BAB VII
Tree

Tujuan

1. Mempelajari variasi bagian-bagian dari tree sebagai suatu bentuk struktur tak linier
2. Mempelajari beberapa hubungan fakta yang direpresentasikan dalam sebuah tree, sehingga mampu merepresentasikan tree dalam permasalahan aslinya
3. Memahami bagaimana menulis program untuk tree, dan bagaimana mengartikannya kembali dalam bentuk permasalahan aslinya

---

Dalam bab ini kita akan mempelajari satu bentuk struktur data tak linier yang mempunyai sifat-sifat khusus, yang dinamakan pohon (*tree*). Struktur ini biasanya digunakan untuk menggambarkan hubungan yang bersifat hirarkis antara elemen-elemen yang ada.

Silsilah keluarga, hasil pertandingan yang berbentuk turnamen, atau struktur organisasi dari sebuah perusahaan adalah contoh dalam organisasi tree. Dalam pemrograman, sebuah pohon terdiri dari elemen-elemen yang dinamakan *node*(simpul) yang mana hubungan antar simpul bersifat hirarki. Simpul yang paling atas dari hirarki dinamakan *root*. Simpul yang berada di bawah root secara langsung, dinamakan anak dari root, yang mana biasanya juga mempunyai anak di bawahnya. Sehingga bisa disimpulkan, kecuali root, masing-masing simpul dalam hirarki mempunyai satu induk (parent).

Jumlah anak sebuah simpul induk sangat bergantung pada jenis dari pohon. Jumlah anak dari simpul induk ini dinamakan faktor percabangan. Pada bab ini pembahasan difokuskan pada *binary tree*, yaitu pohon yang mempunyai faktor percabangan 2.

7.1 Deskripsi dari *Binary Tree*

Sebuah binary tree adalah sebuah pengorganisasian secara hirarki dari beberapa buah simpul, dimana masing-masing simpul tidak mempunyai anak lebih dari 2. Simpul

yang berada di bawah sebuah simpul dinamakan anak dari simpul tersebut. Simpul yang berada di atas sebuah simpul dinamakan induk dari simpul tersebut.

Masing-masing simpul dalam binary tree terdiri dari tiga bagian : sebuah data dan dua buah pointer yang dinamakan pointer kiri dan kanan. Dengan menggunakan tiga bagian ini, kita menampilkan sebuah binary tree dengan melakukan setting pada pointer kiri dan kanan untuk menghubungkan sebuah simpul dengan anak-anaknya. Jika sebuah simpul tidak mempunyai anak pada pointer kiri atau kanan, kita melakukan setting pada pointer tersebut pada NULL, yang menunjukkan akhir dari percabangan adalah pada simpul tersebut. Sebuah percabangan adalah kumpulan dari simpul-simpul yang dimulai dari sebuah root dan diakhiri dengan sebuah simpul terakhir. Simpul terakhir adalah simpul dari tree yang tidak mempunyai anak. Kadang-kadang ketika kita bekerja dengan beberapa pohon dalam satu waktu, maka pohon-pohon tersebut dinamakan sebuah hutan.

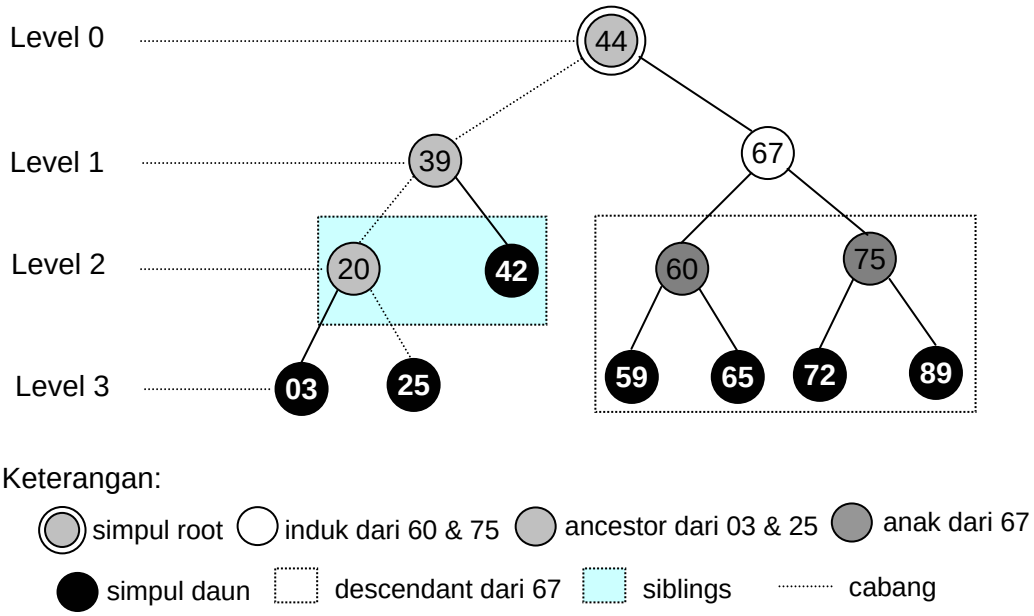
7.2 Istilah-Istilah Dasar

Simpul juga mempunyai *sibling, descendants*, dan *ancestors*. ***Sibling*** dari sebuah simpul adalah anak lain dari induk simpul tersebut. ***Descendants*** dari sebuah simpul adalah semua simpul-simpul merupakan cabang (berada di bawah) simpul tersebut. ***Anchestors*** dari sebuah simpul adalah semua simpul yang berada di atas antara simpul tersebut dengan *root*. Penampilan dari sebuah tree akan ditampilkan dengan berat dari tree tersebut, angka yang menunjukkan jumlah level yang ada di dalamnya.

Tingkat suatu simpul ditentukan dengan pertama kali menentukan akar sebagai bertingkat 1. Jika suatu simpul dinyatakan sebagai tingkat N, maka simpul-simpul yang merupakan anaknya akan berada pada tingkatan N + 1.

Tinggi atau kedalaman dari suatu pohon adalah tingkat maksimum dari simpul dalam pohon dikurangi dengan 1.

Selain tingkat, juga dikenal istilah derajad (degree) dari suatu simpul. Derajad suatu simpul dinyatakan sebagai banyaknya anak atau turunan dari simpul tersebut



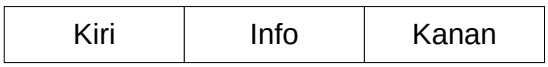
Gambar 7.1 Ilustrasi Sebuah Pohon Biner dengan Kedalaman 3

7.3 Penyajian Pohon Biner

Implementasi pohon biner dalam pemrograman dapat dilakukan dengan beberapa cara. Cara pertama adalah dengan menggunakan link list dan cara lain adalah dengan menggunakan cara berurutan. Dalam bab ini kita lebih banyak mempelajari cara implementasi biner menggunakan link list.

7.3.1 Deklarasi Pohon

Jika kita memperhatikan setiap simpul dalam pohon biner, kita bisa menyusun struktur data yang tepat dari simpul-simpul tersebut. Kita dapat melihat bahwa dalam setiap simpul selalu berisi dua buah pointer untuk menunjuk ke cabang kiri dan cabang kanan, dan informasi yang akan disimpan dalam simpul tersebut. Dengan memperhatikan hal ini, simpul dalam pohon biner disajikan sebagai berikut:



Gambar 7.1 Simpul Pada Pohon Biner

Sesuai dengan gambar 7.1, maka deklarasi list yang sesuai adalah:

```
typedef char TypeInfo;
typedef struct Simpul *Tree;
```

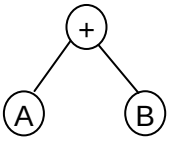
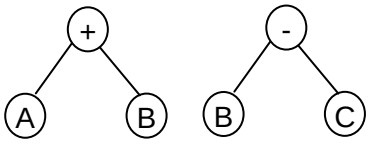
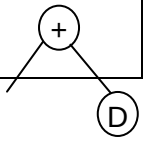
```
struct Simpul {
          TypeInfo Info;
          tree Kiri,     /* cabang kiri */
               Kanan;     /* cabang kanan */
};
```

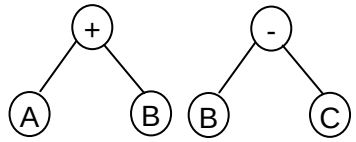
Program 7.1 Deklarasi dari Tree

7.3.2 Membuat Pohon Biner

Tabel 7.1 memperlihatkan bagaimana proses pembentukan pohon biner dari sebuah persamaan.

Tabel 7.1 Pembentukan Pohon Biner dari Persamaan: (A+B)*((B-C)+D)

Karakter yang dibaca	Tumpukan operator	Tumpukan operand	Pohon biner yang terbentuk
(	(		
A	(	A	
+	(+	A	
B	(+	AB	
)		+	
*	*	+	
(	*(	+	
(	*((	+	
B	*((	+B	
-	*((-	+B	
C	*((-	+BC	
)	(*	+-	
+	*(+	+-	
D	*(+	+-D	
)	*	++	

7.4 Metode Traversal

Proses traversing dari sebuah binary tree artinya melakukan kunjungan pada setiap simpul pada suatu pohon biner tepat satu kali. Dengan melakukan kunjungan secara lengkap, kita akan memperoleh informasi secara linear yang tersimpan dalam pohon biner. Dalam melakukan kunjungan pada sebuah pohon biner, kita akan memperlakukan hal yang sama pada setiap simpul pada cabang-cabangnya.

Urutan informasi yang tersimpan dalam pohon biner akan berbeda jika letak simpulnya ditukar. Dalam melakukan kunjungan kita perlu memperhatikan hal ini. Dengan alasan inilah kita bisa melakukan kunjungan dengan tiga cara, yaitu kunjungan secara preorder, inorder, dan secara postorder. Selain itu, berdasarkan kedudukan setiap simpul dalam pohon, juga bisa dilakukan kunjungan secara levelorder. Ketiga macam kunjungan yang pertama bisa dilaksanakan secara rekursif.

Kunjungan preorder, juga disebut dengan *depth first order*, menggunakan urutan:

- Cetak isi simpul yang dikunjungi
- Kunjungi cabang kiri
- Kunjungi cabang kanan

Kunjungan secara inorder, juga sering disebut dengan *symmetric order*, menggunakan urutan:

- Kunjungi cabang kiri
- Cetak isi simpul yang dikunjungi
- Kunjungi cabang kanan

Kunjungan secara postorder menggunakan urutan:

- Kunjungi cabang kiri
- Kunjungi cabang kanan
- Cetak isi simpul yang dikunjungi

7.5 Kesimpulan

1. Tree adalah sebuah struktur linier, biasanya digunakan untuk menggambarkan hubungan yang bersifat hirarkis antara elemen-elemen yang ada.

2. Ada beberapa istilah dalam tree ini, yang mana masing-masing istilah mempunyai arti dalam kaitannya dengan hirarki antar elemen dalam tree tersebut, seperti sibling, descendant dsb.

7.6 Latihan

1. Tuliskan binary tree yang terbentuk dari ekspresi-ekspresi di bawah ini:
   a. (a+b)/(c-d*e)+e+g*h/a
   b. –x-y*z+(a+b+c/d*e)
2. Gambarkan lintasan yang dilalui dari tree yang terbentuk pada soal di atas jika dikunjungi secara preorder untuk soal a dan postorder untuk soal b. Tuliskan juga hasil notasi prefix (soal a) dan postfix (soal b)
3. Perhatikan data dari sebuah tree tentang sebuah silsilah keluarga di bawah ini:

   Data set : {Ahmad, Zulaikha, Zahro, Salman, Salsabila, Zaki, Sholahuddin}

   Root : {Ahmad}

   Anchestor dari Sholahuddin : {Zaki, Ahmad}

   {Zulaikha, Zahro, Zaki} adalah **siblings**

   {Salman, Salsabila} adalah **descendant** dan keduanya anak (**children**) dari Zahro

   Dari keterangan di atas, bentuklah tree dari silsilah keluarga di atas!