Informed Search

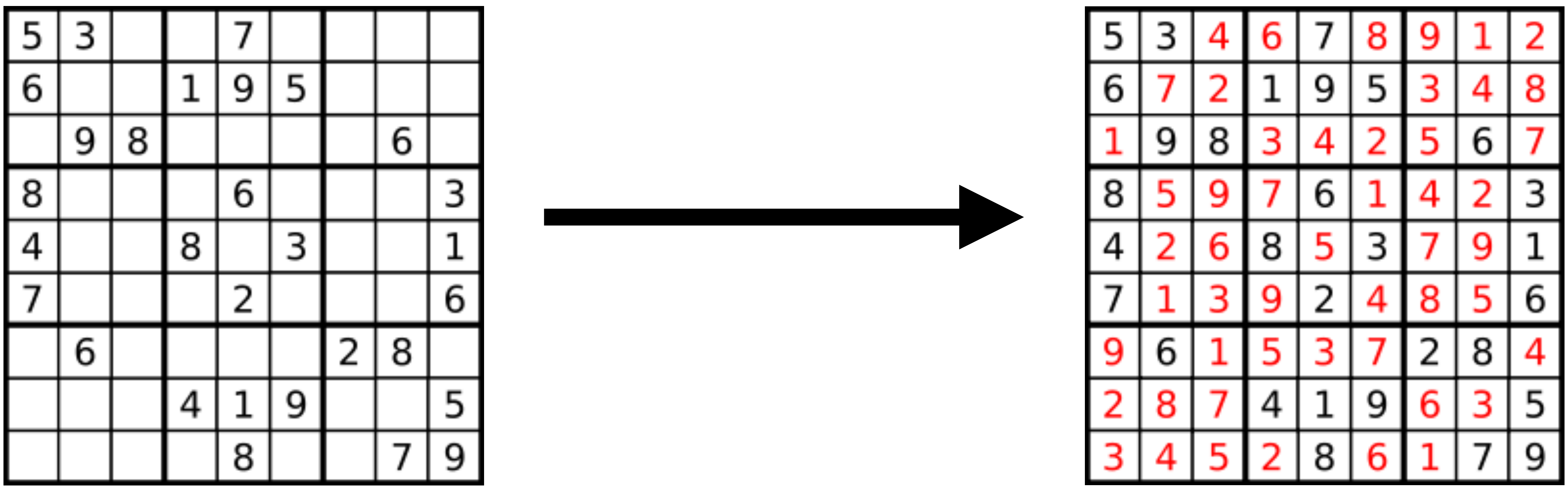
George Konidaris
gdk@cs.duke.edu

DUKE
COMPUTER
SCIENCE

Spring 2015

Recall: Search

DUKE
COMPUTER
SCIENCE

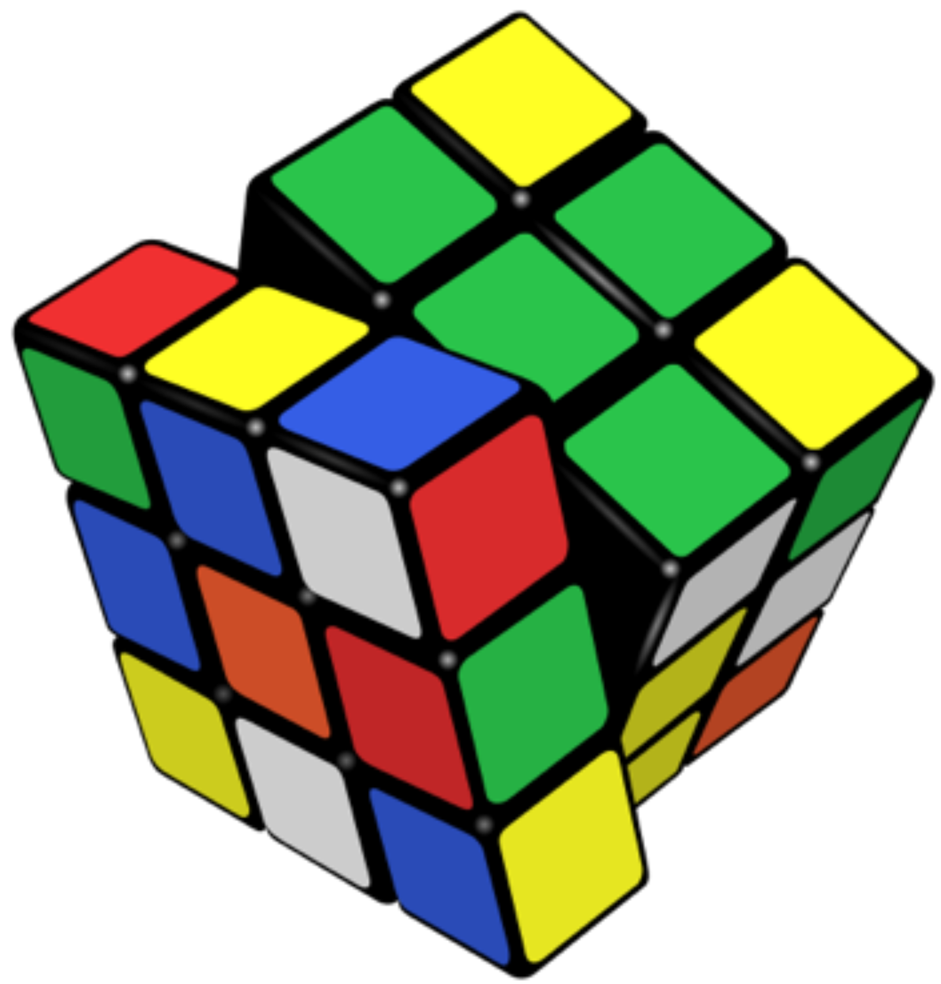


Basic to problem solving:

- *How to take action to reach a goal?*

Informed Search

What if we *know something* about the search?



Formal Definition

Set of states S

Start state $s \in S$

Set of actions A and action rules $a(s) \rightarrow s'$

Goal test $g(s) \rightarrow \{0, 1\}$

Cost function $C(s, a, s') \rightarrow \mathbb{R}^+$

So a search problem is specified by a tuple, (S, s, A, g, C).

Problem Statement

DUKE COMPUTER SCIENCE

Find a sequence of actions $a_1, \ldots, a_n$
and corresponding states $s_1, \ldots, s_n$

… such that:

$$s_0 = s$$ start state

$$s_i = a_i(s_{i-1}), i = 1, \ldots, n$$ legal moves

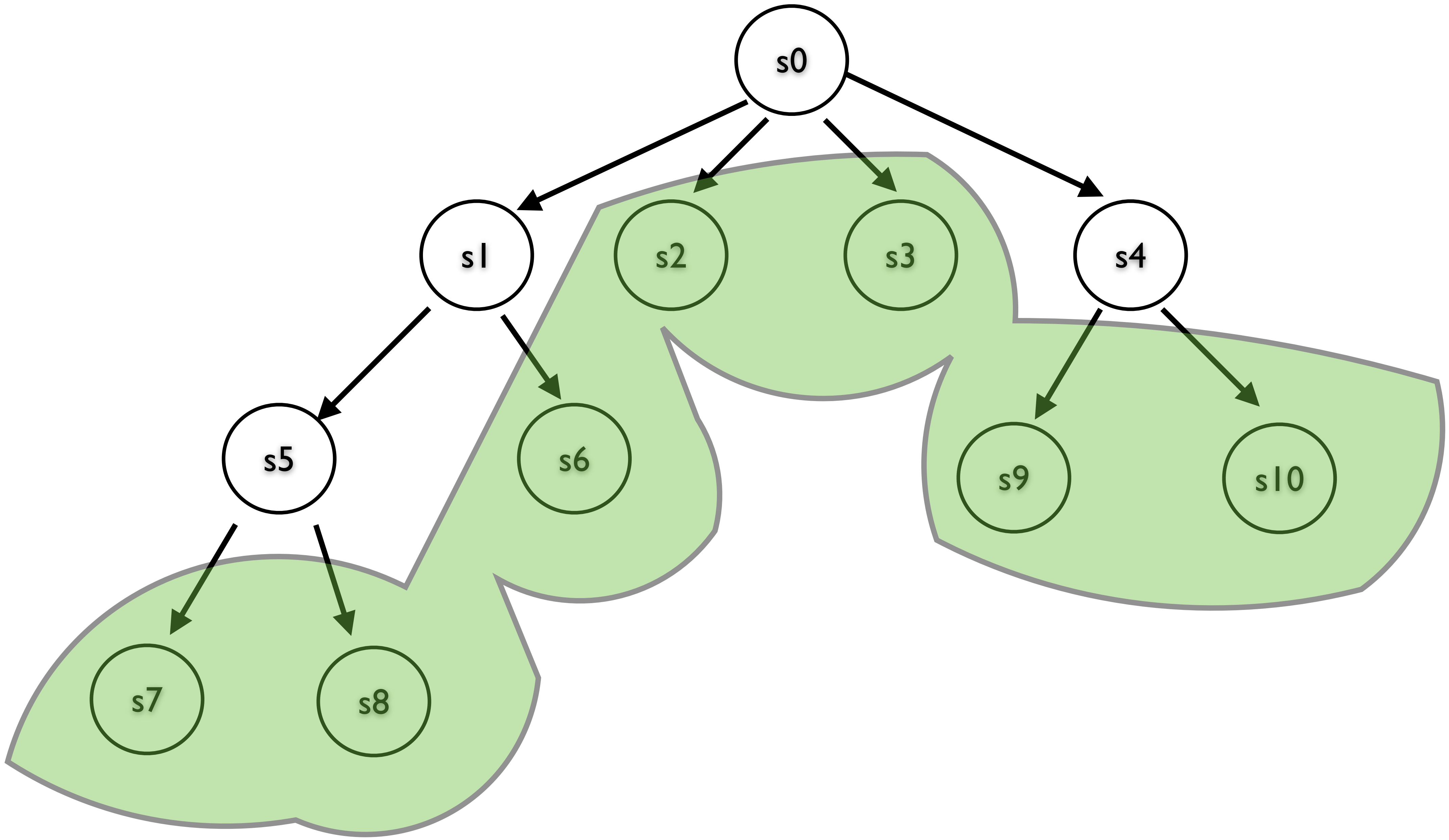
$$g(s_n) = 1$$ end at the goal

while minimizing:

$$\sum_{i=1}^{n} C(s_{i-1}, a, s_i)$$ minimize sum of costs - *rational agent*

The Frontier

Key thing in search is *managing the frontier.*



Uninformed Searches

DUKE COMPUTER SCIENCE

Simple strategy for choosing next node:

- Choose the shallowest one (**breadth-first**)
- Choose the deepest one (**depth-first**)

Neither guaranteed to find the least-cost path.

What if we chose the one with lowest cost?

Uniform-Cost

Order the nodes in the frontier by ***cost-so-far***

- Cost from the start state to that node.

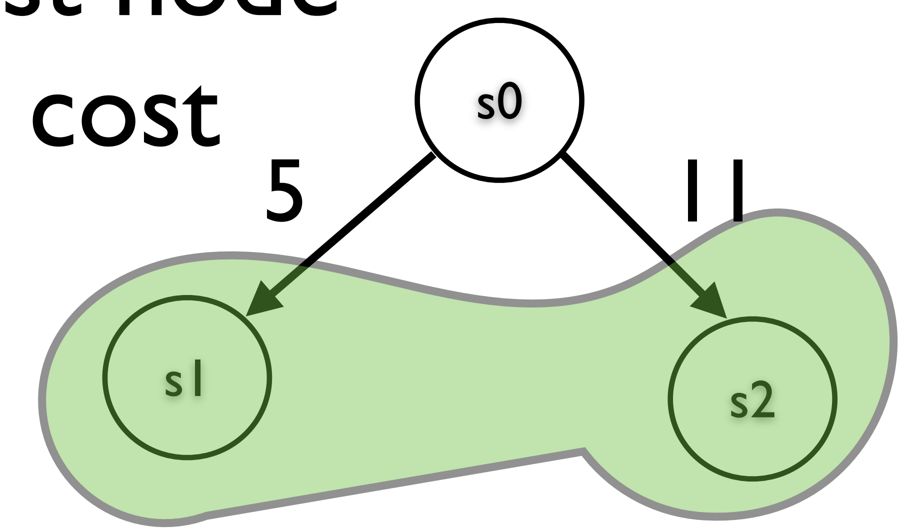
Open the next node with the smallest cost-so-far

- Optimal solution
- Complete (provided no negative costs)

Uniform-Cost

Expand cheapest node

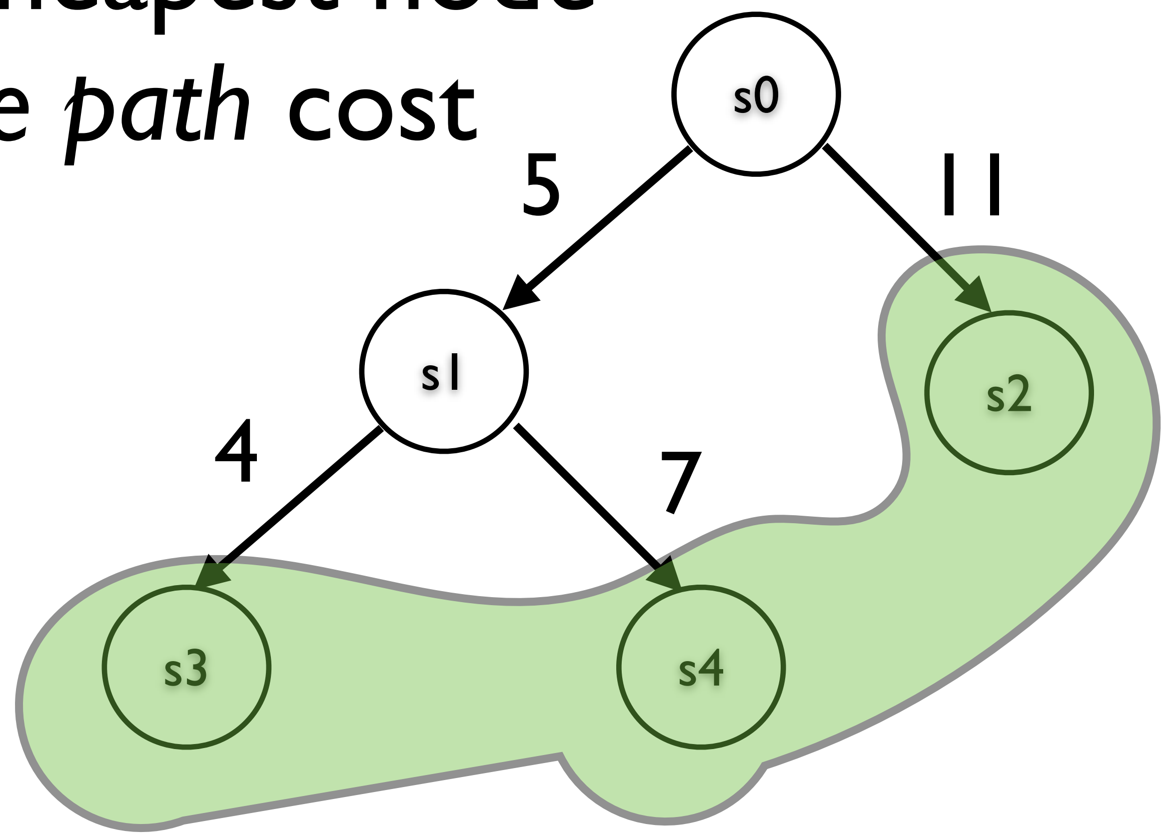
Use *whole path* cost



Uniform-Cost

Expand cheapest node

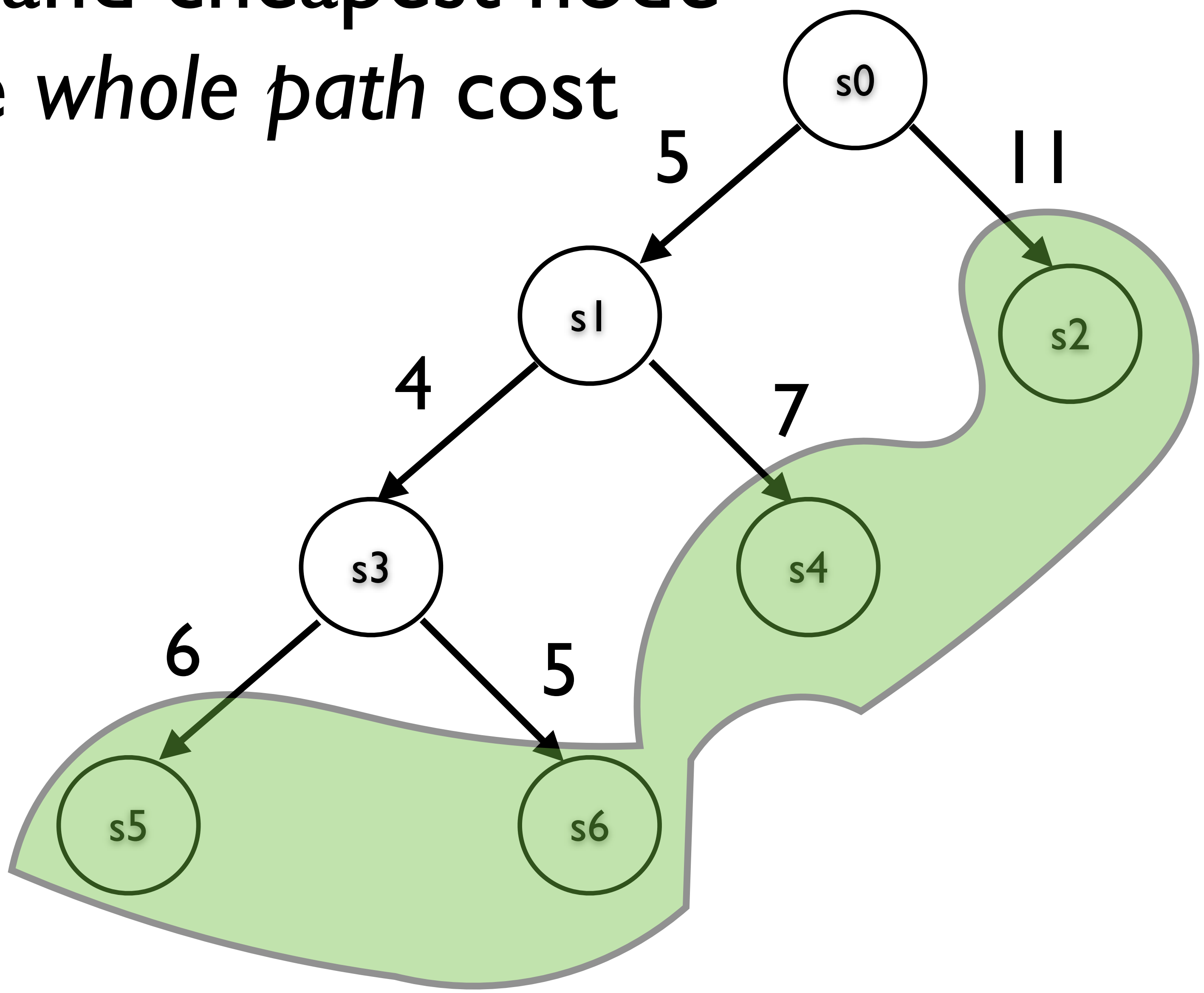
Use *whole path* cost



Uniform-Cost

Expand cheapest node

Use *whole path* cost

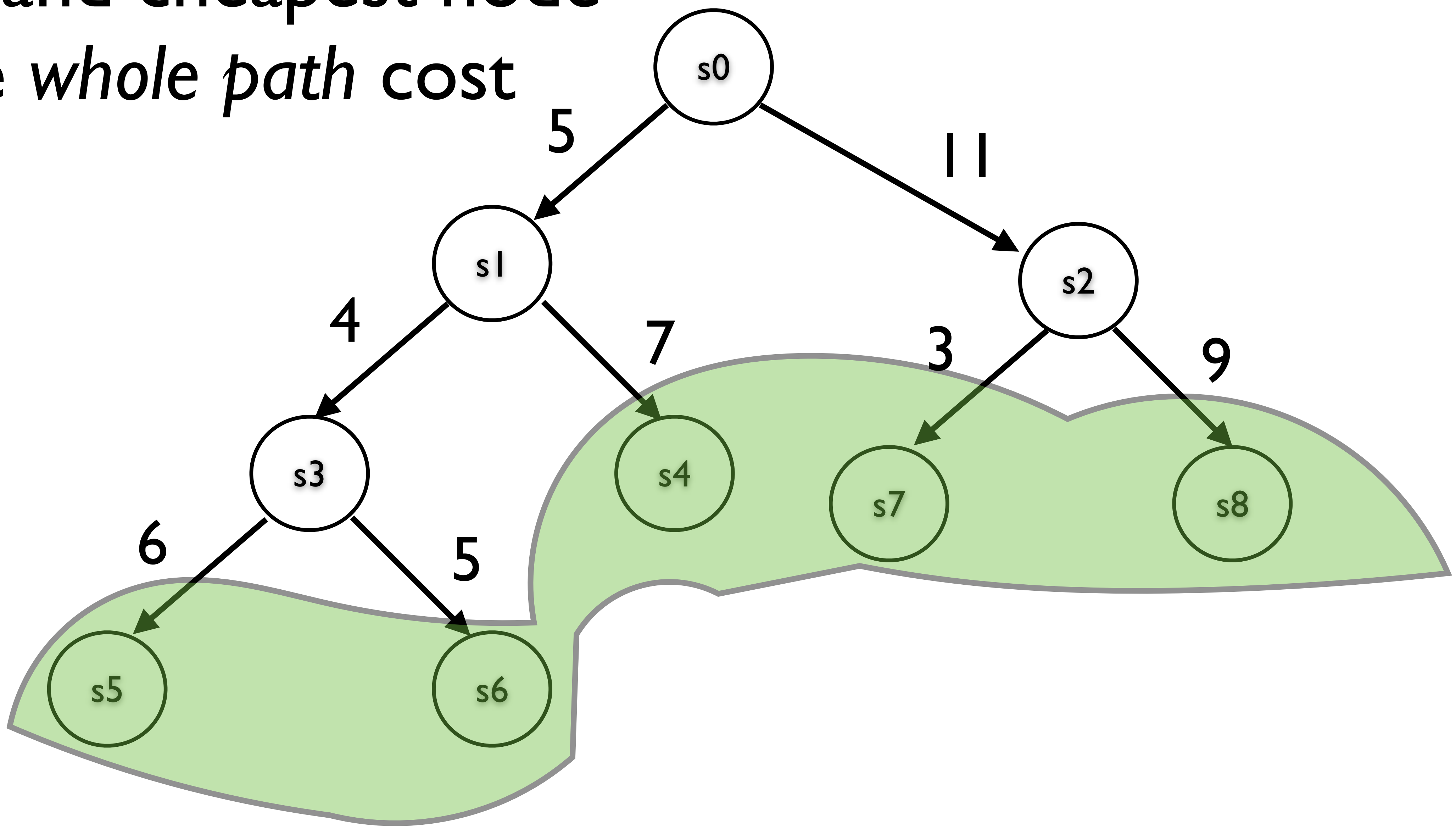


Uniform-Cost

DUKE
COMPUTER
SCIENCE

Expand cheapest node

Use *whole path* cost



What's the Insight?

The *cost-so-far* tells us how much it cost to get to a node.

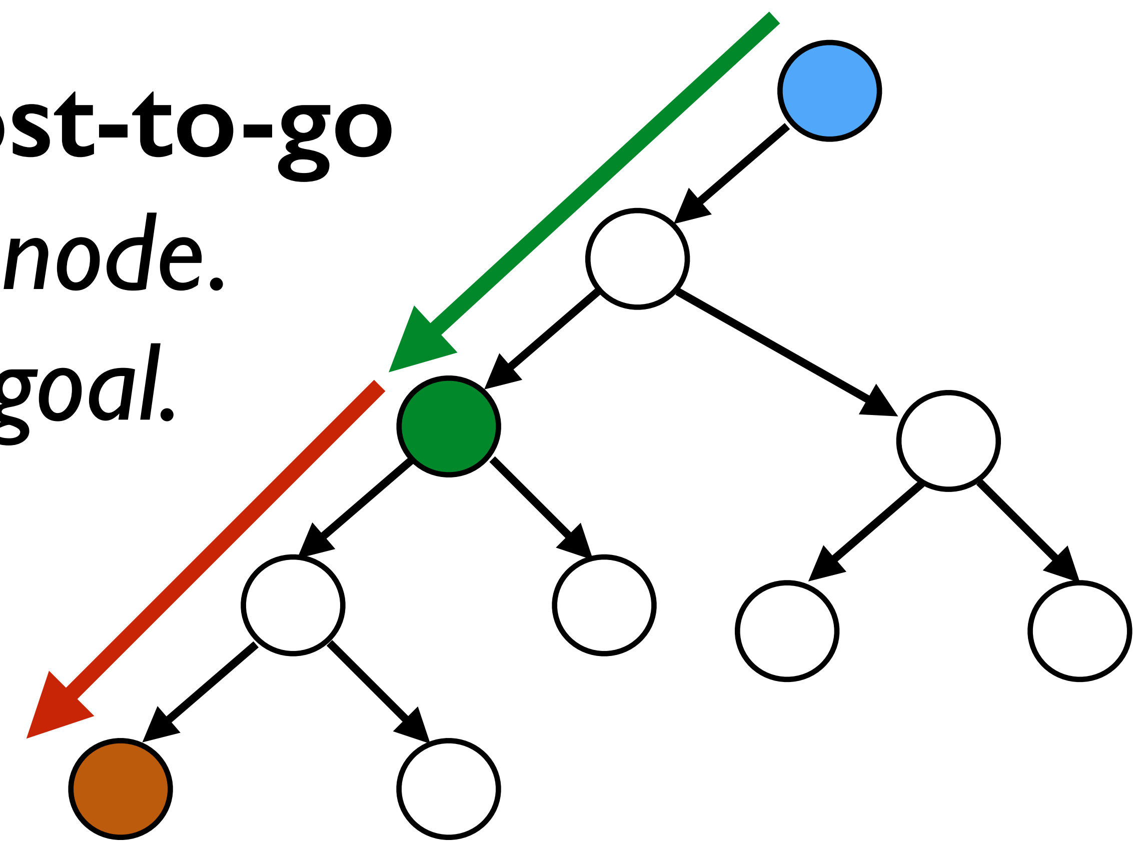
- Go to cheapest nodes first.

What remains?

Total cost = **cost-so-far + cost-to-go**

Cost-so-far: *cost from start to node.*

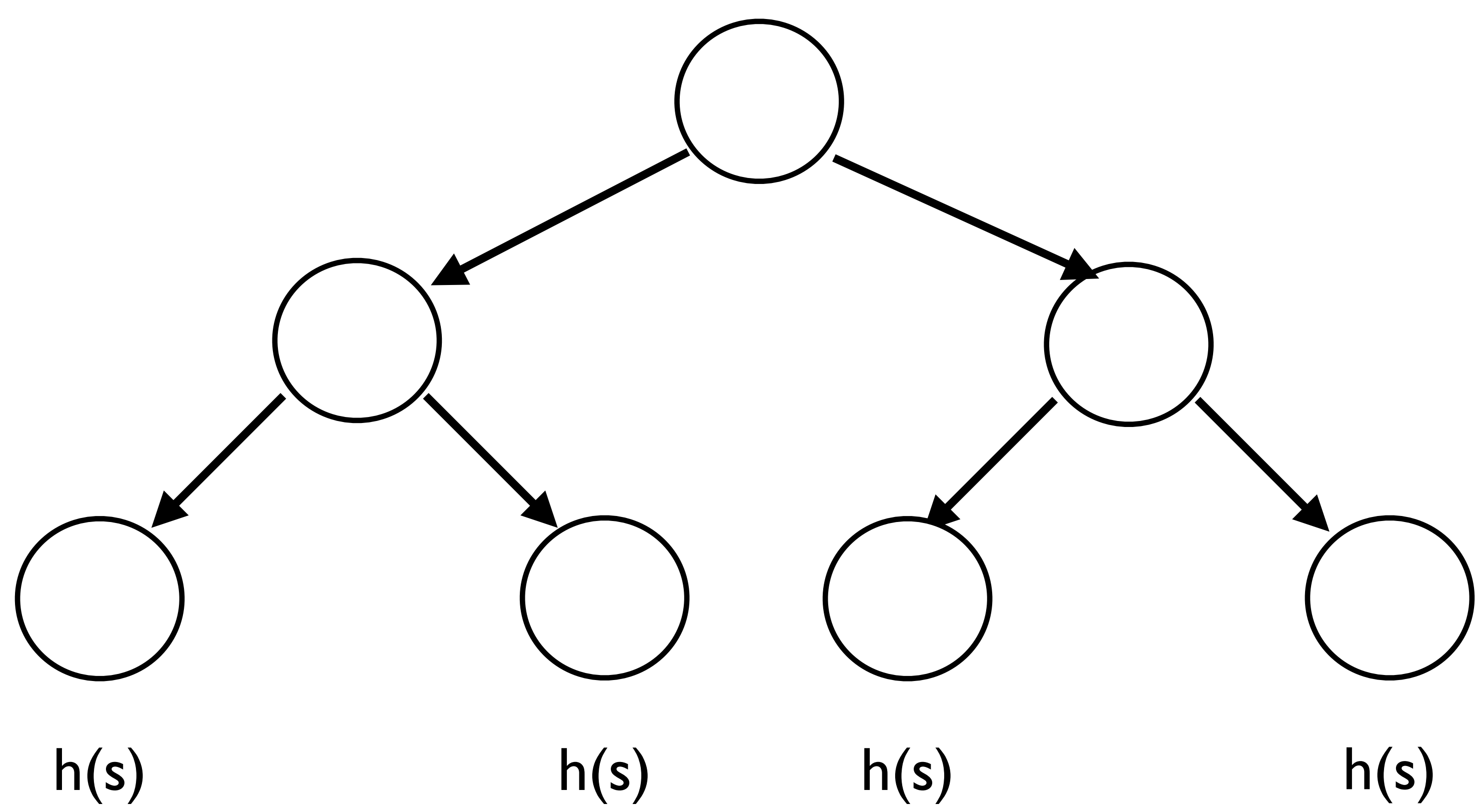
Cost-to-go: *cost from node to goal.*



Informed Search

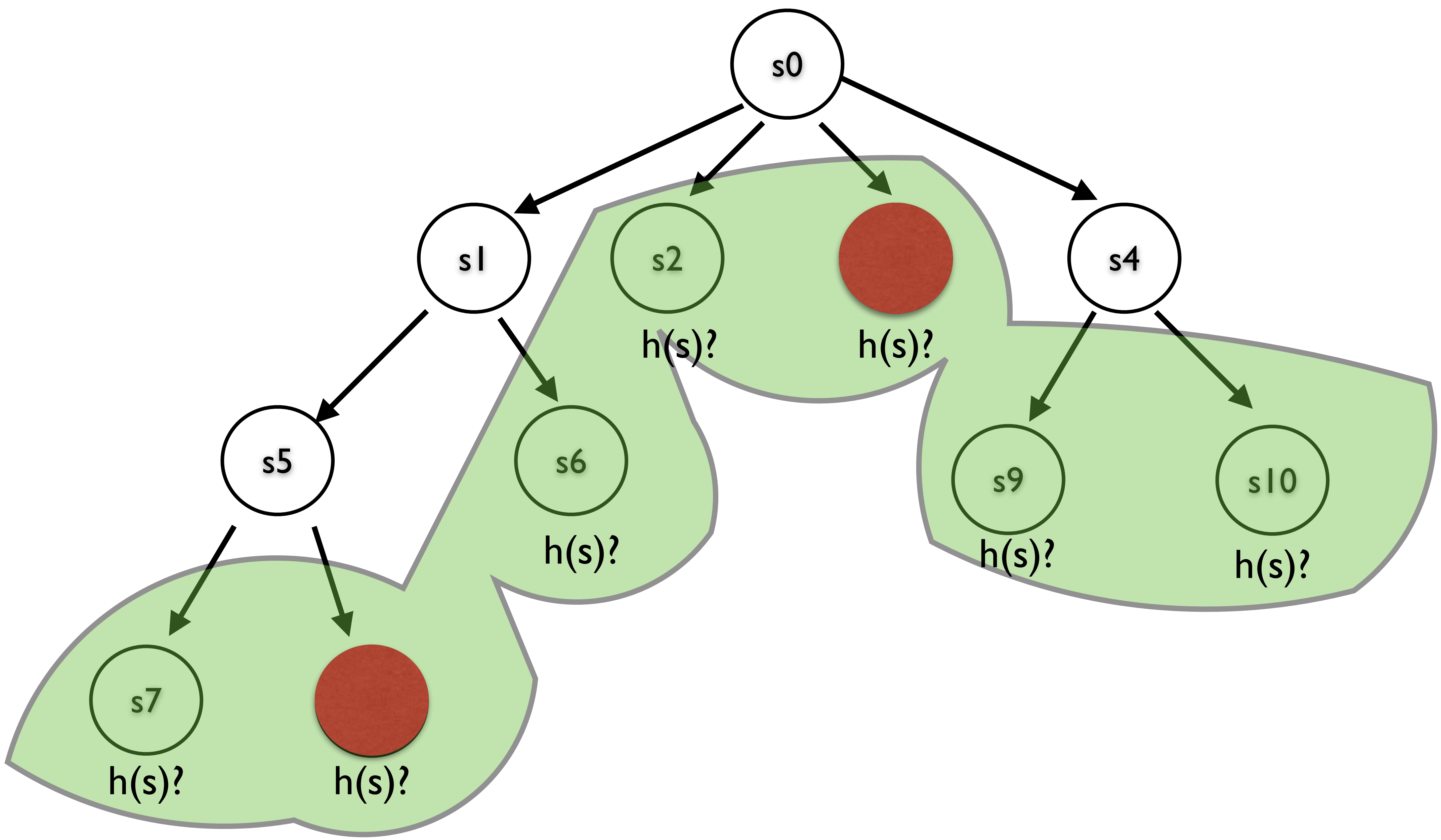
Key idea: *heuristic function*.

- *h(s)* - estimates cost-to-go
- Cost to go *from* state *to* solution.
- **Problem specific (hence *informed*)**



Greed

What if we expand the node with lowest *h(s)*?



A*

A* algorithm:

- *g(s)* - cost *so far* (start to s).
- Expand s that minimizes *g(s)* + *h(s)* both
- Manage frontier as priority queue.

- Admissible heuristic: *never overestimates* cost.

$$h(s) \leq h^*(s)$$

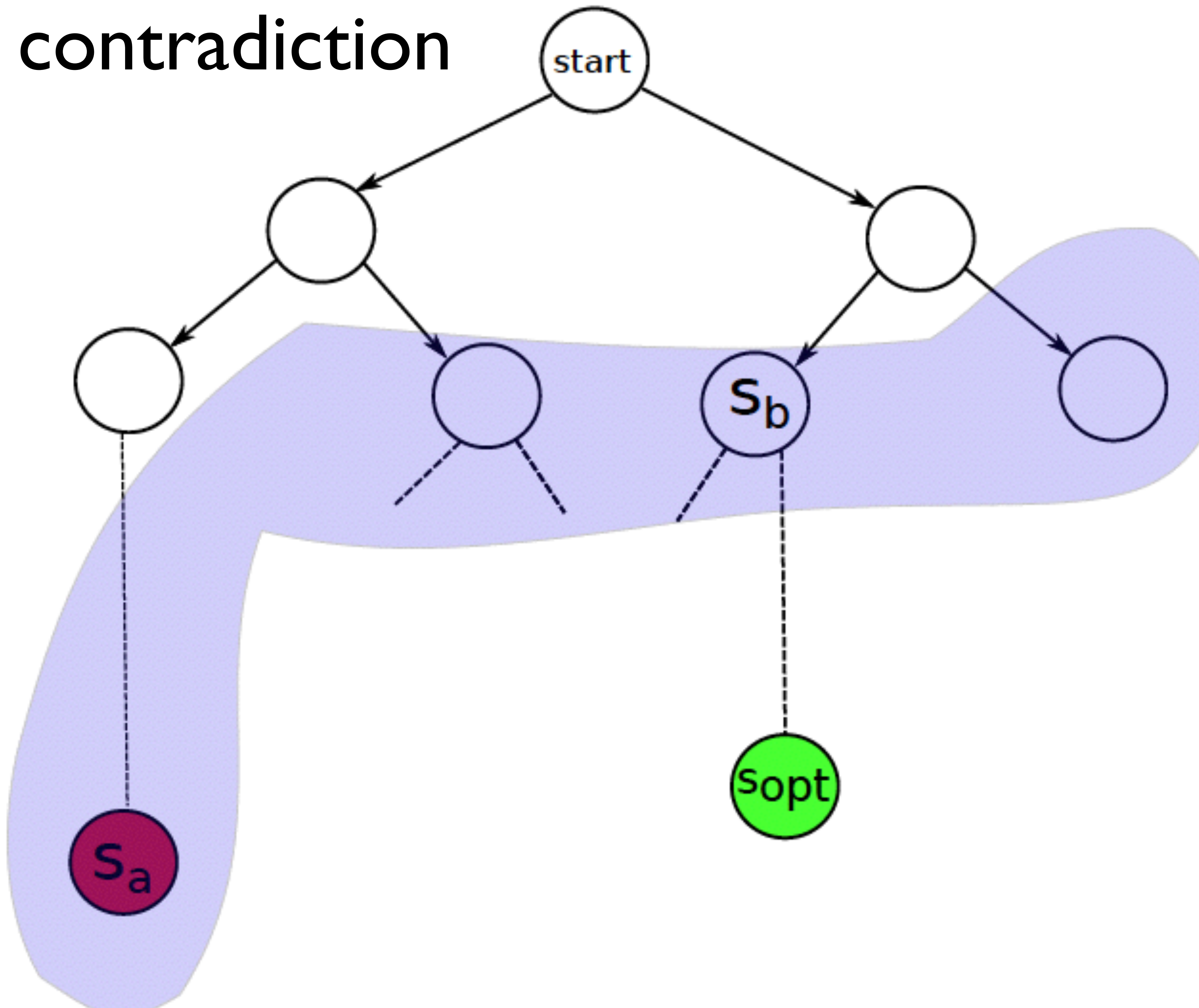
- *h(s)* = *0* if s is a goal state, so *g(s)* + *h(s)* = *c(s)*

- If h is admissible, A* is optimal.
- If h(s) is exact, runs in *O(bd)* time.

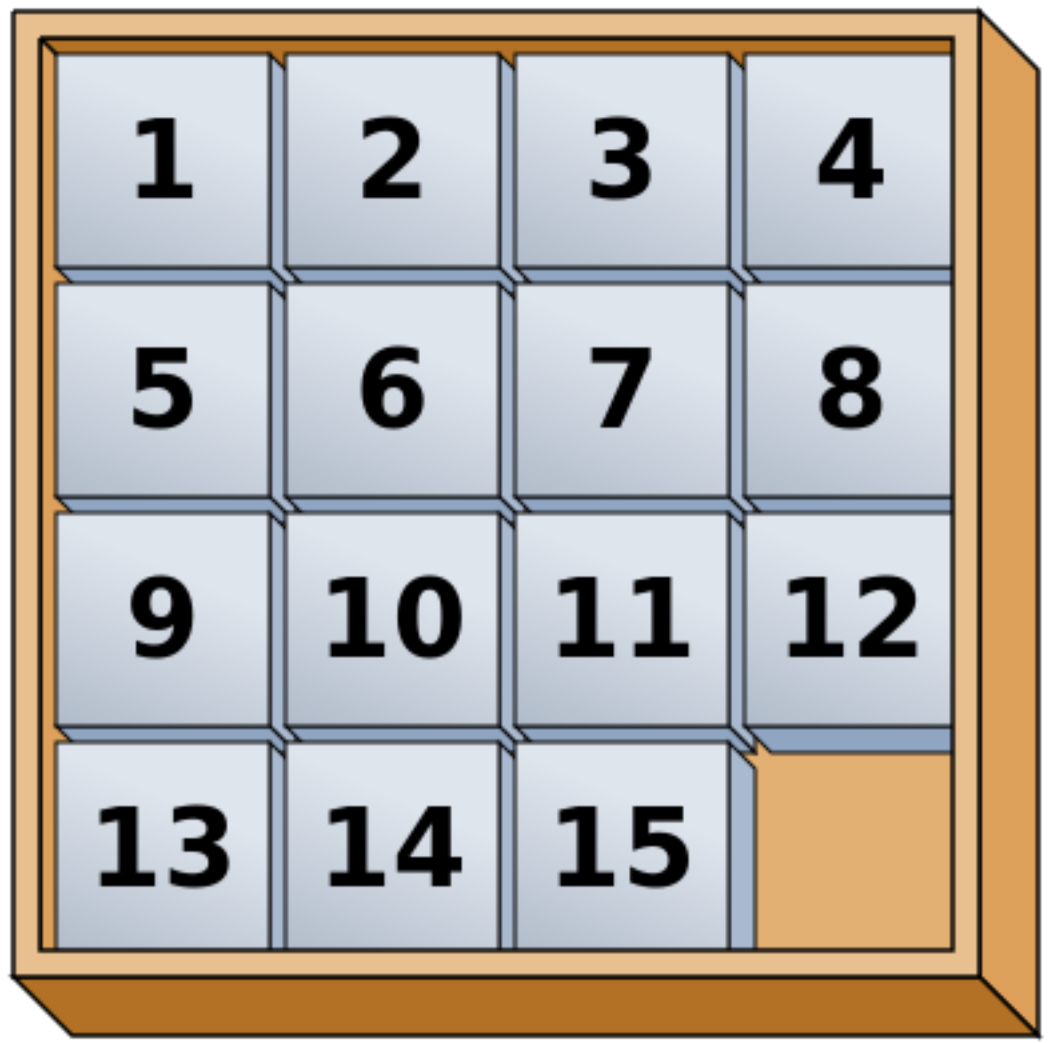
Admissible Heuristics

Optimal solution

Proof by contradiction

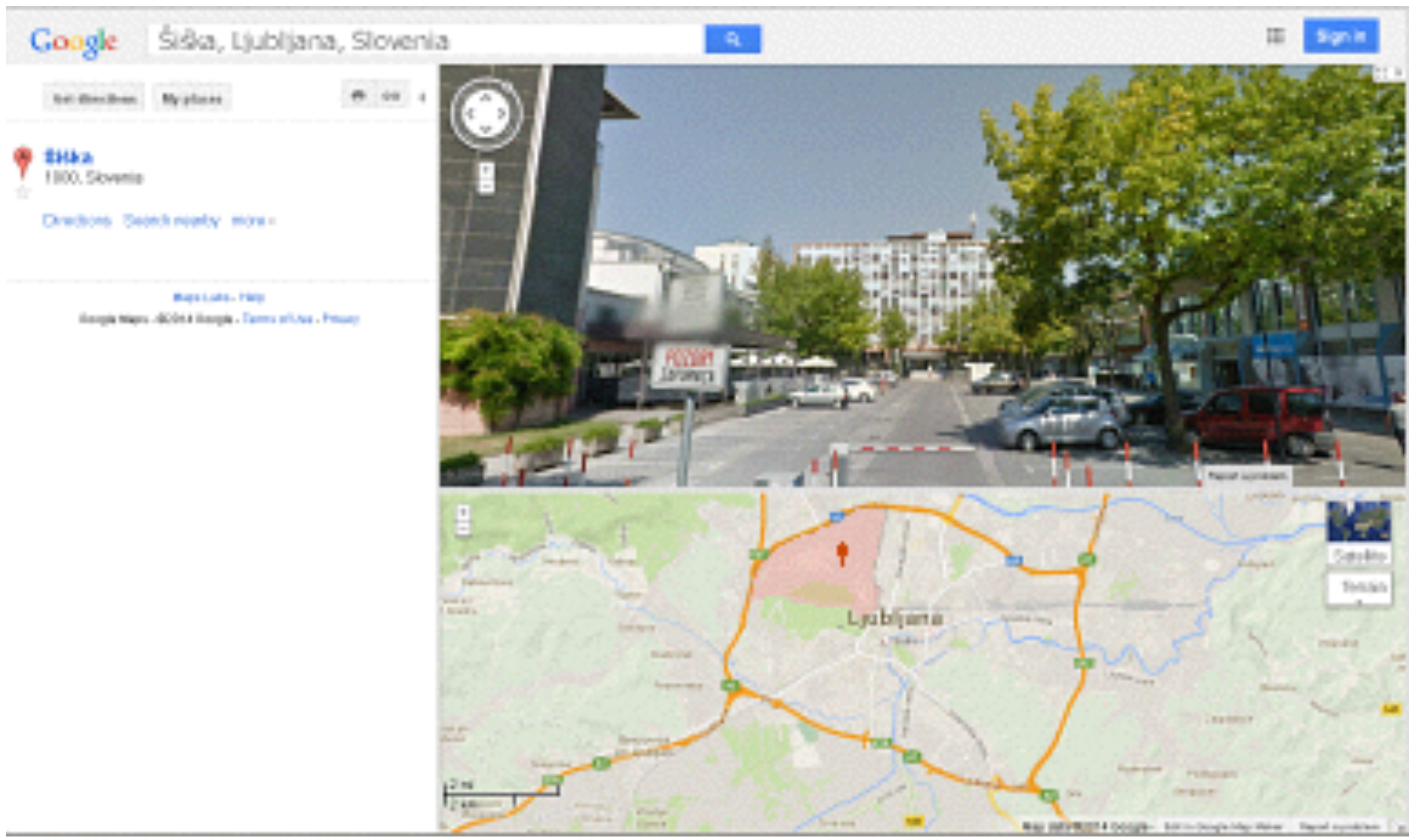


Example Heuristic



Example Heuristics

DUKE
COMPUTER
SCIENCE



More on Heuristics

Heuristic *h1* dominates *h2* if *h1(s)* >= *h2(s)* for all s.

- Is *h1* or *h2* better? (If they're both admissible.)

How might you combine two heuristics?

What is *h(s)* = *k* (constant) for all *s*?

More on Heuristics

A* is optimally efficient: any algorithm using *h* **must** expand the nodes A* expands.

Why?

More on Heuristics

Ideal heuristics:

- Fast to compute.
- Close to real costs.
- Some programs *automatically generate* heuristics.