Problem Solving: Find a valid, finite path in the problem space leading from the initial state to a goal state.

Example problem space
Let our states be A, B, C, D, E, F, R, and S. The initial state is A, and for now we will not specify a goal state. The operators are as follows:

\[
\begin{align*}
\alpha_1 &: A \rightarrow B \\
\gamma_1 &: C \rightarrow A \\
\alpha_2 &: A \rightarrow C \\
\gamma_2 &: C \rightarrow F \\
\beta_1 &: B \rightarrow D \\
\delta_1 &: D \rightarrow E \\
\beta_2 &: B \rightarrow C \\
\rho_1 &: R \rightarrow S
\end{align*}
\]

This problem specification induces the following path tree (with goal state and operator labels omitted):
“Systematic” Search Algorithms

Depth-first Search (DFS):
1. Mark node 0 visited.
2. Choose the deepest visited node \( n \).
   (a) If \( n \) corresponds to a problem-space goal state, declare success and stop;
   (b) otherwise, if \( n \) corresponds to a repeated problem-space state or is childless, remove it and all its descendants;
   (c) otherwise, mark \( n \)'s least-Gorn-numbered unvisited child as visited.
3. If the tree still has nodes, repeat step 2.
4. If the entire tree has been removed, declare failure.

Breadth-first Search (BFS):
1. Mark node 0 touched.
2. Choose the highest touched node \( n \) with untouched children.
   (a) If \( n \) corresponds to a problem-space goal state, declare success and stop;
   (b) otherwise, if \( n \) corresponds to a repeated problem-space state or is childless, delete it and all its descendants;
   (c) otherwise, mark \( n \)'s least-Gorn-numbered untouched child as touched.
3. If the tree still has nodes, repeat step 2.
4. If the entire tree has been deleted, declare failure.

Note: we’re using “visited” and “removed” for DFS and “touched” and “deleted” for BFS to facilitate lecture notation.

Both DFS and BFS will visit all of the nodes in a (finite) tree, just in different orders.