

## Modul 9

# Pohon Biner

Selama ini kita telah mempelajari struktur dalam bentuk *sequential* yakni sebuah struktur data di mana sebuah elemen berada “sebelum” atau “sesudah” elemen lain. Namun terkadang kita memerlukan bentuk yang tidak linier. Nah, di modul ini kita akan mempelajari sebuah struktur hirarkis yang kita namai “pohon”.

### 9.1 Pengantar dan contoh

Pohon atau *Tree* adalah sebuah struktur data yang menyimpan elemen-elemen informasi dalam bentuk hirarkis. Struktur ini terdiri dari *nodes* (simpul) dan *edges* (garis penghubung). Antar simpul mempunyai hubungan “ortu,” “anak,” “leluhur,” “keturunan,” dan sebagainya. Data disimpan dalam *simpul*<sup>1</sup> dan pasangan simpul dihubungkan dengan sebuah *pinggir*<sup>2</sup>.

Pinggir-pinggir ini mewakili relasi antar simpul yang terhubungkan dengan panah/garis untuk membentuk struktur hirarkis yang menyerupai pohon terbalik dengan cabang, daun, dan akar.

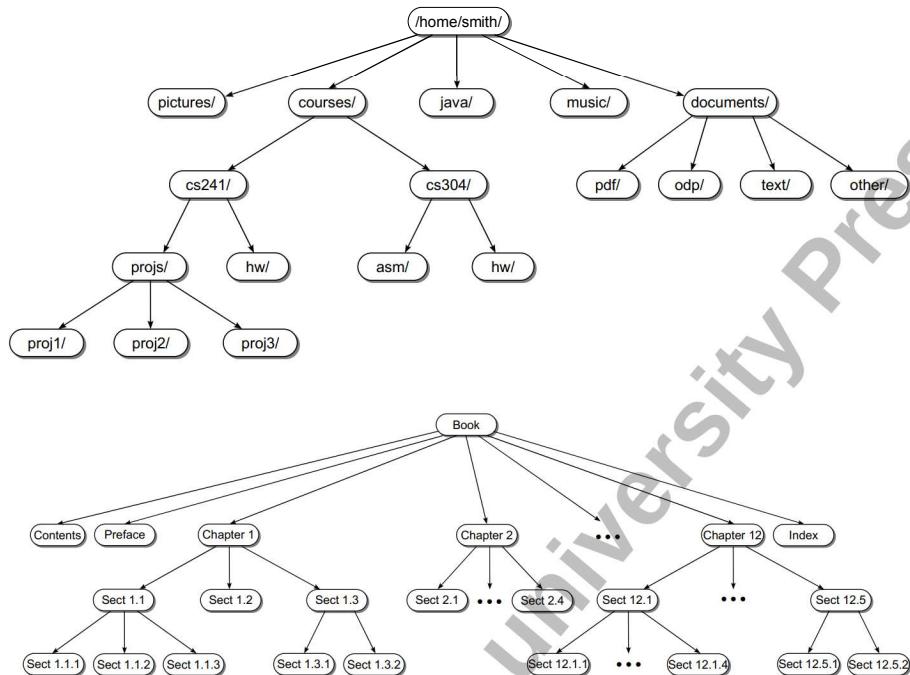
Sebuah contoh struktur pohon adalah direktori dan sub-direktori di sistem berkas pada hard-disk komputer. Pohon bisa dipakai untuk merepresentasikan data yang terstruktur, yang mana akan menghasilkan pembagian data menjadi data yang lebih kecil dan lebih kecil lagi. Contohnya adalah isi sebuah buku yang dibagi menjadi bab, section, dan seterusnya. Gambar 9.1 menunjukkan dua contoh di atas<sup>3</sup>.

**Latihan 9.1** Sebuah contoh pohon yang lain adalah struktur pengalaman domain di web. Misal domain .id berperan sebagai akarnya. Maka di bawahnya ada sub-domain seperti go.id, co.id, ac.id, net.id, sch.id. Dan, contoh berikutnya, di bawah ac.id ada domain seperti ugm.ac.id, uns.ac.id, ui.ac.id, itb.ac.id, ums.ac.id. Di bawah ums.ac.id ada (di antaranya) pmb.ums.ac.id, www.ums.ac.id, krsonline.ums.ac.id, star.ums.ac.id, fki.ums.ac.id. Sekarang gambarlah struktur pohon domain-domain ini seingat kamu. □

<sup>1</sup>Bahasa Inggrisnya: *node*

<sup>2</sup>Bahasa Inggrisnya: *edge*

<sup>3</sup>Gambar-gambar di bab ini diambil dari Rance D. Necaise, *Data Structures and Algorithms Using Python*, John Wiley and Sons, 2011.



**Gambar 9.1:** Contoh-contoh struktur pohon. Atas: direktori dan sub-direktori di sebuah komputer dengan sistem operasi LINUX. Bawah: pembagian isi sebuah buku.

Pada bab ini kita akan mempelajari struktur pohon yang lebih khusus, yakni **pohon biner** (*binary tree*). Pohon biner adalah pohon di mana setiap simpul dapat mempunyai anak paling banyak dua<sup>4</sup>.

## 9.2 Istilah-istilah

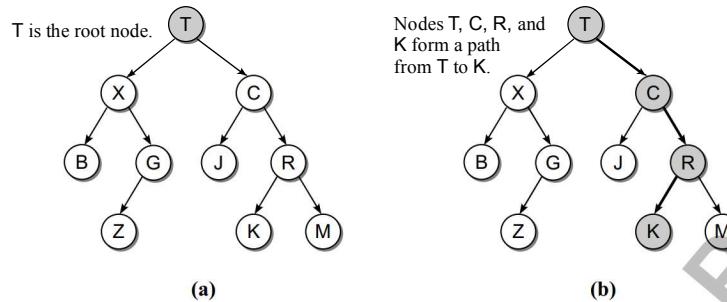
Kita akan menggunakan berbagai istilah yang sebaiknya kamu ketahui maknanya sejak sekarang.

### Akar

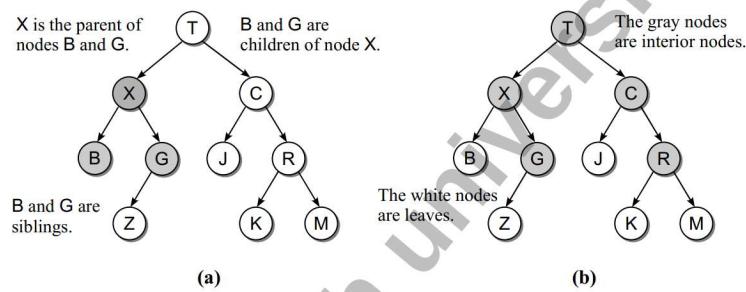
**Simpul akar** (dari frase bahasa Inggris *root node*) adalah simpul yang paling atas. Simpul akar ini adalah simpul yang tidak mempunyai pinggir yang datang. Lihat contoh pada Gambar 9.2(a) di mana akar pohnnya adalah T.

---

<sup>4</sup>Ini bukan program pemerintah RI lewat BKKBN :-). Kita mempelajari pohon biner terlebih dahulu karena lebih sederhana operasinya. Jika kamu sudah menguasai operasi pada pohon biner, maka memahami operasi pada pohon yang lebih umum akan meng-extend konsepnya saja



Gambar 9.2: Contoh pohon biner. (a) pohon dengan akar T. (b) pohon dengan jalur dari T ke K yang dibentuk dari simpul-simpul T-C-R-K.



Gambar 9.3: Contoh pohon biner dengan: (a) hubungan ortu, anak, dan saudara; dan (b) perbedaan antara simpul interior dan simpul daun.

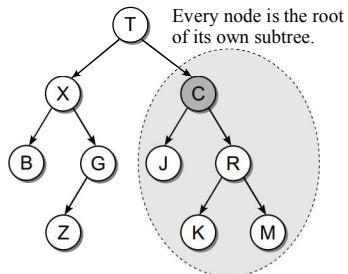
### Jalur

Setelah di simpul akar, simpul-simpul lain diakses melalui pinggir-pinggir. Dimulai dari akar, berjalan mengikuti arah panah sampai simpul yang dituju dicapai. Simpul-simpul yang dikunjungi saat meniti pinggir-pinggir, dari simpul awal sampai simpul tujuan, membentuk suatu *jalur* (diterjemahkan bebas dari kata Bahasa Inggris *path*). Seperti ditunjukkan di Gambar 9.2(b), simpul-simpul yang berlabel T, C, R, dan K membentuk jalur dari T ke K.

### Ortu

Setiap simpul, kecuali simpul akar, mempunyai satu *simpul ortu* (dari frase Bahasa Inggris *parent node*), yang diidentifikasi dari pinggir yang datang. Sebuah simpul hanya dapat mempunyai satu ortu (yakni, hanya satu pinggir datang). Ini menjadikan adanya jalur yang unik dari simpul akar ke simpul-simpul lain yang manapun di pohon itu<sup>5</sup>. Pada Gambar 9.3(a), simpul X adalah ortu dari simpul B dan G.

<sup>5</sup>Maksudnya, hanya ada satu jalur yang bisa dibuat dari akar ke simpul lain. Contoh: pada Gambar 9.2(b), *satu-satunya* jalur dari simpul akar ke simpul K adalah T-C-R-K. Tidak ada yang lain.



Gambar 9.4: Sebuah subpohon dengan simpul akar C.

### Anak

Setiap simpul pada pohon biner dapat mempunyai satu atau dua *anak*, yang membuat hirarki ortu-anak. Anak sebuah simpul diidentifikasi dari pinggir yang mengarah ke luar. Sebagai contoh, simpul B dan G keduanya adalah anak dari simpul X. Semua simpul yang mempunyai ortu yang sama dinamakan *saudara* (diterjemahkan bebas dari kata Bahasa Inggris *sibling*), tapi tidak ada akses langsung antar saudara. Sehingga kita tidak bisa secara langsung mengakses simpul C dari simpul X dan sebaliknya.

### Simpul Interior dan Simpul Daun

Simpul yang memiliki sekurangnya satu anak disebut *simpul interior* atau simpul dalam. Simpul yang tidak mempunyai anak disebut *simpul daun*. Pada Gambar 9.3(b), simpul interior diberi latar belakang abu-abu dan simpul daun diberi latar belakang warna putih.

### Subpohon

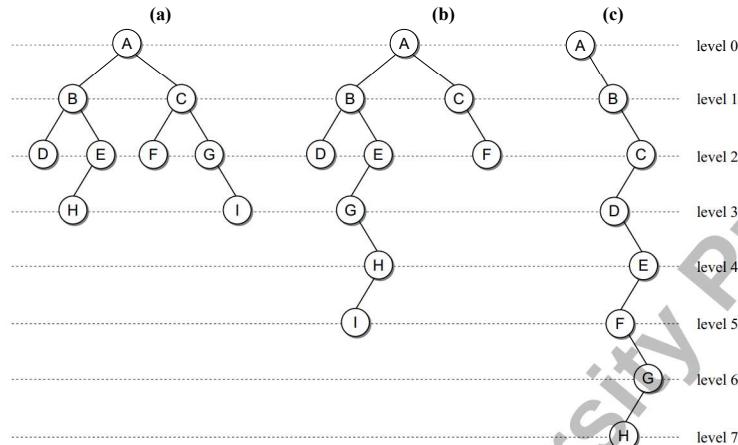
Dilihat dari pengertiannya, sebuah pohon adalah struktur rekursif. Setiap simpul dapat menjadi akar dari *subpohon* (subtree)-nya sendiri, yang terdiri atas simpul-simpul dan pinggir-pinggir dari pohon utamanya yang lebih besar. Gambar 9.4 menunjukkan subpohon dengan C sebagai akarnya.

### Kerabat

Semua simpul pada sebuah subpohon adalah *keturunan* dari simpul akar subpohon itu. Pada pohon contoh di Gambar 9.4, simpul J, R, K, dan M adalah keturunan dari simpul C. *Leluhur* sebuah simpul adalah ortu, kakek, buyut, dan seterusnya naik sampai ke simpul akar. Leluhur sebuah simpul dapat juga diidentifikasi dari simpul-simpul sepanjang jalur dari akar ke simpul itu.

## 9.3 Properti Pohon Biner

Sebuah pohon biner adalah sebuah pohon di mana setiap simpul dapat mempunyai anak paling banyak dua. Satu simpul anak dinamai *anak kiri* dan satunya lagi dinamai *anak kanan*. Di



Gambar 9.5: Ada berbagai bentuk pohon untuk jumlah simpul yang sama.

simi kita akan meninjau beberapa hal terkait pohon biner.

Pohon biner mempunyai beragam bentuk dan ukuran. Bentuknya bermacam-macam bergantung pada jumlah simpul dan bagaimana simpul-simpulnya dihubungkan, lihat Gambar 9.5. Terdapat bermacam properti dan karakter yang diasosiasikan dengan pohon biner, yang semuanya bergantung pada pengaturan letak simpul-simpul di dalam pohon itu.

### Ukuran Pohon

Simpul-simpul di sebuah pohon biner diorganisasi ke dalam beberapa *level* dengan simpul akar di level 0, anaknya di simpul 1, anak dari anaknya di simpul 2, dan seterusnya. Dalam istilah pohon keluarga, satu level adalah satu generasi. Pohon biner di Gambar 9.5(a), sebagai contoh, mempunyai dua simpul di level satu (B dan C), empat simpul di level dua (D, E, F, dan G), serta dua simpul di level tiga (H dan I). Simpul akar selalu menempati level nol.

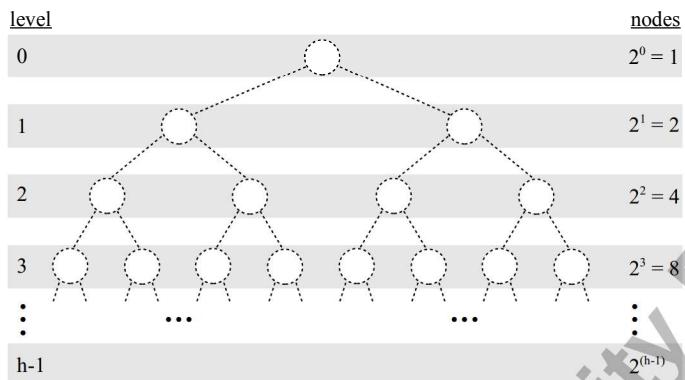
**Kedalaman (depth)** sebuah simpul adalah jarak dia dari simpul akar, di mana jarak adalah banyaknya level yang memisahkan keduanya. Kedalaman sebuah simpul berkaitan dengan level di mana dia berada. Misal simpul G di Gambar 9.5. Di pohon (a), G mempunyai kedalaman 2, di pohon (b) dia memiliki kedalaman 3, dan di (c) kedalamannya adalah 6.

**Ketinggian (height)** sebuah pohon biner adalah jumlah level di pohon itu<sup>6</sup>. Sebagai contoh, pada Gambar 9.5, pohon (a) mempunyai ketinggian 4.

**Lebar (width)** suatu pohon biner adalah jumlah simpul pada level yang mempunyai simpul terbanyak. Pada tiga pohon biner di Gambar 9.5, pohon (a) mempunyai lebar 4, pohon (b) mempunyai lebar 3, dan pohon (c) mempunyai lebar 1.

**Besar atau ukuran (size)** suatu pohon biner adalah, mudah saja, banyaknya simpul di pohon

<sup>6</sup>Jangan mengacaukan kedalaman dan ketinggian! Kedalaman mengacu pada posisi simpul di pohon biner itu, sedangkan ketinggian mengacu pada tinggi pohon biner secara keseluruhan



Gambar 9.6: Slot-slot yang mungkin untuk penempatan simpul-simpul pada sebuah pohon biner.

itu. Sebuah pohon yang kosong mempunyai ketinggian 0, lebar 0, dan besar 0.

Sebuah pohon biner dengan ukuran  $n$  dapat mempunyai ketinggian maksimum  $n$ , yang akan terjadi jika terdapat satu simpul di setiap level (lihat pohon biner pada Gambar 9.5a). Sekarang, berapakah ketinggian minimumnya? Untuk menentukan ini, kita harus mempertimbangkan jumlah simpul maksimum di tiap level, karena simpul-simpul itu semua harus ditata di tiap level dengan kapasitas penuh<sup>7</sup>. Gambar 9.6 mengilustrasikan slot-slot penempatan yang mungkin di sebuah pohon biner.

Karena tiap simpul dapat mempunyai anak paling banyak dua, setiap level berikutnya akan mempunyai simpul dua kali lebih banyak dari simpul sebelumnya. Sehingga level ke  $i$  mempunyai kapasitas sebanyak  $2^i$  simpul. Jika kita menjumlahkan ukuran setiap level, saat semua level-level ini telah penuh sesuai kapasitas (kecuali mungkin level terakhirnya), kita akan menemukan bahwa ketinggian minimum sebuah pohon biner dengan ukuran  $n$  adalah  $\lfloor \log_2 n \rfloor + 1$ . Di sini, lambang  $\lfloor \cdot \rfloor$  adalah operasi pembulatan ke bawah.

**Latihan 9.2** Hitunglah ketinggian minimum sebuah pohon biner dengan ukuran berikut dan buatlah sketsa pohon biner untuk masing-masing ukuran.

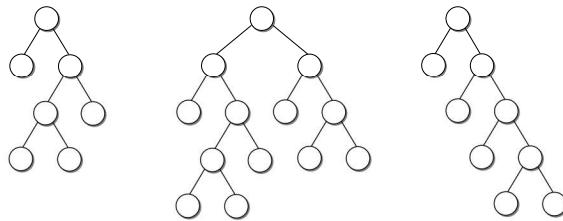
- |      |      |       |       |       |
|------|------|-------|-------|-------|
| a. 2 | c. 5 | e. 8  | g. 12 | i. 20 |
| b. 3 | d. 7 | f. 11 | h. 15 | j. 31 |

□

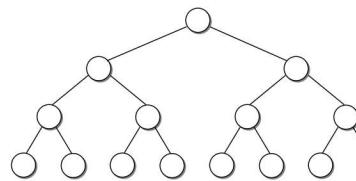
### Struktur Pohon

Ketinggian pohon berperan penting dalam menganalisis kompleksitas-waktu dari beragam algoritma yang dioperasikan ke pohon biner. Sifat struktural pohon biner dapat pula berperan dalam efisiensi suatu algoritma. Bahkan beberapa algoritma mensyaratkan struktur tertentu.

<sup>7</sup>Level 0 dipenuhi terlebih dahulu, lalu level 1 dipenuhi, lalu level 2 dipenuhi, dst.



Gambar 9.7: Beberapa contoh pohon biner penuh.



Gambar 9.8: Sebuah pohon biner sempurna.

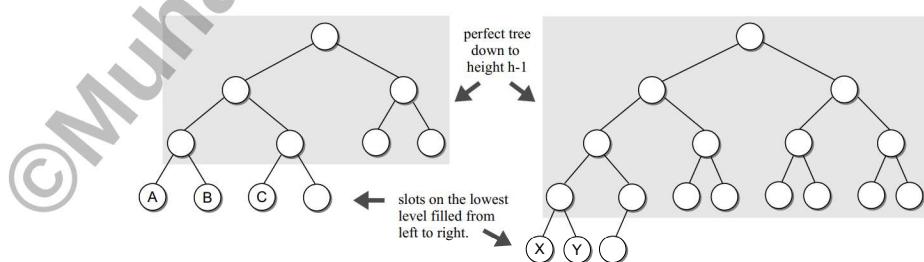
Sebuah **pohon biner penuh** (*full binary tree*) adalah sebuah pohon biner di mana setiap simpul dalam berisi dua anak. Pohon penuh bisa mempunyai bentuk yang bermacam-macam, seperti diperlihatkan di Gambar 9.7.

Sebuah **pohon biner sempurna** (*perfect binary tree*) adalah sebuah pohon biner penuh di mana semua simpul daun berada di level yang sama. Pohon sempurna ini mempunyai slot-slot yang penuh dari atas sampai bawah tanpa ada yang kosong, seperti diperlihatkan di Gambar 9.8.

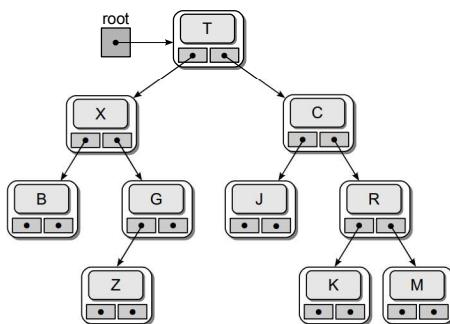
Sebuah pohon biner dengan ketinggian  $h$  disebut **pohon biner komplit** (*complete binary tree*) jika dia merupakan pohon biner sempurna sampai tinggi  $h - 1$  dan semua simpul-simpul di level terakhir mengisi slot-slot yang tersedia dari kiri ke kanan tanpa ada *gap*. Contoh pohon biner komplit ditunjukkan di Gambar 9.9.

**Latihan 9.3** Kerjakan yang berikut ini.

- Adakah kekangan terkait ukuran sebuah pohon biner jika kita ingin membuat sebuah pohon biner penuh? Bisakah 4 simpul menjadi sebuah pohon biner? Bagaimana kalau 5 simpul?



Gambar 9.9: Dua contoh pohon biner komplit.



Gambar 9.10: Implementasi sebuah pohon biner.

- b. Ukuran sebuah pohon biner sempurna adalah tertentu; tidak setiap jumlah simpul bisa disusun menjadi pohon biner sempurna. Buktikan bahwa sejumlah simpul bisa disusun menjadi sebuah pohon biner sempurna jika, dan hanya jika, banyaknya simpul itu adalah  $2^n - 1$  dengan  $n = 1, 2, 3, \dots$

□

#### 9.4 Implementasi

Pohon biner umumnya diimplementasikan dengan sebuah struktur dinamis yang mirip dengan implementasi linked list<sup>8</sup>.

Struktur pohon biasanya dilustrasikan sebagai struktur abstrak memakai simpul-simpul yang diwakili dengan lingkaran atau kotak dan pinggir yang diwakili dengan garis atau panah. Untuk mengimplementasikan sebuah pohon biner, pada tiap simpul kita harus menyimpan isi data simpul itu serta tautan ke kedua anaknya. Dengan demikian kita akan terlebih dahulu mendefinisikan kelas penyimpanan `SimpulPohonBiner` untuk membuat simpul di sebuah pohon biner dengan kode di bawah ini.

```

1 class _SimpulPohonBiner(object):
2     def __init__( self, data ):
3         self.data = data
4         self.kiri = None
5         self.kanan = None
  
```

Seperti kelas penyimpan yang lainnya, kelas simpul pohon ini hanya digunakan untuk keperluan internal struktur pohon yang lebih besar. Gambar 9.10 menggambarkan implementasi fisik pohon biner di Gambar 9.4.

<sup>8</sup>Lihat halaman 33 untuk topik ini

### Sebuah contoh membangun pohon biner

Telah kita ketahui bahwa simpul sebuah pohon biner dibangun dari sebuah class yang di dalamnya terdapat data, tautan ke anak kiri, dan tautan ke anak kanan. Untuk membangun dan mengisi sebuah pohon biner, yang kita lakukan adalah membuat simpul-simpulnya (sekaligus mengisi data-nya, kalau kita merujuk pada class di atas), lalu mentautkan satu simpul dengan simpul lain melalui hubungan ortu-anak.

Pada kode di bawah, mula-mula kita membuat simpul-simpul yang *data*-nya adalah beberapa nama kota/kabupaten di Indonesia (tiap simpul berisi *string* nama sebuah kota/kabupaten). Sesudah itu kita menghubungkan simpul-simpul yang baru saja kita buat, dengan cara mengisi *kiri* dan *kanan* pada simpul-simpul yang berperan sebagai ortu.

```
1 # Membuat simpul-simpul dan mengisi data
2 A = _SimpulPohonBiner('Ambarawa')
3 B = _SimpulPohonBiner('Bantul')
4 C = _SimpulPohonBiner('Cimahi')
5 D = _SimpulPohonBiner('Denpasar')
6 E = _SimpulPohonBiner('Enrekang')
7 F = _SimpulPohonBiner('Flores')
8 G = _SimpulPohonBiner('Garut')
9 H = _SimpulPohonBiner('Halmahera Timur')
10 I = _SimpulPohonBiner('Indramayu')
11 J = _SimpulPohonBiner('Jakarta')
12
13 # Menghubungkan simpul ortu-anak
14 A.kiri = B; A.kanan = C
15 B.kiri = D; B.kanan = E
16 C.kiri = F; C.kanan = G
17 E.kiri = H
18 G.kanan = I
```

Verifikasilah bahwa pohon di atas membentuk pohon yang ditunjukkan di Gambar 9.5a.

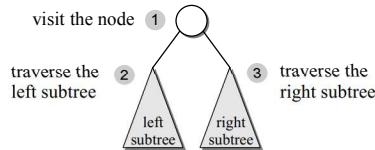
**Latihan 9.4** Kerjakan yang berikut ini.

- Dengan mengacu pada kode di atas, buatlah pohon biner yang diilustrasikan pada Gambar 9.5b dan Gambar 9.5c.
- Buatlah sebuah pohon biner sempurna dari sebagian simpul-simpul di atas. (Mengapa “sebagian”?)

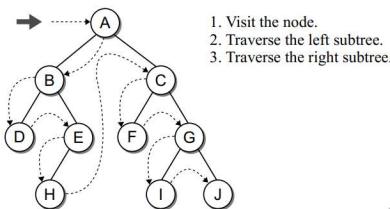
□

### 9.5 Tree Traversals

Yang dimaksud *tree traversals* di sini adalah mengakses setiap simpul-simpul di suatu pohon, satu demi satu. Operasi traversal ini adalah salah operasi yang paling sering dilakukan. Operasi yang sesungguhnya dilakukan saat “mengunjungi” tiap simpul adalah bergantung pada aplikasinya. Namun bisa jadi hanya sekedar mencetak data di tiap simpul atau menyimpan data itu ke suatu berkas.



**Gambar 9.11:** Struktur pohon akan di-traverse secara rekursif. Pertama kunjungi simpulnya. Kedua, traverse subpohon kiri. Ketiga, traverse subpohon kanan.



**Gambar 9.12:** Pengurutan simpul-simpul pada *preorder traversal*.

Kalau kita punya struktur linier seperti daftar bertautan (*linked lists*, hlm. 33), proses traversal agak lebih mudah dibayangkan. Kita tinggal memulai dari simpul (elemen) pertama dan mengunjungi satu persatu melalui tautan antar simpul, sampai simpul terakhir. Tapi bagaimana kalau kita mengunjungi setiap simpul di sebuah pohon biner? Di situ tidak ada jalur tunggal dari akar menuju simpul lain di pohon itu. Jika kita mengikuti tautannya begitu saja, maka sekali kita sampai di sebuah simpul daun, kita tidak akan bisa secara langsung mengakses simpul-simpul lain di pohon itu.

### 9.5.1 Preorder Traversal

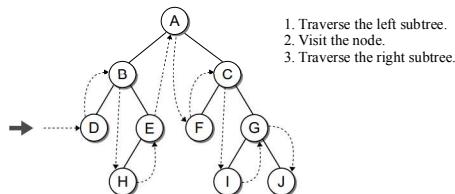
Traversal pohon haruslah dimulai dari simpul akar, karena hanya dari sitalah kita bisa mengakses pohnnya secara keseluruhan. Sesudah mengunjungi simpul akar, kita lalu mengunjungi simpul-simpul<sup>9</sup> di subpohon sebelah kiri diikuti subpohon di sebelah kanan. Karena setiap simpul adalah akar subpohon miliknya, kita dapat mengulang proses yang sama pada tiap simpul, yang menghasilkan solusi rekursif<sup>10</sup>. Kejadian dasar (*base case*) muncul saat tautan anak (kiri maupun kanan) yang ditemui mempunyai nilai `None`, karena ini berarti tidak terdapat lagi subpohon yang harus diproses lewat tautan itu. Operasi rekursif ini dapat ditampilkan secara grafis seperti yang ditunjukkan di Gambar 9.11.

Perhatikan pohon biner yang ditunjukkan di Gambar 9.12. Garis putus-putus di situ menunjukkan pengurutan logikal simpul-simpul saat dikunjungi: A, B, D, E, H, C, F, G, I, J.

Fungsi rekursif untuk *preorder traversal* sebuah pohon biner adalah cukup sederhana, seperti ditunjukkan pada kode di bawah. Argumen `subpohon` hanya mempunyai dua kemungkinan nilai: yaitu `None`, atau kalau bukan itu, berupa tautan ke akar subpohon berikutnya.

<sup>9</sup>Sekali lagi, yang dimaksud “mengunjungi simpul” adalah melakukan operasi yang diperlukan (hal ini bergantung aplikasinya) di simpul itu. Pada contoh-contoh di sini, operasinya adalah menge-print data di simpul itu. Aplikasi lain mungkin akan menyimpan file, mengirim email, menggambar garis, dsb.

<sup>10</sup>Lihat Kotak di halaman 57 untuk paparan sekilas fungsi rekursif.

Gambar 9.13: Pengurutan simpul-simpul pada *inorder traversal*.

```

1 # preorder traversal
2 def preorderTrav( subpohon ):
3     if subpohon is not None :
4         print( subpohon.data )
5         preorderTrav( subpohon.kiri )
6         preorderTrav( subpohon.kanan )

```

Jika isinya bukan `None`, simpul itu dikunjungi terlebih dahulu dan lalu dua subpohnnya dikunjungi. Dari kesepakatan, kita tentukan bahwa yang dikunjungi dulu adalah subpohn kiri sebelum yang kanan. Argumen `subpohn` akan mempunyai nilai `None` saat pohon biner si subpohn itu kosong atau program kita mencoba mengikuti tautan kosong salah satu atau kedua anak simpul.

Jika diberi sebuah pohon biner dengan ukuran  $n$ , sebuah traversal komplet akan mengunjungi setiap simpul masing-masing satu kali.

### 9.5.2 Inorder Traversal

Pada preorder tranversal di atas kita memilih untuk mengunjungi simpulnya dulu baru men-traverse kedua subpohnnya. Cara lain untuk men-traverse sebuah pohon biner adalah dengan *inorder traversal* di mana kita pertama-tama men-traverse subpohn sebelah kiri, lalu mengunjungi simpulnya<sup>11</sup>, dilanjutkan men-traverse subpohn sebelah kanan. Gambar 9.13 menunjukkan urutan kunjungan ke simpul-simpul pada pohon biner yang menjadi contoh: D, B, H, E, A, F, C, I, G, J.

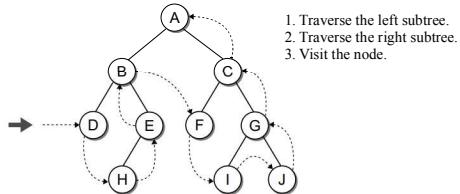
Fungsi rekursif untuk *inorder traversal* sebuah pohon biner tertuang pada kode di bawah. Kodenya hampir sama dengan kode untuk preorder traversal. Perbedaannya adalah operasi kunjungannya berpindah ke tengah, menjadi sesudah traversal subpohn kiri.

```

1 # inorder traversal
2 def inorderTrav( subpohon ):
3     if subpohon is not None :
4         inorderTrav( subpohon.kiri )
5         print( subpohon.data )
6         inorderTrav( subpohon.kanan )

```

<sup>11</sup>lihat maksud “mengunjungi simpul” di Catatan Kaki 9 di hlm. 94.



**Gambar 9.14:** Pengurutan simpul-simpul pada *postorder traversal*.

### 9.5.3 Postorder Traversal

Kita juga bisa melakukan *postorder traversal*, di mana subpohon kiri dan kanan di tiap simpul di-traverse dulu sebelum simpulnya dikunjungi<sup>12</sup>. Fungsi rekursifnya diperlihatkan di bawah.

```
1 # postorder traversal
2 def postorderTrav( subpohon ):
3     if subpohon is not None :
4         postorderTrav( subpohon.kiri )
5         postorderTrav( subpohon.kanan )
6         print( subpohon.data )
```

Pohon contoh dengan urutan pengunjungan simpul memakai *postorder traversal* diperlihatkan di Gambar 9.14. Simpul-simpulnya dikunjungi dengan urutan D, H, E, B, F, I, J, G, C, A. Perhatikan bahwa pada preorder traversal simpul akarnya selalu dikunjungi pertama sedangkan pada postorder traversal simpul akarnya dikunjungi terakhir.

**Latihan 9.5** Perhatikan kembali kode pada halaman 93 (contoh membangun sebuah pohon biner). Cetaklah semua data di simpul-simpulnya dengan cara *preorder traversal*, *inorder traversal*, dan *postorder traversal*. Verifikasilah bahwa yang tercetak adalah sesuai dengan tiga gambar keterangan yang sudah disampaikan. □

## 9.6 Soal-soal untuk Mahasiswa

Sebelum mengerjakan soal-soal di bawah, kerjakan dulu latihan-latihan di atas.

1. Diberikan pohon biner dengan ukuran  $n$ , berapakah jumlah level minimum yang bisa dimuatnya? Berapakah jumlah level maksimumnya? Tentukan untuk nilai  $n$  berikut.  

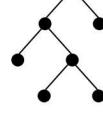
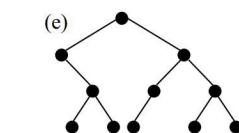
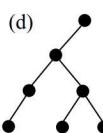
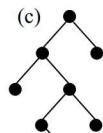
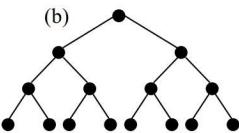
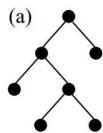
(a)  $n = 10$                       (b)  $n = 35$                       (c)  $n = 76$                       (d)  $n = 345$
  2. Gambarlah semua bentuk pohon biner berukuran 5 yang mungkin. Ada berapa kemungkinan?
  3. Berapakah jumlah simpul maksimum suatu pohon biner dengan jumlah level  $h$ ? Tentukan untuk nilai  $h$  berikut.

---

<sup>12</sup>lihat maksud “mengunjungi simpul” di Catatan Kaki 9 di hlm. 94.

(a)  $h = 3$ (b)  $h = 4$ (c)  $h = 5$ (d)  $h = 6$ 

4. Diberikan pohon-pohon biner seperti di bawah.



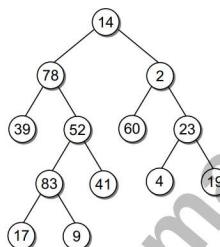
(a) Tunjukkan semua properti struktural yang berlaku pada tiap-tiap pohon di atas: *penuh*, *sempurna*, *komplet*. Ingat bahwa sebuah pohon biner bisa saja bersifat penuh sekaligus sempurna, dan sebagainya.

(b) Tentukan ukuran tiap pohon.

(c) Tentukan ketinggian tiap pohon.

(d) Tentukan lebar tiap pohon.

5. Perhatikan pohon biner berikut.



(a) Tunjukkan urutan pengunjungan simpul untuk

- i. preorder traversal      ii. inorder traversal      iii. postorder traversal

(b) Simpul mana saja yang merupakan simpul daun?

(c) Simpul mana saja yang merupakan simpul dalam?

(d) Simpul mana saja yang berada di level 4?

(e) Tulis semua simpul yang berada di dalam jalur dari simpul akar menuju simpul

- i. 83      ii. 39      iii. 4      iv. 9

(f) Perhatikan simpul 52. Tentukan

- i. keturunannya                    ii. leluhurnya                    iii. saudaranya

(g) Tentukan kedalaman dari tiap-tiap simpul ini:

- i. 78                                ii. 41                                iii. 60                                iv. 19

### *Soal-soal pemrograman*

6. Buatlah fungsi `ukuranPohon(akar)` yang akan mendapatkan ukuran sebuah pohon biner.
7. Buatlah sebuah fungsi `tinggiPohon(akar)` yang akan mendapatkan ketinggian sebuah pohon biner.
8. Buatlah sebuah fungsi yang mencetak data tiap simpul sekaligus level di mana simpul itu berada. Silakan memilih akan memakai *preorder traversal*, *inorder traversal*, atau *postorder traversal*. Contoh sepotong hasilnya adalah seperti di bawah ini (jika kamu memakai *preorder traversal*).

```
>>> cetakDataDanLevel(A)
Ambarawa, level 0
Bantul, level 1
Denpasar, level 2
Enrekang, level 2
Halmahera Timur, level 3
Cimahi, level 1
```