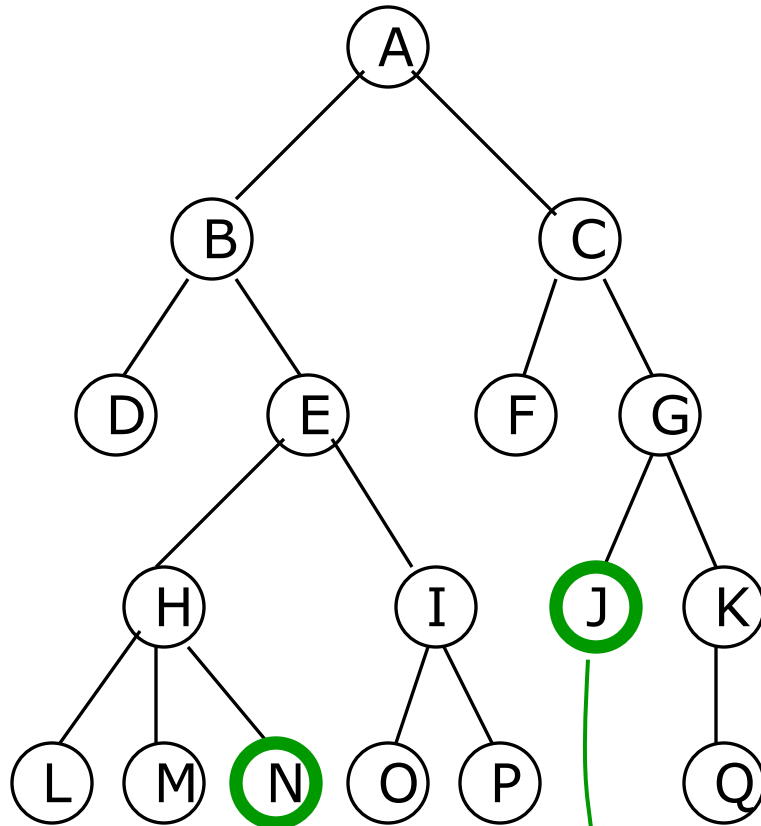


Tree Searching



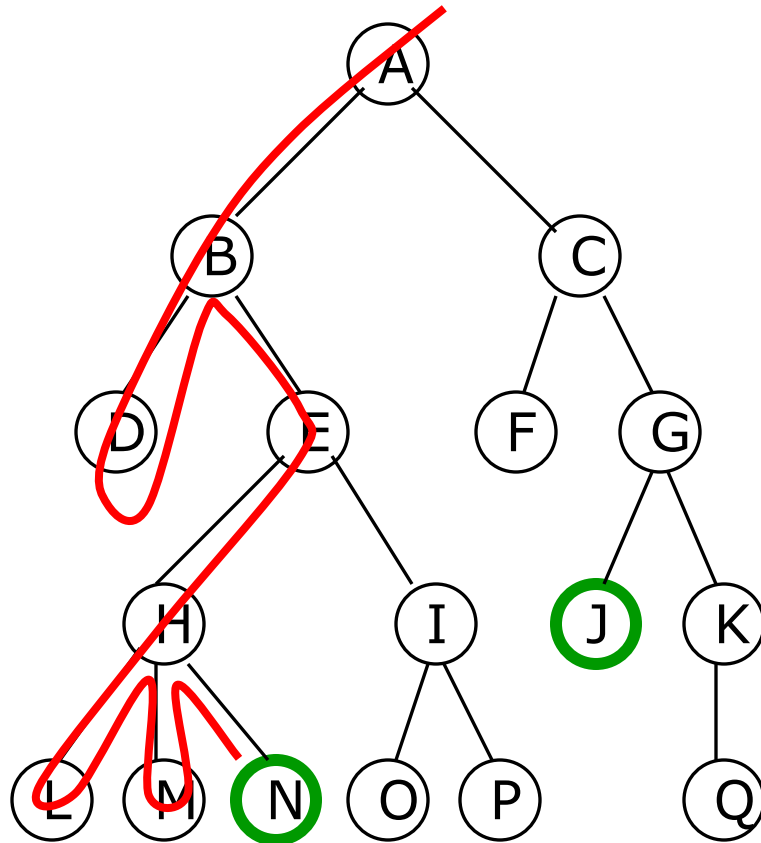
Tree searches



- A **tree search** starts at the root and explores nodes from there, looking for a **goal node** (a node that satisfies certain conditions, depending on the problem)
- For some problems, any goal node is acceptable (**N** or **J**); for other problems, you want a minimum-depth goal node, that is, a goal node nearest the root (only **J**)

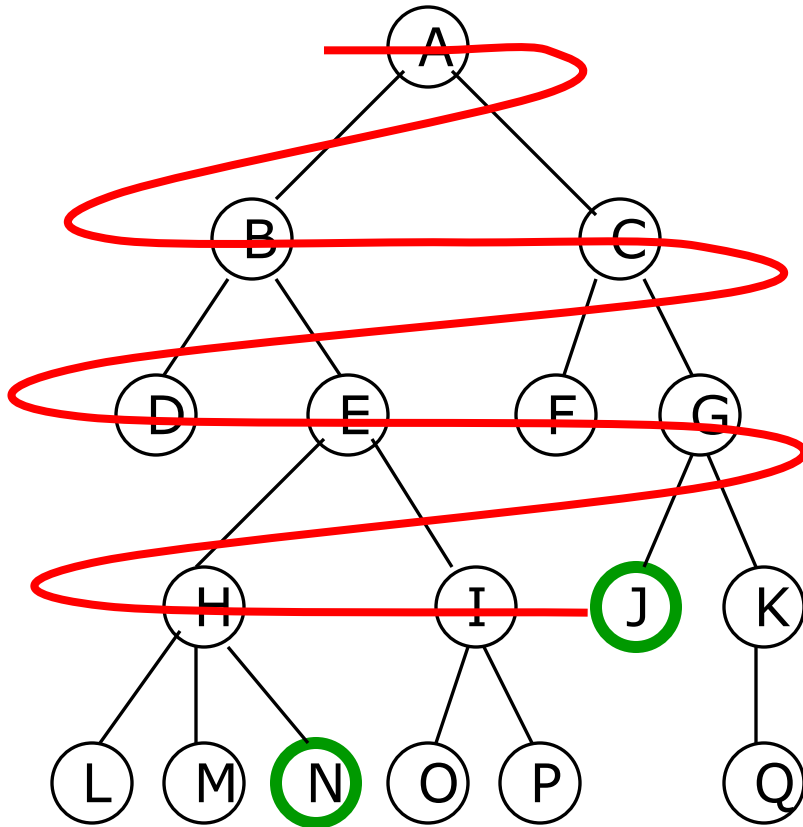
Goal nodes

Depth-first searching



- A **depth-first** search (**DFS**) explores a path all the way to a leaf before **backtracking** and exploring another path
- For example, after searching **A**, then **B**, then **D**, the search backtracks and tries another path from **B**
- Node are explored in the order **A B D E H L M N I O P C F G J K Q**
- **N** will be found before **J**

Breadth-first searching



- A **breadth-first** search (**BFS**) explores nodes nearest the root before exploring nodes further away
- For example, after searching **A**, then **B**, then **C**, the search proceeds with **D**, **E**, **F**, **G**
- Node are explored in the order **A B C D E F G H I J K L M N O P Q**
- **J** will be found before **N**

Comparison of algorithms

- Depth-first searching:

- Put the root node on a **stack**;
while (**stack** is not empty) {
 remove a node from the **stack**;
 if (node is a goal node) return success;
 put all children of node onto the **stack**;
}
return failure;

- Breadth-first searching:

- Put the root node on a **queue**;
while (**queue** is not empty) {
 remove a node from the **queue**;
 if (node is a goal node) return success;
 put all children of node onto the **queue**;
}
return failure;