Tipe Rekursif:
POHON (TREE)

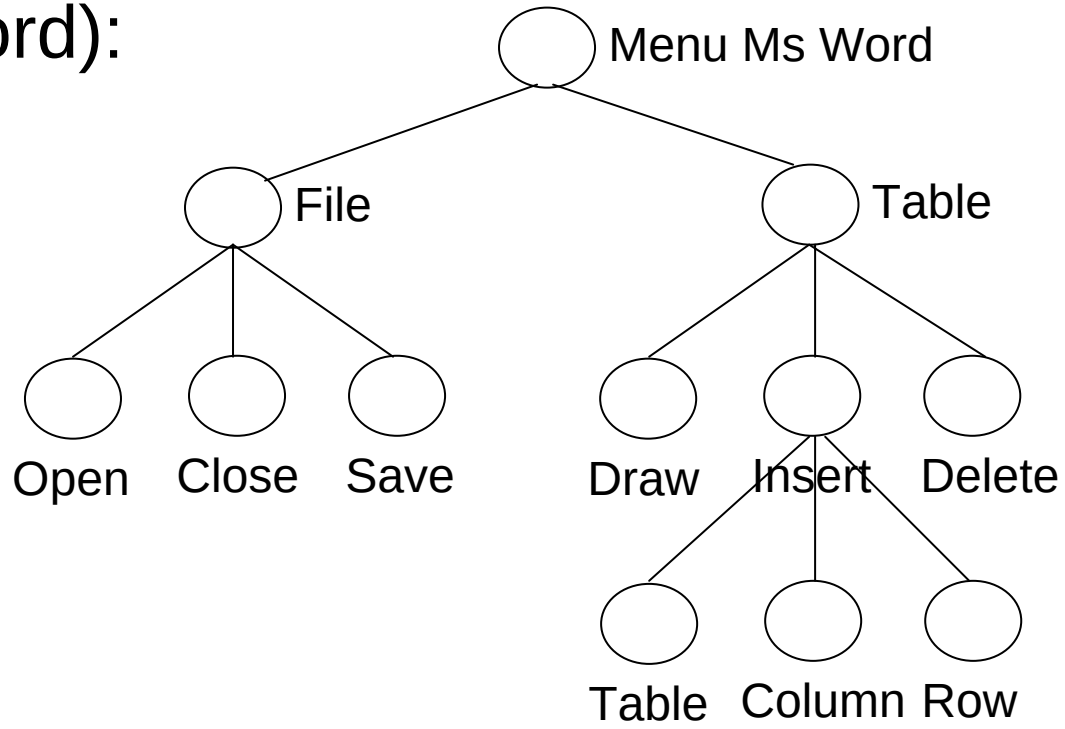
Tim Pengajar IF2030

Tujuan

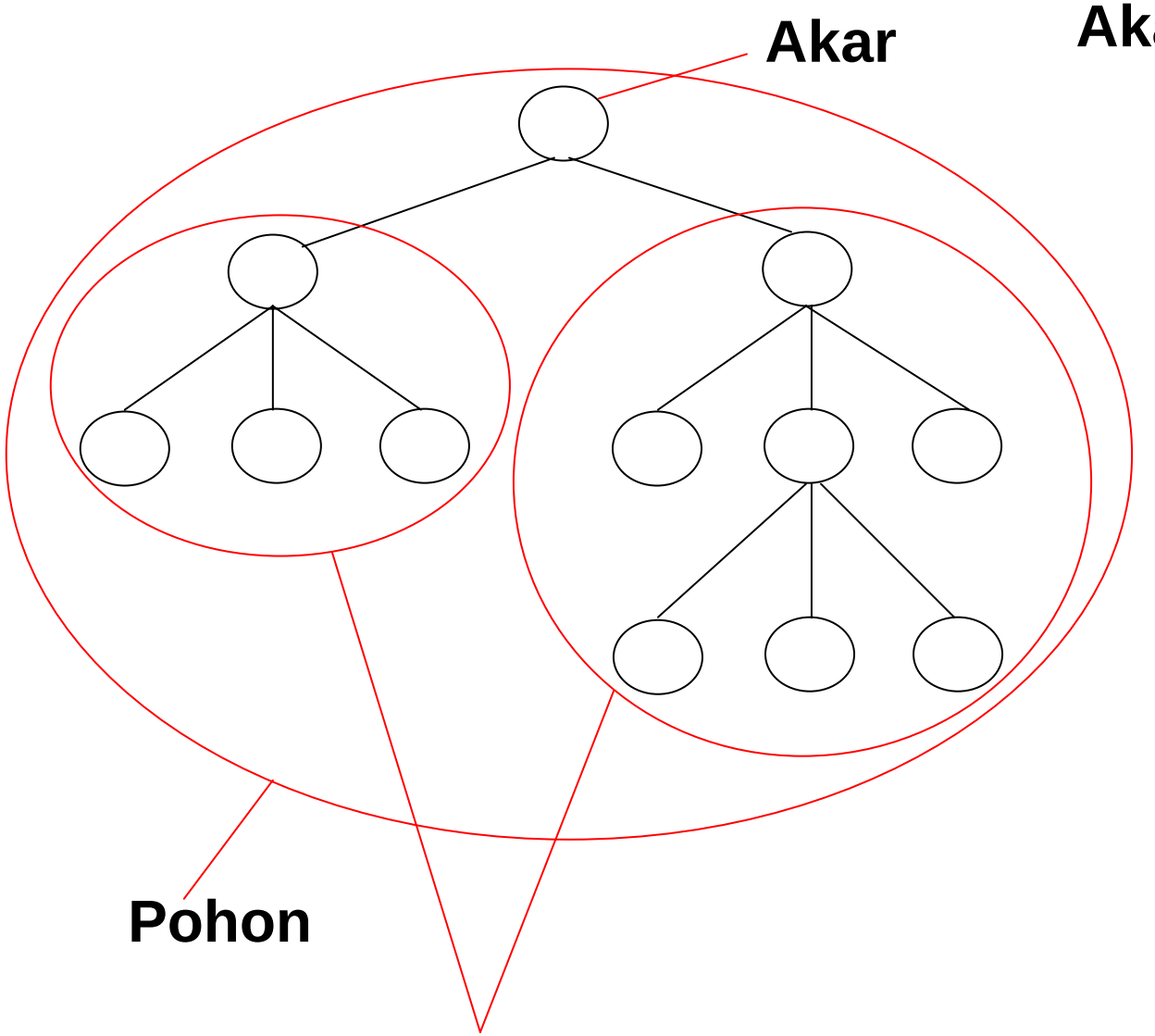
- Mahasiswa memahami definisi pohon dan pohon biner
- Berdasarkan pemahaman tersebut, mampu membuat fungsi sederhana yang memanipulasi pohon
- Mahasiswa mampu mengimplementasi fungsi pemroses pohon dalam LISP → melalui praktikum

Contoh Persoalan Direpresentasi Sbg Pohon

- Menu dalam Aplikasi Komputer
- Contoh (Ms Word):
  - File
    - Open
    - Close
    - Save
  - Table
    - Draw
    - Insert
      - Table
      - Column
      - Row
    - Delete



Pohon

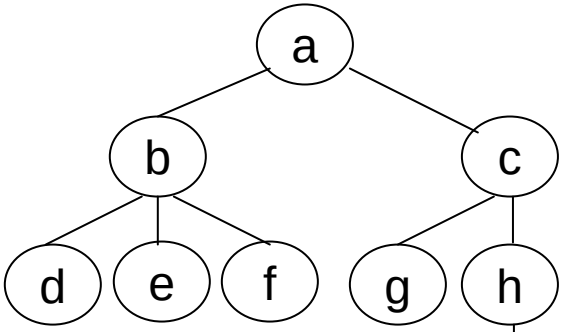


Akar

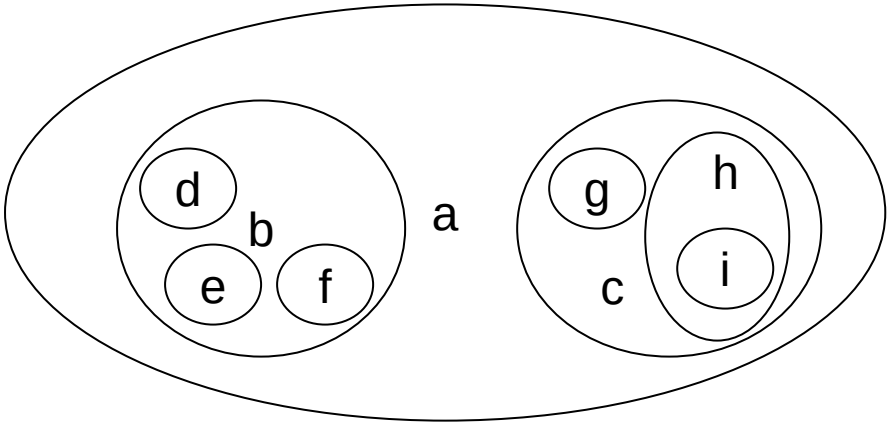
SubPohon

Elemen Pohon:
-Akar → basis
-Sub Pohon (sub himpunan yang berupa pohon) → rekurens

Beberapa Ilustrasi Representasi



Graph



Himpunan

Indentasi

```
a
    b
        d
        e
        f
    c
        g
        h
            i
```

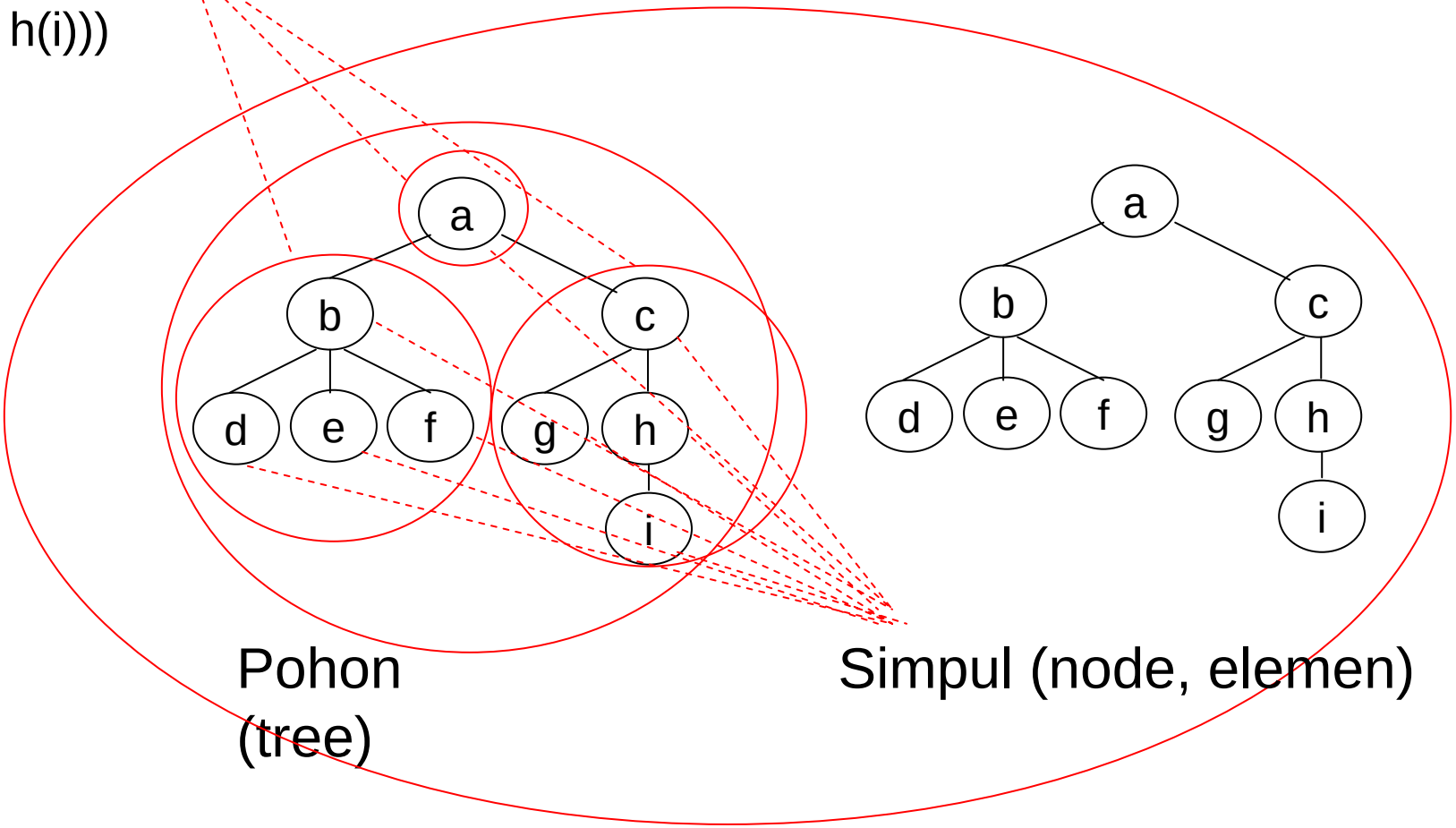
Bentuk Linier

Prefix: (a (b (d (), e (), f ()), c (g (), h (i ()))))
(a (b (d) (e) (f)) (c (g) (h (i))))

Postfix: (((d,e,f) b, (g, (i) h) c) a)

Istilah

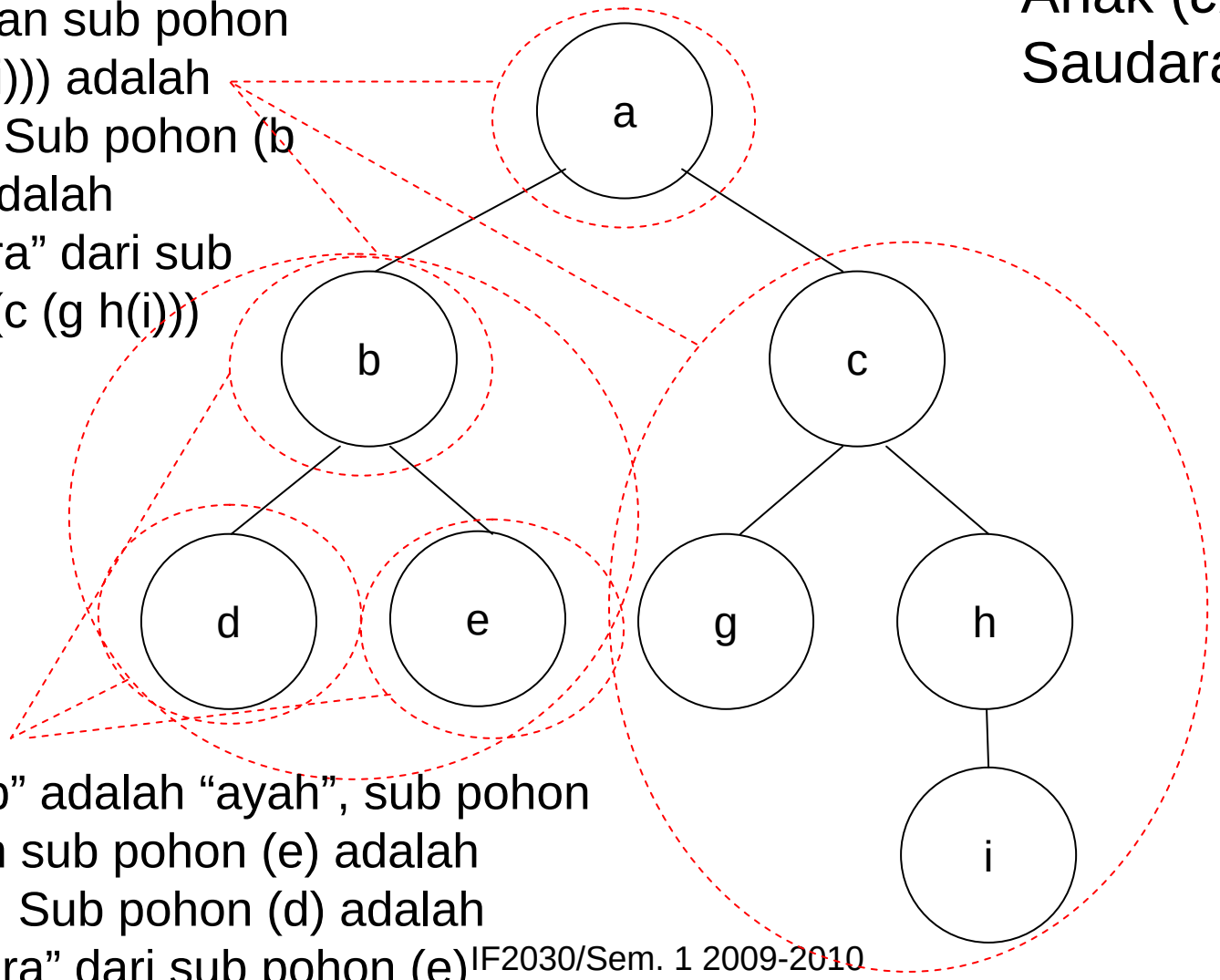
Cabang → akar “a” memiliki cabang 2 sub pohon yaitu (b (d e f)) dan (c (g h(i)))



Istilah

Akar “a” adalah “ayah”, sub pohon (b (d e)) dan sub pohon (c (g h(i))) adalah “anak”. Sub pohon (b (d e)) adalah “saudara” dari sub pohon (c (g h(i)))

Ayah (*father*)
Anak (*child*)
Saudara (*sibling*)

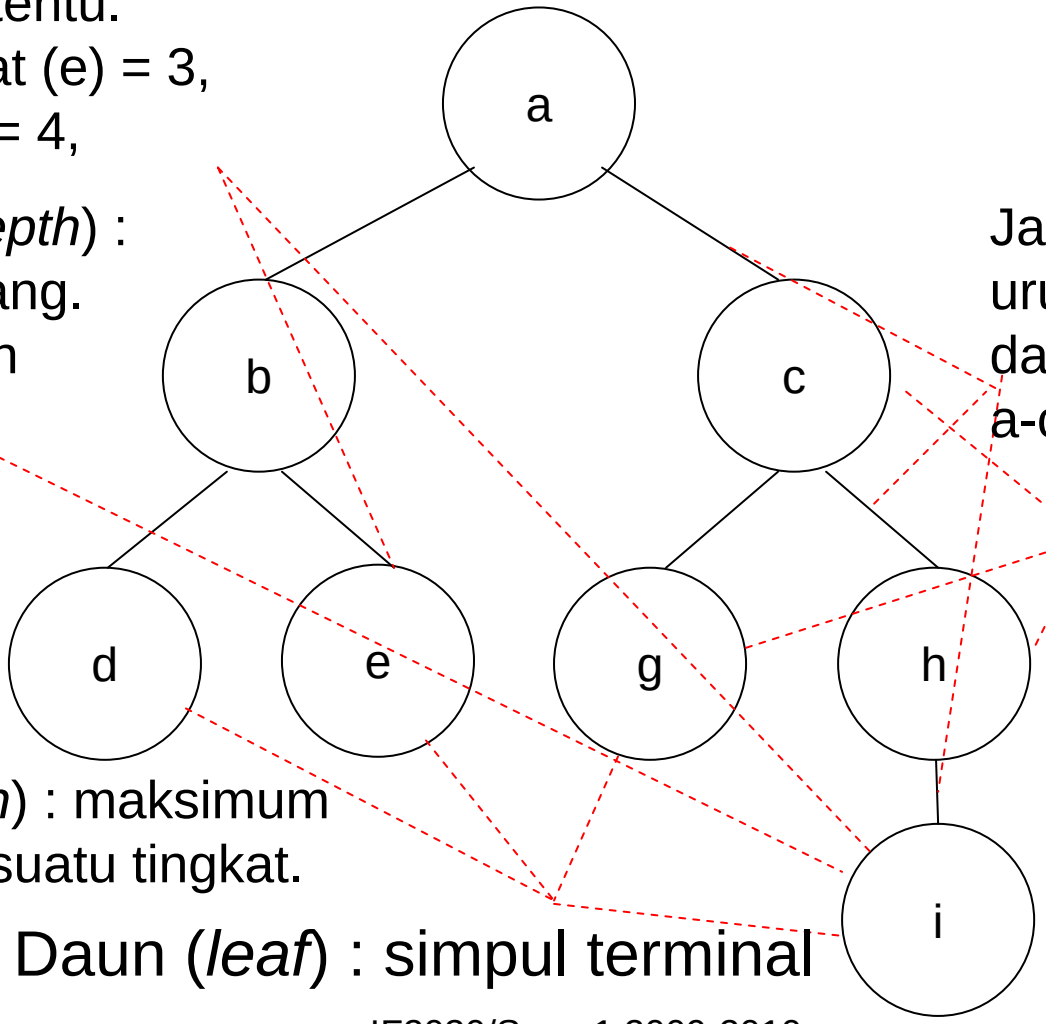


Akar “b” adalah “ayah”, sub pohon (d) dan sub pohon (e) adalah “anak”. Sub pohon (d) adalah “saudara” dari sub pohon (e)

Istilah

Tingkat (*level*) : panjang jalan dari akar sampai simpul tertentu. Cth: tingkat (e) = 3, tingkat (i) = 4,

Kedalaman (*depth*) : tingkat terpanjang. Cth: kedalaman pohon=4



Jalan (*path*) : urutan tertentu dari cabang, cth: a-c-h-i

Derajat : banyaknya anak sebuah simpul. Cth, derajat(c)=2, derajat(h)=1, derajat(g)=0

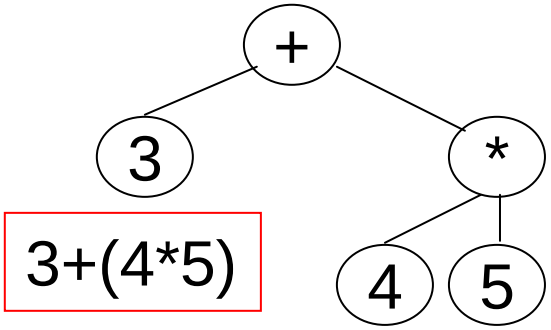
Lebar (*breadth*) : maksimum jml simpul pd suatu tingkat.

Daun (*leaf*) : simpul terminal

Pohon Biner

- Definisi
  1. Mungkin kosong
  2. Ada simpul akar dan dua anak yang berupa pohon biner, satu sub pohon kiri dan satu sub pohon kanan. Anak mungkin kosong (kembali ke definisi 1).

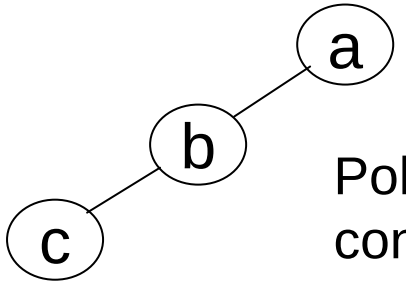
  Penulisan sub pohon:
  dgn ( )



3+(4*5)

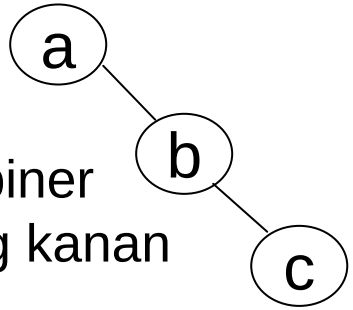
(+ (3 ( ) ( )) (* (4 ( )( )) (5 ( )( ))))
atau (+ (3) (* (4)(5)))

Pohon condong/*skewed tree*



Pohon biner condong kiri

Notasi Prefix: (a (b (c ( ) ( )) ( )) ( ))
atau (a (b (c)( )) ( ))



Pohon biner condong kanan

Notasi Prefix: (a ( ) (b ( )(c ( ) ( )))) atau
(a ( ) (b ( )(c)))

TYPE POHON BINER

DEFINISI DAN SPESIFIKASI TYPE

type Elemen: {tergantung type node}
type PohonBiner:<L: PohonBiner, A: Elemen, R: PohonBiner> {infix}, **atau**
type PohonBiner:<A: Elemen, L: PohonBiner, R: PohonBiner> {prefix}, **atau**
type PohonBiner:<L: PohonBiner, R: PohonBiner , A: Elemen> {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 KONSTRUKTOR

{Perhatikanlah bhw konstruktor pohon biner dg basis pohon kosong ditulis:
a. Infix: //L A R\\
b. Prefix: //A L R\\
c. Postfix: //L R A\\}

Selektor

Akar: pohon biner tidak kosong → elemen

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

Left: pohon biner tidak kosong → pohon biner

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

Right: pohon biner tidak kosong → pohon biner

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

Predikat

IsTreeEmpty: pohon biner → boolean

{IsTreeEmpty(P) true jika P adalah // \\}

IsOneElmt: pohon biner → boolean

{IsOneElmt(P) true jika P adalah //A\\}

IsUnerLeft: pohon biner → boolean

*{IsUnerLeft(P) true jika P **hanya** mengandung sub pohon kiri, P adalah //L A\\}*

IsUnerRight: pohon biner → boolean

*{IsUnerRight(P) true jika P **hanya** mengandung sub pohon kanan, P adalah //A R\\}*

Predikat

IsBiner: pohon biner tidak kosong → boolean

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

IsExistLeft: pohon biner tidak kosong → boolean

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

IsExistRight: pohon biner tidak kosong → boolean

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

Pohon Basis-0

- Definisi rekursif
  - Basis: pohon biner kosong adalah pohon biner {menggunakan predikat IsTreeEmpty}
  - Rekurens: Pohon biner tidak kosong terdiri dari sebuah simpul akar dan dua anak: sub pohon kiri dan sub pohon kanan. Sub pohon kiri dan sub pohon kanan adalah pohon biner

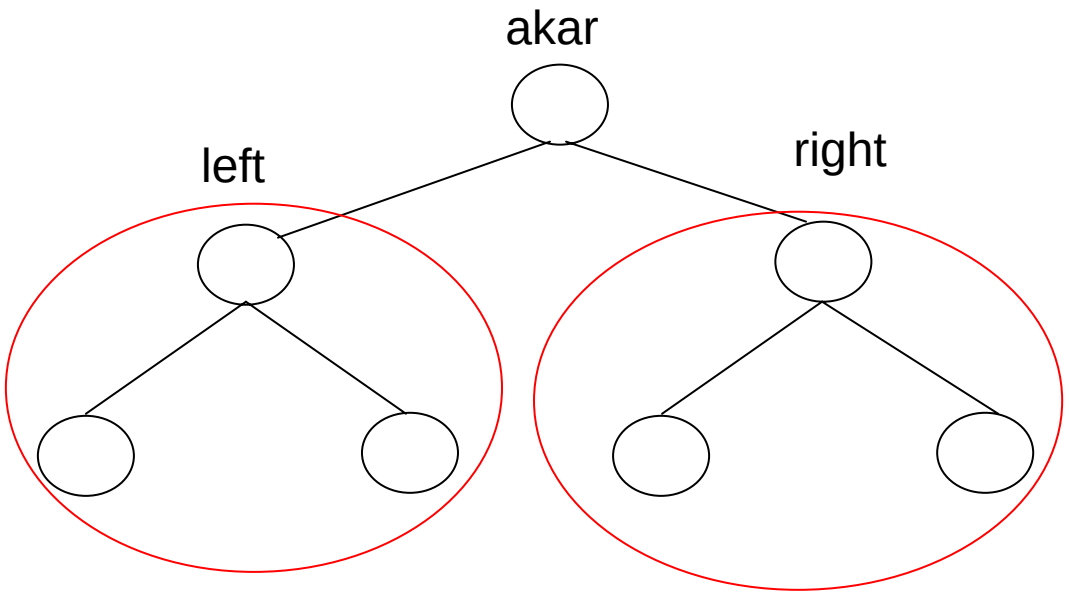
Pohon Basis-1

- Definisi rekursif
  - Basis: pohon biner yang hanya terdiri dari akar {menggunakan predikat IsOneElmt}
  - Rekurens: Pohon biner tidak kosong terdiri dari sebuah simpul akar dan dua anak yang salah satunya pasti tidak kosong: sub pohon kiri dan sub pohon kanan.
    - Gunakan IsUnerLeft, IsUnerRight, IsBiner untuk memastikan tidak terjadi pemrosesan pada pohon kosong

Menghitung Jumlah Elemen

NbElmt: PohonBiner → integer >= 0
{NbElmt(P) memberikan banyaknya elemen dari pohon P}

Berapa jumlah elemen pohon dilihat dari elemen current?



Jumlah elemen =
1 (utk akar) + Jumlah_Elemen (pohon kiri) + Jumlah _Elemen (pohon kanan)

Menghitung Jumlah Elemen, NbElmt, hal 109

- Rekursif
  - Basis 0: jika pohon kosong maka jumlah elemen adalah 0
  - Rekurens: jumlah elemen = 1 (current element) + jumlah elemen pohon kiri + jumlah elemen pohon kanan

```
NbElmt (P):
  if IsTreeEmpty(P) then 0  {basis 0}
  else  {rekurens}
      NbElmt(Left(P)) + 1 + NbElmt(Right(P))
```

Menghitung Jumlah Elemen, NbElmt, hal 111

- Basis 0

```
NbElmt (P):
  if IsTreeEmpty(P) then 0  {basis 0}
  else  {rekurens}
     NbElmt(Left(P)) + 1 + NbElmt(Right(P))
```

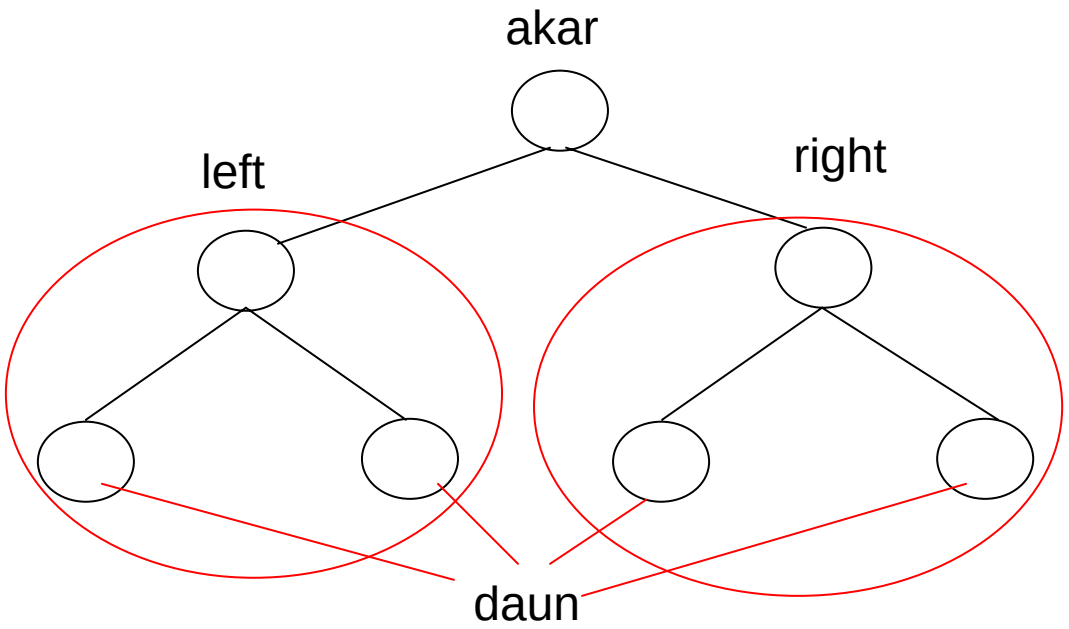
- Basis 1

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

Menghitung Jumlah Daun

NbDaun: PohonBiner → integer >= 0
{NbDaun(P) memberikan banyaknya daun dari pohon P}

Berapa jumlah daun pohon dilihat dari elemen current?



Jumlah daun =
0 (utk akar) + Jumlah daun (pohon kiri) + Jumlah daun (pohon kanan)

Menghitung Jumlah Daun, NbDaun, hal 109

- Analisis Kasus
  - Pohon kosong : jumlah daun = 0
  - Pohon tidak kosong: jumlah daun dihitung dengan fungsi menghitung jumlah daun dengan basis-1

```
NbDaun (P):
  if IsTreeEmpty(P) then 0  { analisis kasus }
  else
     NbDaun1(P)
```

Menghitung Jumlah Daun, NbDaun, hal 112

- Rekursif: Basis 1

```
NbDaun1 (P):
    if IsOneElmt(P) then 1  {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))
```

Latihan

- Buatlah realisasi dengan spesifikasi sebagai berikut:
  1. IsMember: PohonBiner, elemen → boolean
     { IsMember(P,X) true jika ada elemen di P yang bernilai X }
  2. IsSkewLeft: PohonBiner → boolean
     { IsSkewLeft(P) true jika P adalah pohon condong kiri. Pohon kosong dianggap sebagai pohon yang condong kiri. Pohon satu elemen dianggap sebagai pohon condong kiri. }
  3. LevelOfX: PohonBiner tdk kosong, elemen → integer
     { LevelOfX(P,X) mengirimkan level simpul X yang merupakan elemen dari P. Prekondisi: X pasti ada di P. P adalah pohon yang tidak kosong dan elemen-elemen P unik.}

Solusi Latihan 1
IsMember(P,X)

Definisi dan Spesifikasi

IsMember: PohonBiner, elemen → boolean

{ IsMember(P,X) true jika ada elemen di P yang bernilai X }

Realisasi

```
IsMember(P,X):
        if IsTreeEmpty(P) then false {Basis-0}
        else {Rekurens}
                if (Akar(P) = X) then true
                else
                        IsMember(Left(P),X) or IsMember(Right(P),X)
```

Solusi Latihan 2
IsSkewLeft(P)

Definisi dan Spesifikasi

IsSkewLeft: PohonBiner → boolean

{ IsSkewLeft(P) true jika P adalah pohon condong kiri. Pohon kosong dianggap sebagai pohon yang condong kiri. Pohon satu elemen dianggap sebagai pohon condong kiri. }

IsSkewLeft1: PohonBiner tidak kosong → boolean

{ IsSkewLeft1(P) true jika P adalah pohon condong kiri. }

Realisasi

```
IsSkewLeft(P) : if IsTreeEmpty(P) then true { Analisis Kasus }
                else IsSkewLeft1(P)
IsSkewLeft1(P) :
        if IsOneElmt(P) then true { Basis-1 }
        else { Rekurens }
                if IsUnerLeft(P) then IsSkewLeft1(Left(P))
                else false
```

Solusi Latihan 3
LevelOfX (P,X)

Definisi dan Spesifikasi

LevelOfX: PohonBiner, elemen → integer

{ LevelOfX(P,X) mengirimkan level simpul X yang merupakan elemen dari P. Prekondisi: X pasti ada di P. P adalah pohon yang tidak kosong dan elemen-elemen P unik.}

IsMember: PohonBiner tidak kosong, elemen → boolean

{ IsMember(P,X) true jika ada elemen di P yang bernilai X. }

Realisasi

```
LevelOfX(P,X):
        if Akar(P) = X then 1 {Basis-1}
        else { Rekurens }
                if IsMember(Left(P),X) then 1+LevelOfX(Left(P),X)
                else { pasti IsMember(Right(P),X) = true }
                        1+LevelOfX(Right(P),X)
```

MakeListPreOrder

Definisi dan Spesifikasi

MakeListPreOrder : PohonBiner → list of node

{ MakeListPreOrder(P) : Jika P adalah pohon kosong, maka menghasilkan list kosong. Jika P bukan pohon kosong: menghasilkan list yang elemennya adalah semua node pohon P dengan urutan preorder. }

Realisasi

{ primitif-primitif list of node mengacu pada ADT list of integer}

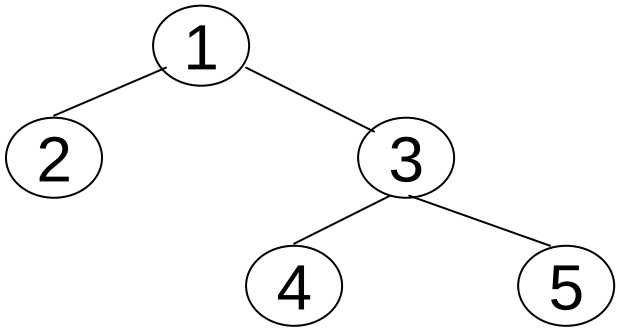
```
MakeListPreOrder(P):
        if (IsTreeEmpty(P)) then [ ]  {basis-0}
        else {rekurens}
                Konso (Akar(P),
                        Konkat (MakeListPreOrder(Left(P),
                                MakeListPreOrder(Right(P))))
```

Translasi LISP

Contoh Representasi Pohon Biner di LISP

- Menggunakan List
- Contoh representasi prefix

  (1 (2 () ()) (3 (4 () ()) (5 () ())))



Selektor

```
(defun Akar (PB)
    (car PB))

(defun Left (PB)
    (car (cdr PB)))

(defun Right (PB)
    (car (cdr (cdr PB))))
```

Predikat

```
(defun IsTreeEmpty (P)
    (null P))

(defun IsOneElmt (P)
   (and (not (IsTreeEmpty P))
        (IsTreeEmpty (Left P))
        (IsTreeEmpty (Right P))))
```

Predikat

```
(defun IsUnerLeft (P)
   (and (IsTreeEmpty (Right P))
        (not (IsTreeEmpty (Left P)))))

(defun IsUnerRight (P)
   (and (IsTreeEmpty (Left P))
        (not (IsTreeEmpty (Right P)))))

(defun IsBiner (P)
   (and (not (IsTreeEmpty (Right P)))
        (not (IsTreeEmpty (Left P)))))
```

Fungsi Lain

```
(defun NbElmt (P)
   (if (IsTreeEmpty P) 0  ; basis-0
      (+ (NbElmt (Left P))   ; rekurens
         1
         (NbElmt (Right P)))))
```

Tugas di Rumah

- Materi pra-praktikum:
  - ADT pohon biner
    - Butuh ADT list of node → adaptasikan dari ADT list of integer

Kuis-1

- Kuis 1 akan dilaksanakan pada hari Kamis, 10 Sept 2009 pada jam kuliah (14.00-selesai)
- Tempat: di kelas masing-masing
- Materi: seluruh materi mulai dari minggu 1 s.d. minggu 4 (dalam notasi fungsional)

Praktikum Ke-4

- Praktikum ke-4 akan dilaksanakan pada:
  - Senin, 28 Sept 2009
  - Selasa, 29 Sept 2009
- Materi: pohon biner
- Untuk melihat pembagian shift terbaru, lihat pengumuman di:
  - Mailing list
  - Situs kuliah online