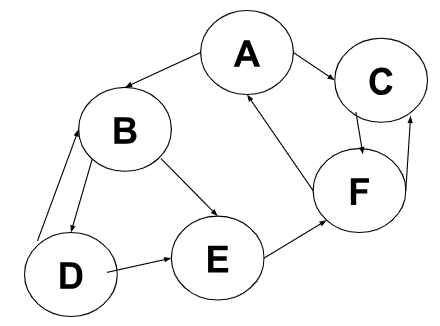
**University of Guelph**  
**School of Computer Science**  
CIS\*2520: Data Structures  
Fall 2024, Lab 7: Graph (Solution)  
Week of Nov. 18 – 22

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. Given the graph with the following vertices and edges:



1. Create the adjacency list representation for this graph.

Answer: Here is the adjacency list representation for a directed graph

A: B, C

B: D, E

C: F

D: B, E

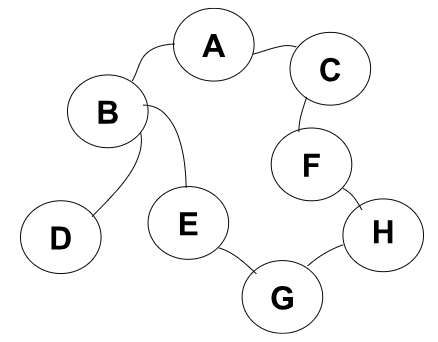
E: F

F: A, C

1. Create the adjacency matrix representation for this graph.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F |
| A | 0 | 1 | 1 | 0 | 0 | 0 |
| B | 0 | 0 | 0 | 1 | 1 | 0 |
| C | 0 | 0 | 0 | 0 | 0 | 1 |
| D | 0 | 1 | 0 | 0 | 1 | 0 |
| E | 0 | 0 | 0 | 0 | 0 | 1 |
| F | 1 | 0 | 1 | 0 | 0 | 0 |

1. Consider the following undirected graph with vertices A,B,C,D,E,F,G,H and edges:



1. Perform a Breadth-First Search (BFS) starting from vertex A. List the order in which vertices are visited.

Answer: BFS operates using the FIFO list.

Steps of BFS from Vertex A

Initialize: Start with A in the queue.

Visit A: Dequeue A and add its adjacent vertices B and C to the queue.

Visit B: Dequeue B and add its unvisited neighbors D and E to the queue.

Visit C: Dequeue C and add its unvisited neighbor F to the queue.

Visit D: Dequeue D; it has no unvisited neighbors.

Visit E: Dequeue E and add its unvisited neighbor G to the queue.

Visit F: Dequeue F and add its unvisited neighbor H to the queue.

Visit G: Dequeue G; it has no unvisited neighbors.

Visit H: Dequeue H; it has no unvisited neighbors.

The order in which vertices are visited in BFS starting from A:

A,B,C,D,E,F,G,H

1. Using the BFS traversal, determine the shortest path from vertex A to vertex H.

Answer: We find the shortest path by tracking the first time we encounter each vertex.

Start at A:

Queue: [A]

Paths: { A: [A] }

Visit A and add neighbors B and C:

Queue: [B, C]

Paths: { A: [A], B: [A, B], C: [A, C] }

Visit B and add neighbors D and E:

Queue: [C, D, E]

Paths: { A: [A], B: [A, B], C: [A, C], D: [A, B, D], E: [A, B, E] }

Visit C and add neighbor F:

Queue: [D, E, F]

Paths: { A: [A], B: [A, B], C: [A, C], D: [A, B, D], E: [A, B, E], F: [A, C, F] }

Visit D (no new neighbors).

Visit E and add neighbor G:

Queue: [F, G]

Paths: { A: [A], B: [A, B], C: [A, C], D: [A, B, D], E: [A, B, E], F: [A, C, F], G: [A, B, E, G] }

Visit F and add neighbor H:

Queue: [G, H]

Paths: { A: [A], B: [A, B], C: [A, C], D: [A, B, D], E: [A, B, E], F: [A, C, F], G: [A, B, E, G], H: [A, C, F, H] }

Visit H: Reached H with path [A, C, F, H], the shortest path.

The shortest path from A to H is: A → C → F → H

1. Represent the graph using an adjacency list.

Answer: Adjacency List:

A: B, C

B: A, D, E

C: A, F

D: B

E: B, G

F: C, H

G: E, H

H: F, G

1. Explain how BFS ensures that the shortest path is found in an unweighted graph like this one.

Answer: BFS explores vertices level by level, ensuring the shortest path is found as it visits all neighbors before moving to the next level. BFS uses a queue, processing vertices in the order of their distance from the source. The first time BFS reaches a vertex, it has taken the fewest steps, ensuring the shortest path in an unweighted graph.

1. What is the main difference in the search strategy between BFS and DFS? Choose one.
2. BFS explores as deeply as possible before backtracking, while DFS explores level by level.
3. BFS explores level by level, visiting all nodes at the current depth before moving to the next level, while DFS explores as deeply as possible along each branch before backtracking.
4. Both BFS and DFS explore nodes at the same depth before moving to the next depth.
5. DFS uses a queue, and BFS uses a stack to manage node exploration.

Answer: B

1. Consider a graph G with V vertices and E edges. What are the time and space complexities of Breadth-First Search (BFS) and Depth-First Search (DFS) in terms of V and E?

Answer:

BFS:

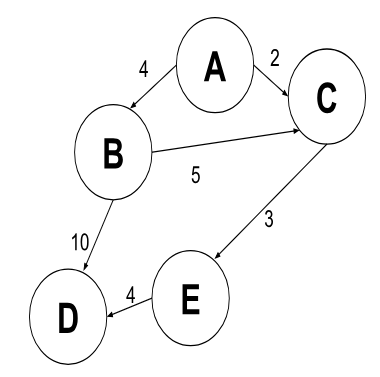
Time Complexity: O(V+E) (visits each vertex and edge once)

Space Complexity: O(V) (queue and visited vertices)

DFS:

Time Complexity: O(V+E) (visits each vertex and edge once)

Space Complexity: O(V) (stack and visited vertices)

1. Consider the following weighted, directed graph with vertices A,B,C,D,E and the following edges with their respective weights:
2. Use Dijkstra's algorithm to find the shortest path from vertex A to each of the other vertices in the graph. Show each step of the algorithm, including updates to the tentative distances and the final shortest path tree.

Answer: Start at vertex A ; Tentative Distances: { A: 0, B: ∞, C: ∞, D: ∞, E: ∞ }

Visit A:

Update neighbors:

B: 0 + 4 = 4

C: 0 + 2 = 2

Distances: { A: 0, B: 4, C: 2, D: ∞, E: ∞ }

Visit C (smallest distance):

Update neighbors:

E: 2 + 3 = 5

Distances: { A: 0, B: 4, C: 2, D: ∞, E: 5 }

Visit B:

Update neighbors:

D: 4 + 10 = 14

Distances: { A: 0, B: 4, C: 2, D: 14, E: 5 }

Visit E:

Update neighbors:

D: 5 + 4 = 9 (shorter than 14)

Distances: { A: 0, B: 4, C: 2, D: 9, E: 5 }

Visit D:

No updates needed.

Final Distances: { A: 0, B: 4, C: 2, D: 9, E: 5 }

Shortest Paths:

A →B: 4

A →C: 2

A →E: 5 (via C)

A →D: 9 (via C →D)

1. What is the shortest path from A to D and its total cost?

Answer: The shortest path from A →D is :

Path: A →C →E →D

Total cost: 2+3+4= 9

1. If an additional edge C→B with weight 1 were added, would this change the shortest path from A → D? Explain your answer.

Answer: No, adding an edge C →B with weight of 1 will not change the shortest path from A to D.

Total cost is 9 (reference to part b)

Cost with new edge:

A →C: 2

C →B: 1

B →D: 10

Cost = 2+1+10 = 13, which is greater than the existing path cost of 9

1. If a Directed Acyclic Graph (DAG) has multiple valid topological sorts, which of the following statements is true? Choose one.
2. All valid topological sorts will be identical.
3. Different valid topological sorts will have the same vertices but a different order.
4. Multiple topological sorts are only possible in graphs with cycles.
5. Only the first topological sort generated is valid.

Answer: B

1. In the source-removal algorithm for topological sorting, what happens to the outgoing edges of a vertex when it is removed from the graph? Choose one.
2. They are also removed from the graph.
3. They are left in the graph as self-loops.
4. They are added to the end of the result list.
5. They are reversed to point in the opposite direction.

Answer: A