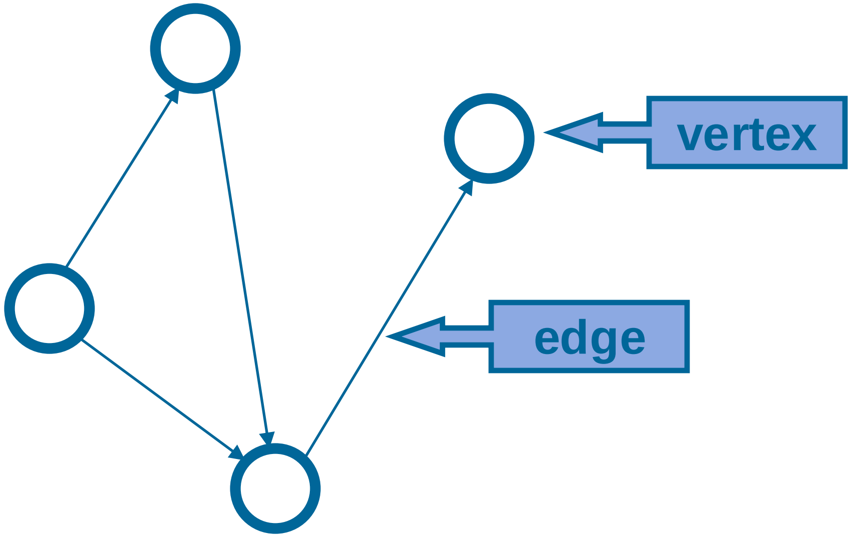
Data Structures and Algorithms – Lab 7

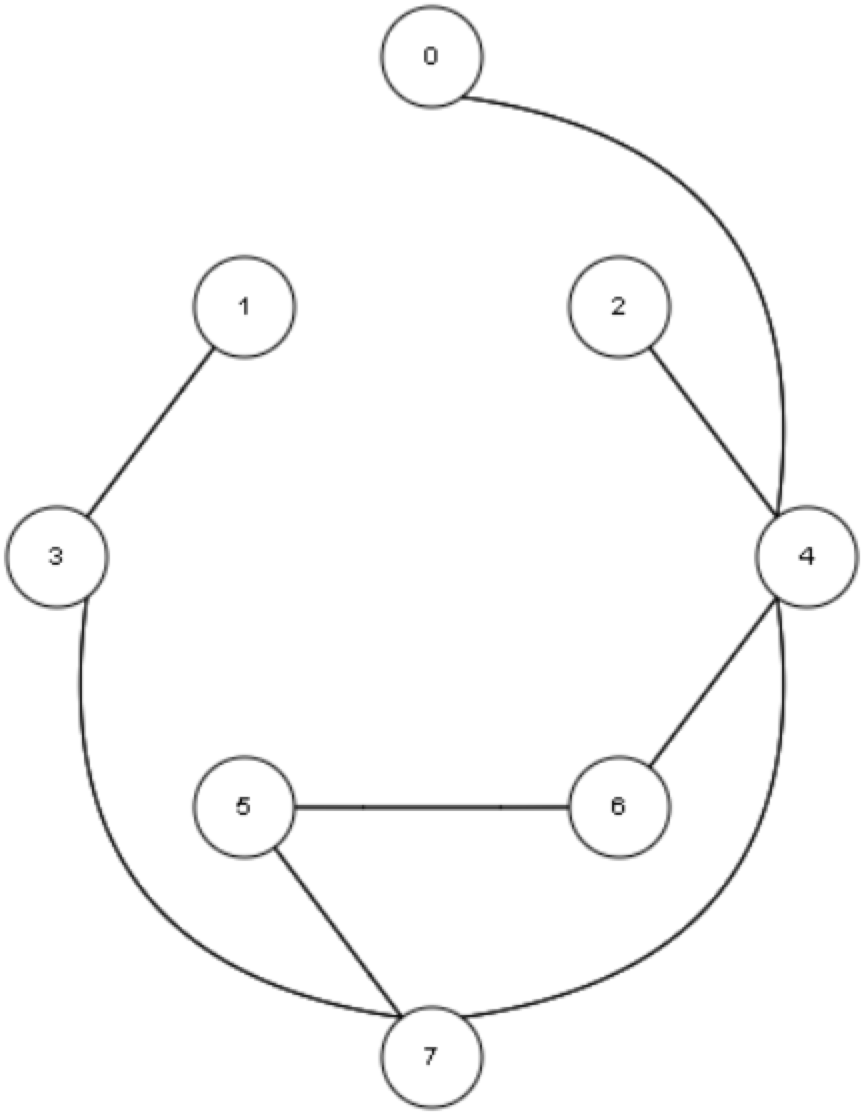
What is a graph?

- A set of vertices and edges
  - Directed/Undirected
  - Weighted/Unweighted
  - Cyclic/Acyclic

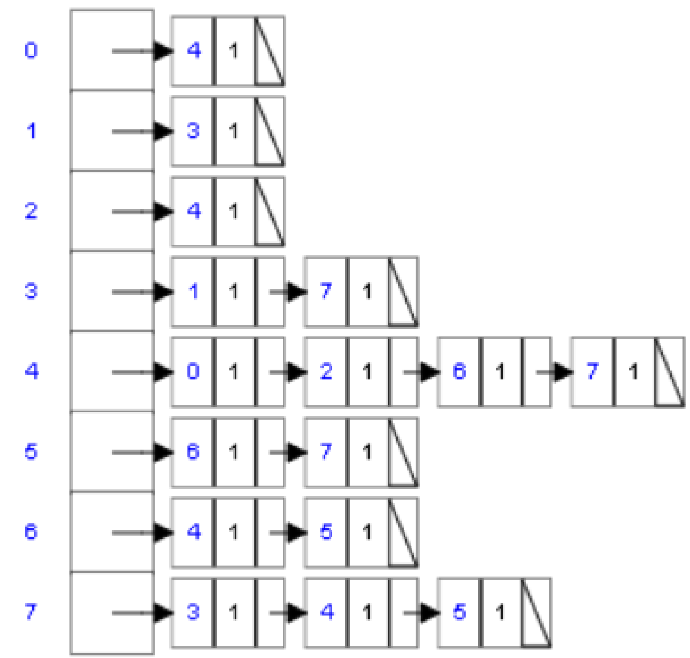


Representation of Graphs

- Adjacency Matrix
  - A V x V array, with matrix[i][j] storing whether there is an edge between the i^{th} vertex and the j^{th} vertex
- Linked List of Neighbours
  - One linked list per vertex, each storing directly reachable vertices

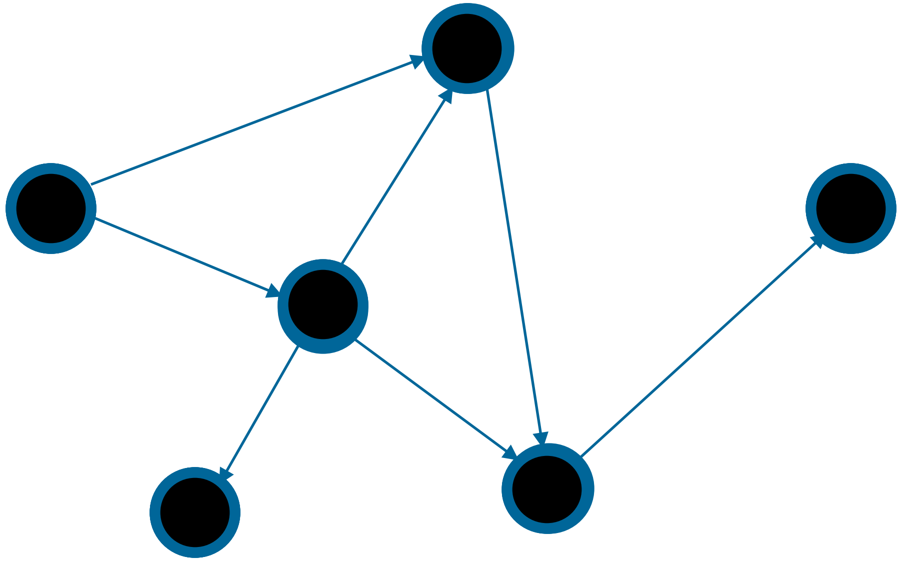


	0	1	2	3	4	5	6	7
0					1			
1				1				
2					1			
3		1						1
4	1		1				1	1
5							1	1
6					1	1		
7				1	1	1		



Depth-First Search (DFS)

- Strategy: Go as far as you can (if you have not visit there), otherwise, go back and try another way



Implementation

DFS (vertex **u**) {

    mark **u** as *visited*

    for each vertex **v** directly reachable from **u**

        if **v** is *unvisited*

            DFS (**v**)

}

```
DFS(4)
    DFS(0)
    DFS(2)
    DFS(6)
        DFS(5)
            DFS(7)
                DFS(3)
                    DFS(1)
```

- Initially all vertices are marked as *unvisited*

Application of DFS: Topological Sort

- Topological order:
  A numbering of the vertices of a directed acyclic graph such that every edge from a vertex numbered i to a vertex numbered j satisfies i<j
- Topological Sort:
  Finding the topological order of a directed acyclic graph

Example: Teacher's Problem

- Emily wants to distribute candies to *N* students one by one, with a rule that if student A is teased by B, A can receive candy before B.
- Given lists of students teased by each students, find a possible sequence to give the candies

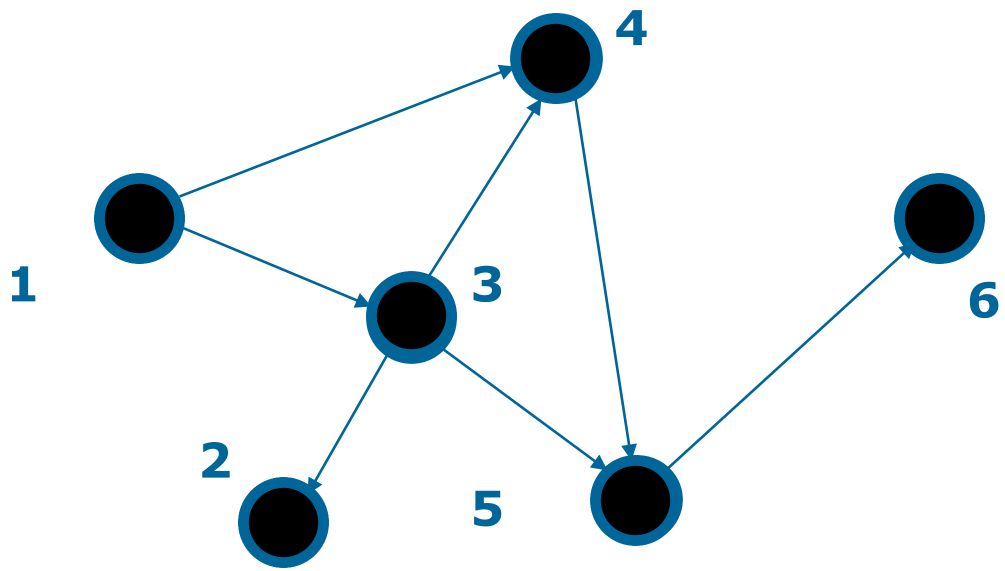
Breadth-First Search (BFS)

- Instead of going as far as possible, BFS tries to search all paths.
- BFS makes use of a queue to store visited (but not dead) vertices, expanding the path from the earliest visited vertices.

Simulation of BFS

- Queue:

1	4	3	5	2	6



Implementation

```
while queue Q not empty
    dequeue the first vertex u from Q
    for each vertex v directly reachable from u
        if v is unvisited
            enqueue v to Q
            mark v as visited
```

BFS(4)

4 0 2 6 7 5 3 1

- Initially all vertices except the start vertex are marked as *unvisited* and the queue contains the start vertex only

Application of BFS: Shortest Path

- If all edges have the same cost, we find the minimum distance between two nodes A and B by performing a BFS from node A and stop when node B was found.

Example: The travelling salesman problem is the problem of finding the shortest path that goes through every vertex exactly once, and returns to the start

There is more…

- Other Graph Searching Algorithms:
  - Bidirectional search (BDS)
  - Iterative deepening search (IDS)

Graph Modeling

- Conversion of a problem into a graph problem
- Essential in solving most graph problems

Basics of graph modeling

- Identify the vertices and the edges
- Identify the objective of the problem
- State the objective in graph terms
- Implementation:
  - construct the graph from the input instance
  - run the suitable graph algorithms on the graph
  - convert the output to the required format

Well-known Applications

- Social networks
- The salesman problem
- The timetable problem

Ex. 1

- Open adjacencymatrix.cpp and solve the exercises marked with ///Task
  - ///Task: correct the constructor argument based on the number of the vertices from the ppt from the lab
  - ///Task: complete the adding edges based on the ppt from the lab
  - ///Task: apply DFS from vertex 4 and BFS from vertex 4

Exercise 2

- Let’s consider un undirected graph, representing a social network. Given an user, display all his friends (or information about them) having the degree <=N (N is given).
- A is friend with B if there is an edge between A and B; we say that the degree of friendship is 1. Friends of friends have the degree of friendship 2. Use the matrix representation of graphs from Ex. 1.

Exercise 3

- Check if a graph is bipartite and if so, display the components of those two sets A and B. Use the matrix representation of graphs from Ex. 1.
- Check your code for the following graphs:
  - G1=({ 1,2,3,4,5,6,7,8,9},{ 12, 13, 45, 56, 75, 24, 58, 79, 43, 89})
  - G2=({ 1,2,3,4,5,6,7,8,9},{ 12, 13, 45, 56, 75, 24, 58, 79, 43, 89,47})

Tips

- In the mathematical field of graph theory, a bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint sets and such that every edge connects a vertex in to one in ; that is, and are each independent sets. Equivalently, a bipartite graph is a graph that does not contain any odd-length cycles.(Wikipedia)

- Use BFS:

http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/breadthSearch.htm