



## LP-II Assignment-01

DOP: 7-Jan'22 Pos: 13-Jan'22

12 7 p gst	my while quere is not county sevent a					
25 wooddaga	Title: BFS and DFS algorithm.					
	topoloxa dan app dalle Mas					
Problem Statement 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1						
	Implement depth fixt search algorithm & Breadth					
	first search algorithm, Use an undirected graph					
	& develop a recursive algorithm for searching all					
	the vortices of a graph or tree data shouture					
24123	1) It stork Herri Beak nocto 4 an in 1et					
	Objectives note that don't but Is you					
	17 To leasen about graph-search algorithms.					
	15 To formalize & implement constraints in peacel					
291	The Uniproblems which was district					
	2 d la la la la constante de l					
31. 1	Soffware & Hardware Requirements.					
8-CTB RAM 4 512 GB S50						
	vs-code latest version who appear and					
Python-3-8011mm						
	Theory Related Coments:					
Aluna 9	BFS? hadrogx? Mann?					
	1) Breadth-fixb-search for graph is a graph					
2009	seasching algorithm.					
	2) of steats trous 2004 node + explores and 4re					
	neighbouring modes.					
	Algorithm:					
	"> select a start mode					
	1) Create queue & enqueu steat node					



PICT, PUNE	Assigned OF					
	11) while queue is not empty repeat step 4 f 5.					
	14) Dequeu a mode N & enqueue all neighbones					
	of N that are not explored					
	VP Sit explored of N to true					
	vi) exit.					
MADED	tme: O(V+E) sprue: O(E)					
0	DPS: Out multipole among a galanch 2					
50	> 1 s graph searching algorithm.					
	2) It stock from 200f nocle 4 go in reursire					
	way. If uses stack data algorithm					
	It is leave about graph seasch about house					
123	Algorithm: Danislam & Danislam Danislam					
	is select any arbitrary vertex & call dis	102				
	that node					
	11) mark when mode as visited					
	us for all adjacent unwested nodes all dfs.					
	time complexity: 10 (V+E) del aban 24					
	V = Vertices E => Edges					
	Test cases stormed bothor website					
	Produ	. 1				
	Graph Experted Actual	Result				
	12354 12354	Pass				
onp		COS				
20har susualdrian						
	(4)—(5)					
	show tota a dulis					
	is about a sugar & sugar (1)					

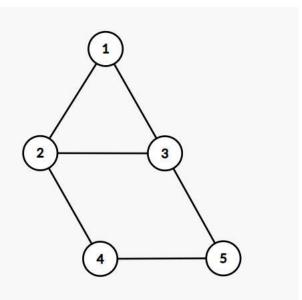


PICT, PUNE							
	0	12345	123 45	Pas			
		12345	12345				
	2 3						
	45						
	(1)	123	123	Pan.			
	30	45	45				
		123	12-3				
	3-5	45	45				
	Conclusions						
	Through this	Through this amonment we have swentilly					
	Implemente	Through this assignment we have successfully implemented 4 bouned graph search technoques.					
		turned graph sain sellinging.					
		BOAN O'S LANGE STORY	Marchael March				
TO PERSON							
Lander							
<u> </u>				gradient is an in-			
THE REAL PROPERTY.				Shows in			
The same of the same							

## Code:

```
# Shubham - 31118
# undirected graph
class Graph:
    def __init__(self, n) -> None:
        self.n = n
        self.m = 0
        self.adj_list = [[] for _ in range(n+7)]
    def add adge(self, u, v):
        self.m += 1
        self.adj list[u].append(v)
        self.adj_list[v].append(u)
    def print_graph(self):
        print("Graph (Adjacency List Representation):")
        for u in range(self.n):
            print(u, end="-> ")
            for v in self.adj_list[u]:
                print(v, end=" ")
            print()
    def dfs_ut(self, u, vis):
        vis.append(u)
        print(u, end=" ")
        for v in self.adj list[u]:
            if v not in vis:
                self.dfs_ut(v, vis)
    def dfs(self):
        print("Depth-first-traversal: ")
        vis = []
        for i in range(1, self.n+1):
            if i not in vis:
                self.dfs_ut(i, vis)
                print()
    def bfs_ut(self, vis, que):
        if not que:
            return
        u = que[0]
        que.pop(0)
        print(u, end=" ")
```

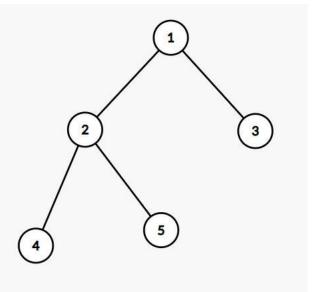
```
for v in self.adj_list[u]:
            if v not in vis:
                vis.append(v)
                que.append(v)
        self.bfs_ut(vis, que)
    def bfs(self):
        print("Breadth-first-traversal: ")
        vis = []
        for i in range(1, self.n+1):
            if i not in vis:
                vis.append(i)
                self.bfs_ut(vis, [i])
                print()
def main():
    # 1 - based indexing
    n = int(input("Enter number of vertices: "))
    m = int(input("Enter number of edges: "))
    g = Graph(n)
    print("Enter the edges details")
    for i in range(m):
        inp = input()
        [u_, v_] = inp.split(' ')
        u = int(u_)
        v = int(v)
        g.add_adge(u, v)
    g.print_graph()
    g.dfs()
    g.bfs()
main()
```



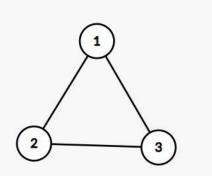
```
Shubham@Shubham-AcerSwift MINGW64 /d/College-Stuff-6th-Sem/LPII (main)

$ py 31118_A01.py
Enter number of vertices: 5
Enter number of edges: 6
Enter the edges details

1 2
2 3
3 1
2 4
4 5
3 5
0->
1-> 2 3
2-> 1 3 4
3-> 2 1 5
4-> 2 5
Depth-first-traversal:
1 2 3 5 4
Breadth-first-traversal:
1 2 3 4 5
```



```
Shubham@Shubham-AcerSwift MINGW64 /d/College-Stuff-6th-Sem/LPII (main)
$ py 31118_A01.py
Enter number of vertices: 5
Enter number of edges: 4
Enter the edges details
1 2
2 3
1 3
4 5
Graph:
0->
1-> 2 3
2-> 1 3
3-> 2 1
4-> 5
Depth-first-traversal:
1 2 3 4 5
Breadth-first-traversal:
1 2 3 4 5
```





```
Shubham@Shubham-AcerSwift MINGW64 /d/College-Stuff-6th-Sem/LPII (main)
$ py 31118_A01.py
Enter number of vertices: 5
Enter number of edges: 4
Enter the edges details
1 2
2 3
1 3
4 5
Graph:
0->
1-> 2 3
2-> 1 3
3-> 2 1
4-> 5
Depth-first-traversal:
1 2 3
4 5
Breadth-first-traversal:
1 2 3
4 5
```