MOST IMPORTANT QUESTIONS 🔥 7-Mark Questions

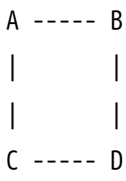
1. Define Graph Terminology

Introduction

Graph ek non-linear data structure hai jo vertices aur edges se milkar banta hai.

Graphs relationships ko represent karte hain.

Example of Graph



Important Terminologies

---

1. Vertex (Node)

Graph ka individual point vertex kehlata hai.

Example:

```
A, B, C, D
```

---

2. Edge

Do vertices ko connect karne wali line edge hoti hai.

Example:

```
A — B
```

---

3. Degree

Kisi vertex se connected edges ki total number.

Example:

```
Degree of A = 2
```

---

4. Path

Vertices ka sequence jisme vertices edges se connected ho.

Example:

```
A → B → D
```

---

5. Cycle

Closed path ko cycle kehte hain.

Example:

```
A → B → C → A
```

---

6. Connected Graph

Jisme har vertex kisi path se connected ho.

---

7. Weighted Graph

Edges ke saath weight/cost attached hota hai.

Example:

```
A --5-- B
```

---

8. Directed Graph

Edges ki direction hoti hai.

Example:

```
A → B
```

---

9. Undirected Graph

Edges ki direction nahi hoti.

Example:

```
A — B
```

Applications

- Social networks
- Maps
- Computer networking

Conclusion

Graph terminology graph concepts samajhne ke liye very important hai.

---

2. Differentiate Directed and Undirected Graph

Directed Graph

Directed graph me edges ki direction hoti hai.

Example:

```
A → B
```

Yaha:

A se B possible hai but B se A necessary nahi.

Undirected Graph

Undirected graph me edges ki direction nahi hoti.

Example:

```
A – B
```

Yaha:

A se B aur B se A dono possible.

Difference Between Directed and Undirected Graph

Directed Graph	Undirected Graph
Edges have direction	Edges have no direction
One-way connection	Two-way connection
Represented using arrows	Represented using lines
Used in web links	Used in social networks
In-degree and out-degree exist	Only degree exists

Applications of Directed Graph

- Instagram followers
- Web page links
- Traffic systems

Applications of Undirected Graph

- Facebook friendship
- Road maps
- Communication networks

Conclusion

Directed graph one-way relationship represent karta hai while undirected graph two-way relationship represent karta hai.

---

3. Explain Graph Representation

Introduction

Graph representation graph ko memory me store karne ka method hai.

Mainly two methods:

1. Adjacency Matrix
2. Adjacency List

---

1. Adjacency Matrix

2D matrix use hoti hai.

Suppose graph:

```
A – B
|
C
```

Matrix:

	A	**B**	**C**
A	0	1	1
B	1	0	0
C	1	0	0

Working

```
1 means edge exists
0 means no edge
```

Advantages

- Simple representation
- Fast edge checking

Disadvantages

- Memory wastage
- Not suitable for sparse graph

---

2. Adjacency List

Har vertex ke neighbors ki list store hoti hai.

Example:

```
A → B,C
B → A
C → A
```

Advantages

- Memory efficient
- Better for sparse graph

Disadvantages

- Searching slower

Difference

Adjacency Matrix	Adjacency List
Uses 2D array	Uses linked list
More memory	Less memory
Fast edge checking	Slower edge checking

Conclusion

Adjacency matrix dense graphs ke liye useful hai while adjacency list sparse graphs ke liye better hai.

---

4. Explain DFS Traversal

Introduction

DFS ka full form hai:

```
Depth First Search
```

DFS graph ko depth wise traverse karta hai.

DFS stack ya recursion use karta hai.

Working

DFS:

- pehle ek path me depth tak jata hai
- phir backtrack karta hai

Algorithm

```
1. Start vertex visit karo
2. Mark visited
3. Adjacent unvisited vertex par jao
4. Repeat until all visited
```

Example

Graph:

```
A → B → D
|
C
```

DFS Traversal:

```
A → B → D → C
```

Applications

- Cycle detection
- Maze solving
- Topological sorting

Advantages

- Less memory required
- Easy recursive implementation

Disadvantages

- Shortest path guarantee nahi

Conclusion

DFS graph traversal ka important technique hai jo depth wise traversal perform karta hai.

---

5. Explain BFS Traversal

Introduction

BFS ka full form hai:

```
Breadth First Search
```

BFS graph ko level by level traverse karta hai.

BFS queue use karta hai.

Working

BFS:

- first neighbors visit karta hai
- phir next level par jata hai

Algorithm

```
1. Start vertex enqueue karo
2. Visit and dequeue
3. Adjacent vertices enqueue karo
4. Repeat until queue empty
```

Example

Graph:

```
A → B
|
C → D
```

BFS Traversal:

```
A → B → C → D
```

Applications

- Shortest path
- GPS navigation
- Network broadcasting

Advantages

- Shortest path find karta hai
- Systematic traversal

Disadvantages

- More memory required

Difference Between DFS and BFS

DFS	BFS
Uses stack	Uses queue
Depth traversal	Level traversal
Less memory	More memory
Shortest path not guaranteed	Shortest path guaranteed

Conclusion

BFS level wise graph traversal karta hai aur shortest path problems me useful hai.

14-Mark Questions

Explain graph representation and traversal.

Explain DFS and BFS with example.

Explain Kruskal and Prim algorithms.

Explain Dijkstra shortest path algorithm.

Compare different graph algorithms. Explain question one by one in detail

1. Explain Graph Representation and Traversal

Introduction

Graph ek non-linear data structure hai jo vertices aur edges se milkar banta hai.

Graph representation ka matlab graph ko memory me store karna hai.

Graph traversal ka matlab graph ke vertices ko systematically visit karna hai.

Graph Representation

Graphs ko mainly do methods se represent karte hain:

---

1. Adjacency Matrix

Adjacency matrix ek 2D array hoti hai.

Suppose graph:

```
A – B
|
C
```

Matrix:

	A	**B**	**C**
A	0	1	1

	A	B	C
B	1	0	0
C	1	0	0

Working

```
1 means edge exists
0 means no edge
```

Advantages

- Simple representation
- Fast edge checking

Disadvantages

- Memory wastage
- Sparse graph ke liye inefficient

---

2. Adjacency List

Har vertex ke adjacent vertices ki list store hoti hai.

Example:

```
A → B,C
B → A
C → A
```

Advantages

- Less memory
- Better for sparse graphs

Disadvantages

- Searching slower

Graph Traversal

Traversal means graph ke nodes ko visit karna.

Mainly two traversals:

1. DFS
2. BFS

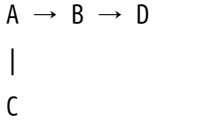
Depth First Search (DFS)

DFS graph ko depth wise traverse karta hai.

DFS stack ya recursion use karta hai.

DFS Example

Graph:



Traversal:

A → B → D → C

Breadth First Search (BFS)

BFS graph ko level by level traverse karta hai.

BFS queue use karta hai.

BFS Example

A → B → C → D

Applications of Traversal

- Path finding
- Networking
- GPS systems
- Web crawling

Conclusion

Graph representation graph ko memory me efficiently store karta hai while traversal graph ke vertices ko systematically visit karta hai.

---

2. Explain DFS and BFS with Example

Introduction

DFS aur BFS graph traversal algorithms hain.

DFS (Depth First Search)

DFS graph ko depth wise traverse karta hai.

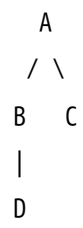
DFS stack ya recursion use karta hai.

DFS Algorithm

```
1. Start vertex visit karo
2. Mark visited
3. Adjacent unvisited vertex par jao
4. Repeat until all vertices visited
```

DFS Example

Graph:



DFS Traversal:

A → B → D → C

Advantages of DFS

- Less memory
- Recursive implementation easy

Disadvantages

- Shortest path guarantee nahi

Applications

- Cycle detection
- Maze solving
- Topological sorting

BFS (Breadth First Search)

BFS graph ko level wise traverse karta hai.

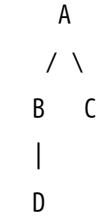
BFS queue use karta hai.

BFS Algorithm

```
1. Start vertex enqueue karo
2. Visit and dequeue
3. Adjacent vertices enqueue karo
4. Repeat until queue empty
```

BFS Example

Graph:



BFS Traversal:

```
A → B → C → D
```

Advantages of BFS

- Shortest path find karta hai
- Systematic traversal

Disadvantages

- More memory required

Difference Between DFS and BFS

DFS	BFS
Uses stack	Uses queue
Depth traversal	Level traversal
Less memory	More memory
No shortest path guarantee	Finds shortest path

Conclusion

DFS depth wise traversal karta hai while BFS level wise traversal karta hai.

---

3. Explain Kruskal and Prim Algorithms

Introduction

Kruskal aur Prim algorithms:

```
Minimum Spanning Tree (MST)
```

find karne ke liye use hote hain.

MST:

- all vertices include karta hai
- no cycle hota
- minimum total cost hoti hai

Kruskal Algorithm

Kruskal algorithm smallest edge select karta hai.

Steps

```
1. All edges sort karo
2. Smallest edge choose karo
3. Cycle avoid karo
4. Repeat until MST complete
```

Example

Edges:

```
AB=1
BC=2
AC=3
```

MST:

```
AB + BC
```

Total cost:

```
3
```

Advantages

- Simple implementation
- Sparse graph ke liye efficient

Disadvantages

- Sorting required

Prim Algorithm

Prim algorithm ek vertex se start karta hai aur minimum edge choose karta hai.

Steps

```
1. Start vertex select karo
2. Minimum edge choose karo
3. New vertex add karo
4. Repeat until all vertices included
```

Example

Start from A:

```
Select minimum connected edges
```

Advantages

- Dense graph ke liye better

Disadvantages

- More complex

Difference Between Kruskal and Prim

Kruskal	Prim
Edge based	Vertex based
Uses sorting	Uses priority queue
Best for sparse graph	Best for dense graph
May form forest initially	Always connected

Conclusion

Kruskal aur Prim algorithms MST banane ke liye use hote hain. Kruskal edge based hai while Prim vertex based hai.

---

4. Explain Dijkstra Shortest Path Algorithm

Introduction

Dijkstra algorithm shortest path find karta hai.

Used in:

- Google Maps
- GPS
- Routing systems

Working Principle

Algorithm minimum distance vertex select karta hai aur distances update karta hai.

Algorithm

```
1. Start vertex choose karo
2. Distance = 0
3. Remaining distances = infinity
4. Minimum distance vertex select karo
5. Adjacent distances update karo
6. Repeat until all vertices processed
```

Example

Suppose:

```
A-B = 2
A-C = 5
B-C = 1
```

Shortest path A to C:

```
A → B → C
```

Cost:

```
3
```

Advantages

- Fast shortest path finding
- Efficient routing

Disadvantages

- Negative weights handle nahi karta

Applications

- GPS
- Networking
- Airline routes

Conclusion

Dijkstra algorithm weighted graph me shortest path efficiently find karta hai.

---

5. Compare Different Graph Algorithms

Introduction

Different graph algorithms different purposes ke liye use hote hain.

Comparison Table

Algorithm	**Purpose**	**Data Structure Used**	**Main Use**
DFS	Depth traversal	Stack	Cycle detection

Algorithm	Purpose	Data Structure Used	Main Use
BFS	Level traversal	Queue	Shortest path
Kruskal	MST	Sorting	Sparse graphs
Prim	MST	Priority Queue	Dense graphs
Dijkstra	Shortest path	Priority Queue	Routing

DFS Features

- Depth wise traversal
- Recursive possible
- Less memory

BFS Features

- Level wise traversal
- Finds shortest path
- More memory

Kruskal Features

- Edge based MST
- Uses sorting

Prim Features

- Vertex based MST
- Better for dense graph

Dijkstra Features

- Finds shortest path
- No negative weight support

Applications

- Networking
- GPS
- Social media
- Web crawling

Conclusion

Different graph algorithms different applications ke liye useful hote hain. DFS and BFS traversal ke liye, Kruskal and Prim MST ke liye aur Dijkstra shortest path ke liye use hota hai.