UNIT–4 NOTES

GRAPHS IN DATA STRUCTURE

---

1. Introduction to Graph

Graph ek non-linear data structure hai jo:

```
Vertices (Nodes)
```

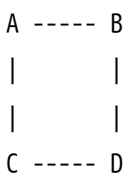
aur

```
Edges
```

se milkar banta hai.

Graph objects ke relationship ko represent karta hai.

Example of Graph



Here:

- A, B, C, D = Vertices
- Lines = Edges

Applications of Graph

- Social networks
- Google Maps
- Computer networks
- Routing systems
- Web page linking

Terminologies of Graph

---

1. Vertex

Graph ka node.

Example:

```
A, B, C
```

---

2. Edge

Vertices ko connect karne wali line.

Example:

```
A — B
```

---

3. Degree

Node se connected edges ki number.

---

4. Path

Vertices ka sequence.

Example:

```
A → B → D
```

---

5. Cycle

Closed path.

Example:

```
A → B → C → A
```

---

6. Connected Graph

Har vertex kisi na kisi path se connected ho.

---

7. Weighted Graph

Edges ke saath weight/cost attached hota hai.

Classification of Graph

---

1. Directed Graph

Edges ki direction hoti hai.

Example:

```
A → B
```

Applications

- Instagram follow system
- Web page links

---

2. Undirected Graph

Edges ki direction nahi hoti.

Example:

```
A — B
```

Applications

- Facebook friendship
- Road maps

Other Types of Graph

---

1. Weighted Graph

Edge ke saath cost hoti hai.

Example

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

---

2. Unweighted Graph

Edges without weights.

---

3. Cyclic Graph

Contains cycle.

---

4. Acyclic Graph

No cycle.

Graph Representation

Graphs ko mainly do ways me represent karte hain:

---

1. Adjacency Matrix

2D matrix use hoti hai.

Example

```
   A B C
A  0 1 1
B  1 0 0
C  1 0 0
```

Advantages

- Simple representation
- Fast edge checking

Disadvantages

- Memory wastage

---

2. Adjacency List

Har vertex ke neighbors ki list.

Example

```
A → B,C
B → A
C → A
```

Advantages

- Memory efficient
- Better for sparse graphs

Disadvantages

- Searching slower

Graph Traversal

Traversal means graph ke vertices ko visit karna.

Types of Graph Traversal

---

1. Depth First Search (DFS)

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 visited
```

Example

```
A → B → D → C
```

Applications of DFS

- Cycle detection
- Maze solving
- Topological sorting

Advantages

- Less memory
- Simple recursive implementation

Disadvantages

- May take long path

---

2. Breadth First Search (BFS)

BFS graph ko level by level 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
```

Example

```
A → B → C → D
```

Applications of BFS

- Shortest path
- Network broadcasting
- GPS navigation

Advantages

- Finds shortest path
- Systematic traversal

Disadvantages

- More memory required

Difference Between DFS and BFS

DFS	BFS
Uses stack	Uses queue
Depth wise traversal	Level wise traversal
Less memory	More memory
May not find shortest path	Finds shortest path
Recursive possible	Queue required

Minimum Spanning Tree (MST)

MST ek weighted connected graph ka minimum cost spanning tree hota hai.

Properties of MST

- Includes all vertices
- No cycles
- Minimum total weight

Kruskal Algorithm

Kruskal MST banane ke liye smallest edge select karta hai.

Steps

```
1. Sort edges by weight
2. Select smallest edge
3. Avoid cycle
4. Repeat until MST complete
```

Advantages

- Simple implementation
- Efficient for sparse graphs

Disadvantages

- Sorting required

Prim’s Algorithm

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

Steps

```
1. Start from any vertex
2. Select minimum edge
3. Add new vertex
4. Repeat until all vertices included
```

Advantages

- Good for dense graphs

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

Dijkstra’s Shortest Path Algorithm

Dijkstra algorithm shortest path find karta hai.

Used in:

- Google Maps
- GPS
- Routing systems

Working

```
1. Start vertex select karo
2. Minimum distance vertex choose karo
3. Adjacent distances update karo
4. Repeat until all vertices processed
```

Advantages

- Fast shortest path finding
- Efficient routing

Disadvantages

- Negative weights handle nahi karta

Comparison of Graph Algorithms

Algorithm	Purpose
DFS	Depth traversal
BFS	Level traversal
Kruskal	MST
Prim	MST
Dijkstra	Shortest path

Applications of Graph

- Social networks
- Google Maps

- Airline routes
- Computer networking
- Web crawling
- Recommendation systems

Advantages of Graph

- Complex relationships represent karta hai
- Efficient routing
- Dynamic structure

Disadvantages

- Complex implementation
- Memory usage high

MOST IMPORTANT QUESTIONS 🔥

7-Mark Questions

1. Define graph terminology.
2. Differentiate directed and undirected graph.
3. Explain graph representation.
4. Explain DFS traversal.
5. Explain BFS traversal.
6. Define MST.
7. Explain Kruskal algorithm.
8. Explain Prim algorithm.
9. Explain Dijkstra algorithm.

14-Mark Questions

1. Explain graph representation and traversal.
2. Explain DFS and BFS with example.
3. Explain Kruskal and Prim algorithms.
4. Explain Dijkstra shortest path algorithm.
5. Compare different graph algorithms.