Breadth and Depth First Search Activity

Instructions

For each graph below, perform the requested traversal starting from the specified vertex. When a vertex has multiple unvisited neighbors, visit them in **numerical order** (smallest first). Provide both the traversal order and analyze the complexity.

Part 1: Understanding O(V + E) Complexity

Problem 1: Basic Complexity Analysis

Consider a graph G = (V, E) with |V| = 8 and |E| = 10.

Graph 1 (Adjacency List Representation):

```
0: [1, 3]
1: [0, 2, 4]
2: [1, 5]
3: [0, 4, 6]
4: [1, 3, 5, 7]
5: [2, 4]
6: [3, 7]
7: [4, 6]
```

a) BFS from vertex 0

What is the traversal order?

Answer:

How many vertices are visited?

Answer:

How many edges are examined?

Answer:

b) DFS traversal order from vertex 0

What is the traversal order?

Answer:

How many vertices are visited?

Answer:

How many edges are examined?

Answer:

c) Complexity Questions

In the worst case for a fully connected graph, how many vertices will BFS/DFS visit?

Answer:

In an adjacency list representation, how many times will each edge be examined during BFS?

Answer:

Why is the time complexity O(V + E) and not O(V × E)?

Answer:

Part 2: Disconnected Graphs

Problem 2: Multiple Components

Graph 2 (Adjacency List):

```
|V| = 10, |E| = 7

0: [1, 2]
1: [0, 3]
2: [0]
3: [1]
4: [5, 6]
5: [4]
6: [4, 7]
7: [6]
8: [9]
9: [8]
```

a) BFS from vertex 0

What is the traversal order?

Answer:

Which vertices are NOT reachable from 0?

Answer:

b) To visit ALL vertices, from which additional vertices must you start BFS/DFS?

Answer:

c) If you run BFS on the entire graph (starting from 0, then any unvisited vertex), does the complexity of BFS change?

Answer:

Part 3: Dense vs Sparse Graphs

Problem 3: Graph Density Analysis

Graph 3a (Sparse - Adjacency List):

```
|V| = 6, |E| = 5 (tree structure)

0: [1, 2]
1: [0, 3]
2: [0, 4, 5]
3: [1]
4: [2]
5: [2]
```

Graph 3b (Dense - Adjacency List):

```
|V| = 6, |E| = 15 (complete graph)

0: [1, 2, 3, 4, 5]
1: [0, 2, 3, 4, 5]
2: [0, 1, 3, 4, 5]
3: [0, 1, 2, 4, 5]
4: [0, 1, 2, 3, 5]
5: [0, 1, 2, 3, 4]
```

For both graphs, perform BFS from vertex 0:

a) Graph 3a BFS

What is the traversal order?

Answer:

b) Graph 3b BFS

What is the traversal order?

Answer:

c) Analysis

In Graph 3a (sparse), what is |E| in terms of |V|?

Answer:

In Graph 3b (dense), what is |E| in terms of |V|?

Answer:

For sparse graphs, O(V + E) ≈ O(?)

Answer:

For dense graphs, O(V + E) ≈ O(?)

Answer:

Part 4: Adjacency Matrix vs Adjacency List

Problem 4: Representation Comparison

Consider the following graph:

```
|V| = 5, |E| = 6

0: [1, 2]
1: [0, 2, 3]
2: [0, 1, 4]
3: [1, 4]
4: [2, 3]
```

a) Convert to Adjacency Matrix representation:

Answer:

```
    0  1  2  3  4
0 [              ]
1 [              ]
2 [              ]
3 [              ]
4 [              ]
```

b) BFS Complexity Analysis:

Using **Adjacency List**:

What is the time complexity to find all neighbors of a vertex v?

Answer:

Total time complexity of BFS using an adjacency list?

Answer:

Using **Adjacency Matrix**:

What is the time complexity to find all neighbors of a vertex v?

Answer:

Total time complexity of bfs using an adjacency matrix?

Answer:

c) Which representation is better for sparse graphs? Why?*

Answer:

Part 5: Application Preview

Problem 5: Distance and Reachability

Graph 5 (Adjacency List):

```
|V| = 9, |E| = 10

0: [1, 3]
1: [0, 2, 4]
2: [1, 5]
3: [0, 4, 6]
4: [1, 3, 7]
5: [2, 8]
6: [3]
7: [4, 8]
8: [5, 7]
```

a) Perform BFS from vertex 0 and record the distance (shortest path length) to each vertex:

Answer:

b) Why does BFS (but not DFS) give shortest paths in unweighted graphs?

Answer:

c) How would you modify BFS to actually compute and store these distances?

Answer:

Part 6: Cycle Detection

Problem 6: Understanding Graph Structure

Graph 6 (Adjacency List):

```
|V| = 7, |E| = 7

0: [1, 2]
1: [0, 3, 4]
2: [0, 5]
3: [1]
4: [1, 5, 6]
5: [2, 4]
6: [4]
```

a) Perform DFS and find the traversal order from vertex 0 then classify each edge as:

- **Tree edge** (T): Edge to an unvisited vertex OR
- **Back edge** (B): Edge to an ancestor in DFS tree

Answer:

DFS traversal order:

(0,1):

(0,2):

(1,3):

(1,4):

(2,5):

(4,5):

(4,6):

b) Does this graph contain a cycle? How can you tell from the DFS edge classification?

Answer:

Challenge Problems

Challenge 1: Worst-Case Analysis

For a graph with |V| = n vertices:

- What is the maximum number of edges in a simple undirected graph?
- What is the maximum number of edges in a simple directed graph?
- In the worst case, is $O(V + E) = O(V^2)$?

Challenge 2: Practical Complexity

A social network has 1 million users and 50 million friendships (undirected graph).

- What is |V|?
- What is |E|?
- Is this graph sparse or dense?
- If we use adjacency list, how much space do we need?
- If we use adjacency matrix, how much space do we need?
- For BFS/DFS, which representation is more efficient?

Summary: Key Takeaways

Concept	Key Point
BFS Complexity	O(V + E) because each vertex visited once, each edge examined constant times
Sparse Graphs	E = O(V), so complexity ≈ O(V)
Dense Graphs	E = $O(V^2)$, so complexity ≈ $O(V^2)$
Adjacency List	Best for sparse graphs: O(V + E) BFS/DFS
Adjacency Matrix	Best for dense graphs or edge queries: $O(V^2)$ BFS/DFS
Disconnected Graphs	Run BFS/DFS from each component: still O(V + E) total
Shortest Paths	BFS finds shortest paths in unweighted graphs
Cycle Detection	DFS finds cycles via back edges
Connected Components	Multiple BFS/DFS calls, count how many times started

---

Course content developed by Declan Gray-Mullen for WNEU with Claude