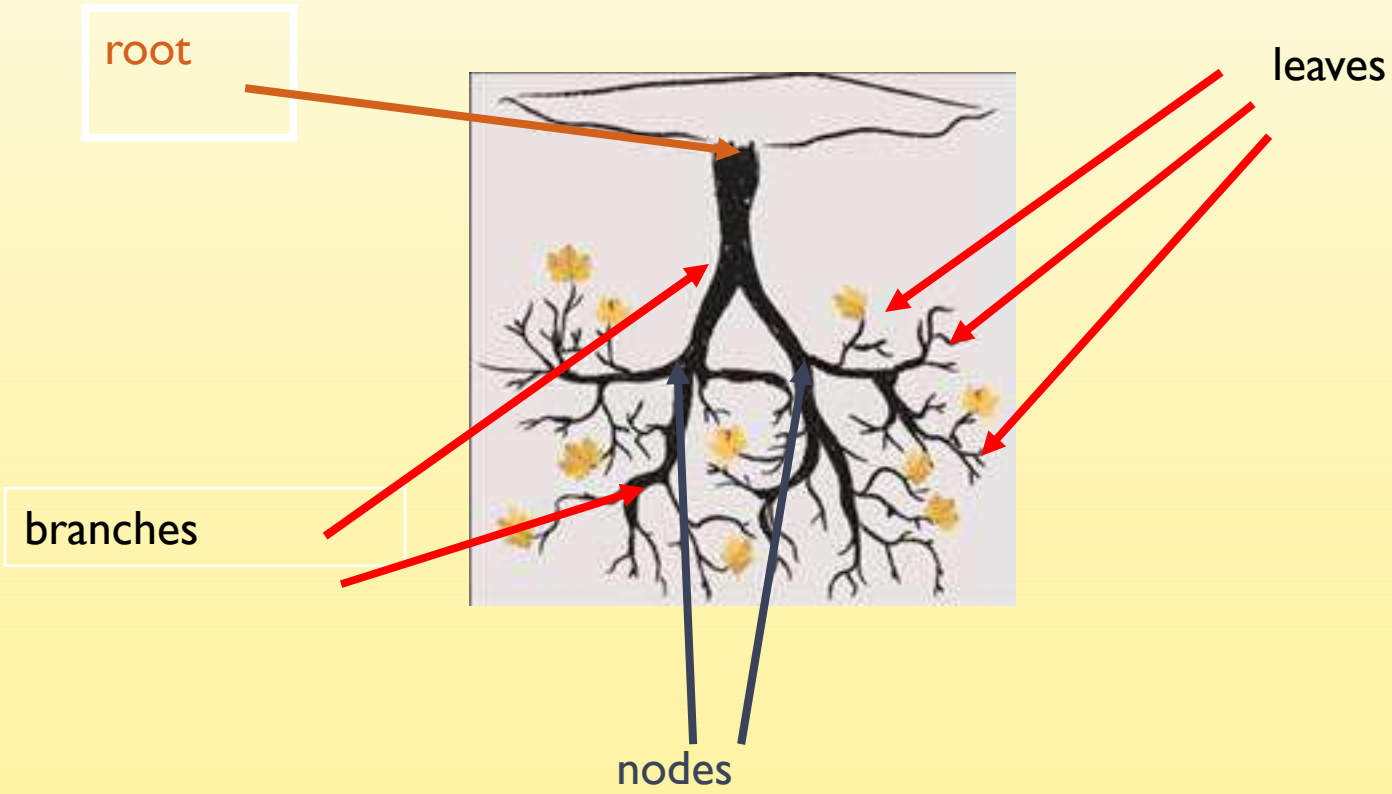
Pohon

Real World

Computer Scientist's View



Penerapan Struktur Pohon

- Mengorganisasikan informasi berdasarkan struktur lojik
- Cara mengakses terhadap suatu elemen

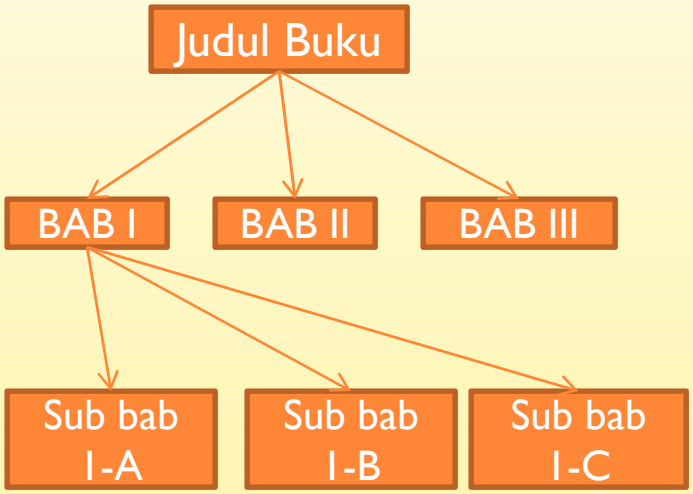
Contoh Persoalan yang di representasi sebagai pohon

- Pohon Keputusan
- Pohon Keluarga dan klasifikasi dalam botani
- Pohon sintak dan pohon ekspresi aritmatika
- Pohon dekomposisi bab dari sebuah buku
- Pohon menu dari suatu aplikasi komputer

Definisi rekurens dari pohon

- Sebuah pohon adalah himpunan terbatas tidak kosong, dengan elemen yang dibedakan sbb:
  - Sebuah elemen dibedakan dari yang lain, yang disebut sebagai **akar** dari pohon
  - Elemen yang lain (jika ada) dibagi menjadi beberapa sub himpunan yang disjoint, dan masing-masing sub himpunan tersebut adalah **POHON** yang disebut sebagai **Sub Pohon** dari pohon yang dimaksud

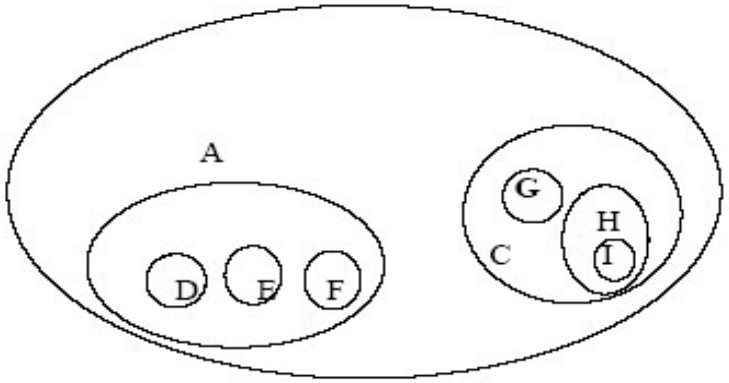
Contoh:



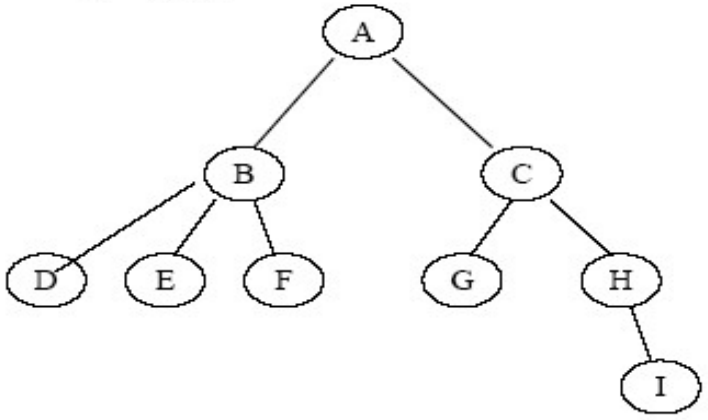
- Sebuah **buku** dipandang sebagai **pohon** dan **Judul buku** adalah **akar**
- Buku-buku dibagi menjadi bab-bab, dan masing-masing bab adalah sub pohon yang juga mengandung **judul** sebagai **akar** dari bab tersebut.
- **Bab** dibagi menjadi sub bab yang juga diberi **judul**. **Sub bab** adalah pohon dengan **judul sub bab** sebagai **akar**

Representasi pohon

a) Himpunan yang saling melingkupi



b) Graph



c) Indentasi

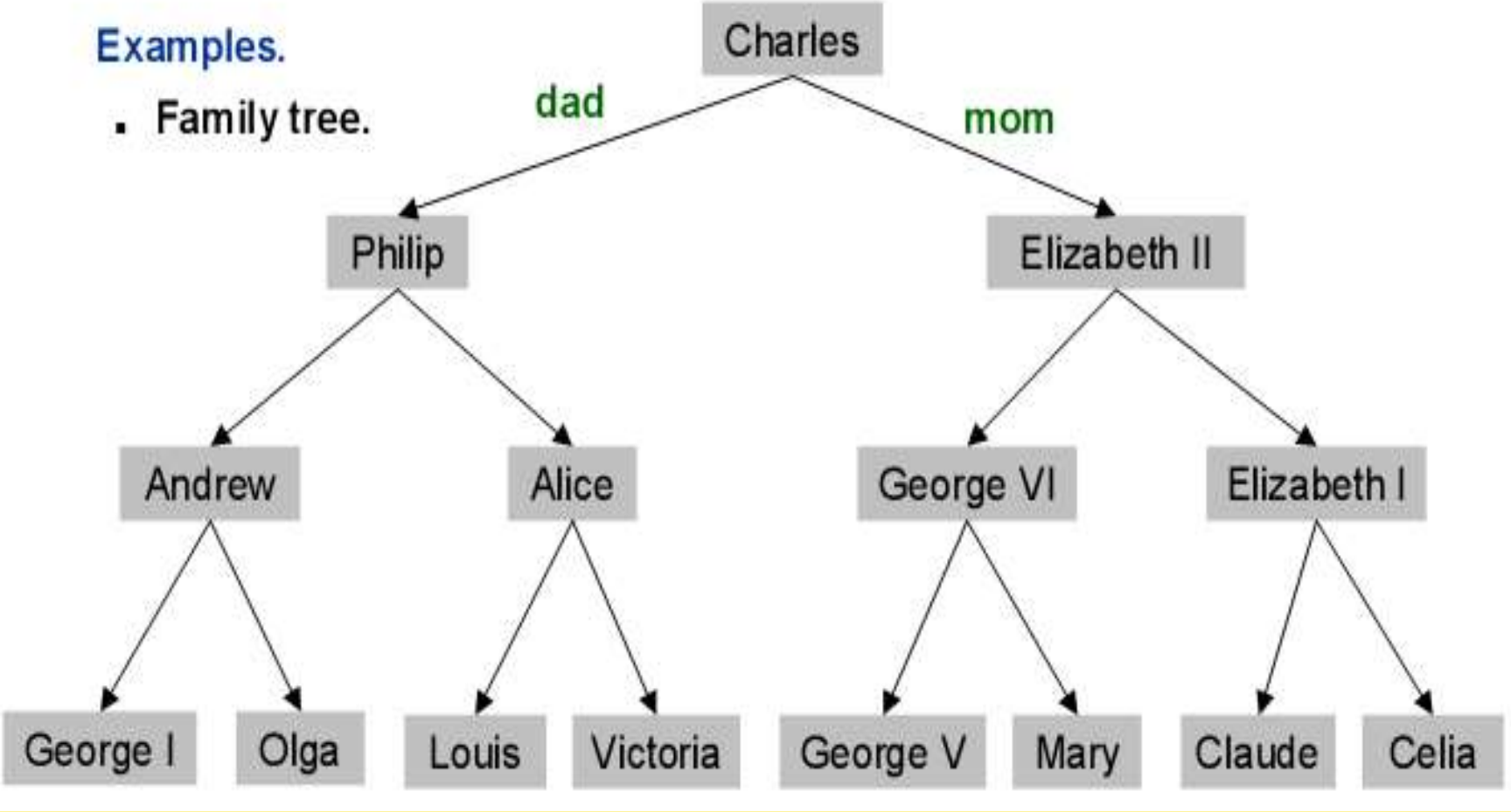
A
B
D
E
F
C
G
H
I

d) Bentuk linier :

Prefix : (A (B(D(),E(),F()), C(G(),H(I()))), atau
(A(B(D)(E)(F))(C(G)(H(I))))

Posfix : ((D,E,F)B,(G,(I) H) C))

Tree Example

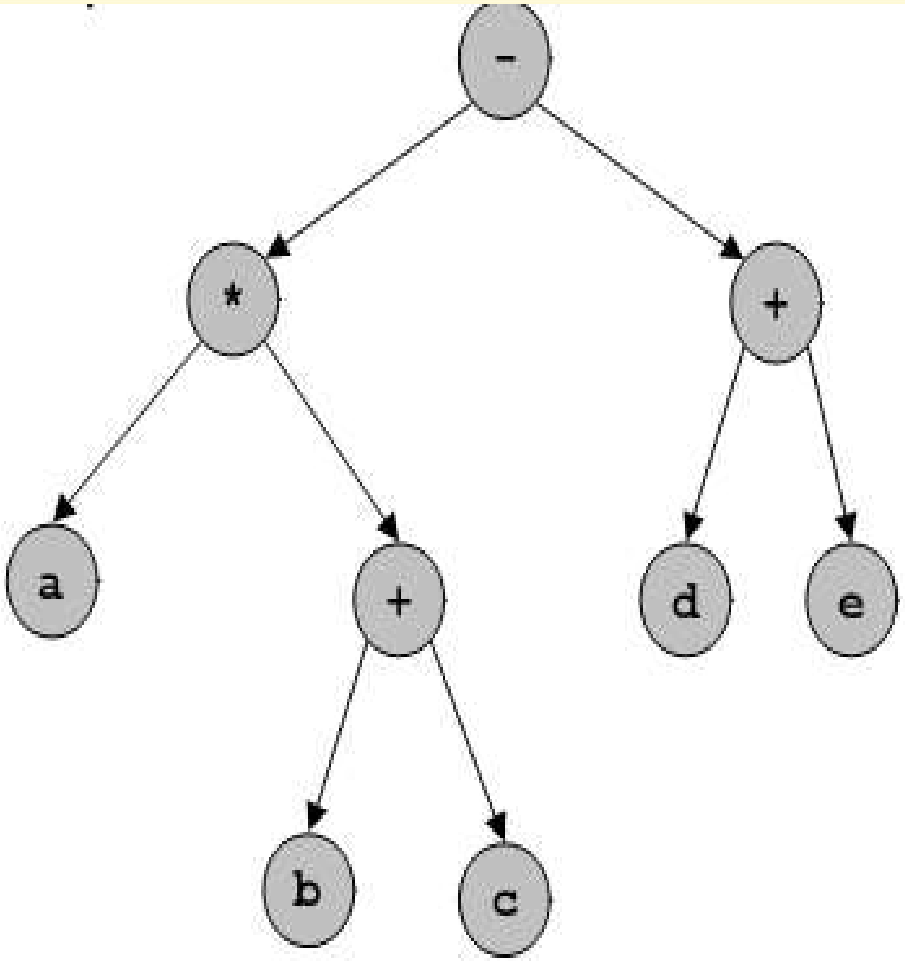


Tree Example

Examples.

- Family tree.
- Parse tree.

  (a * (b + c)) - (d + e)



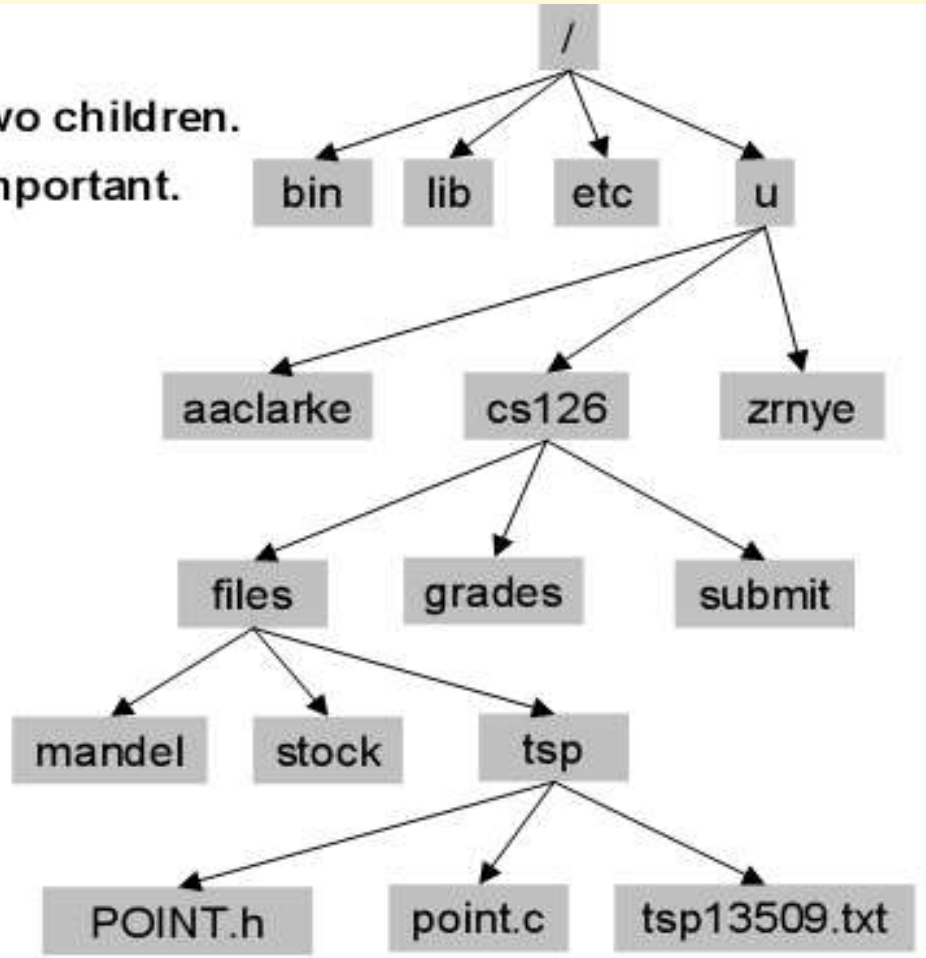
Tree Example

Trees.

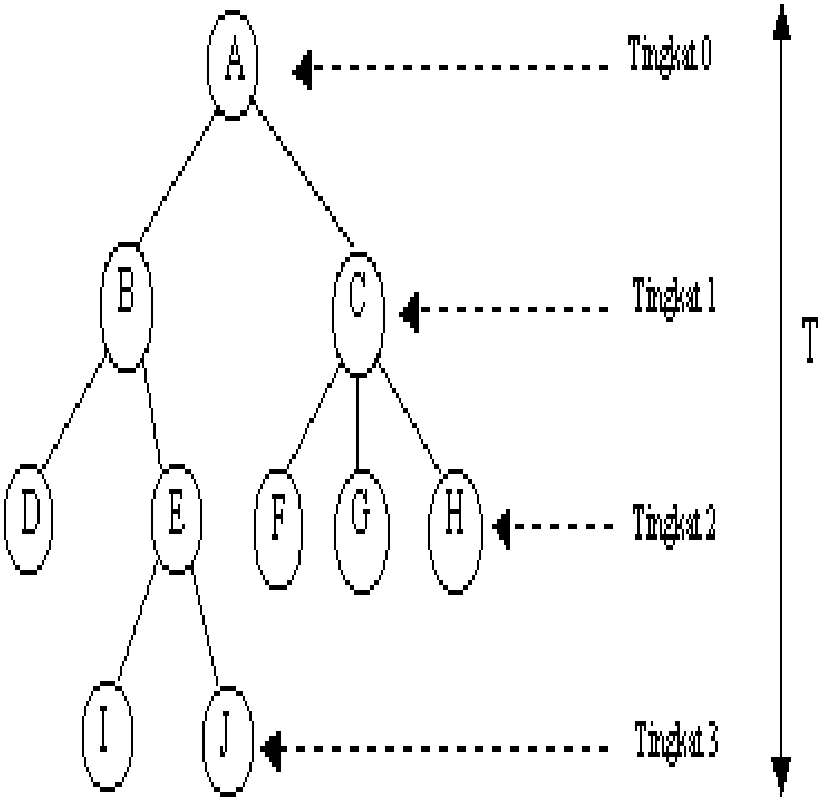
- Nodes need not have exactly two children.
- Order of children may not be important.

Examples.

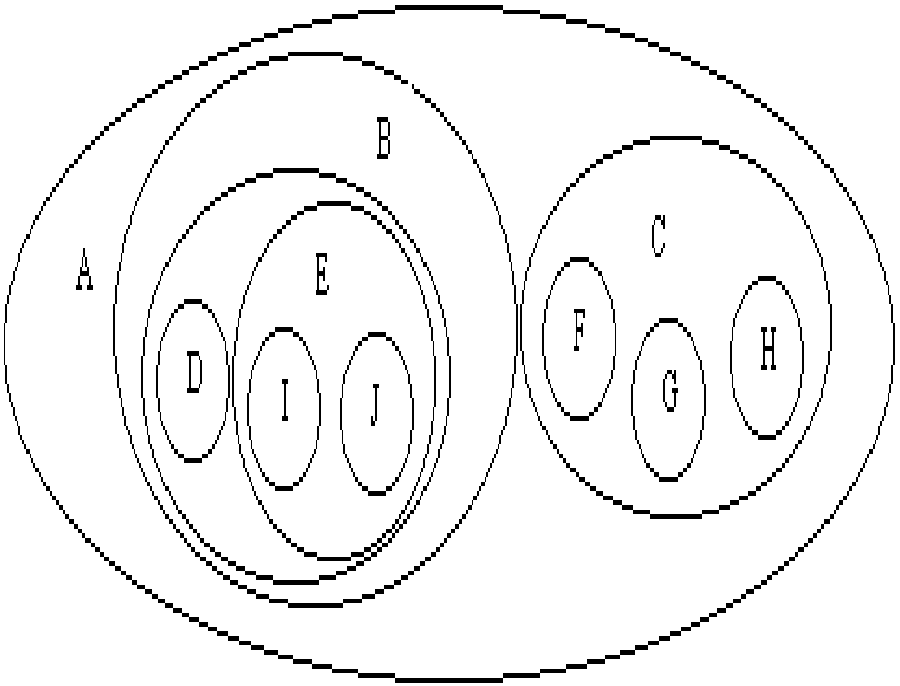
- Family tree.
- Parse tree.
- Unix file hierarchy.
  - not binary



Representasi Tree



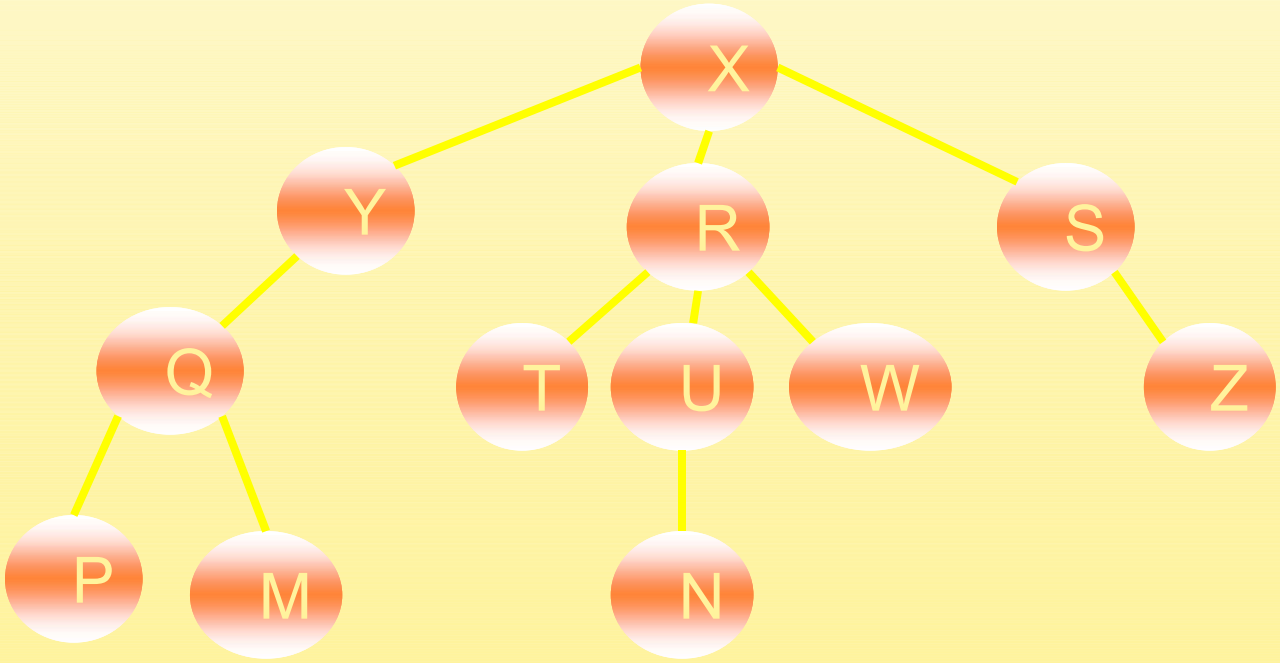
Gambar 6.1. Contoh sebuah Pohon beserta Tingkatnya



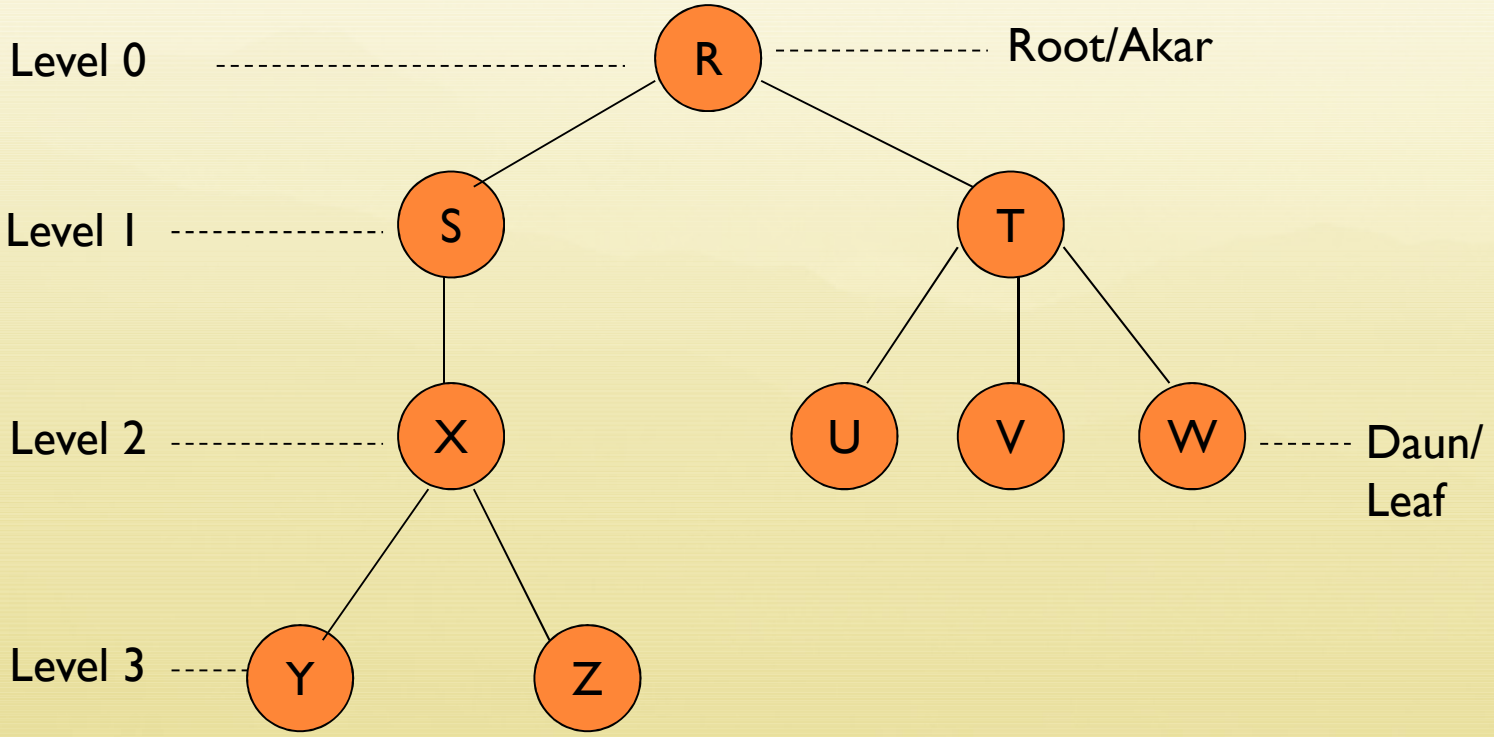
Gambar 6.2. Diagram Venn

Latihan

- Buat diagram venn dari tree di bawah ini :



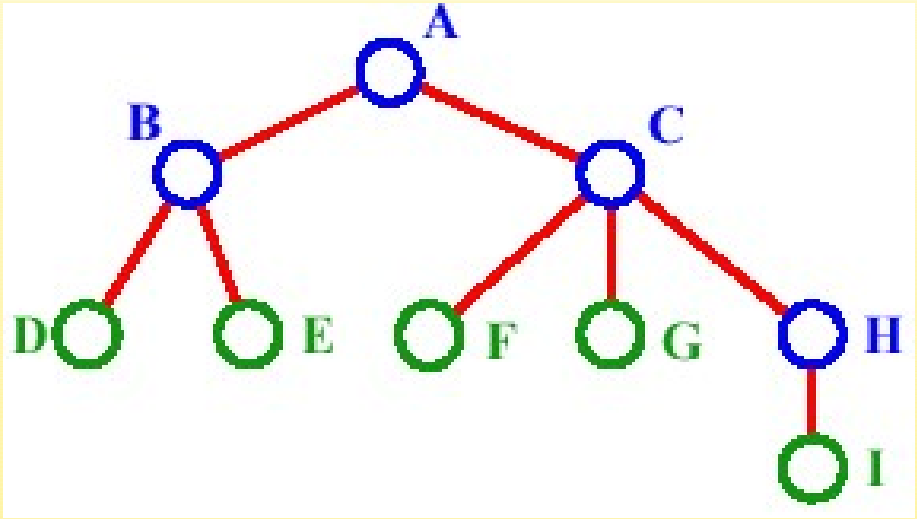
Contoh Tree (Pohon)



Istilah Tree (Pohon)

Predecesor	Node yang berada diatas node tertentu.
Successor	Node yang berada dibawah node tertentu.
Ancestor	Seluruh node yang terletak sebelum node tertentu dan terletak pada jalur yang sama
Descendant	Seluruh node yang terletak setelah node tertentu dan terletak pada jalur yang sama
Parent	Predecessor satu level di atas suatu node.
Child	Successor satu level di bawah suatu node.
Sibling	Node-node yang memiliki parent yang sama
Subtree	Suatu node beserta descendantnya.
Size	Banyaknya node dalam suatu tree
Height	Banyaknya tingkatan dalam suatu tree
Root	Node khusus yang tidak memiliki predecessor.
Leaf	Node-node dalam tree yang tidak memiliki successor.
Degree	Banyaknya child dalam suatu node

Latihan



Ancestor (F)?

Descendant (B)?

Parent (I)?

Child (C)?

Sibling (G)?

Size?

Height?

Root?

Leaf?

Degree (C)?

Tree (Pohon)

- Dimana,

  Ancestor (F) = C,A

  Descendant (B) = D,E

  Parent (I) = H

  Child (A) = B,C

  Sibling (F) = G,H

  Size = 9

  Height = 3/4

  Root = A

  Leaf = D,E,F,G,I

  Degree (C) = 3

Pohon N-aire

- Pohon N-aire adalah pohon yang setiap level anaknya boleh berbeda-beda jumlahnya, dan anaknya tersebut adalah pohon N-aire
- Definisi rekursif
  - Basis-1 : pohon yang hanya terdiri dari akar adalah pohon N-aire
  - Rekurens : sebuah pohon N-aire terdiri dari akar dan sisanya (anak-anaknya) adalah list pohon N-aire

Pohon N-aire

- Definisi dan Spesifikasi Type
  - Type elemen : {tergantung type node}
  - Type PohonN-ner : <A:Elemen, PN:PohonN-ner> {notasi prefix}
  - Type PohonN-ner : <PN:PohonN-ner, A:Elemen, > {notasi postfix}

  {Pohon N-ner terdiri dari akar yang berupa elemen dan list dari pohon N-aire yang menjadi anaknya}

Pohon N-aire

- Definisi dan Spesifikasi Selektor
  - Akar: PohonN-ner tidak kosong → Elemen

  {Akar (P) adalah akar dari P. Jika P adalah (A,PN)=Akar(P) adalah A}

  - Anak: PohonN-ner tidak kosong → list of PohonN-ner

  {Anak(P) adalah list of pohon N-ner yang merupakan anak-anak (sub pohon) dari P. Jika P adalah (A,PN)=Anak(P) adalah PN

Definisi dan spesifikasi pohon N-aire

DEFINISI DAN SPESIFIKASI KONSTRUKTOR

{ Perhatikanlah bahwa konstruktor pohon N-ner dengan basis pohon kosong dituliskan sebagai

a.Prefix : (A,P,N)

b. Posfix : (PN,A) }

DEFINISI DAN SPESIFIKASI PREDIKAT

IsTreeNEmpty : PohonN-ner → boolean

{IsTreeNEmpty(PN) true jika PN kosong : () }

IsOneElmt : PohonN-ner → boolean

{IsOneElmt(PN) true jika PN hanya terdiri dari Akar }

Definisi dan spesifikasi predikat lain

DEFINISI DAN SPESIFIKASI PREDIKAT LAIN

NbNElmt : PohonN-ner → integer ≥ 0

{NbNElmt(P) memberikan banyaknya node dari pohon P :
Basis 1: NbNElmt ((A)\) = 1
Rekurens : NbNElmt ((A,PN)) = 1 + NbELmt(PN) }

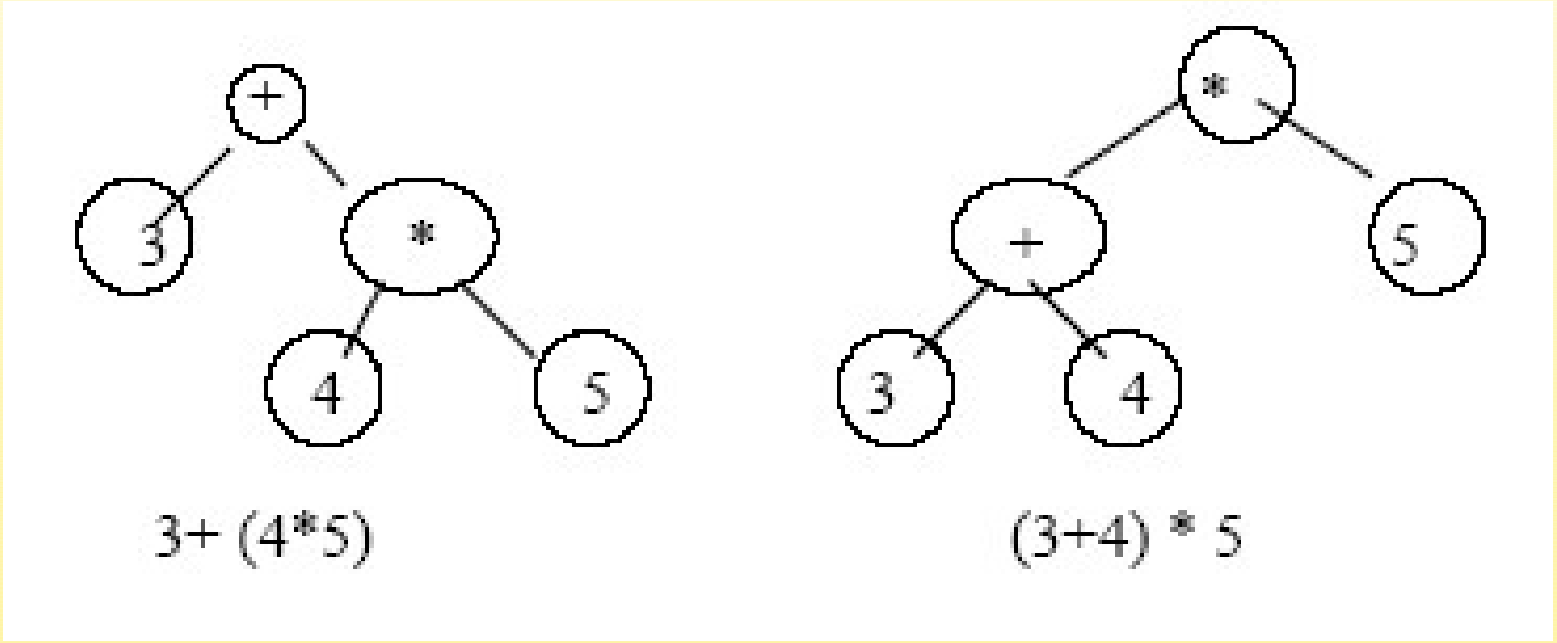
NbNDaun : PohonN-ner → integer ≥ 0

{NbNDaun (P) memberikan banyaknya daun dari pohon P :
Basis 1: NbNDaun (A) = 1
Rekurens : NbDaun ((A,PN)) = NbNDaun(PN)

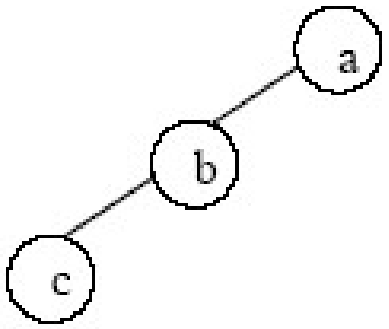
Pohon Biner

- Sebuah pohon biner adalah himpunan terbatas yang
  - Mungkin kosong
  - Terdiri dari sebuah simpul yang disebut akar dan dua buah himpunan lain yang disjoint yang merupakan pohon biner, yang disebut sebagai sub pohon kiri dan sub pohon kanan dari pohon biner tersebut

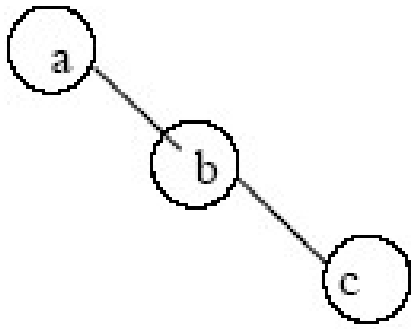
Contoh pohon ekspresi aritmatika



Pohon condong

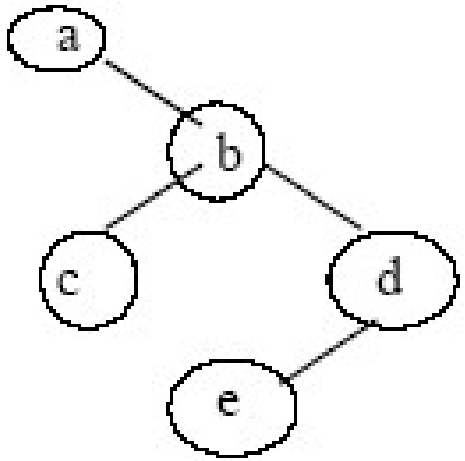


Pohon biner condong kiri



Pohon biner condong kanan

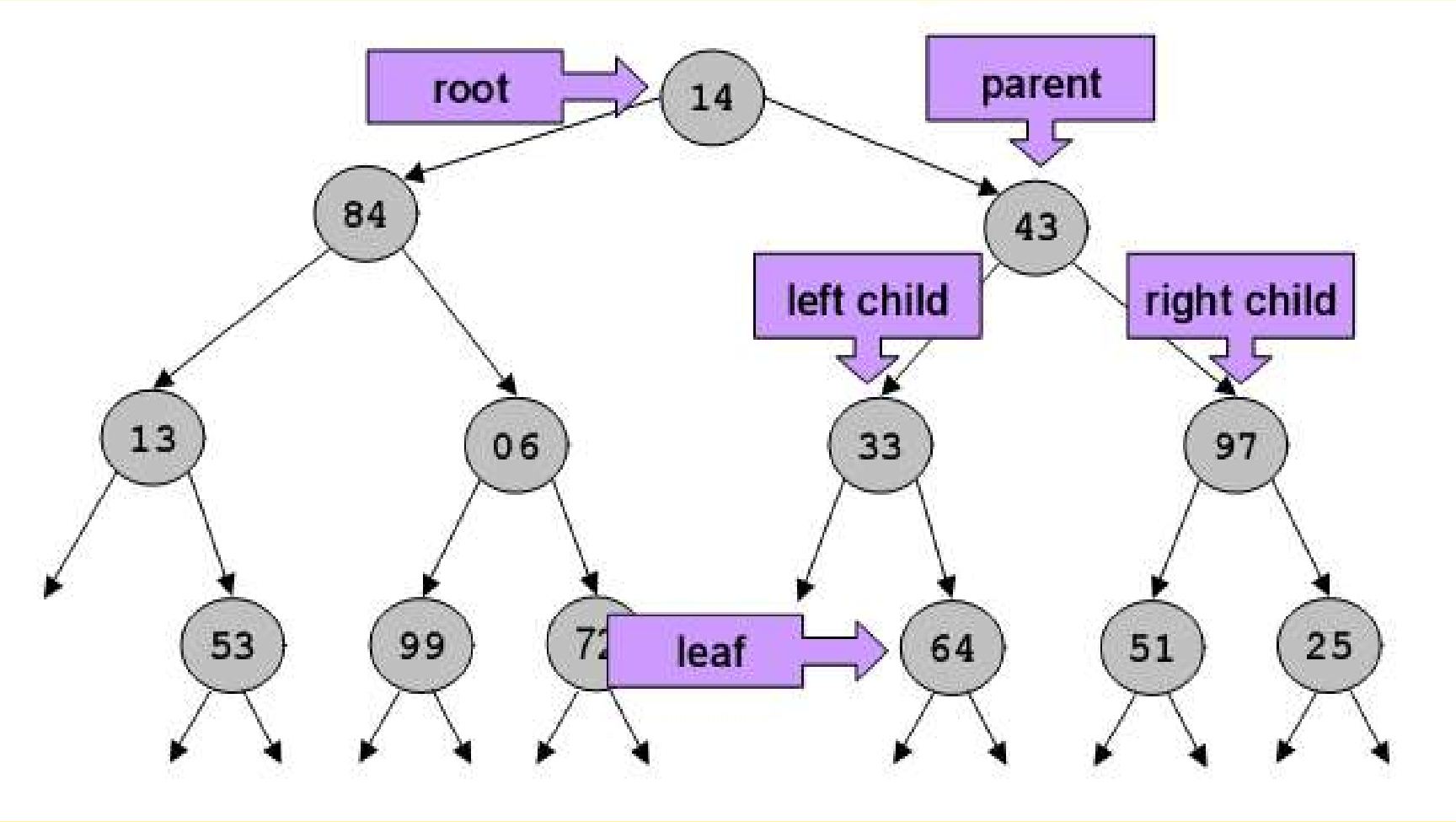
Penulisan sub pohon



Notasi prefix :

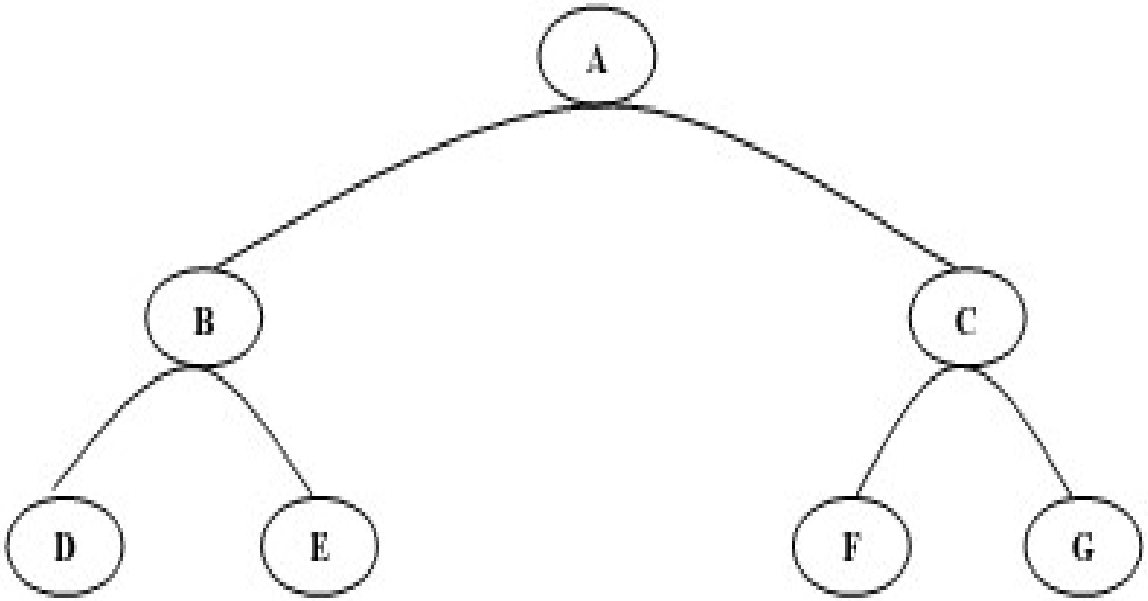
(a ( ), (b (c () ())(d (e) ()))), atau
(a() (b (c) (d (e) ())))

Binary Tree (2)



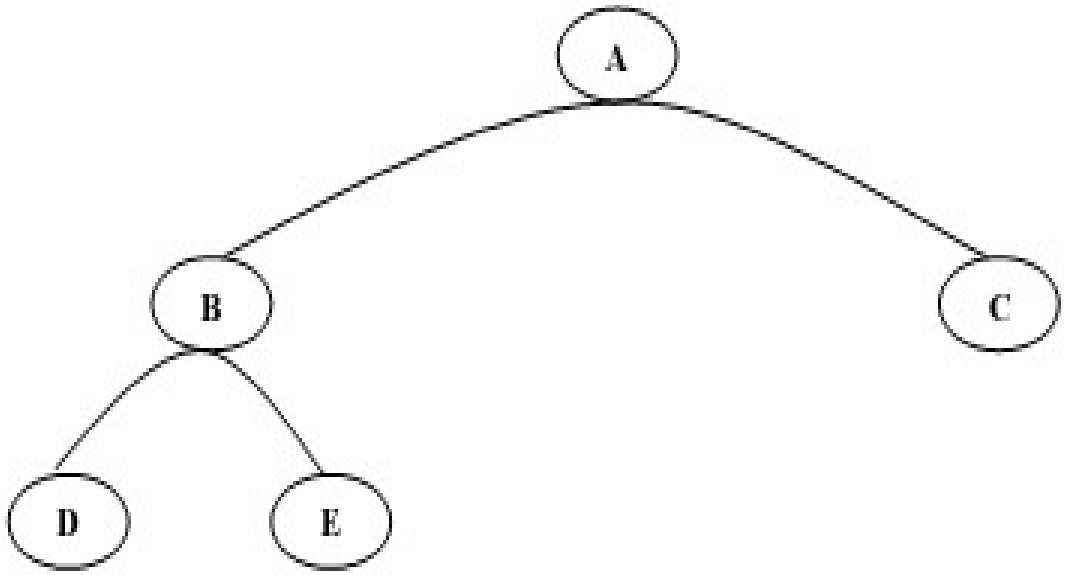
Binary Tree

- Full Binary Tree: semua node (kecuali leaf) pasti memiliki 2 anak dan tiap subtree memiliki panjang path yang sama.



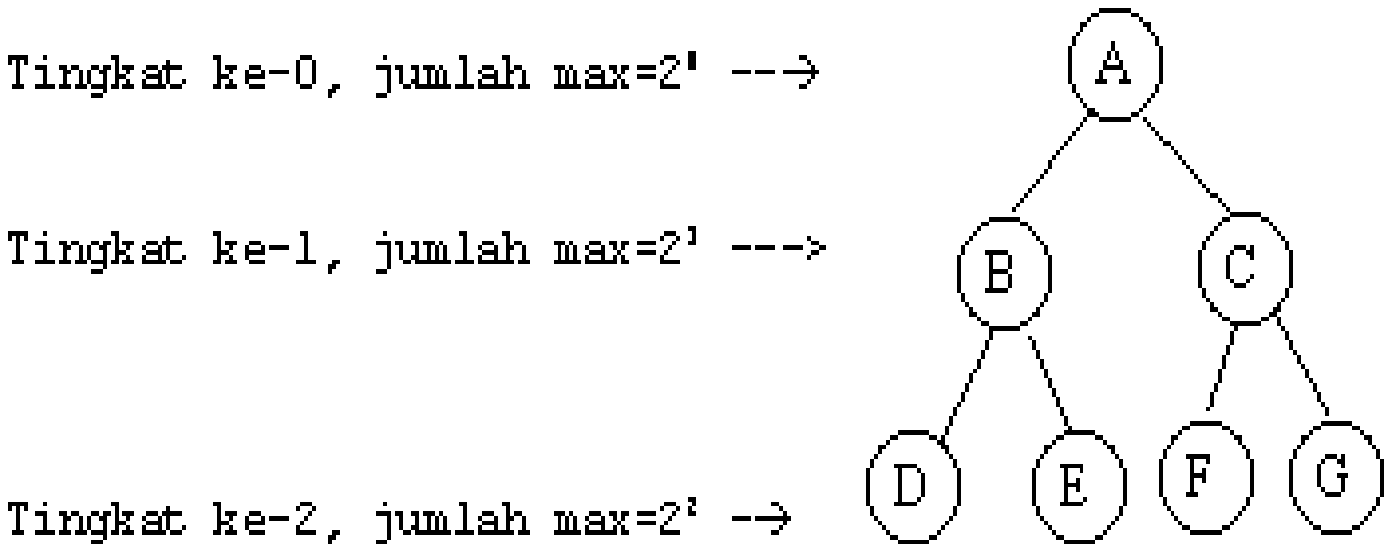
Binary Tree

- Complete Binary Tree: mirip dengan full binary tree, tapi tiap subtree boleh memiliki panjang path yang berbeda dan tiap node (kecuali leaf) memiliki 2 anak.



Node pada binary tree

- Jumlah maksimum node pada setiap tingkat adalah 2^n
- Node pada binary tree maksimum berjumlah 2^n-1



Gambar 6.4. Pohon Biner Tingkat 2 Lengkap

Pohon Biner Basis-0

- Definisi rekursif
  - Basis : pohon biner kosong adalah pohon biner
  - Rekurens : pohon biner yang tidak kosong, terdiri dari sebuah node yang disebut akar, dan sub pohon kiri dan sub pohon kanan sebagai anak-anaknya yang juga merupakan pohon biner

Pohon Biner Basis-0

TYPE POHON BINER: Model -0, dengan basis pohon kosong

DEFINISI DAN SPESIFIKASI TYPE

type Elemen : { tergantung type node }

type PohonBiner : <L : PohonBiner, A : Elemen, R : PohonBiner > {notasi Infix}, atau

type PohonBiner : < A : Elemen, L : PohonBiner, R : PohonBiner > {notasi prefix}, atau

type PohonBiner : <L : PohonBiner, R : PohonBiner, A : Elemen > {notasi postfix }

{Pohon Biner terdiri dari Akar yang berupa elemen, L dan R adalah Pohon biner yang merupakan subPohon kiri dan subpohon kanan }

DEFINISI DAN SPESIFIKASI SELEKTOR

Akar : PohonBiner tidak kosong → Elemen

{ Akar(P) adalah Akar dari P. Jika P adalah /L,A,R\ = Akar(P) adalah A }

Left : PohonBiner tidak kosong → PohonBiner

{ Left(P) adalah sub pohon kiri dari P. Jika P adalah /L,A,R\ = Left (P) adalah L }

Right : PohonBiner tidak kosong → PohonBiner

{Right(P) adalah sub pohon kanan dari P. Jika P adalah /L,A,R\ = Right (P) adalah R }

DEFINISI DAN SPESIFIKASI KONSTRUKTOR

{Perhatikanlah bahwa konstruktor pohon biner dengan basis pohon kosong dituliskan sebagai

a. Infix : /L A R

b. Prefix : /A L R

c. Posfix : /L R A\ }

Pohon Biner Basis-0

DEFINISI DAN SPESIFIKASI PREDIKAT

IsTreeEmpty : PohonBiner → boolean

{IsTreeEmpty (P) true jika P adalah Pohon biner kosong : (/ \) }

DEFINISI DAN SPESIFIKASI PREDIKAT LAIN

NbElmt : PohonBiner → integer ≥ 0

{NbElmt(P) memberikan Banyaknya elemen dari pohon P :
Basis : NbElmt (//\ \) = 0
Rekurens : NbElmt (/L,A,R\) = NbElmt(L) + 1 + NbELmt(R) }

NbDaun : PohonBiner → integer ≥ 0

{ definisi : Pohon kosong berdaun 0 }
{NbDaun (P) memberikan Banyaknya daun dari pohon P :
Basis-1 : NbDaun (//\ \) = 0
Rekurens :
***NbDaun1** (P)*

RepPrefix: PohonBiner → list of element

{RepPrefix (P) memberikan representasi linier (dalam bentuk list), dengan urutan elemen list sesuai dengan urutan penulisan pohon secara prefix :
Basis : RepPrefix (//\ \) = []
Rekurens :RepPrefix (/L,A,R\) = [A] o RepPrefix(L) o RepPrefix (R) }

Pohon Biner Basis-0

REALISASI

```
NbElmt (P) : {boleh model basis-0 }
   if IsTreeEmpty?(P) then {Basis 0} 0
   else {Rekurens } NbElmt(Left(P) + 1 + NbElmt(Right(P)


NbDaun (P) :
 if IsEMpty?(P) then 0
 else {Pohon tidak kosong:minimal mempunyai satu akar, sekaligus daun}
   { aplikasi terhadap Jumlah Daun untuk Basis-1 }
     NbDaun1 (P)


RepPrefix (P) :
   if IsTreeEmpty(P) then {Basis 0} []
   else {Rekurens }
     KonsoL(KonsoL(Akar(P), RepPrefix(Left(P)), RepPrefix(Right(P)))
```

Pohon Biner Basis-I

- Definisi rekursif
  - Basis : pohon biner yang terdiri dari akar
  - Rekurens : pohon biner yang tidak kosong, terdiri dari sebuah node yang disebut akar, dan sub pohon kiri dan sub pohon kanan sebagai anak-anaknya yang juga merupakan pohon biner tidak kososng

Pohon Biner Basis-1

TYPE POHON BINER : Model-1: pohon minimal mempunyai satu elemen

DEFINISI DAN SPESIFIKASI TYPE

type Elemen : { tergantung type node }

typ} PohonBiner : <L : PohonBiner, A : Elemen, R : PohonBiner > {notasi Infix}, atau

type PohonBiner : < A : Elemen, L : PohonBiner, R : PohonBiner > {notasi prefix }, atau

type PohonBiner : <L : PohonBiner, R : PohonBiner, A : Elemen > {notasi postfix }

{Pohon Biner terdiri dari Akar yang berupa elemen, L dan R adalah Pohon biner yang merupakan subPOhon kiri dan subpohon kanan }

DEFINISI DAN SPESIFIKASI SELEKTOR

Akar : PohonBiner tidak kosong → Elemen

{ Akar(P) adalah Akar dari P. Jika P adalah //L A R\\ = Akar(P) adalah A }

Left : PohonBiner tidak kosong → PohonBiner

{ Left(P) adalah sub pohon kiri dari P. Jika P adalah //L A R\ \, Left (P) adalah L }

Right : PohonBiner tidak kosong → PohonBiner

{Right(P) adalah sub pohon kanan dari P. Jika P adalah //L A R\\,Right (P) adalah R}

DEFINISI DAN SPESIFIKASI KONSTRUKTOR

{Perhatikanlah bahwa konstruktor pohon biner dengan basis pohon kosong dituliskan sebagai

a. Infix : //L A R

b. Prefix : //A L R

c. Posfix : //L R A

atau bahkan notasi lain yang dipilih}

Pohon Biner Basis-I

DEFINISI DAN SPESIFIKASI PREDIKAT

IsTreeEmpty : PohonBiner → boolean

{IsTreeEmpty(P) true jika P kosong : (/// \\) }

IsOneElmt : PohonBiner → boolean

{IsOneElement(P) true jika P hanya mempunyai satu elemen, yaitu akar (/// A \\\) }

IsUnerLeft : PohonBiner → boolean

{IsUnerLeft(P) true jika P hanya mengandung sub pohon kiri tidak kosong: (///L A \\) }

IsUnerRight : PohonBiner → boolean

{IsUnerRight (P) true jika P hanya mengandung sub pohon kanan tidak kosong: (///A R\\) }

IsBiner : PohonBiner tidak kosong → boolean

{IsBiner(P) true jika P mengandung sub pohon kiri dan sub pohon kanan :
(///L A R\\) }

IsExistLeft : PohonBiner tidak kosong → boolean

{IsExistLeft (P) true jika P mengandung sub pohon kiri }

IsExistRight : PohonBiner tidak kosong → boolean

{ExistRight(P) true jika P mengandung sub pohon kanan }

Pohon Biner Basis-I

DEFINISI DAN SPESIFIKASI PREDIKAT LAIN

NbElmt : PohonBiner → integer ≥ 0

{NbElmt(P) memberikan Banyaknya elemen dari pohon P :
Basis : NbElmt (///A\ \) = 1
Rekurens : NbElmt (///L,A,R\\ ) = NbElmt(L) + 1 + NbELmt(R)
NbElmt (///L,A,\ ) = NbElmt(L) + 1
NbElmt (///A,R\\ ) = 1 + NbELmt(R) }

NbDaun1 : PohonBiner → integer ≥ 1

{ Prekondisi : Pohon P tidak kosong }

{NbDaun (P) memberikan Banyaknya daun dari pohon P :
Basis : NbDaun1 (///A\ \) = 1
Rekurens : NbDaun1 (///L,A,R\\) = NbDaun1 (L) + NbDaun1(R)
NbDaun1 (///L,A,\\) = NbDaun1 (L)
NbDaun1 (///A,R\\) = NbDaun1 (R)

RepPrefix: PohonBiner → list of element

{RepPrefix (P) memberikan representasi linier (dalam bentuk list), dengan urutan elemen list sesuai dengan urutan penulisan pohon secara prefix :
Basis : RepPrefix (///A\ \) = [A]
Rekurens : RepPrefix (///L,A,R\\) = [A] o RepPrefix(L) o RepPrefix (R)
RepPrefix (///L,A\\) = [A] o RepPrefix(L)
RepPrefix (///A,R\\) = [A] o RepPrefix (R)
}

Pohon Biner Basis-1

REALISASI

```
NbElmt (P) : {P tidak kosong }
   if IsOneElmt(P) then 1
   else depend on P
            IsBiner(P) : NbElmt(Left(P) + 1 + NbElmt(Right(P)
            IsUnerLeft(P) : NbElmt(Left(P) + 1
            IsUnerRight(P) : 1 + NbElmt(Right(P)

NbDaun (P) :
   if 1Element?(P) then {Basis}
         1
   else {Rekurens }
         depend on P
            IsBiner(P) : NbDaun1(Left(P)) +  NbDaun1(Right(P))
            IsUnerLeft(P) : NbDaun1(Left(P))
            IsUnerRight(P) : NbDaun1(Right(P))

RepPrefix (P) :
   if 1Elmt?(P) then [Akar(P)]
   else depend on P
            IsBiner(P):KonsoL(KonsoL(Akar(P),RepPrefix(Left(P)),
                     RepPrefix(Right(P)))
            IsUnerLeft(P) : KonsoL(Akar(P),RepPrefix(Left(P))
            IsUnerRight(P) : KonsoL(Akar(P),RepPrefix(Right(P))
```

Class PohonBiner

```
#DefSpek
#type PohonBiner : < A: elemen, L: PohonBiner, R: PohonBiner>
#<A,L,R> adalah type bentukan pohon biner dimana A adalah akar, L adalah daun kiri, dan R adalah daun kanan
class PohonBiner:
      def __init__(self,A,L,R):
            self.A = A
            self.L = L
            self.R = R

#DefSpek
#MakePB: 3 integer ---> PohonBiner
#MakePB(A,L,R) menghasilkan sebuah pohon biner dengan A adalah akar, L adalah daun kiri, dan R adalah daun kanan
def MakePB(A,L,R):
      return PohonBiner(A,L,R)
```

Akar, Sub Pohon Kiri dan Kanan

```
#Fungsi Akar
#DefSpek
#Akar (p) pohon biner tak kosong ---> pohon biner
def Akar(P):
      return P.A

#Fungsi Left
#DefSpek
#Left : pohon biner tak kosong ---> pohon biner
#Left (P) adalah sub pohon kiri dari P jika /L, A, R\ maka left (P) adalah L
def Left(P):
      return P.L

#Fungsi Right
#DefSpek
#Right : pohon biner tak kosong ---> pohon biner
#Right (P) adalah sub pohon kanan dari P jika /L, A , R\ maka right (P) adalah R
def Right(P):
      return P.R
```