Question Bank Answers (Data Structures & Algorithms – IAE2)

> *CONTENT WARNING:*
> *READING THIS DOCUMENT MAY CAUSE SUDDEN BURSTS OF INTELLIGENCE.*
> *PROCEED WITH CAUTION.*

1. Explain graph with example
2. Explain types of graph with example
3. What is the difference between undirected & directed graph
4. Terminology related to graph
5. Explain Spanning Trees
8. What is communication network? Explain with example (same example) connected → 3&9
9. Explain adjacency matrix, Linked adjacency list, Array adjacency lists
10. What are different types of searching techniques
11. Explain what is hashing function & its types / examples also
12. What is collision resolution?
13. What is sorting & explain its algorithm & types of sorting
14. What do you mean by double hashing?
15. Explain BFS & DFS with algorithm

1. Explain graph with example

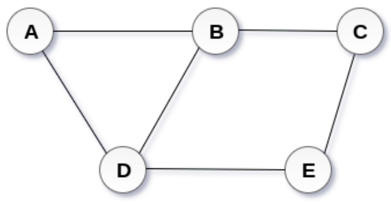
A graph is a data structure that consists of a set of vertices (or nodes) connected by edges. Graphs can be used to represent various real-world problems, such as social networks, transportation systems, and more.
A graph G can be defined as an ordered set G(V, E) where V(G) represents the set of vertices and E(G) represents the set of edges which are used to connect these vertices.

Components of a Graph

1. Vertices (Nodes): The individual elements of the graph.

2. Edges: The connections between the vertices. Edges can be directed (one-way) or undirected (two-way).

→ **Example:**
A Graph G(V, E) with 5 vertices (A, B, C, D, E) and six edges ((A,B), (B,C), (C,E), (E,D), (D,B), (D,A)) is shown in the following figure.



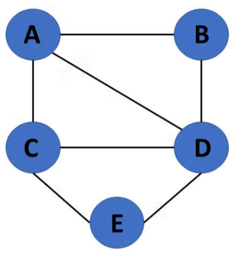
Undirected Graph

→ **Applications:**

Communication Network
Driving Distance / Time map
Street Map

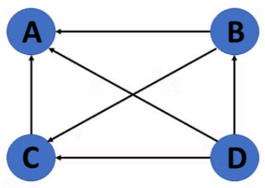
2. Explain types of graph with example

1. **Undirected Graphs**:
   A graph in which edges have no direction, i.e., the edges do not have arrows indicating the direction of traversal.



Example: A social network graph where friendships are not directional.

2. **Directed Graphs**: A graph in which edges have a direction, i.e., the edges have arrows indicating the direction of traversal.



Example: A web page graph where links between pages are directional.

3. What is the difference between undirected & directed graph

Undirected Graph	Directed Graph
A graph where edges have no direction, indicating a two-way relationship between vertices.	A graph where edges have a specific direction, indicating a one-way relationship between vertices.
Edges are represented with lines (e.g., A — B).	Edges are represented with arrows (e.g., A → B).
Each vertex has a single degree, counting all edges connected to it.	Each vertex has an in-degree (incoming edges) and an out-degree (outgoing edges).
Symmetric; the relationship is mutual (e.g., friendships).	Asymmetric; the relationship is not mutual (e.g., following on social media).
Generally simpler traversal, as edges can be traversed in both directions.	Traversal can vary based on edge direction, leading to different algorithms (e.g., DFS, BFS).
Used in modeling undirected friendships, road networks, and undirected relationships.	Used in modeling web links, task scheduling, and one-way streets.
Can have undirected cycles; simpler algorithms for detection (e.g., DFS).	Can have directed cycles; algorithms specifically for directed graphs are needed.

4. Terminology related to graph

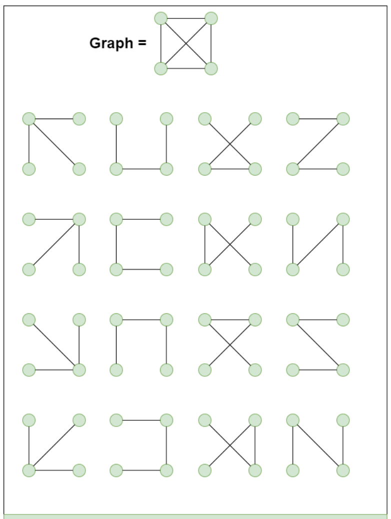
- **Vertex (Node)**: A fundamental part of a graph, representing an entity (e.g., a person in a social network).
- **Edge**: A connection between two vertices. It can be directed or undirected.
- **Degree**: The number of edges connected to a vertex.
    - **In-degree**: The number of incoming edges for a vertex (in directed graphs).

    - **Out-degree**: The number of outgoing edges from a vertex (in directed graphs).

- **Path**: A sequence of vertices connected by edges. A path can be simple (no repeated vertices) or contain cycles.
- **Cycle**: A path that starts and ends at the same vertex, with no other repeated vertices.

5. Explain Spanning Trees

A spanning tree is a subset of Graph G, such that all the vertices are connected using minimum possible number of edges. Hence, a spanning tree does not have cycles and a graph may have more than one spanning tree.



All Possible Spanning Trees of the Graph

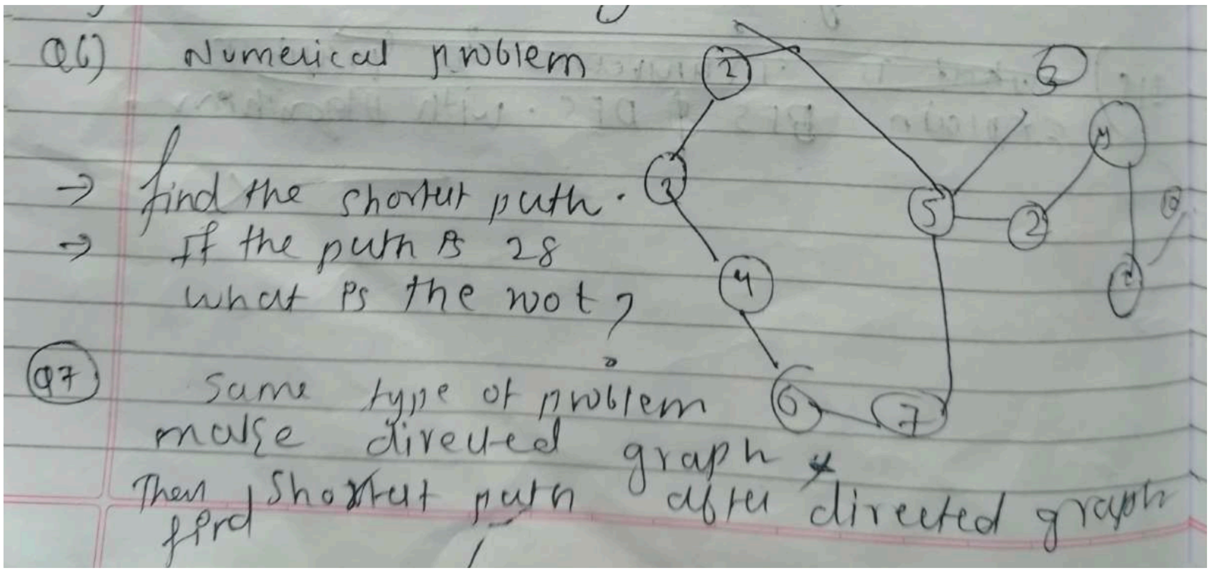
Properties of a spanning tree:

- A Spanning tree does not exist for a disconnected graph.
- For a connected graph having **N** vertices then the number of edges in the spanning tree for that graph will be **N-1**.
- A Spanning tree does not have any cycle.
- **Cayley's Formula:** It states that the number of spanning trees in a complete graph with N vertices is

$$N^{N-2}$$

- For example: N=4, then maximum number of spanning tree possible == 16 (shown in the above image).

$$4^{4-2}$$



Q6) Numerical problem

→ find the shortest path.

→ If the path is 28 what is the wot ?

Q7 Same type of problem make directed graph

Then find Shortest path after directed graph

2
1
4
6
7
5
2
3
4
2

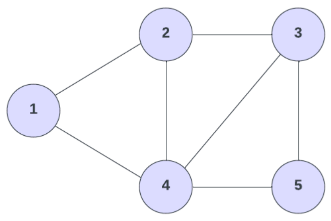
8. What is communication network? Explain with example (same example) connected → 3&9

9. Explain adjacency matrix, Linked adjacency list, Array adjacency lists

→ **Adjacency Matrix:**

Definition: An adjacency matrix is a two-dimensional array that stores the edges between two vertices as boolean values. The rows and columns of the matrix represent the vertices of the graph.

Let's look at an example:



$$\begin{array}{cc} & j \\ i & \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 & 1 & 0 \\ 3 & 0 & 1 & 0 & 1 & 1 \\ 4 & 1 & 1 & 1 & 0 & 1 \\ 5 & 0 & 0 & 1 & 1 & 0 \end{array} \end{array}$$

Formula: matrix[i][j]

matrix[1][2] = 1

matrix[2][5] = 0

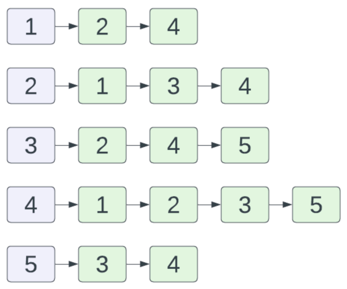
At the top we have a graph with some vertices and edges. At the bottom we have the adjacency matrix representation of the graph. The rows and columns represent the vertices of the graph. If we look at the vertex with the value of 1, we can see that it has edges to vertices 2 and 4. So we put a 1 in the first row for 2 and 4. This is an undirected graph, so it goes the other way as well. So we can put a 1 in the 2 row and the 4 row for 1. If this were a directed graph, we would only put a 1 in the 1 row for 2 and 4.

The formula to see if a vertex `i` is connected to vertex `j` is `matrix[i][j]`. `i` is the row and `j` is the column. So `matrix[1][2]` is 1, which means that vertex 1 is connected to vertex 2. `matrix[2][1]` is also 1, which means that vertex 2 is connected to vertex 1.

→ Linked Adjacency List:

Definition: A linked adjacency list is a collection of linked lists or dynamic arrays, where each list corresponds to a vertex and contains a list of its adjacent vertices.

Example:
Using the same graph, the linked adjacency list might look like this:



→ **Array Adjacency List**

Definition: An array adjacency list is similar to a linked list but uses an array to store the neighbors of each vertex. Each index of the array corresponds to a vertex, and the elements at that index represent its adjacent vertices.

Example:

aList[1]→ [2][4]
aList[2]→ [1][3][4]
aList[3]→ [2][4][5]
aList[4]→ [1][2][3][5]
aList[5]→ [3][4]

10. What are different types of searching techniques

1. Linear Search

Description: This is the simplest searching technique. It involves checking each element in a list sequentially until the desired element is found or the list ends.

Time Complexity: O(n) in the worst case, where n is the number of elements.

Use Case: Suitable for unsorted or small datasets.

Example: Searching for a number in an unsorted array.

2. Binary Search

Description: This technique is more efficient than linear search but requires the data to be sorted. It divides the search interval in half repeatedly, checking whether the target value is greater than or less than the middle element.

Time Complexity: O(logn).

Use Case: Efficient for large, sorted datasets.

Example: Searching for a number in a sorted array.

3. Depth-First Search (DFS)

Description: A graph traversal technique that explores as far down a branch as possible before backtracking. It can be implemented using a stack or recursion.

Time Complexity: O(V+E), where V is vertices and E is edges.

Use Case: Useful for solving problems like maze solving, topological sorting, and pathfinding.

4. Breadth-First Search (BFS)

Description: This graph traversal method explores all neighbors at the present depth before moving on to nodes at the next depth level. It is typically implemented using a queue.

Time Complexity: O(V+E).

Use Case: Shortest path finding in unweighted graphs, social network analysis.

11. Explain what is hashing function & its types / examples also

A **hashing function** is a mathematical algorithm that transforms input data (often called a "key") into a fixed-size string of bytes, typically a hash code. The output is usually a "hash value" or "hash code." Hash functions are commonly used in various applications, such as data storage, retrieval, and integrity verification.

Characteristics of a Good Hashing Function

- **Deterministic**: The same input will always produce the same output.
- **Fast Computation**: It should be quick to compute the hash value for any given input.
- **Collision Resistance**: It should be difficult to find two different inputs that produce the same hash value.
- **Pre-image Resistance**: Given a hash value, it should be infeasible to reverse-engineer the original input.

Types of Hashing Functions

- **Cryptographic Hash Functions**:
  - **Description**: Designed to be secure and resist attacks. Used in security applications and protocols, such as SSL/TLS and digital signatures.

    - **Examples**:
        - **SHA-256 (Secure Hash Algorithm)**: Produces a 256-bit hash and is widely used in blockchain technology.
        - **MD5 (Message Digest Algorithm 5)**: Produces a 128-bit hash; however, it's considered broken and not secure for cryptographic purposes.
- **Non-Cryptographic Hash Functions**:
    - **Description**: Primarily used for data structures like hash tables. They prioritize speed and efficiency over security.
    - **Examples**:
        - **MurmurHash**: Known for its speed and good distribution; widely used in hash tables.
        - **FNV (Fowler–Noll–Vo)**: Simple and fast hash function often used in hash tables.

Checksum Functions:

- **Description**: Used for error-checking in data transmission. They provide a simple way to detect changes to raw data.
- **Examples**:
    - **CRC32 (Cyclic Redundancy Check)**: Commonly used in network communications and file storage to detect errors.
    - **Adler-32**: A checksum algorithm that is faster than CRC but less reliable.

Examples of Hash Functions

SHA-256:

- **Input**: "Hello, World!"
- **Output**: `a591a6d40bf420404a011733cfb7b190d62c65bf0bcda190ef3b28b691d3e4d`

MD5:

- **Input**: "Hello, World!"
- **Output**: `65a1055e3b2196d1060f2e0b6e35d6c1`

MurmurHash:

- **Input**: "Hello, World!"
- **Output**: This would vary based on the version of MurmurHash used, but it generally produces a quick hash value.

Applications of Hashing Functions

- **Hash Tables**: Used for efficient data retrieval by mapping keys to values.
- **Data Integrity Verification**: Ensures that data has not been altered (e.g., using checksums).
- **Digital Signatures**: Ensures the authenticity and integrity of messages.
- **Password Storage**: Hashing passwords before storing them to protect user data.

12. What is collision resolution?

Collision resolution is a technique used in hash tables to handle situations where two different keys hash to the same index or slot in the table. This is known as a **collision**. Effective collision resolution strategies are crucial for maintaining the efficiency and performance of hash tables.

Types of Collision Resolution Techniques

Chaining:

- **Description**: Each slot in the hash table contains a linked list (or another data structure) of all entries that hash to that index. When a collision occurs, the new key-value pair is simply added to the linked list at that index.

Open Addressing:

- **Description**: When a collision occurs, the algorithm searches for the next available slot using a probing sequence. The key-value pair is placed in the first empty slot found.

Types of Open Addressing: Linear Probing & Quadratic Probing

Separate Chaining with Balanced Trees:

- **Description**: Instead of using linked lists, each slot in the hash table contains a balanced tree (e.g., a red-black tree) to store the entries. This is often used when the number of collisions is high.

13. What is sorting & explain its algorithm & types of sorting

Sorting is the process of arranging the elements of a list or array in a specific order, typically in ascending or descending order. Sorting algorithms are used to reorder data in a systematic way.

Common Sorting Algorithms

i. **Bubble Sort**

   - **Description**: Repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process repeats until the list is sorted.
   - **Time Complexity**: $O(n^2)$

ii. **Selection Sort**

- **Description**: Divides the list into a sorted and an unsorted region. It repeatedly selects the smallest (or largest) element from the unsorted region and moves it to the end of the sorted region.
- **Time Complexity**: $O(n^2)$

iii. Insertion Sort

- **Description**: Builds a sorted array one element at a time by repeatedly taking the next element from the unsorted section and inserting it into the correct position in the sorted section.
- **Time Complexity**: $O(n^2)$

iv. Merge Sort

- **Description**: A divide-and-conquer algorithm that divides the list into halves, recursively sorts each half, and then merges the sorted halves.
- **Time Complexity**: $O(n \log n)$

v. Quick Sort

- **Description**: Another divide-and-conquer algorithm that selects a 'pivot' element and partitions the array into two sub-arrays, sorting the elements around the pivot.
- **Time Complexity**: Average case $O(n \log n)$, worst case $O(n^2)$

14. What do you mean by double hashing?

Double hashing is a collision resolution technique used in hash tables. It works by using two hash functions to compute two different hash values for a given key. The first hash function is used to compute the initial hash value, and the second hash function is used to compute the step size for the probing sequence.

Double hashing has the ability to have a low collision rate, as it uses two hash functions to compute the hash value and the step size. This means that the probability of a collision occurring is lower than in other collision resolution techniques such as linear probing or quadratic probing.

15. Explain BFS & DFS with algorithm

1. Breadth-First Search (BFS)

Description:
BFS explores all neighbors of a node before moving to the next level of nodes. It uses a queue to keep track of nodes that need to be explored.

Algorithm:

Initialize:

- Create a queue and enqueue the starting node.
- Create a set to keep track of visited nodes.

While the queue is not empty:

- Dequeue a node from the front of the queue.
- If it hasn't been visited:
  - Mark it as visited.
  - Process the node (e.g., print or store it).
  - Enqueue all unvisited neighbors of the node.

Depth-First Search (DFS)

Description:
DFS explores as far down a branch of the graph as possible before backtracking. It can be implemented using recursion or a stack.

Algorithm (using recursion):

Initialize:

- Create a set to keep track of visited nodes.

Define a recursive function that takes a node:

- If the node hasn't been visited:
  - Mark it as visited.
  - Process the node.
  - For each unvisited neighbor, call the function recursively.

Mohammed Anas Nathani