Graph Algorithms in DSA: Beginner to Advanced Guide

Subtitle: Learn to represent, traverse, and solve problems on graphs using BFS, DFS, and advanced graph techniques.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Graphs are essential data structures used to model networks, relationships, and dependencies. Understanding graph representation and traversal techniques is crucial for **competitive programming, interview problems, and real-world applications**.

Step 1: Graph Representation

- **Adjacency Matrix** → 2D array, good for dense graphs
- **Adjacency List** → array of lists, good for sparse graphs
- **Edge List** → list of all edges

Example (Adjacency List in JS):

```
let graph = {};

graph[0] = [1,2];

graph[1] = [0,3];

graph[2] = [0,3];

graph[3] = [1,2];
```

Step 2: Graph Traversals

1. Breadth-First Search (BFS)

- Explores neighbors level by level
- Use queue

```
function bfs(graph, start) {

  let visited = new Set();

  let queue = [start];
```

```
  visited.add(start);


  while (queue.length) {
    let node = queue.shift();
    console.log(node);
    for (let neighbor of graph[node] || []) {
      if (!visited.has(neighbor)) {
        visited.add(neighbor);
        queue.push(neighbor);
      }
    }
  }
}
```

2. Depth-First Search (DFS)

- Explores as far as possible along each branch
- Use recursion or stack

```
function dfs(graph, node, visited=new Set()) {
  if (visited.has(node)) return;
  visited.add(node);
  console.log(node);
  for (let neighbor of graph[node] || []) dfs(graph, neighbor, visited);
}
```

Step 3: Common Graph Problems

1. **Connected Components** → BFS/DFS
2. **Detect Cycle in Graph** → DFS or Union-Find
3. **Shortest Path in Unweighted Graph** → BFS
4. **Dijkstra / Bellman-Ford** → Weighted shortest paths
5. **Topological Sorting** → DAG only
6. **Minimum Spanning Tree** → Kruskal / Prim

Step 4: Graph Patterns

- **Single Source Shortest Path** → BFS / Dijkstra
- **All Pairs Shortest Path** → Floyd-Warshall
- **Cycle Detection** → DFS / Union-Find
- **Topological Order** → DFS or Kahn's Algorithm
- **Bipartite Check** → BFS / DFS coloring

Step 5: Common Mistakes

- Forgetting to mark nodes as visited → infinite loops
- Confusing directed vs undirected edges
- Not handling disconnected components
- Using wrong data structure for weighted/unweighted graphs

Step 6: Practice Plan for Graphs

Day 1:

- BFS and DFS traversal → 5 problems

Day 2:

- Connected components, cycle detection → 5–7 problems

Day 3:

- Shortest path problems → 5–7 problems

Day 4:

- Minimum spanning tree → 3–5 problems

Day 5:

- Topological sorting → 3–5 problems

Day 6–7:

- Mixed graph problems → 10 problems

Mini Exercises

1. Count connected components in a graph
2. Detect cycle in directed and undirected graphs
3. Find shortest path from source in unweighted graph
4. Check if a graph is bipartite

Key Takeaways

- Graphs model **networks and relationships**
- BFS and DFS are foundational for traversal
- Advanced problems include **shortest paths, MST, topological sort**
- Always check for **disconnected components and cycles**
- Practice improves **intuition for graph patterns**

Visit haas.dev for more DSA graph algorithm guides, problem sets, and interview preparation resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app