CSC 384
Introduction to Artificial Intelligence

Heuristic Search

Alice Gao and Randy Hickey

Winter 2023

Learning Goals

By the end of this lecture, you should be able to

- Describe motivations for applying heuristic search algorithms.
- Trace the execution of and implement Greedy best-first search and A* search algorithm.
- Describe properties of Greedy best-first and A* search algorithms.
- Design an admissible heuristic function for a search problem.
- Describe strategies for choosing among multiple heuristic functions.

Outline

- Why Heuristic Search
- Greedy Best-First Search
- A* Search
- Constructing Heuristics

WHY HEURISTIC SEARCH

Why Heuristic Search?

How would _____ choose which state to expand?

- Humans
- An uninformed search algorithm

5	3	
8	7	6
2	4	1

1	2	3
4	5	
7	8	6

Uninformed v.s. Heuristic Search

Uninformed Search

- Has no problem-specific knowledge.
- Treats all the states in the same way.
- Does not know which state is closer to a goal.

Heuristic Search

- Has problem-specific knowledge, i.e., a heuristic.
- Considers some states to be more promising than others.
- Estimates which state is closer to a goal using the heuristic.

The Heuristic Function

Definition of a heuristic function:

A search heuristic $h(n)$ is an **estimate** of the cost of the **cheapest** path from node n to a goal node.

A good $h(n)$ has the properties below:

- Problem-specific.
- Non-negative.
- $h(n) = 0$ if n is a goal node.
- Can compute $h(n)$ easily without search.

Two Useful Functions

- Consider a state n.
- The cost function $g(n)$:
  - the cost of the path from the initial state to state n.
- The heuristic function $h(n)$:
  - the estimate of the cheapest path from state n to a goal state.

- UCS: remove the state w/ the lowest $g(n)$
- GBFS: remove the state w/ the lowest $h(n)$
- A*: remove the state w/ the lowest $f(n) = g(n) + h(n)$

GREEDY BEST-FIRST SEARCH

Greedy Best-First Search

- Frontier is a priority queue ordered by $h(n)$.
- Expands the node with the smallest $h(n)$.

GBFS Properties - Completeness

Is GBFS guaranteed to find a solution if a solution exists?

- No.
- Could you construct a counterexample?

GBFS Properties - Optimality

Is GBFS guaranteed to find an optimal solution?

- No.
- Could you construct a counterexample?

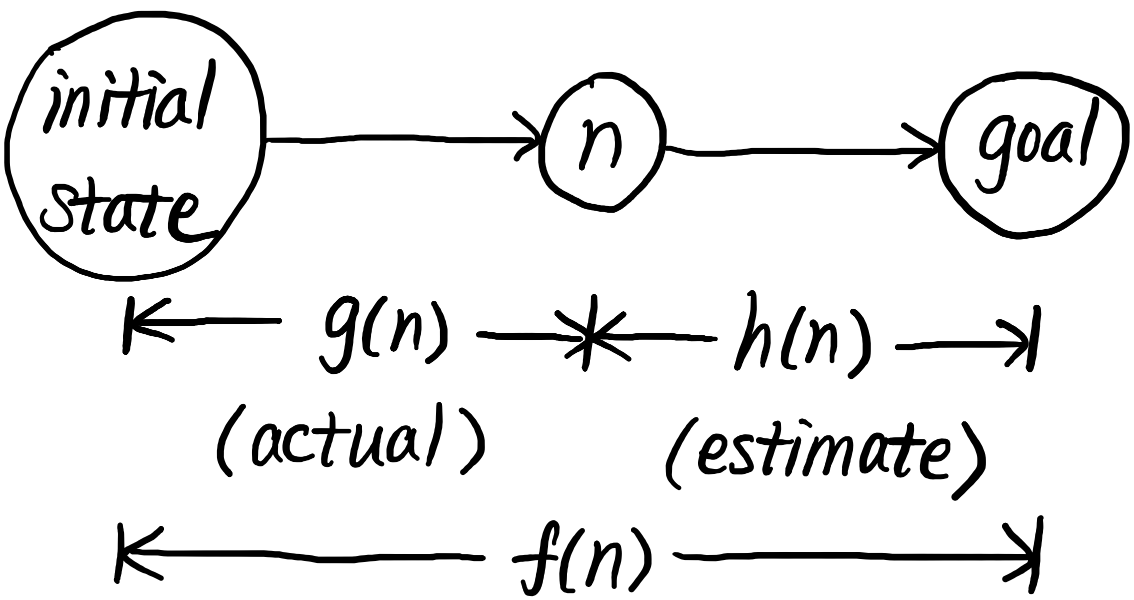
GBFS Properties – Space and Time Complexities

- Both are exponential.
- If $h(n) = 0$ for every state n, then GBFS becomes an uninformed search algorithm.

A* SEARCH

A* Search

- Frontier is a priority queue ordered by
$$f(n) = g(n) + h(n).$$
- Expands the node with the smallest $f(n)$.



Integer Example

- States: the non-negative integers {0, 1, 2, …}.
- Initial state: 0.
- Goal state: 5.
- Successor function: S(n) = {n+1, n+2}.
- Cost function: C(n, n+1) = 1, C(n, n+2) = 3.
- Heuristic function:
  - If $n \leq 5, h(n) = 5 - n$.
  - Otherwise, $h(n) = 0$.

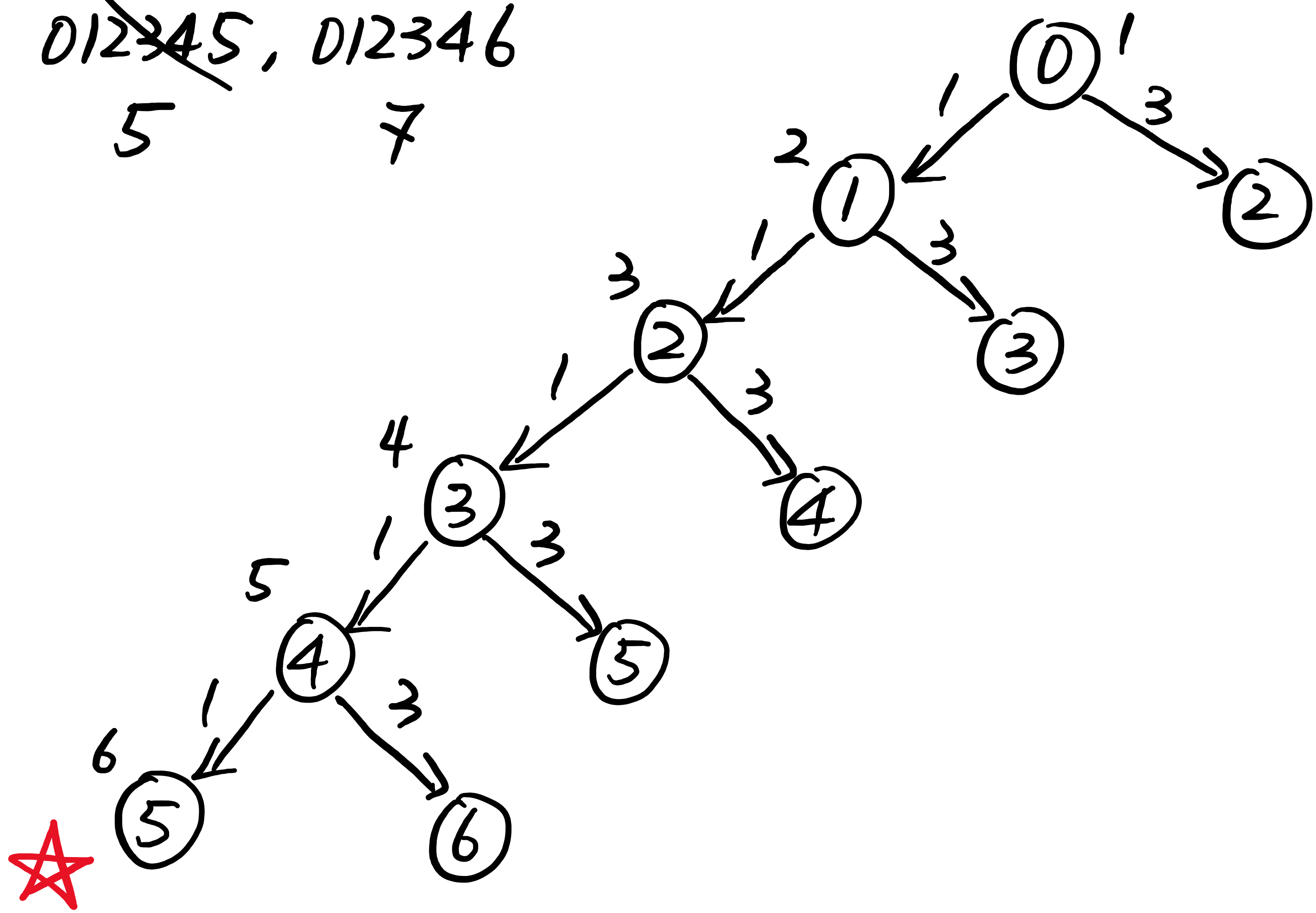
A* Search on Integer Example

frontier:
(f value)

A* Search on Integer Example

frontier: ~~0~~, ~~01~~, 02, ~~012~~, 013, ~~0123~~, 0124, ~~01234~~, 01235, ~~012345~~, 012346

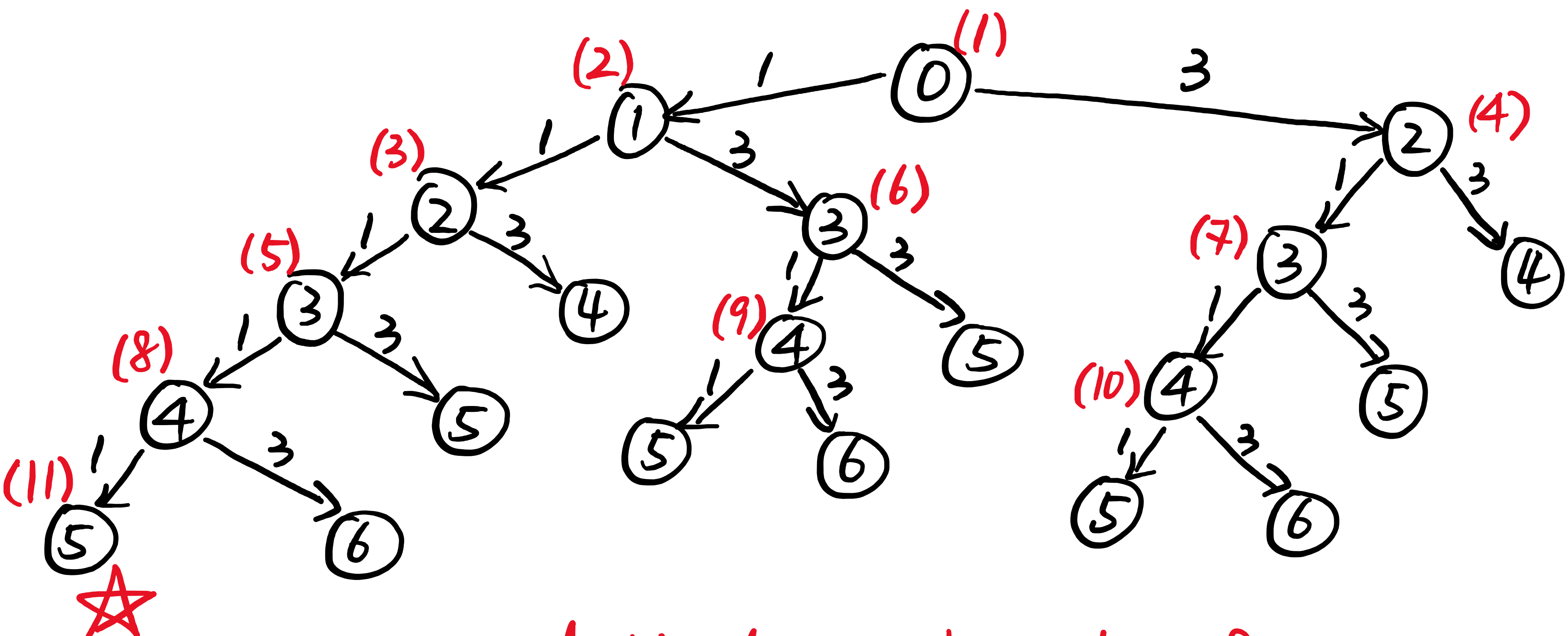
f value: 5, 5, 6, 5, 6, 5, 6, 5, 6, 5, 7



UCS on Integer Example

tie-breaking rule: expand oldest node on the frontier.

frontier: ~~0(5)~~, ~~01(1)~~, ~~02(3)~~, ~~012(2)~~, ~~013(4)~~, ~~0123(3)~~, 0124(5), ~~023(4)~~, 024(6), ~~01234(4)~~, 01235(6), ~~0134(5)~~, 0135(7), ~~0234(5)~~, 0235(7), ~~012345(5)~~, 012346(7), 01345(6), 01346(8), 02345(6), 02346(8)



red #'s denote the order of expansion.

A* Search Properties

- Complete?
  - Yes, under some mild conditions.
- Optimal?
  - Yes, if $h(n)$ satisfies a mild condition.
- Space Complexity
  - Exponential
- Time Complexity
  - Exponential

Admissible Heuristic Definition

A heuristic function $h(n)$ is admissible if and only if it never overestimates the cost of the cheapest path from state n to a goal state.

Let $h^*(n)$ denote the cost of the cheapest path from state n to a goal state. Then, an admissible $h(n)$ satisfies:

For every state n, $0 \leq h(n) \leq h^*(n)$.

Admissible Heuristic Intuition

What should an admissible $h(n)$ be if

- n is a goal state, or
- there is no path from state n to a goal state?

- A lower bound on the actual cost.
- Optimistic: it thinks the goal is closer than it really is.
- Ensures that A* doesn't miss any promising paths.
  - If both $g(n)$ and $h(n)$ are low, $f(n)$ will be low and A* will explore state n before considering more expensive paths.

A* is Optimal and Optimally Efficient.

Theorem (Optimality):

A* is optimal if and only if $h(n)$ is admissible.

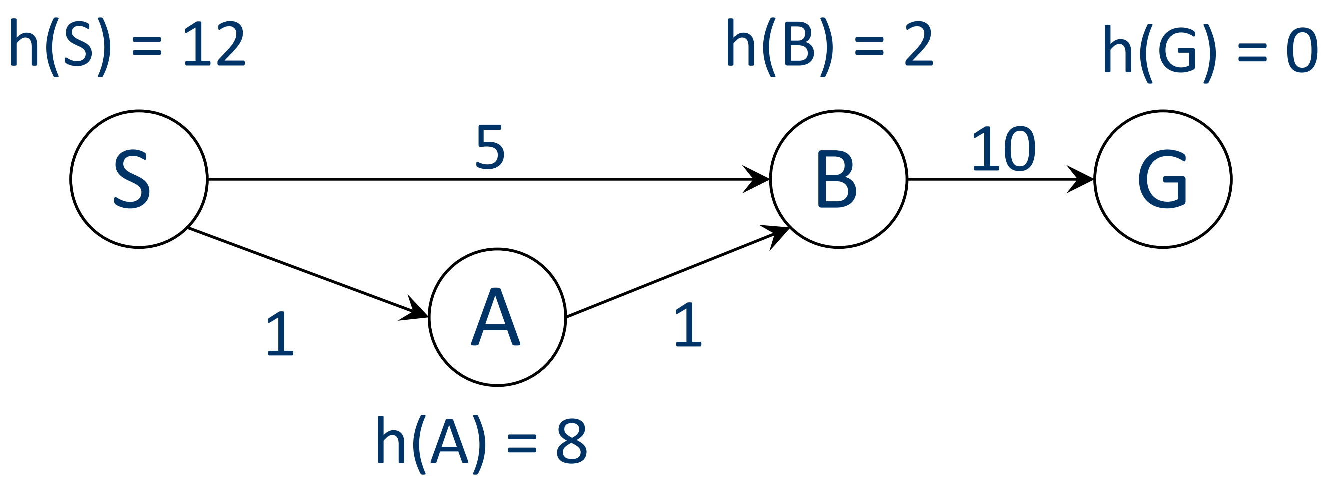
Theorem (Optimal Efficiency):

Among all optimal search algorithms that start from the **same initial state** and use the **same heuristic function**, A* expands the **fewest** states.

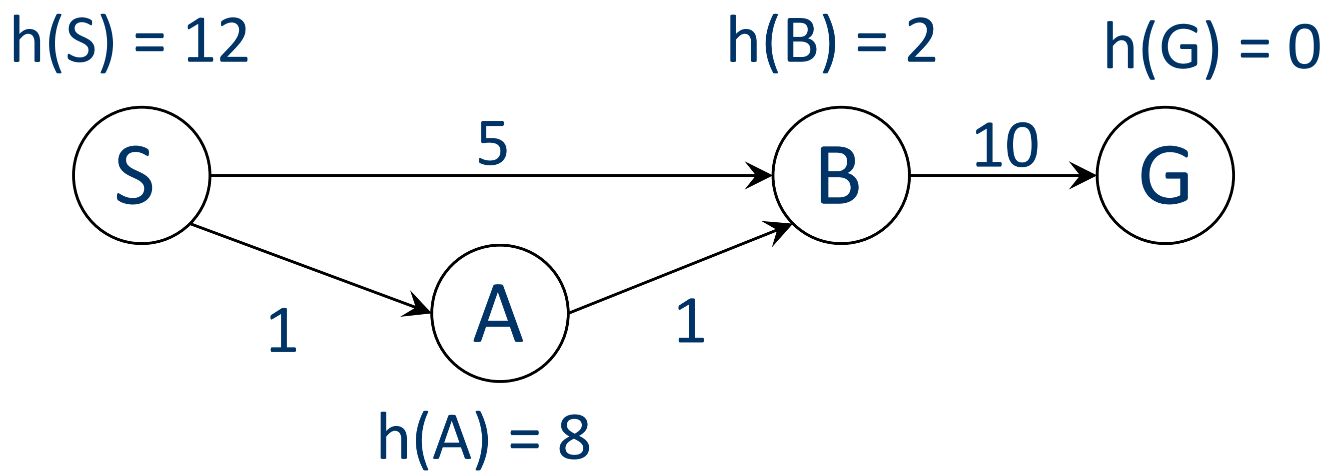
What about A* with Multi-Path Pruning?

Is A* with multi-path pruning optimal?

- Let's try the example below.



A* with Multi-Path Pruning Example



frontier :

explored :

solution found:

Optimal solution :

Consistent (Monotone) Heuristic

A heuristic function $h(n)$ is consistent if and only if

1. $h(n)$ is admissible, and
2. For every two neighboring states n_1 and n_2,
$$h(n_1) \leq g(n_1, n_2) + h(n_2).$$

Notes:

- If there are more than one edge from n_1 to n_2, the inequality must hold for all the edges.
- A consistent heuristic is admissible.

A* with Multi-Path Pruning

Theorem:

If $h(n)$ is consistent, A* w/ multi-path pruning is optimal.

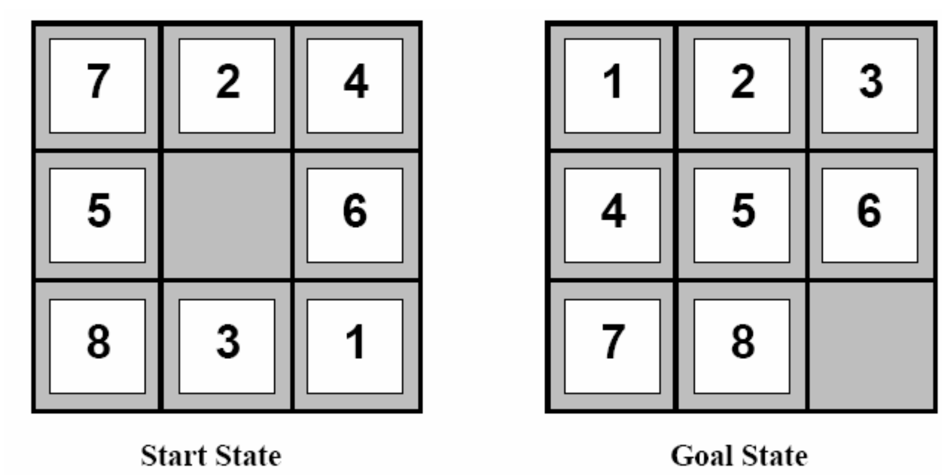
How do we ensure that $h(n)$ is consistent?

- Most admissible heuristic functions are consistent.
- Verify the definition of consistency.

CONSTRUCTING HEURISTICS

Two Admissible H(n) for 8-Puzzle

- Manhattan distance heuristic:
  - The sum of the Manhattan distances of the tiles to their goal positions
- Misplaced tile heuristic:
  - The number of tiles that are NOT in their goal positions



Constructing an Admissible Heuristic

1. Define a relaxed problem by simplifying or removing constraints on the original problem.

2. Solve the relaxed problem without search.

3. The cost of the optimal solution to the relaxed problem is an admissible heuristic for the original problem.

Constructing Admissible Heuristics for 8-Puzzle

8-puzzle rule: A tile can move from square A to B if

- A and B are adjacent, and
- B is empty.

Which heuristic functions can we derive from relaxed versions of this problem?

Removing One Constraint

Which heuristics can we derive from the following relaxed 8-puzzle?

A tile can move from square A to square B
if A and B are adjacent ~~and B is empty~~.

A. The Manhattan distance heuristic
B. The Misplaced tile heuristic
C. Another heuristic not described above

Removing Both Constraints

Which heuristics can we derive from the following relaxed 8-puzzle?

A tile can move from square A to square B

~~if A and B are adjacent and B is empty.~~

A. The Manhattan distance heuristic

B. The Misplaced tile heuristic

C. Another heuristic not described above

Dominating Heuristics

Definition (Dominating Heuristic):

$h_1(n)$ dominates $h_2(n)$ if and only if

- $h_1(n) \geq h_2(n)$, for every state n.
- $h_1(n) > h_2(n)$, for at least one state n.

Theorem:

If $h_1(n)$ dominates $h_2(n)$, A* with $h_1(n)$ never expands more states than A* with $h_2(n)$ for any problem.

Which 8-Puzzle Heuristic is Better?

Which of the two 8-puzzle heuristics is better?

A. The Manhattan distance heuristic dominates the Misplaced tile heuristic.
B. The Misplaced tile heuristic dominates the Manhattan distance heuristic.
C. Neither heuristic dominates the other one.