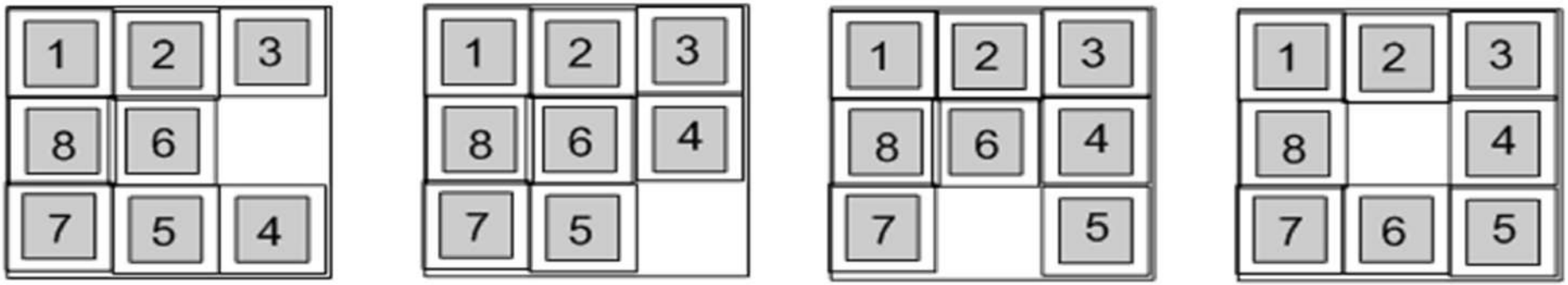
# Artificial Intelligence

## Lecture 26

# Problem Solving by Search

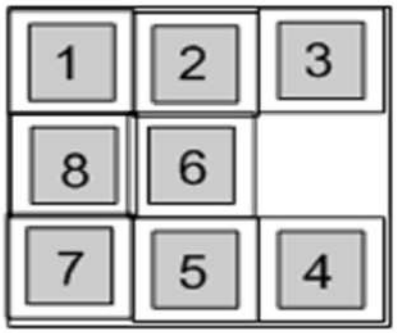
*Prepared by:*

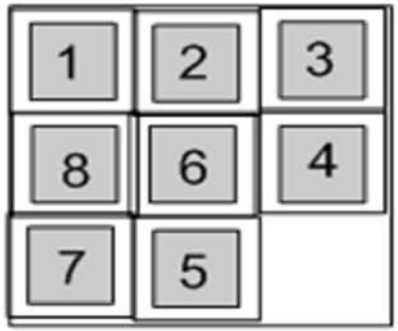
**Md. Mijanur Rahman, Prof. Dr.**
Dept. of CSE, JKKNIU
Email: mijanjkkniu@gmail.com

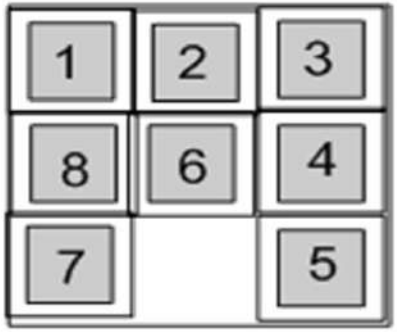


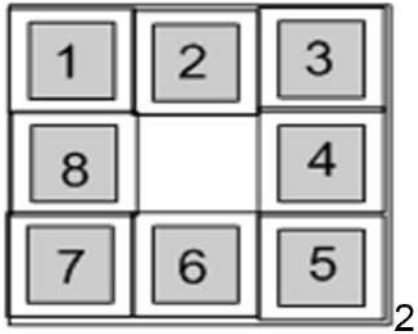
# Lecture Outlines

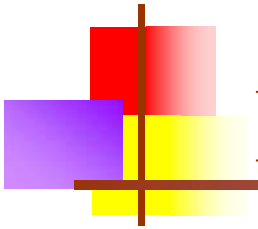
- Problem solving
- Search and examples of search problems
- Searching for solutions
- Tree searches and search strategies
- Search algorithms
- …



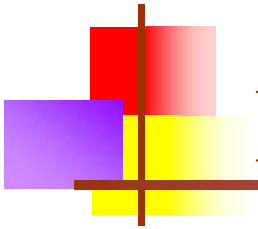





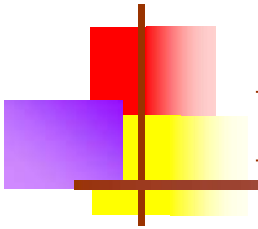
# Problem Solving

- An important aspect of intelligence is *goal-based* problem solving.
- The solution of many problems (e.g. noughts and crosses, timetabling, chess) can be described by finding a sequence of actions that lead to a desirable goal.
- Each action changes the *state* and the aim is to find the sequence of actions and states that lead from the initial (start) state to a final (goal) state.



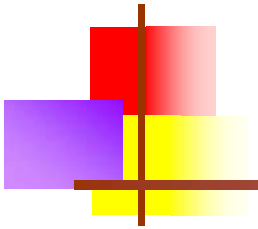
# Problem Solving

- Problem solving involves:
  - problem definition - detailed specifications of inputs and what constitutes an acceptable solution;
  - problem analysis;
  - knowledge representation;
  - problem solving - selection of best techniques.



# Problem Solving

- A well-defined problem can be described by:
  - **Initial state**
  - **Operator or successor function** - for any state x returns s(x), the set of states reachable from x with one action
  - **State space** - all states reachable from initial by any sequence of actions
  - **Path** - sequence through state space
  - **Path cost** - function that assigns a cost to a path. Cost of a path is the sum of costs of individual actions along the path
  - **Goal test** - test to determine if at goal state

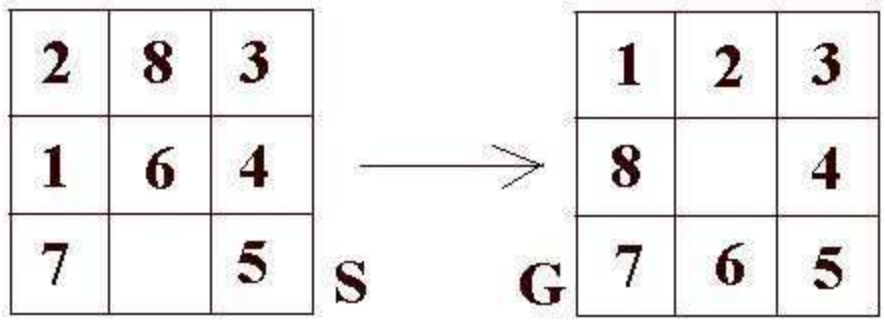


# Search

- Search is the systematic examination of states to find path from the start/root state to the goal state.
- The set of possible states, together with *operators* defining their connectivity constitute the *search space*.
- The output of a search algorithm is a solution, that is, a path from the initial state to a state that satisfies the goal test.
- Search techniques fall into three groups:
  - Methods which find *any* start - goal path,
  - Methods which find the *best* path, and finally
  - Search methods in the face of adversaries

# Example Problems

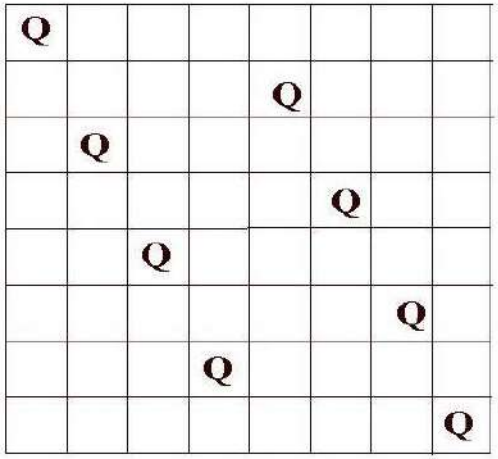
- The real art of problem solving is in deciding the description of the states and the operators.
- The 8-puzzle:



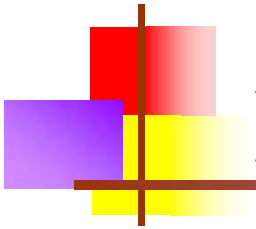

  - State: location of blank
  - Operator: blank moves left, right, up and down
  - Goal Test: match G
  - Path Cost: each step costs 1 so cost is length of path

# Example Problems

- The 8-queens problem:

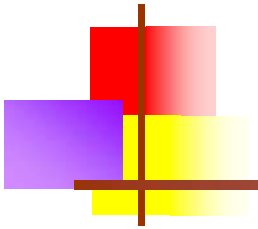



  - Aim of problem is to place 8 queens on chess board so none attacks another, diagram shows a solution.



# Example Problems

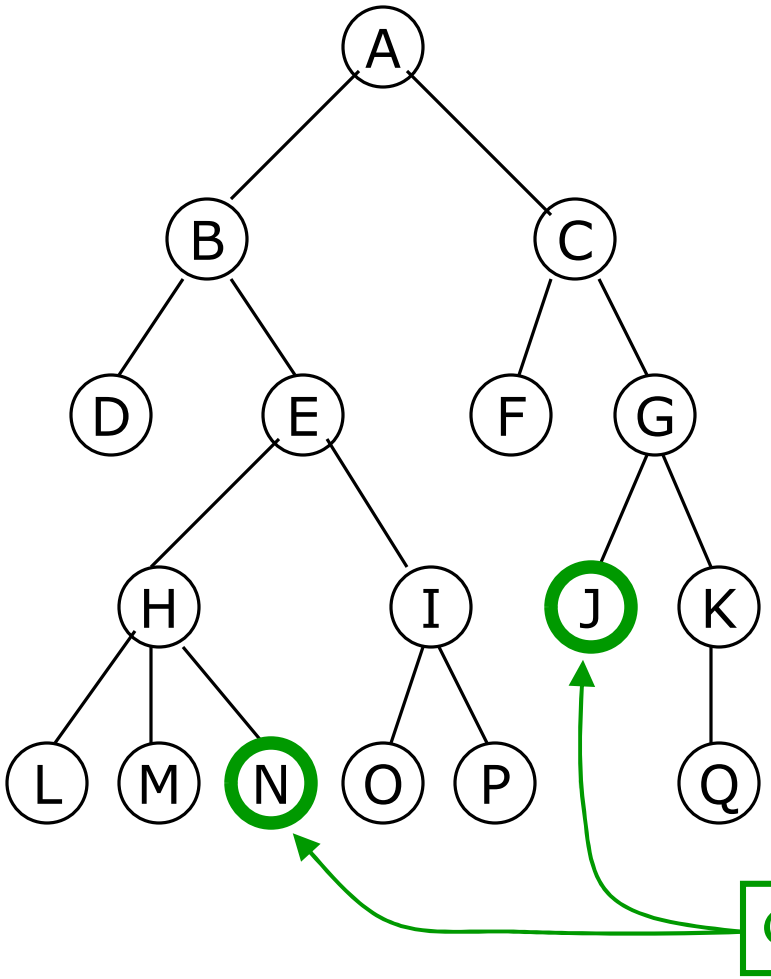
- **Real Problems**
  - Route finding
  - Travelling salesman
  - Scheduling and planning
  - Theorem proving
  - VLSI layout
  - Speech recognition
  - Model based vision
  - Robot navigation



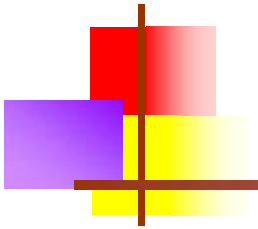
# Searching for solutions

- **Searching for solutions: Graphs or nets versus trees**
- The map of all paths within a state-space is a *graph* of *nodes* which are connected by *links*.
- Like graphs, trees have nodes, but they are linked by *branches*. The start node is called the *root* and nodes at the other ends are *leaves*.
- The aim of search is *not* to produce complete physical trees in memory, but rather explore as little of the virtual tree looking for root-goal paths.

# Tree searches

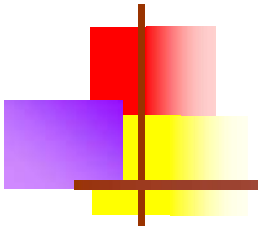



- A tree search starts at the root and explores nodes from there, looking for a goal node (a node that satisfies certain conditions, depending on the problem)
- For some problems, any goal node is acceptable (N or J); for other problems, you want a minimum-depth goal node, that is, a goal node nearest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



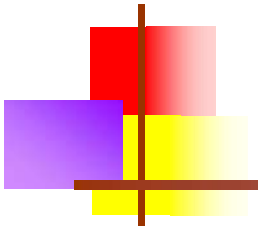
# Search Strategies…

- A search strategy simply determines which node in the fringe to expand next.
- **Measuring performance**
  - What are the ways we measure the performance of a search strategy or algorithm?
  - **Completeness:** will the algorithm find a solution if one exists?
  - **Optimality:** will the algorithm find the optimal solution? (lowest path cost among all solutions)
  - **Time complexity:** how long does it take to find a solution?
  - **Space complexity:** how much memory is needed to perform the search?



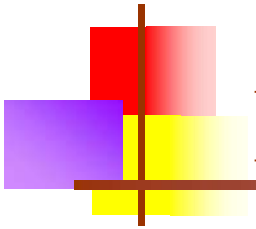
# Search Strategies…

- **Measuring performance**
- Some properties of the search space we will use to answer these questions are as follows:
- **Branching factor:** *b,* maximum number of successors of any node
- **Depth of solution:** *d,* is the minimum number of steps to get to the goal
- **Maximal path length in the space:** what is the longest path we could follow?



# Search Strategies

- **Types:**
- **Uninformed or Blind search** - can only move according to position in search.
- Uninformed search strategies use only the information available in the problem definition.
  - Depth-first search
  - Breadth-first
- **Informed or Heuristic search** - use *domain-specific* information to decide where to search next.
  - Hill climbing
  - Beam search
  - Best-first search



# Direction of search

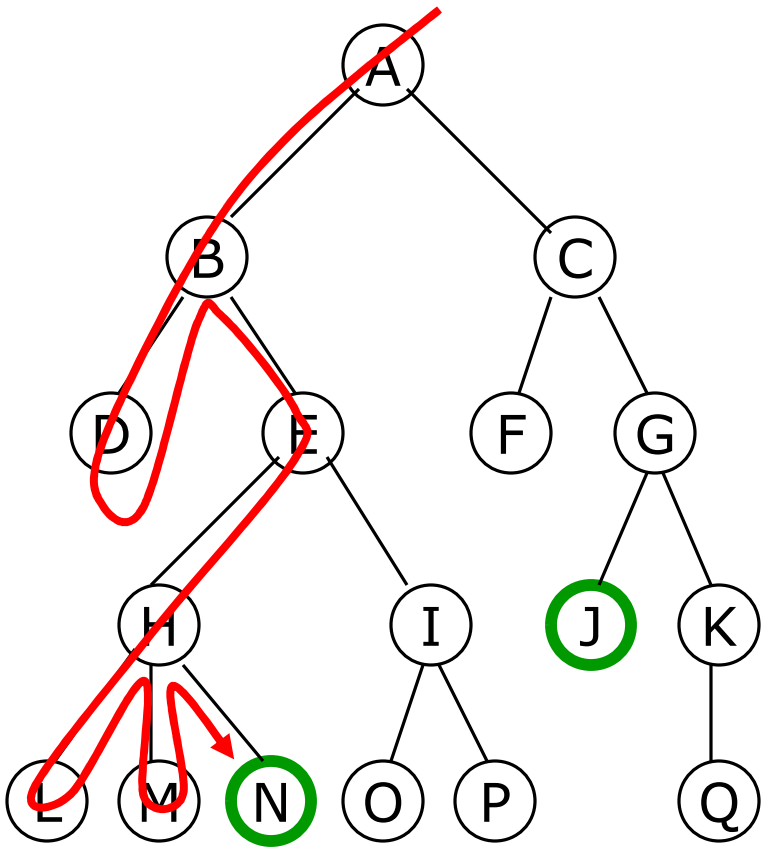
- Forward search: from start to goal
- Backward search: from goal to start
- Bidirectional search

- In expert systems:

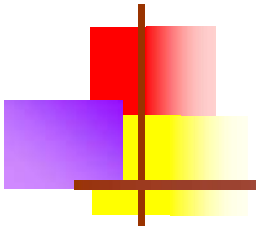
  Forward chaining

  Backward chaining

# 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



# Depth-first Algorithm

- The depth-first algorithm is:

Step-1: Form a one element queue Q consisting of the root node.

Step-2: Until the Q is empty or the goal has been reached,

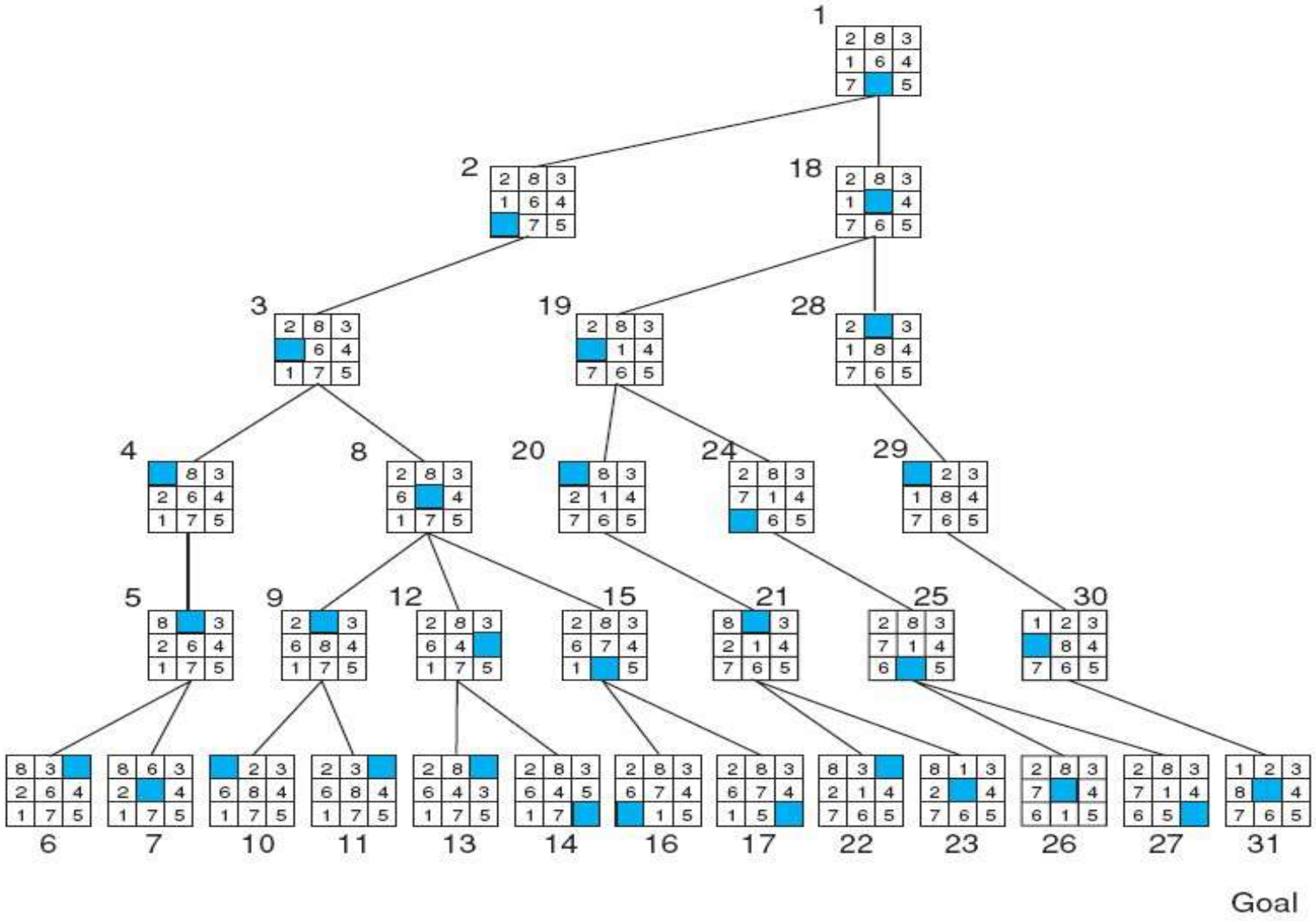
determine if the first element in the Q is the goal.

a) If it is, do nothing.

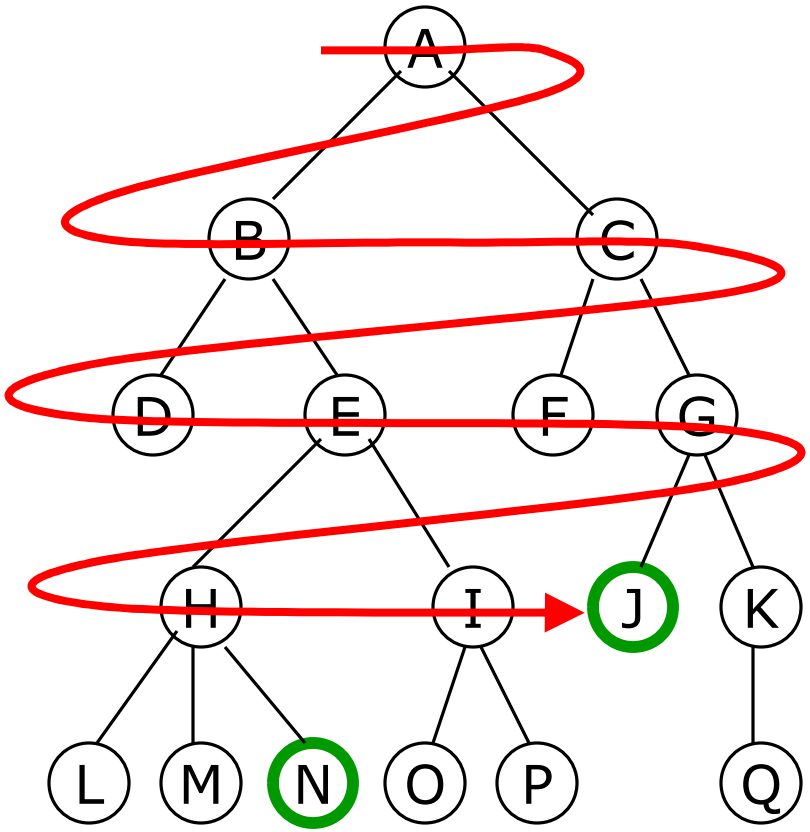
b) If is isn't, remove the first element from the Q and add the first element's children, if any, to the FRONT of the Q.

Step-3: If the goal is reached, success; else failure.

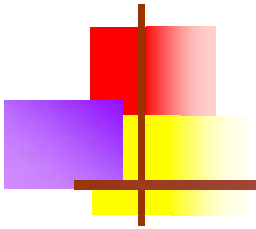
# DFS for Puzzle Problem



# 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



# Breadth-first Algorithm

- The breadth-first algorithm is:

Step-1: Form a one element queue Q consisting of the root node.

Step-2: Until the Q is empty or the goal has been reached,

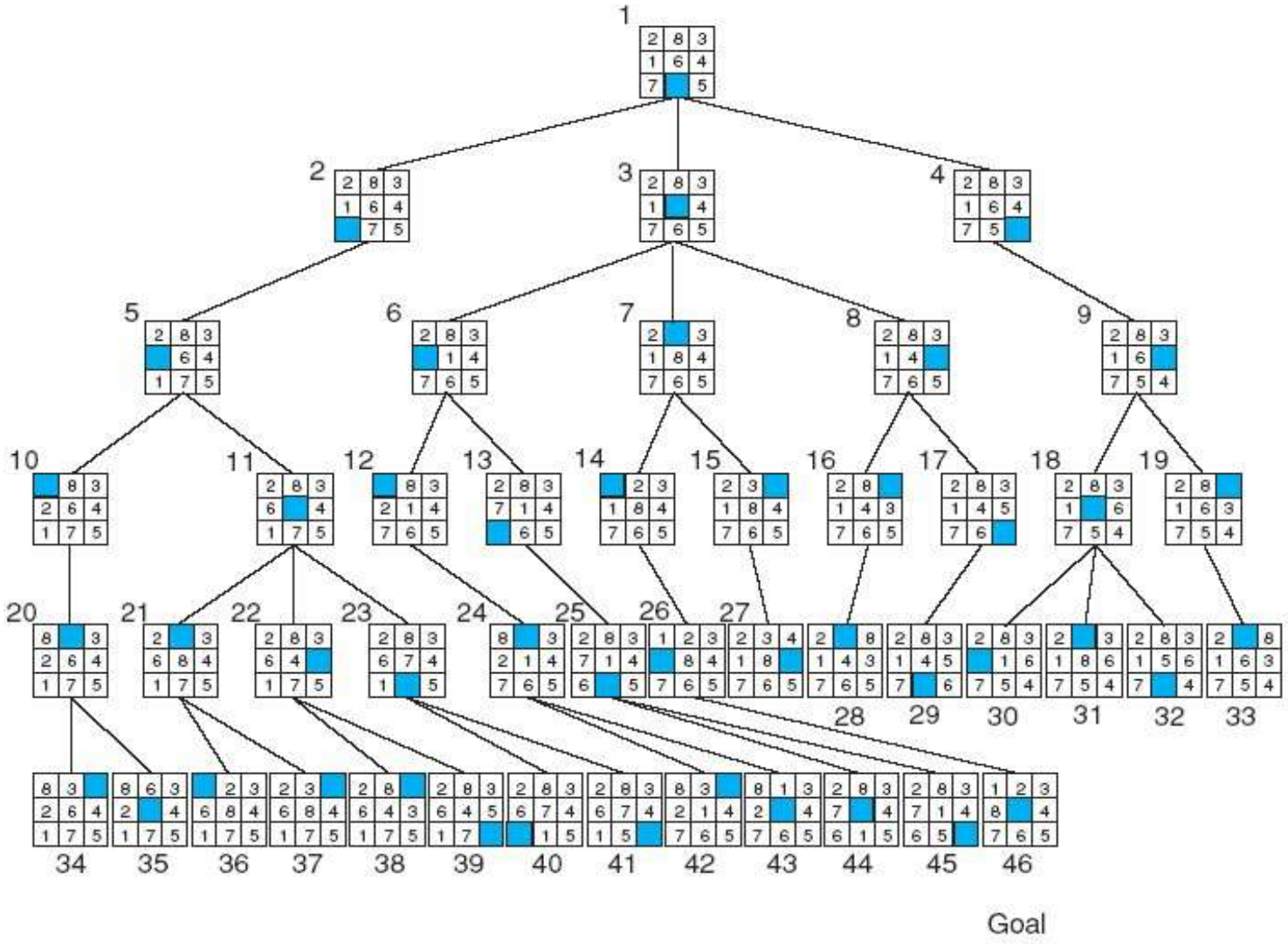
determine if the first element in the Q is the goal.

a) If it is, do nothing.

b) If is isn't, remove the first element from the Q, and add the first element's children, if any, to the BACK of the Q.

Step-3: If the goal is reached, success; else failure.

# BFS for Puzzle Problem



**Problem Solving by Search**

---

**TO BE CONTINUED…**