PROBLEM SOLVING WITH SEARCH

Introduction

- The solution is a fixed sequence of actions
- Search is the process of looking for the sequence of actions that reaches the goal
- Once the agent begins executing the search solution, it can ignore its percepts (**open-loop system**)
- Search control strategies
  - Recursive
  - Iteration
- Search Problems
  - Toy and Real world problems

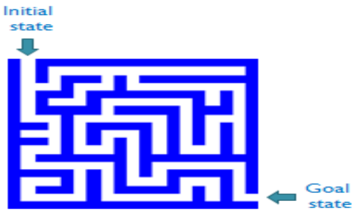
Why study Game

- Games are fun!
- Clear criteria for success
- Games often define very large search spaces
- Games can be a good model of many competitive activities
  - Military confrontations, negotiation, auctions,
- Games are a traditional hallmark of intelligence

Game vs. Search

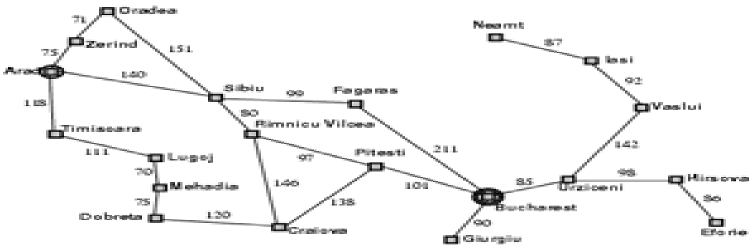
- Search – no adversary
  - Solution is (heuristic) method for finding goal
  - Heuristics and search techniques can find *optimal* solution
  - Evaluation function: estimate of cost from start to goal through given node
  - Examples: path planning, scheduling activities
- Games – adversary
  - Solution is strategy (strategy specifies move for every possible opponent reply).
  - Time limits force an *approximate* solution
  - Evaluation function: evaluate “goodness” of game position
  - Examples: chess, checkers, Othello, backgammon

Search Problem Components



- **Initial state**
- **Actions**
- **Transition model**
  - What is the result of performing a given action in a given state?
- **Goal state**
- **Path cost**
  - Assume that it is a sum of nonnegative *step costs*
- The **optimal solution** is the sequence of actions that gives the lowest path cost for reaching the goal

Example: Romania

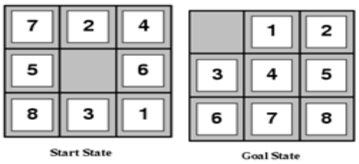


- On vacation in Romania; currently in Arad, Flight leaves tomorrow from Bucharest
- **Initial state:** Arad
- **Actions:** Go from one city to another
- **Transition model:** If you go from city A to city B, you end up in city B
- **Goal state:** Bucharest
- **Path cost:** Sum of edge costs

State Space

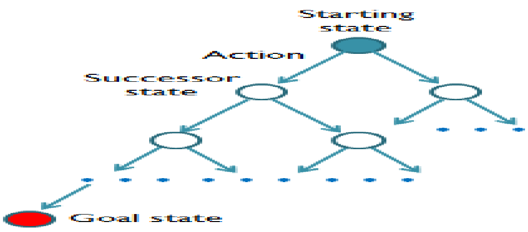
- The initial state, actions, and transition model define the **state space** of the problem
  - The set of all states reachable from initial state by any sequence of actions
  - Can be represented as a **directed graph** where the nodes are states and links between nodes are actions

Example: The 8-puzzle



- **States:** Locations of tiles
  - 8-puzzle: 181,440 states, 15-puzzle: 1.3 trillion states, 24-puzzle: 10^{25} states
- **Actions:** Move blank left, right, up, down
- **Path cost:** 1 per move
- Optimal solution of n-Puzzle is NP-hard

Tree Search

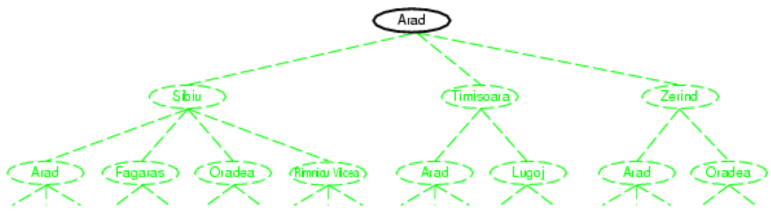


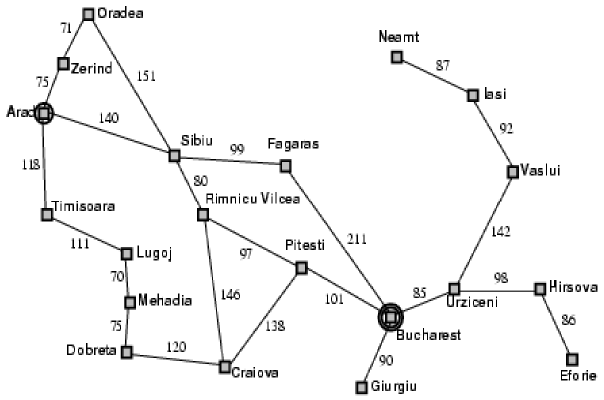
- Let's begin at the start node and **expand** it by making a list of all possible successor states, maintain a **fringe** or a list of unexpanded states, at each step, pick a state from the fringe to expand, keep going until you reach the goal state. Try to expand as few states as possible
- "What if" tree of possible actions and outcomes
- The root node corresponds to the starting state
- The children of a node correspond to the **successor states** of that node's state
- A path through the tree corresponds to a sequence of actions
  - A solution is a path ending in the goal state

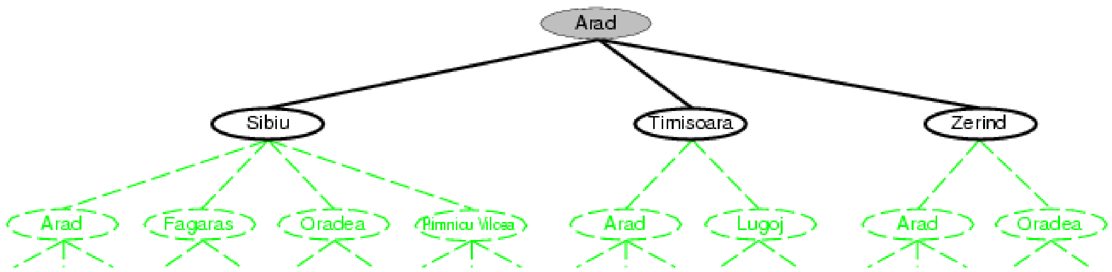
Tree Search Algorithm Outline

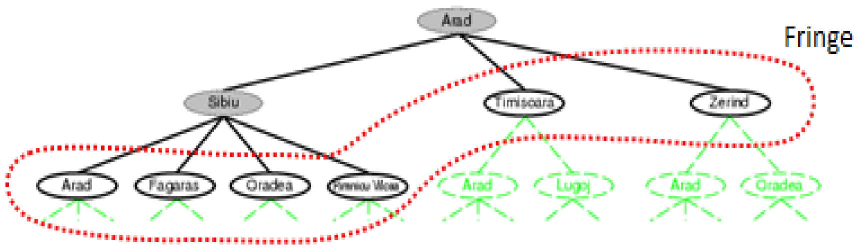
- Initialize the **fringe** using the **starting state**
- While the fringe is not empty
  - Choose a fringe node to expand according to **search strategy**
  - If the node contains the **goal state**, return solution
  - Else **expand** the node and add its children to the fringe

Tree Search Example









Search Strategies

- A **search strategy** is defined by picking the order of node expansion
- Strategies are evaluated along the following dimensions:
  - **Completeness:** does it always find a solution if one exists?
  - **Optimality:** does it always find a least-cost solution?
  - **Time complexity:** number of nodes generated
  - **Space complexity:** maximum number of nodes in memory
- Time and space complexity are measured in terms of
  - *b:* maximum branching factor of the search tree
  - *d:* depth of the least-cost solution
  - *m*: maximum length of any path in the state space (may be infinite)

Uninformed search strategies

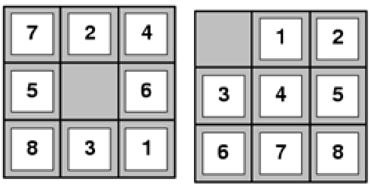
- Uninformed(blind) search strategies use only the information available in the problem definition
- The most common uninformed search strategies are Breadth-first search and Depth-first search

Heuristic function

- Heuristic function $h(n)$ estimates the cost of reaching goal from node n
- Improve the efficiency of search process
- Solve complex problem
- It is knowledge about domain
- Admissible or monotonic
- 8-puzzel problem is solve by heuristics
- The problem is
    - Which 8-puzzel move is best?
    - What heuristic can decided?
    - Which move is best?

8-Puzzel Problem: The puzzle consists of 3x3 grid

- **State space:** The configuration of 8-tiels on the board
- **Solution:** The optimal sequence of the operators
- **Action:** Move towards the black spaces
    - Condition: the move is with in the board
    - Direction: right, left, up and down
- To find which move is best?
- We can apply 3 different heuristic approaches
    - Count the **<u>correct positions</u>** of each tile, compare with the goal state.
    - Count the **<u>incorrect position</u>** of each tile, compare with the goal state.
    - Count **<u>how far away</u>** each tile is from its correct potion.
- Manhattan distance
- In the first approach
    - Easy to compute(fast and take less memory)
    - Probably simplest heuristics
    - The highest number return by the heuristic is the best move
- In the second approach
    - The lowest number return by the heuristic is the best move
- In the last approach
    - The smallest number return by the heuristic is the best move
- Heuristics for the 8-puzzle
    - $h_1(n)$ = number of misplaced tiles
    - $h_2(n)$ = total Manhattan distance (number of squares from desired location of each tile)



Start State Goal State

- h_1(start) = 8
- h_2(start) = 3+1+2+2+2+3+3+2 = 18
- Are h_1 and h_2 admissible?