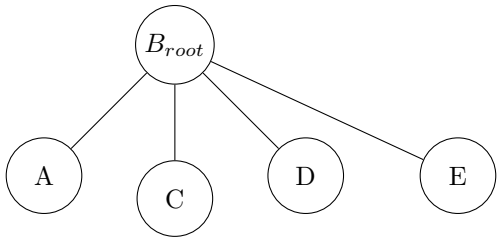
CS1800 Homework 6 Solutions

1 Build-a-Graph

I Acyclic Graph Where Every Pair of Nodes Has an Edge

This graph does not exist. An acyclic graph cannot have an edge between every pair of nodes, as this would necessarily create cycles among the nodes.

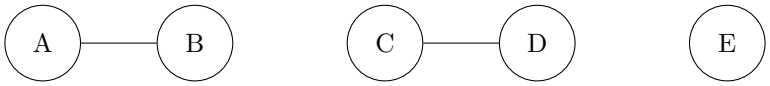
II Rooted Tree Where B is the Root



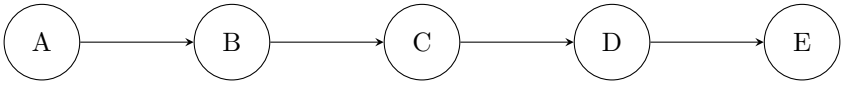
This structure satisfies the acyclic and tree properties, with B designated as the root node.

III Graph with No Cycles That is Not a Tree

A forest (a set of disjoint trees) meets this criterion. For example, with nodes A, B, C, D, and E, we could have two disjoint trees, such as $A - B$, $C - D$, and E, which is a graph without cycles and that is not a single tree.

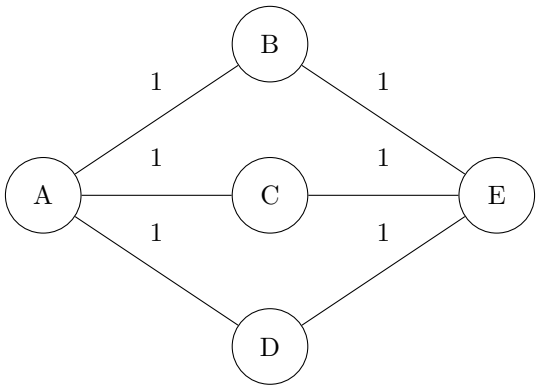


Alternatively we could use a directed graph with nodes A, B, C, D, and E and edges $A \to B$, $B \to C$, $C \to D$, and $D \to E$, which is acyclic but not a tree.



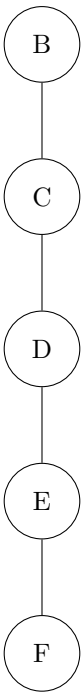
IV Weighted Graph Where Every Path from A to E Has Weight 2

Consider a graph with paths $A - B - E$, $A - D - E$, and $A - C - E$, where each edge has a weight of 1. Both paths from A to E thus have a total weight of 2, satisfying the condition.



V Rooted Tree with Minimal Number of Leafs

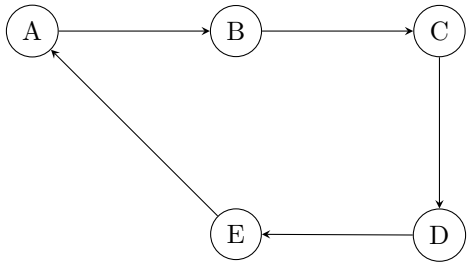
To minimize the number of leaf nodes in a rooted tree of 5 nodes, we arrange the nodes in a straight line where each node connects to exactly one other, except the root. For example:



This structure has only one leaf node F, which minimizes the number of leaves.

VI Strongly Connected Directed Graph with 5 Nodes with Minimal Edges

To form a strongly connected directed graph with 5 nodes and the minimum number of edges, we create a directed cycle among the nodes: $A \to B \to C \to D \to E \to A$. This configuration has exactly 5 edges, ensuring strong connectivity.



2 Family Tree

I Child of D

The children of D is C and E.

II Parent of I

The parent of I is H.

III Sibling of D

The sibling of D is A.

IV Ancestor of H

The ancestors of H are I, G, and F, as are on the path from H to the root node F.

V Descendent of B

The descendents of B are all nodes below it in the tree: A, D, C, and E.

VI Neighbors of D or G Not Also Neighbors of B

The neighbors of D are B, F and E, and the neighbors of G are D, I and F. Excluding neighbors of B, which are A and D, the nodes satisfying this condition are E, F and I.

3 Graph Representation

I Adjacency List Representation of $G_{undirected}$

The adjacency list of $G_{undirected}$ is:

$$
\begin{aligned}
&1 : 2, 3 \\
&2 : 1, 3 \\
&3 : 1, 2, 4 \\
&4 : 3
\end{aligned}
$$

II Adjacency Matrix Representation of $G_{undirected}$

The adjacency matrix for $G_{undirected}$ is:

$$
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}
\begin{array}{c}
\begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\
\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}
\end{array}
$$

III Adjacency List Representation of $G_{directed}$

The adjacency list for $G_{directed}$ is:

$$
\begin{aligned}
&1 : 2, 3 \\
&2 : \text{(none)} \\
&3 : 2, 4 \\
&4 : 3
\end{aligned}
$$

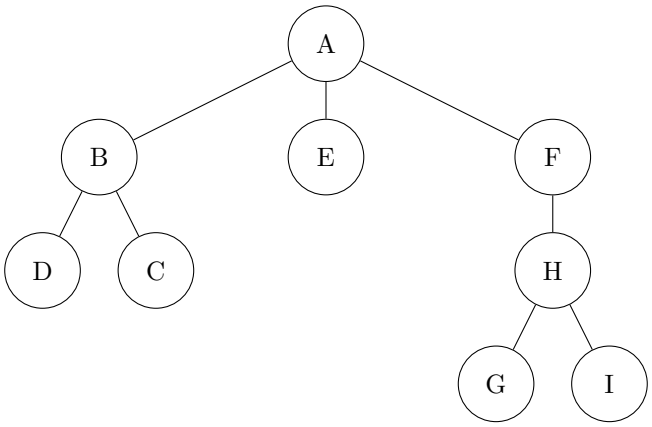
IV Adjacency Matrix Representation of $G_{directed}$

The adjacency matrix for $G_{directed}$ is:

$$
\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}
\begin{array}{c}
\begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\
\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}
\end{array}
$$

4 BFS / DFS Traversals

Redrawing the graph to better visualize the nodes and their connections:



I Breadth First Search (BFS) starting at Node A

In Breadth First Search, we explore nodes level by level from the starting node.

1. Start at node A.
2. Visit the children of A: B, E, F.
3. Visit the children of B: D, C.
4. Visit the children of F: H.
5. Visit the children of H: G, I.

Order: $A, B, E, F, D, C, H, G, I$

II Depth First Search (DFS) starting at Node A

In Depth First Search, we explore as far down each branch as possible before backtracking.

1. Start at node A.
2. Move to B.
3. Move to D.
4. Backtrack to B, then move to C.
5. Backtrack to A, then move to E.
6. Backtrack to A, then move to F.

7. Move to H.
8. Move to G.
9. Backtrack to H, then move to I.

Order: $A, B, D, C, E, F, H, G, I$

III Breadth First Search (BFS) starting at Node F

Starting from F and using BFS:

1. Start at node F.
2. Visit the children of F: H.
3. Visit the children of H: G, I.

Order: F, H, G, I

IV Depth First Search (DFS) starting at Node F

Starting from F and using DFS:

1. Start at node F.
2. Move to H.
3. Move to G.
4. Backtrack to H, then move to I.

Order: F, H, G, I

5 Shortest Path (Dijkstra's Algorithm)

Using Dijkstra's algorithm, we compute the shortest path from A to G. Below is the table showing the path weights and predecessors for each node.

iteration	node visited	A	B	C	D	E	F	G
0	A	start:0	A:5	A:5	none	none	none	none
1	B	start:0	A:5	A:5	B:14	B:7	none	none
2	C	start:0	A:5	A:5	C:6	B:7	C:14	none
3	D	start:0	A:5	A:5	C:6	B:7	D:9	none
4	E	start:0	A:5	A:5	C:6	B:7	D:9	E:15
5	F	start:0	A:5	A:5	C:6	B:7	D:9	F:13

Now by backtracking from G to A.

$$G \to F \to D \to C \to A$$

Thus, shortest path from A to G is:

$$A \to C \to D \to F \to G, \text{ with a total weight of } 13.$$

6 Extra Credit: Airline Connectivity

Given a graph representing flights among cities, if every pair of cities is connected by exactly one airline, there exists at least one airline that can connect any two cities via a sequence of flights, thus forming a connected graph for that airline.