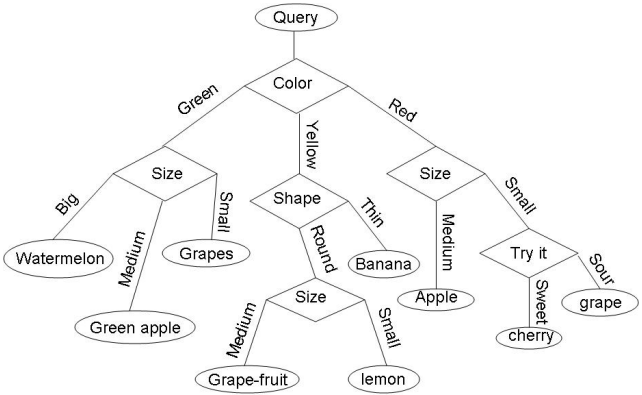
Today: Trees

Computer Programming 143 – Lecture 29
Dynamic Data Structures III

Faculty of Engineering

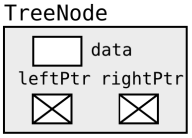
Stellenbosch
UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

Informal tree introduction



12.7 Trees II

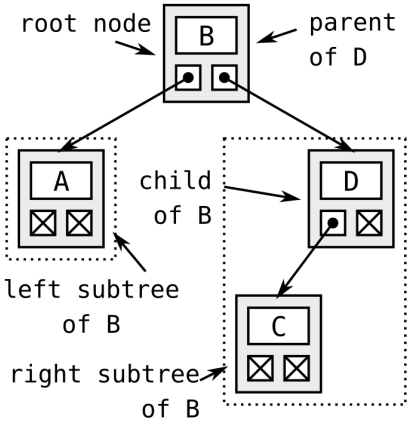
```
struct treeNode {
  struct treeNode *leftPtr;
  int data;
  struct treeNode *rightPtr;
};
typedef struct treeNode TreeNode;
typedef TreeNode *TreeNodePtr;
```



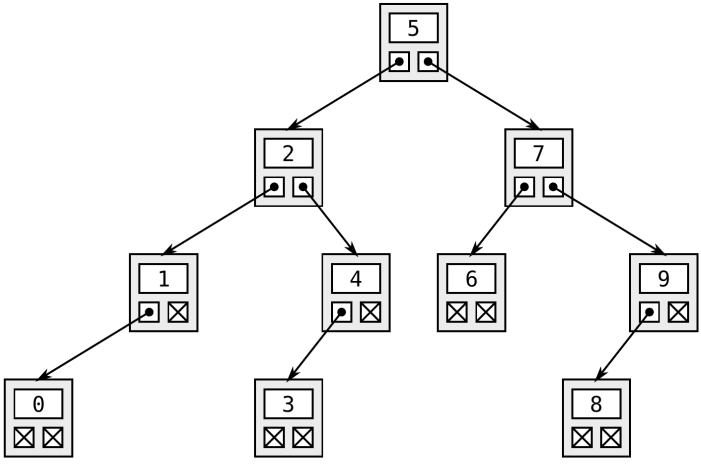
Use of trees

- Efficient search and sorting algorithms
- Store and evaluation of mathematical expressions

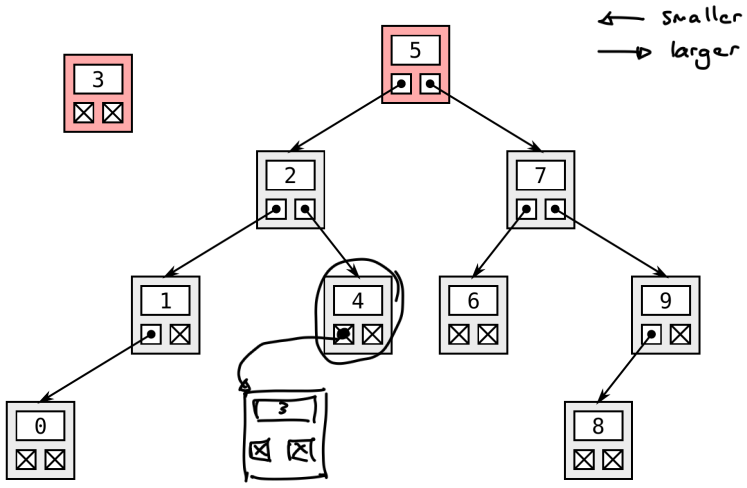
Tree concepts



12.7 Sorted Binary Trees II



12.7 Insertion in Sorted Binary Trees



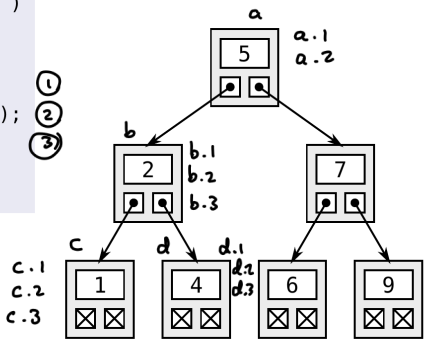
12.7 Traversal of Sorted Binary Trees I

Order of traversal of trees

- For some applications, all the nodes of the tree need to be visited
- The order in which the nodes are visited vary and may include:
  - Inorder: at each node,
    1. Traverse the node's left subtree
    2. Process the node value
    3. Traverse the node's right subtree
  - Preorder: at each node,
    1. Process the node value
    2. Traverse the node's left subtree
    3. Traverse the node's right subtree
  - Postorder: at each node,
    1. Traverse the node's left subtree
    2. Traverse the node's right subtree
    3. Process the node value
- Easily implemented using recursive functions

12.7 Inorder Traversal of Trees

```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );     (1)
    printf( "%3d", treePtr->data );  (2)
    inOrder( treePtr->rightPtr );    (3)
  }
}
```

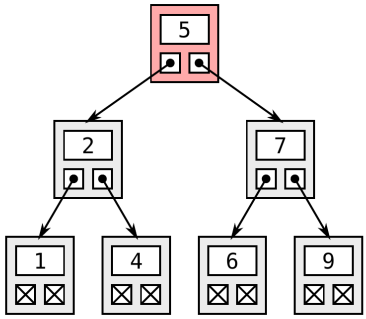


Output

1 2 4 5

12.7 Inorder Traversal of Trees

```
void inOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    inOrder( treePtr->leftPtr );
    printf( "%3d", treePtr->data );
    inOrder( treePtr->rightPtr );
  }
}
```



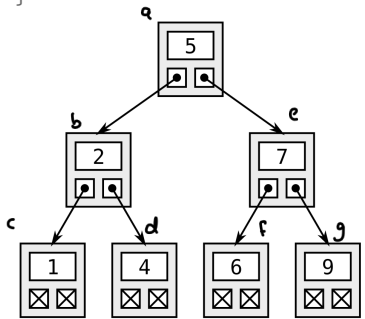
Output

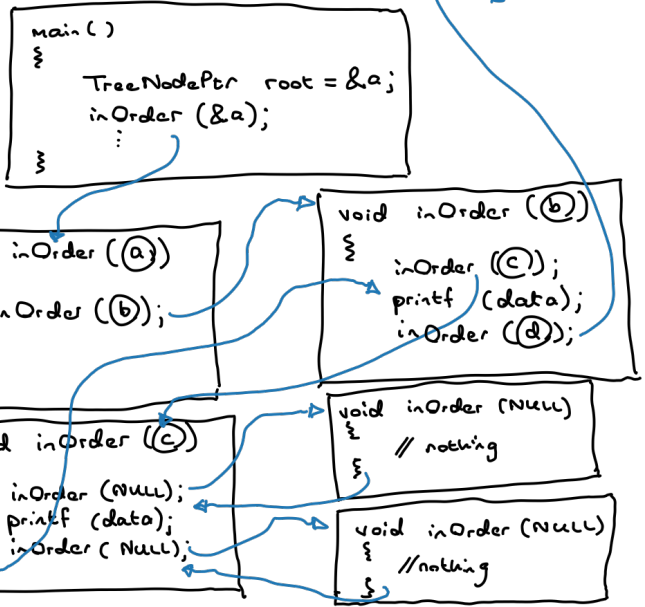
```
1 2 4 5 6 7 9
```

1 2 4 5 6 7 9

Inorder traversal of trees

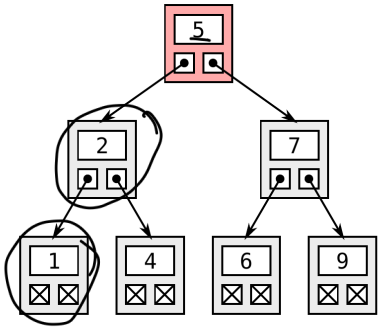
```
void inOrder(TreeNodePtr treePtr)
{
    if (treePtr != NULL)
    {
        inOrder(treePtr->leftPtr);
        printf("%3d", treePtr->data);
        inOrder(treePtr->rightPtr);
    }
}
```





12.7 Preorder Traversal of Trees

```
void preOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    printf( "%3d", treePtr->data );
    preOrder( treePtr->leftPtr );
    preOrder( treePtr->rightPtr );
  }
}
```

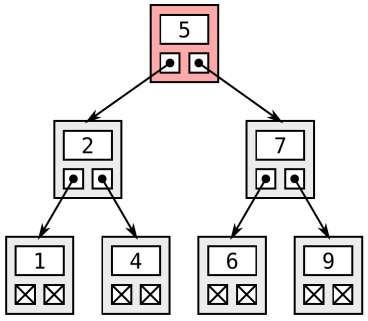


Output

5 2 1 4 7 6 9

12.7 Postorder Traversal of Trees

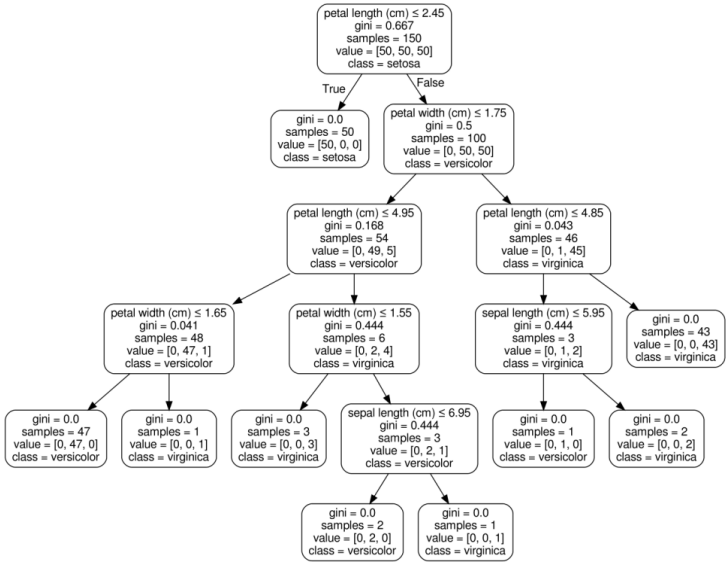
```
void postOrder( TreeNodePtr treePtr )
{
  if ( treePtr != NULL ) {
    postOrder( treePtr->leftPtr );
    postOrder( treePtr->rightPtr );
    printf( "%3d", treePtr->data );
  }
}
```



Output

```
1 4 2 6 9 7 5
```

Examples of trees


petal length (cm) ≤ 2.45
gini = 0.667
samples = 150
value = [50, 50, 50]
class = setosa
True
False
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm) ≤ 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
petal length (cm) ≤ 4.95
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
petal length (cm) ≤ 4.85
gini = 0.043
samples = 46
value = [0, 1, 45]
class = virginica
petal width (cm) ≤ 1.65
gini = 0.041
samples = 48
value = [0, 47, 1]
class = versicolor
petal width (cm) ≤ 1.55
gini = 0.444
samples = 6
value = [0, 2, 4]
class = virginica
sepal length (cm) ≤ 5.95
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 43
value = [0, 0, 43]
class = virginica
gini = 0.0
samples = 47
value = [0, 47, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
sepal length (cm) ≤ 6.95
gini = 0.444
samples = 3
value = [0, 2, 1]
class = versicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

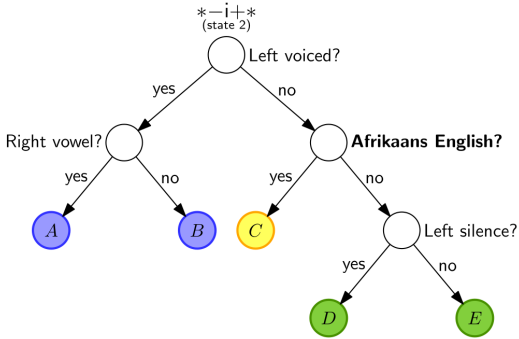


Figure 4.4: An example of a multi-accent decision-tree, taking into account the accent of the clustered triphone HMM states as well as the phonetic character of the left and right contexts.