STRUKTUR DATA

BINARY TREE / POHON BINER

Rosa Ariani Sukamto

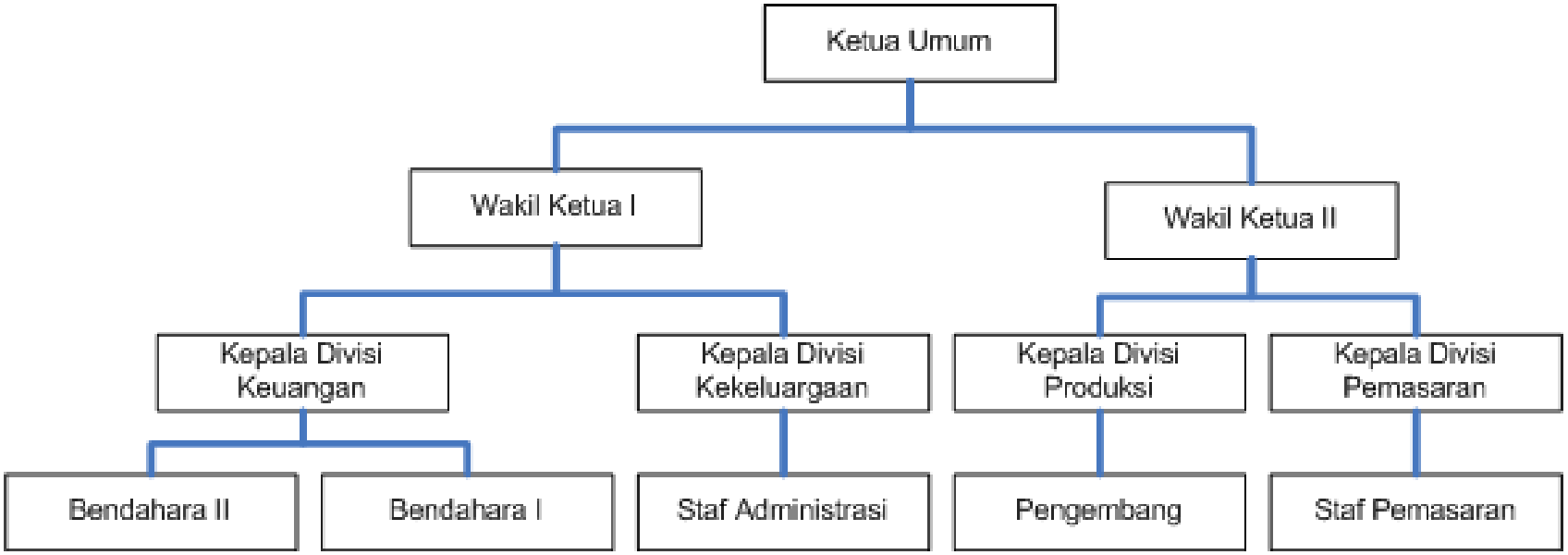
ROSA ARIANI SUKAMTO

Blog: http://hariiniadalahhadiah.wordpress.com

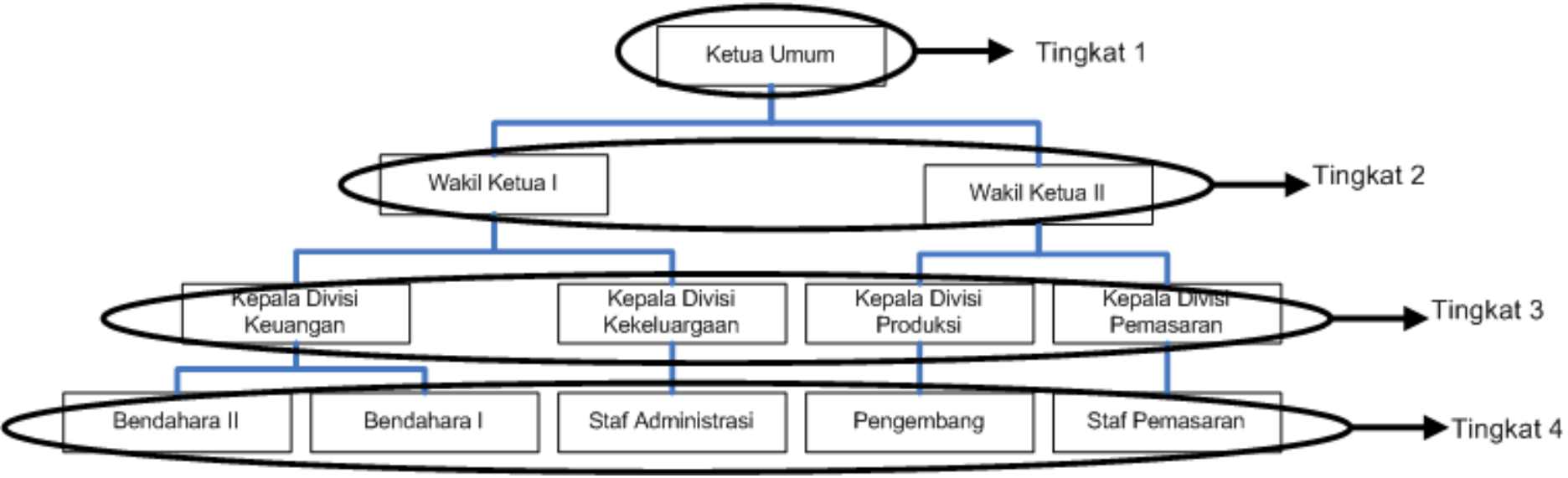
Facebook: https://www.facebook.com/rosa.ariani.sukamto

Email: rosa_if_itb_01@yahoo.com

POHON (1)

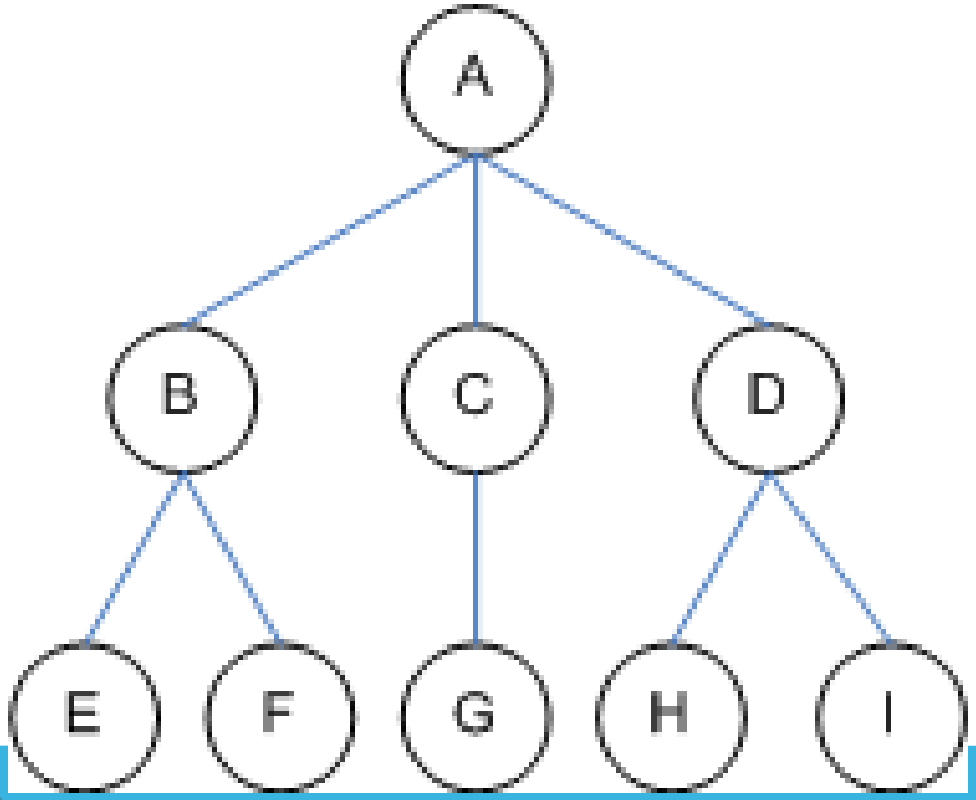

Ketua Umum
Wakil Ketua I
Wakil Ketua II
Kepala Divisi Keuangan
Kepala Divisi Kekeluargaan
Kepala Divisi Produksi
Kepala Divisi Pemasaran
Bendahara II
Bendahara I
Staf Administrasi
Pengembang
Staf Pemasaran

POHON (2)


Ketua Umum
Tingkat 1
Wakil Ketua I
Wakil Ketua II
Tingkat 2
Kepala Divisi Keuangan
Kepala Divisi Kekeluargaan
Kepala Divisi Produksi
Kepala Divisi Pemasaran
Tingkat 3
Bendahara II
Bendahara I
Staf Administrasi
Pengembang
Staf Pemasaran
Tingkat 4

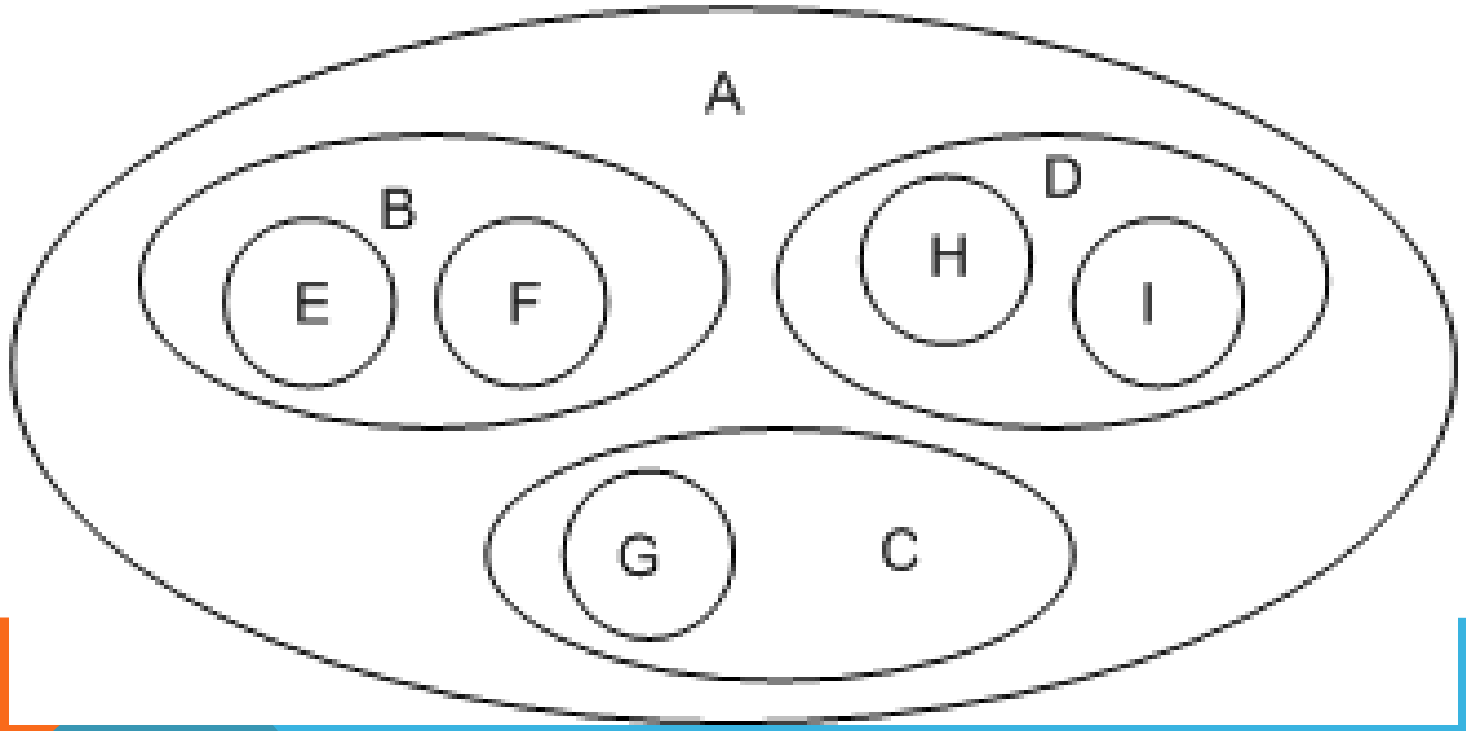
PENGGAMBARAN POHON (1)

Grafik



PENGGAMBARAN POHON (2)

Diagram Venn



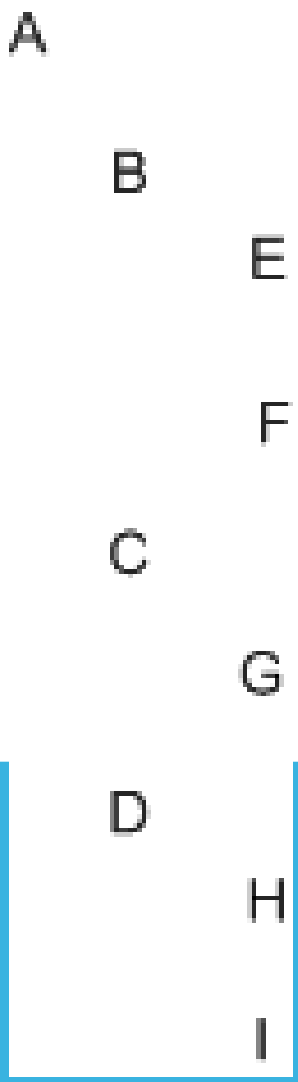
PENGGAMBARAN POHON (3)

Notasi Kurung

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

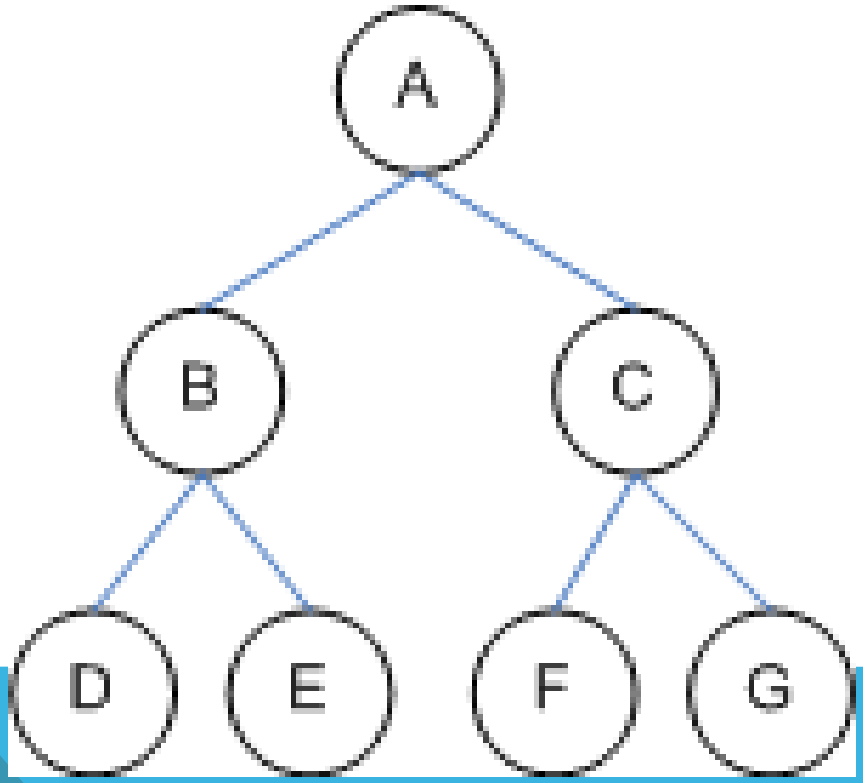
PENGGAMBARAN POHON (4)

Indentasi



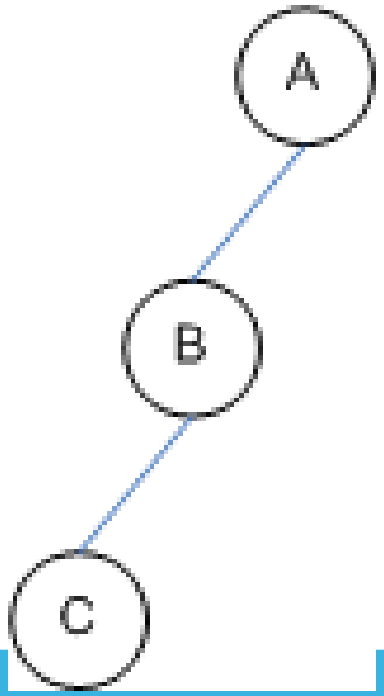
POHON BINER (1)

Pohon Biner Lengkap



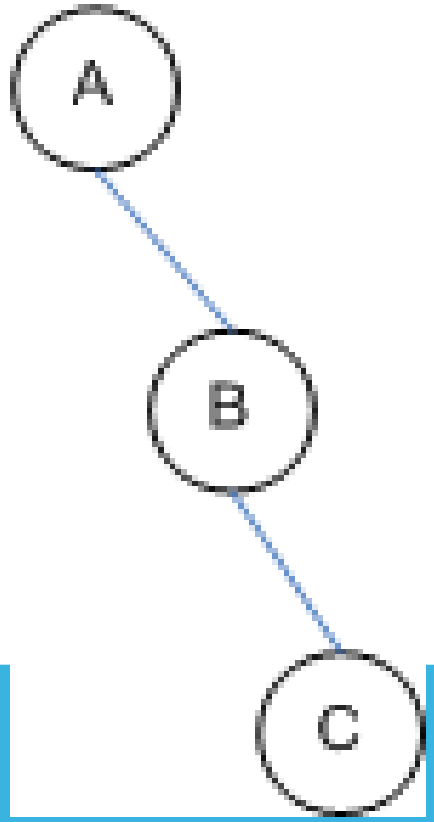
POHON BINER (2)

Pohon Biner Condong Kiri (*left skewed binary tree*)



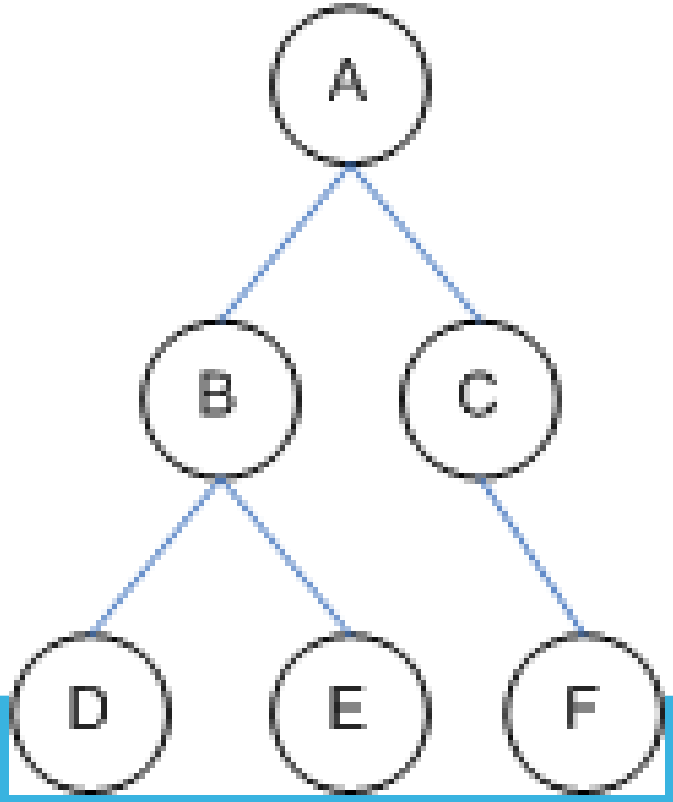
POHON BINER (3)

Pohon Biner Condong Kanan (*right skewed binary tree*)



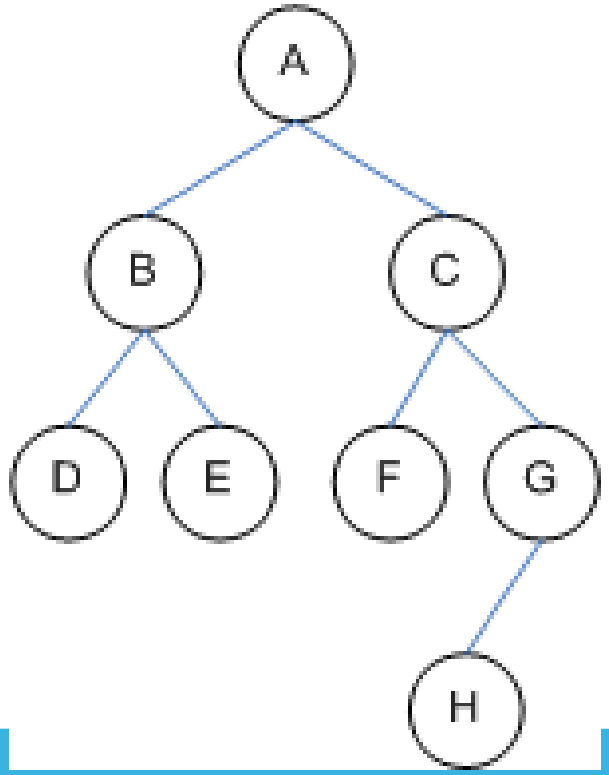
POHON BINER (4)

Pohon Biner Sembarang



OPERASI KUNJUNGAN POHON BINER (1)

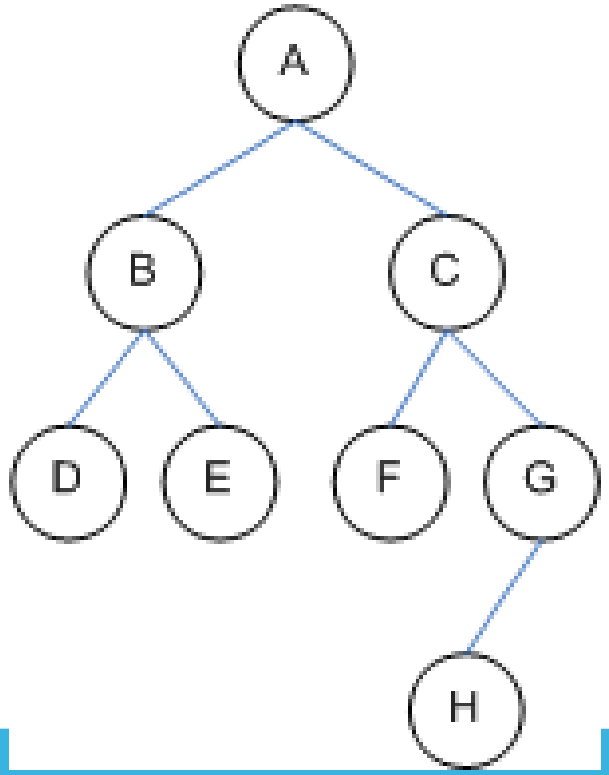
PreOrder : Kunjungan dari akar, kemudian pohon kiri, lalu pohon kanan



A – B – D – E – C – F – G – H

OPERASI KUNJUNGAN POHON BINER (2)

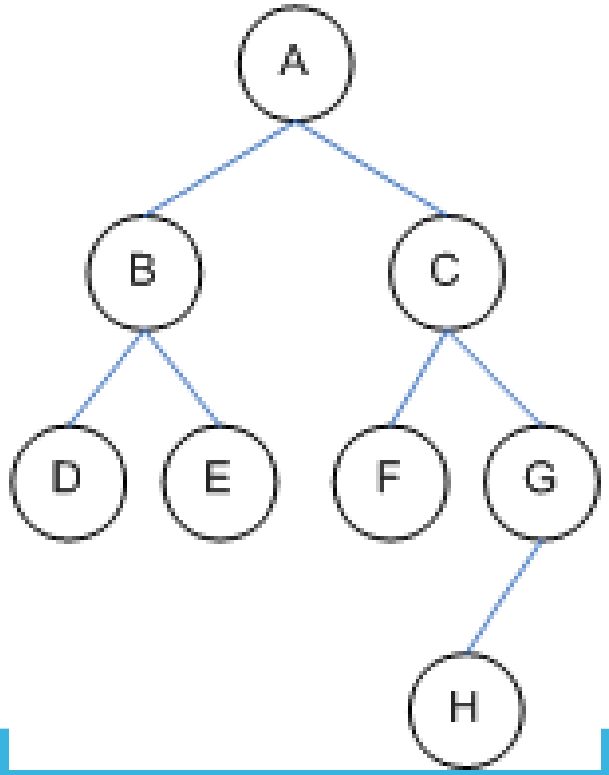
InOrder : Kunjungan dari pohon kiri, kemudian akar, lalu pohon kanan



D – B – E – A – F – C – H – G

OPERASI KUNJUNGAN POHON BINER (3)

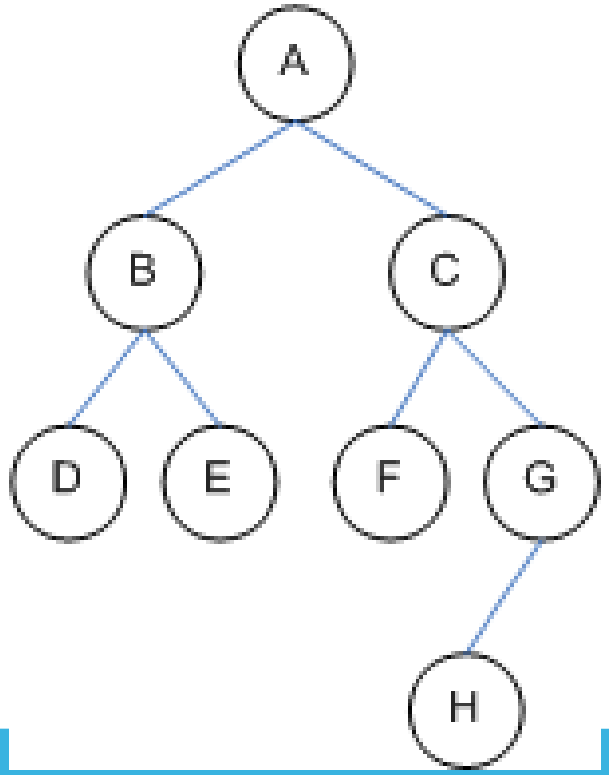
PostOrder : Kunjungan dari pohon kiri, kemudian kanan, lalu akar



D – E – B – F – H – G – C – A

OPERASI KUNJUNGAN POHON BINER (4)

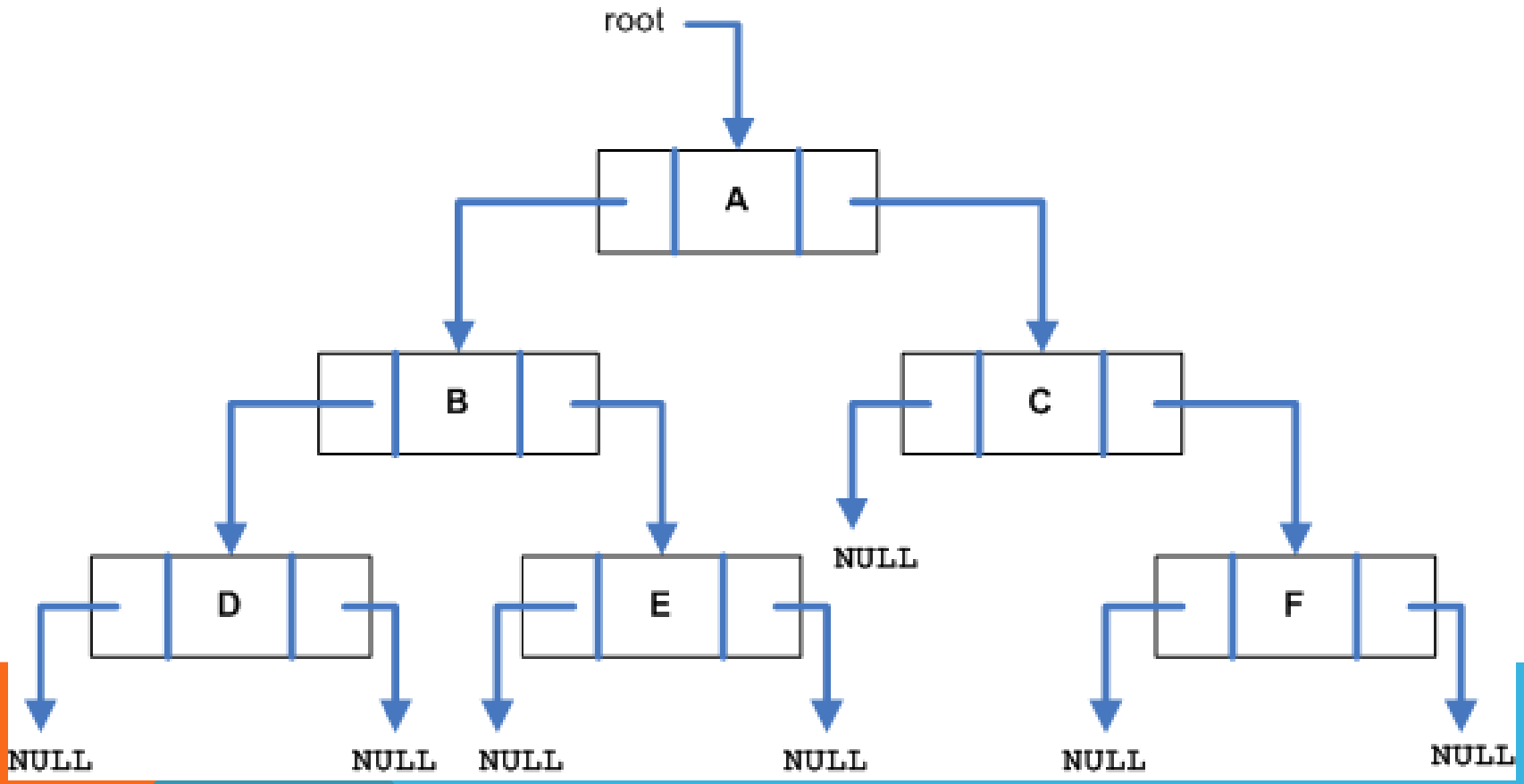
LevelOrder : Kunjungan per tingkatan bagian pohon



A – B – C – D – E – F – G – H

IMPLEMENTASI POHON BINER (1) - ELEMEN

IMPLEMENTASI POHON BINER (2) - POHON



DEKLARASI ELEMEN DAN INISIALISASI

```
#include <stdio.h>
#include <malloc.h>

typedef struct smp *alamatsimpul;
typedef struct smp{
  char info;
  alamatsimpul right;
  alamatsimpul left;
}simpul;

typedef struct{
  simpul* root;
}tree;
```

```
void makeTree(char c, tree *T){

  simpul *node;
  node = (simpul *) malloc
   (sizeof (simpul));
  node->info = c;
  node->right = NULL;
  node->left = NULL;
  (*T).root = node;

}
```

ADDRIGHT DAN ADDLEFT

```
void addRight(char c, simpul
   *root){



  if(root->right == NULL){
    /*jika sub pohon kanan
   kosong*/
    simpul *node;
    node = (simpul *) malloc
   (sizeof (simpul));
    node->info = c;
    node->right = NULL;
    node->left = NULL;
    root->right = node;
  }
  else{
    printf("sub pohon kanan telah
   terisi \n");
  }
}
```

```
void addLeft(char c, simpul
   *root){



  if(root->left == NULL){
    /*jika sub pohon kiri
   kosong*/
    simpul *node;
    node = (simpul *) malloc
   (sizeof (simpul));
    node->info = c;
    node->right = NULL;
    node->left = NULL;
    root->left = node;
  }
  else{
    printf("sub pohon kiri telah
   terisi \n");
  }
}
```

DELRIGHT DAN DELLEFT

```
void delRight(simpul *root){


  simpul *node = root->right;
  root->right = NULL;
  free(node);

}
```

```
void delLeft(simpul *root){


  simpul *node = root->left;
  root->left = NULL;
  free(node);

}
```

PREORDER DAN INORDER

```
void printTreePreOrder(simpul
   *root){



  if(root != NULL){
    printf(" %c ", root->info);
    printTreePreOrder(root-
   >left);
    printTreePreOrder(root-
   >right);
  }


}
```

```
void printTreeInOrder(simpul
   *root){



  if(root != NULL){
    printTreeInOrder(root->left);
    printf(" %c ", root->info);
    printTreeInOrder(root-
   >right);
  }


}
```

POSTORDER DAN COPYTREE

```
void printTreePostOrder(simpul
   *root){


  if(root != NULL){
    printTreePostOrder(root-
   >left);
    printTreePostOrder(root-
   >right);
    printf(" %c ", root->info);
  }


}
```

```
void copyTree(simpul *root1,
   simpul *root2){


  if(root1 != NULL){
    root2 = (simpul *) malloc
   (sizeof (simpul));
    root2->info = root1->info;
    if(root1->left != NULL){
      copyTree(root1->left,
   root2->left);
    }
    if(root1->right != NULL){
      copyTree(root1->right,
   root2->right);
    }
  }


}
```

ISEQUAL

```
int isEqual(simpul *root1, simpul
   *root2){

  int hasil = 1;

  if((root1 != NULL)&&
    (root2 != NULL)){
    if(root1->info !=
        root2->info){
      hasil = 0;
    }
    else{
      isEqual(root1->left,
        root2->left);
      isEqual(root1->right,
        root2->right);
    }
  }
  else{
    if((root1 != NULL)||
        (root2 != NULL)){
      hasil = 0;
    }
  }

  return hasil;

}
```

MAIN

```
int main(){
  tree T;
  makeTree('A', &T);
  addLeft('B', T.root);
  addRight('C', T.root);
  addLeft('D', T.root->left);
  addRight('E', T.root->left);
  addRight('F', T.root->right);

  printf("================\n");
  printf("preOrder\n");
  printTreePreOrder(T.root);
  printf("\n================\n");
  printf("inOrder\n");
  printTreeInOrder(T.root);
  printf("\n================\n");
  printf("postOrder\n");
  printTreePostOrder(T.root);
  printf("\n================\n");
```

```
   tree T2;

  copyTree(T.root, T2.root);
  if(isEqual(T.root, T2.root) == 1){

    printf("pohon sama\n");
  }
  else{
    printf("pohon tidak sama\n");
  }

  delRight(T.root->left);
  delLeft(T.root->left);
  printf("================\n");
  printf("preOrder setelah
    dihapus\n");
  printTreePreOrder(T.root);
  printf("\n================\n");

  return 0;
}
```

DAFTAR PUSTAKA

S, Rosa A. dan M. Shalahuddin. 2010. Modul Pembelajaran: Struktur Data. Modula: Bandung.