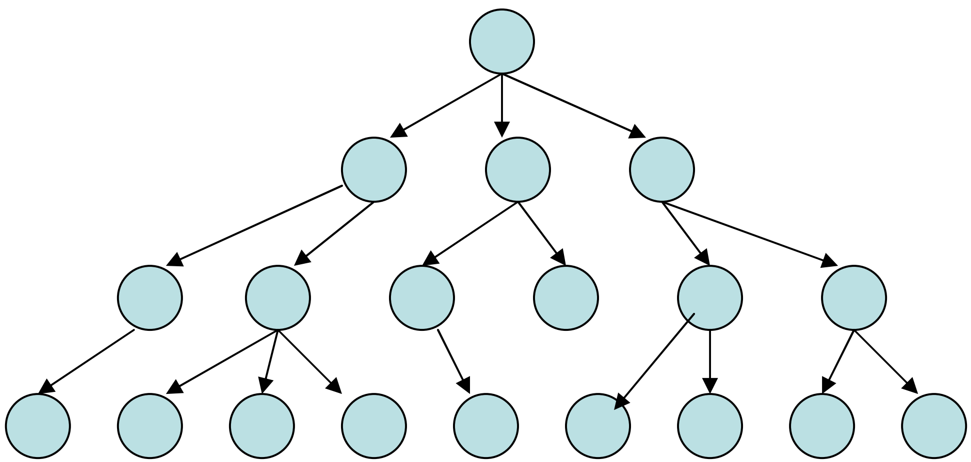
Solving Problems by Searching



Terminology

- State
- State Space
- Goal
- Action
- Cost
- State Change Function
- Problem-Solving Agent
- State-Space Search

Formal State-Space Model

Problem = (S, s, A, f, g, c)

S = state space	
s = initial state	
A = actions	
f = state change function	f: S x A -> S
g = goal test function	g: S -> {true,false}
c = cost function	c: S x A x S -> R

$$x \xrightarrow{a} y$$

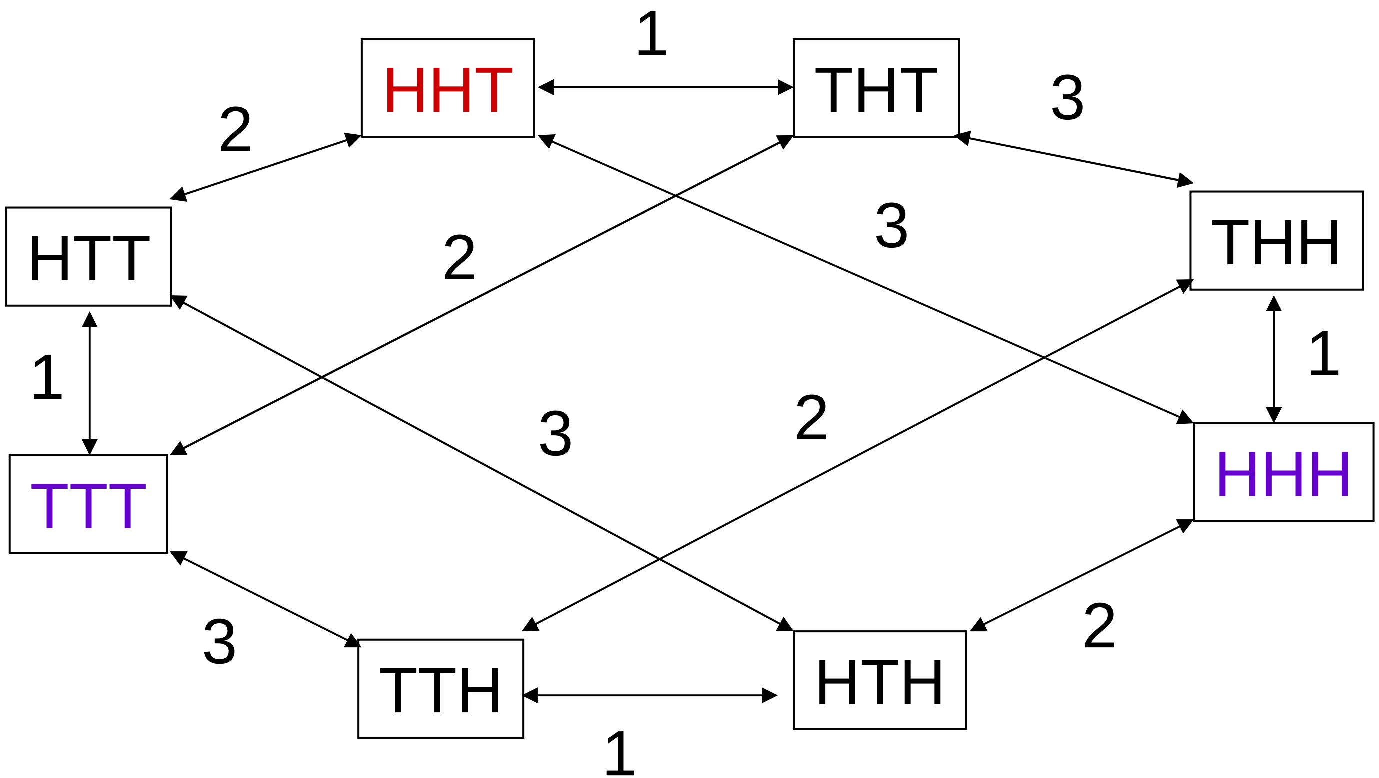
- How do we define a solution?
- How about an optimal solution?

3 Coins Problem
A Very Small State Space Problem

- There are 3 (distinct) coins: coin1, coin2, coin3.
- The initial state is H H T
- The legal operations are to turn over exactly one coin.
  - 1 (flip coin1), 2 (flip coin2), 3 (flip coin3)
- There are two goal states: H H H
  T T T

What are S, s, A, f, g, c ?

State-Space Graph



- What are some solutions?
- What if the problem is changed to allow only 3 actions?

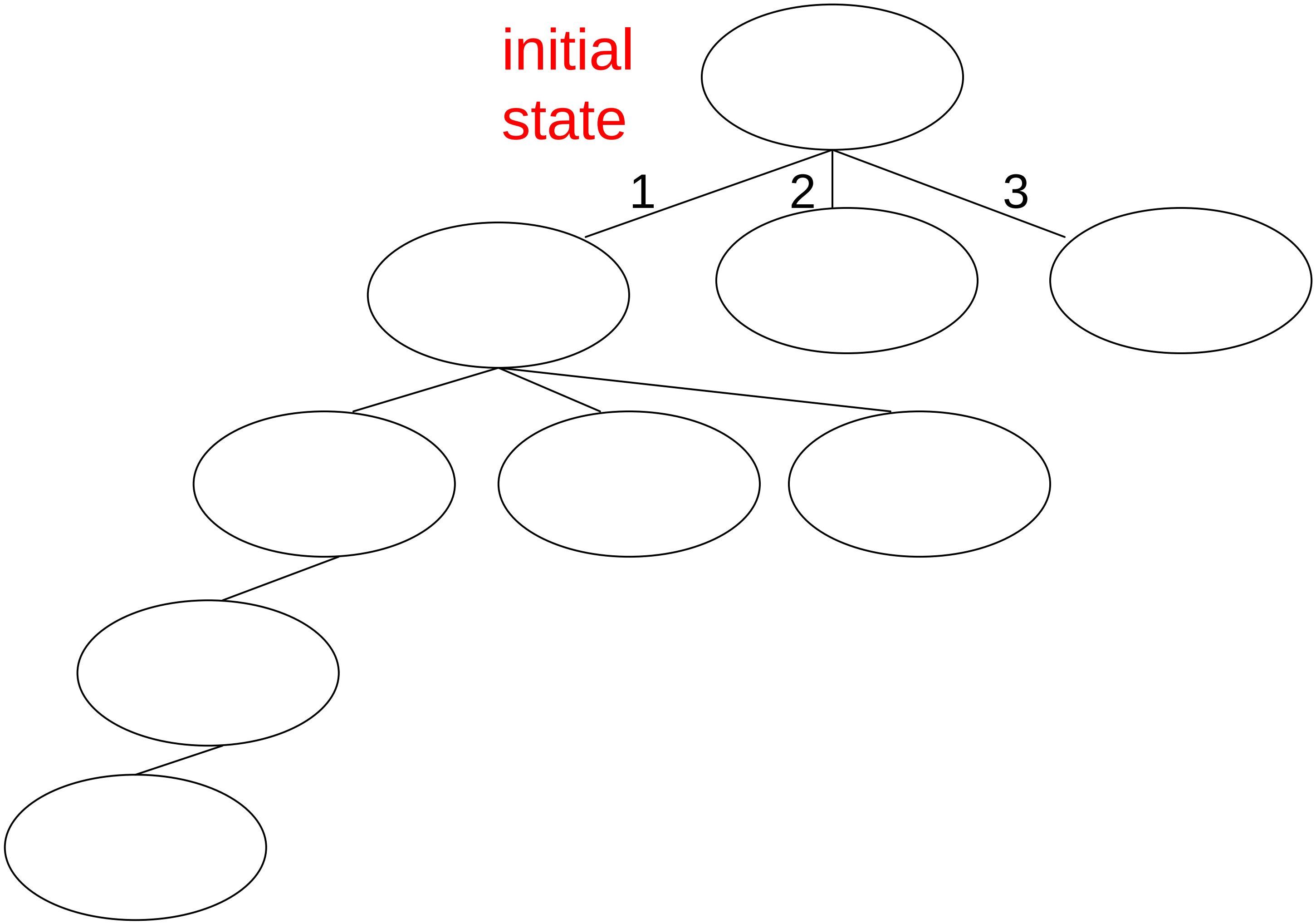
Modified State-Space Problem

- How would you define a state for the new problem?

- How do you define the operations (1, 2, 3) with this new state definition?

- What do the paths to the goal states look like now?

How do we build a search tree for the modified 3 coins problem?



The 8-Puzzle Problem

one initial state

7	2	4
5	B	6
8	3	1

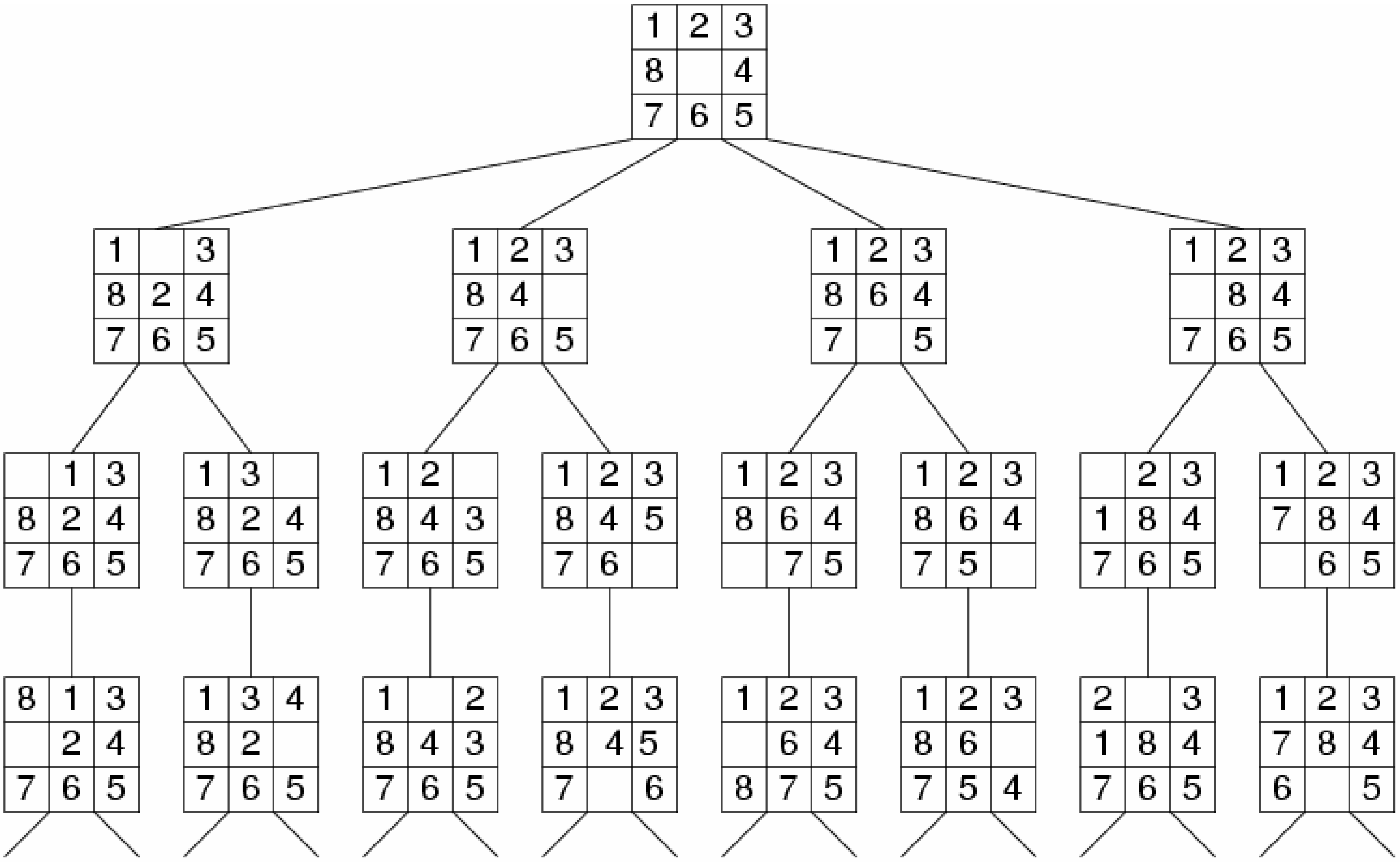
goal state

B	1	2
3	4	5
6	7	8

B=blank

1. Formalize a state as a data structure
2. Show how start and goal states are represented.
3. How many possible states are there?
4. How would you specify the state-change function?
5. What is the goal test?
6. What is the path cost function?
7. What is the complexity of the search?

Search Tree Example: Fragment of 8-Puzzle Problem Space



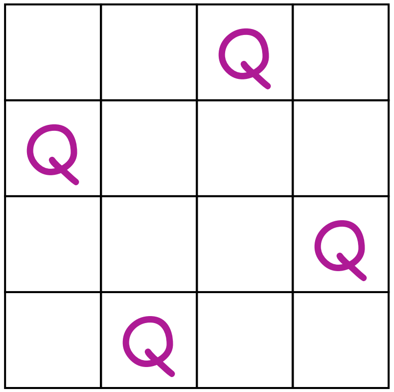
Another Example: N Queens

- Input:
  - Set of states

  - Operators [and costs]

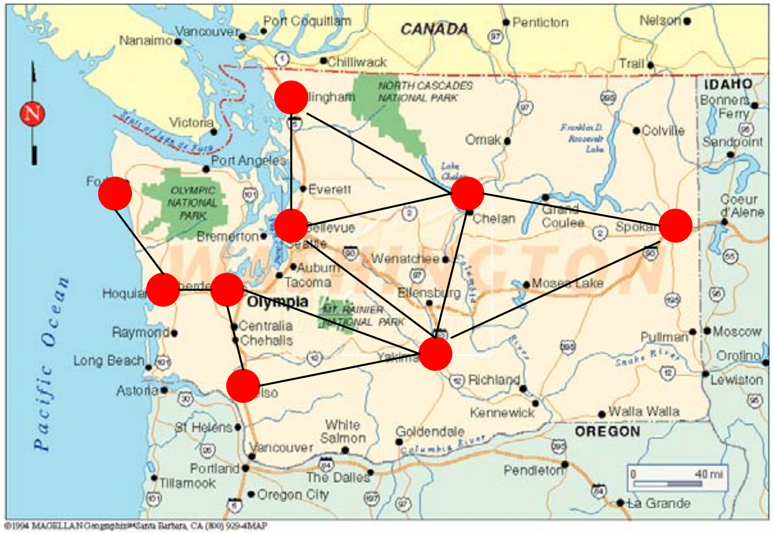
  - Start state

  - Goal state (test)



- Output

Example: Route Planning



- Input:
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal state (test)
- Output:

Search Strategies

- Blind Search (Ch 3)
  - Depth first search
  - Breadth first search
  - Depth limited search
  - Iterative deepening search
- Informed Search (Ch 4)
- Constraint Satisfaction (Ch 5)

Depth First Search

- Maintain stack of nodes to visit
- Evaluation
  - Complete?

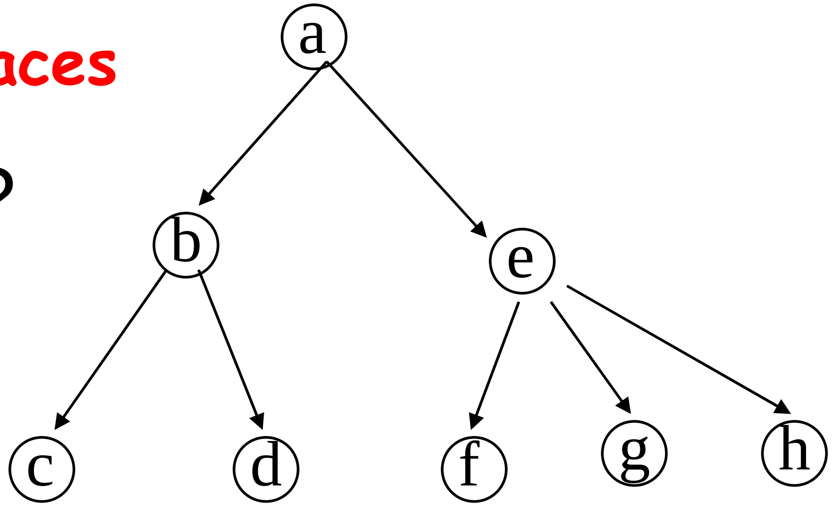
    Not for infinite spaces

  - Time Complexity?

    O(b^d)

  - Space ?

    O(d)



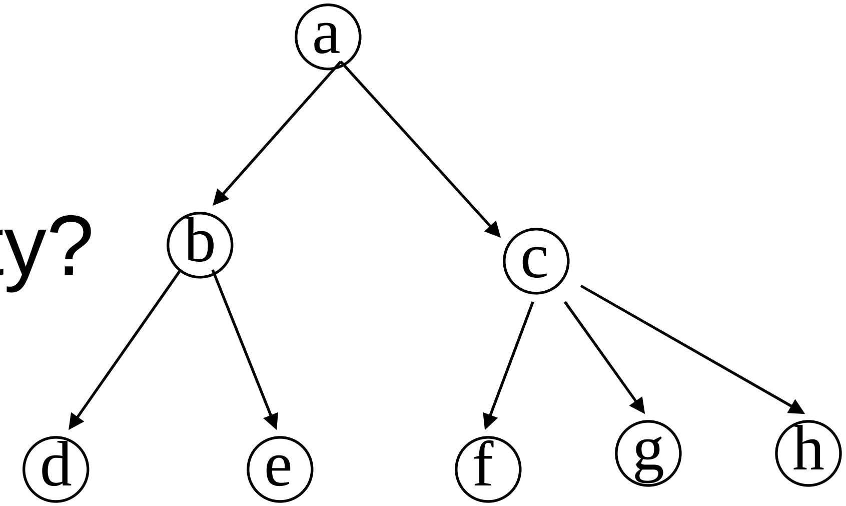
Breadth First Search

- Maintain queue of nodes to visit
- Evaluation
  - Complete?

    Yes
  - Time Complexity?

    O(b^d)
  - Space?

    O(b^d)

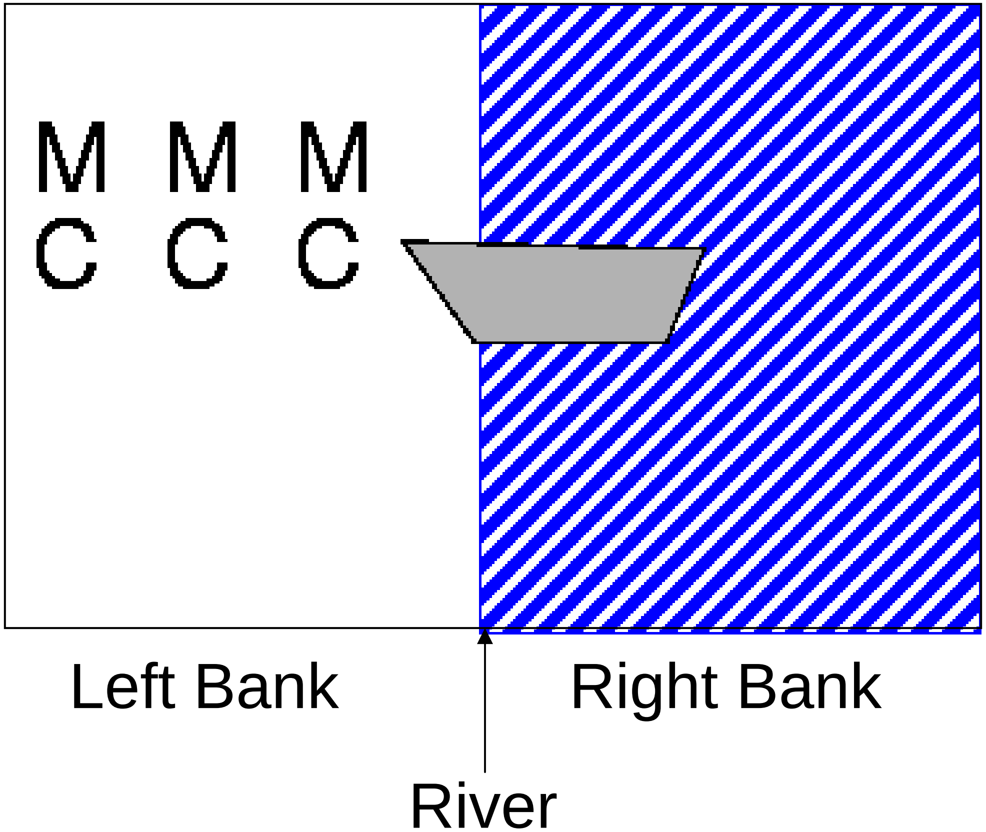


The Missionaries and Cannibals Problem (from text problem 3.9)

- Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people.
- If there are ever more cannibals than missionaries on one side of the river, the cannibals will eat the missionaries. (We call this a “dead” state.)
- Find a way to get everyone to the other side, without anyone getting eaten.

Missionaries and Cannibals Problem

Missionaries and Cannibals Problem



Missionary and Cannibals Notes

- Define your state as (M,C,S)
  - M: number of missionaries on left bank
  - C: number of cannibals on left bank
  - S: side of the river that the boat is on

- When the boat is moving, we are in between states. When it arrives, everyone gets out.

When is a state considered “DEAD”?

1. There are more cannibals than missionaries on the left bank. (Bunga-Bunga)

2. There are more cannibals than missionaries on the right bank. (Bunga-Bunga)

3. There is an ancestor state of this state that is exactly the same as this state. (Why?)

Assignment (problem 3.9b, which is part of the first homework set)

- Implement and solve the problem
  - First with a blind depth-first search using a stack and/or recursion.
  - Second with a blind breadth-first search.
  - Definitely avoid repeated states.
  - Keep track of how many states are searched in each.
- Use the computer language of your choice for this assignment.
  - Java
  - C++
  - Lisp or Lisp variant

Is memory a limitation in search?

- Suppose:
  - 2 GHz CPU
  - 1 GB main memory
  - 100 instructions / expansion
  - 5 bytes / node

  - 200,000 expansions / sec
  - Memory filled in 100 sec … < 2 minutes

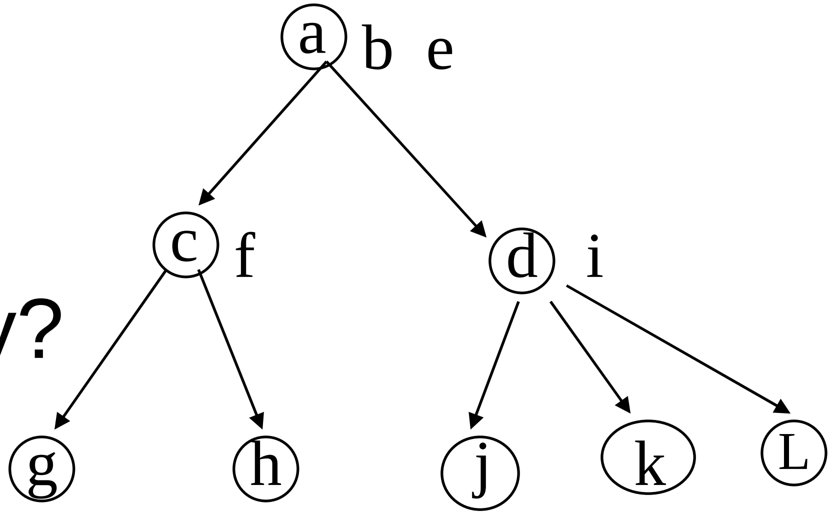
Iterative Deepening Search

- DFS with limit; incrementally grow limit
- Evaluation
  - Complete?

    Yes
  - Time Complexity?

    O(b^d)
  - Space Complexity?

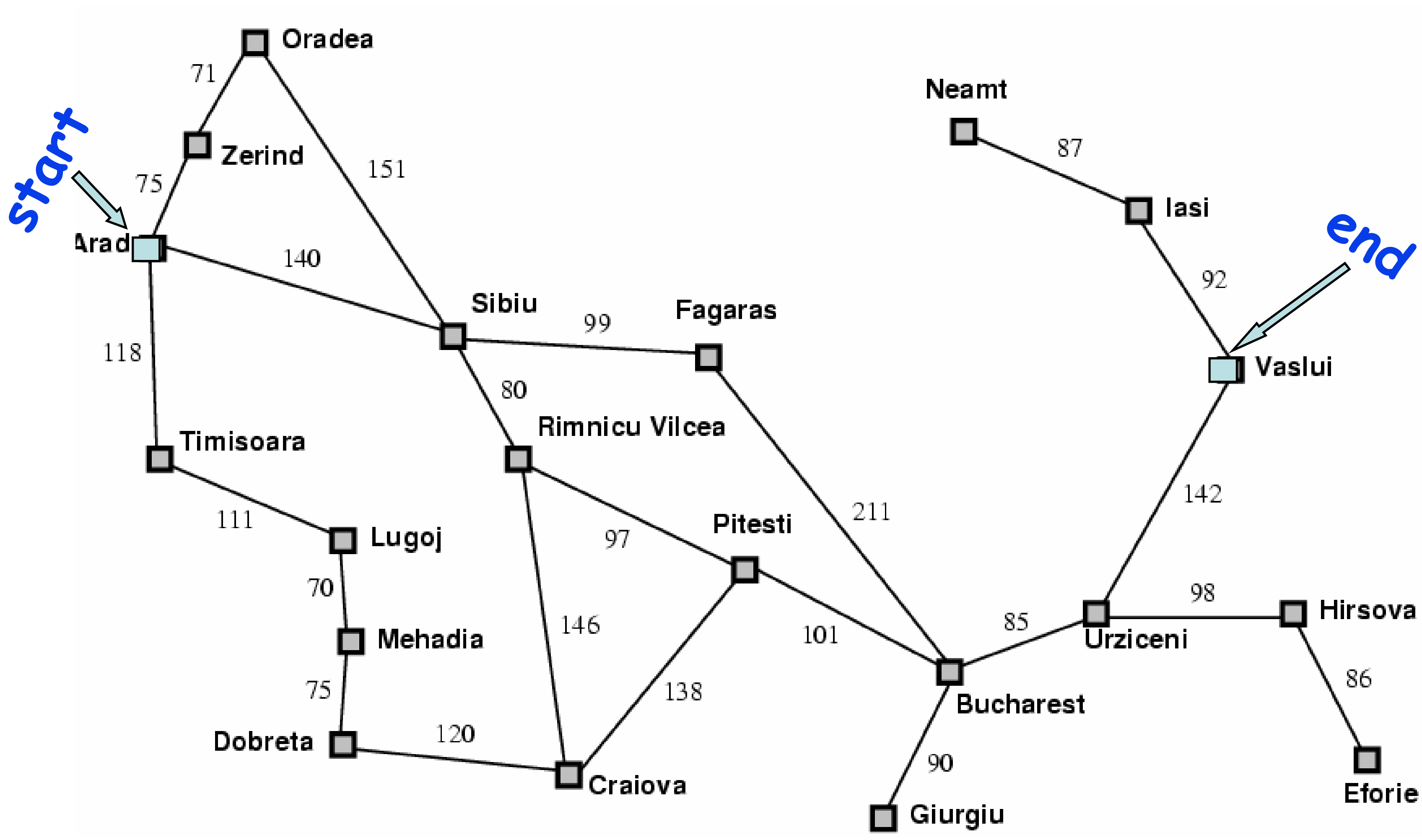
    O(d)



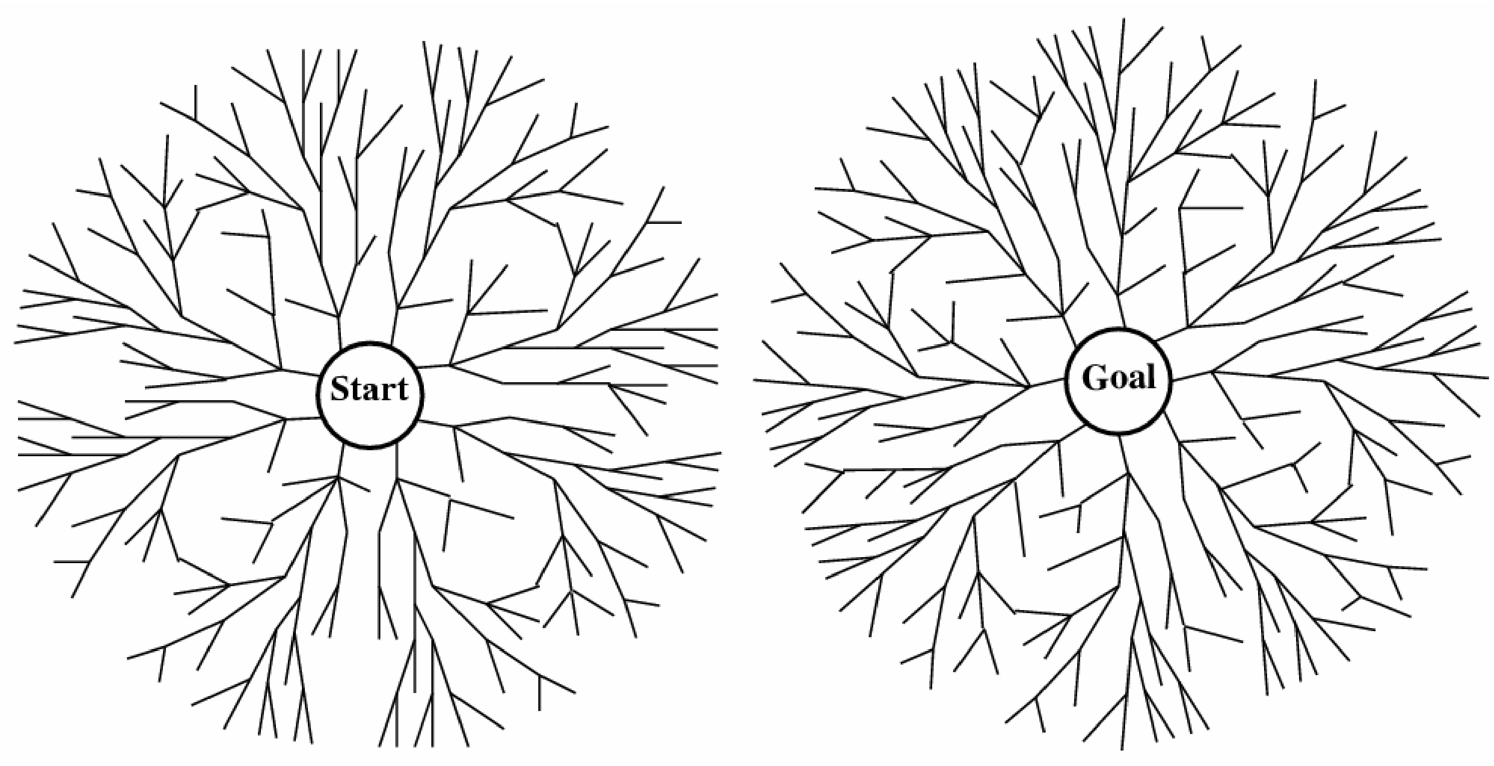
Cost of Iterative Deepening

b	ratio ID to DFS
2	3
3	2
5	1.5
10	1.2
25	1.08
100	1.02

Forwards *vs.* Backwards



vs. Bidirectional



Problem

- All these methods are too slow for real applications (blind)
- Solution → add guidance
- → "informed search"