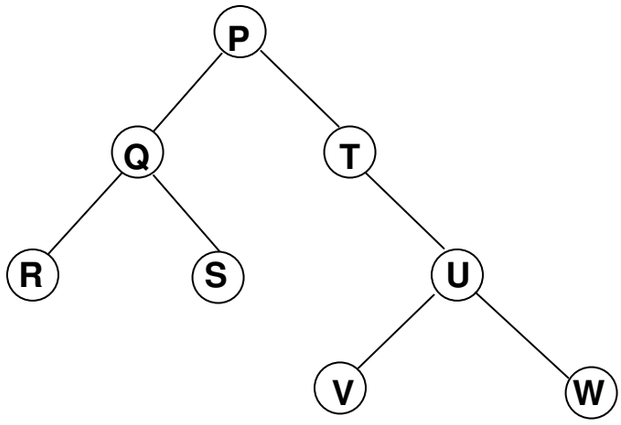
BAB 7

POHON BINAR

Pohon (Tree) adalah graf terhubung yang tidak mengandung sirkuit. Karena merupakan graf terhubung maka pada pohon selalu terdapat path atau jalur yang menghubungkan kedua simpul di dalam pohon. Pohon dilengkapi dengan *Root (akar)*.

Contoh : Pohon berakar T



Sifat utama pohon berakar :

1. Jika pohon mempunyai simpul sebanyak n, maka banyaknya ruas adalah (n-1). Pada contoh : banyak simpul adalah 8 maka banyaknya ruas adalah 7.
2. Mempunyai simpul khusus yang disebut Root (Akar), jika simpul tersebut memiliki derajat keluar ≥ 0 dan derajat masuk = 0. Simpul P merupakan root.
3. Mempunyai simpul yang disebut Leaf (Daun), jika simpul tersebut memiliki derajat keluar = 0 dan derajat masuk = 1. Simpul R, S, V, W merupakan daun pada pohon T.
4. Setiap simpul mempunyai tingkatan (level), dimulai dari root dengan level 0 sampai dengan level n pada daun yang paling bawah.
   Pada contoh :
   P mempunyai level 0
   Q, T mempunyai level 1
   R, S, U mempunyai level 2
   V, W mempunyai level 3
   Simpul yang mempunyai level yang sama disebut Bersaudara (Brother)
5. Pohon mempunyai ketinggian (kedalaman / height) yaitu level tertinggi +1. Ketinggian pohon T adalah 3+1 = 3
6. Pohon mempunyai berat (bobot / weight) yaitu banyaknya daun pada pohon. Berat pohon T adalah 4

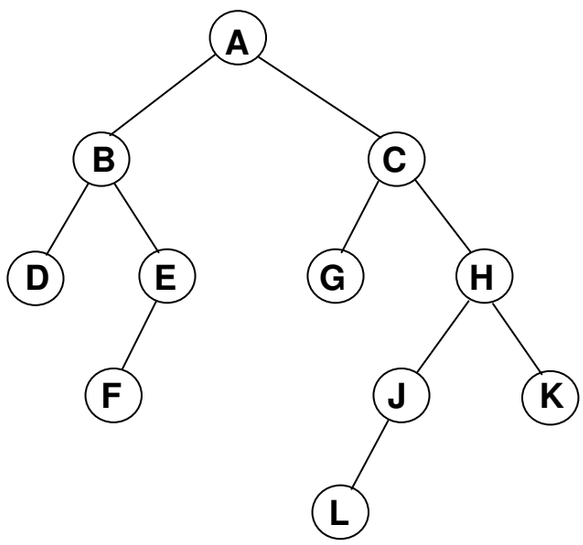
POHON BINAR (BINARY TREE)

Pohon binar adalah himpunan simpul yang terdiri dari 2 subpohon (yang disjoint / saling lepas) yaitu subpohon kiri dan subpohon kanan.

Setiap simpul dari pohon binar mempunyai derajat keluar maksimum = 2.

Pendefinisian pohon binar bersifat rekursif. Pohon binar acapkali disajikan dalam bentuk diagram.

Contoh :

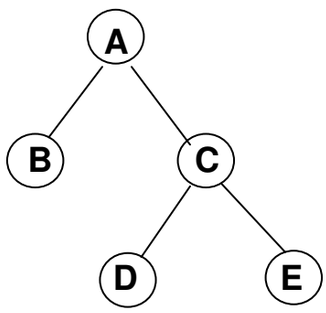


Untuk menggambarkan suksesor kiri dan suksesor kanan, dibuat garis ke kiri bawah dan ke kanan bawah. B adalah suksesor kiri dari A, sedangkan C adalah suksesor kanan dari A. Subpohon kiri dari A mengandung simpul B, D, E dan F, sedangkan subpohon kanan mengandung simpul C, G, H, J, K dan L.

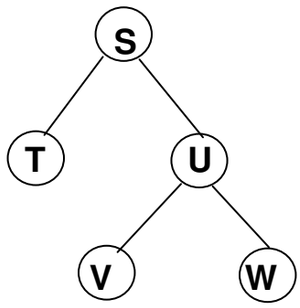
Pada contoh di atas :
Root adalah A
Simpul yang mempunyai 2 anak adalah simpul A, B, C dan H.
Simpul yang mempunyai 1 anak adalah simpul E dan J.
Simpul yang tidak mempunyai anak disebut daun (terminal) adalah D, F, G, K dan L.

Perhatikan pohon T1 dan T2 dan T3 ini :

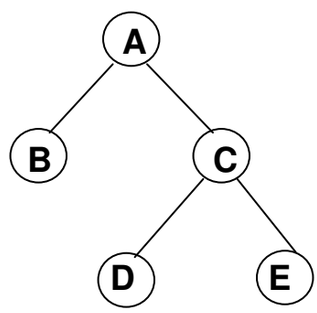
T1



T2



T3



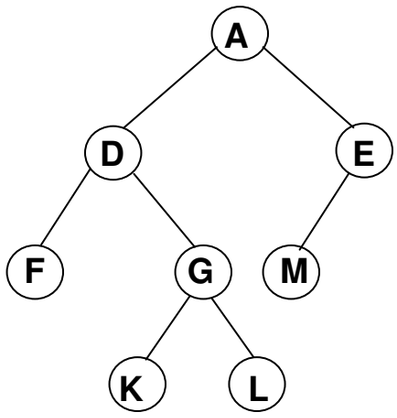
Dua pohon binar disebut similar jika mempunyai struktur (bangun/susunan) pohon yang sama.
Contoh : Pohon binar T1 dan T2 adalah similar.

Kedua pohon binar disebut Salinan (Ekivalen/Copy) jika :
1. Mempunyai struktur pohon yang sama (similar)
2. Elemen yang sama pada simpul yang bersesuaian.
   Contoh : Pohon binar T1 dan T3 adalah ekivalen

TERMINOLOGI PADA POHON BINAR

Terminologi hubungan keluarga banyak digunakan dalam terminologi pada pohon binar. Misalnya istilah anak kiri dan anak kanan, untuk menggantikan suksesor kiri dan suksesor kanan, serta istilah ayah untuk pengganti predesesor.

Contoh :



K misalnya adalah keturunan kanan dari D, tetapi bukan keturunan dari F, E ataupun M. Simpul G adalah ayah dari K dan L. Di sini K dan L adalah bersaudara, masing-masing anak kiri dan anak kanan dari G.

Garis yang ditarik dari Simpul N ke suksesor disebut Ruas dan sederetan ruas yang berturutan disebut Jalur atau path. Sebuah jalur yang berakhir pada daun (terminal) disebut Cabang.

Garis AD maupun GL adalah contoh ruas. Barisan ruas (AD, DG, GL) adalah jalur dari simpul A ke simpul L, jalur tersebut merupakan cabang, karena berakhir di simpul terminal (daun) L.

Dari contoh pohon binar di atas :
Root : A
Simpul Daun adalah : F, K, L dan M
Level 0 adalah simpul A
Level 1 adalah simpul D dan E
Level 2 adalah simpul F, G dan M
Level 3 adalah simpul K dan L
Ketinggian (kedalaman) = 3 + 1 = 4
Berat (bobot) = 4
Cabang (AD, DG, GK) ataupun (AD, DG, GL) mengandung simpul dengan jumlah maksimum, yakni = 4, sedangkan cabang (AE, EM) serta (AD, DF) hanya mengandung 3 simpul.

POHON BINAR LENGKAP

Setiap simpul dari pohon binar paling banyak mempunyai dua anak. Simpul akar bertingkat = 0, hanya terdiri dari 1 simpul. Anaknya bertingkat = 1 terdiri paling banyak 2 simpul, demikian seterusnya, simpul dengan tingkat = r paling banyak ada 2^r.

Suatu pohon binar T dikatakan lengkap atau complete, bila setiap tingkatnya kecuali mungkin tingkat yang terakhir, mempunyai semua simpul yang mungkin, yakni 2^r simpul untuk tingkat ke-r, dan bila semua simpul pada tingkat terakhir muncul di bagian kiri pohon.

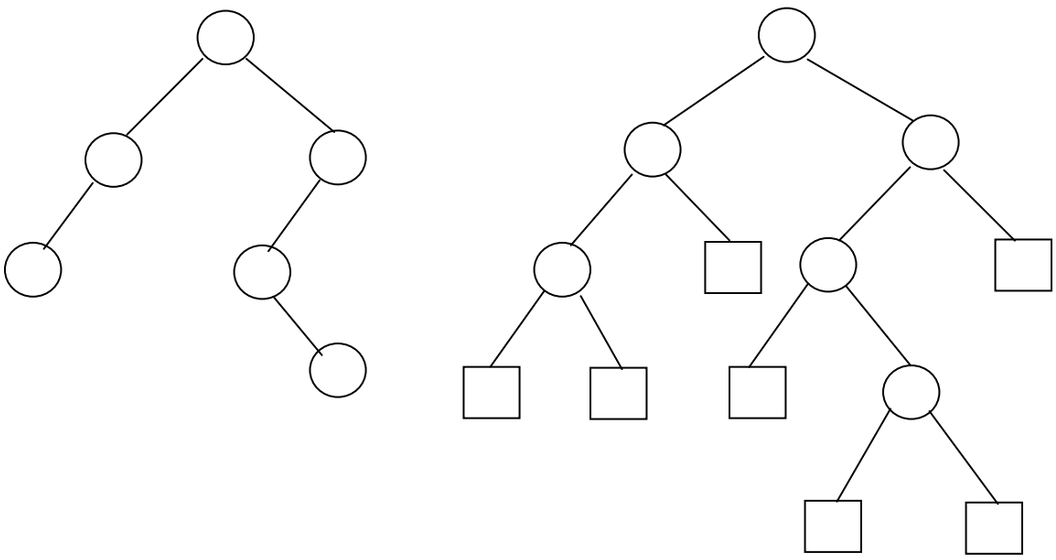
Kita dapat memberi label pohon binar lengkap menggunakan integer 1, 2, …, n dari kiri ke kanan generasi demi generasi. Hal ini untuk mempermudah kita untuk mengetahui ayah serta anak dari suatu simpul pada pohon binar lengkap.

Dalam hal ini anak kiri dari simpul K adalah 2 * K dan anak kanan dari simpul K adalah 2*K+1. Sedangkan ayah dari K adalah INT(K/2). Notasi INT(P) adalah integer terbesar yang lebih kecil atau sama dengan P. Jadi INT(3 2/3) = 3, INT(15/2) = 7, INT(4) = 4, dan sebagainya.

POHON-2

Pohon Binar T dikatakan Pohon-2 atau pohon binar yang dikembangkan bila setiap simpul mempunyai 0 atau 2 anak. Dalam kasus ini, simpul dengan 2 anak disebut simpul internal, sedangkan simpul tanpa anak disebut simpul eksternal. Dalam diagram, seringkali diadakan pembedaan antara simpul internal dan eksternal. Simpul internal digambarkan sebagai lingkaran, sedangkan simpul eksternal sebagai bujursangkar.

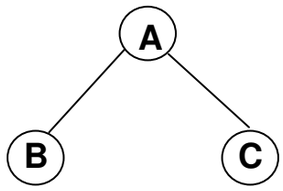
Contoh :



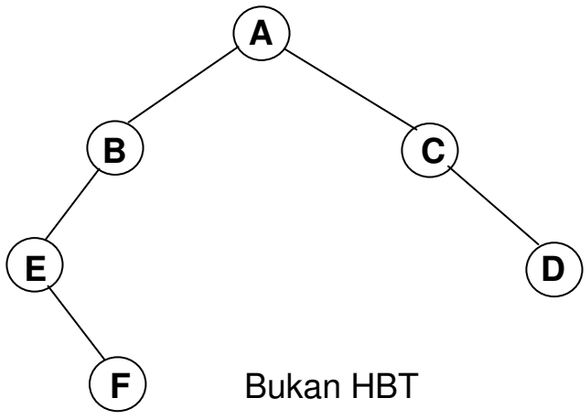
POHON KETINGGIAN SEIMBANG

Pohon binar yang mempunyai sifat bahwa ketinggian subpohon kiri dan subpohon kanan dari pohon tersebut berbeda paling banyak 1, disebut Pohon Ketinggian Seimbang atau Height Balanced Tree (HBT).

Contoh :



HBT



Bukan HBT

KETINGGIAN MINIMUM DAN MAKSIMUM POHON BINAR

Jika banyaknya simpul = N, maka :
1. Ketinggian Minimum adalah : $H_{min} = INT(^2\log N) + 1$
2. Ketinggian Maksimum adalah : N

Contoh : untuk N = 8
Ketinggian Minimum adalah : $H_{min} = INT(^2\log N) + 1$
$= INT(^2\log 8) + 1$
$= INT(^2\log 2^3) + 1$
$= INT(3) + 1$
$= 3 + 1$
$= 4$

Ketinggian Maksimum adalah : 8

PENYAJIAN POHON BINAR DALAM MEMORI

Penyajian pohon binar dalam memori dengan dua cara, yaitu :
1. Penyajian Kait (link)
2. Penyajian Sequential.

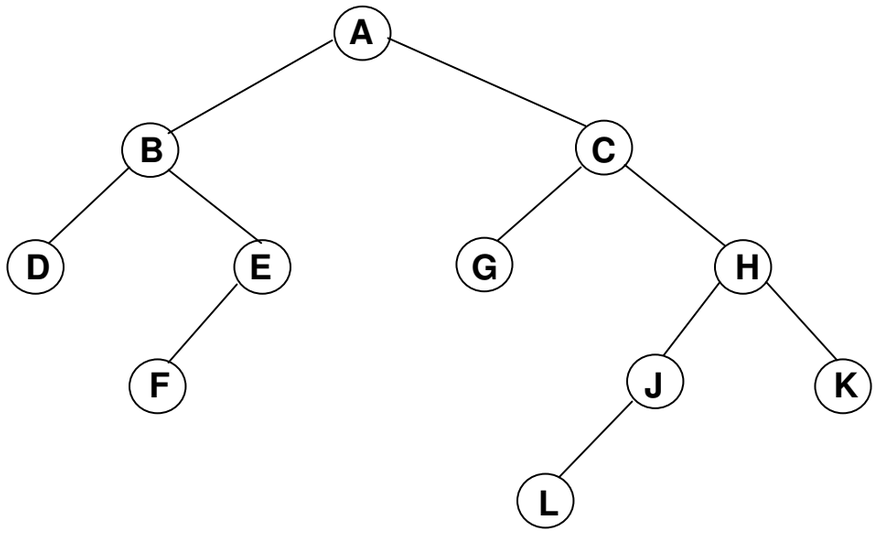
1. PENYAJIAN KAIT

Kalau tidak dinyatakan lain, suatu pohon binar T akan disimpan dalam memori secara penyajian kait.

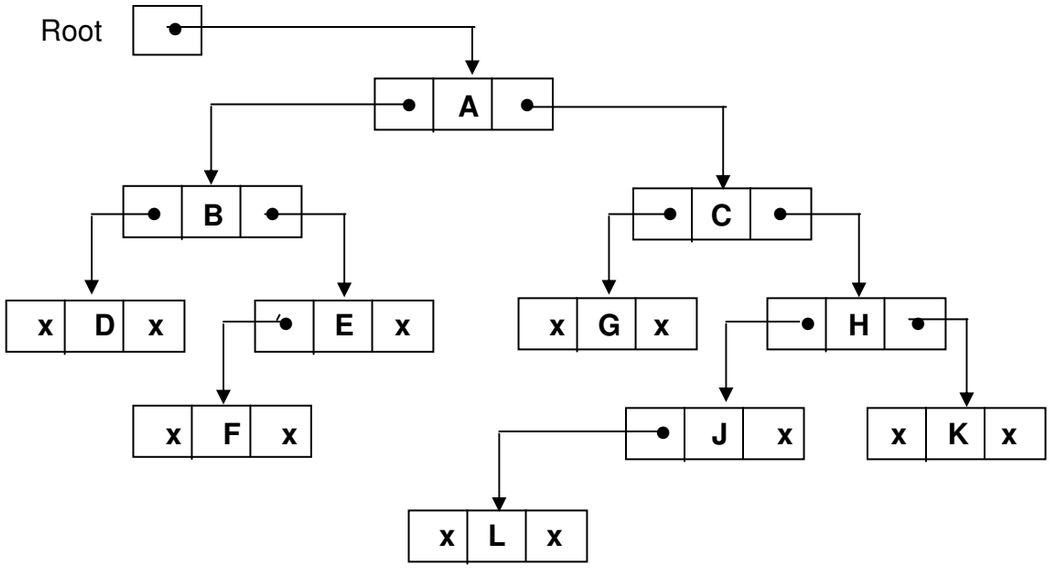
Penyajian ini menggunakan tiga array sejajar INFO, LEFT dan RIGHT, serta variabel penuding ROOT.

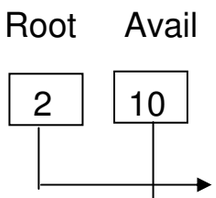
Masing-masing simpul N dari pohon T berkorespondensi dengan suatu lokasi K, sedemikan sehingga :
(1) INFO(K) berisi data pada simpul N.
(2) LEFT(K) berisi lokasi dari anak kiri simpul N.
(3) RIGHT(K) berisi lokasi dari anak kanan simpul N.

Contoh :
Pohon Binar T



Skema Penyajian Kait dari pohon binar T





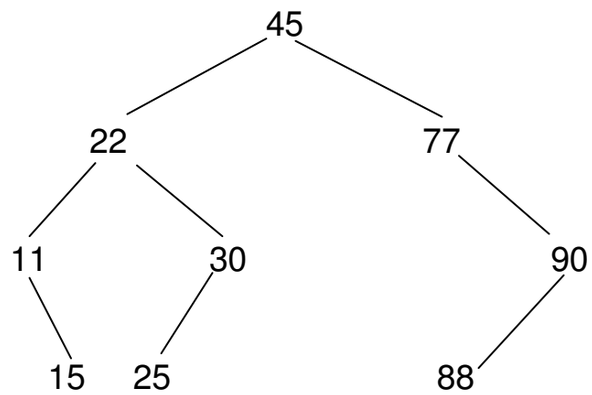
	INFO	LEFT	RIGHT
1	L	0	0
2	A	7	5
3		8	
4	G	0	0
5	C	4	17
6		13	
7	B	16	12
8		6	
9	F	0	0
10		3	
11	J	1	0
12	E	9	0
13		15	
14		19	
15		18	
16	D	0	0
17	H	11	20
18		14	
19		0	
20	K	0	0

2. PENYAJIAN SEQUENTIAL

Penyajian pada pohon binar T ini hanya menggunakan sebuah array linear tunggal TREE sebagai berikut :

1. Akar R dari pohon T tersimpan sebagai TREE[1]
2. Jika simpul N menduduki TREE[K] maka anak kirinya tersimpan dalam TREE[2*K] dan anak kanannya dalam TREE[2*K+1]

Contoh :



TREE

1	45
2	22
3	77
4	11
5	30
6	
7	90
8	
9	15
10	25
11	
12	
13	
14	88
15	
.	
.	
.	
29	

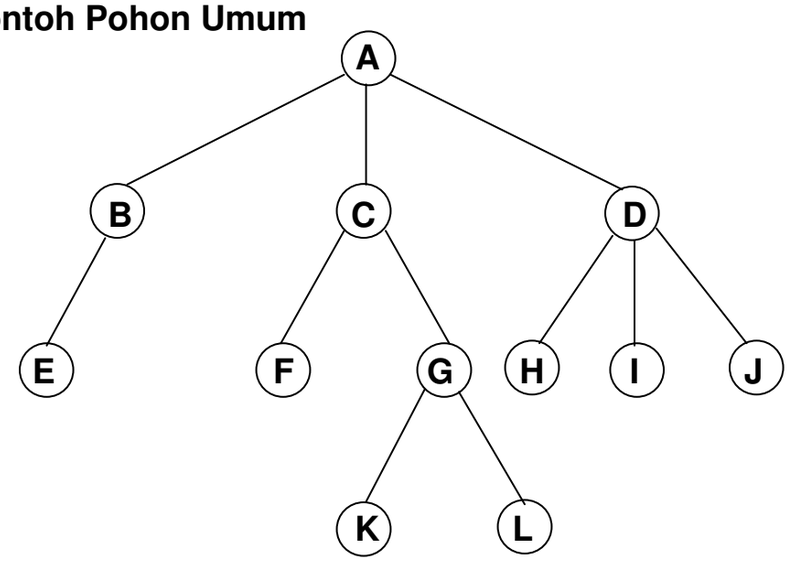
Dapat dilihat bahwa penyajian membutuhkan 14 lokasi dalam array TREE, meskipun T hanya mempunyai 9 simpul. Kenyataannya, bila kita memasukkan elemen nol sebagai suksesor dari simpul terminal, dibutuhkan TREE[29] untuk suksesor kanan dari TREE[14].

PENYAJIAN POHON UMUM SECARA POHON BINAR

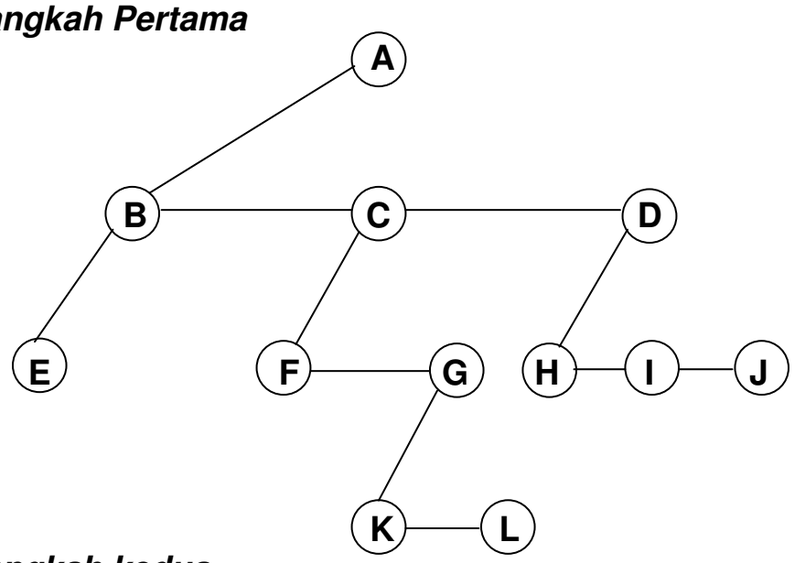
Kita dapat menyajikan pohon umum secara pohon binar dengan algoritma sebagai berikut :

1. Tambahkan ruas baru yang menghubungkan 2 simpul bersaudara yang berdampingan, lalu kita hapus ruas dari simpul ayah ke anak, kecuali ruas ke simpul anak paling kiri.
2. Rotasikan sebesar 45° , searah putaran jarum jam terhadap pohon hasil langkah pertama.

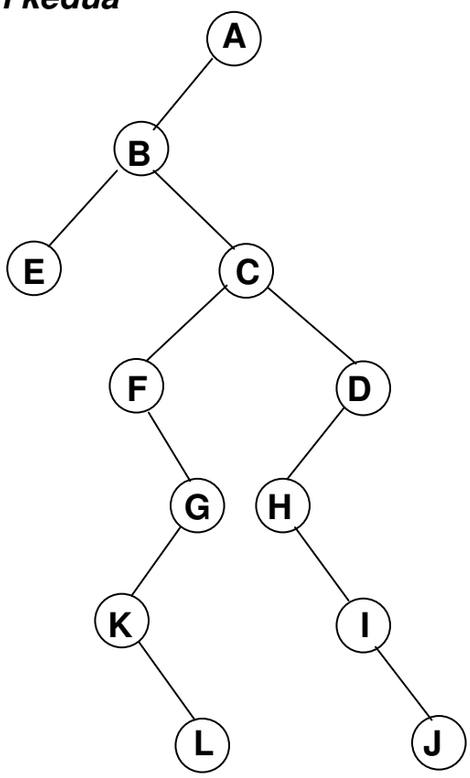
Contoh Pohon Umum



Langkah Pertama



Langkah kedua



NOTASI PREFIX, INFIX DAN POSTFIX SERTA TRAVERSAL POHON

Traversal adalah proses kunjungan dalam pohon, dengan setiap Simpul hanya dikunjungi tepat satu kali.

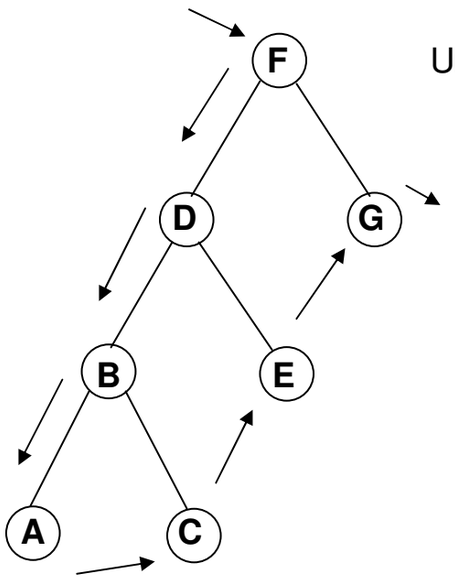
Tiga kegiatan yang terdapat dalam traversal pohon binar adalah :
1. Mengunjungi simpul akar (root)
2. Melakukan traversal subpohon kiri, dan
3. Melakukan traversal subpohon kanan

Terdapat tiga macam traversal pohon, yaitu :
1. Traversal Pre-order, dilakukan berturut-turut :
   a. Kunjungi simpul akar
   b. Lakukan traversal subpohon kiri
   c. Lakukukan traversal subpohon kanan

2. Traversal In-order, dilakukan berturut-turut :
   a. Lakukan traversal subpohon kiri
   b. Kunjungi simpul akar
   c. Lakukan traversal subpohon kanan

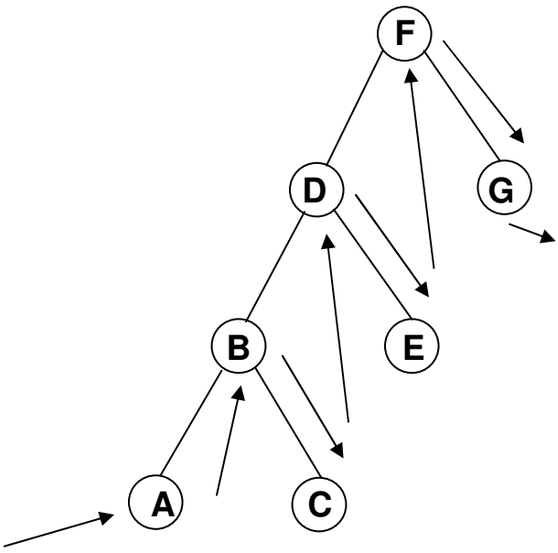
3. Traversal Post-order, dilakukan berturut-turut :
   a. Lakukan traversal subpohon kiri
   b. Lakukan traversal subpohon kanan
   c. Kunjungi simpul akar

Contoh :



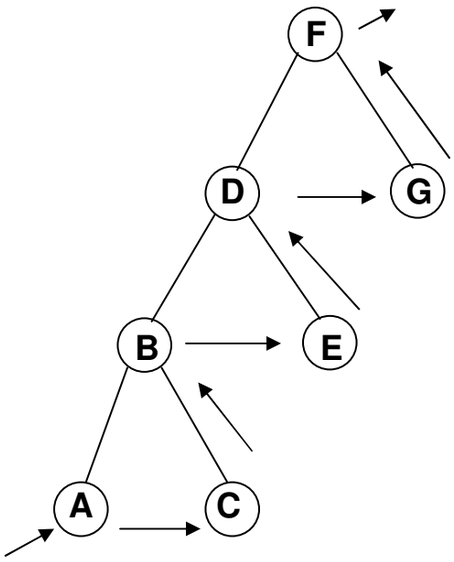
Untai yang dihasilkan secara Pre-order

FDBACEG



Untai yang dihasilkan secara In-order

ABCDEFG



Untai yang dihasilkan secara Post-order

ACBEDGF

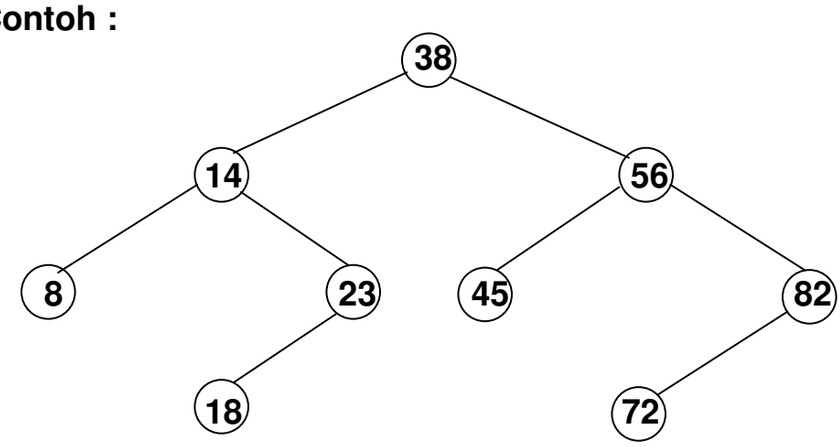
POHON CARI BINAR

Pandang T suatu pohon binar, maka T disebut pohon cari binar (pohon terurut binar) bila masing-masing simpul N dari T mempunyai sifat :

Nilai dari N selalu lebih besar dari setiap nilai simpul pada subpohon kiri dari N, dan selalu lebih kecil dari setiap nilai simpul pada subpohon kanan dari N.

Definisi di atas menjamin bahwa traversal ***in-order*** terhadap pohon cari binar selalu menghasilkan ***untai terurut***.

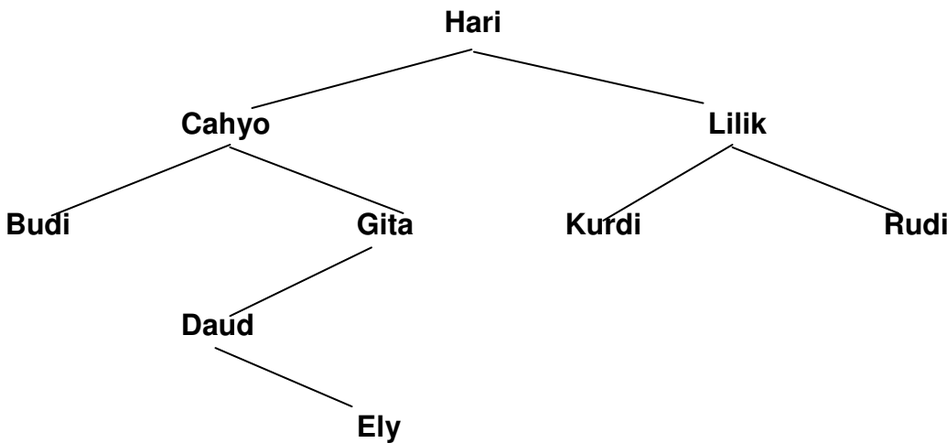
Contoh :

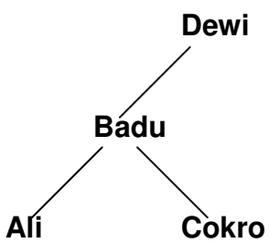


Dicatat bahwa dalam definisi Pohon Cari Binar di atas, semua nilai/label simpul adalah berbeda. Kalau diijinkan terjadi adanya label yang sama, dapat dilakukan sedikit modifikasi.

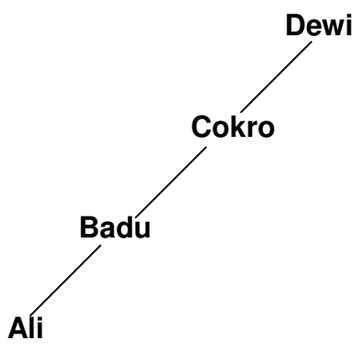
Definisi Pohon Cari Binar menjadi : *Bila N adalah simpul dari Pohon maka nilai semua simpul pada subpohon kiri dari N adalah lebih kecil atau sama dengan nilai simpul N, dan nilai semua simpul pada subpohon kanan dari N adalah lebih besar dari nilai simpul N.*

Contoh :

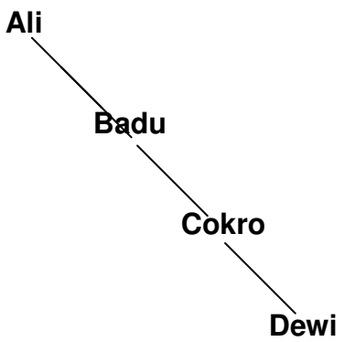




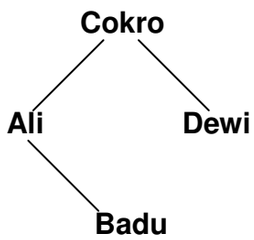
(a)



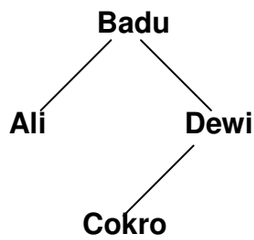
(b)



(c)



(d)



(e)

Buatlah 10 pohon cari binar yang berbeda dari (a, b, c, d, e) di atas!

CARI DAN PENYISIPAN SIMPUL POHON CARI BINAR

Pandang bahwa diberikan sebuah ITEM informasi. Algoritma di bawah ini akan menemukan lokasi dari ITEM dalam Pohon Cari Binar T, atau menyelipkan ITEM dalam Pohon Cari Binar T, atau menyelipkan ITEM sebagai simpul baru dari T dalam posisi yang tepat.

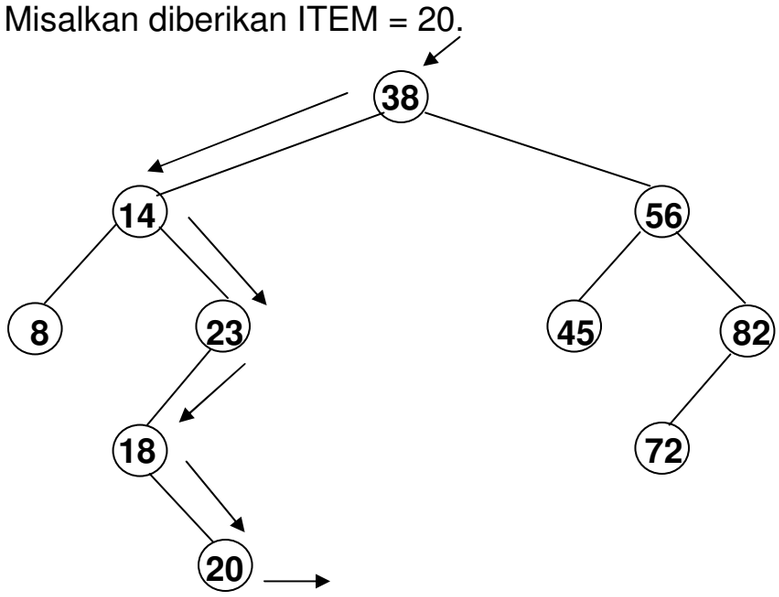
ALGORITMA

Algoritma bekerja sebagai berikut :

(a) Bandingkan ITEM dengan simpul akar N dari Pohon, jika ITEM < N, proses subpohon kiri dari N, jika ITEM > N proses subpohon kanan dari N.

(b) Ulangi langkah (a) sampai dari hal berikut ditemui :
   (1) Ditemukan simpul n sedemikian sehingga ITEM = N, dalam hal ini pencarian berhasil.
   (2) Dijumpai subpohon hampa, ini menunjukkan bahwa pencarian tidak berhasil, dan kita selipkan ITEM mengisi subpohon yang hampa tadi.

Contoh : Misalkan diberikan ITEM = 20.



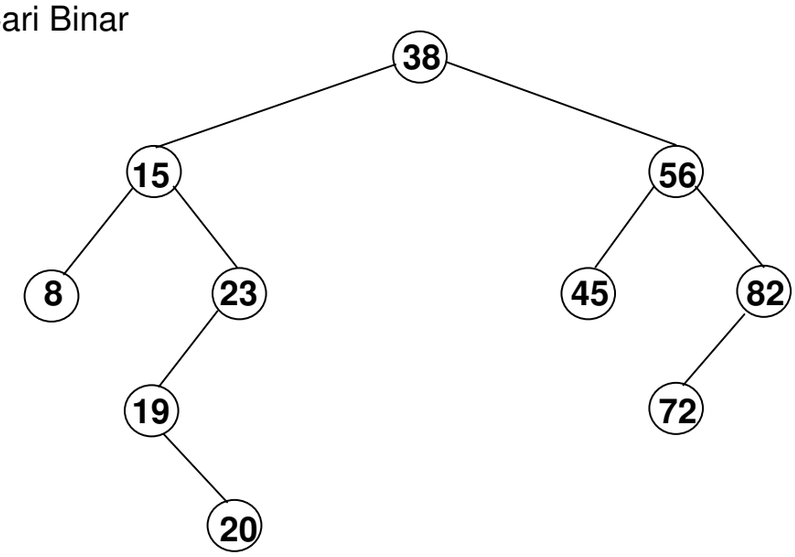
PENGHAPUSAN SIMPUL POHON CARI BINAR

Operasi penghapusan suatu simpul dari Pohon Cari Binar bukan merupakan hal yang mudah. Pertama-tama kita harus tetapkan lebih dahulu simpul yang akan kita hapus, bila merupakan simpul daun, maka proses akan berlangsung dengan mudah, karena simpul daun tersebut akan dapat langsung kita hapuskan dari Pohon Cari yang bersangkutan. Jika simpul yang akan dihapuskan mempunyai sebuah subpohon kanan, maka untuk menggantikan posisi simul yang dihapuskan tersebut, kita ambil simpul akar subpohon kiri dan sebaliknya. Jika simpul yang akan dihapuskan mempunyai dua buah subpohon yaitu subpohon kiri dan subpohon kanan, maka untuk menggantikan posisi dari simpul yang dihapuskan tersebut, kita tentukan simpul dari salah satu subpohon kiri atau subpohon kanan sedemikian sehingga bangun pohon yang terbentuk kembali, memenuhi sifat sebagai Pohon Cari Binar.

PROSEDUR untuk penghapusan suatu simpul dari Pohon Cari Binar

(1) Jika Pohon Hampa, maka penghapusan yang dilakukan gagal. Berhenti.
(2) Jika n < Ro (akar), subpohon kiri dari Ri diselidiki sampai ditemukan simpul yang telah ditentukan untuk dihapus.
(3) Jika n > Ro, maka subpohon kanan dari Ri diselidiki sampai ditemukan simpul yang telah ditentukan untuk dihapus.
(4) Jika n = Ro, subpohon kiri dan subpohon kanan hampa, maka hapus Ro.
(5) Jika n = Ro, dan subpohon kirinya hampa, maka hapus Ro, kemudian ambil akar dari subpohon kanan untuk menggantikan posisi Ro. Pohon baru akan memenuhi sifat sebagai Pohon Cari lagi.
(6) Jika n = Ro, dan subpohon kanannya hampa, maka hapus Ro, kemudian ambil akar dari subpohon kiri untuk menggantikan posisi Ro. Pohon baru akan memenuhi sifat sebagai Pohon Cari lagi.
(7) Jika n = Ro, subpohon kanan dan subpohon kiri tidak hampa, maka untuk menggantikan posisi Ro yang dihapus, kita tentukan suatu simpul mungkin dari subpohon kiri atau mungkin dari subpohon kanan, sedemikian sehingga Pohon yang terbentuk kembali memenuhi sifat sebagai Pohon Cari lagi.

Pohon Cari Binar



1. Dari pohon cari binar di atas, lakukan penghapusan simpul 56!

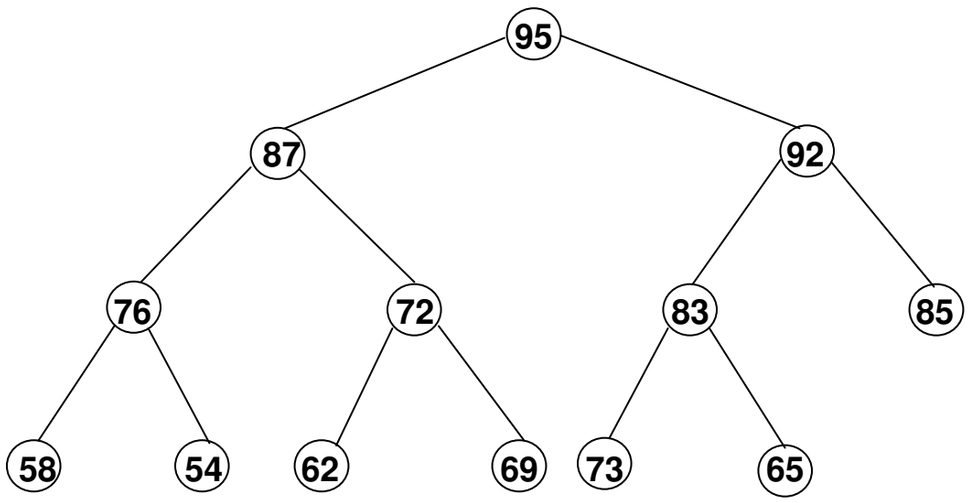
2. Dari pohon cari binar di atas, lakukan penghapusan berturut-turut terhadap simpul 45, 23 dan 56!

3. Bila diketahui data : 44, 55, 12, 42, 94, 18, 7, 67
   Buatlah pohon cari binar dari data di atas!
   Cari dan sisipkan : 38 terhadap pohon cari binar yang terbentuk!

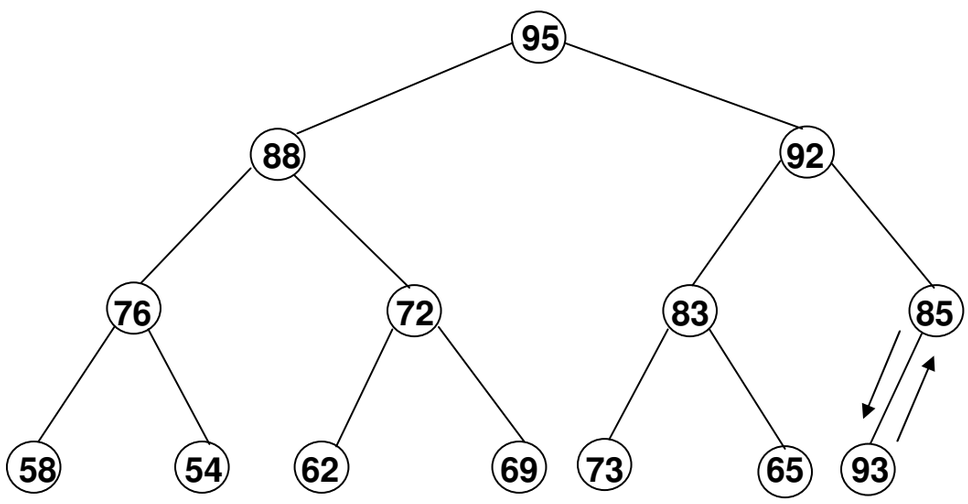
HEAP

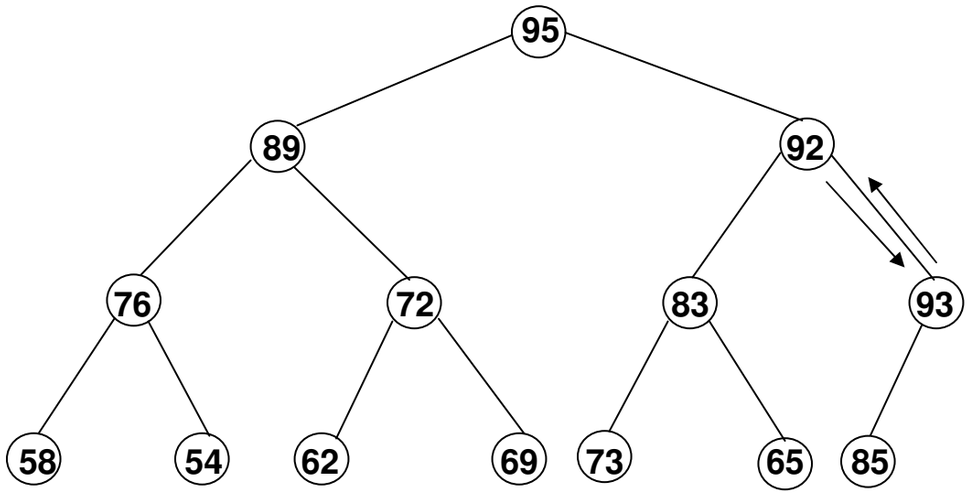
Suatu pohon binar adalah *Heap*, jika *nilai setiap simpul lebih besar atau sama dengan nilai anaknya*, disebut ***Maxheap***.
Kalau *nilai setiap simpul lebih kecil atau sama dengan nilai anaknya*, disebut ***Minheap***.

Contoh :

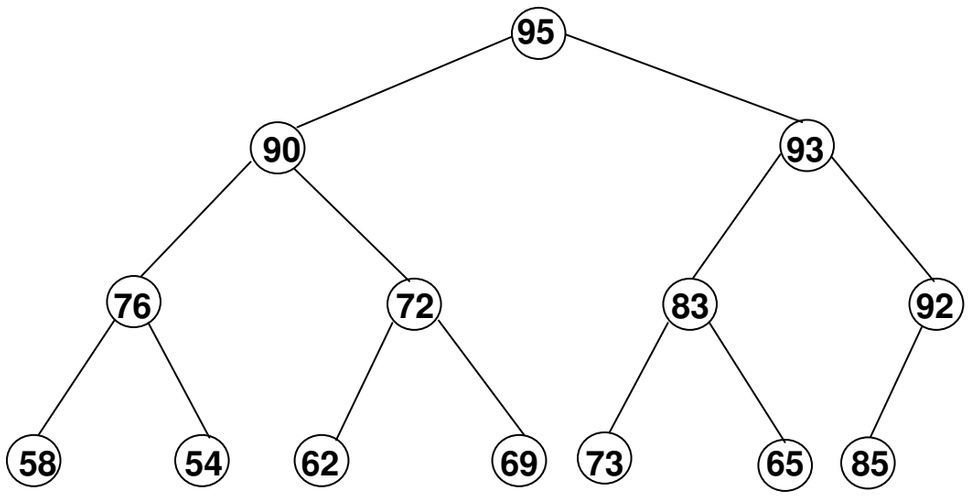


Kita dapat memasukkan elemen baru ke dalam sebuah Heap. Misalkan elemen = 93. Mula-mula elemen tersebut diletakkan sebagai elemen terakhir dari heap, yaitu elemen tree[14] pada daftar berurutan. Selanjutnya memerika apakah elemen tersebut lebih besar dari ayahnya. Bila lebih besar, lakukan pertukaran, elemen tersebut sekarang menjadi ayah. Demikian seterusnya sampai kondisi ini tidak tercapai.





Sehingga heap yang terbentuk :



Kemudian lakukan heapsort!

Algoritma heapsort :

1. Jatuhkan root ke posisi terakhir dari simpul dalam urutan almost complete. (Pertukarkan antara root dengan simpul terakhir dari urutan almost complete. Data pada simpul terakhir sudah terurut)
2. Sisanya bentuk heap kembali, kemudian lakukan sort kembali.

Buatlah maxheap dari data berikut ini :
45, 22, 77, 11, 30, 90, 15, 25, 88.
Tambahkan 35 ke dalam maxheap tersebut, kemudian lakukan heapsort!