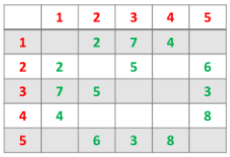
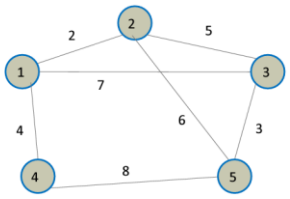
GRAPH REPRESENTATION

Adjacency Matrix



	1	2	3	4	5
1		2	7	4	
2	2		5		6
3	7	5			3
4	4				8
5		6	3	8	



Space Complexity

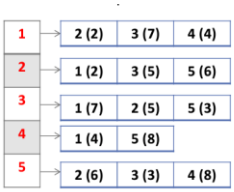
$O(V^2)$, where V is the number of vertices

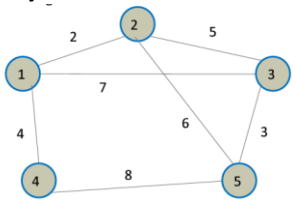
Time Complexity

O(1), to check edges exist

O(V) to get all the neighbours of a vertex

Adjacency List





Space Complexity

O(V+E), where V is the number of vertices and E is the number of edges

Time Complexity

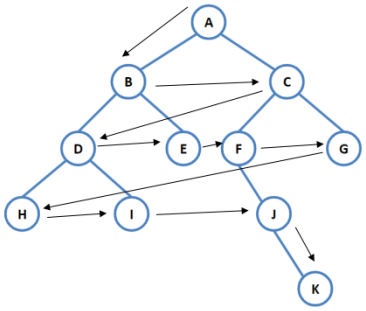
O(X) to check if edges exist

O(X) to get all the neighbours of a vertex

*X means the number of adjacent vertices of a vertex

BFS & DFS

Breadth First Search(BFS)



- Go Wide
- A->B->C->D->E->F->G->H->I->J->K
- Implemented using a queue(FIFO)

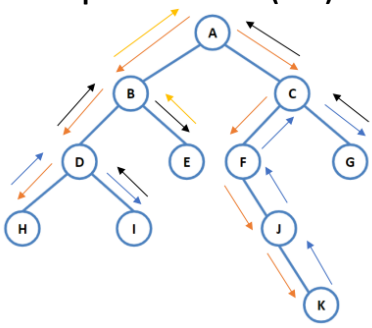
Space Complexity

O(V+E), where V is the number of vertices and E is the number of edges. O(V) for the discovered queue, and O(V+E) for graph representation using adjacency list

Time Complexity

O(V+E), because each vertex is visited once and each edge is visited once or twice depending if the graph is directed or undirected

Depth First Search(DFS)



- Go Deep
- A->B->D->H->I->E->C->F->J->K->G
- Implemented using a stack(LIFO)

Space Complexity

O(V+E), where V is the number of vertices and E is the number of edges. O(V) for the discovered stack, and O(V+E) for graph representation using adjacency list

Time Complexity

O(V+E), because each vertex is visited once and each edge is visited once or twice depending if the graph is directed or undirected

2021 Semester 1

Graph Representation

Question 16

For each of the following operations, determine its time complexity.

In this question,

3 Marks

- *V* refers to the number of vertices in the graph
- *E* refers to the number of edges in the graph
- *N(x)* refers to the number of neighbors of vertex-*x*.

Assume that in the adjacency list representation, the interior lists are unsorted.

Determining if an edge from *u* to *v* exists in an adjacency matrix	· O(V+E) · O(N(u)) · O(V^2) · O(1) · O(V)
Determining if an edge from *u* to *v* exists in an adjacency list	· O(V+E) · O(N(u)) · O(V^2) · O(1) · O(V)
Finding all neighbors of *u* in an adjacency matrix	· O(V+E) · O(N(u)) · O(V^2) · O(1) · O(V)
Finding all neighbors of *u* in an adjacency list	· O(V+E) · O(N(u)) · O(V^2) · O(1) · O(V)
Performing a complete BFS traversal in an adjacency matrix	· O(V+E) · O(N(u)) · O(V^2) · O(1) · O(V)
Performing a complete BFS traversal in an adjacency list	· O(V+E) · O(N(u)) · O(V^2) · O(1) · O(V)

- Determining if an edge from *u* to *v* exists in an adjacency matrix is O(1).
- Determining if an edge from *u* to *v* exists in an adjacency list is O(N(u)).
- Finding all neighbours of *u* in an adjacency matrix is O(V).
- Finding all neighbours of *u* in an adjacency list is O(N(u)).
- Performing a complete BFS traversal in an adjacency matrix is $O(V^2)$. This is because for BFS, for each vertex we need to get all the neighbours of the vertex. In an adjacency list representation, finding all the neighbours for each vertex is O(V+E), but for adjacency matrix representation, finding all the neighbours for each vertex is $O(V^2)$.
- Performing a complete BFS traversal in an adjacency list is O(V+E).

2022 Semester 1

For each of the following operations, determine its **worst-case big-Θ complexity**.

In this question,

2 Marks

- The graph *G* is a directed weighted graph.
- *V* refers to the number of vertices in the graph.
- *E* refers to the number of edges in the graph.
- *N(A)* refers to the number of neighbors of vertex *A*.

Assume that in the adjacency list representation, the **interior lists are unsorted.**

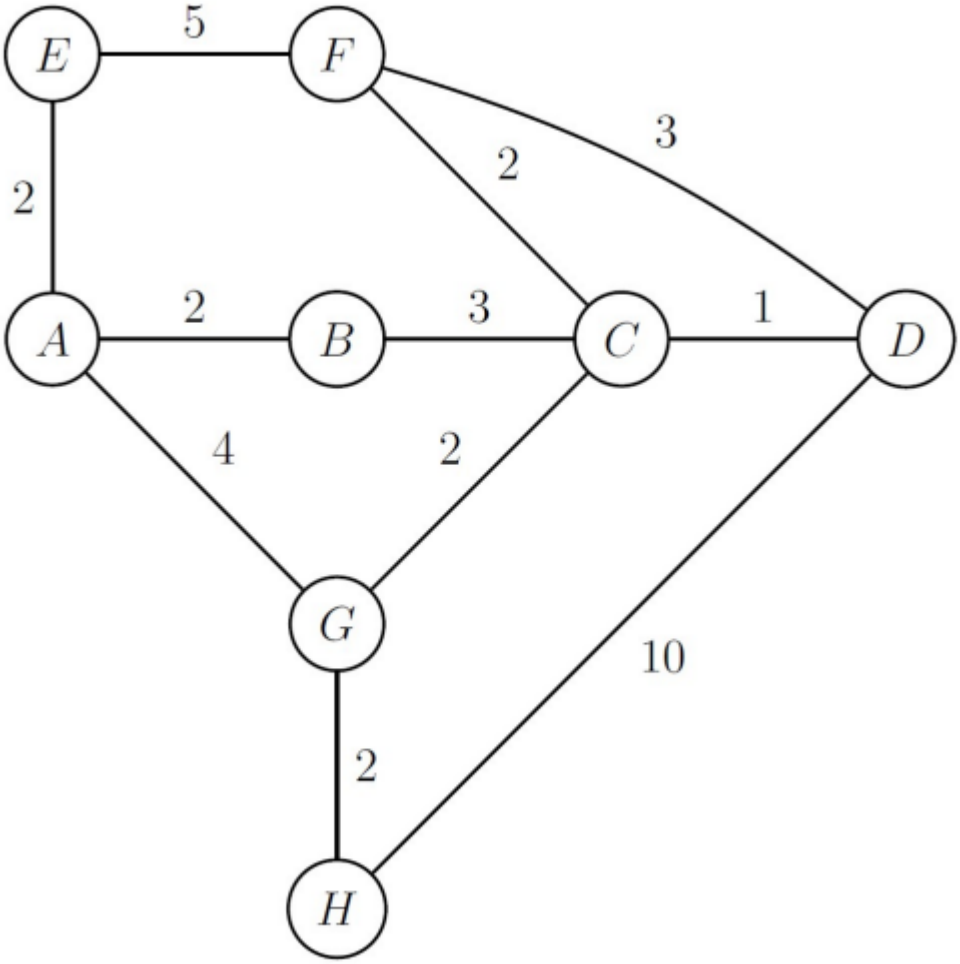
Time complexity to obtain all incoming edges for vertex *A* in an adjacency matrix representation.	· Θ(log V) · Θ(V) · Θ(V^2) · Θ(N(A)) · Θ(V+E) · Θ(log E) · Θ(1) · Θ(E)
Time complexity to determine if an edge from vertex *A* to vertex *B* exists in an adjacency list representation.	· Θ(log V) · Θ(V) · Θ(V^2) · Θ(N(A)) · Θ(V+E) · Θ(log E) · Θ(1) · Θ(E)
Time complexity to determine if an edge from vertex *A* to vertex *B* exists in an adjacency matrix representation.	· Θ(log V) · Θ(V) · Θ(V^2) · Θ(N(A)) · Θ(V+E) · Θ(log E) · Θ(1) · Θ(E)
Time complexity to obtain all incoming edges for vertex *A* in an adjacency list representation.	· Θ(log V) · Θ(V) · Θ(V^2) · Θ(N(A)) · Θ(V+E) · Θ(log E) · Θ(1) · Θ(E)

- Time complexity to obtain all incoming edges for vertex A in an adjacency matrix representation is O(V). We can find all incoming edges for a vertex by looping through the rows.

- Time complexity to determine if an edge from vertex A to vertex B exists in an adjacency list representation is O(N(A)). We just have to loop through all possible neighbours of vertex A.

- Time complexity to determine if an edge from vertex A to vertex B exists in an adjacency matrix representation is O(1). Just go to Matrix[A][B] and if it is None, then there is no edge from vertex A to vertex B.

- Time complexity to obtain all incoming edges for vertex A in an adjacency list representation is O(V+E). We just need to go through each vertex in the adjacency list and loop through it's corresponding list of neighbours to find all incoming edges for vertex A.



Perform a **depth-first search** on the graph given above **starting from node A**.

Whenever you have a choice between two nodes, **break ties in ascending alphabetical order**.

1 Mark

Associate each node with the **number for its position in the visiting order**, ie. node A should be associated with *i* if A is *i*-th visited node.

For this question, you can ignore the edge weights.

A->B->C->D->F->E->H->G

Perform a **breadth-first search** on the graph given above **starting from node A**.

Whenever you have a choice between two nodes, **break ties in ascending alphabetical order**.

1 Mark

What is the **maximum height** of the resulting tree from the breadth-first-search you have performed? **Just type the numerical answer.**

For this question, you can ignore the edge weights.

3. The height of a tree is the number of edges from the root node to the leaf node.