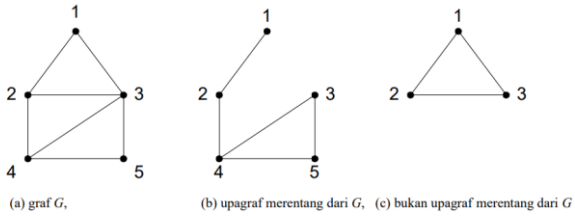
Dua buah simpul v_1 dan v_2 dikatakan terhubung jika terdapat lintasan dari v_1 ke v_2. Pada Gambar 2.1 graf G_3, simpul 1 dan 4 disebut terhubung karena terdapat lintasan dari simpul 1 ke 4, yaitu e_4 lalu e_1. Sebuah graf dikatakan terhubung jika setiap pasang simpul v_i dan v_j pada himpunan V terdapat lintasan dari v_i ke v_j. Contoh graf terhubung dapat dilihat pada Gambar 2.3.



Gambar 2.3 Contoh graf terhubung

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

Misalnya, $G_1=(V_1, E_1)$ adalah upagraf dari sebuah graf $G=(V, E)$ jika $V_1 \subseteq V$ dan $E_1 \subseteq E$. Upagraf G_1 disebut upagraf merentang jika $V_1=V$.

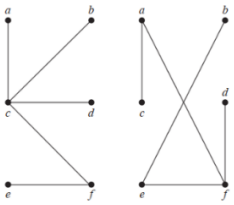


Gambar 2.4 Contoh upagraf merentang

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

B. Pohon

Pohon adalah graf terhubung sederhana yang tidak berarah dan tidak mempunyai sirkuit.



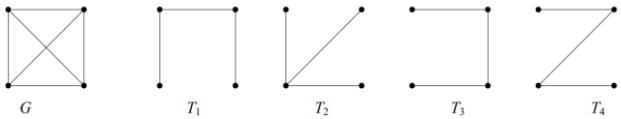
Gambar 2.5 Contoh dua pohon yang berbeda

Sumber: K. Rosen, *Discerete Mathematics and Its Applications 7th ed*. New York: McGraw-Hill, 2011, ch. 11.

Pohon memiliki beberapa sifat atau properti. Misalnya, jika G=(V, E) adalah graf sederhana yang tak berarah dan memiliki simpul sejumlah n simpul, semua pernyataan di bawah ini ekuivalen, yaitu:

1. G adalah pohon
2. Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal
3. G terhubung dan memiliki m=n-1 buah sisi
4. G tidak mempunyai sirkuit dan memiliki m=n-1 buah sisi
5. G tidak mempunyai sirkuit dan penambahan satu sisi pada G hanya akan membuat satu sirkuit
6. G terhubung dan semua sisinya adalah jembatan

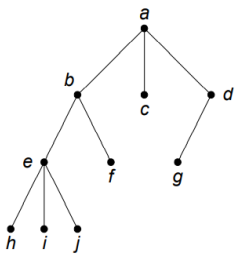
Pohon merentang dari sebuah graf terhubung adalah upagraf merentang yang berupa pohon. Misalnya G=(V, E) adalah graf terhubung yang tak berarah dan bukan pohon. G dapat diubah menjadi pohon $T=(V_1, E_1)$ dengan $V_1=V$ dan $E_1 \subseteq E$. T dapat diperoleh dengan memutus sirkuit pada graf. Pohon merentang yang berbobot minimum disebut pohon merentang minimum.



Gambar 2.6 Graf lengkap G dengan empat pohon merentangnya

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf

Pohon berakar adalah pohon yang satu buah simpulnya diperlakukan sebagai akar dan setiap sisinya diberi arah yang arahnya menjauh dari akar tersebut sehingga menjadi graf berarah. Biasanya arah panah pada sisi dibuang.

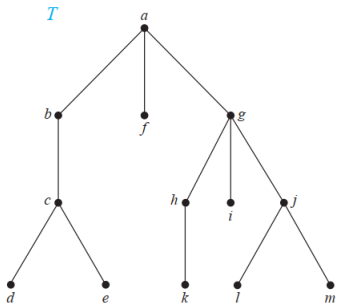


Gambar 2.7 Contoh pohon berakar

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag2.pdf

Pohon berakar mempunyai beberapa terminologi, yaitu:

1. Anak (*child* atau *children*) dan orang tua (*parent*)



Gambar 2.8 Pohon T

Sumber: K. Rosen, *Discerete Mathematics and Its Applications 7th ed*. New York: McGraw-Hill, 2011, ch. 11.

Jika v adalah simpul pada T selain akar, orang tua dari v adalah simpul unik u sedemikian sehingga ada sisi langsung dari u ke v. Jika u adalah orang tua dari v, v disebut anak dari u. Pada pohon T, anak-anak dari simpul a adalah b, f, dan g. Sebaliknya, a adalah orang tua dari anak-anak tersebut.

2. Lintasan (*path*)

Pada pohon T, lintasan dari a ke d adalah a, b, c, d. Panjang lintasan a ke d adalah 3.

3. Saudara kandung (*sibling*)

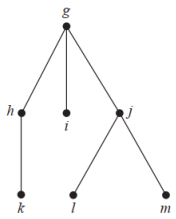
Pada pohon T, l adalah saudara kandung dari m. Namun, c bukan merupakan saudara kandung m karena orang tua mereka berbeda. Maka dari itu, saudara kandung adalah simpul yang mempunyai orang tua yang sama.

4. Leluhur (*ancestors*) dan keturunan (*descendants*)

Leluhur dari sebuah simpul kecuali akar adalah simpul dalam lintasan dari akar ke simpul tersebut kecuali simpul tersebut (termasuk akar). Keturunan dari sebuah simpul v adalah simpul yang mempunyai v sebagai leluhurnya. Pada pohon T, leluhur dari e adalah c,b, dan a. Keturunan dari b adalah c,d, dan e.

5. Upapohon (*subtree*)

Jika a adalah sebuah simpul di dalam pohon, upapohon dengan a sebagai akarnya adalah upagraf dari pohon yang mengandung simpul a, semua keturunannya, dan semua sisi yang berisisian dengan keturunannya.



Gambar 2.9 Upapohon dari pohon T dengan g sebagai akarnya

Sumber: K. Rosen, *Discerete Mathematics and Its Applications 7th ed*. New York: McGraw-Hill, 2011, ch. 11.

6. Derajat (*degree*)

Derajat sebuah simpul adalah jumlah upapohon atau jumlah anak pada simpul tersebut. Pada pohon T, derajat g adalah 3, sedangkan derajat c adalah 2. Derajat yang dimaksudkan di sini adalah derajat keluar. Derajat maksimal dari semua simpul adalah derajat pohon itu sendiri. Dengan demikian, derajat pohon T adalah 3.

7. Daun (*leaf*)

Daun adalah simpul yang berderajat nol atau tidak mempunyai anak. Pohon T mempunyai beberapa daun, yaitu d, e, k, l, dan m.

8. Simpul dalam (*vertices*)

Simpul dalam adalah simpul yang mempunyai anak. Pada pohon T, simpul b, c, g, h, j adalah simpul dalam.

9. Aras (*level*) atau tingkat

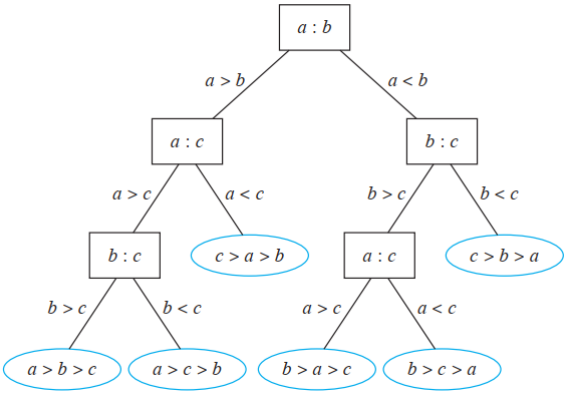
Akar mempunyai aras 0, sedangkan simpul lainnya mempunyai aras, yaitu 1 ditambah panjang lintasan dari akar ke simpul tersebut.

10. Tinggi (*height*) atau kedalaman (*depth*)

Tinggi atau kedalaman dari suatu pohon adalah aras maksimum pohon tersebut. Kedalaman pohon T adalah 3.

Pohon terurut adalah pohon berakar yang urutan anak-anaknya penting. Sebuah pohon berakar disebut pohon n-ary jika setiap simpul dalamnya mempunyai maksimal n anak. Pohon tersebut dikatakan pohon n-ary penuh jika setiap simpul dalamnya tepat mempunyai n anak. Pohon n-ary dengan n=2 disebut pohon biner. Dalam pohon biner, terdapat istilah anak kiri dan anak kanan. Karena ada perbedaan urutan anak, pohon biner adalah pohon terurut.

Pohon keputusan (*decision tree*) adalah pohon berakar yang setiap simpulnya merepresentasikan sebuah kemungkinan hasil dari sebuah keputusan dan daunnya merepresentasikan kemungkinan solusi yang bisa didapatkan. Pohon keputusan merupakan salah satu aplikasi dari pohon berakar dan digunakan untuk memodelkan suatu masalah sehingga bisa ditemukan solusinya. Salah satu contohnya, yaitu pohon keputusan untuk mengurutkan tiga buah elemen yang dapat dilihat pada Gambar 2.10. Pada pohon keputusan tersebut, terdapat enam daun yang berkorespondensi terhadap enam kemungkinan solusi yang bisa didapatkan.



Gambar 2.10 Pohon keputusan mengurutkan 3 elemen

Sumber: K. Rosen, *Discerete Mathematics and Its Applications 7th ed*. New York: McGraw-Hill, 2011, ch. 11.

C. Struktur Data

Data adalah istilah yang mengacu pada semua jenis informasi, sedangkan struktur data berhubungan dengan cara data diorganisasikan. Kebanyakan struktur data mempunyai empat operasi dasar, yaitu:

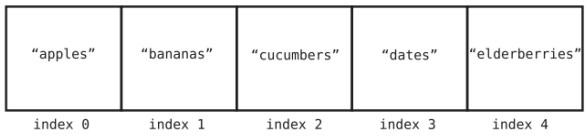
1. *Read*: mencari sesuatu pada posisi tertentu, misalnya mencari nilai pada suatu indeks tertentu di dalam *array*
2. *Search*: mencari suatu nilai di dalam struktur data, misalnya mencari eksistensi suatu nilai di dalam *array*
3. *Insert*: menambah sebuah nilai baru, misalnya menambahkan nilai 4 ke dalam *array of integer*
4. *Delete*: menghapus suatu nilai tertentu, misalnya menghapus mahasiswa dengan nama Budi di dalam *array of* mahasiswa

Struktur data ada banyak jenisnya baik yang sudah tersedia dalam bahasa pemrograman maupun yang harus dibuat menggunakan konsep ADT atau *Abstract Data Type*. Beberapa jenis struktur data, antara lain:

1. *Array*

Array terdiri dari sejumlah elemen bertipe sama yang

diletakkan di memori secara kontigu. Elemennya diakses melalui indeks yang dimulai dari nol untuk kebanyakan bahasa pemrograman.

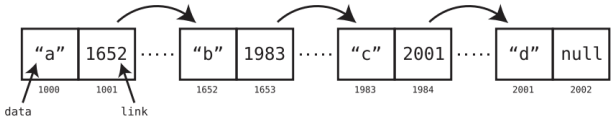


Gambar 2.11 Visualisasi *array of string* dengan empat elemen

Sumber: J. Wengrow, *A Common-Sense Guide to Data Structures and Algorithms 2nd ed*. Raleigh: The Pragmatic Bookshelf, 2020, ch. 1.

2. *Linked list*

Linked list terdiri dari sejumlah elemen yang diimplementasikan dengan *pointer*. *Linked list* terdiri atas *node* yang terkait dengan *node* lain. *Node* merupakan sebuah *tuple* yang terdiri dari sebuah nilai dengan tipe tertentu dan sebuah penunjuk atau *pointer* atau *link* ke *node* lain. Penunjuk bisa tidak menunjuk ke mana pun (*null*). Maka dari itu, struktur data ini mirip seperti *array*, tetapi penyimpanan elemennya dimungkinkan tidak kontigu. Karena memori dialokasi sesuai kebutuhan, secara umum struktur data ini mengorbankan efisiensi ruang (memori) demi efisiensi waktu.



Gambar 2.12 Visualisasi *linked list* dengan empat elemen

Sumber: J. Wengrow, *A Common-Sense Guide to Data Structures and Algorithms 2nd ed*. Raleigh: The Pragmatic Bookshelf, 2020, ch. 14.

3. *Stack*

Stack adalah *list* linier yang memenuhi beberapa aturan. Pertama, elemen puncaknya (*top*) hanya satu-satunya elemen yang bisa dibaca. Kedua, operasi penyisipan hanya bisa dilakukan di atas *top*. Terakhir, operasi penghapusan hanya bisa dilakukan pada *top*. Maka dari itu, operasi pada *stack* menggunakan prinsip *Last In, First Out* (LIFO).

4. *Queue*

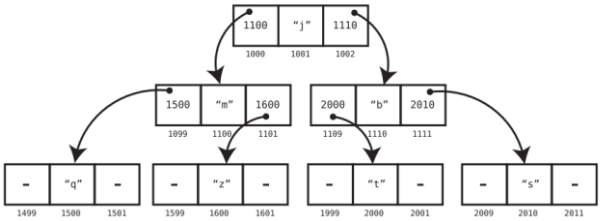
Queue adalah sederetan elemen yang memenuhi beberapa aturan, Pertama, *queue* mengenali elemen pertama (*head*) dan elemen terakhirnya (*tail*). Kedua, *head* hanya satu-satunya elemen yang bisa dibaca. Ketiga, operasi penyisipan selalu dilakukan setelah *tail*, sedangkan operasi penghapusan selalu dilakukan pada *head*. Maka dari itu, operasi pada *queue* menggunakan prinsip *First In, First Out* (FIFO).

5. *Hash table*

Hash table merupakan deretan sepasang nilai yang terdiri dari *key* dan *value*. Fungsi *hash* adalah fungsi yang digunakan untuk memetakan data berukuran "berapa pun" menjadi nilai yang beukuran tetap atau tertentu. Fungsi *hash* mengubah *key* menjadi *hash* yang digunakan sebagai indeks.

6. Pohon

Pohon adalah salah satu struktur data berbasis *node*. Di dalam pohon, setiap *node* bisa mempunyai hubungan lebih dari satu dengan *node* lainnya. Pada struktur data ini, terdapat satu elemen yang dibedakan dari yang lain, yaitu akar. Jika ada elemen lainnya, elemen-elemen tersebut dibagi menjadi beberapa subhimpunan yang *disjoint* dan masing-masing subhimpunan tersebut merupakan sebuah pohon juga.



Gambar 2.13 Visualisasi *struktur data pohon*

Sumber: J. Wengrow, *A Common-Sense Guide to Data Structures and Algorithms 2nd ed*. Raleigh: The Pragmatic Bookshelf, 2020, ch. 15.

Jenis pohon yang biasa digunakan dalam struktur data adalah *Binary Search Tree* (BST). Sebuah BST hanya mempunyai masing-masing maksimal satu anak kanan dan kiri. Keturunan dari *node* kiri mempunyai nilai yang lebih kecil daripada nilai *node* tersebut, begitu sebaliknya untuk keturunan dari *node* kanan.

7. Graf

Graf adalah struktur data yang biasa merepresentasikan keterhubungan antarobjek. Sebuah graf terdiri sekumpulan simpul dan sisi. Di dalam struktur data, graf bisa diimplementasikan dengan *adjacency matrix, adjacency list, incidence matrix, incidence list, edge list*.

III. Pohon Keputusan dalam Memilih Struktur Data

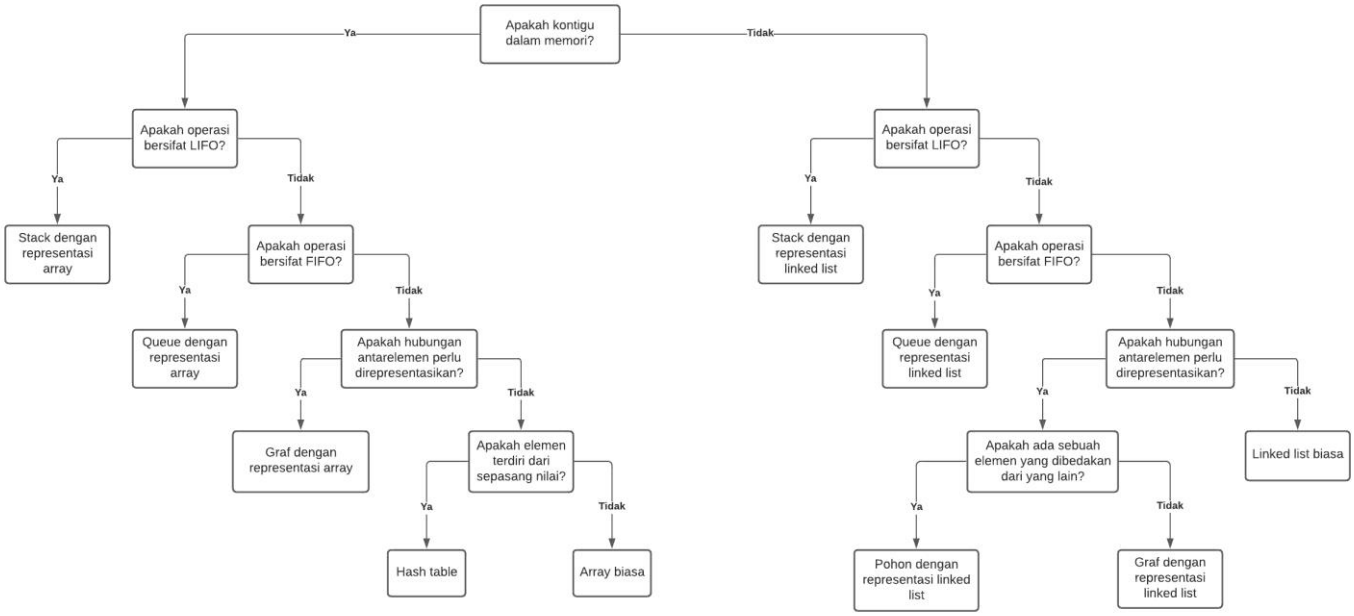
Pertama-pertama, untuk membuat pohon keputusan dalam memilih struktur data, perlu diketahui terlebih dahulu masing-masing bentuk implementasi struktur data. Secara umum, struktur data yang diimplementasikan dalam pohon keputusan ini menggunakan struktur data *array* dan *linked list. Array* bisa mengimplementasikan *queue*, *stack*, dan graf. Sementara itu, *linked list* juga bisa mengimplementasikan tiga struktur data tersebut dan juga struktur data pohon. Dalam pohon keputusan yang dibuat dalam makalah ini, *hash table* termasuk kategori struktur data yang diimplementasikan dengan *array*. Hal ini karena *hash table* disimpan dalam format *array*.

Penentuan akar pada pohon keputusan ini akan melihat model struktur data di dalam memori, Struktur data *array* kontigu dalam memori, sedangkan struktur data berkait atau *linked list* sebaliknya. Hal ini merupakan karakteristik yang paling membedakan antara dua struktur tersebut. Oleh karena itu, yang menjadi akar pohon keputusan ini adalah pertanyaan "Apakah kontigu dalam memori?".

Selanjutnya, untuk menentukan simpul dalam lainnya, dapat dilihat melalui karakteristik masing-masing struktur data. Karakteristik ini dapat berupa operasi yang dilakukannya, tujuan umum dipakainya, atau ciri khusus sebuah struktur data. Misalnya, operasi pada *queue* menerapkan prinsip FIFO, sedangkan *stack* menerapkan prinsip LIFO, *hash table* dipakai jika elemennya terdiri dari sepasang nilai, atau terdapat elemen

yang dibedakan dari yang lain pada struktur data pohon. Oleh karena itu, yang menjadi simpul dalam pada pohon keputusan ini adalah pertanyaan tentang operasi yang akan dilakukan, tujuan dari penggunaan struktur data, atau ciri khusus yang terdapat pada struktur data.

Pohon keputusan pemilihan struktur data dapat dilihat pada Gambar 3.1. Simpul pada pohon keputusan ini ada sebanyak 19 simpul dan sepuluh simpul di antaranya merupakan daun. Daun pada pohon keputusan ini merepresentasikan solusi dari pohon keputusan ini, yaitu struktur data yang cocok digunakan. Maksud dari *array* biasa dan *linked list* biasa pada solusi dari pohon keputusan ini adalah *array* dan *linked list* yang tidak merepresentasikan struktur data lain.



Gambar 3.1 Pohon keputusan dalam memilih struktur data

Sumber: Dokumen pribadi penulis

IV. STUDI KASUS

A. Studi Kasus Pertama

Pada studi kasus pertama ini, misalnya seseorang mendapatkan tugas untuk membuat sebuah program yang memodelkan sistem pemesanan makanan di sebuah restoran. Spesifikasi program tersebut adalah program menerima masukan pengguna sebanyak orang yang berada di restoran tersebut, yaitu nama makanan, jumlah makanan, dan waktu kedatangan orang tersebut. Waktu pembuatan makanan diasumsikan sama untuk tiap makanan dan harga makanan sudah tertera pada spesifikasi tugas. Pesanan yang dibuat terlebih dahulu adalah pesanan orang yang datang pertama kali. Selain itu, program juga harus diimplementasikan menggunakan struktur data *array*.

Untuk memilih struktur data untuk pembuatan program pada ilustrasi diatas, dapat digunakan pohon keputusan pada Gambar 3.1. Pertama, jika ditinjau dari akar, yaitu "Apakah kontigu dalam memori?", jawabannya adalah "Ya" karena program harus diimplementasikan menggunakan *array* sehingga keputusan akan menuju ke anak kiri. Selanjutnya pada simpul "Apakah operasi bersifat LIFO?", jawabannya adalah "Tidak" karena orang yang datang terakhir akan dibuatkan pesanannya paling terakhir sehingga tidak mungkin pembuatan pesanan orang tersebut selesai terlebih dahulu daripada orang lain. Oleh karena itu, keputusan akan menuju simpul "Apakah operasi bersifat FIFO?". Jawaban dari pertanyaan ini adalah "Ya" karena pesanan orang yang datang pertama akan dibuatkan terlebih dahulu sehingga pembuatan pesanan orang tersebut selesai terlebih dahulu daripada orang lain. Pada akhirnya keputusan akan menuju daun "Queue dengan representasi *array*". Daun tersebut merupakan solusi dari masalah pemilihan struktur data pada ilustrasi studi kasus pertama ini.

B. Studi Kasus Kedua

Pada studi kasus kedua, misalnya seseorang ingin membuat program yang merepresentasikan hubungan pertemanan di dalam sebuah jurusan dalam suatu universitas. Setiap orang bisa berteman dengan orang lain, tetapi tidak dengan dirinya sendiri. Jumlah seseorang bisa bertambah jika berkenalan dengan orang lain. Pada program ini, operasi umum yang harus dilakukan adalah penambahan teman pada seseorang, pengecekan status pertemanan antara dua orang, perhitungan jumlah teman seseorang, dan fungsi yang mengembalikan nama orang dengan teman terbanyak. Program tersebut tidak perlu kontigu di dalam memori.

Persoalan pemilihan struktur data pada studi kasus kedua ini juga bisa diselesaikan menggunakan pohon keputusan yang terdapat pada Gambar 3.1 seperti pada studi kasus pertama. Pertama-tama, pertanyaan pada akar pohon dapat dijawab

dengan “Tidak” karena berdasarkan spesifikasi program, program tidak perlu diimplementasikan secara kontigu di dalam memori. Selanjutnya, keputusan akan mengarah pada simpul “Apakah operasi bersifat LIFO?”. Jawaban dari simpul tersebut adalah “Tidak” karena operasi-operasi yang dilakukan pada program tersebut tidak ada yang bersifat LIFO. Lalu, keputusan akan menuju simpul “Apakah operasi bersifat FIFO?”. Jawaban dari simpul ini sama dengan jawaban dengan simpul sebelumnya, yaitu “Tidak” karena tidak ada operasi yang bersifat FIFO pada program tersebut. Oleh karena itu, keputusan akan menuju simpul “Apakah hubungan antarelemen perlu diimplementasikan?”. Jawaban dari simpul ini jelas “Ya” karena program yang ingin dibuat merepresentasikan hubungan pertemanan. Lalu, simpul selanjutnya, yaitu “Apakah ada sebuah elemen yang dibedakan dari yang lain?” mendapatkan jawaban “Tidak”. Hal ini karena semua orang tersebut terdapat pada satu jurusan dan universitas yang sama serta dapat berteman satu sama lain. Pada akhirnya, keputusan akan menuju daun “Graf dengan representasi *linked list*”. Daun tersebut merupakan solusi dari permasalahan ini.

V. KESIMPULAN

Pemilihan struktur data dalam pembuatan program menjadi hal yang penting untuk dicermati. Secara umum representasi struktur data dibagi menjadi dua, yaitu representasi dengan *array* dan *linked list*. Ada banyak struktur data yang dapat dipilih dan struktur data tersebut memiliki karakteristiknya masing-masing. Maka dari itu, pohon keputusan dapat membantu pemilihan struktur data dengan melihat persoalan pada program yang ingin dibuat. khususnya model setiap elemen pada data serta operasi-operasi yang dilakukan pada program. Berdasarkan pohon keputusan yang telah dibuat, terdapat kemungkinan sepuluh solusi atau pilihan struktur data yang dapat digunakan.

VI. UCAPAN TERIMA KASIH

Pertama-tama, penulis mengucapkan terima kasih yang sebesar-besarnya kepada Tuhan Yang Maha Esa karena atas rahmat dan karunia-Nya, penulis bisa menyelesaikan penulisan makalah ini dengan baik. Penulis juga berterima kasih kepada orang tua dan keluarga penulis karena sudah memberikan dukungan terhadap pelaksanaan kuliah penulis. Selain itu, penulis juga mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi, M.T., Ibu Dra. Harlili, M.Sc., dan Ibu Dr. Nur Ulfa Maulidevi, S.T., M.Sc. atas bimbingan dan ilmu yang diajarkan selama perkuliahan Matematika Diskrit semester I tahun ajaran 2021/2022.

REFERENSI

[1] K. Rosen, *Discerete Mathematics and Its Applications*, 7th ed. New York: McGraw-Hill, 2011, ch. 11.

[2] R. Munir, *Matematika Diskrit*, 3rd ed. Bandung: Penerbit Informatika, 2010, ch. 9.

[3] J. Wengrow, *A Common-Sense Guide to Data Structures and Algorithms*, 2nd ed. Raleigh: The Pragmatic Bookshelf, 2020.

[4] Robert Sedgewick and Kevin Wayne, *Algorithms*, 4th ed. Westford: Addison-Wesley, 2011.

[5] R. Munir, “Pohon (Bag.1).” Homepage Rinaldi Munir. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf (diakses pada tanggal 3 Desember 2021).

[6] R. Munir, “Pohon (Bag.2).” Homepage Rinaldi Munir. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag2.pdf (diakses pada tanggal 3 Desember 2021).

[7] R. Munir, “Graf (Bag.1).” Homepage Rinaldi Munir. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020- 2021/Graf-2020-Bagian1.pdf (diakses pada tanggal 11 Desember 2021).

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.

Jakarta, 11 Desember 2021

Mohamad Daffa Argakoesoemah 13520118