Lecture No.11

Data Structures

Code for Simulation

```
// print the final avaerage wait time.

double avgWait = (totalTime*1.0)/count;
cout << "Total time:     " << totalTime << endl;
cout << “Customer:       " << count << endl;
cout  << "Average wait: " << avgWait << endl;
```

Priority Queue

```
#include "Event.cpp"
#define PQMAX 30

class PriorityQueue {
public:
   PriorityQueue() {
        size = 0; rear = -1;
   };
   ~PriorityQueue() {};

   int full(void)
   {
        return ( size == PQMAX ) ? 1 : 0;
   };
```

Priority Queue

```
Event* remove()
{
    if( size > 0 ) {
          Event* e = nodes[0];
           for(int j=0; j < size-2; j++ )
                  nodes[j] = nodes[j+1];

           size = size-1; rear=rear-1;
           if( size == 0 ) rear = -1;

           return e;
    }
    return (Event*)NULL;
    cout << "remove - queue is empty." << endl;
};
```

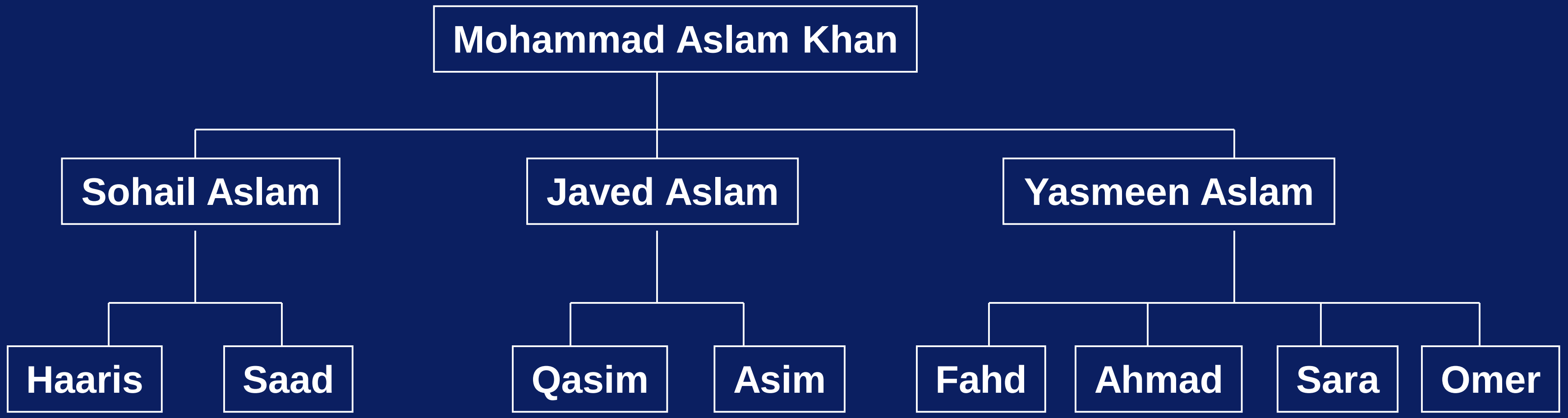
Priority Queue

```
   int insert(Event* e)
   {
       if( !full() ) {
             rear = rear+1;
             nodes[rear] = e;
             size = size + 1;
             sortElements(); // in ascending order
             return 1;
       }
       cout << "insert queue is full." << endl;
       return 0;
   };

   int length() { return size; };
};
```

Tree Data Structures

- There are a number of applications where linear data structures are not appropriate.
- Consider a genealogy tree of a family.



Tree Data Structure

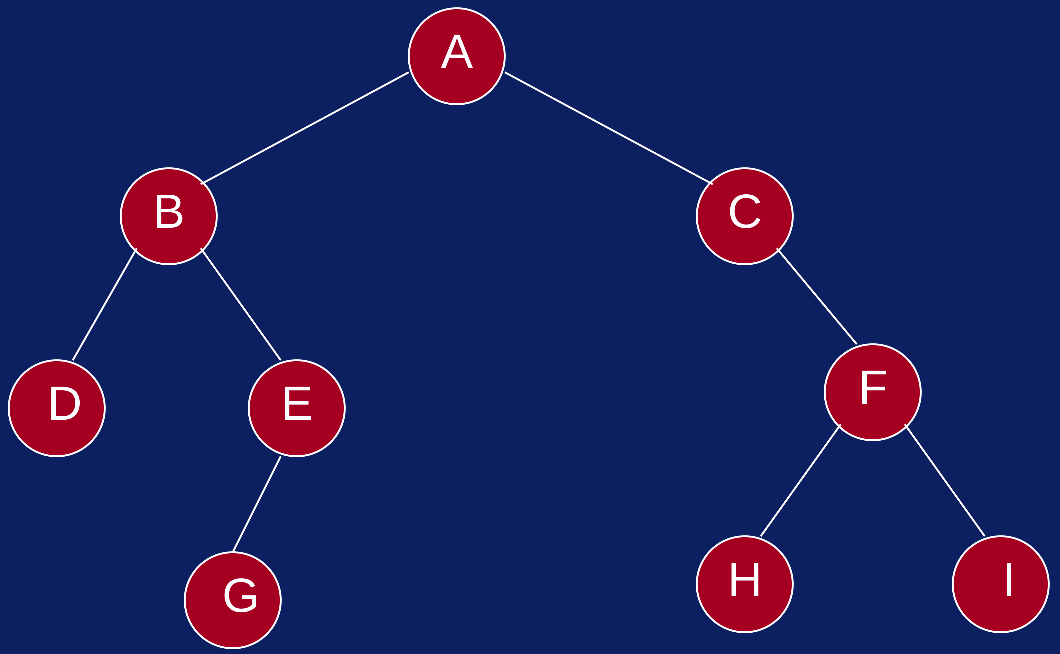
- A linear linked list will not be able to capture the tree-like relationship with ease.
- Shortly, we will see that for applications that require searching, linear data structures are not suitable.
- We will focus our attention on *binary trees*.

Binary Tree

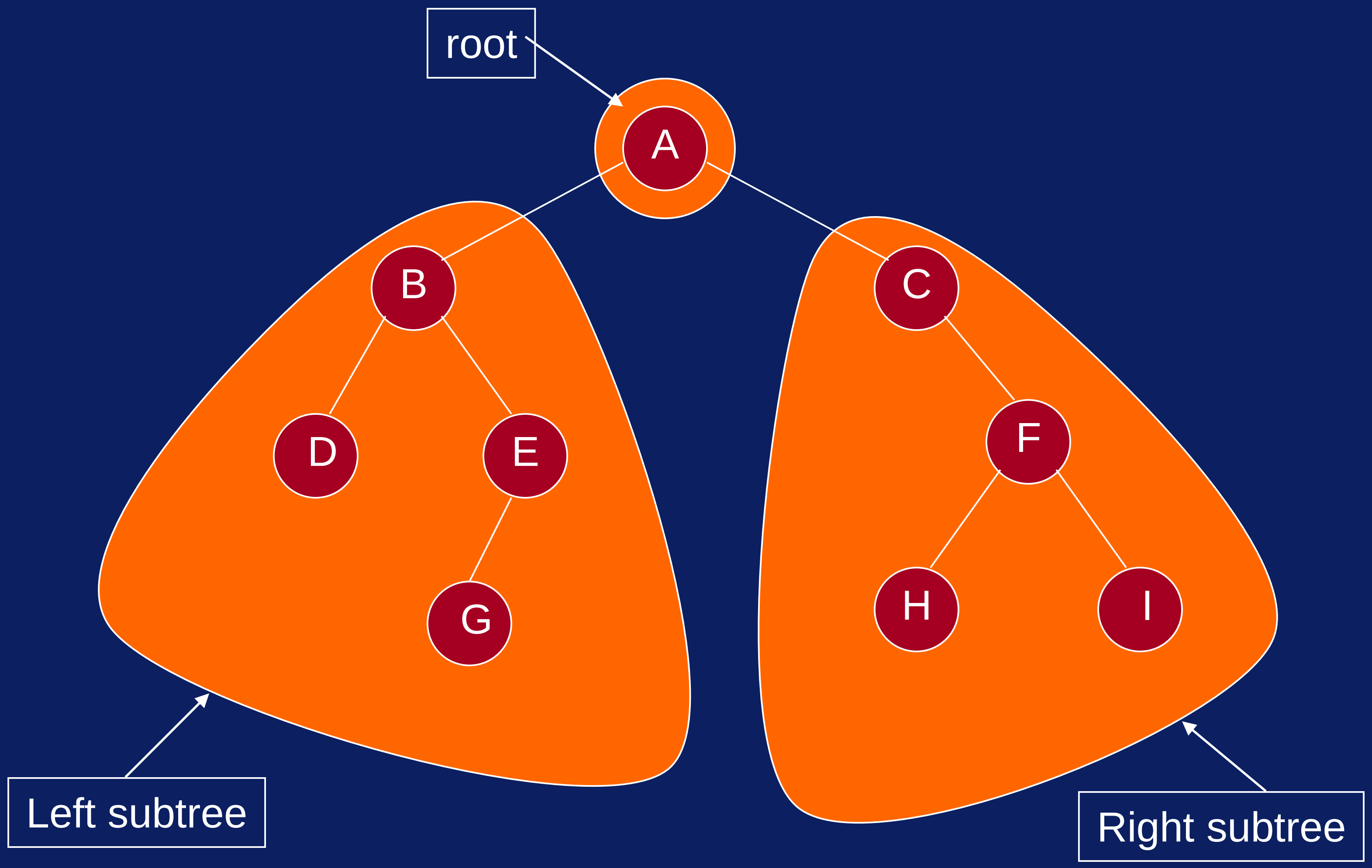
- A *binary tree* is a finite set of elements that is either empty or is partitioned into *three* disjoint subsets.
- The first subset contains a single element called the *root* of the tree.
- The other two subsets are themselves binary trees called the *left* and *right subtrees*.
- Each element of a binary tree is called a *node* of the tree.

Binary Tree

- Binary tree with 9 nodes.

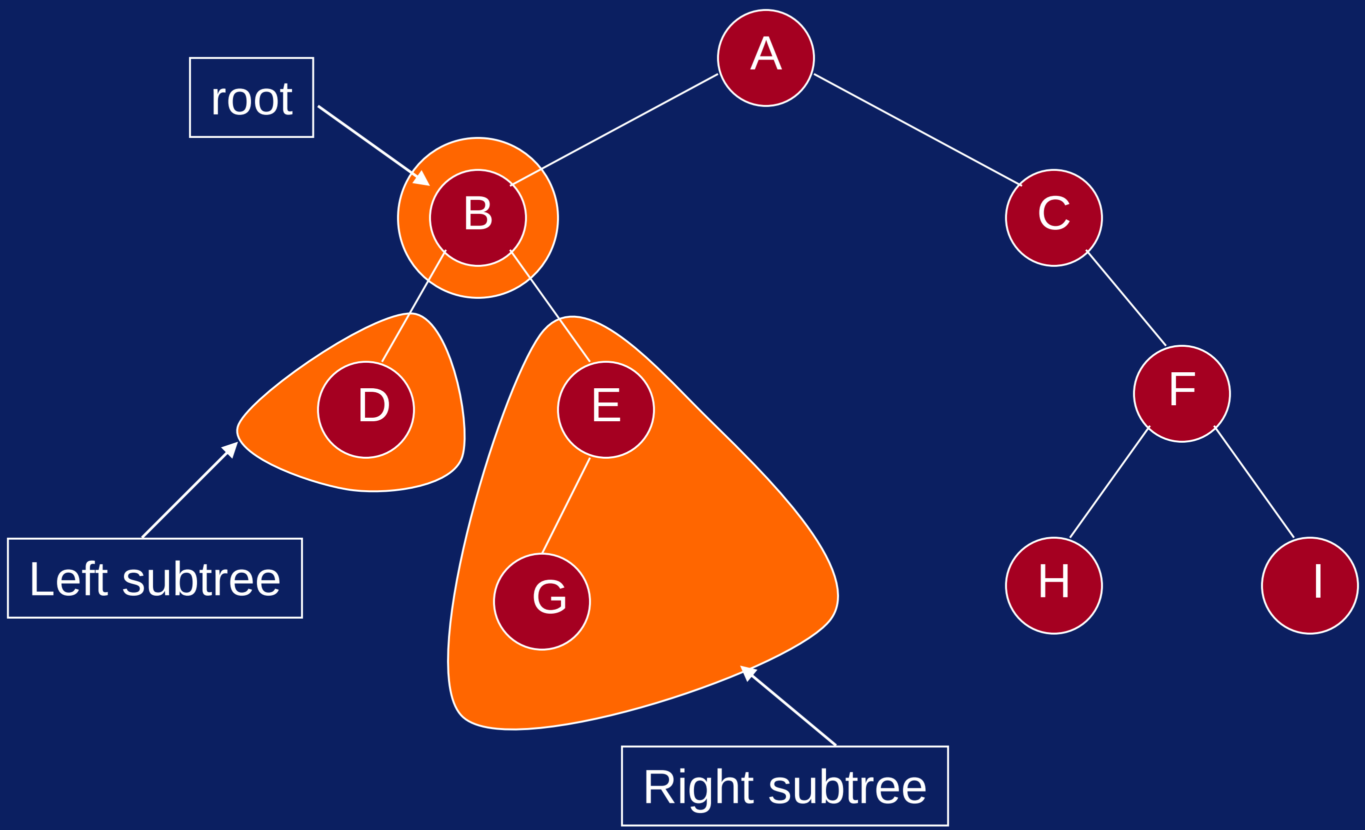


Binary Tree



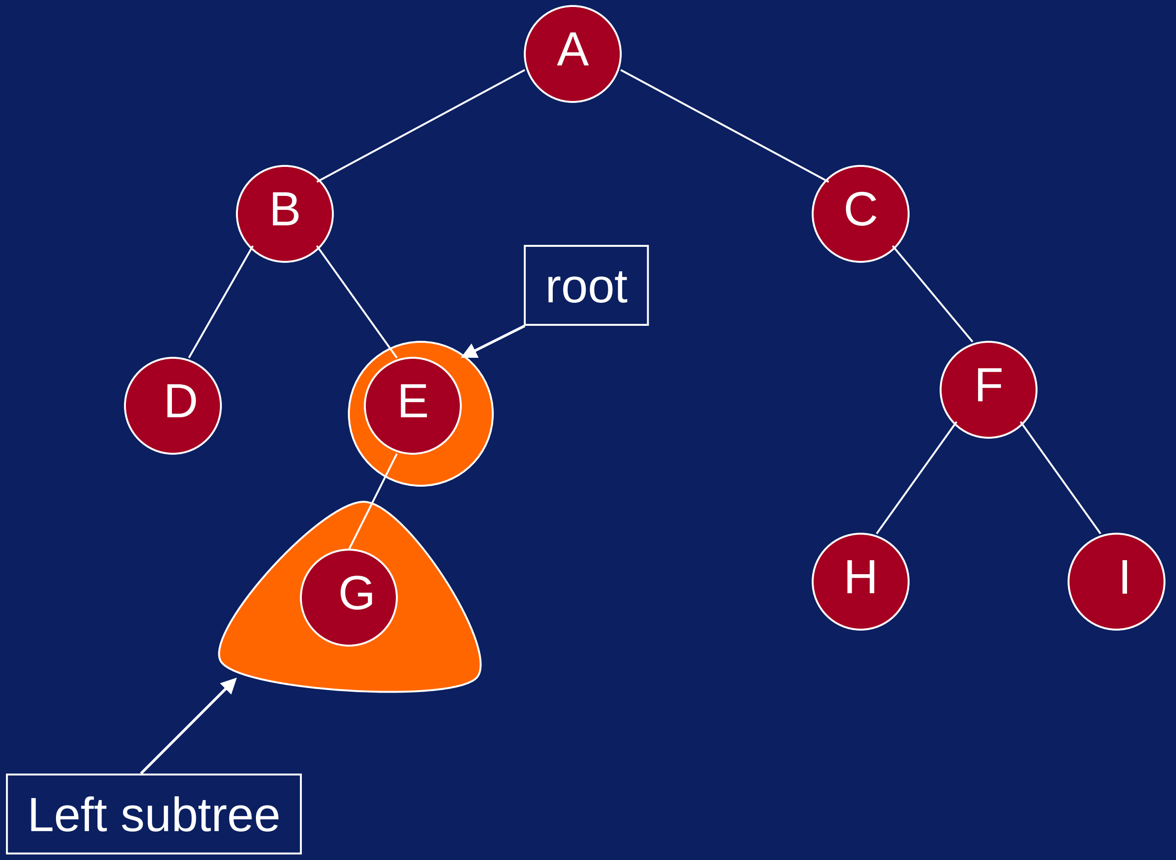
Binary Tree

- Recursive definition



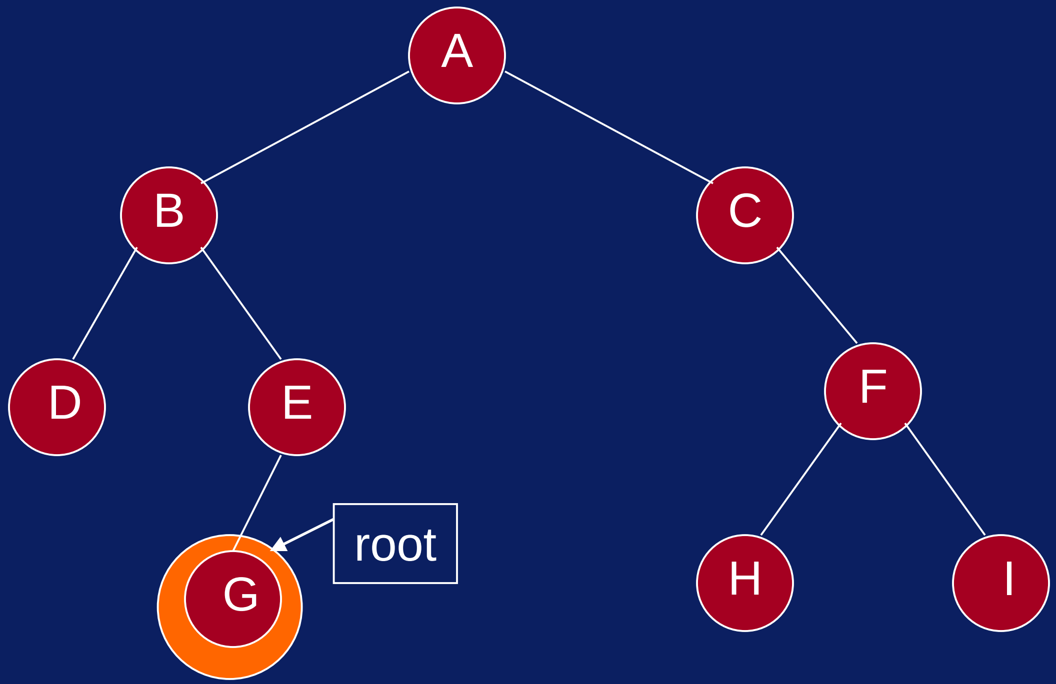
Binary Tree

- Recursive definition



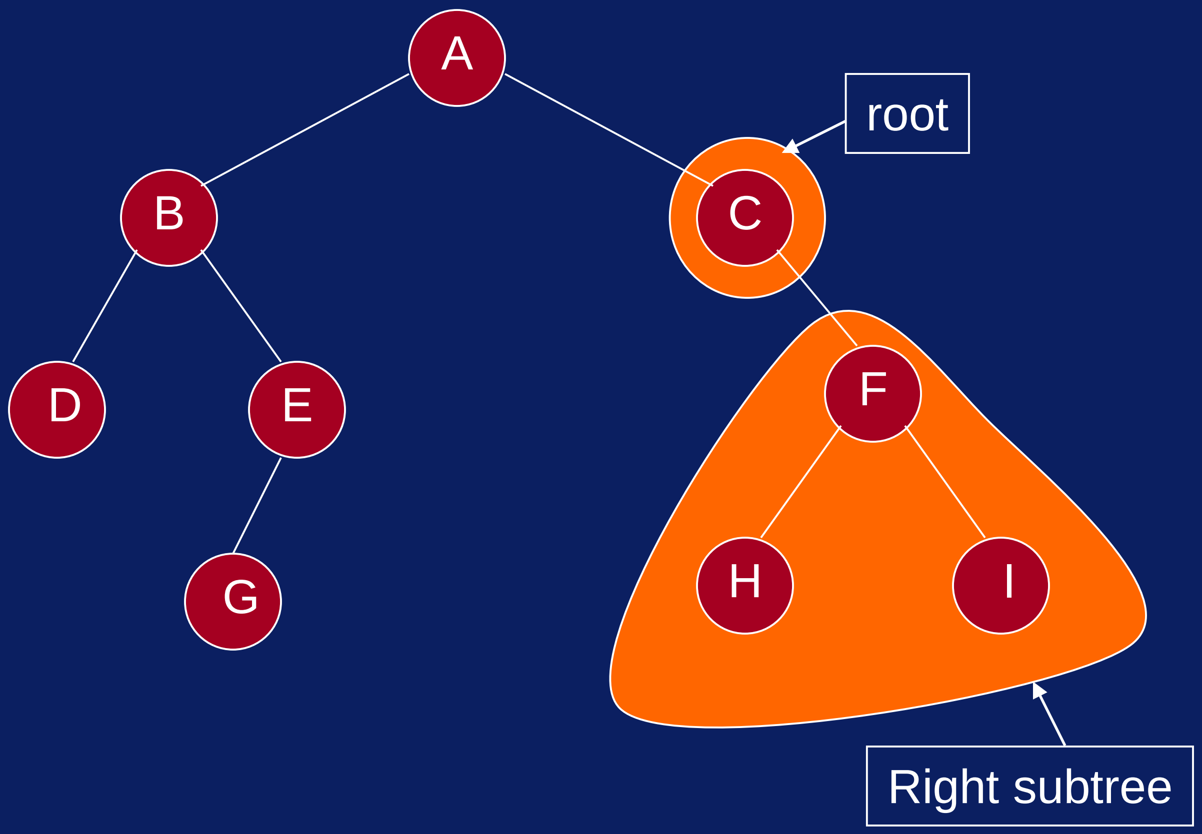
Binary Tree

- Recursive definition



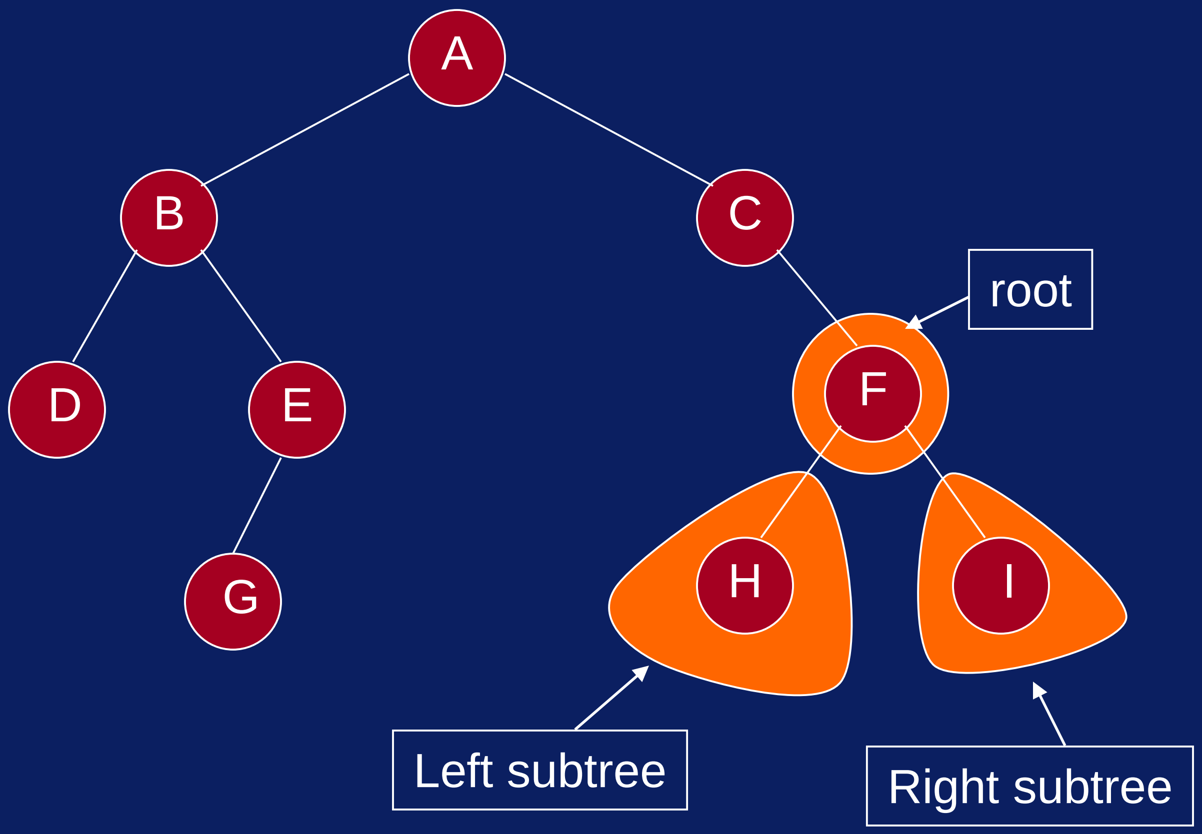
Binary Tree

- Recursive definition



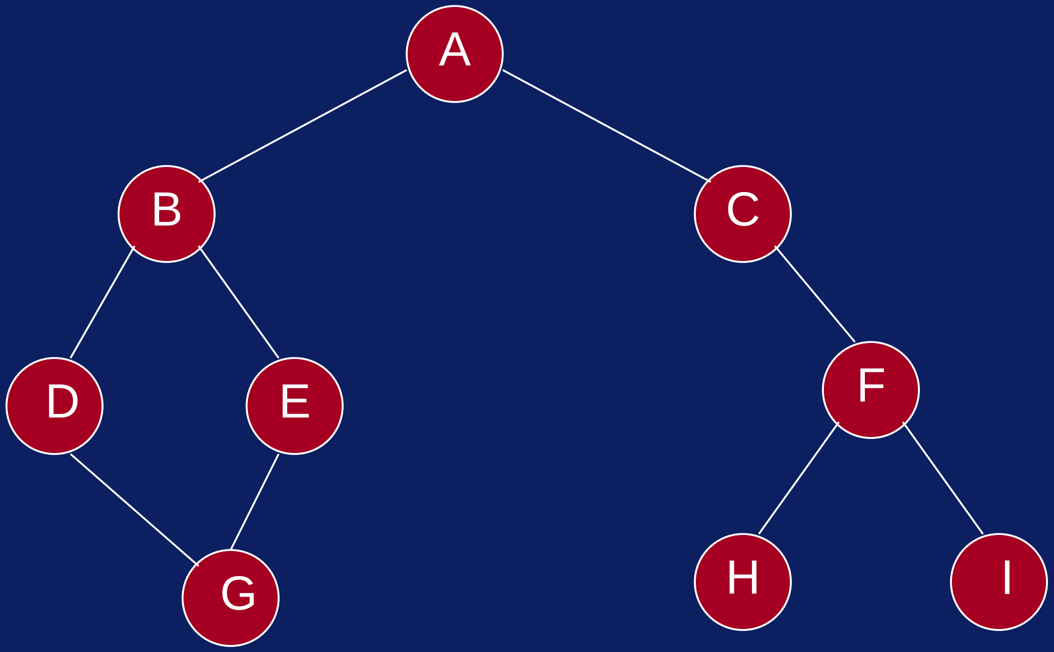
Binary Tree

- Recursive definition



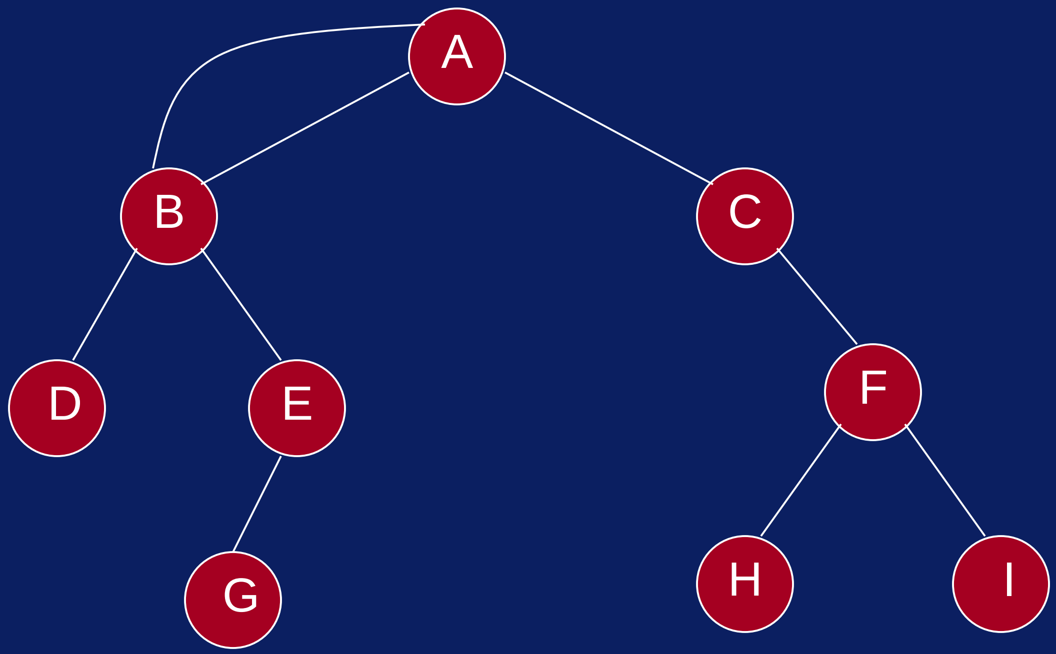
Not a Tree

- Structures that are not trees.



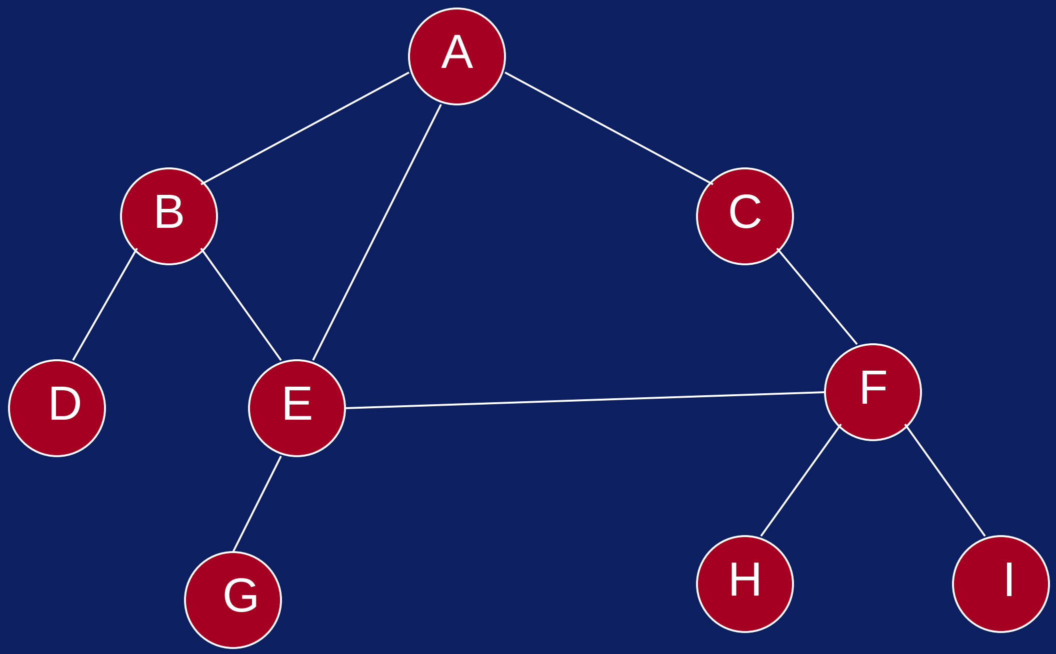
Not a Tree

- Structures that are not trees.

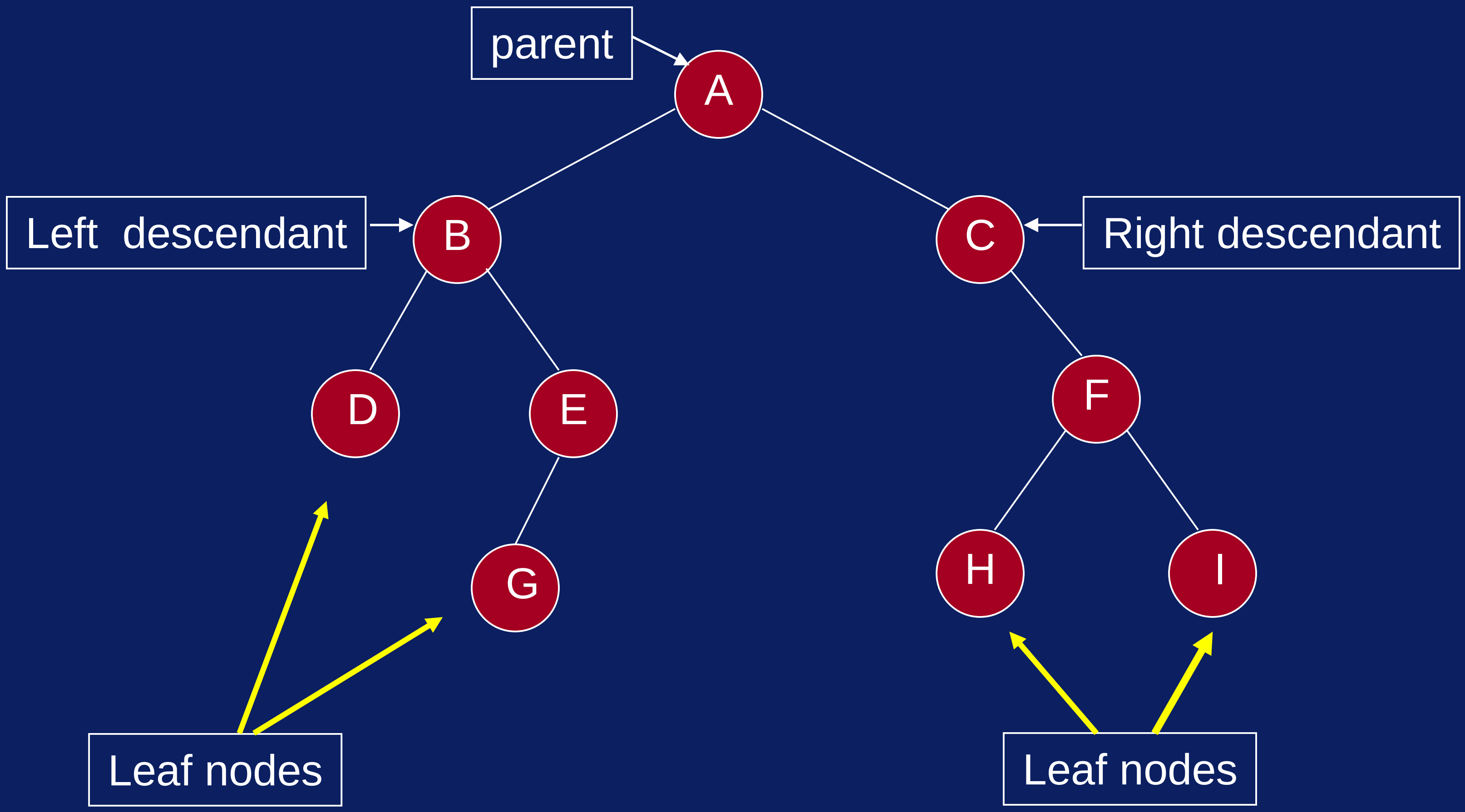


Not a Tree

- Structures that are not trees.

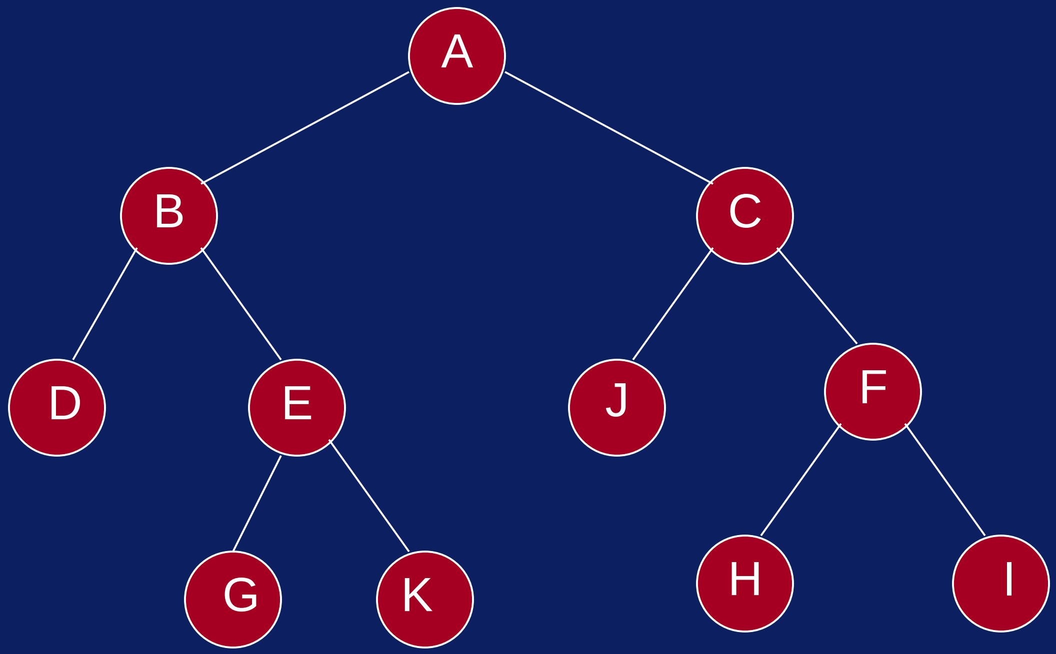


Binary Tree: Terminology



Binary Tree

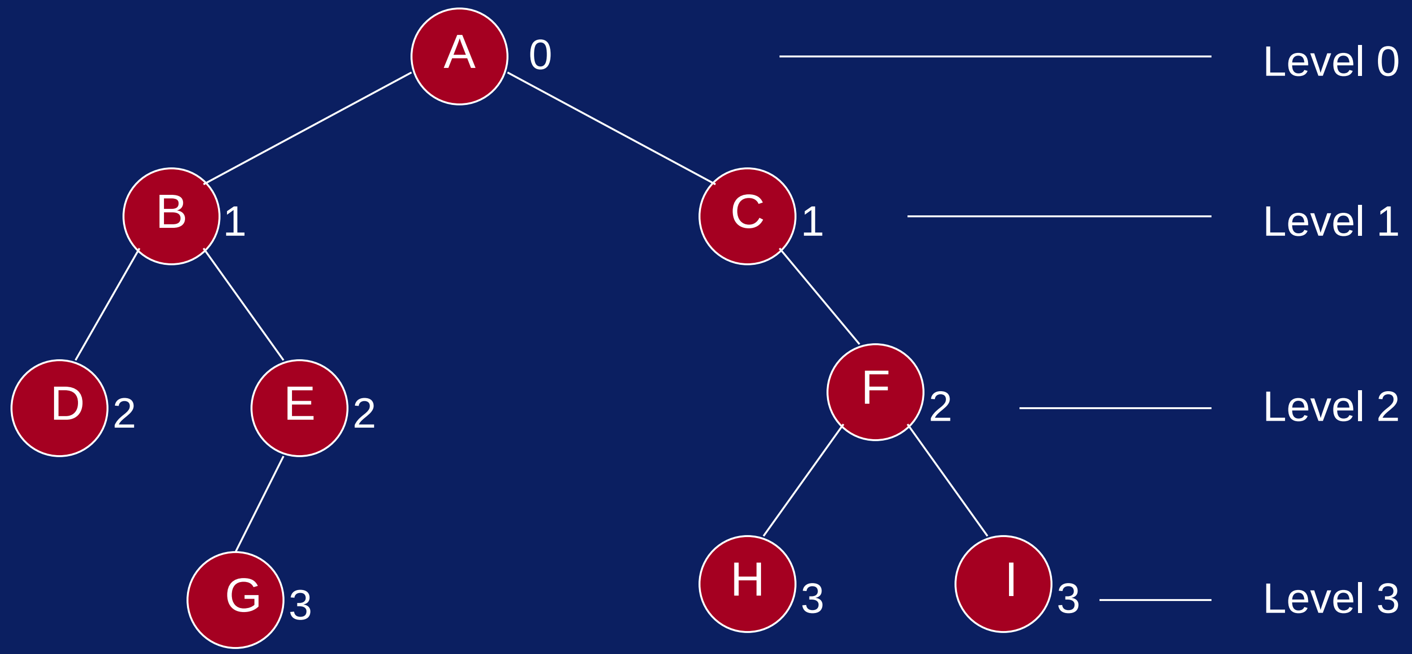
- If every non-leaf node in a binary tree has non-empty left and right subtrees, the tree is termed a *strictly binary tree*.



Level of a Binary Tree Node

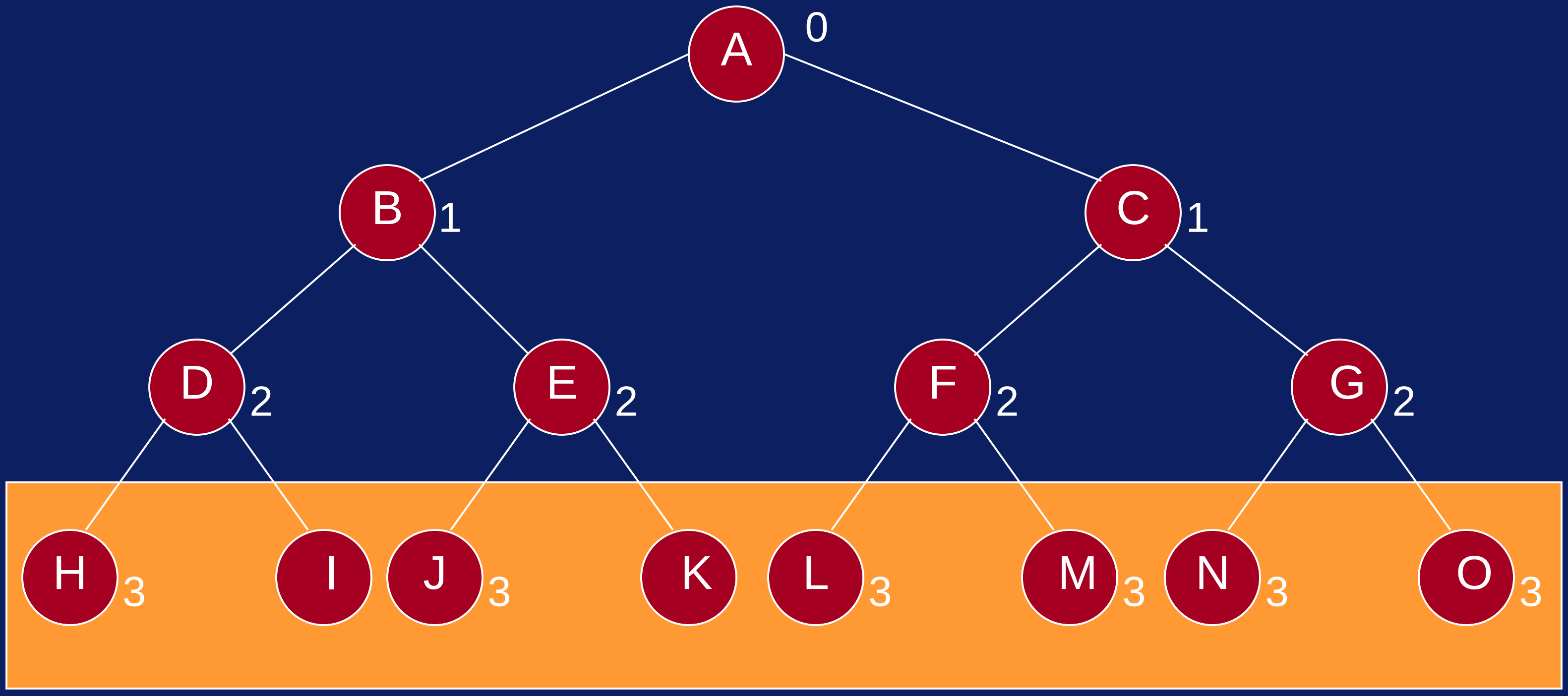
- The *level* of a node in a binary tree is defined as follows:
  - Root has level 0,
  - Level of any other node is one more than the level its parent (father).
- The *depth* of a binary tree is the maximum level of any leaf in the tree.

Level of a Binary Tree Node

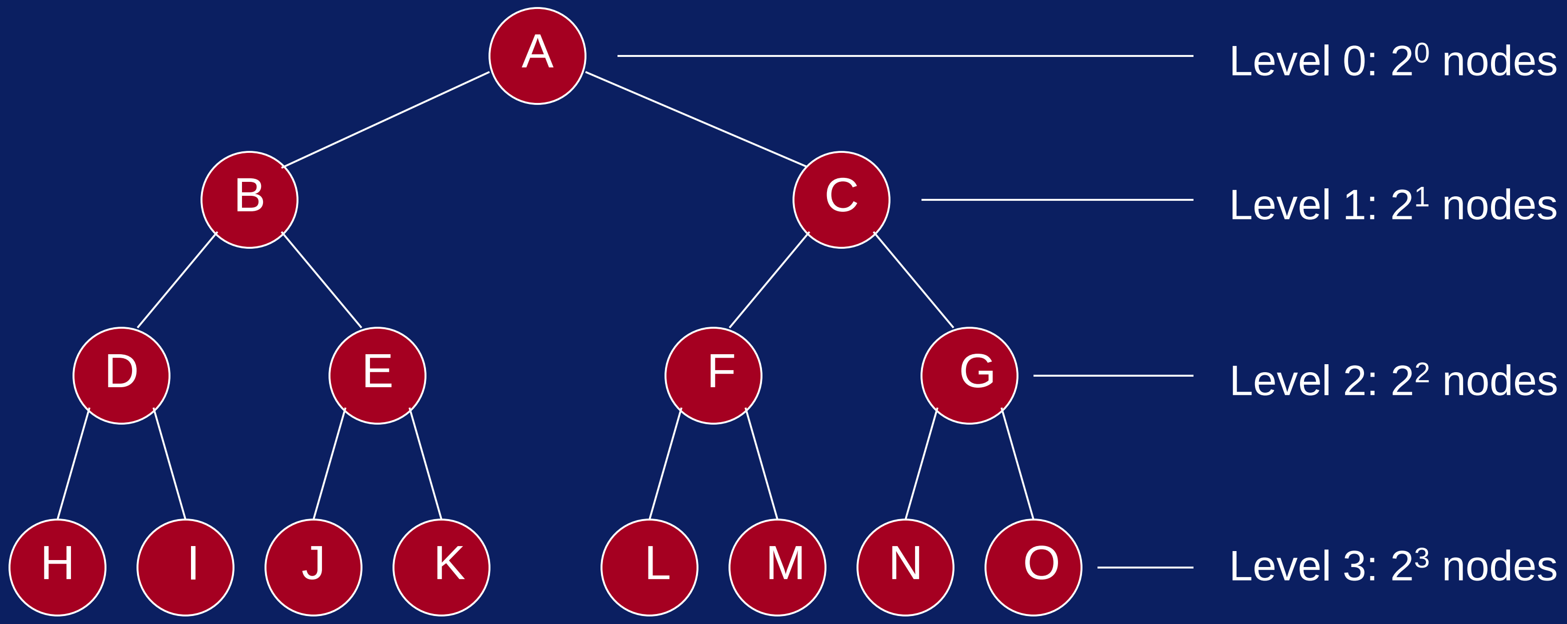


Complete Binary Tree

- A *complete binary tree* of depth *d* is the strictly binary all of whose leaves are at level *d*.



Complete Binary Tree



Complete Binary Tree

- At level *k*, there are 2^k nodes.
- Total number of nodes in the tree of depth *d*:

$$2^0 + 2^1 + 2^2 + \dots\dots\dots\dots + 2^d = \sum_{j=0}^{d} 2^j = 2^{d+1} - 1$$

- In a complete binary tree, there are 2^d leaf nodes and $(2^d - 1)$ non-leaf (inner) nodes.

Complete Binary Tree

- If the tree is built out of ‘n’ nodes then

$$n = 2^{d+1} - 1$$

or $$\log_2(n+1) = d+1$$

or $$d = \log_2(n+1) - 1$$

- I.e., the depth of the complete binary tree built using ‘n’ nodes will be $\log_2(n+1) - 1$.
- For example, for n=1,000,000, $\log_2(1000001)$ is less than 20; the tree would be 20 levels deep.
- The significance of this shallowness will become evident later.

Operations on Binary Tree

- There are a number of operations that can be defined for a binary tree.
- If *p* is pointing to a node in an existing tree then
  - left(*p*) returns pointer to the left subtree
  - right(*p*) returns pointer to right subtree
  - parent(*p*) returns the father of *p*
  - brother(*p*) returns brother of *p*.
  - info(p) returns content of the node.