STRUKTUR DATA
14 Pohon Biner

Semester Gasal 2025/2026

S1 Informatika FSM UNDIP

Level Abstraksi

1. Definisi Fungsional
   nama tipe bentukan, operasi fungsional primitif
2. Representasi Logik
   struktur tipe bentukan, spesifikasi fungsi/prosedur
3. Representasi/implementasi Fisik

   a) representasi kontigu, struktur bersifat statis

   b) representasi berkait, struktur bersifat dinamis

Pohon Konsep

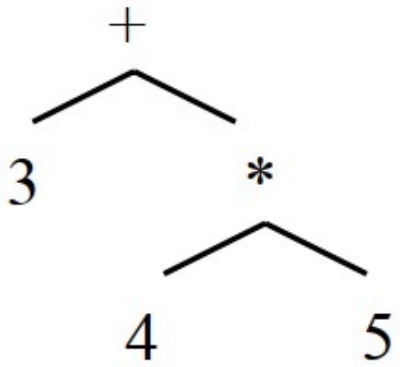
1) ADT Atomik/tunggal
2) ADT Majemuk/jamak/kolektif
   a) Representasi Kontigu ~> indexed array
   b) Representasi Berkait ~> linked list
      i. Linear: single, double
      ii. Circular: single, double
      iii. Tree: **binary**, N-ary
      iv. Graph: directed, indirected

Konsep Pohon Biner

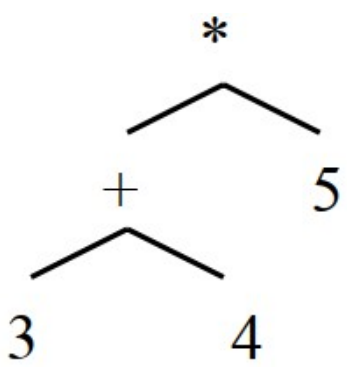
Definisi rekursif pohon biner = himpunan terbatas yang:
- **mungkin kosong**, atau
- terdiri atas sebuah simpul yang disebut **akar** dan dua buah himpunan lain yang disjoint yang juga merupakan pohon biner, yang disebut sebagai sub pohon **kiri** dan sub pohon **kanan** dari pohon biner tersebut.

Perhatikanlah perbedaan pohon biner dengan pohon biasa : pohon biner mungkin kosong, sedangkan pohon n-aire tidak mungkin kosong.

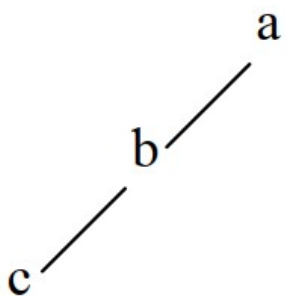
Ilustrasi Pohon Biner



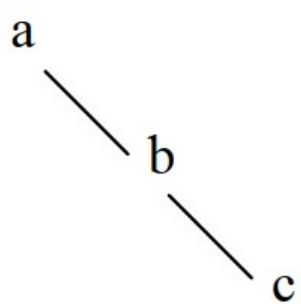
3 + (4 * 5)



(3 +4) * 5



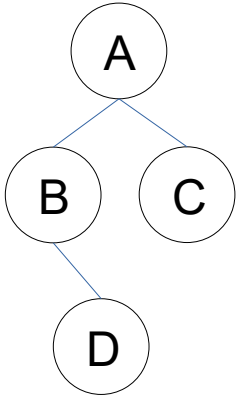
Pohon condong kiri



Pohon condong kanan

Konsensus Pohon Biner

- Node kosong NIL digambarkan dengan ( )
- Penulisan elemen secara linier prefix misalnya:
  lengkap: Pb = A(B(( ),D(( ),( ))),C(( ),( )))
  ringkas: Pb = A(B((),D)),C)



Definisi Fungsional Pohon Biner

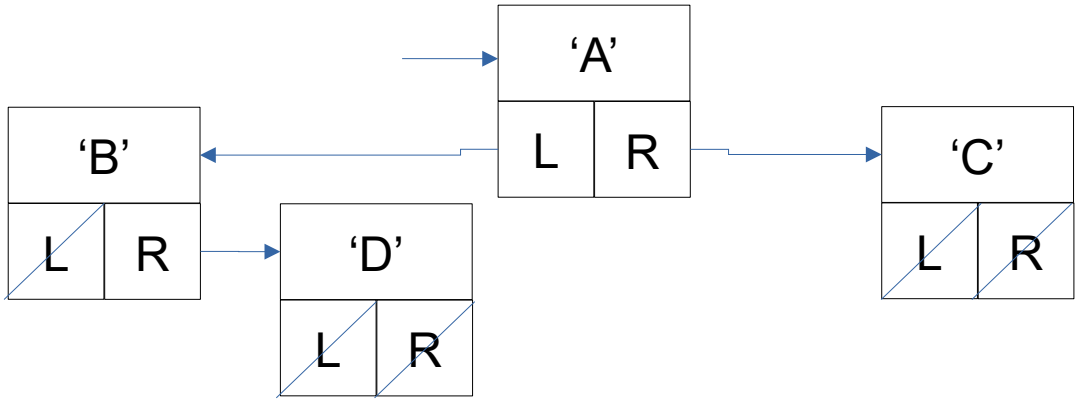
- Diberikan B, L dan R adalah pohon biner
- **IsEmpty** : B → boolean { Tes apakah pohon biner kosong }
- **MakeTree** : e,L,R → B { Membentuk sebuah pohon biner }
- **AddDaunTerkiri** : e x B → B { Menyisipkan sebuah elemen sebagai daun terkiri}
- **DelDaunTerkiri** : B → B x e { Menghapus daun terkiri}
- **UpdateInfo** : e x B → B { Mengubah informasi elemen pohon }

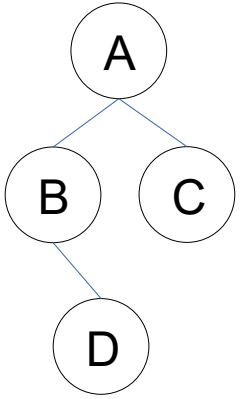
Definisi Fungsional Pohon Biner

- Operator **Holistik**: terkait objek secara utuh, misalnya penciptaan, pemeriksaan kekosongan, pemusnahan, penyambungan, pemecahan.
- Operator **Elementer**: terkait elemen, misalnya penambahan elemen, pengurangan elemen, update informasi elemen, pencarian elemen, penjelajahan.

Definisi Logik Pohon Biner

- type infotype = character
- type node = < Info : infotype,
- Left : BinTree,
  Right : BinTree >
- type BinTree = pointer to node
- {BinTree kosong bila nilainya NIL}





Konstruktor Pohon Biner

- function **Tree**(Akar:infotype, L:BinTree, R:BinTree)-> BinTree
  { Menghasilkan sebuah pohon biner dari A, L, dan R, jika alokasi berhasil, atau pohon kosong (Nil) jika alokasi gagal }
- procedure **MakeTree**(input Akar:infotype, input L:BinTree, input R:BinTree, output P:BinTree)
  { I.S. Sembarang }
  { F.S. Menghasilkan sebuah pohon P }
  { Menghasilkan sebuah pohon biner P dari A, L, dan R, jika alokasi berhasil, atau pohon kosong (Nil) jika alokasi gagal }

Selektor Pohon Biner

- `function` **`GetAkar`** `(P : BinTree) -> infotype`
  `{ Mengirimkan nilai Akar pohon biner P }`
- `function` **`GetLeft`** `(P : BinTree) -> BinTree`
  `{ Mengirimkan Anak Kiri pohon biner P }`
- `function` **`GetRight`** `(P : BinTree) -> BinTree`
  `{ Mengirimkan Anak Kanan pohon biner P }`

Predikat Pohon Biner

- function **IsEmptyTree** (P:BinTree)-> boolean
  { Mengirimkan true jika pohon biner kosong}
- function **IsDaun** (P:BinTree)-> boolean
  { Mengirimkan true jika pohon biner tidak kosong, namun anak kiri dan anak kanan kosong}
- function **IsBiner** (P:BinTree)-> boolean
  { Mengirimkan true jika pohon biner tidak kosong P adalah pohon biner: mempunyai subpohon kiri dan subpohon kanan}

Predikat Pohon Biner

- function **IsUnerLeft**(P:BinTree)-> boolean
  { Mengirimkan true jika pohon biner tidak kosong P adalah pohon unerleft: hanya mempunyai subpohon kiri }
- function **IsUnerRight**(P:BinTree)-> boolean
  { Mengirimkan true jika pohon biner tidak kosong P adalah pohon unerright: hanya mempunyai subpohon kanan}

Penjelajahan Pohon Biner

```
procedure PrintTree (input P : BinTree,
input h : integer)
{ I.S. P terdefinisi, h adalah jarak
indentasi }
{ F.S. Semua simpul P sudah ditulis dengan
indentasi }
```

- `procedure` **`Preorder`** `(input P : BinTree)`
  `{ I.S. P terdefinisi }`
  `{ F.S. Semua simpul P sudah diproses secara Preorder: akar, kiri, kanan dengan Proses (P) }`

Penjelajahan Pohon Biner

- procedure **Inorder** (input P : BinTree)
- { I.S. P terdefinisi }
- { F.S. Semua simpul P sudah diproses secara Inorder: kiri, akar, kanan dengan Proses(P) }
- procedure **Postorder** (input P : BinTree)
- { I.S. P terdefinisi }
- { F.S. Semua simpul P sudah diproses secara Postorder: kiri, kanan, akar dengan Proses(P) }

Penjelajahan Pohon Biner

- `function` **`NbElm`**`(P:BinTree)-> integer`
  `{ Mengirimkan banyaknya node dari P}`
- `function` **`NbDaun`**`(P:BinTree)-> integer`
  `{ Mengirimkan banyaknya daun (node) pohon biner P }`

Pencarian Pohon Biner

- function **SearchX**(P:BinTree, X:infotype) -> boolean
  { Mengirimkan true jika ada node dari P yang bernilai X }
- function **CountX**(P:BinTree, X:infotype) -> integer
  { Mengirimkan banyaknya node dari P yang bernilai X }

Operator Lain di Pohon Biner

- function **IsSkewLeft** (P : BinTree)-> boolean
  { Mengirim true jika P adalah pohon condong kiri }
- function **IsSkewRight** (P : BinTree) -> boolean
  { Mengirim true jika P adalah pohon condong kanan }
- function **Level**(P:BinTree, X:infotype)-> integer
  { Mengirimkan level dari node X yang merupakan salah satu simpul dari pohon biner P. Akar(P) level-nya adalah 1. Pohon P tidak kosong. }

Operator Lain di Pohon Biner

- procedure **AddDaun** (input/Output P : BinTree, input X, Y : infotype, input Kiri : boolean)
  { I.S. P tidak kosong, X adalah salah satu daun Pohon Biner P }
  { F.S. P bertambah simpulnya, dengan Y sebagai anak kiri X (jika Kiri), atau sebagai anak Kanan X (jika not Kiri) }
- procedure **DelDaun** (input/output P : BinTree, input X : infotype)
  { I.S. P tidak kosong, X adalah salah satu daun }
  { F.S. X dihapus dari P }

```
procedure AddDaunTerkiri (input/output P :
BinTree, input X : infotype)
{ I.S. P boleh kosong }
{ F.S. P bertambah simpulnya, dengan X sebagai
simpul daun terkiri }
```

Pustaka

1. Inggriani Liem. Diktat Struktur Data. ITB. 2008
2. Debdutta Pal, Suman Halder. Data Structures and Algorithms with C. Alpha Science International Ltd. 2018.
3. AHO, Alfred V., John E. Hopcroft, Jeffrey D. Ullman. Data Structures and Algorithm. Addison Weshley Publishing Compani.1987