Problem Solving as Search

CMPSCI 383
September 15, 2011

Today's lecture

- Problem-solving as search
- Uninformed search methods
- Problem abstraction

Bold Claim:

Many problems faced by intelligent agents,
when properly formulated,
can be solved by a single family of generic approaches.

Search

"...an agent with several immediate options of unknown value can decide what to do by first examining different possible sequences of actions that lead to states of known value."

(Russell & Norvig, p. 65)

Search terminology

- This process looks for sequences of actions that are solutions to some problem
- This state occurs at the start of a solution
- This is the set of states that are all reachable from the initial state
- A state that is the end result of the search

- *What is* **Search?**
- *What is the* **Initial state?**
- *What is the* **State space?**
- What is a **Goal state?**

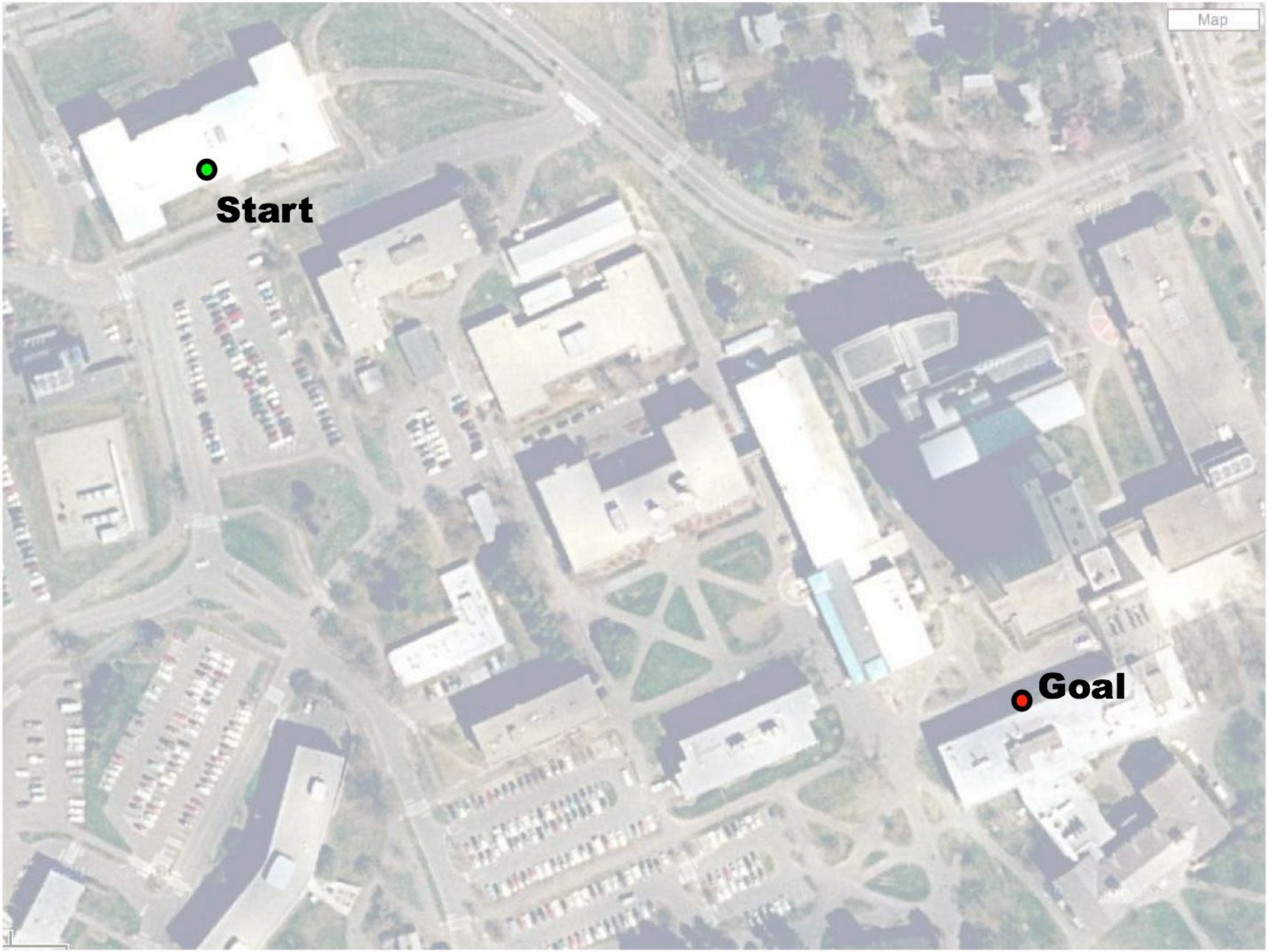
Some More Search terminology

- Applicable action
- Transition model
- Successor
- Goal test
- Path
- Path cost function
- Step cost
- Solution
- Optimal solution

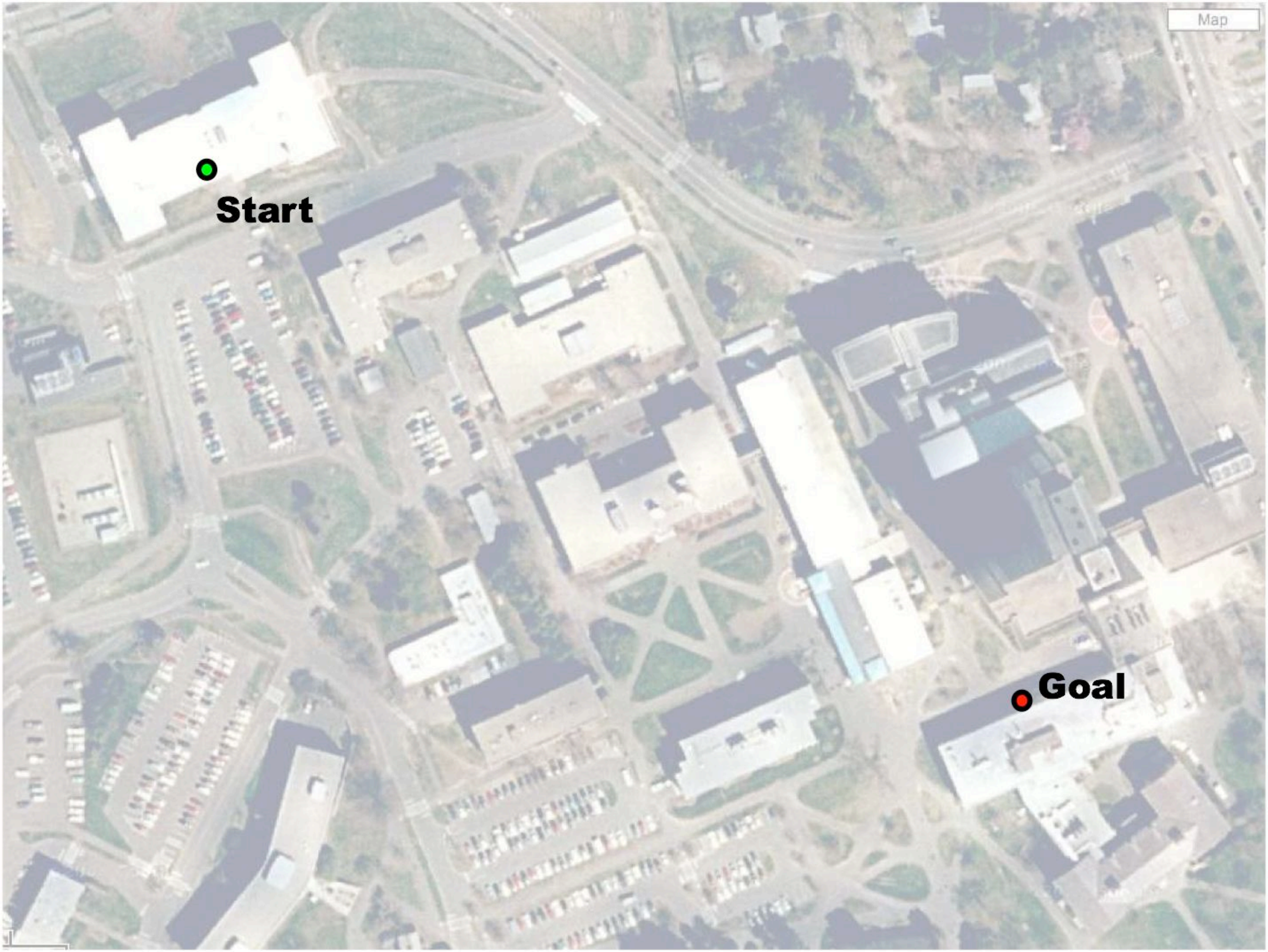
Search basics

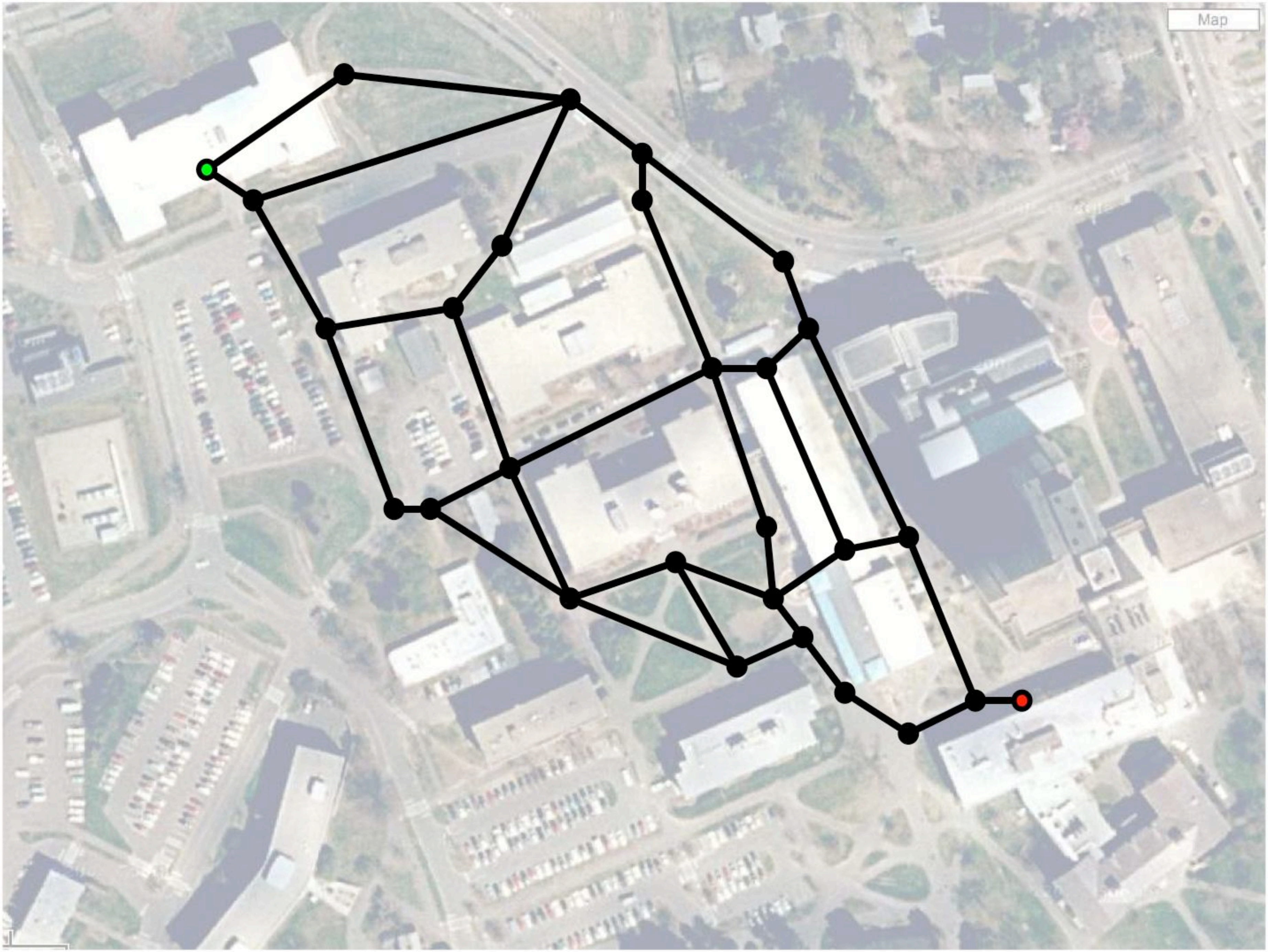
- Basic formulation of search problems
  - Initial state
  - Transition model (or successor function)
  - Goal state
  - Path cost
- Solutions
  - *A path* between the initial and goal states
  - *Quality* is measured by path cost
  - *Optimal solutions* have the lowest cost of any possible path


Map


Map
Start
Goal

How would you represent this as a search problem?


Map
Start
Goal


Map

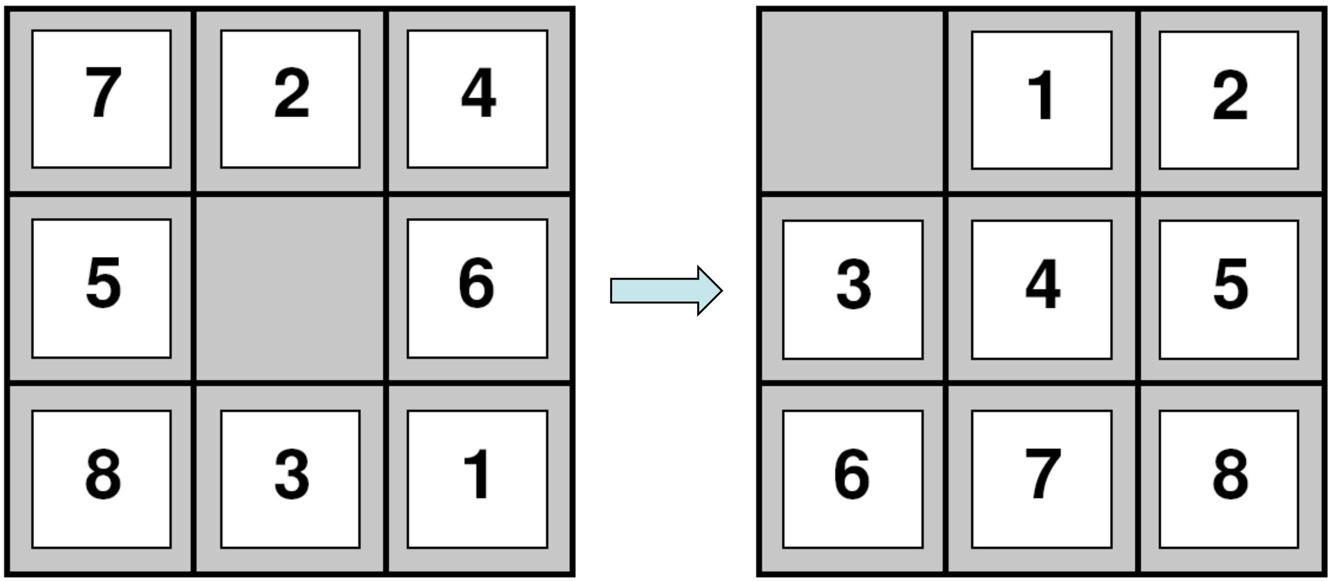
What has been abstracted away?

Assumptions of simple search abstraction

- **Static** — The world does not change on its own, and our actions don't change it.
- **Discrete** — A finite number of individual states exist rather than a continuous space of options.
- **Observable** — States can be determined by observations.
- **Deterministic** — Each action has only one out for each state
- **Known** — Transition model is known

Examples of Toy Problems

Sliding-Blocks Puzzles like the 8 puzzle

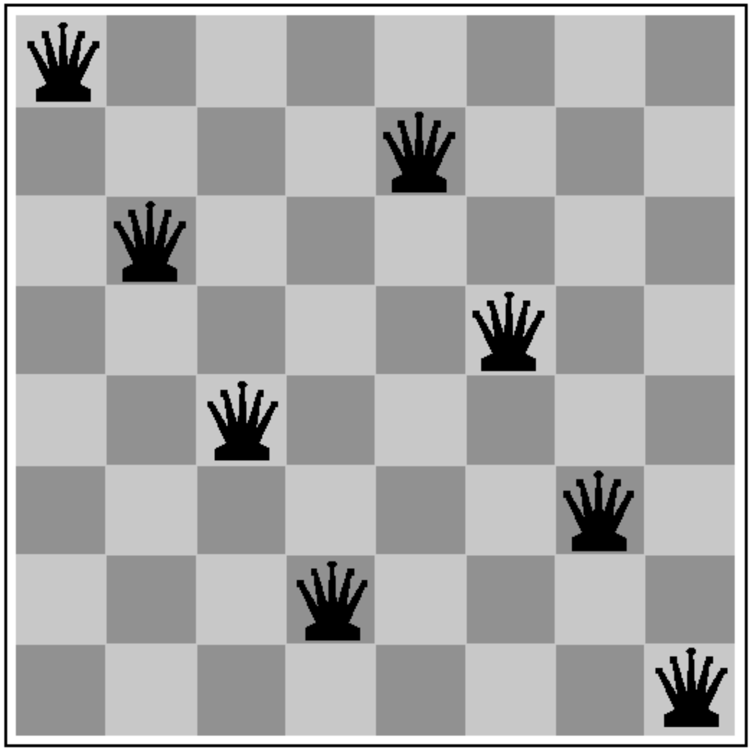


Start State

Goal State

Toy Problems

8 Queens Problem



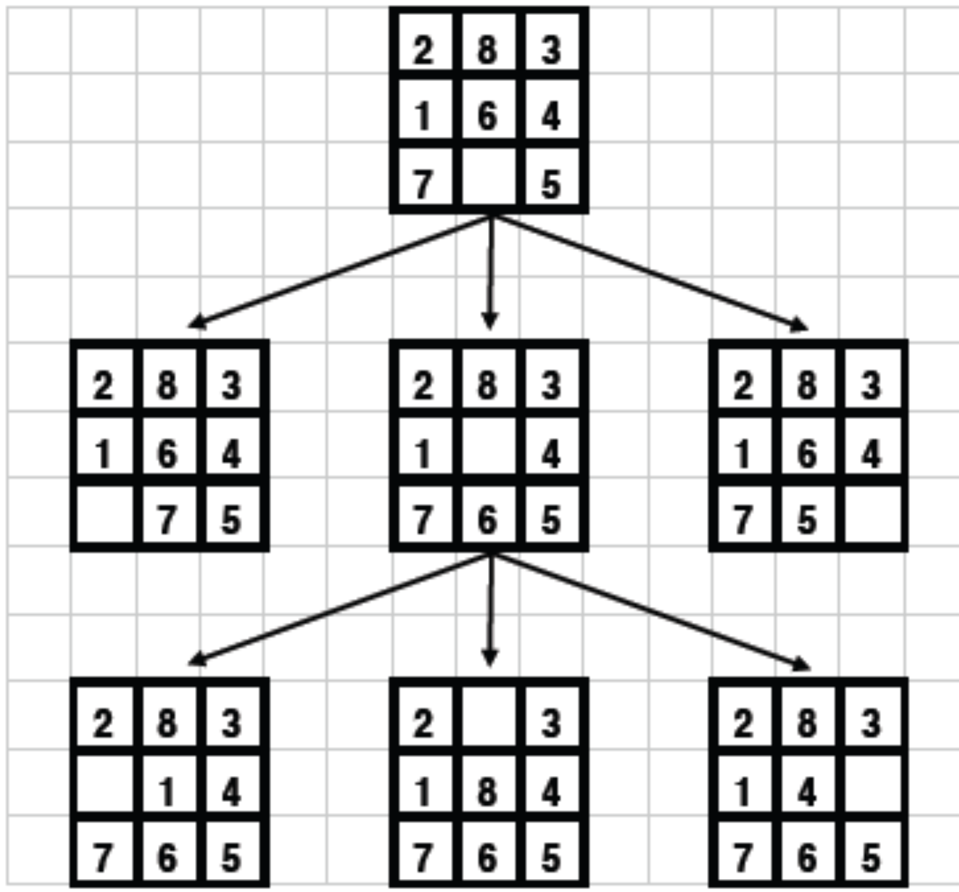
Some Real-World Problems

- Route-Finding problems
- Signal interpretation (e.g. speech understanding)
- Theorem proving (e.g. resolution techniques)
- Combinatorial optimization (e.g. VLSI layout)
- Robot navigation (e.g. path planning)
- Factory scheduling (e.g. flexible manufacturing)
- Symbolic computation (e.g. symbolic integration)
- Protein design
- …

More Search terminology

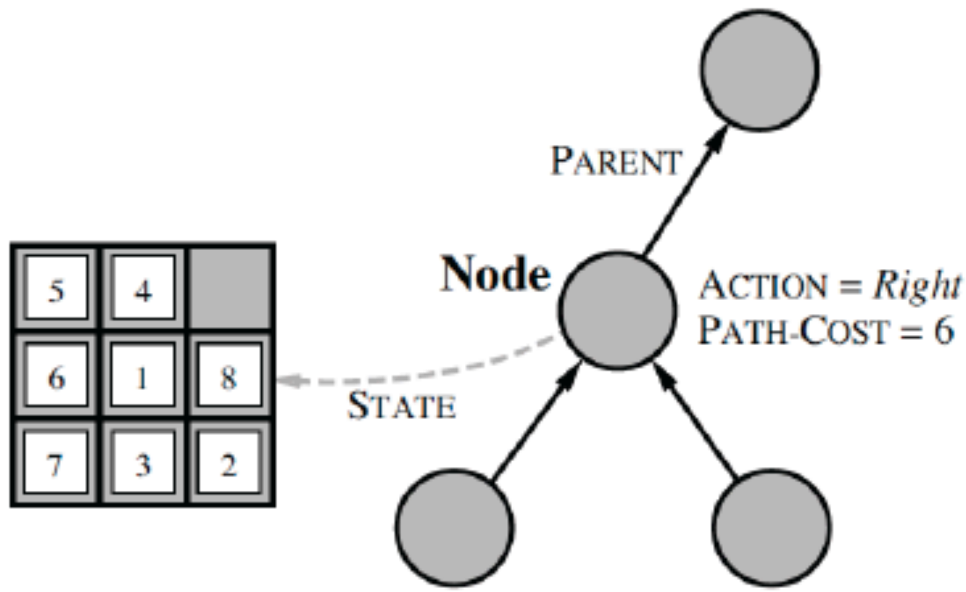
- **Search tree:** data structure is defined by the initial state and the successor function
- **Leaf nodes:** nodes that have no successors in the search tree
- **Node expansion:** operation that adds nodes to a leaf node using the successor function
- **Frontier:** set of all leaf nodes available for expansion (open list)
- **Search strategy:** general approach defines which states to expand in the search tree

Search Tree


2 8 3
1 6 4
7 5
2 8 3
1 6 4
7 5
2 8 3
1 4
7 6 5
2 8 3
1 6 4
7 5
2 8 3
1 4
7 6 5
2 3
1 8 4
7 6 5
2 8 3
1 4
7 6 5

Representing a Node

```
(defstructure node
   state
   parent-node
   operator
   depth
   path-cost)
```



Informal Description of general Tree and Graph-Search Algorithms

function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

---

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

How do you evaluate a search strategy?

- **Completeness** — Does it always find a solution if one exists?
- **Optimality** — Does it find the best solution?
- **Time complexity**
- **Space complexity**

Uninformed search methods

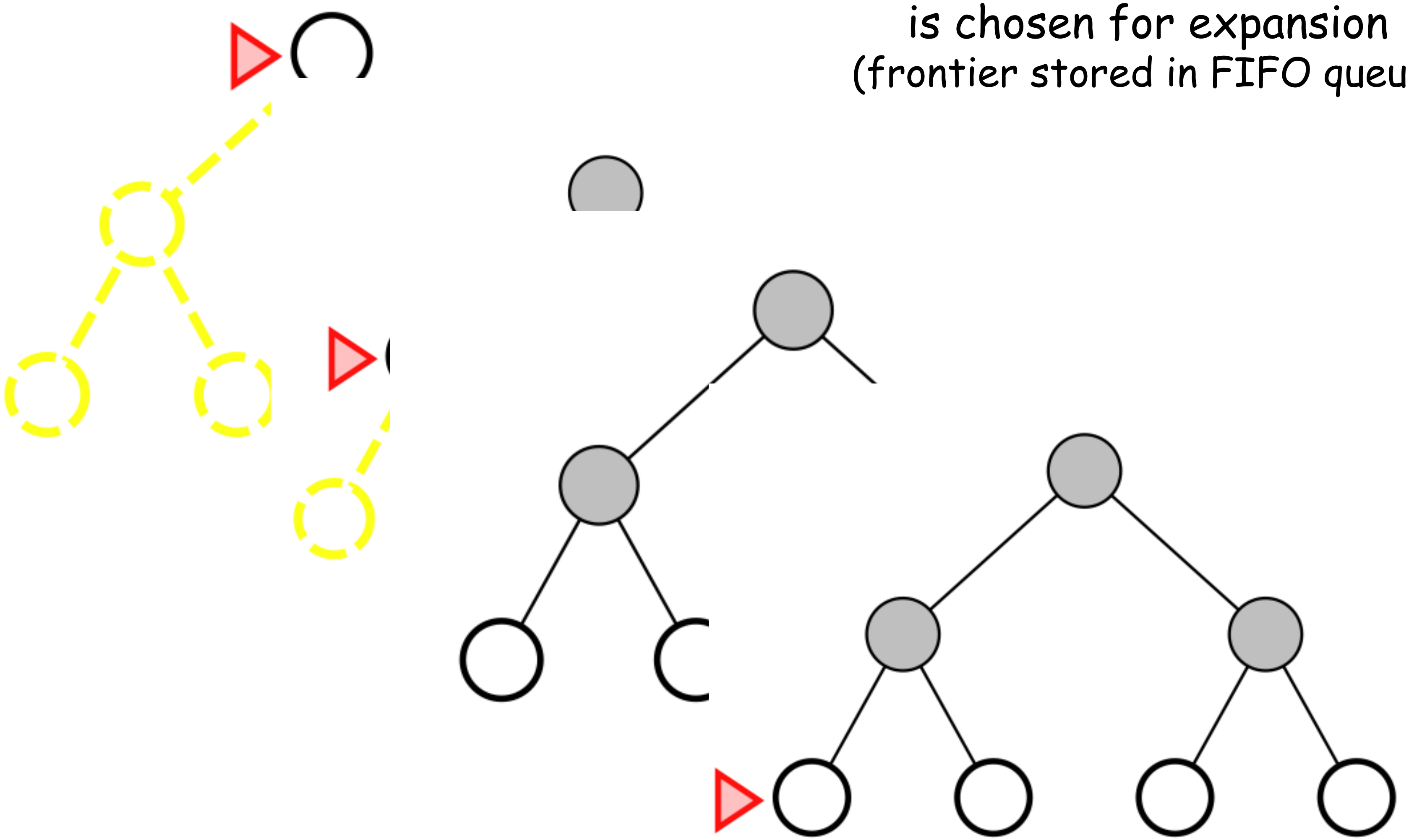
- These methods have no information about which nodes are on promising paths to a solution.
- Also called: *blind search*
- Distinguished by the order in which nodes are expanded.

Some Uninformed Search Strategies

- Breadth-first search
  - Variant — Uniform-cost search
- Depth-first search
- Depth-limited search
- Iterative deepening depth-first search
  - Variant — iterative lengthening search

Breadth-first search

Shallowest unexpanded node
is chosen for expansion
(frontier stored in FIFO queue)



Breadth-first search

- Complete?
- Yes (if *b* finite)

- Optimal?
- Yes, if cost = 1 per step
  Not optimal in general

- Time
- $1+b+b^2+b^3+\ldots+b^d+b(b^d-1) = O(b^{d+1})$

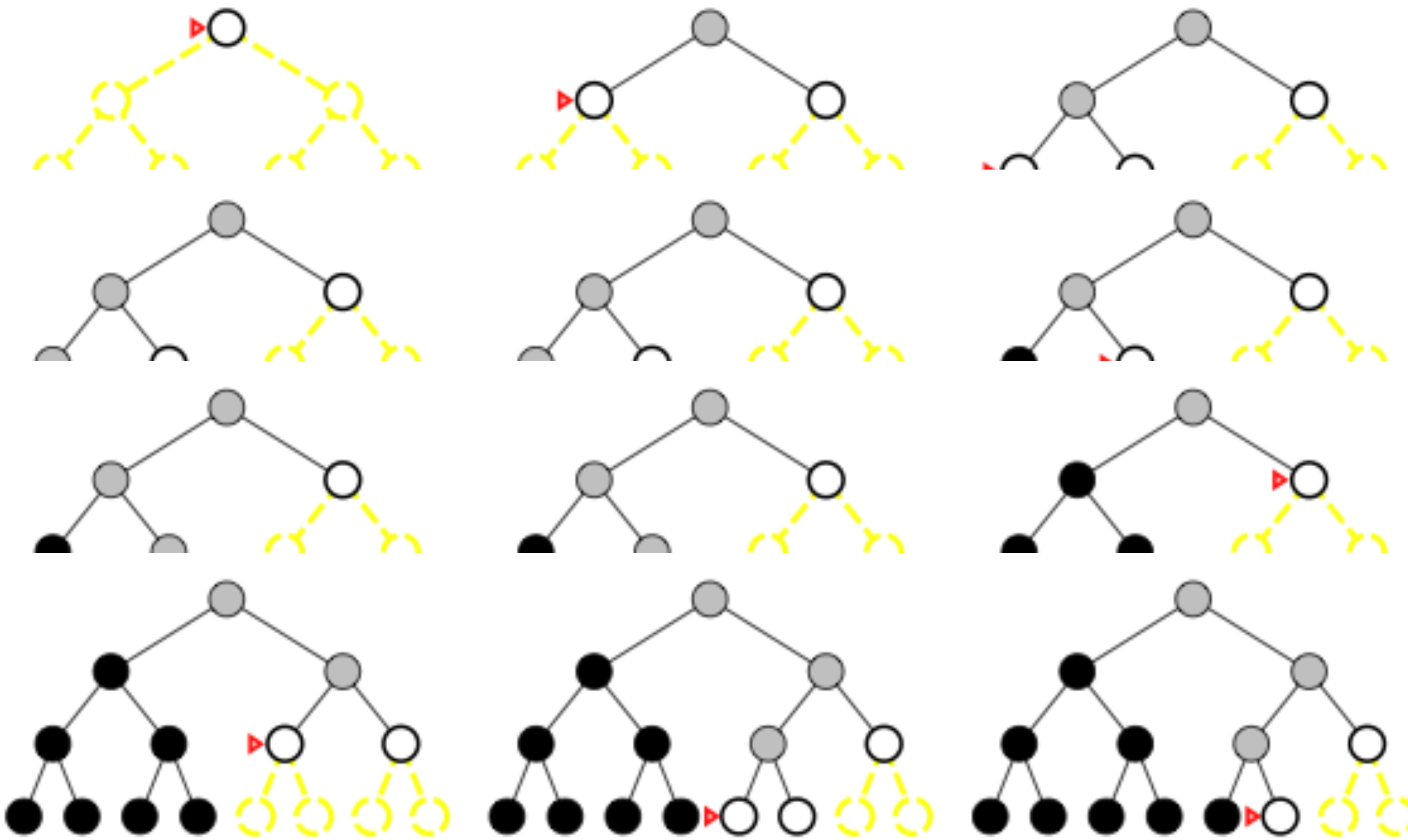
- Space
- $O(b^{d+1})$

Is $O(b^{d+1})$ a big deal?

Depth	Time	Memory

Assuming: b=10, 10K nodes/sec, 1K bytes/node

Depth-first search: Expand deepest node in frontier (LIFO)

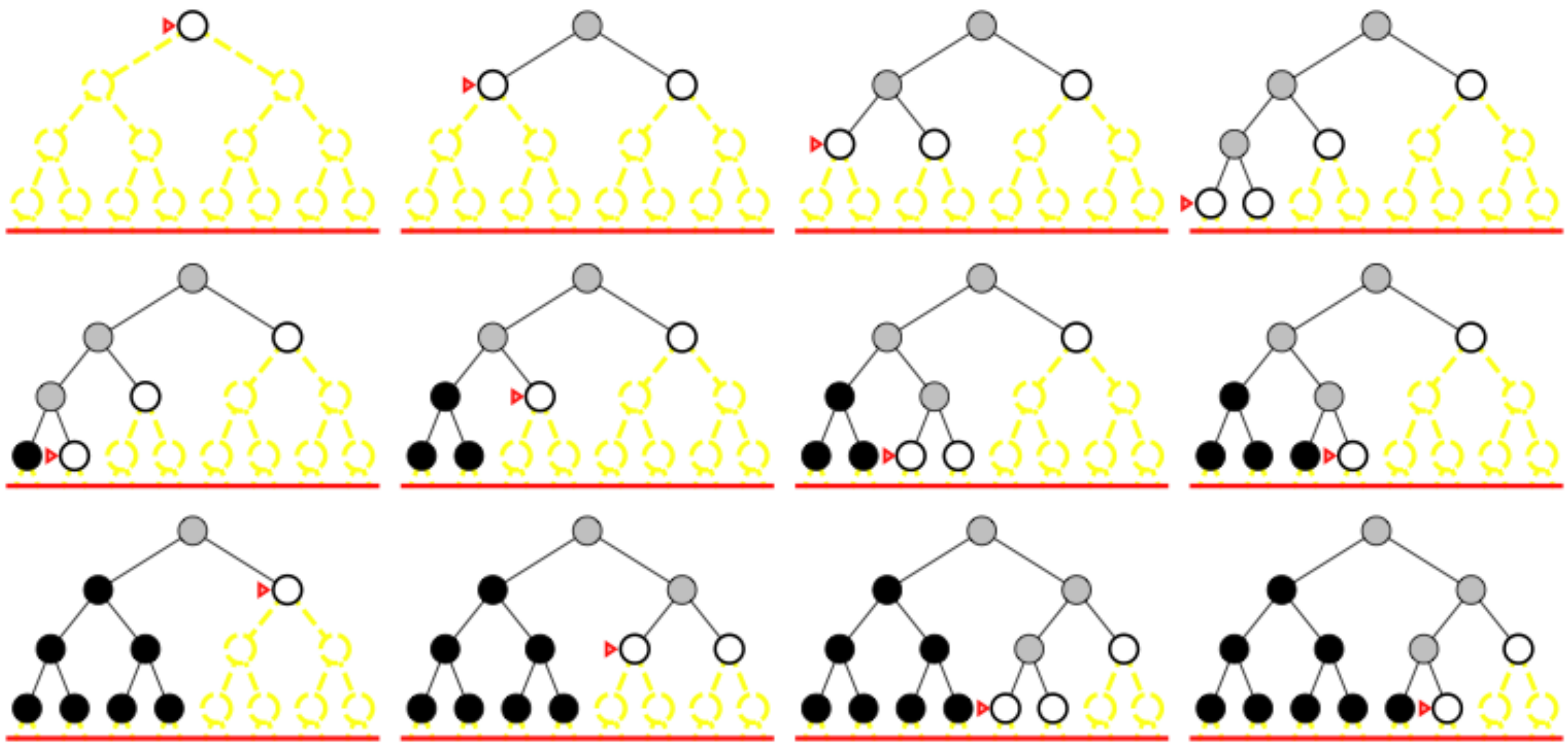


Depth-first search

- Complete?
- Yes, if graph search version and finite
  No, if tree search version*
- Optimal?
- No
- Time
- $O(b^m)$ for tree search version where m is max depth of any node
- Space
- $O(bm)$ for tree search version

Linear space!!

Iterative-deepening search

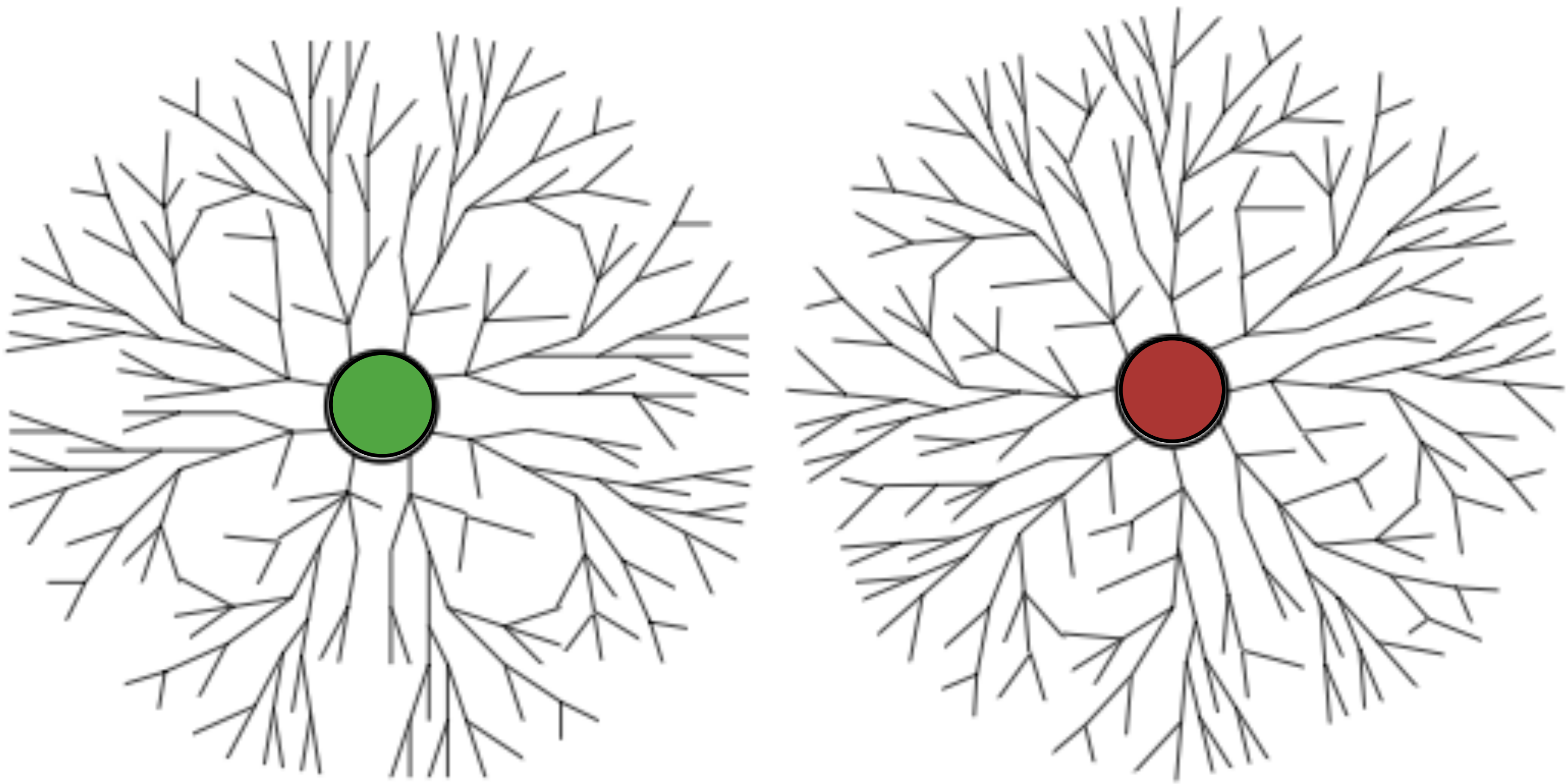


Iterative-deepening search

- Complete?
- Optimal?
- Time
- Space

- Yes
- Yes, if step-cost = 1
- $(d+1)b^0+db^1+(d-1)b^2+\ldots+b^d = O(b^d)$
- $O(bd)$

Bi-directional search



Comparing Blind Search Strategies

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Figure 3.21 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.

Next Class

- We will post Assignment 1 before Monday, the drop with W deadline
- Next class: Informed Search Strategies

  Sections 3.5 – 3.7