MODUL XI

Struktur Data

Judul	Struktur Pohon Dengan Array	
Penyusun	**Distribusi**	**Perkuliahan**
Nixon Erzed	Teknik Informatika Universitas Esa Unggul	Pertemuan – XI online

Tujuan :

Mahasiswa memahami struktur pohon dan representasi statik pohon dengan array

Materi :

1. Representasi Pohon dengan Array
2. Manipulasi Pohon

REPRESENTASI POHON DENGAN ARRAY

Penggunaan array untuk merepresentasikan struktur pohon, merupakan pilihan lain selain *linked list.* Struktur array bersifat statis, sehingga ukuran array yang dideklarasikan harus sudah memperhitungkan kebutuhan maksimum ruang data dalam beroperasinya atau dieksekusinya program.

Representasi pohon dengan array, berupa tabel dengan struktur/record bentukan:

- → elemen dari record : field untuk data, field-field pointer anak
- → isi dari field pointer : nomor record (index array) data anak

Berikut ini akan dipaparkan representasi pohon biner, pohon 4 (quadtree) dan pohon n-cabang.

a. untuk Pohon Biner :

	Data	LC	RC
# 1			
#2			
Dst			

LC → Left Child atau anak kiri

RC → Right Child atau anak kanan

Singkatan LC dan RC umum digunakan dalam deklarasi struktur data pohon biner

Deklarasi struktur data untuk pohon biner :

```
Type
    simpul = record of
        Data        : <TipeData>
        Left, Right : integer {LC dan RC}
    End;
Var
    BTree : array [1..n] of simpul
```

b. untuk Quadtree

	Data	NW	NE	SE	SW
# 1					
# 2					
Dst					

NW → north west atau anak kiri atas
NE → north east atau anak kanan atas
SE → south eats atau anak kanan bawah
SW → south west atau anak kiri bawah

Penggunaan simbolik arah mata angin, disesuaikan dengan penggunaan popular pohon empat-an (quadtree) untuk merepresantasi bidang gambar/image.

NW	NE
SW	SE

Deklarasi Data untuk Pohon 4 Cabang (*Quadtree*)

```
Type
    simpul  =  record of
      |  data   :  <TipeData>
      |  nw, ne, se, sw  :  integer
    end;
Var
     QTree : array [1 .. n] of simpul
```

Bandingkan dengan deklarasi quadtree dengan Linked list berikut ini :

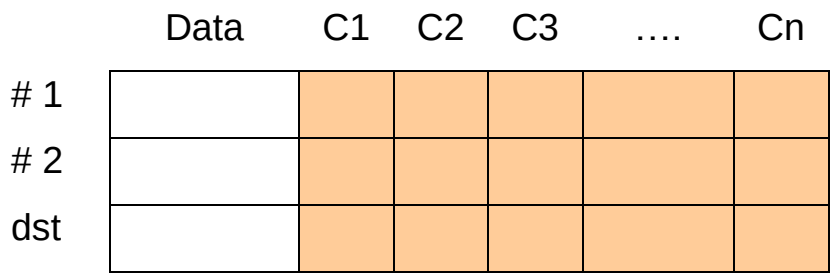
```
Type
    pTree  =  ^Simpul
    Simpul  =  record of
        data   :  <TipeData>
        NW, NE, SW, SE :  pTree
    End;
```

Yang harus diperhatikan type pointer berisi data alamat ruang memory, dengan format standar mikroprosesor, untuk sistem saat umumnya 1 alamat = 32 bit atau 4 byte

Jika data adalah satu bilangan bulat (integer) yang juga berukuran 4 byte, maka unkuran 1 simpul : 4 byte data + 16 byte pointer, atau dengan kata lain 80% ruang simpul adalah untuk pointer. Sehingga penggunaan linked list akan memboroskan ruang memory.

c. untuk n-ary Tere

(n-Ary Tree → Ener Tree Atau Eneri Tree

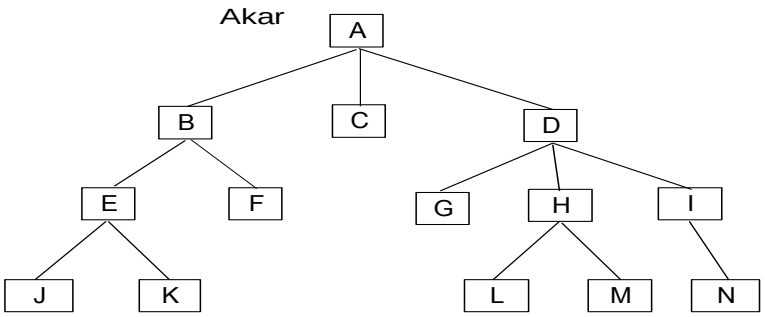


Deklarasi Data untuk Pohon ***n*** Cabang :

```
Type
    Simpul   = record
       |  data                 :  <type data>;
       |  c1, c2, c3, … cn  :  integer
    end

Var
     nTree  : array [1.. n] of simpul
```

Contoh ilustrasi Tree yang disusun dengan array :



	data	C1	C2	C3
# 1.	A	2	3	4
# 2.	B	5	7	0
# 3.	C	0	0	0
# 4.	D	8	10	11
# 5.	E	6	9	0
# 6.	J	0	0	0
# 7.	F	0	0	0
# 8.	G	0	0	0
# 9.	K	0	0	0
# 10.	H	14	12	0
# 11.	I	13	0	0
# 12.	M	0	0	0
# 13.	N	0	0	0
# 14.	L	0	0	0

```
Type
      Simpul   = record
            data    :  CHAR
            C1, C2, C3  :  integer
        end

 var
      TTree  : array [1.. n] of simpul
```

Contoh Membangun Pohon Biner

Dimiliki data integer yang diorganisasikan dengan pohon biner, dengan ketentuan jika data tersebut lebih kecil dari parent maka tempat sebagai anak kiri, jika lebih besar atau sama tempat sebagai anak kanan, jika anak kiri atau kanan sudah ada maka tolak

Misalkan kedatangan data adalah sebagai berikut :

100 anak 0

72 anak 100

80 anak 72

52 anak 72

150 anak 100

95 anak 72 → akan ditolak

86 anak 150

102 anak 150 → akan ditolak, anak kiri sudah ada

Dst

Input : data dan siapa parent-nya

	data	LC	RC
1	100	2	5
2	72	4	3
3	80	0	0
4	52	0	0
5	150	6	0
6	86	0	0
7			
8			

Logika proses

Baca data dan parent

data adalah root jika parent adalah 0

Cek posisi pada parent → jika available →

tempatkan ke array, isikan 0 pada LC dan RC

sisipkan nomor record ke LC / RC dari parent

ambil data berikutnya

data habis jika Data dan Parent bernilai 0

```
Type
        SBin  = record of
                    Dat  : integer
                    LC, RC : integer
                End-SBin
Var
      Lat : array [1..100] of SBin
      Dt, PDt  : integer
      NoRec : integer


Program BangunPohon
Begin
     NoRec ⇐ 1
     Read(Dt, PDt)
     Repeat
          TambahSimpul(NoRec, Dt, PDt)
          Read(Dt, PDt)
          NoRec ⇐ NoRec + 1
     Until  Dt = 0  and PDt = 0
End-BangunPohon
```

```
Procedure TambahSimpul
(var nRec  : integer;    Data, Parent : integer )
Var    k   :  integer

Begin
     If  Parent = 0
      then  Lat [nRec].dat  ⇐  data
              Lat [nRec].LC  ⇐  0
               Lat [nRec].RC  ⇐ 0
      Else
              k   ⇐  1
              while Lat[ k ].dat <> parent  and  k < nRec
                do  k ⇐ k +1
              end-while

              If   Lat [k].dat = parent
               Then

                  If   data < parent and Lat [ k ].LC = 0
                    Then
                          Lat [nRec].dat  ⇐  data
                          Lat [nRec].LC  ⇐  0
                         Lat [nRec].RC  ⇐ 0
                         Lat [ k ] .LC  ⇐ nRec
                         nRec  ⇐ nRec + 1
                    End-if

               If   data > parent and Lat [ k ].RC = 0
                 Then
                          Lat [nRec].dat  ⇐  data
                          Lat [nRec].LC  ⇐  0
                          Lat [nRec].RC  ⇐ 0
                          Lat [ k ] .RC  ⇐ nRec
                          nRec  ⇐ nRec + 1
               End-if
        Else write  (“data ditolak”)
      End-if
   End-if
End-procedur
```