	PUNE INSTITUTE OF COMPUTER TECHNOLOGY PUNE - 411043	
	Department of Electronics & Telecommunication	
	ASSESMENT YEAR: 2020-2021	CLASS: SE 5
	SUBJECT: DATA STRUCTURES	
EXPT No: 6	LAB Ref: SE/2020-21/	Starting date: 05/12/2020
	Roll No: 22119	Submission date: 05/12/2020
Title:	Creation of Graph	

Prerequisites:	• DEVC++ IDE • Graphs in C • Structures and linked list in c
Objectives:	• Learn the concepts nonlinear data structure in implementation of graph (Cyclic data structure). • Apply BFS traversal to visit graph nodes using queue structure. • Apply DFS traversal to visit graph nodes using stack structure

Theory:

Graphs are one of the most interesting data structures in computer science. Graphs and the trees are somewhat similar by their structure. two important differences between trees and graphs.

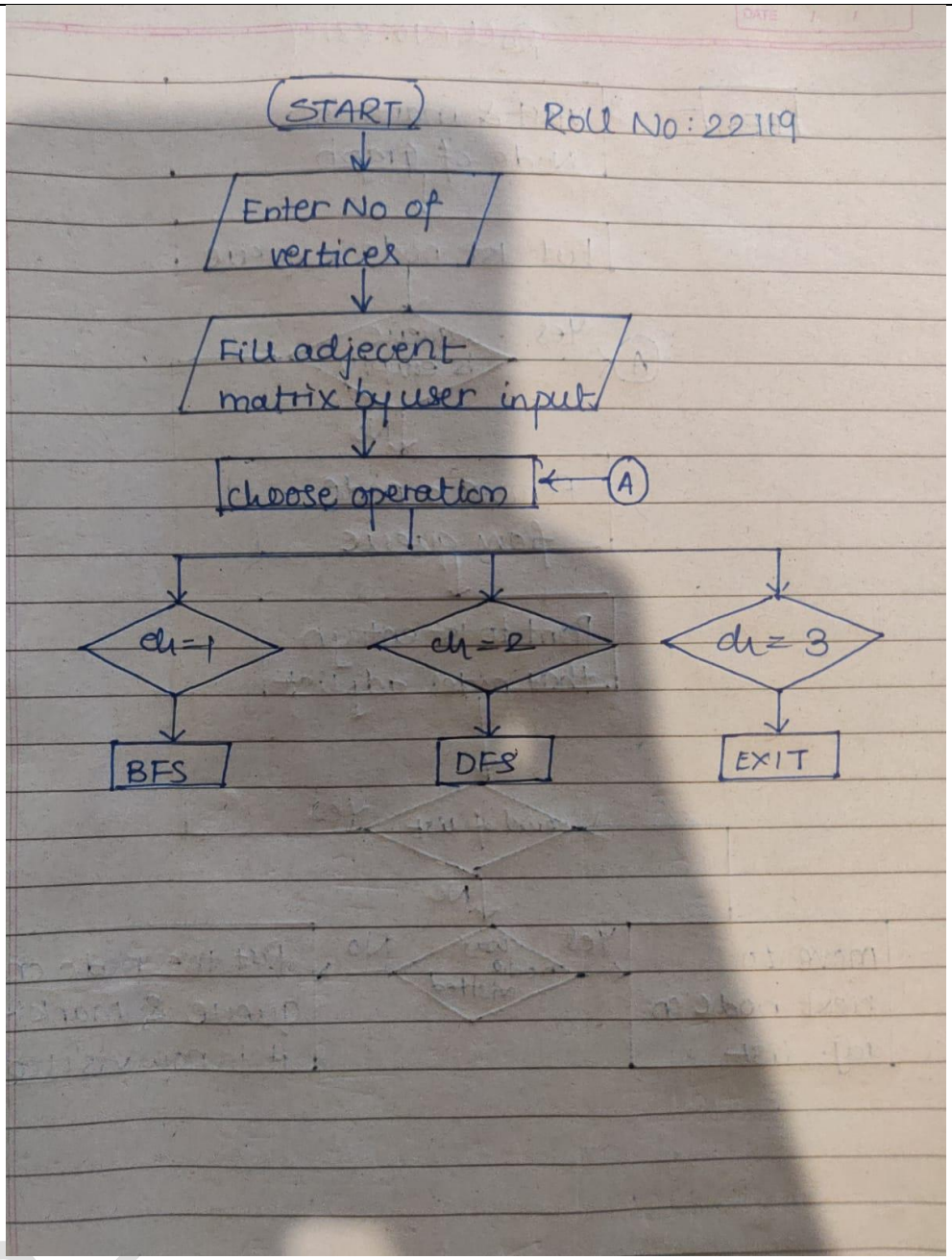
1. Unlike trees, in graphs, a node can have many parents.
2. The link between the nodes may have values or weights.

Every graph has two components, Nodes and Edges. Nodes are implemented by class, structures or as Link-List nodes. Edges represent the connection between nodes.

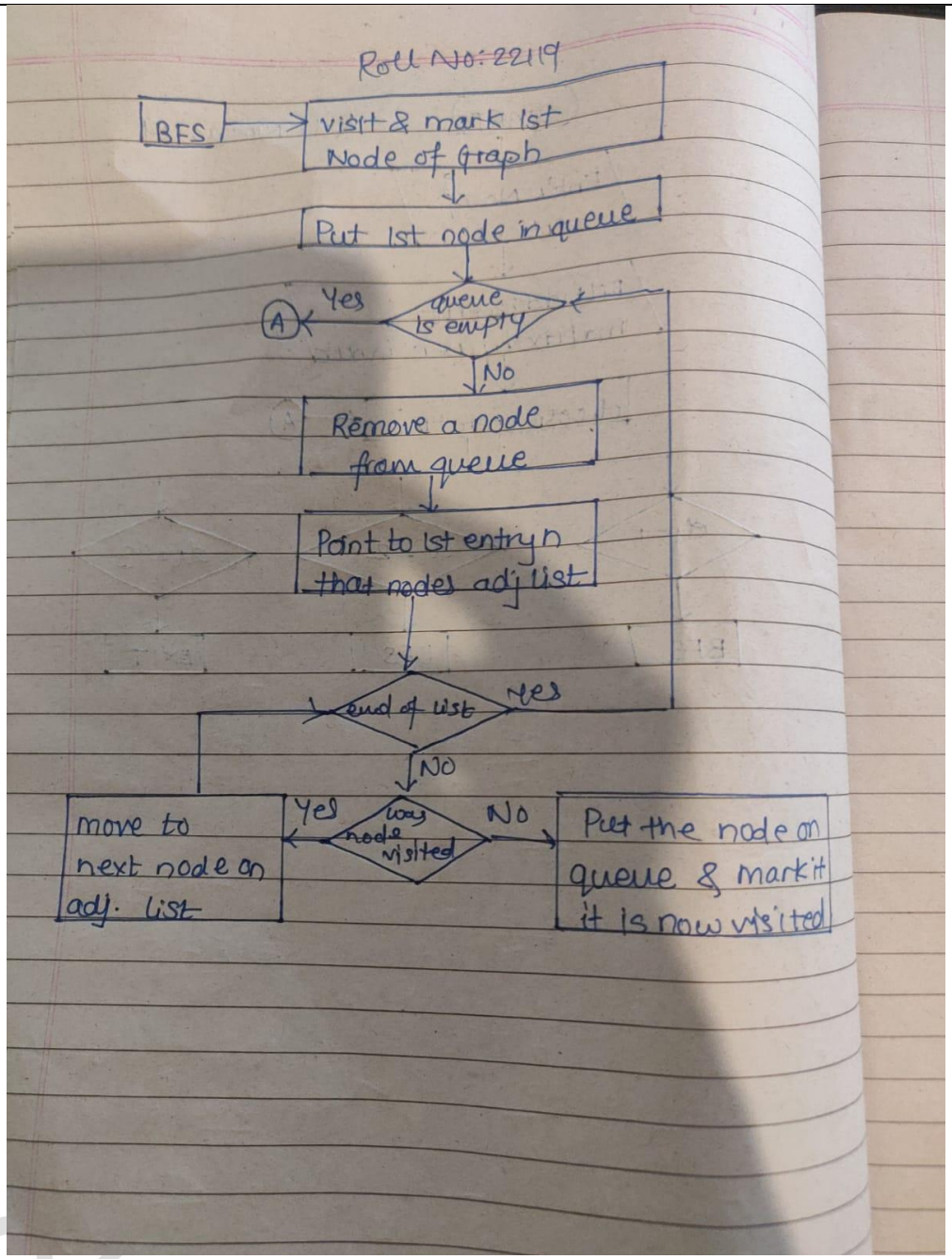
Adjacency Matrix
It is a two-dimensional array with Boolean flags.

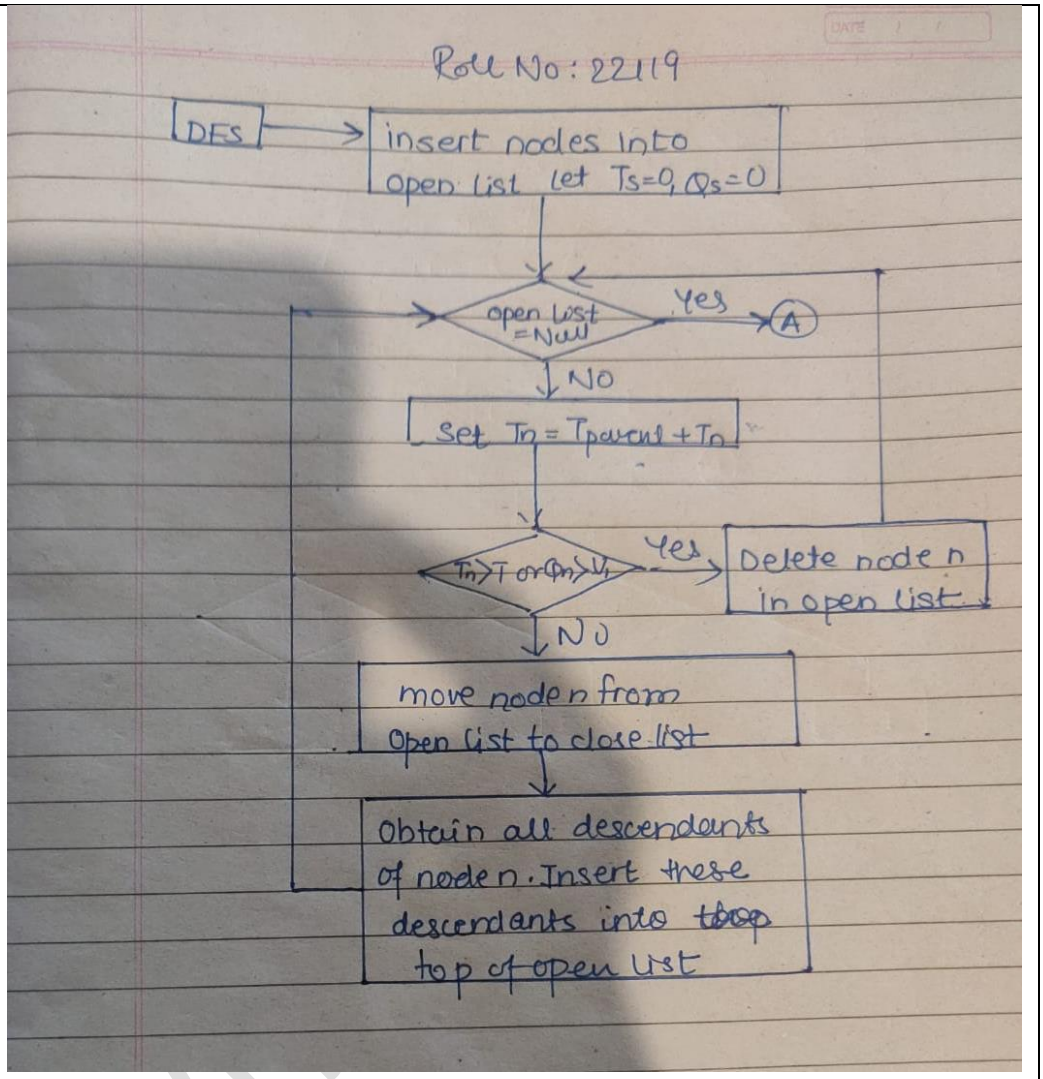
Graph Traversal
The breadth first search (BFS) and the depth first search (DFS) are the two algorithms used for traversing and searching a node in a graph. They can also be used to find out whether a node is reachable from a given node or not.

Algorithm	**Breadth First Search (B.F.S)** **Step 1**: Push the root node in the Queue. **Step 2**: Loop until the queue is empty. **Step 3**: Remove the node from the Queue. **Step 4**: If the removed node has unvisited child nodes, mark them as visited and insert the unvisited children in the queue. **Depth First Search (D.F.S)** **Step 1**: Push the root node in the Stack. **Step 2**: Loop until stack is empty. **Step 3**: Peek the node of the stack. **Step 4**: If the node has unvisited child nodes, get the unvisited child node, mark it as traversed and push it on stack. **Step 5**: If the node does not have any unvisited child nodes, pop the node from the stack.

Flow-chart	

START
Roll No: 22119
Enter No of vertices
Fill adjecent matrix by user input
choose operation
A
ch=1
ch=2
ch=3
BFS
DFS
EXIT


Roll No: 22119
BFS
visit & mark 1st Node of Graph
Put 1st node in queue
A
Yes
queue is empty
No
Remove a node from queue
Point to 1st entry in that nodes adj list
End of list
Yes
No
move to next node on adj. list
Yes
Was node visited
No
Put the node on queue & mark it it is now visited


Roll No: 22119
DES
insert nodes into open list let Ts=0, Qs=0
open list = Null
Yes
A
NO
set Tn = Tparent + Tn
Tn > T or Qn > Vt
Yes
Delete node n in open list
NO
move node n from Open list to close list
Obtain all descendants of node n. Insert these descendants into top of open list

ERROR	None
REMEDY	None
CONCLUSION:	
	• Hence, we have learnt the concept of nonlinear data structure • BFS traversal to visit graph nodes using queue structure done • DFS traversal to visit graph nodes using queue structure
REFERENCES:	
	1) Seymour Lipschutz, Data Structure with C, Schaum's Outlines, Tata McGrawHill 2) Yedidyah Langsam – Data structures using C and C++ - PHI Publications (2nd Edition). 3) Yashavant Kanetkar, Data Structures Through C, BPB Publication, 2nd Edition

Continuous Assessment			**Assessed By**
RPP (5)	**ARR (5)**	**Total (10)**	**Signature:**
			Date: