AIM :
Implement BFS searach algorithm using an undirected graph and developa recursive algorithm for searching all the vertices of graph OR tree data structure

---

Introduction

Breadth First Search

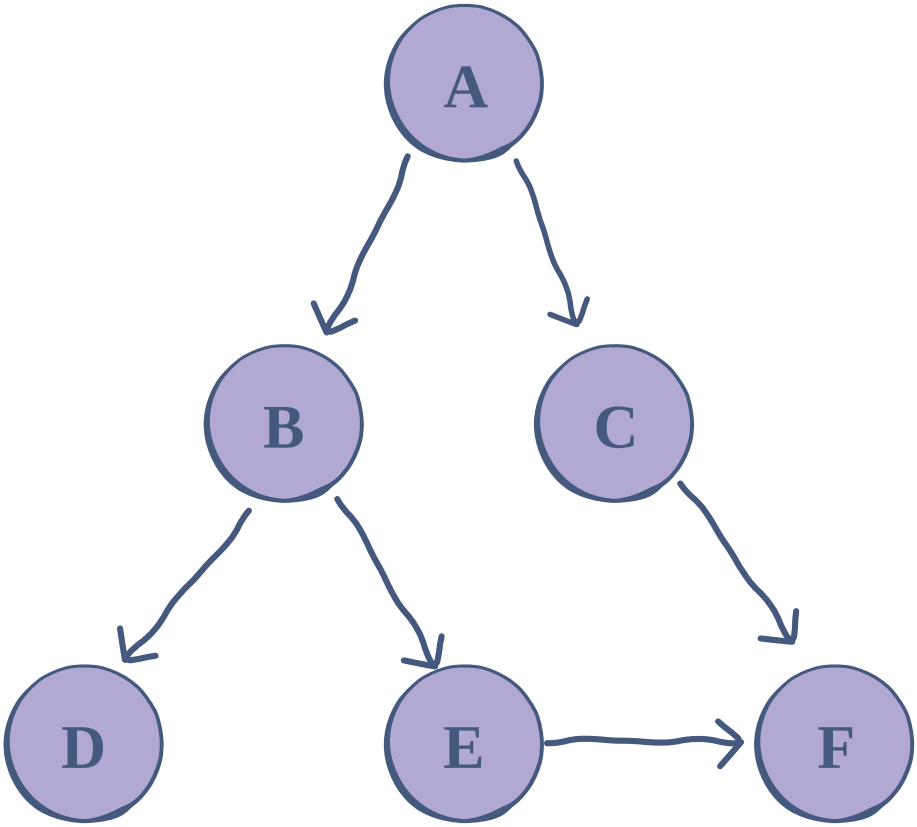
Breadth-First-Search(BFS)is an algorithm used for tree traversal on graphs or tree data structure ,BFS can be easily implemented using recursionand data structures like dictionaries and lists

The Algorithm

1. pick any node .visit the adjacent unvisited vertex mark it as visited,display it and insert it in queue
2. If there are no remaining adjacent vertices left, remove the first vertex from the queue.
3. Repeat setp 1 and step 2 untill the queue is empty or the desired node is found

Implementation

Consider this graph implemented in the code below:



In [1]:

```
graph = {
  'A' : ['B','C'],
  'B' : ['D', 'E'],
  'C' : ['F'],
  'D' : [],
  'E' : ['F'],
  'F' : []
}

visited = [] # List to keep track of visited nodes.
queue = []     #Initialize a queue

def bfs(visited, graph, node):
  visited.append(node)
  queue.append(node)

  while queue:
    s = queue.pop(0)
    print (s, end = " ")

    for neighbour in graph[s]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

# Driver Code
bfs(visited, graph, 'A')
```

```
A B C D E F
```

Explanation

- **Lines 3-10 :** The illustrated graph is represented using an adjacency list an easy way to do this in python is to use a dictionary data structure, where each vertex has a stored list of its adjacent nodes
- **Line 12:** *visited* is a list that is used to keep track of visited nodes
- **Line 13:** *queue* is a list that is used to keep track of nodes currently in the queue.
- **Line 29:** The arguments of the *bfs* function are the the *visited* list the *graph* in the form of the a dictionary and the starting node *A*
- **Lines 15-26:** follows the algorithms decribe above:
  - It checks and appends the starting node to the *visited* list and the *queue*
  - The while the queue cotains elements, it keeps taking out nodes from the queue, appends the nighbors of that node to tha queue if they are unvisited , and marks them as visited
  - This continues untill the queue is empty

Time Complexity

Since all of the nodes and vertices are visited, the time complexity for BFS on a graph is O(V+E); where V is the number of vertices and E is the number of edges