- Rule 2: If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the
 vertices from the stack, which do not have adjacent vertices.)
- Rule 3: Repeat Rule 1 and Rule 2 until the stack is empty.

SOURCE CODE

```
trom Collections import default dict
class Graph:
    det __init_ - (self):
        self. value = defaultdict (list)
    def add Connection (self, parent, child):
         self. value [parent], append (child)
   def DFS (self, start):
        Vigited = [start]
         Stack = [start]
         Print (start, end = " ) 370 bus 398 monologist or margory a sign
         while stack;
   V as show and them S = stack [-1]
               if any (item for item in self. value[s] it item
                        not in visited):
             for item in Citem for item in self. vale
     [2] if item not in visited):
      Stack, append (item)
                        Visited append (item)
 His als es iv been ead by better to the the the the
             Print (item, end = ""
                         break
                                            LCORITHM: DFS
              else
                  stack.pop() were well and manufacture to the state of the
```

Visited = [start].

def BFS (self, start):

queue = [start]

while queue:

x = queue. pop(o)

Print (x, end = "")

for item in self. value [x]:

if item not in visited:

queue. append (item)

Nisted: append (item)

g = Geraph ()

g. add Connection (1, a)

g. add Connection (1, 2)

g. add Connection (2,3)

g. add Connection (2,6)

g. add Connection (4,5)

g. add Connection (4,5)

g. add Connection (4,7)

g. add Connection (7,96)

Print ("DFS Traversal?")

g. DFS(1)

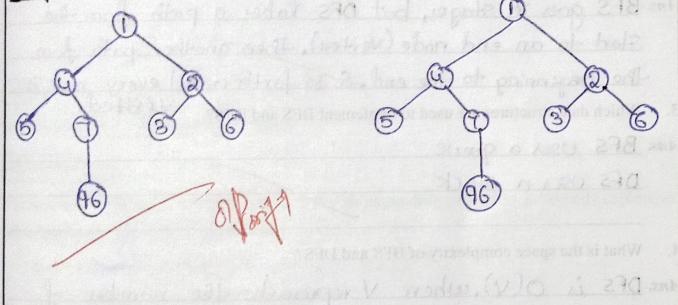
Print (MBFS Traversal!")

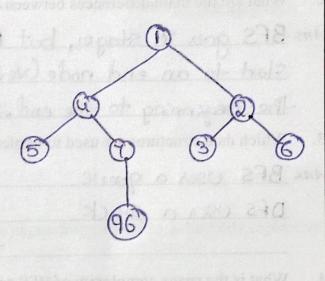
g. BFS(1)

OUTPUT

Traversal 2 3 6 7 96 BFS Traversal

Graphis





end it is almost all at each as

Regalt :-

Using IDLE (Python 3. 1.64-bit) we have siccenfully executed DFS & BFS Pagram

VIVA QUESTIONS

What is a Graph and tree?

Ans. Graphs are networks Consisting of nodes Connected by edges (oi) avcs. Trees are non-linear data structures That store data hierarchically & are made up of nodes

2. What are the main differences between BFS and DFS? Connected by edges.

Ans. BFS goes in stages, but DFS takes a path from the Start to an end node (Verten), then another path from the beginning to the end & so forth until every nock is

3. Which data structures are used to implement BFS and DFS?

Ans. BFS uses a gipule DFS uses a stack

What is the space complexity of BFS and DFS?

Ans. DFS is O(V), where V represents the number of Vertices in the graph. & for BFS it is O(V). where I represent the number of vortices in the

How does BFS handle cycles in a graph?

Ans. Initialize an empty queve. This queve will be holding a pair, noder and the previous node.

Week-1

Aimi-

Write a program to implement BFS and DFS Traversal

Algorithm: Bfs:-

-> Each vertex or node in graph is known. For instance, you can mark the node as v.

In case the Vertex V is not accessed then odd the Vertex V into the BFS Queue.

-> Start the BFS search, and after Completion, mark Vertex V our visited.

-> The BFs queue is still not empty, hence remove the vertex V of the graph from the queue.

-> Retrieve all the remaining vertices on the graph that are adjacent to the vertex V.

-> for each adjacent vertex let's say V1, in case it is not visited yet then add V1 to the BFS queue

-> BFS will visit VI and mark it as visited and delete it from the queve.

Algorithm: DFS

Depth First Search (DFS) algorithm troverses a graph in a depthward motion and user a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

- -> Rule 1: Visit the adjacent unvisited vertex. Mark it on visited. Display it. Push it in a stack.
- -> Rule 2: It no adjacent vertex is found, popupal vertex from the stack. (It will popupall the vertices from the stack, which do not have adjacent vertices.)
- -> Rule 3: Repeat Rule 1 and Rule 2 until the stack is empty.

```
Source Code :-
from Collections import defauthdict
claus Evraph:
    def -- init -- (self):
    def add Connection (Self, Pavent, Child):
         Self. Value [Parent]. append (child)
    def DFS (Self, Stort):
         Visited = [start] (Mallenten & Blow)
         Stark = Start
                            e (open) notasanallobous
          Print (Start, end ="
                                  Email 210 " ) kang
          while stack:
             S = Stack [-i]
              if any (item for item in self. value[s] it
                              item not in visited):
               -britem in [item for item in self value
                          if item not in visited]:
                Stack append (item)
                  visited append (item)
                   Print (item, end = " ")
                   break
                else:
                  Stack. Pop()
       def BFS (Self, Start):
                                    Che Trovoried:
            Visited = [start]
                                     of t d of 1
            queve = [start]
             while queue:
                X = queve, pop (o)
                Print(x, end="")
               for item in self-value [x]: >i+
                 queve. append (item)
                      Visited-append (item)
```

9= Graph() Whatfor tropping woodsolled mont g. addConnection (1, u) g. add Connection (1,2) g. addConnection (2,3) g. add Connection (2,6) g. add Connection (4,5) 18, 462) 270 466 g. add Connection (4,7) hale = totaly g. add Connection (7,96) Print (" DFS. Traversal: ") g. ofs(i) Print ("In BFS Traversal;") 9. BFS(1) Graph= DFS & Output: OFS Traversal: 2 13 6 1 4 5 7 96 BFS Traversal & the part of the 142573 Resolati Using IDLE (Python 3.9.64-bit) we have & couply programa executed DFS & BFS