

## COMP210: Artificial Intelligence

### Lecture 5. Search strategies

Trevor Bench-Capon  
Room 215, Ashton Building

<http://www.csc.liv.ac.uk/~tbc/COMP210>

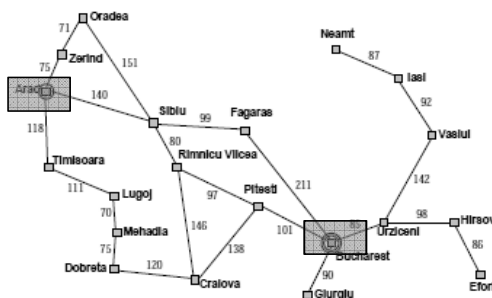
- Last Time
  - basic ideas about problem solving;
  - state space;
  - solutions as paths;
  - the notion of solution cost; and
  - the importance of using the correct level of abstraction.
- Today
  - Automating Search
    - Blind (uninformed, brute force) strategies

### Problem Solving as Search

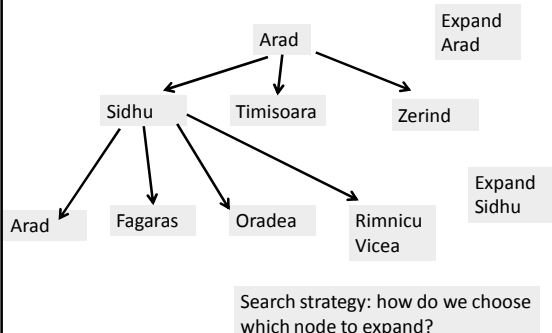
- In the state space view of the world, finding a solution is finding a path through the state space.
- When we (as humans) solve a problem like the 8-puzzle we have some idea of what constitutes the next best move.
- It is hard to program this kind of approach.
- Instead we start by programming the kind of repetitive task that computers are good at.
- A brute force approach to problem solving involves exhaustively searching through the space of all possible action sequences to find one that achieves the goal.

### Example: Romania Problem

Travel from Arad to Bucharest



### The Search Tree



### Search Tree Exploration

- The tree is built by taking the initial state and identifying the states that can be obtained by a single application of the operators available.
- These new states become the *children of the initial* state in the tree.
- These new states are then examined to see if they are the goal state.
- If not, the process is repeated on the new states.
- We can formalise this description by giving an algorithm for it.
- Different algorithms for different choices of nodes to expand

## General Algorithm for Search

```

agenda = initial state;
while agenda not empty do
  pick node from agenda;
  new nodes = apply operations to state;
  if goal state in new nodes then return
    solution;
  else add new nodes to agenda;

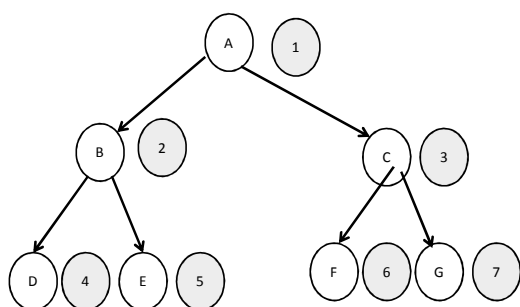
```

- Question: How to pick states for expansion?
- Two obvious strategies:
  - depth first search;
  - breadth first search.

## Breadth First Search

- Start by expanding initial state — gives tree of depth 1.
- Then expand all nodes that resulted from previous step
  - gives tree of depth 2.
- Then expand all nodes that resulted from previous step, and so on.
- Expand nodes at all depth  $n$  before going to level  $n + 1$ .

## Breadth First Search



## General Breadth First Search

```

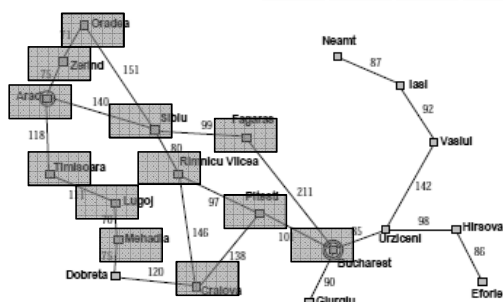
/* Breadth first search */
• agenda = initial state;
while agenda not empty do
  pick node from front of agenda;
  new nodes = apply operations to
    state;
  if goal state in new nodes then
    return solution;
  • else APPEND new nodes to END of
    agenda

```

## Example: Romania BFS

Travel from Arad to Bucharest

D=0   D=1   D=2   D=3



## Properties of Breadth First Search

- Advantage: guaranteed to reach a solution if one exists.
- Finds the shortest (cheapest, best) solution in terms of the number of operations applied to reach a solution.
- Disadvantage: time taken to reach solution.
  - Let  $b$  be branching factor — average number of operations that may be performed from any level.
  - If solution occurs at depth  $d$ , then we will look at  $1 + b + b^2 + \dots + b^d$  nodes before reaching solution — exponential.
  - The memory requirement is  $bd$



**Today:**  
**Basic Search Strategies**

- Breadth-first search is complete but expensive.
- Depth-first search is cheap but incomplete
- Can't we do better than this?
- Next time
  - Prolog continued
- Next week
  - Improving on blind search