AIM :
Implement DFS algorithm use on undirected graph and develop a recursive algorithm for searching all the vertices of graph OR tree data structure

---

Introduction

Depth First Search

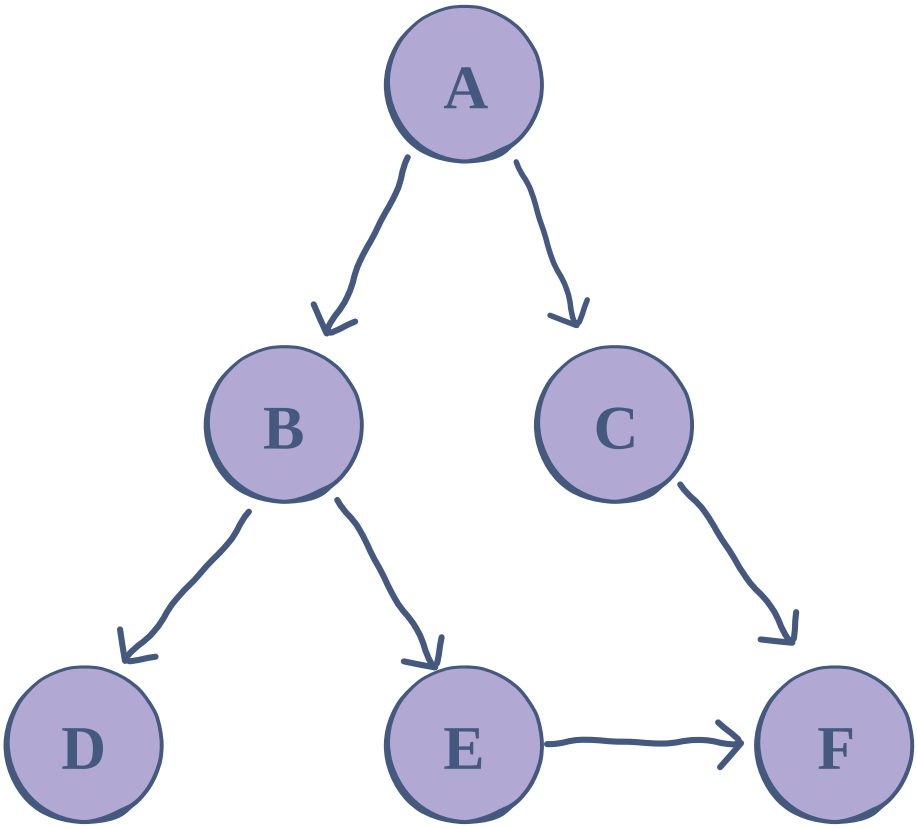
Depth-First-Search(DFS) is an algorithms for tree traversal on graph or tree data structure it can be implemented easily recursion and data structures like dictionaries and sets

The Algotithm ¶

1. pick any node . if it is unvisited , mark it as visited and recur on all its adjacent nodes
2. Repeat until all the nodes are visited , or the node to be searched is found

Implementation

Consider this graph implemented in the code below:



In [3]:

```
#using a python dictionary to act as an adjacency list
graph = {
    'A':['B','C'],
    'B':['D','E'],
    'C':['F'],
    'D':[],
    'E':['F'],
    'F':[],
}

visited = set() #Set to keep track of visited nodes

def dfs(visited,graph,node):
    if node not in visited:
        print (node)
        visited.add(node)
        for nighbour in graph[node]:
            dfs(visited,graph,nighbour)

#driver code
dfs(visited,graph,'A')
```

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

Explanation

- **lines 2-9:** The illustrated graph is represented using an **adjacency list-** an easy way to do it in python is to use a dictionary data structure. Each vetex has a list of its adjacent nodes stored
- **line 11:** *visited* is a set that is used to keep track of visited nodes.
- **line 21:** The *dfs* function is called and is passed the *visited* set , the *graph* in the from of dictionary and *A* which is the starting node
- **Lines 13-18:** *dfs* follows the algoruthm describe above
  - it first cheacks it the current node is unvisited - if yes , it is appended in the *visited* set
  - Then for each neightbor of the current node, the *dfs* function is invoked again
  - The base case is invoked when all the nodes are visited the function then returns

Time Complexity

since all the nodes and vertices and visited the average time complexity for DFS on graph is O(V +E) where V is the number of vertices and E is the number of edges. In case of DFS on tree the time complexity is O(V), where V is the number of nodes
average time complexity because a set's in operation has an average time complexity of O(1) O(1) . If we used a list, the complexity would be higher.

In [ ]:

```

```