Python
WEEK 6
Graphs & Trees

AI Academy

COMPUTER PROGRAMMING WITH PYTHON

Instructor - James E. Robinson, III

Teaching Assistant - Travis Martin

LIGHTNING REVIEW

- Variables
- Input / Output
- Expressions
- Functions
- Conditional Control
- Looping
- Data Types
- Logging
- Functions
- Scope
- **Recursion**
- Decorators
- Dynamic Prg
- Exceptions
- **Classes**
- **Objects**
- Encapsulation
- Public v/s Private
- Dunder Methods
- Instances
- Inheritance
- Types of Inheritance
- Polymorphism
- Method Overriding
- **Queue**
- **Stacks**

TOPICS COVERED

- Graphs
  - Types
- Trees
  - Elements of a Tree
- Binary Trees
- Traversal Methods
  - Depth First Traversal
  - Breadth First Traversal
- Searching

GRAPHS

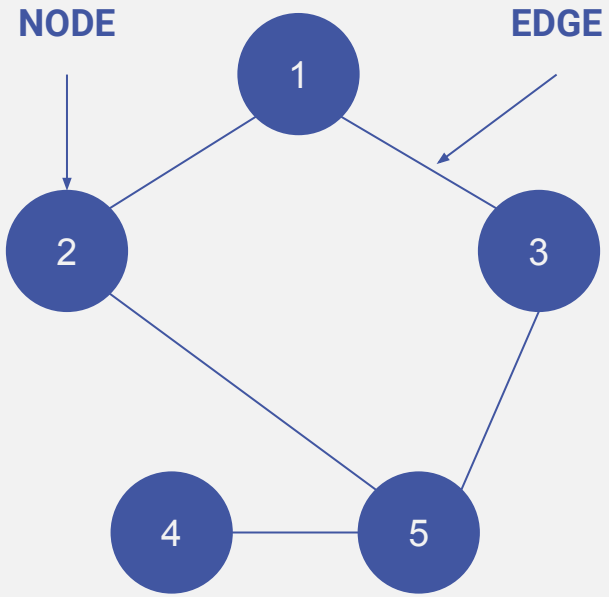
COLLECTION OF NODES CONNECTED BY EDGES

A Graph is a non-linear data structure consisting of finite nodes and edges between them

Node
The fundamental data element of a graph

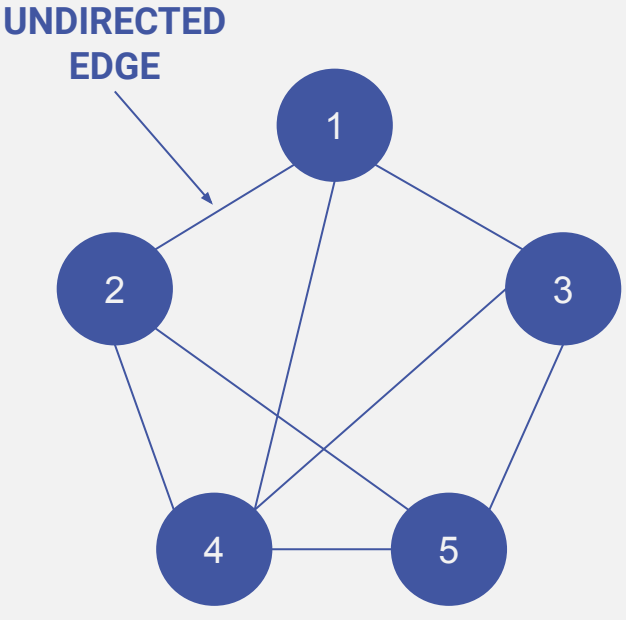
Edges
The connection between nodes that form the network of the graph

Types
Undirected Graphs & Directed Graphs

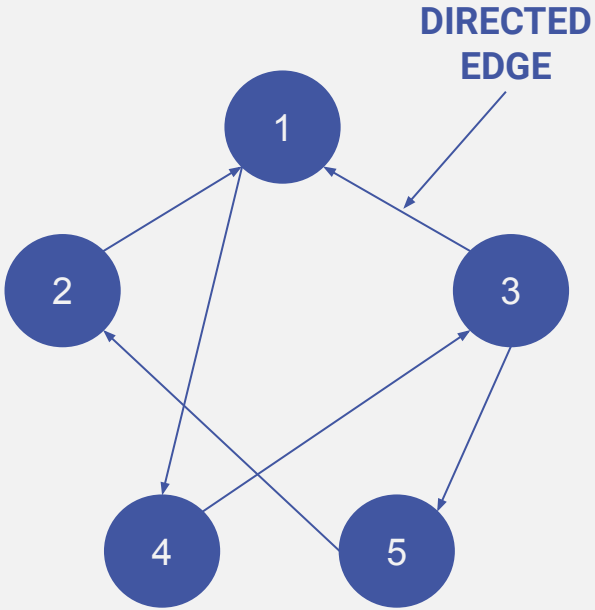


GRAPHS - TYPES

UNDIRECTED AND DIRECTED GRAPHS



UNDIRECTED GRAPH



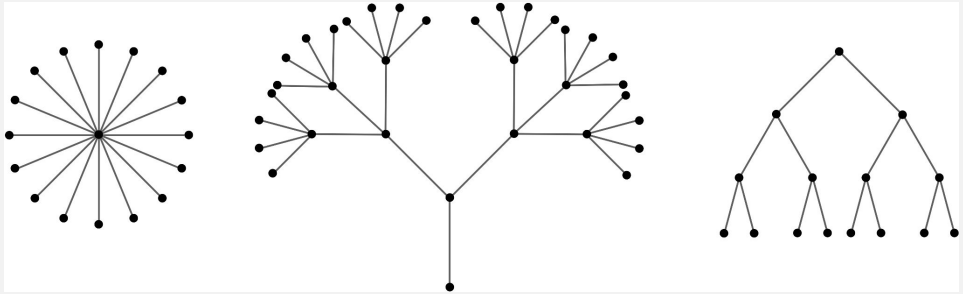
DIRECTED GRAPH

TREES

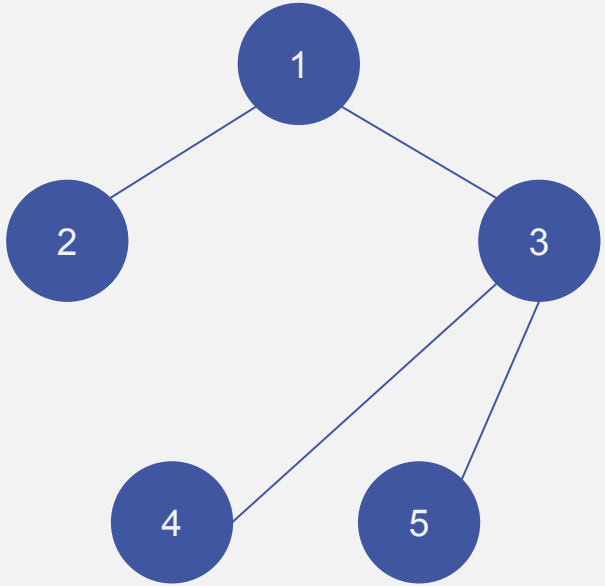
CONNECTED ACYCLIC UNDIRECTED GRAPHS

A Tree is an undirected graph in which any two vertices are connected by exactly one path and no cycles are formed

It is a widely used data type that simulates a hierarchical tree structure



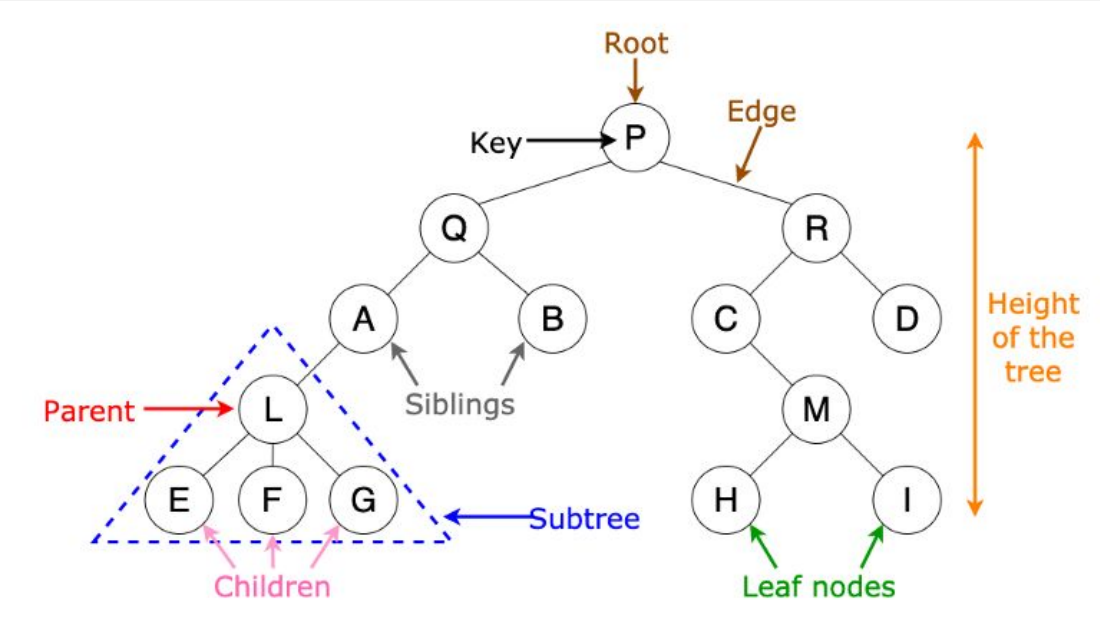
EXAMPLES OF TREES



ELEMENTS OF A TREE

DEFINITIONS

Root
Base node in a tree that has no parent
Leaf
Node with no children
Subtree
A tree which is a child of a node
Parent
Node which is a predecessor of any node
Child
Node which is a successor of any node
Sibling
Another Node with same parent



ELEMENTS OF A TREE

DEFINITIONS

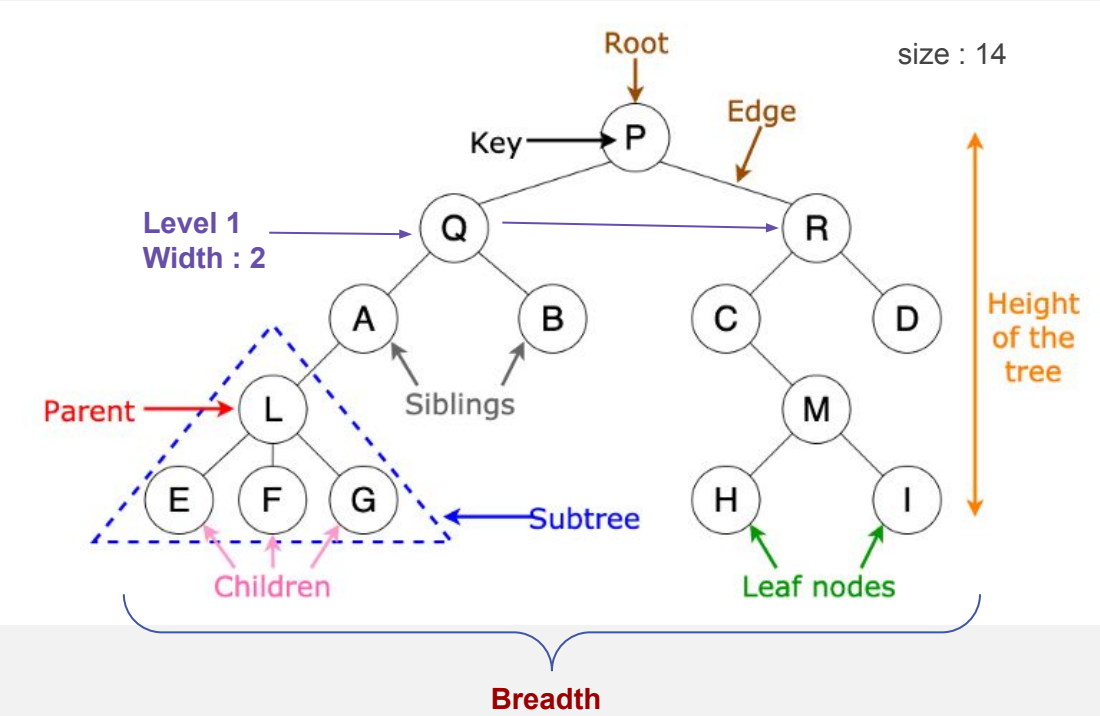
Height of tree
The length of the path from root of that tree to its farthest node

Size of a tree
Number of nodes in the tree

Breadth
The number of leaves

Level
Number of edges along the unique path between it and the root node.

Width of a level
The number of nodes in a level.

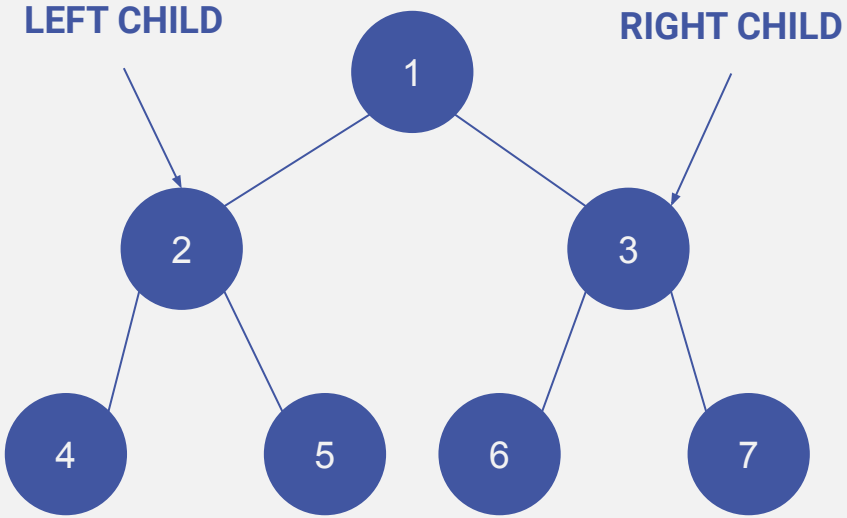


BINARY TREES

EACH NODE HAS AT MOST 2 CHILDREN

The 2 children are referred to as the left child and the right child

In computing, binary trees are mainly used for searching and sorting as they provide a means to store data hierarchically



TREE TRAVERSAL

VISITING EACH NODE IN A TREE DATA STRUCTURE EXACTLY ONCE

Depth First Traversal

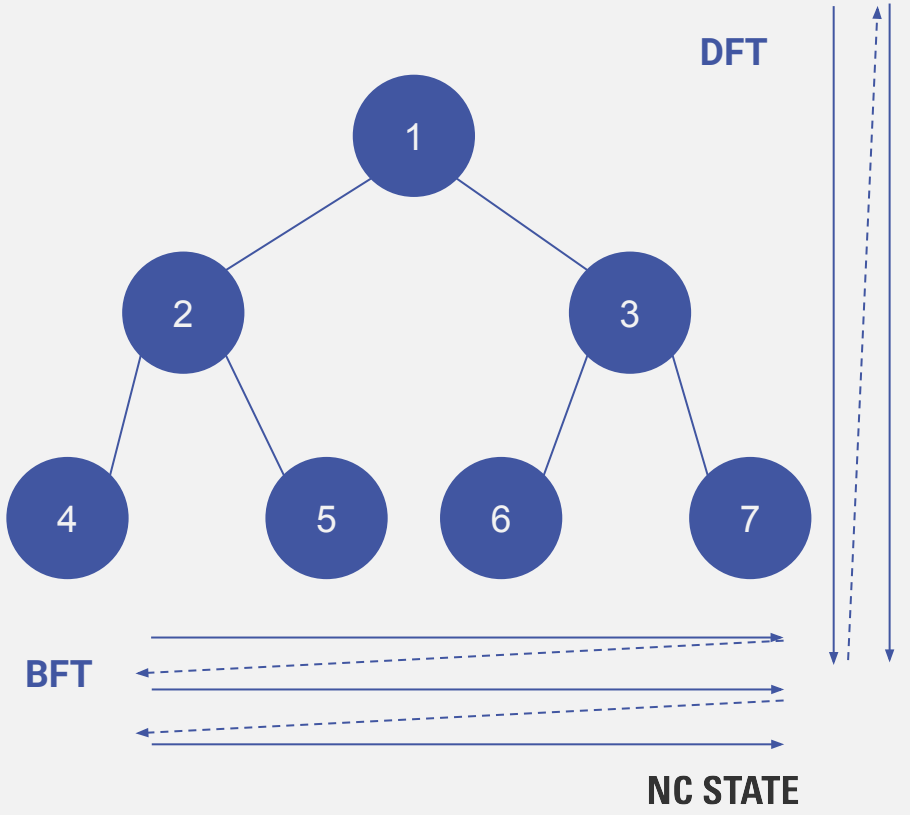
Start from root and move heightwise visiting all nodes in a branch before moving on to the next.

Can be implemented using a Stack

Breadth First Traversal

Start at the root and move widthwise visiting all nodes in a level before moving on to the next.

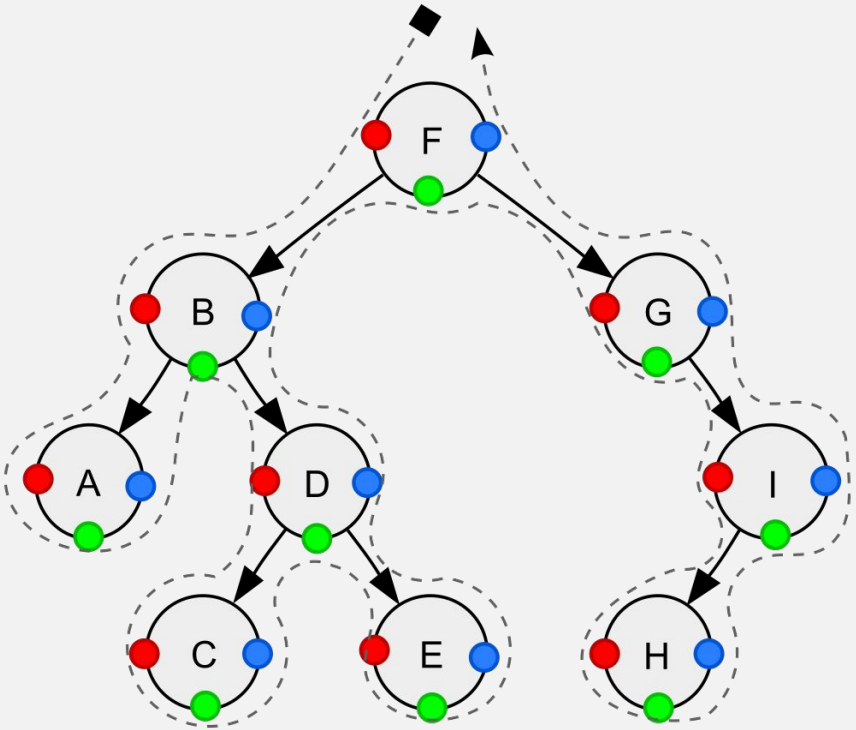
Can be implemented using a Queue



AI Academy

DEPTH FIRST TRAVERSAL

TYPES OF DEPTH FIRST TRAVERSAL



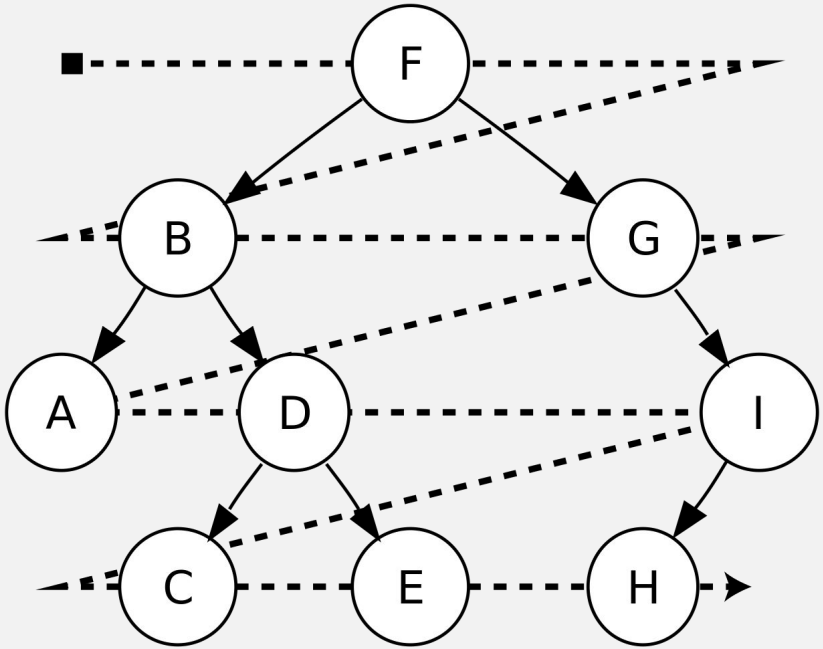
Inorder Traversal - (green)
A, B, C, D, E, F, G, H, I

Preorder Traversal - (red)
F, B, A, D, C, E, G, I, H

Postorder Traversal - (blue)
A, C, E, D, B, H, I, G, F

BREADTH FIRST TRAVERSAL

TYPES OF BREADTH FIRST TRAVERSAL



Level Order Traversal

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

SEARCHING

TRAVERSAL METHODS HELP ARE USEFUL FOR SEARCHING ALGORITHMS

Depth First Search (DFS)
Using a depth first or preorder traversal to search for a value since you have to access the node to get to the children anyway.

Uses: acyclic graph and topological order

Breadth First Search (BFS)
Using a breadth first or level order traversal to search for a value.

Uses: bipartite graph, and shortest path etc

TRAVERSING

EXAMPLE

```
graph = {
  '5' : ['3','7'],
  '3' : ['2', '4'],
  '7' : ['8'],
  '2' : [],
  '4' : [],
  '8' : []
}
```

```
def dft(visited, graph, node):
    if node not in visited:
        print (node)
        visited.append(node)
        for neighbour in graph[node]:
            dft(visited, graph, neighbour)
```

```
def bft(visited, graph, node):
  visited.append(node)
  queue.append(node)
  while queue:
    m = queue.pop(0)
    print (m, end = " ")
    for neighbour in graph[m]:
      if neighbour not in visited:
visited.append(neighbour)
queue.append(neighbour)
```

```
def main():
        visited = [ ]
        dft(visited, graph, '5')
        visited = [ ]
        queue = [ ]
        bft(visited, graph, '5')

main()
```

Note:

These are general standard algorithms, their only variation would arise from a different representation of graph.

NEW MODULE INTRODUCTION

NETWORKX

WEEK SUMMARY

- Learned about Graphs and types of graphs
- Learned concept Trees and types of trees
- Understood all elements of a tree
- Learned types of traversals
- Learned the uses and benefits of tree traversals

THANK YOU

FOR ADDITIONAL QUERIES OR DOUBTS
CONTACT:
jerobins@ncsu.edu

AI Academy