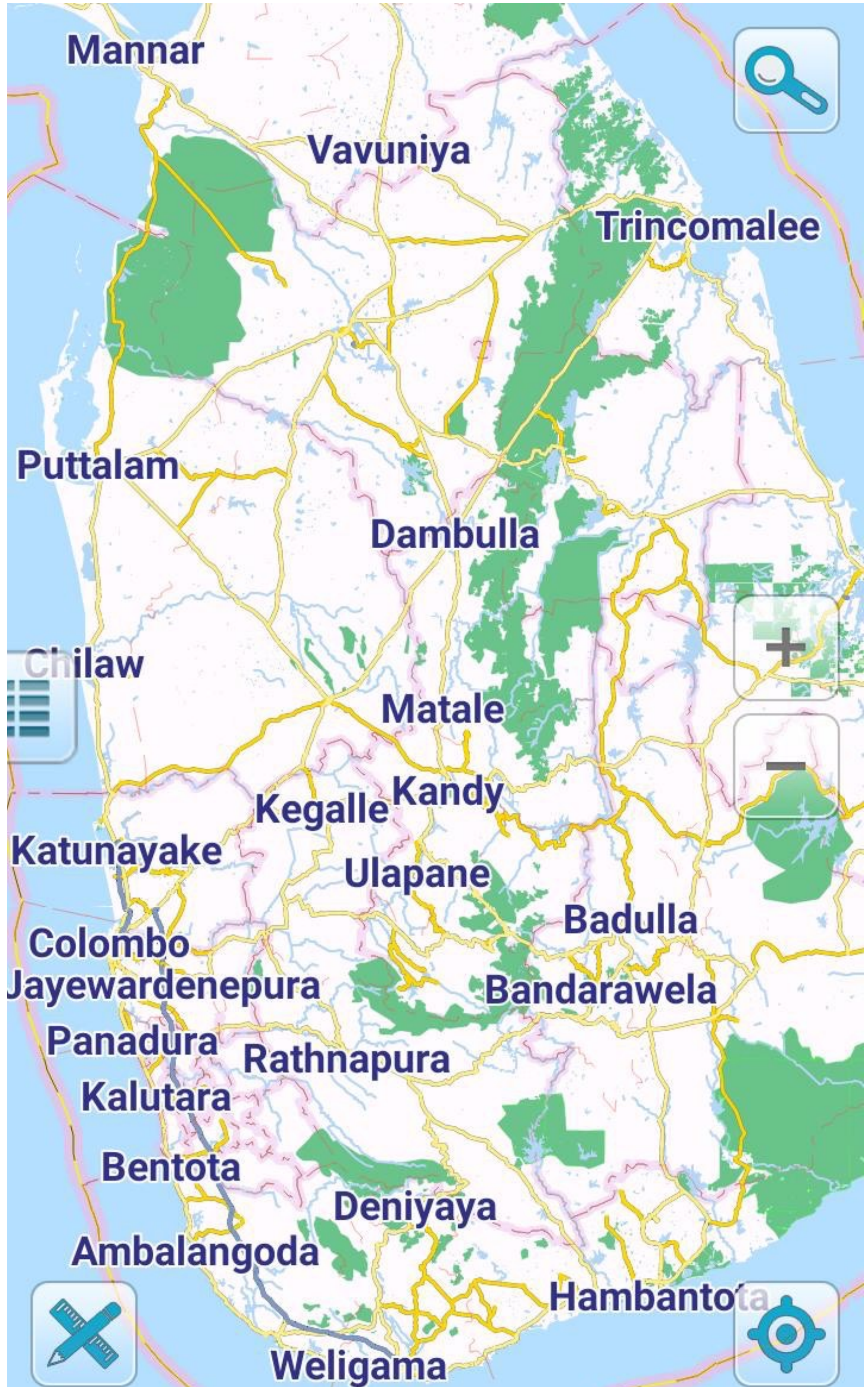
INFORMATICS INSTITUTE OF TECHNOLOGY

CM 2602 - Artificial Intelligence

SOLVING PROBLEMS BY SEARCHING

Introduction

- In this topic, we see how an agent can find a sequence of actions that achieves its goals, when no single action will do.


Mannar
Vavuniya
Trincomalee
Puttalam
Dambulla
Chilaw
Matale
Kandy
Kegalle
Katunayake
Ulapane
Badulla
Colombo
Jayewardenepura
Bandarawela
Panadura
Rathnapura
Kalutara
Bentota
Deniyaya
Ambalangoda
Hambantota
Weligama

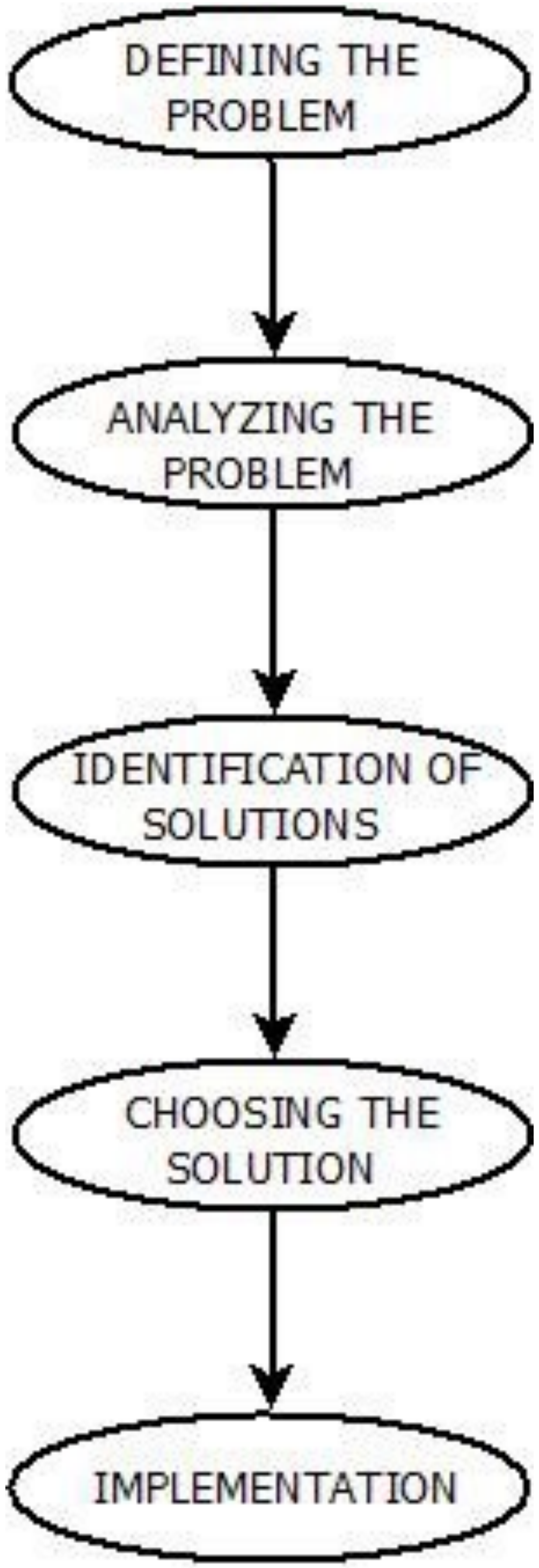
Problem Solving:

- Four general steps in problem solving:
  - Goal Formulation:
    - What are the successful world states
  - Problem Formulation:
    - What actions and states to consider given the goal
  - Search:
    - Examine different possible sequences of actions that lead to states of known value and then choose the best sequence
  - Execute:
    - Perform the actions on the basis of the solution

Problem Solving Agents

- Problem-solving agent: a type of goal-based agent
  - Decide what to do by finding sequences of actions that lead to desirable states

Problem Solving Process



Problem Solving

- First we need define the problem.

- In AI, we defines a problem using five components.
    - Initial State
    - Available Actions (Operators to change the state)
    - Transition Model
    - Goal Test
    - Path Cost

State

- A **state** is representation of elements in a given moment.

State Space

- The **state space** is the set of all states reachable from the initial state.
- It forms a graph (or map) in which the nodes are states and the arcs between nodes are actions.
- A **path** in the state space is a sequence of states connected by a sequence of actions.
- The solution of the problem is part of the map formed by the state space.

Initial State

- Starting state of the problem.

Goal State

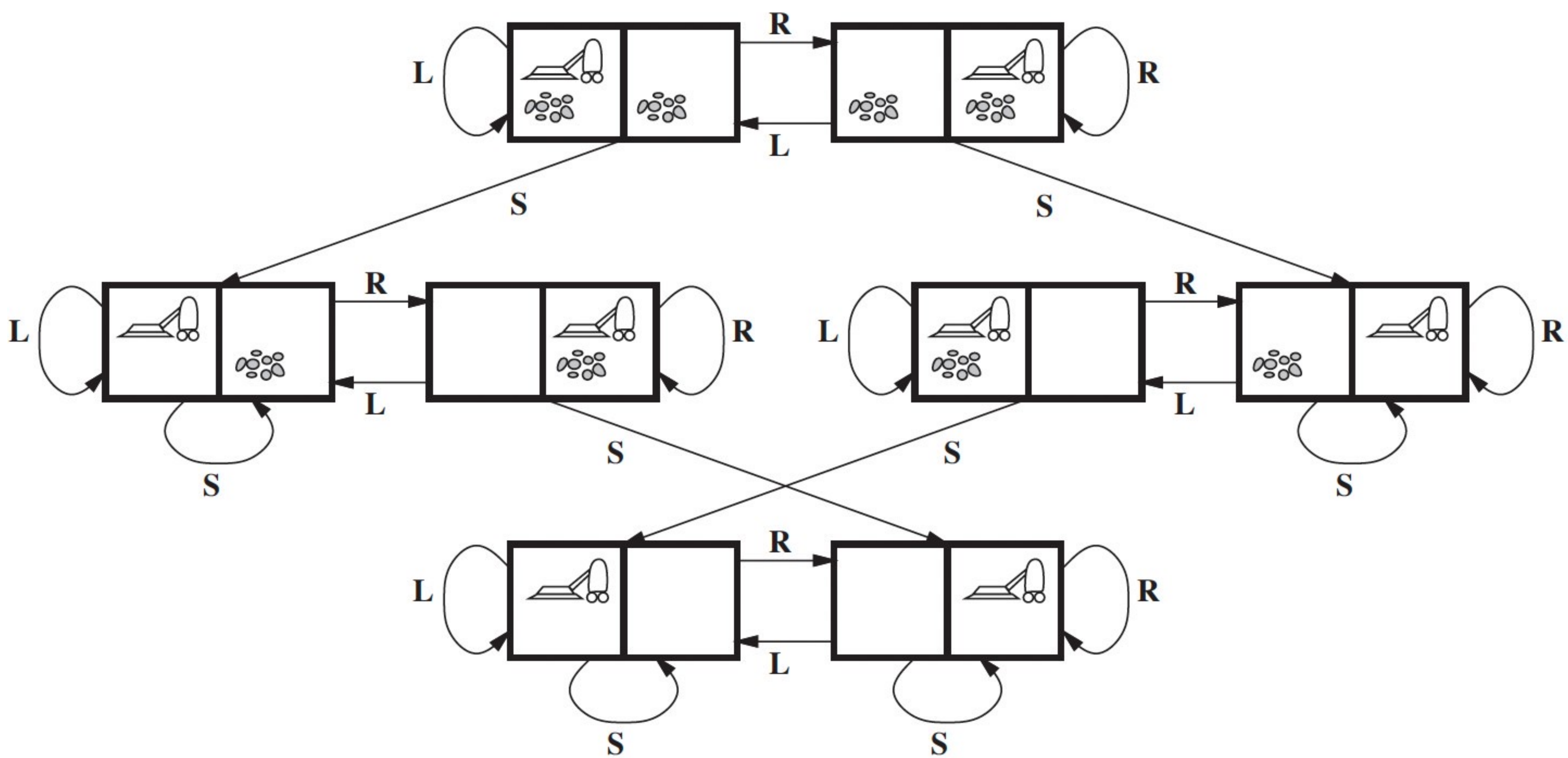
- Final state of the problem.

Problem Solution

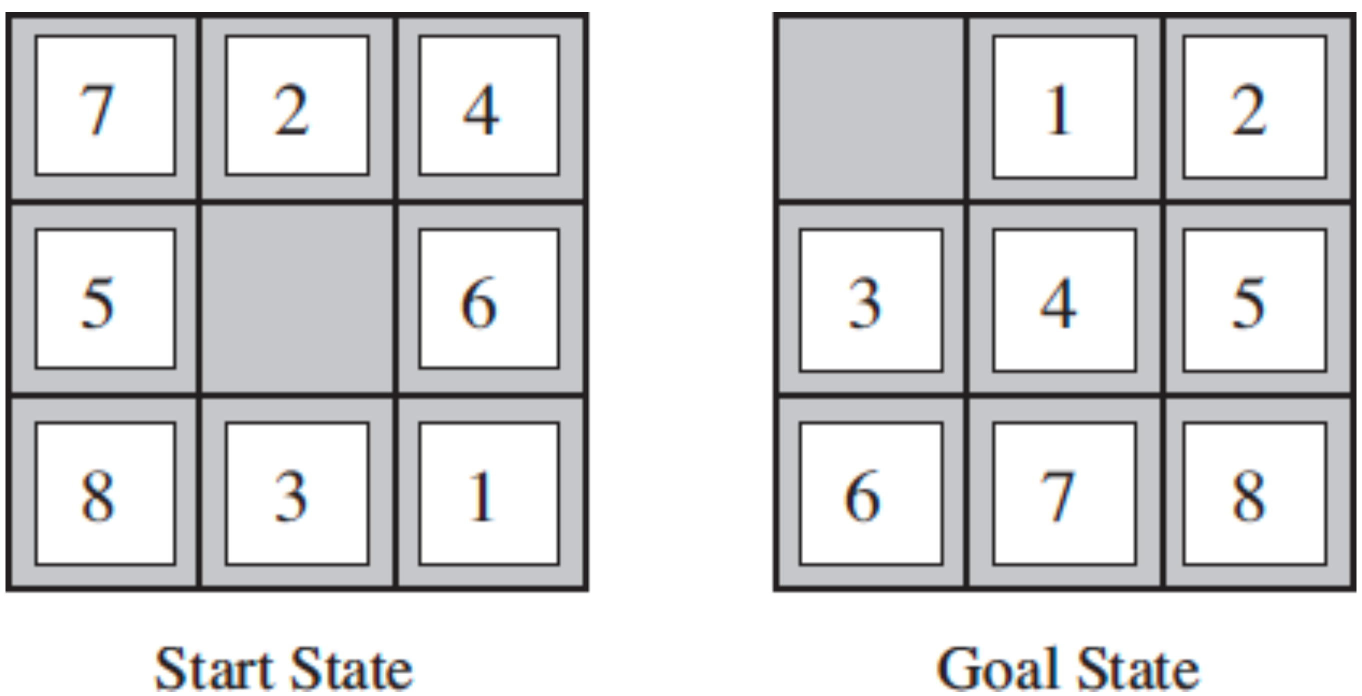
- A **solution** in the state space is a path from the initial state to a goal state.

- **Path/solution cost**: function that assigns a numeric cost to each path, the cost of applying the operators to the states
- Solution quality is measured by the path cost function, and an **optimal solution** has the lowest path cost among all solutions.

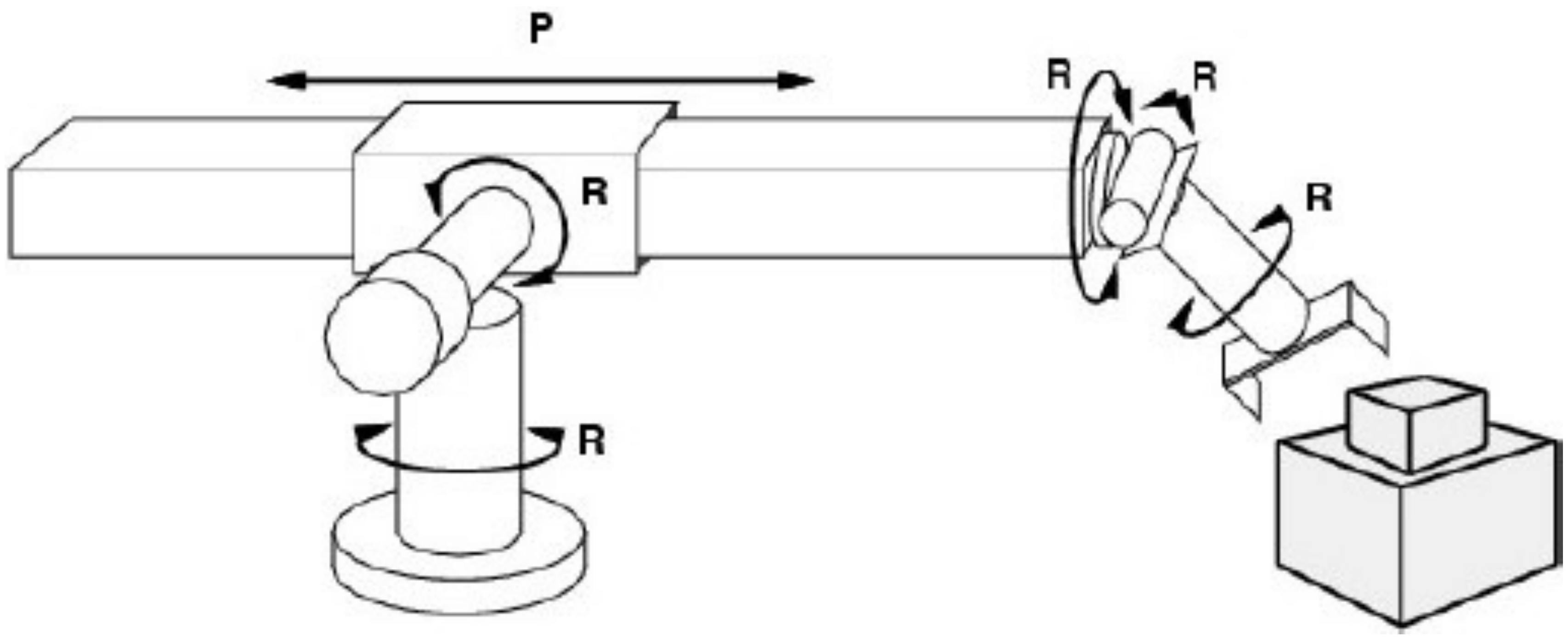
Example (Vacuum Cleaner)



Example (8-puzzle)



Example (Robotic Assembly)



Problem Solving by Searching

- In general, searching refers to as finding information one needs.
- Searching is the most commonly used technique of problem solving in artificial intelligence.
- The searching algorithm helps us to search for solution of particular problem.

Properties of a Search Algorithm

- **Completeness**: Is the algorithm guaranteed to find a solution when there is one?
- **Optimality**: Does the strategy find the optimal solution?
- **Time complexity**: How long does it take to find a solution? (with respect to number of inputs)
- **Space complexity**: How much memory is needed to perform the search? (with respect to number of inputs)

Types of Search

- **Uninformed Search (Blind Search)**
- **Informed Search (Heuristic Search)**

Types of Uninformed Search

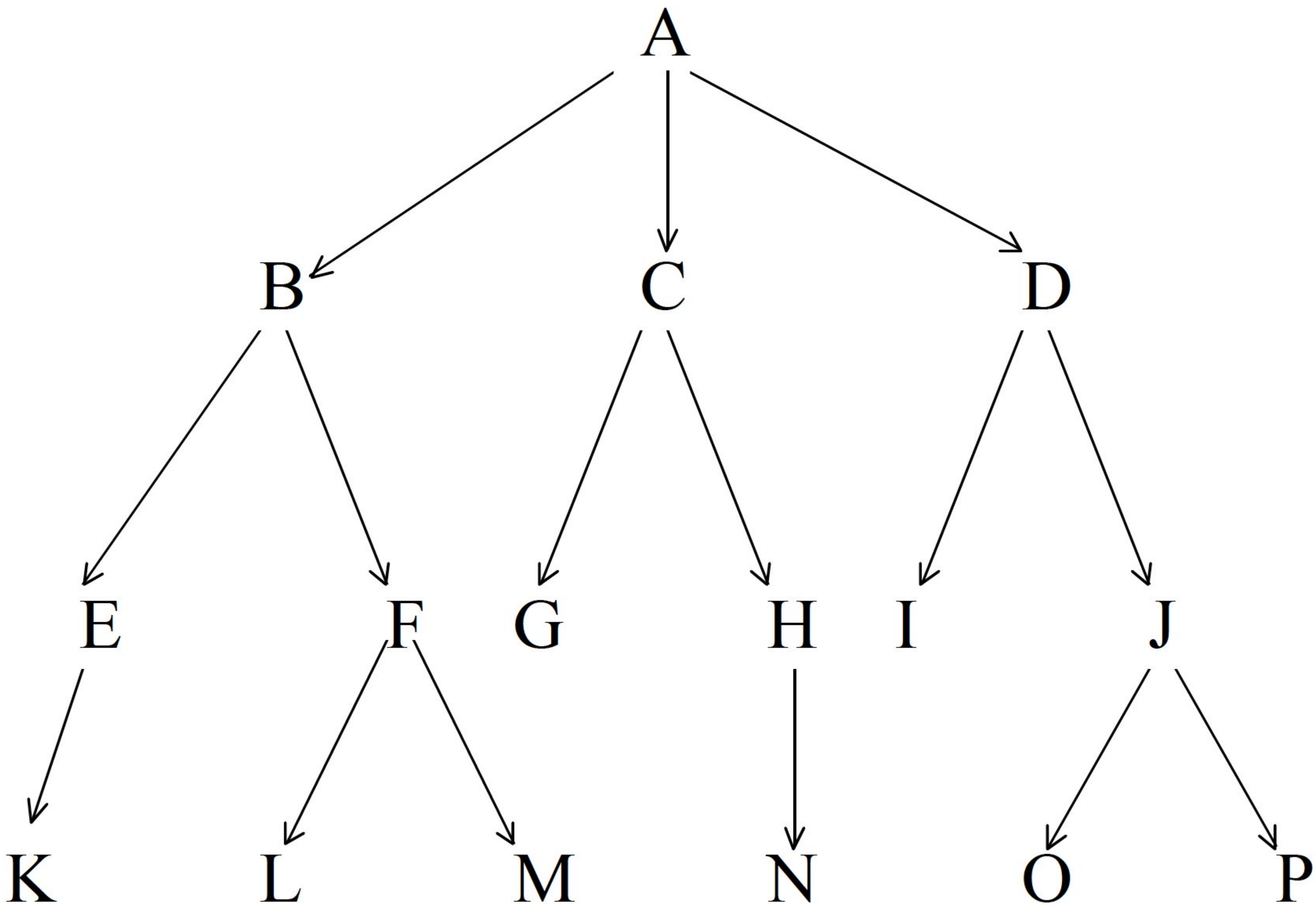
- Breadth-first Search
- Depth-first Search
- Uniform Cost Search
- Depth-limited search
- Iterative deepening depth-first search

Breadth-firstSearch

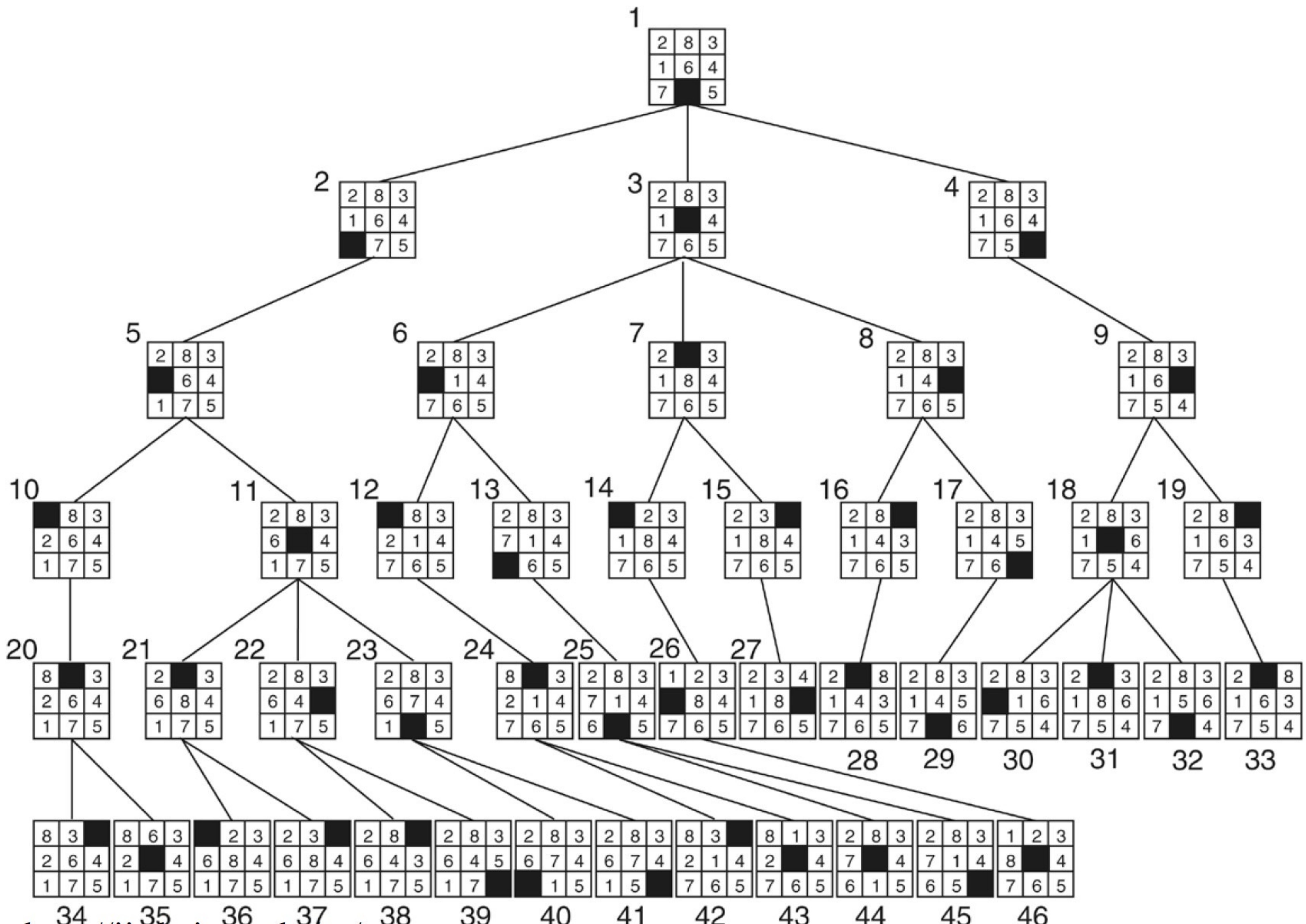
- Breadth-first search is a simple strategy in which the root node is expanded first, then all the successors of the root node are expanded next, then their successors, and so on.

- In general, all the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded.

Breadth-first Search - Example

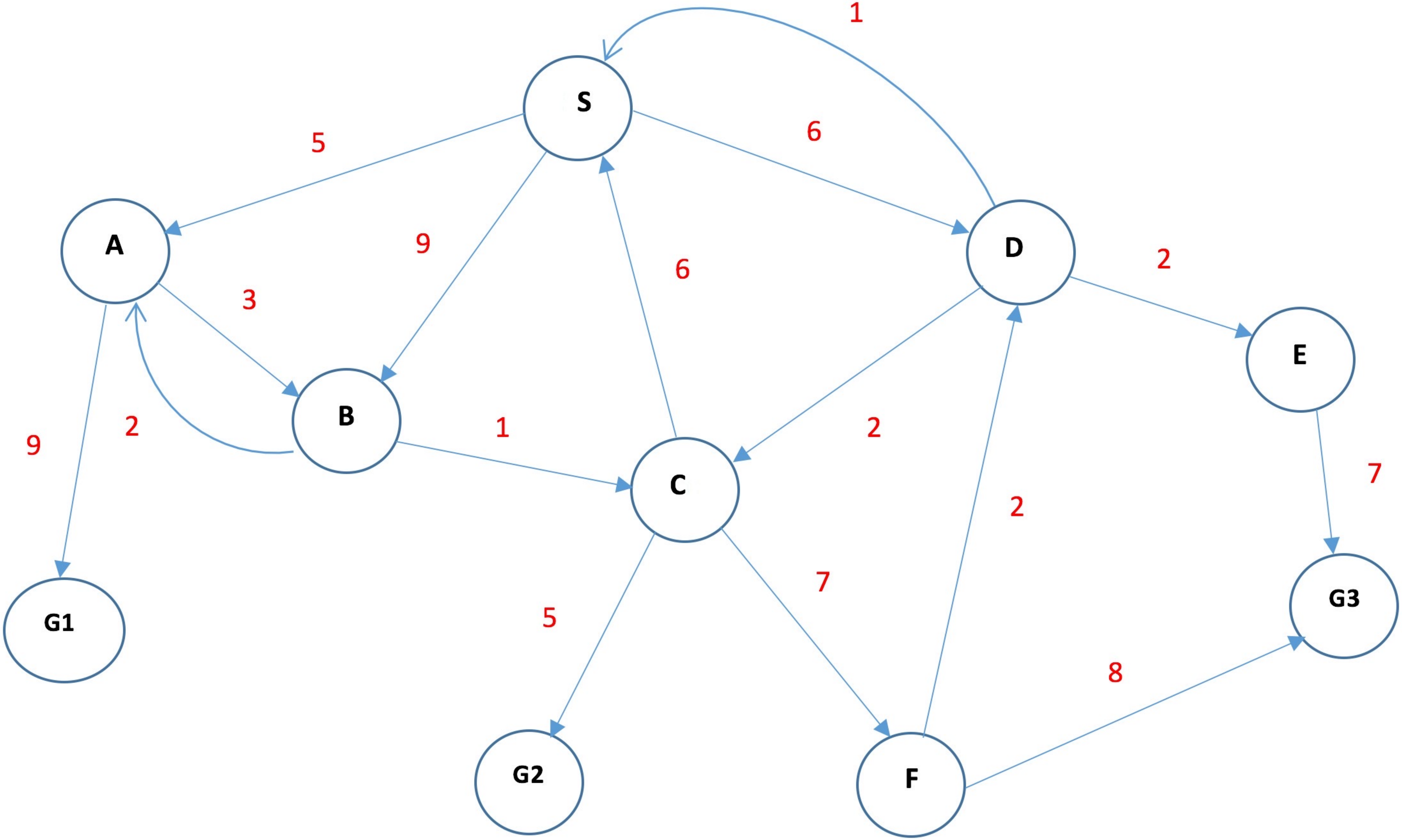


Breadth-first Search - Example



n http://iis.kaist.ac.kr/es/

Breadth-first Search - Example



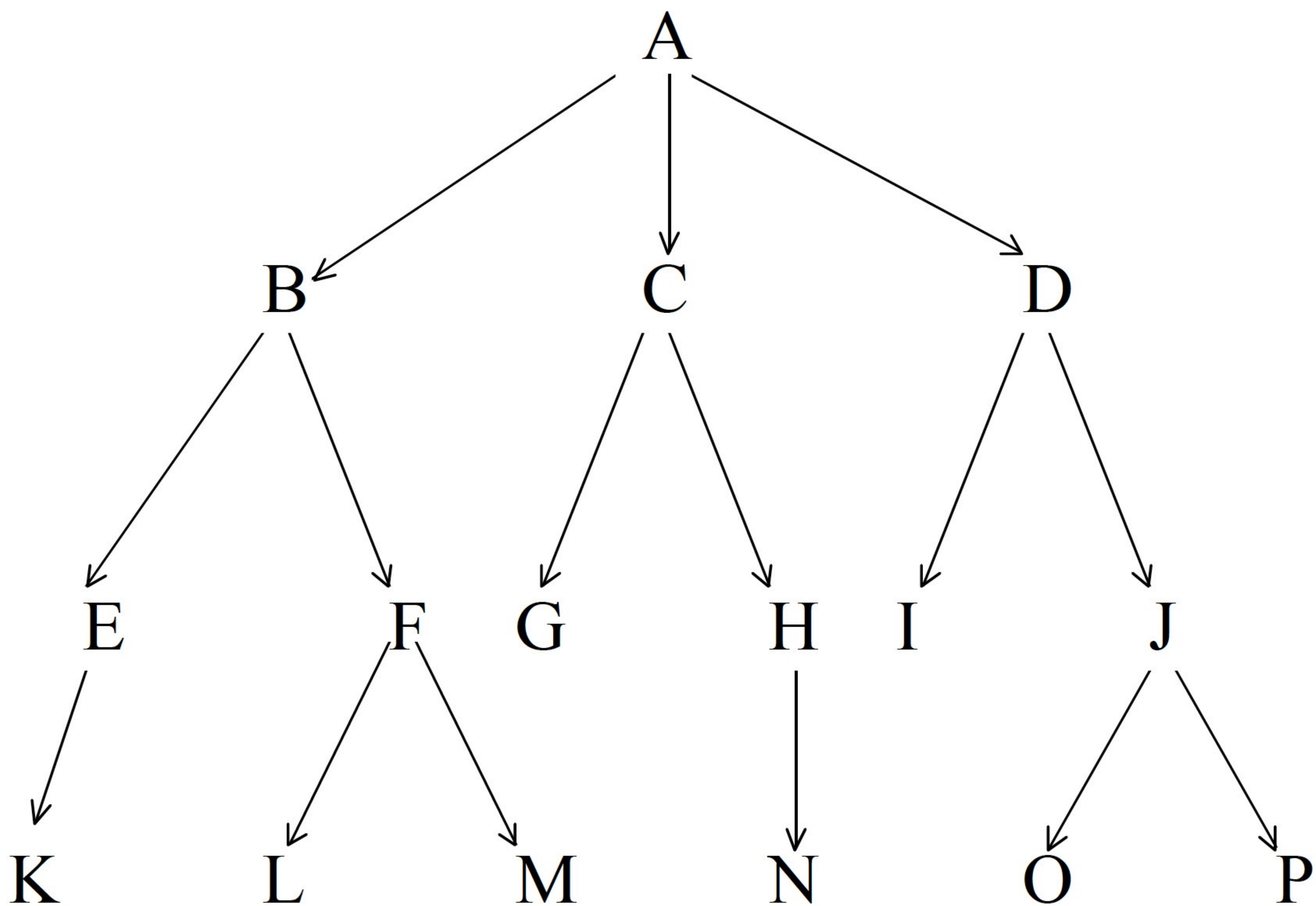
Breadth-first Search

- Pros:
  - Guarantee to find a solution
  - Find a solution with minimal steps
- Cons:
  - Need lot of memory
  - Time Consuming

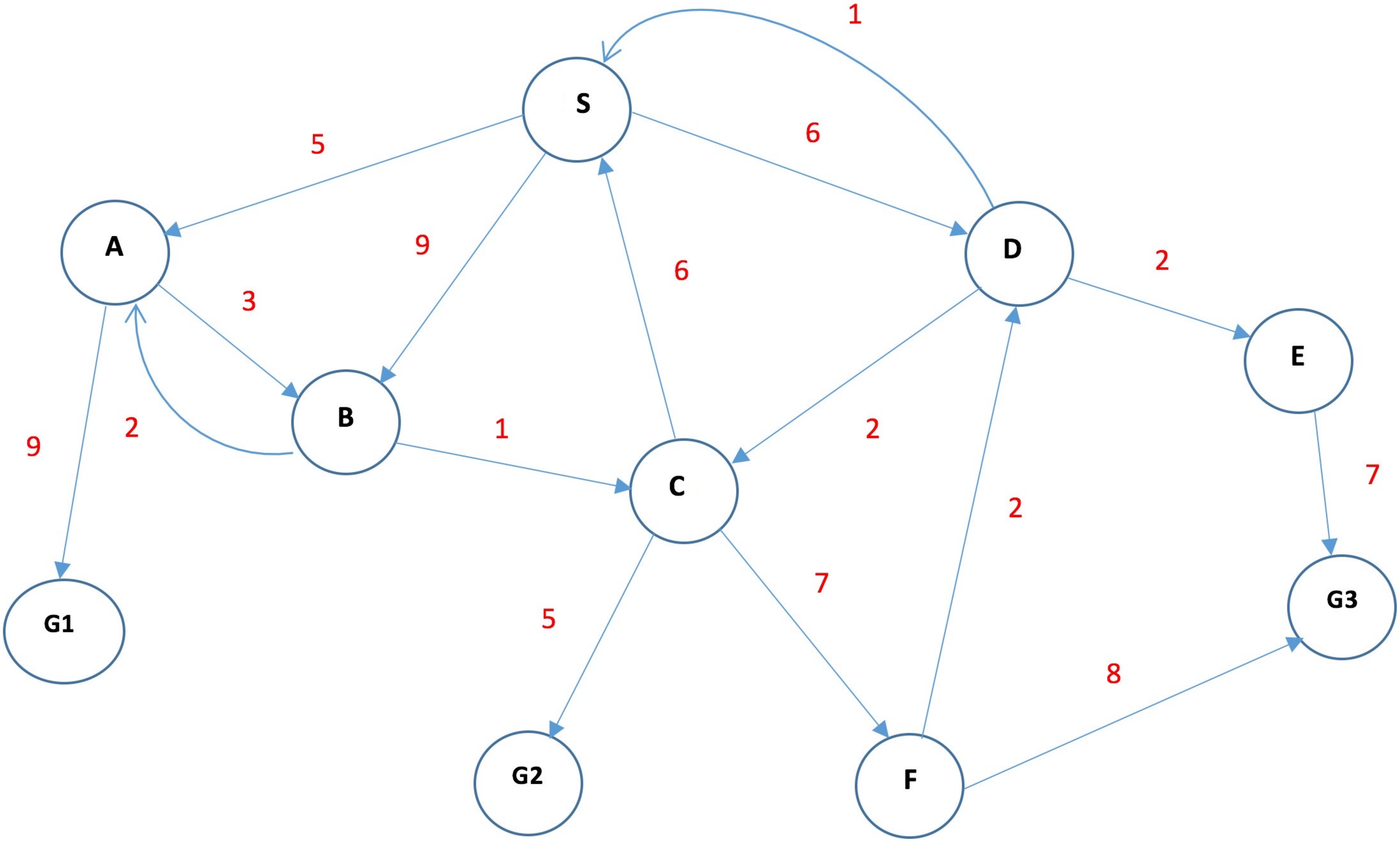
Depth-first Search

- Depth-first search always expands the deepest node in the current frontier of the search tree.
- That is, when a state is examined, all of its children and descendants are examined before any of its siblings.

Depth-first Search - Example



Depth-first Search - Example



Depth-first Search

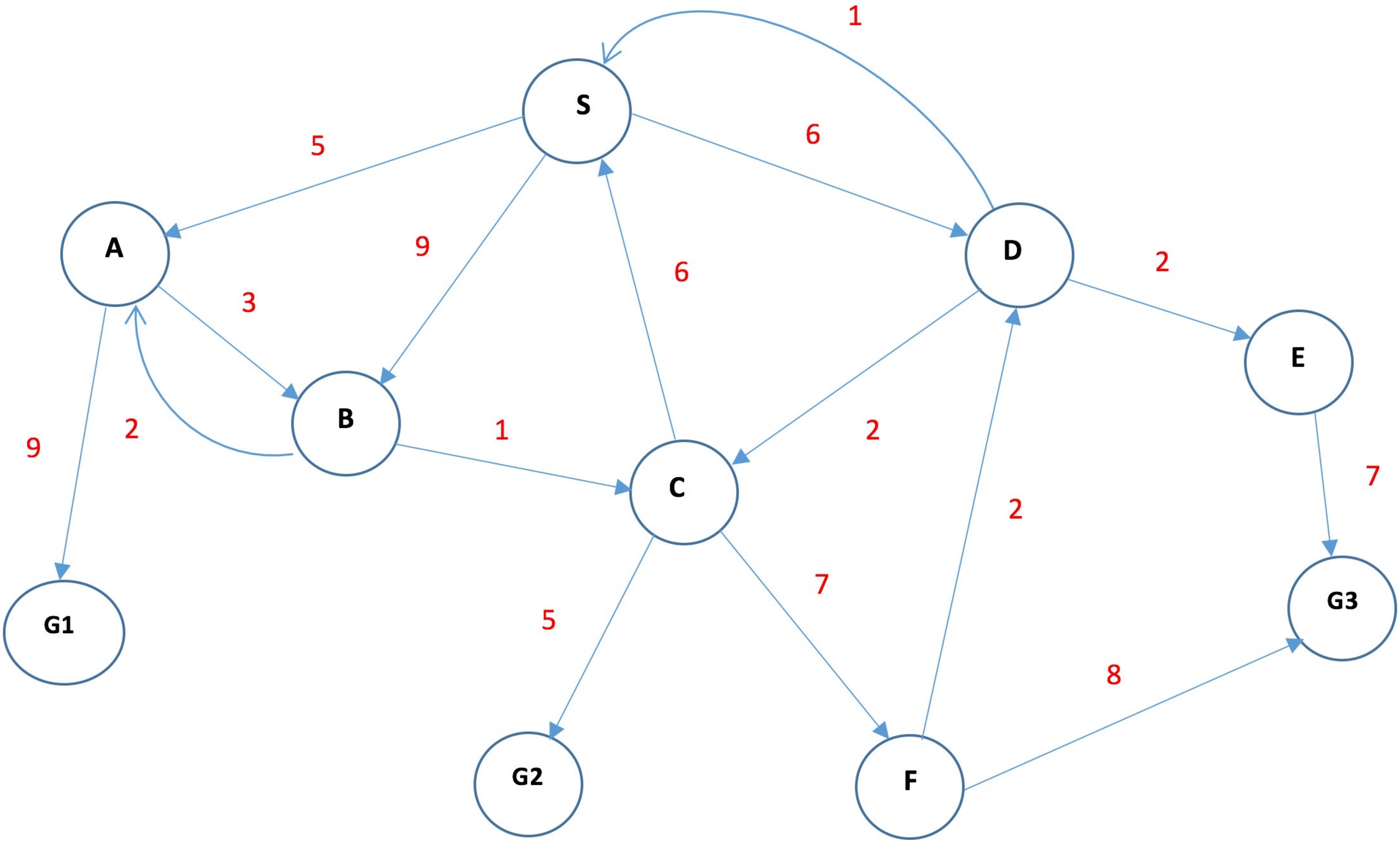
- Pros:
  - Memory efficient
  - Less time

- Cons:
  - Not guaranteed to find a solution

Uniform cost Search

- Expands the node with the lowest path cost

Uniform Cost Search - Example



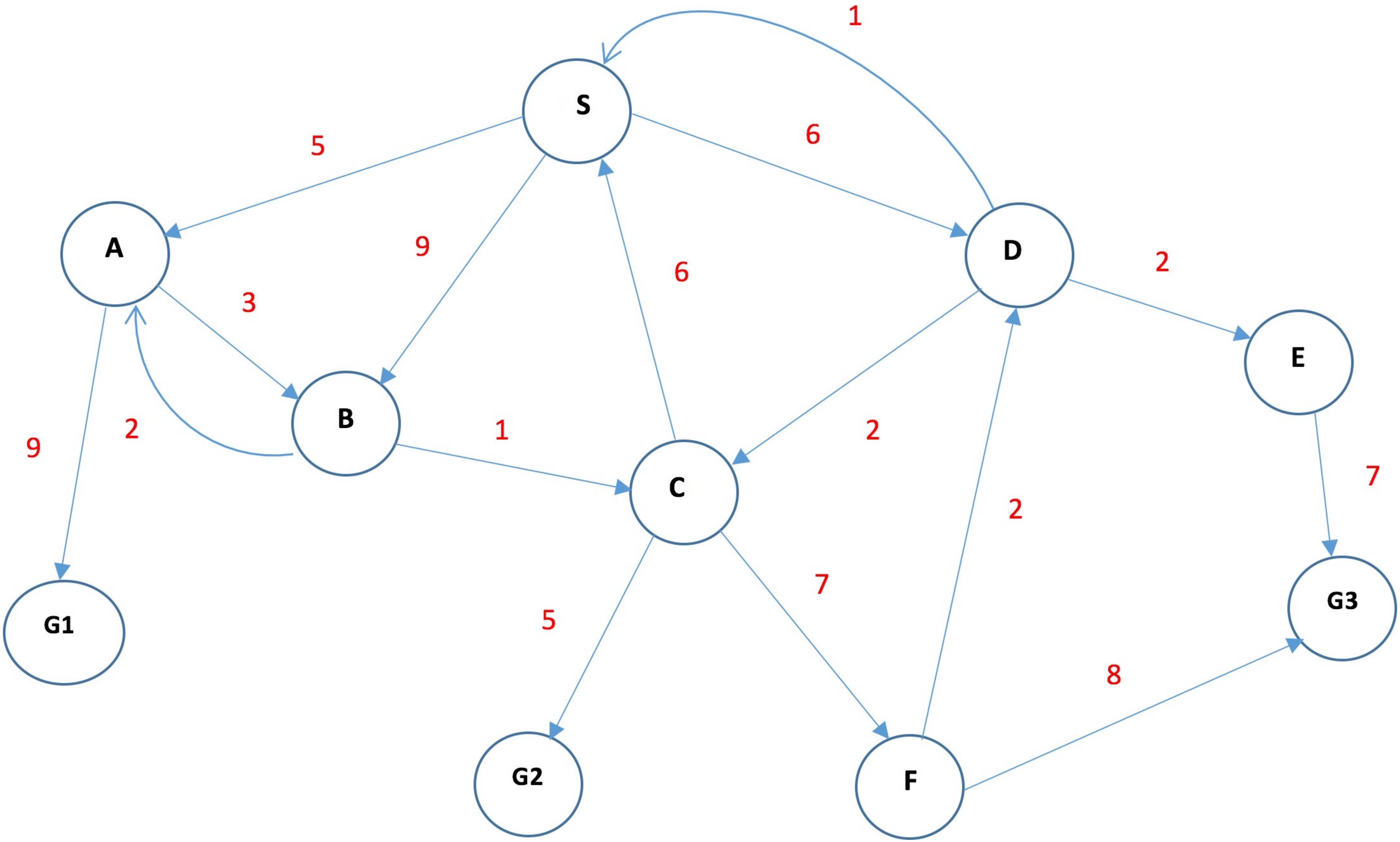
Uniform Cost Search

- Pros:
  - Complete
  - Optimal
- Cons:
  - Explore options in every direction

Depth Limited Search

- It is similar to DFS with a predetermined limit.
- Node at the depth limit will treat as it has no successor nodes further.

Depth Limited Search - Example



Depth Limited Search

- Pros:
  - Memory Efficient
  - Does not run in to infinite loops
- Cons:
  - Incompleteness
  - Not Optimal

Iterative Deepening depth-first Search

- It is a combination of Depth First Search (DFS) and Breadth First Search (BFS).

Iterative Deepening depth-first Search

- Pros:
  - Faster than BFS
  - Not going to infinite loops
- Cons:
  - Repeat all the work in the previous phase

Comparing uninformed search strategies

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes^{a}	Yes^{a,b}	No	No	Yes^{a}
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$
Optimal?	Yes^{c}	Yes	No	No	Yes^{c}

Evaluation of tree-search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: ^{a} complete if b is finite; ^{b} complete if step costs $\geq \epsilon$ for positive ϵ; ^{c} optimal if step costs are all identical; ^{d} if both directions use breadth-first search.