Agents that Plan Ahead: A* Search

Russell and Norvig: Chapter 3.1-3.4, 3.5-3.6

CSE 240: Winter 2023

Lecture 4

Guest Lecture: Prof. Marinescu

Announcements

- Assignment 1 is up
- Prof. Marinescu lecturing today.
- Quizzes will be all remote on Canvas.
- Prof. Gilpin will update the class on Tuesday.

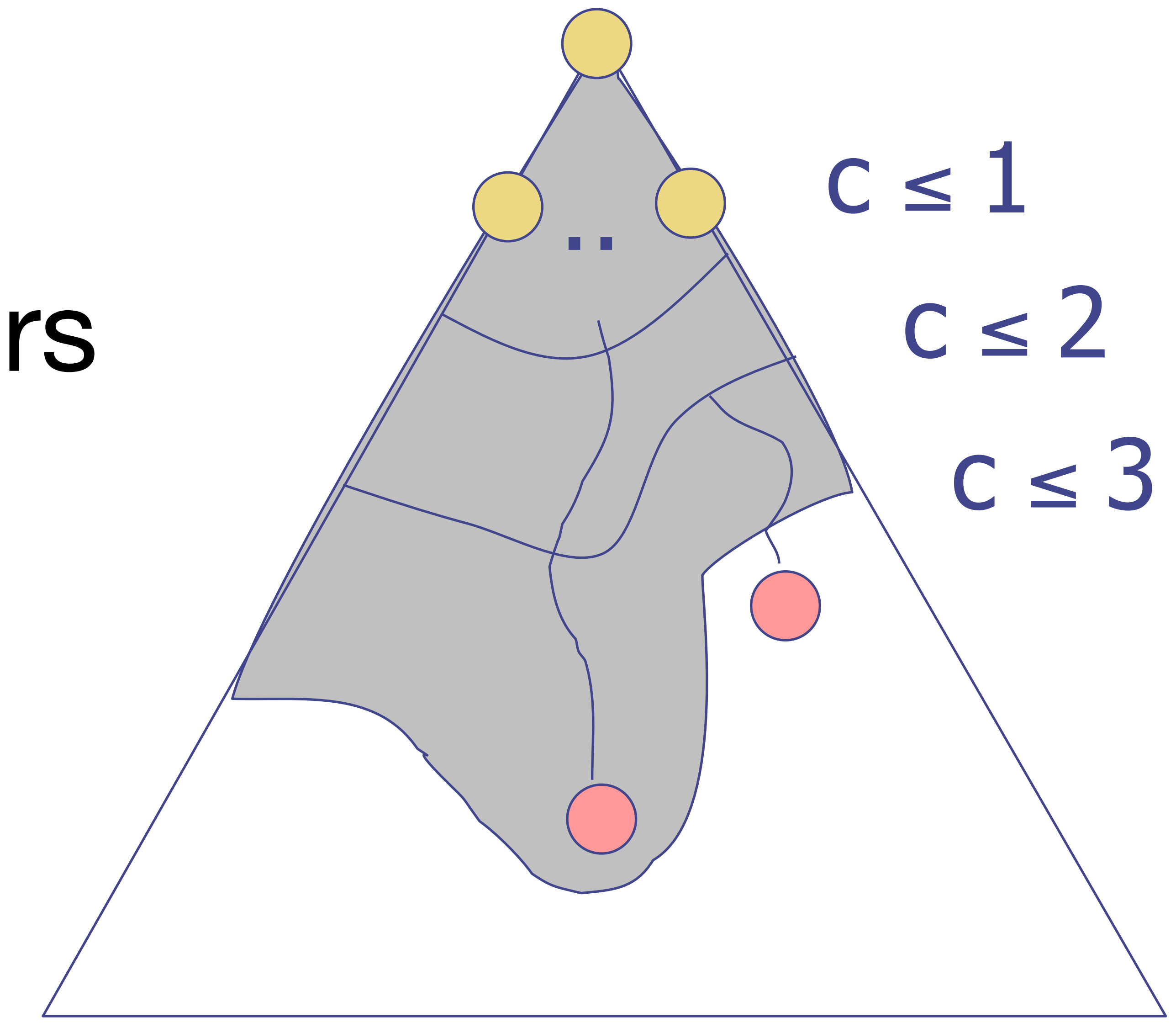
Agenda

Today

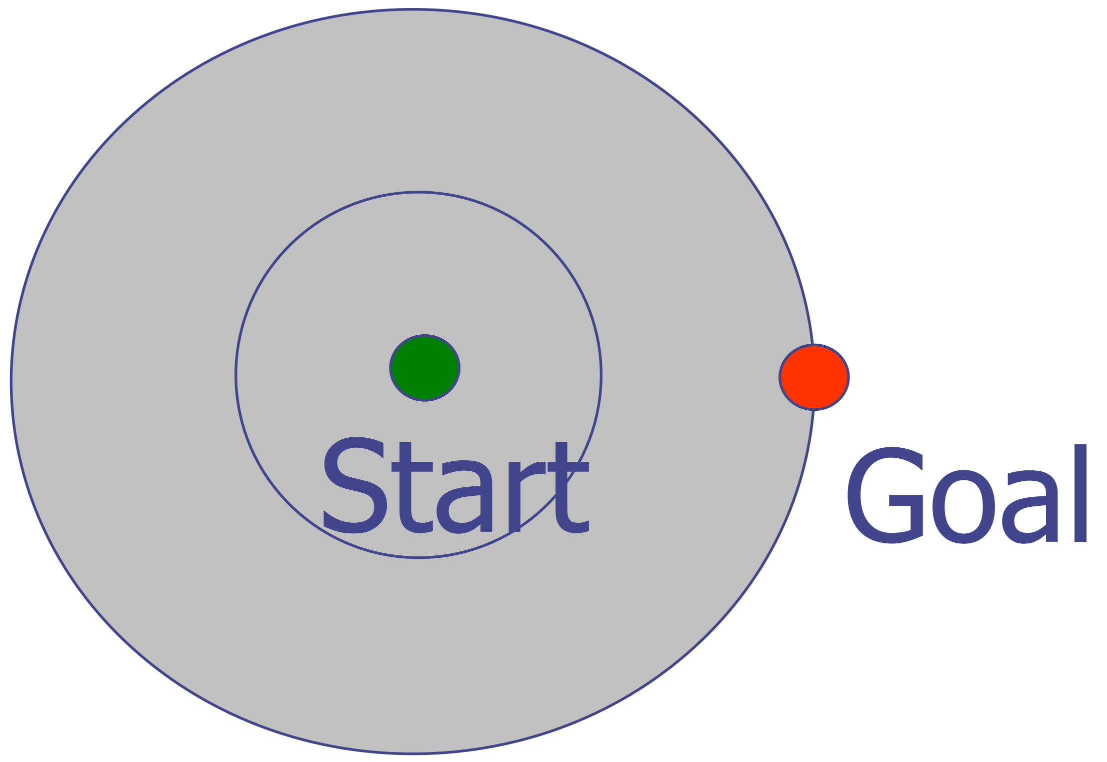
- Informed search strategies
  - A* search algorithm
  - Heuristics

Recap: Uniform Cost Issues

- Remember: explores increasing cost contours



- The good: UCS is complete and optimal!
- The bad:
  - Explores options in every “direction”
  - No information about goal location

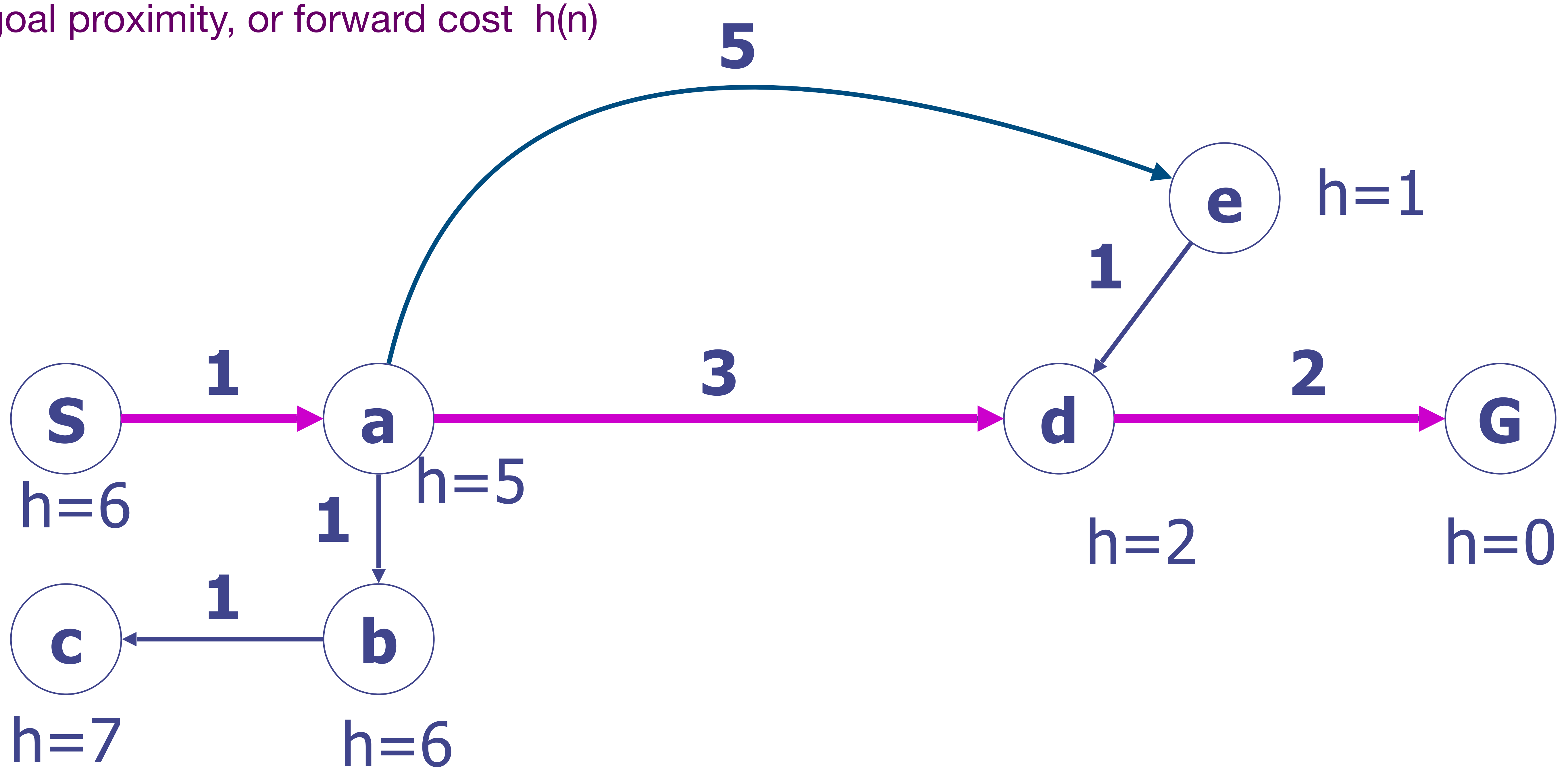


A* Search

Combining UCS and Greedy

$$f(n) = g(n) + h(n)$$

- Uniform-cost orders by path cost, or backward cost $g(n)$
- Greedy orders by goal proximity, or forward cost $h(n)$



- A* Search orders by the sum: $f(n) = g(n) + h(n)$

Node	Fringe	f(n)
s	s->a	**6**
s->a	s->a->b	**8**
s->a	s->a->d	**6**
s->a	s->a->e	**7**

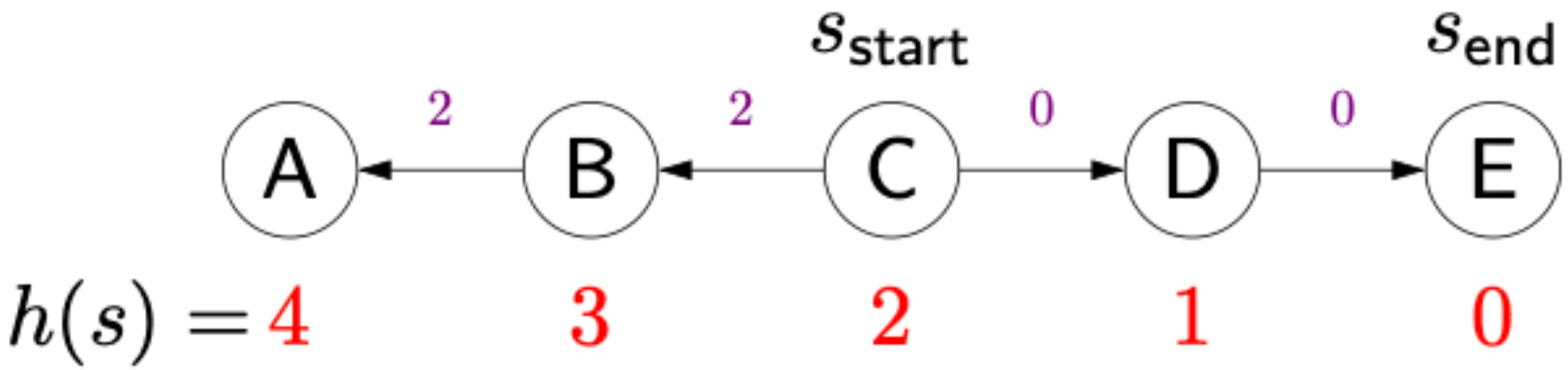
Example: Teg Grenager

Another Way to Implement A*

Run UCS with modified edge costs in order to account for closeness to the goal state

$$Cost'(s,a) = Cost(s,a) + h(succ(s,a)) - h(s)$$

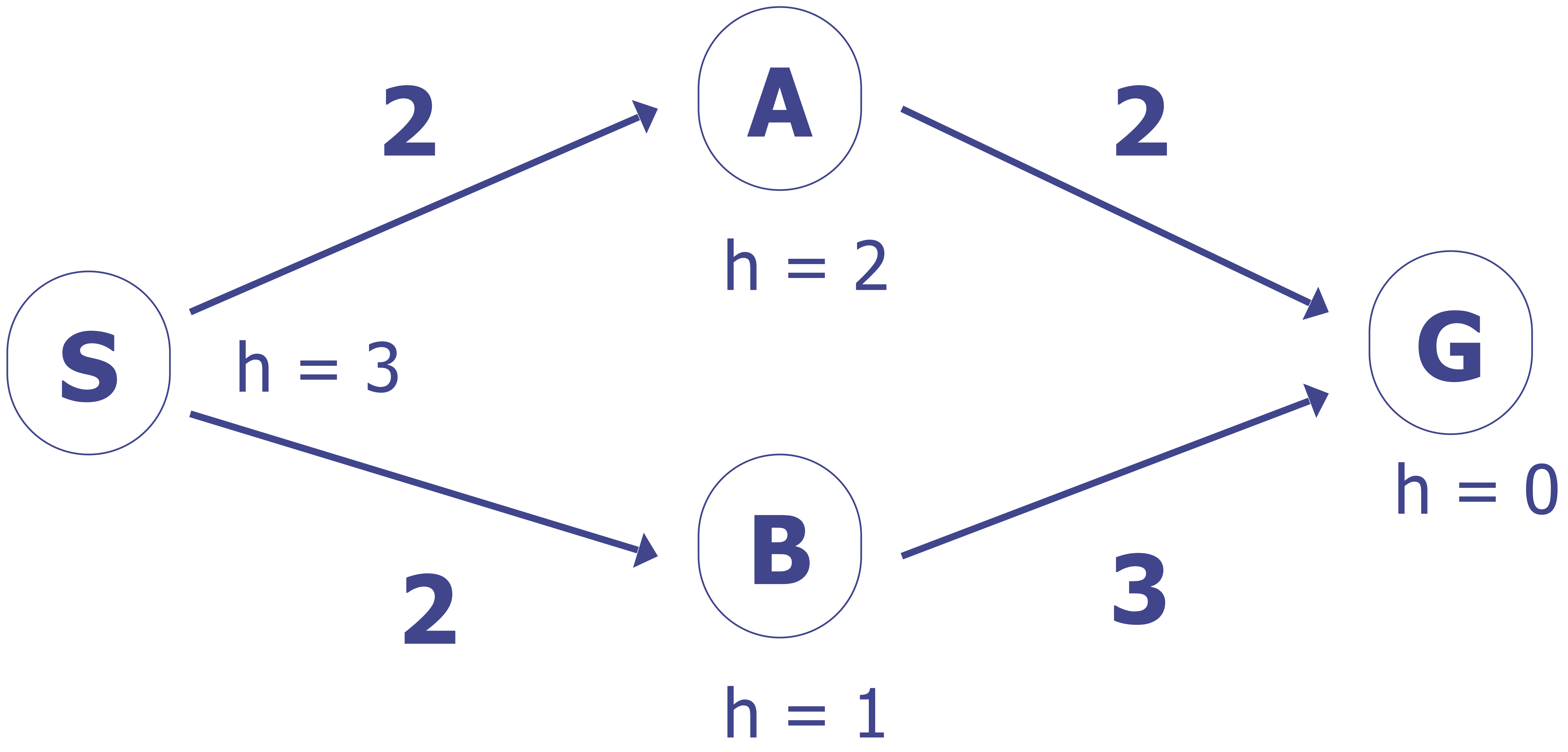
- Intuition: add a penalty for how much action ‘a’ takes us away from the end state



$$\text{Cost}'(C,B) = \text{Cost}(C,B) + h(B) - h(C) = 1 + (3-2) = 2$$

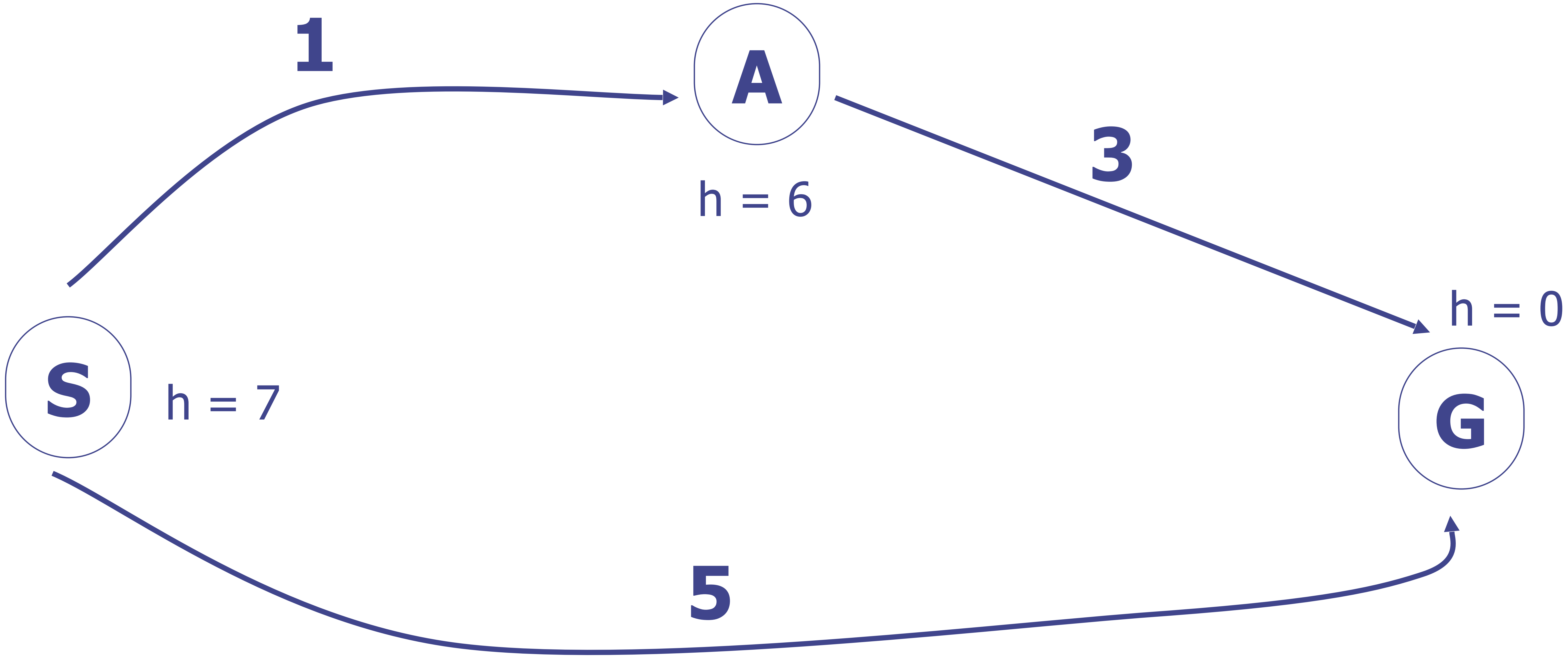
When should A* terminate?

- Should we stop when we enqueue a goal?



- No: only stop when we dequeue a goal

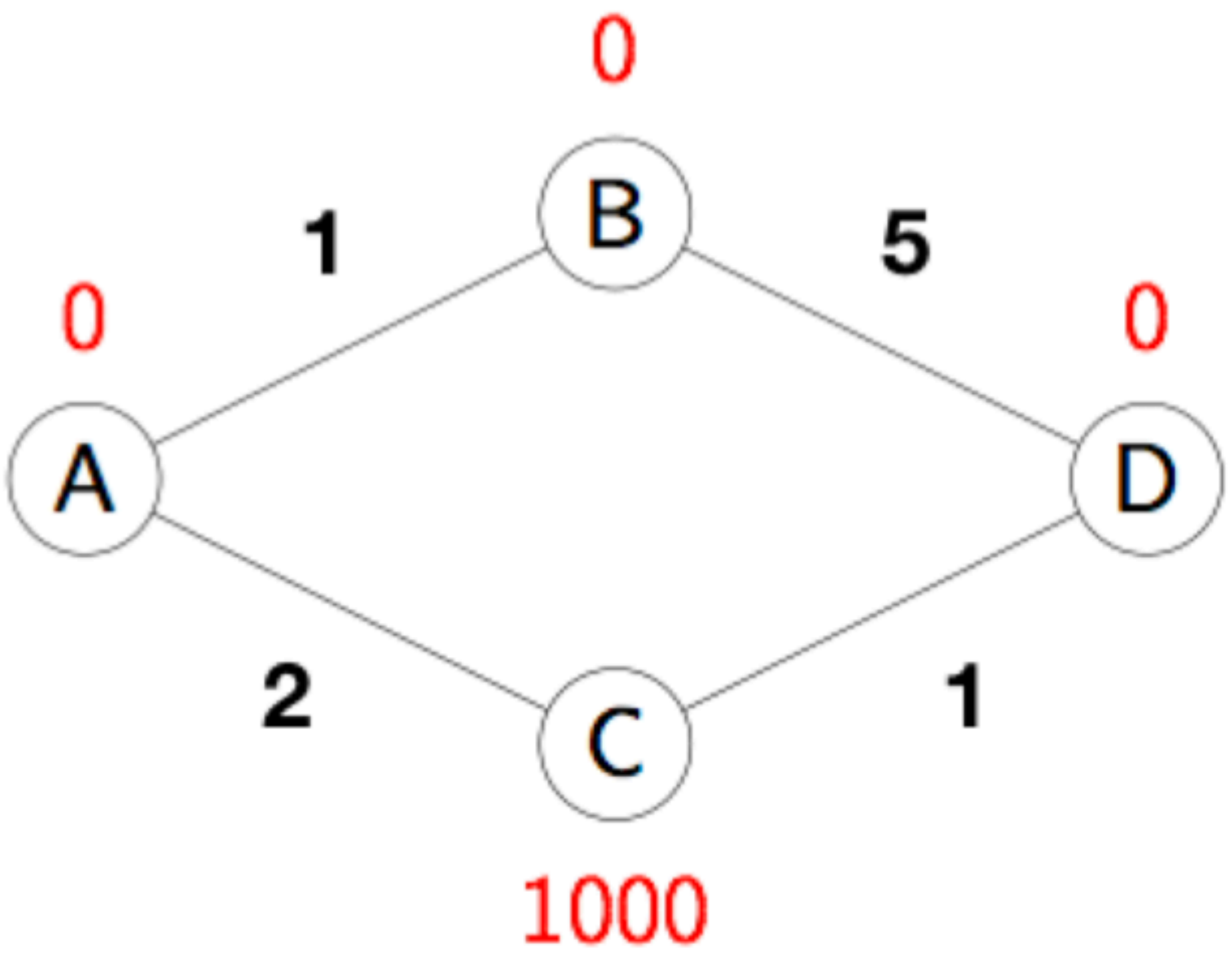
Is A* Optimal?



- What went wrong?
- Actual bad goal cost < estimated good goal cost
- We need estimates to be less than actual costs!

An Example Heuristic

- Would any heuristic work?
- Doesn't work because of the negative modified edge costs (or being pessimistic about the correct path)



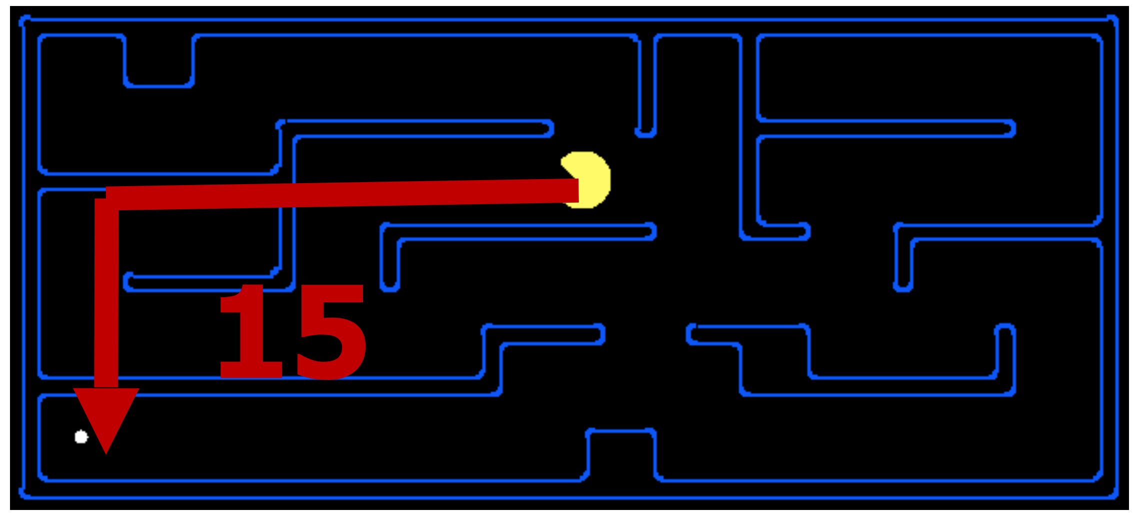
Admissible Heuristics

- A heuristic h is admissible (optimistic) if:

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

- where $h^*(n)$ is the true cost to a nearest goal

- Example:

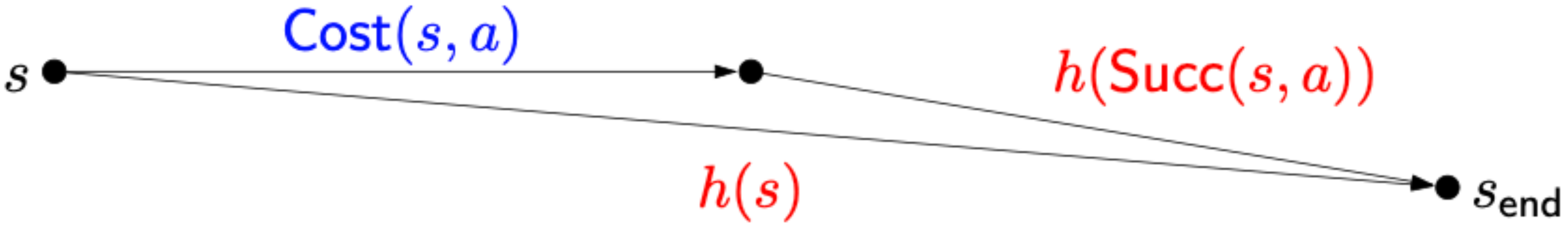


- Coming up with admissible heuristics is most of what's involved in using A* in practice.

Consistent Heuristic

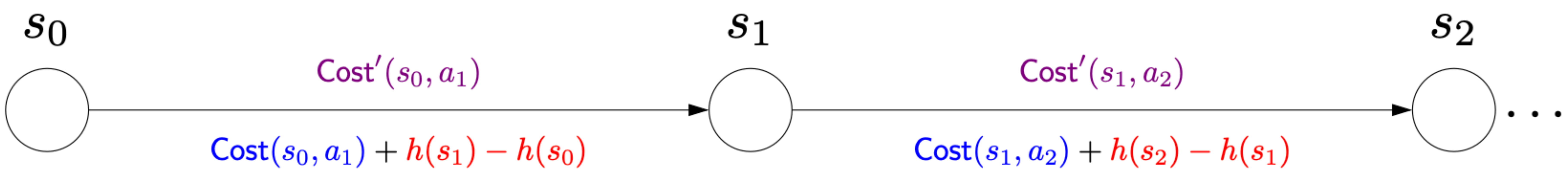
A heuristic h is "**consistent**" if

- $\boldsymbol{Cost'(s,a) = Cost(s,a) + h(succ(s,a)) - h(s) \geq 0}$
- $\boldsymbol{h(s_{end}) = 0}$



Correctness of A*

- If h is consistent, A* returns the minimum cost path.
- Consider any path
- Key identity:



$$\underbrace{\sum_{i=1}^{L} \text{Cost}'(s_{i-1}, a_i)}_{\text{modified path cost}} = \underbrace{\sum_{i=1}^{L} \text{Cost}(s_{i-1}, a_i)}_{\text{original path cost}} + \underbrace{h(s_L) - h(s_0)}_{\text{constant}}$$

- Therefore, A* solves the original problem using UCS, and therefore the algorithm is complete.

Efficiency of A*

A* explores all states satisfying

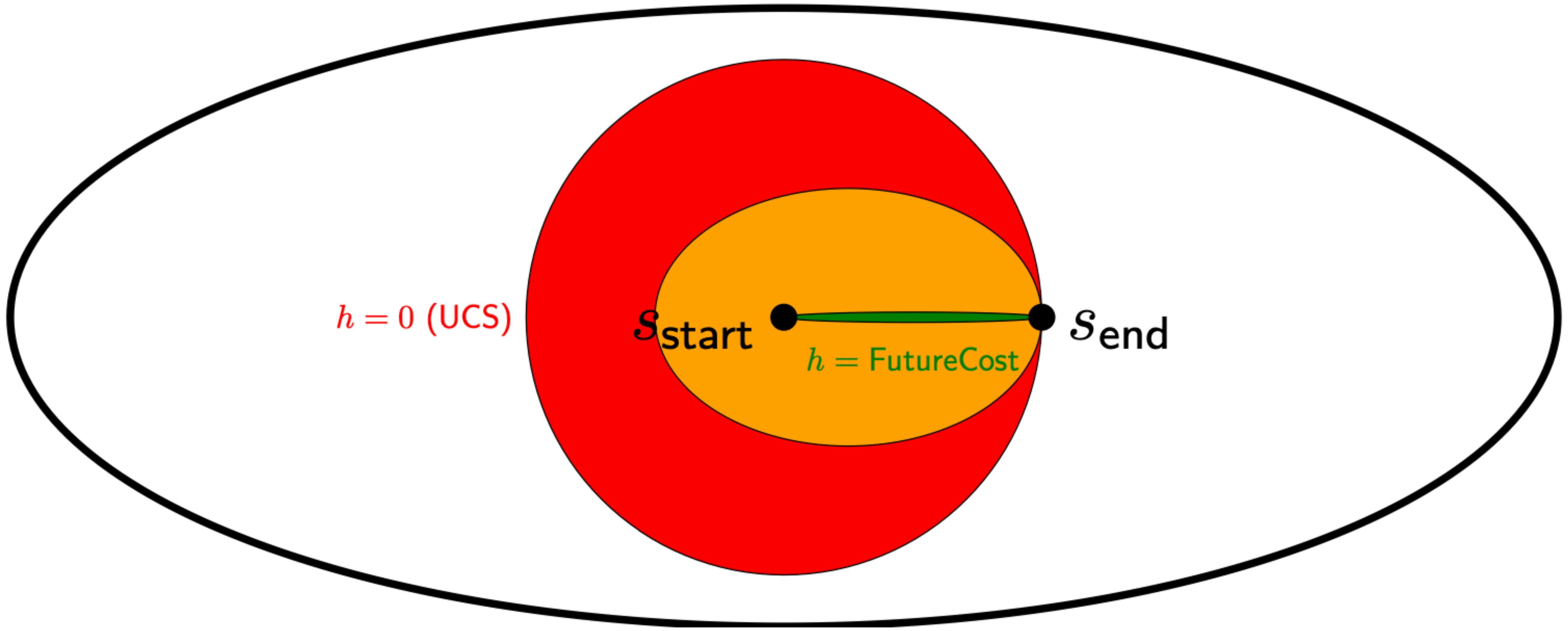
$$f(s) \leq f(s_{end}) - h(s)$$

- Interpretation: the larger h(s), the better
- Proof: A* explores all nodes ‘s’ such that:

$$f(s) + h(s) \leq f(s_{end}) + h(s_{end})$$
$$f(s) + h(s) \leq f(s_{end})$$
$$f(s) \leq f(s_{end}) - h(s)$$

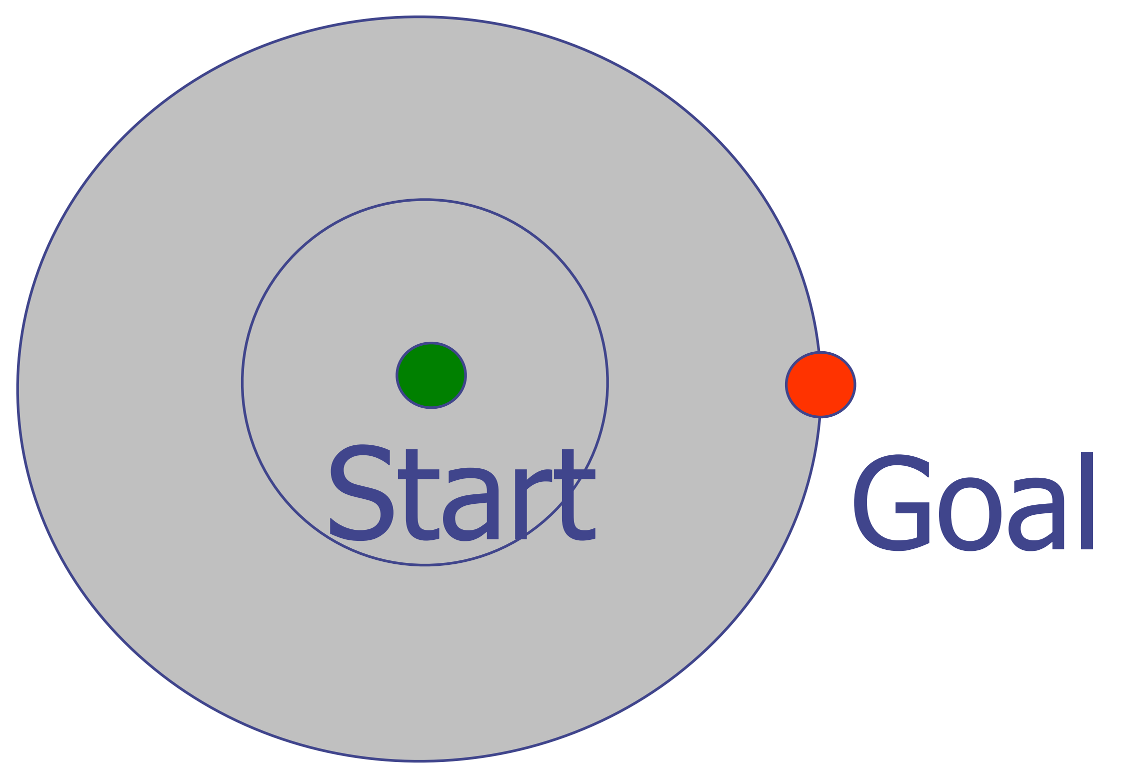
Amount Explored

- If h(s)=0, then A*is the same as UCS.
- If h(s) = FutureCost(s), then A* only explores nodes on a minimum cost path.
- Usually h(s) is somewhere in between.

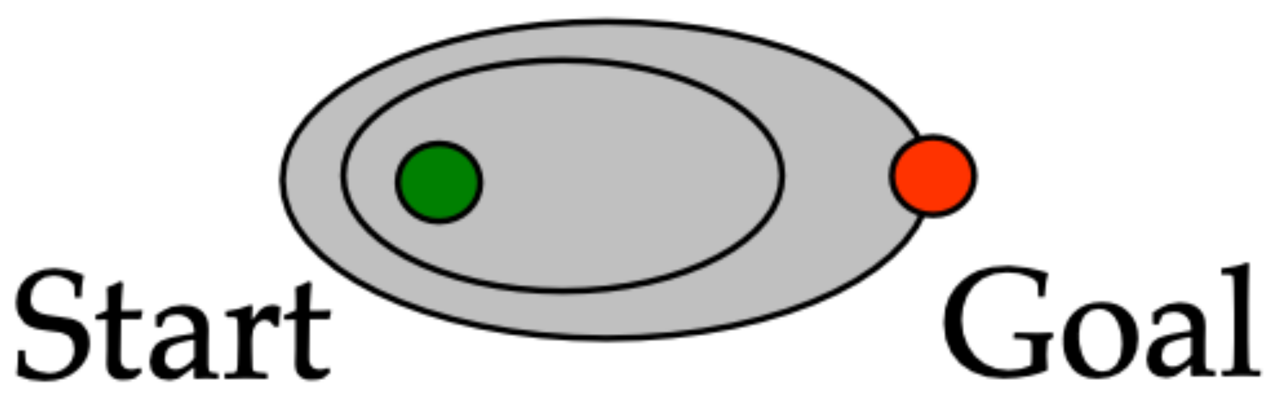


UCS versus A* Contours

- Uniform-cost expands equally in all "directions"



- A* expands mainly toward the goal, but does hedge its bets to ensure optimality



How Do we Get Good Heuristics?

Just Relax!

Relaxation

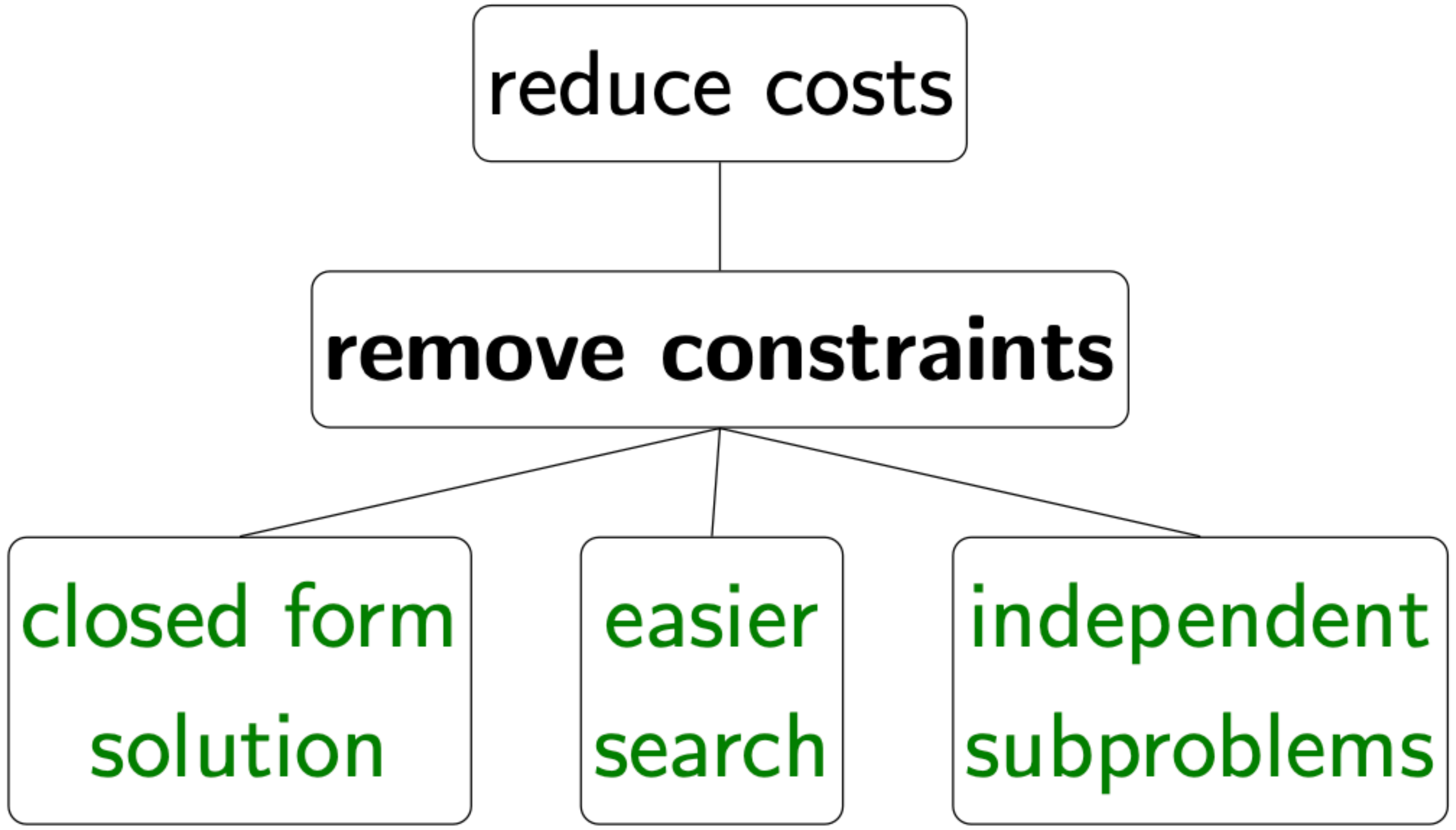
- Ideally, we use h(s) = FutureCost(s), but that's as hard as solving the original problem.

Key idea: relaxation

Constraints make life hard. Get rid of them.

But this is just for the heuristic!

Relaxation Overview



combine heuristics using max

Closed Form Solution

Example: knock down walls

Goal: move from triangle to circle

Hard   Easy

Heuristic:

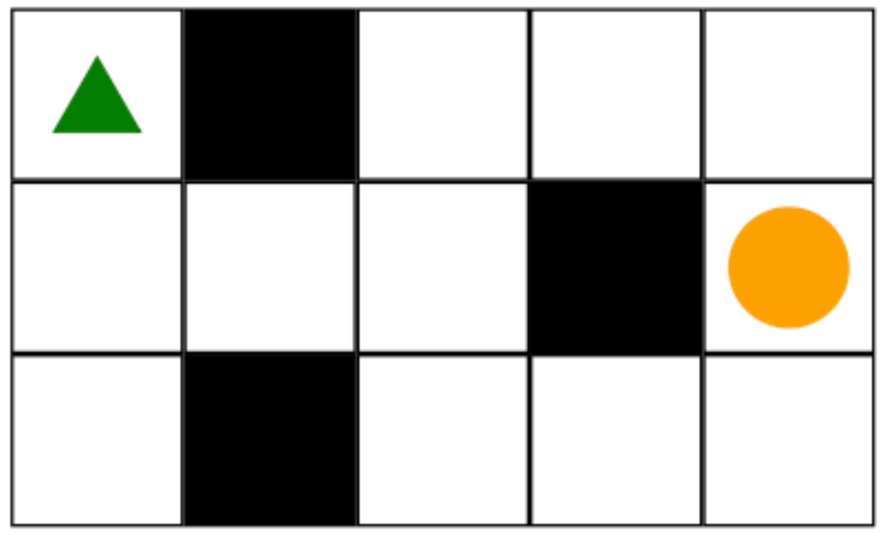
$$h(s) = \text{ManhattanDistance}(s, (2, 5))$$

e.g., $h((1,1)) = 5$

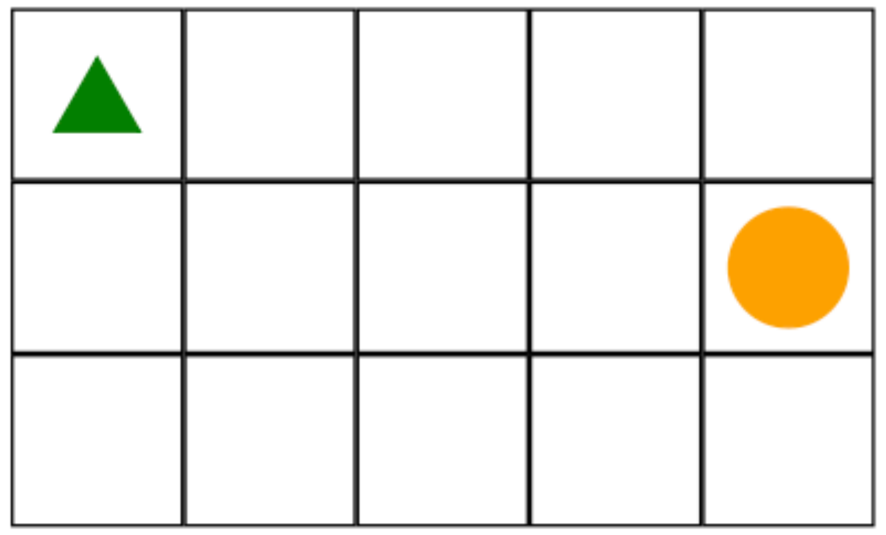
CE 4: What is a Relaxation of this Problem?

Example: knock down walls

Goal: move from triangle to circle



Hard



Easy

Heuristic:

$$h(s) = \text{ManhattanDistance}(s, (2, 5))$$

e.g., $h((1, 1)) = 5$

Easier Search

Example: original problem

Start state: 1

Walk action: from s to $s+1$ (cost: 1)

Tram action: from s to $2s$ (cost: 2)

End state: n

Constraint: can't have more tram actions than walk actions.

State: (location, #walk - #tram)

Number of states goes from $O(n)$ to $O(n^2)$!

Easier Search

Example: relaxed problem

Start state: 1

Walk action: from s to $s+1$ (cost: 1)

Tram action: from s to $2s$ (cost: 2)

End state: n

~~Constraint: can't have more tram actions than walk actions.~~

Original state: (location, #walk - #tram)

Relaxed state: location

Easier Search

- Compute relaxed FutureCost_{rel}(location) for each location (1, . . . , n) using dynamic programming or UCS
- Modify UCS to compute all past costs in reversed relaxed problem (equivalent to future costs in relaxed problem!)

Example: reversed relaxed problem

Start state: n

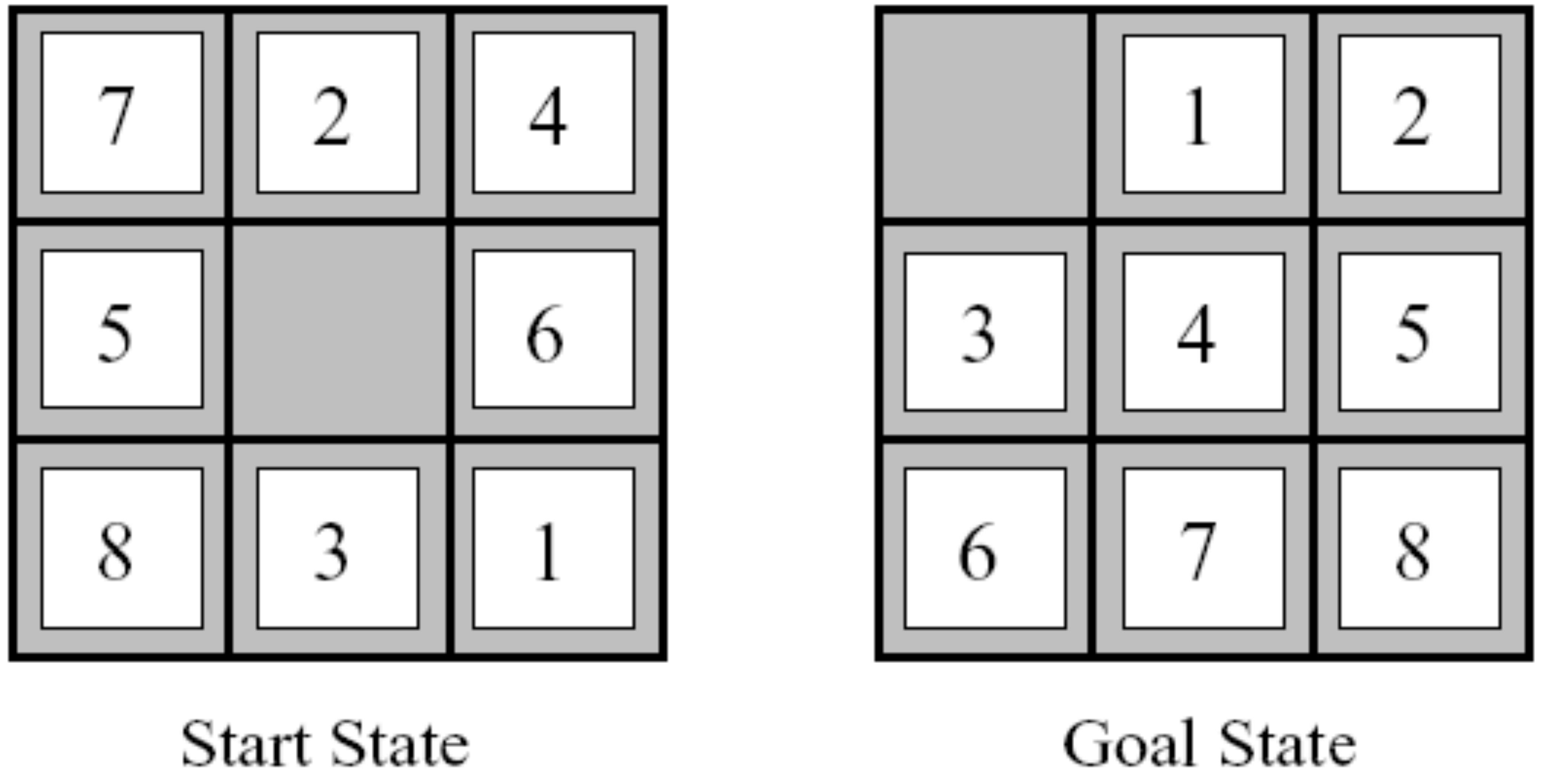
Walk action: from s to $s - 1$ (cost: 1)

Tram action: from s to $s/2$ (cost: 2)

End state: 1

- Define heuristic for original problem:h(location, #walk-#tram) = FutureCost_{rel}(location)

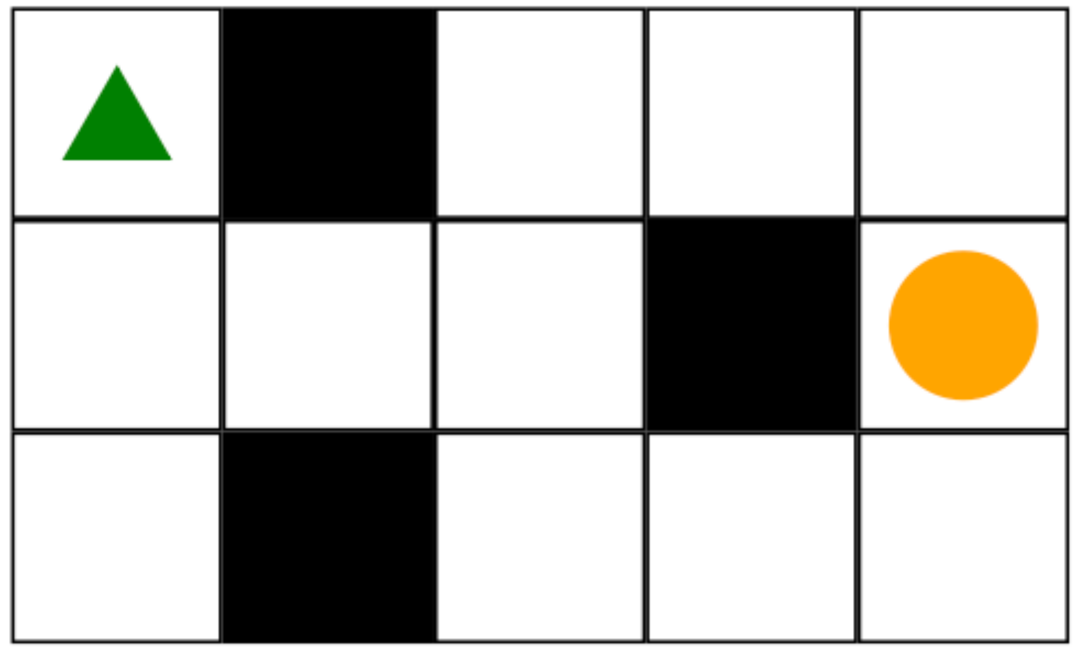
Independent Subproblems



- Original problem: tiles cannot overlap (constraint)
- Relaxed problem: tiles can overlap (no constraint)
- Relaxed solution: 8 indep. problems, each in closed form

General Framework

- Removing constraints (knock down walls, walk/tram freely, overlap pieces)
- Reducing edge costs (from ∞ to some finite cost)
- Example:



- Original: Cost((1, 1), East) = ∞
- Relaxed: Cost_rel((1,1), East) = 1

General Framework

Definition: relaxed search problem

A **relaxation** P_{rel} of a search problem P has costs that satisfy:

$$\text{Cost}_{\text{rel}}(s, a) \leq \text{Cost}(s, a).$$

Definition: relaxed heuristic

Given a relaxed search problem P_{rel}, define the **relaxed heuristic** $h(s) = \text{FutureCost}_{\text{rel}}(s)$, the minimum cost from s to an end state using $\text{Cost}_{\text{rel}}(s, a)$.

Consistency

Theorem: consistency of relaxed heuristics

Suppose $h(s) = \text{FutureCost}_{\text{rel}}(s)$ for some relaxed problem P_{rel}.

Then $h(s)$ is a consistent heuristic.

- Proof:

$$h(s) \leq \text{Cost}_{\text{rel}}(s, a) + h(\text{Succ}(s, a)) \text{ [triangle inequality]}$$

$$\leq \text{Cost}(s, a) + h(\text{Succ}(s, a)) \text{ [relaxation]}$$

Trade-off

- Efficiency
  - h(s) = FutureCost_rel (s) must be easy to compute
  - Closed form, easier search, independent subproblems
- Tightness
  - heuristic h(s) should be close to FutureCost(s)
  - Don’t remove too many constraints

Recap

Week 2 Summary

- Solving problems by searching
  - Informed search strategies
  - Heuristics functions

Next Week

- Search in complex environments
  - Hill climbing, simulated annealing, local beam search, evolutionary algorithm.