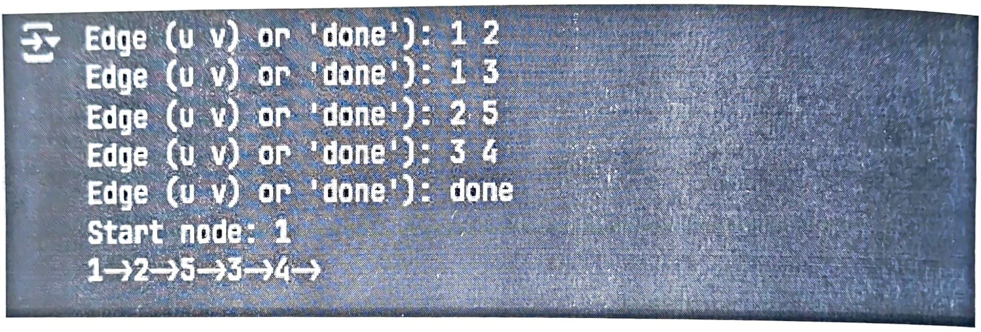
PROGRAM

```
graph = {}
def add-~~graph~~(u, v):
        graph.Setdefault(U, []).append(v)
        graph.Set default(v, []).append(u)
def dfs(n, V=Set()):
    if n not in v: V.add(n); print(n, end="→"); [dfs(nei, v)
                                        for nei in graph.get(n [])]
while (e:= input("Edge (U V) or 'done'): "))!= 'done':
        add-edge(*map(int, e.split()))
dfs(int(input("Start node: ")))
```

GOOGLE COLLABS



```
Edge (u v) or 'done'): 1 2
Edge (u v) or 'done'): 1 3
Edge (u v) or 'done'): 2 5
Edge (u v) or 'done'): 3 4
Edge (u v) or 'done'): done
Start node: 1
1→2→5→3→4→
```

OUTPUT

```
Edge (U V) or 'done'): 1 2
Edge (U V) or 'done'): 1 3
Edge (U V) or 'done'): 2 5
Edge (U V) or 'done'): ~~done~~ 3 4
Edge (U V) or 'done'): done
Start node: 1
1→2→5→3→4→
```

10/09/2024

EXP NO: 2

EXP NAME: DEPTH FIRST SEARCH ALGORITHM

AIM: To implement depth first search algorithm in python.

ALGORITHM:

1) Declare graph as an empty set. dictionary
2) Get the number of nodes as U V (Eg: 1 2)
3) Repeat process till we get 'done'.
4) Also get starting node.
5) The add edge function creates an empty list for dest node and adds the node to that like U, [V] and V, [U]
6) Next is the dfs traversal where each node is checked if in visited then adds it if not and prints node
7) Then all connected nodes are visited of that node.
8) Thus process continues.

RESULT:

Thus DFS has been implemented using python.