Fundamental Algorithm Techniques
Problem Set #5 - Solution

Zhumakhanbet Akbota

1 Problem 1: Tree Definition Equivalences

We need to show that these 7 ways to define a tree all mean the same thing:

1. Connected graph with no cycles
2. One piece of a forest (forest = graph with no cycles)
3. Connected graph with at most $V - 1$ edges
4. Removing any edge breaks the connection
5. Graph with no cycles and at least $V - 1$ edges
6. Adding any edge creates a cycle
7. There is exactly one path between any two vertices

1.1 Simple Explanation

All these definitions describe the same thing: a tree. We show this by proving that definition (1) is the same as each of the others.

Let G be a graph with n vertices and m edges.

1.2 Proof: (1) $\Leftrightarrow$ (2)

If (1) then (2): A connected graph with no cycles is a tree. A forest is just a graph with no cycles, so a tree is one connected piece of a forest.

If (2) then (1): If it's one connected piece of a forest, it's connected and has no cycles (since the forest has no cycles).

1.3 Proof: (1) $\Leftrightarrow$ (3)

If (1) then (3): A connected graph with no cycles always has exactly $n - 1$ edges.

- With 1 vertex: 0 edges = $n - 1$ (correct)
- With more vertices: Every tree has at least one leaf (vertex with only one edge). Remove the leaf and its edge. You still have a tree with $n - 1$ vertices. By the same logic, it has $(n - 1) - 1$ edges. Add back the leaf: total is $n - 1$ edges.

If (3) then (1): A connected graph with $n - 1$ edges has no cycles. If it had a cycle, we could remove one edge from the cycle and still be connected, but then we'd have fewer than $n - 1$ edges, which is impossible for a connected graph.

1.4 Proof: (1) ⇔ (4)

If (1) then (4): In a connected graph with no cycles, every edge is needed. If you remove an edge between vertices u and v, they become disconnected. If they weren't disconnected, there would be another path between them, and that path plus the removed edge would form a cycle.

If (4) then (1): If removing any edge breaks the connection, the graph is connected. If it had a cycle, we could remove one edge from the cycle and still be connected, which contradicts (4).

1.5 Proof: (1) ⇔ (5)

If (1) then (5): From above, a tree has exactly $n-1$ edges, so at least $n-1$ edges. It has no cycles by definition.

If (5) then (1): A graph with no cycles and at least $n-1$ edges has exactly $n-1$ edges (more would create a cycle). With $n-1$ edges and no cycles, it must be connected (if not, some piece would have too few edges).

1.6 Proof: (1) ⇔ (6)

If (1) then (6): In a connected graph with no cycles, adding any edge creates a cycle. Why? Because there's already a path between the two vertices you connect, and that path plus the new edge makes a cycle.

If (6) then (1): If adding any edge creates a cycle, the graph has no cycles. If it weren't connected, we could add an edge between two separate pieces without creating a cycle, which contradicts (6).

1.7 Proof: (1) ⇔ (7)

If (1) then (7): In a connected graph with no cycles, there's a path between any two vertices. If there were two different paths, they would form a cycle together.

If (7) then (1): If there's exactly one path between any two vertices, the graph is connected. If it had a cycle, there would be two different paths between vertices on the cycle.

1.8 Conclusion

Since all definitions are equivalent to (1), they are all equivalent to each other.

2 Problem 2: Sparse Graph Representation

We have two graphs on vertices $\{A, B, C, D, E\}$ stored in CSC format.

What is CSC format?

- `col_pointers[i]` tells us where to start looking for edges from vertex i
- `row_indices` lists which vertices are connected
- For vertex i, look at positions `col_pointers[i]` to `col_pointers[i+1]-1` in `row_indices`

2.1 Graph 1 (Undirected)

Given:

- `col_pointers` = [0, 2, 5, 8, 11, 12]
- `row_indices` = [1, 2, 0, 2, 3, 0, 1, 3, 1, 2, 4, 3]

Step by step:

- Vertex A (0): positions 0-1 → rows [1, 2] → edges: A-B, A-C
- Vertex B (1): positions 2-4 → rows [0, 2, 3] → edges: B-A, B-C, B-D
- Vertex C (2): positions 5-7 → rows [0, 1, 3] → edges: C-A, C-B, C-D
- Vertex D (3): positions 8-10 → rows [1, 2, 4] → edges: D-B, D-C, D-E
- Vertex E (4): positions 11-11 → row [3] → edge: E-D

(a) Adjacency Matrix:

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Rows and columns: A, B, C, D, E

(b) Graph Visualization:

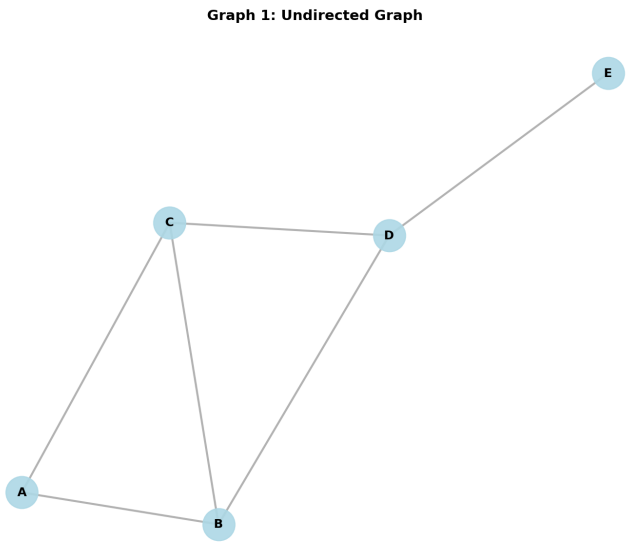

Figure 1: Graph 1: Undirected graph with vertices A, B, C, D, E. Edges: A-B, A-C, B-C, B-D, C-D, D-E.

2.2 Graph 2 (Directed)

Given:

- `col_pointers` = [0, 0, 2, 4, 5, 7]
- `row_indices` = [0, 3, 0, 1, 2, 1, 3]

Step by step:

- Vertex A (0): positions 0-(-1) → no edges (empty)
- Vertex B (1): positions 0-1 → rows [0, 3] → edges: B → A, B → D
- Vertex C (2): positions 2-3 → rows [0, 1] → edges: C → A, C → B
- Vertex D (3): positions 4-4 → row [2] → edge: D → C
- Vertex E (4): positions 5-6 → rows [1, 3] → edges: E → B, E → D

(a) Adjacency Matrix:

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Entry $[i, j] = 1$ means edge from column j to row i.

(b) Graph Visualization:

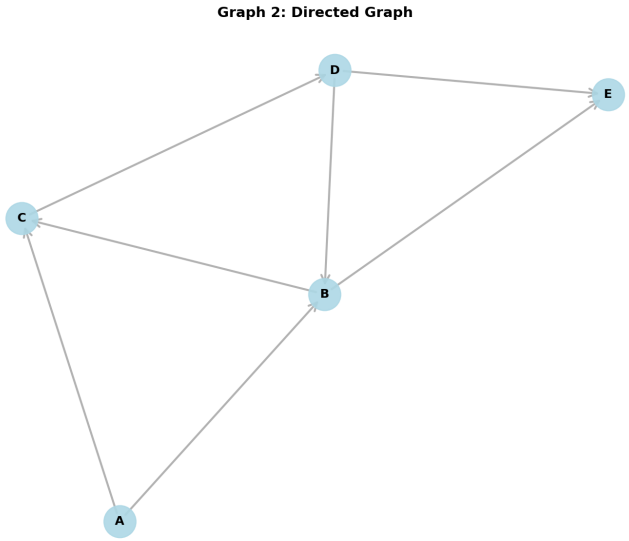

Figure 2: Graph 2: Directed graph with vertices A, B, C, D, E. Edges: B → A, B → D, C → A, C → B, D → C, E → B, E → D.

(c) Finding the Cycle:
Let's trace paths:

- Start at D: D → C
- From C: C → B
- From B: B → D
- Back to D: we have a cycle!

The unique cycle is: D → C → B → D
We can verify:

- D → C: exists (matrix entry [2,3] = 1)
- C → B: exists (matrix entry [1,2] = 1)
- B → D: exists (matrix entry [3,1] = 1)

This is the only cycle in the graph.