

# CS4618 Artificial Intelligence I

Today: Informed Search

Thomas Jansen

October 5<sup>th</sup>

# Plans for Today

## ① Introduction

Overview

## ② Informed Search Algorithms

Motivation and Greedy Best-First Search

A\* Search and Variants

## ③ Evaluating Heuristic Functions

Heuristic Functions for 8-Puzzle

Evaluating Heuristics

## ④ Summary

Summary & Take Home Message



# Informed Search

Remember uninformed search ( $\hat{=}$  blind search)



# Informed Search

Remember uninformed search ( $\hat{=}$  blind search)

- problem given by
  - initial state
  - possible actions
  - transition model
  - goal test
  - path cost



# Informed Search

Remember uninformed search ( $\hat{=}$  blind search)

- **problem** given by
  - initial state
  - possible actions
  - transition model
  - goal test
  - path cost
- **blind search**  $\hat{=}$  search algorithm knows **only** problem

## Informed Search

Remember uninformed search ( $\hat{=}$  blind search)

- problem given by
  - initial state
  - possible actions
  - transition model
  - goal test
  - path cost
- blind search  $\hat{=}$  search algorithm knows **only** problem

Informed Search additionally **evaluation function**  $f$  used  
 $f(v)$  estimates cost of node  $v$



## Informed Search

Remember uninformed search ( $\hat{=}$  blind search)

- problem given by
  - initial state
  - possible actions
  - transition model
  - goal test
  - path cost
- blind search  $\hat{=}$  search algorithm knows **only** problem

Informed Search additionally **evaluation function**  $f$  used  
 $f(v)$  estimates cost of node  $v$

**Application** algorithm explores node  $v$   
 with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$



## Informed Search

Remember uninformed search ( $\hat{=}$  blind search)

- problem given by
  - initial state
  - possible actions
  - transition model
  - goal test
  - path cost
- blind search  $\hat{=}$  search algorithm knows **only** problem

Informed Search additionally **evaluation function**  $f$  used  
 $f(v)$  estimates cost of node  $v$

**Application** algorithm explores node  $v$   
 with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
 $\hat{=}$  best-first search (like uni.-cost search)



## Informed Search

Remember uninformed search ( $\hat{=}$  blind search)

- problem given by
  - initial state
  - possible actions
  - transition model
  - goal test
  - path cost
- blind search  $\hat{=}$  search algorithm knows **only** problem

Informed Search additionally **evaluation function**  $f$  used  
 $f(v)$  estimates cost of node  $v$

**Application** algorithm explores node  $v$   
 with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
 $\hat{=}$  best-first search (like uni.-cost search)

**often**  $f$  incorporates **heuristic component**  
**heuristic component**  $\hat{=}$  additional knowledge  
 about problem



# Heuristic Functions

**Remember** algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function**  $h$



# Heuristic Functions

**Remember** algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function**  $h$

**Definition** **heuristic function**  $h$   
maps **states** to estimated cost of a cheapest path  
(of actions) from the state to a goal state



# Heuristic Functions

**Remember** algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function**  $h$

**Definition** **heuristic function**  $h$   
maps **states** to estimated cost of a cheapest path  
(of actions) from the state to a goal state

## Consequences

- $h(v) = 0 \Leftrightarrow v$  is goal state

# Heuristic Functions

**Remember** algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function  $h$**

**Definition** **heuristic function  $h$**   
maps **states** to estimated cost of a cheapest path  
(of actions) from the state to a goal state

## Consequences

- $h(v) = 0 \Leftrightarrow v$  is goal state
- $h(v) \geq 0$  (since we assume non-negative cost)

# Heuristic Functions

**Remember** algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function**  $h$

**Definition** heuristic function  $h$   
maps **states** to estimated cost of a cheapest path  
(of actions) from the state to a goal state

## Consequences

- $h(v) = 0 \Leftrightarrow v$  is goal state
- $h(v) \geq 0$  (since we assume non-negative cost)

**Remark**  $h$  depends on states  
**not** on the path to the state in the search tree



## Greedy Best-First Search

**Remember** informed search algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function**  $h$



## Greedy Best-First Search

**Remember** informed search algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function**  $h$

**Definition** of greedy best-first search  $f = h$



## Greedy Best-First Search

**Remember** informed search algorithm explores node  $v$   
 with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
 ( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function**  $h$

**Definition** of greedy best-first search  $f = h$

**Example** travelling from Cork to Dublin **by bike**  
**goal** minimise actual travel distance



## Greedy Best-First Search

**Remember** informed search algorithm explores node  $v$   
 with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
 ( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function**  $h$

**Definition** of greedy best-first search  $f = h$

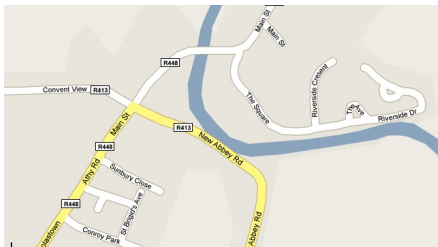
**Example** travelling from Cork to Dublin **by bike**  
**goal** minimise actual travel distance  
**possible** additional heuristic information  
 $\forall$  places: straight-line distance  $\text{place} \leftrightarrow \text{Dublin}$

# Greedy Best-First Search

**Remember** informed search algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function  $h$**

**Definition** of greedy best-first search  $f = h$

**Example** travelling from Cork to Dublin **by bike**  
**goal** minimise actual travel distance  
**possible** additional heuristic information  
 $\forall$  places: straight-line distance place  $\leftrightarrow$  Dublin



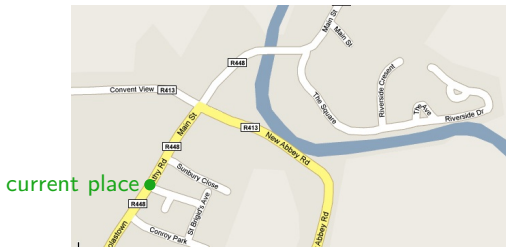


## Greedy Best-First Search

**Remember** informed search algorithm explores node  $v$   
 with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
 ( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function  $h$**

**Definition** of greedy best-first search  $f = h$

**Example** travelling from Cork to Dublin **by bike**  
**goal** minimise actual travel distance  
**possible** additional heuristic information  
 $\forall$  places: straight-line distance place  $\leftrightarrow$  Dublin



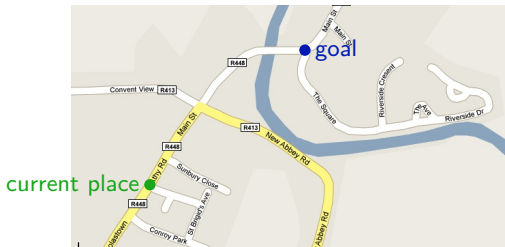


## Greedy Best-First Search

**Remember** informed search algorithm explores node  $v$   
 with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
 ( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function  $h$**

**Definition** of greedy best-first search  $f = h$

**Example** travelling from Cork to Dublin **by bike**  
**goal** minimise actual travel distance  
**possible** additional heuristic information  
 $\forall$  places: straight-line distance place  $\leftrightarrow$  Dublin

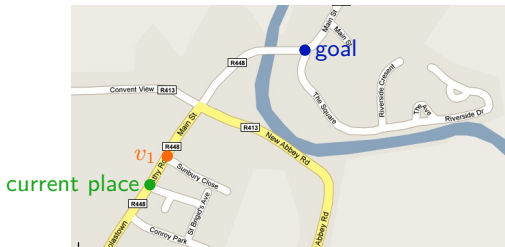


# Greedy Best-First Search

**Remember** informed search algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function  $h$**

**Definition** of greedy best-first search  $f = h$

**Example** travelling from Cork to Dublin **by bike**  
**goal** minimise actual travel distance  
**possible** additional heuristic information  
 $\forall$  places: straight-line distance place  $\leftrightarrow$  Dublin

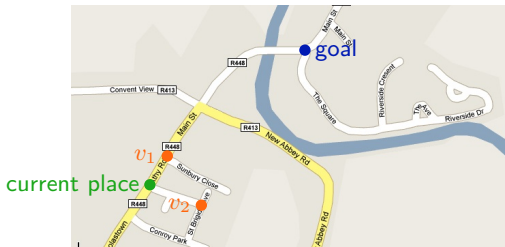


# Greedy Best-First Search

**Remember** informed search algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function  $h$**

**Definition** of greedy best-first search  $f = h$

**Example** travelling from Cork to Dublin **by bike**  
**goal** minimise actual travel distance  
**possible** additional heuristic information  
 $\forall$  places: straight-line distance place  $\leftrightarrow$  Dublin

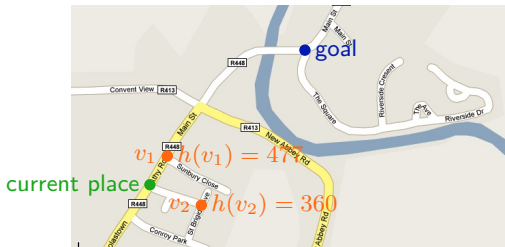


# Greedy Best-First Search

**Remember** informed search algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function  $h$**

**Definition** of greedy best-first search  $f = h$

**Example** travelling from Cork to Dublin **by bike**  
**goal** minimise actual travel distance  
**possible** additional heuristic information  
 $\forall$  places: straight-line distance place  $\leftrightarrow$  Dublin

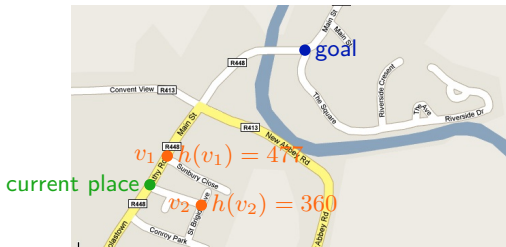


# Greedy Best-First Search

**Remember** informed search algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function  $h$**

**Definition** of greedy best-first search  $f = h$

**Example** travelling from Cork to Dublin **by bike**  
**goal** minimise actual travel distance  
**possible** additional heuristic information  
 $\forall$  places: straight-line distance place  $\leftrightarrow$  Dublin



**Observe** may lead to **wrong decisions** and even **infinite loops**



# Properties of Greedy Best-First Search



# Properties of Greedy Best-First Search

- Completeness

# Properties of Greedy Best-First Search

- Completeness  
in general **no**



## Properties of Greedy Best-First Search

- **Completeness**  
in general **no**  
but **yes** if state space finite and no nodes are 'repeated'



## Properties of Greedy Best-First Search

- Completeness  
in general **no**  
but **yes** if state space finite and no nodes are 'repeated'
- Optimality



# Properties of Greedy Best-First Search

- **Completeness**  
in general **no**  
but **yes** if state space finite and no nodes are 'repeated'
- **Optimality**  
in general **no**



# Properties of Greedy Best-First Search

- Completeness  
in general **no**  
but **yes** if state space finite and no nodes are 'repeated'
- Optimality  
in general **no**
- Time Complexity

# Properties of Greedy Best-First Search

- **Completeness**  
in general **no**  
but **yes** if state space finite and no nodes are 'repeated'
- **Optimality**  
in general **no**
- **Time Complexity**  
in 'tree version'  $O(b^m)$



## Properties of Greedy Best-First Search

- **Completeness**  
in general **no**  
but **yes** if state space finite and no nodes are 'repeated'
- **Optimality**  
in general **no**
- **Time Complexity**  
in 'tree version'  $O(b^m)$
- **Space Complexity**



# Properties of Greedy Best-First Search

- **Completeness**  
in general **no**  
but **yes** if state space finite and no nodes are 'repeated'
- **Optimality**  
in general **no**
- **Time Complexity**  
in 'tree version'  $O(b^m)$
- **Space Complexity**  
in 'tree version'  $O(b^m)$



# Properties of Greedy Best-First Search

- Completeness  
in general **no**  
but **yes** if state space finite and no nodes are 'repeated'
- Optimality  
in general **no**
- Time Complexity  
in 'tree version'  $O(b^m)$
- Space Complexity  
in 'tree version'  $O(b^m)$

Why consider greedy best-first search at all?

## Properties of Greedy Best-First Search

- **Completeness**  
in general **no**  
but **yes** if state space finite and no nodes are 'repeated'
- **Optimality**  
in general **no**
- **Time Complexity**  
in 'tree version'  $O(b^m)$
- **Space Complexity**  
in 'tree version'  $O(b^m)$

Why consider greedy best-first search at all?

**Fact** heuristic function  $h$  matters  
may lead to substantial reduction in time and space complexity



# A\* Search

**Remember** informed search algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function  $h$**



# A\* Search

**Remember** informed search algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function  $h$**

**Definition** of A\* search  $f = g + h$   
 $g(v)$  = path cost from start node to  $v$   
(**Remark** slight abuse of notation due to different domains)



# A\* Search

**Remember** informed search algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function  $h$**

**Definition** of A\* search  $f = g + h$   
 $g(v)$  = path cost from start node to  $v$   
(**Remark** slight abuse of notation due to different domains)

**Remember** **heuristic function  $h$**   
maps **states** to estimated cost of a cheapest path  
(of actions) from the state to a goal state

# A\* Search

**Remember** informed search algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function**  $h$

**Definition** of A\* search  $f = g + h$   
 $g(v)$  = path cost from start node to  $v$   
(**Remark** slight abuse of notation due to different domains)

**Remember** **heuristic function**  $h$   
maps **states** to estimated cost of a cheapest path  
(of actions) from the state to a goal state

**Thus**  $f(v)$  = estimated cost of cheapest solution via  $v$

# A\* Search

**Remember** informed search algorithm explores node  $v$   
with  $f(v) = \min\{f(w) \mid w \in \text{frontier}\}$   
( $\hat{=}$  best-first search)  
 $f$  incorporates **heuristic function**  $h$

**Definition** of A\* search  $f = g + h$   
 $g(v)$  = path cost from start node to  $v$   
(**Remark** slight abuse of notation due to different domains)

**Remember** **heuristic function**  $h$   
maps **states** to estimated cost of a cheapest path  
(of actions) from the state to a goal state

**Thus**  $f(v)$  = estimated cost of cheapest solution via  $v$

**Remark** **depending** on  $h$  A\* search can be **complete** and **optimal**



# Admissible Heuristics



# Admissible Heuristics

## Definition (Admissibility)

A heuristic function  $h$  is called **admissible** if and only if

$$\forall s \in \text{state space: } h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$$

$$\text{transition-model}(\text{transition-model}(\dots$$

$$\text{transition-model}(s, a_1), a_2), \dots, a_n) \text{ is goal} \}$$

holds.



# Admissible Heuristics

## Definition (Admissibility)

A heuristic function  $h$  is called **admissible** if and only if

$$\forall s \in \text{state space: } h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$$

$$\text{transition-model}(\text{transition-model}(\dots$$

$$\text{transition-model}(s, a_1), a_2), \dots, a_n) \text{ is goal} \}$$

holds.

Observation

# Admissible Heuristics

## Definition (Admissibility)

A heuristic function  $h$  is called **admissible** if and only if

$$\forall s \in \text{state space: } h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid \\ \text{transition-model}(\text{transition-model}(\dots \\ \text{transition-model}(s, a_1), a_2), \dots, a_n) \text{ is goal} \}$$

holds.

## Observation

- admissible  $h$  never overestimates

# Admissible Heuristics

## Definition (Admissibility)

A heuristic function  $h$  is called **admissible** if and only if

$$\forall s \in \text{state space: } h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid \\ \text{transition-model}(\text{transition-model}(\dots \\ \text{transition-model}(s, a_1), a_2), \dots, a_n) \text{ is goal} \}$$

holds.

## Observation

- admissible  $h$  never overestimates
- admissible  $h$  are optimistic

# Admissible Heuristics

## Definition (Admissibility)

A heuristic function  $h$  is called **admissible** if and only if

$$\forall s \in \text{state space: } h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid \\ \text{transition-model}(\text{transition-model}(\dots \\ \text{transition-model}(s, a_1), a_2), \dots, a_n) \text{ is goal} \}$$

holds.

## Observation

- admissible  $h$  never overestimates
- admissible  $h$  are optimistic

**Example** **straight-line distance** for street distances

# Consistent Heuristics

# Consistent Heuristics

## Definition (Consistency)

A heuristic function  $h$  is called **consistent** if and only if

$\forall s \in \text{state space, applicable action } a,$

$$s' = \text{transition-model}(s, a) : h(s) \leq \text{cost}(a) + h(s')$$

holds.



# Consistent Heuristics

## Definition (Consistency)

A heuristic function  $h$  is called **consistent** if and only if

$\forall s \in \text{state space, applicable action } a,$

$$s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$$

holds.

**Remark** sometimes also called **monotone**

# Consistent Heuristics

## Definition (Consistency)

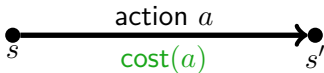
A heuristic function  $h$  is called **consistent** if and only if

$\forall s \in \text{state space, applicable action } a,$

$$s' = \text{transition-model}(s, a) : h(s) \leq \text{cost}(a) + h(s')$$

holds.

**Remark** sometimes also called **monotone**



# Consistent Heuristics

## Definition (Consistency)

A heuristic function  $h$  is called **consistent** if and only if

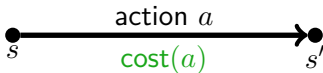
$\forall s \in \text{state space, applicable action } a,$

$$s' = \text{transition-model}(s, a) : h(s) \leq \text{cost}(a) + h(s')$$

holds.

**Remark** sometimes also called **monotone**

goal



# Consistent Heuristics

## Definition (Consistency)

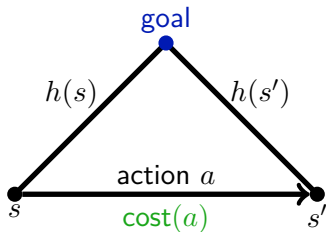
A heuristic function  $h$  is called **consistent** if and only if

$\forall s \in \text{state space, applicable action } a,$

$$s' = \text{transition-model}(s, a) : h(s) \leq \text{cost}(a) + h(s')$$

holds.

**Remark** sometimes also called **monotone**



# Consistent Heuristics

## Definition (Consistency)

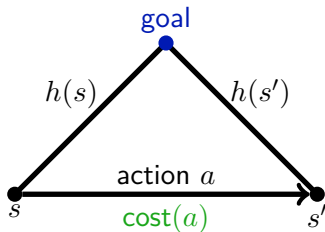
A heuristic function  $h$  is called **consistent** if and only if

$\forall s \in \text{state space, applicable action } a,$

$$s' = \text{transition-model}(s, a) : h(s) \leq \text{cost}(a) + h(s')$$

holds.

**Remark** sometimes also called **monotone**



**Observation** monotonicity  $\hat{=}$  triangle inequality

# Consistent Heuristics

## Definition (Consistency)

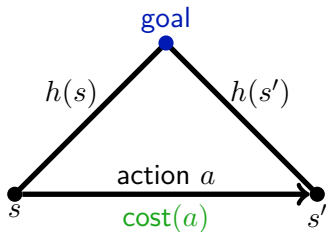
A heuristic function  $h$  is called **consistent** if and only if

$\forall s \in \text{state space, applicable action } a,$

$$s' = \text{transition-model}(s, a) : h(s) \leq \text{cost}(a) + h(s')$$

holds.

**Remark** sometimes also called **monotone**



**Observation** monotonicity  $\hat{=}$  triangle inequality

**Observation** monotonicity  $\Rightarrow$  admissibility



# Admissibility and Consistency

Remember  $h$  admissible

$\Leftrightarrow \forall s \in \text{state space}: h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$   
 transition-model(transition-model( $\dots$   
 (transition-model( $s, a_1$ ),  $a_2$ ),  $\dots$ ,  $a_n$ ) is goal}  
 (no overestimating)

$h$  consistent

$\Leftrightarrow \forall s \in \text{state space}, \text{applicable action } a,$   
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (triangle inequality)



# Admissibility and Consistency

Remember  $h$  admissible

$\Leftrightarrow \forall s \in \text{state space}: h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$   
 transition-model(transition-model( $\dots$   
 (transition-model( $s, a_1$ ),  $a_2$ ),  $\dots$ ,  $a_n$ ) is goal}  
 (no overestimating)

$h$  consistent

$\Leftrightarrow \forall s \in \text{state space}, \text{applicable action } a,$   
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (triangle inequality)

Claim  $h$  consistent  $\Rightarrow h$  admissible

# Admissibility and Consistency

Remember  $h$  admissible

$\Leftrightarrow \forall s \in \text{state space}: h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$   
 transition-model(transition-model( $\dots$   
 (transition-model( $s, a_1$ ),  $a_2$ ),  $\dots$ ,  $a_n$ ) is goal}  
 (no overestimating)

$h$  consistent

$\Leftrightarrow \forall s \in \text{state space}, \text{applicable action } a,$   
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (triangle inequality)

Claim  $h$  consistent  $\Rightarrow h$  admissible

Consider minimal distance  $d$  from  $s$  to goal

# Admissibility and Consistency

Remember  $h$  admissible

$\Leftrightarrow \forall s \in \text{state space}: h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$   
 transition-model(transition-model( $\dots$   
 (transition-model( $s, a_1$ ),  $a_2$ ),  $\dots$ ,  $a_n$ ) is goal}  
 (no overestimating)

$h$  consistent

$\Leftrightarrow \forall s \in \text{state space}, \text{applicable action } a,$   
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (triangle inequality)

Claim  $h$  consistent  $\Rightarrow h$  admissible

Consider minimal distance  $d$  from  $s$  to goal

Observe  $d = 0$

# Admissibility and Consistency

Remember  $h$  admissible

$\Leftrightarrow \forall s \in \text{state space}: h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$   
 transition-model(transition-model( $\dots$   
 (transition-model( $s, a_1$ ),  $a_2$ ),  $\dots$ ,  $a_n$ ) is goal}  
 (no overestimating)

$h$  consistent

$\Leftrightarrow \forall s \in \text{state space}, \text{applicable action } a,$   
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (triangle inequality)

Claim  $h$  consistent  $\Rightarrow h$  admissible

Consider minimal distance  $d$  from  $s$  to goal

Observe  $d = 0$  no action,  $h(s) = 0$

# Admissibility and Consistency

Remember  $h$  admissible

$\Leftrightarrow \forall s \in \text{state space}: h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$   
 transition-model(transition-model( $\dots$   
 (transition-model( $s, a_1$ ),  $a_2$ ),  $\dots$ ,  $a_n$ ) is goal}  
 (no overestimating)

$h$  consistent

$\Leftrightarrow \forall s \in \text{state space}, \text{applicable action } a,$   
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (triangle inequality)

Claim  $h$  consistent  $\Rightarrow h$  admissible

Consider minimal distance  $d$  from  $s$  to goal

Observe  $d = 0$  no action,  $h(s) = 0$  ✓

# Admissibility and Consistency

Remember  $h$  admissible

$\Leftrightarrow \forall s \in \text{state space}: h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$   
 transition-model(transition-model( $\dots$   
 (transition-model( $s, a_1$ ),  $a_2$ ),  $\dots$ ,  $a_n$ ) is goal}  
 (no overestimating)

$h$  consistent

$\Leftrightarrow \forall s \in \text{state space}, \text{applicable action } a,$   
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (triangle inequality)

Claim  $h$  consistent  $\Rightarrow h$  admissible

Consider minimal distance  $d$  from  $s$  to goal

Observe  $d = 0$  no action,  $h(s) = 0$  ✓

Observe  $d = 1$

# Admissibility and Consistency

Remember  $h$  admissible

$\Leftrightarrow \forall s \in \text{state space}: h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$   
 transition-model(transition-model( $\dots$   
 (transition-model( $s, a_1$ ),  $a_2$ ),  $\dots$ ,  $a_n$ ) is goal}  
 (no overestimating)

$h$  consistent

$\Leftrightarrow \forall s \in \text{state space}$ , applicable action  $a$ ,  
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (triangle inequality)

Claim  $h$  consistent  $\Rightarrow h$  admissible

Consider minimal distance  $d$  from  $s$  to goal

Observe  $d = 0$  no action,  $h(s) = 0$  ✓

Observe  $d = 1$   $h(s) \leq \text{cost}(a) + h(s')$

# Admissibility and Consistency

Remember  $h$  admissible

$\Leftrightarrow \forall s \in \text{state space}: h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$   
 transition-model(transition-model( $\dots$   
 (transition-model( $s, a_1$ ),  $a_2$ ),  $\dots$ ,  $a_n$ ) is goal}  
 (no overestimating)

$h$  consistent

$\Leftrightarrow \forall s \in \text{state space}$ , applicable action  $a$ ,  
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (triangle inequality)

Claim  $h$  consistent  $\Rightarrow h$  admissible

Consider minimal distance  $d$  from  $s$  to goal

Observe  $d = 0$  no action,  $h(s) = 0$  ✓

Observe  $d = 1$   $h(s) \leq \text{cost}(a) + h(s') = \text{cost}(a)$

# Admissibility and Consistency

Remember  $h$  admissible

$\Leftrightarrow \forall s \in \text{state space}: h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$   
 transition-model(transition-model( $\dots$   
 (transition-model( $s, a_1$ ),  $a_2$ ),  $\dots$ ,  $a_n$ ) is goal}  
 (no overestimating)

$h$  consistent

$\Leftrightarrow \forall s \in \text{state space}, \text{applicable action } a,$   
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (triangle inequality)

Claim  $h$  consistent  $\Rightarrow h$  admissible

Consider minimal distance  $d$  from  $s$  to goal

Observe  $d = 0$  no action,  $h(s) = 0$  ✓

Observe  $d = 1$   $h(s) \leq \text{cost}(a) + h(s') = \text{cost}(a)$  ✓

# Admissibility and Consistency

Remember  $h$  admissible

$\Leftrightarrow \forall s \in \text{state space}: h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$   
 transition-model(transition-model( $\dots$   
 (transition-model( $s, a_1$ ),  $a_2$ ),  $\dots$ ,  $a_n$ ) is goal}  
 (no overestimating)

$h$  consistent

$\Leftrightarrow \forall s \in \text{state space}, \text{applicable action } a,$   
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (triangle inequality)

Claim  $h$  consistent  $\Rightarrow h$  admissible

Consider minimal distance  $d$  from  $s$  to goal

Observe  $d = 0$  no action,  $h(s) = 0$  ✓

Observe  $d = 1$   $h(s) \leq \text{cost}(a) + h(s') = \text{cost}(a)$  ✓

Observe  $d > 1$

# Admissibility and Consistency

Remember  $h$  admissible

$\Leftrightarrow \forall s \in \text{state space}: h(s) \leq \min \{ \text{cost}(a_1, a_2, \dots, a_n) \mid$   
 transition-model(transition-model( $\dots$   
 (transition-model( $s, a_1$ ),  $a_2$ ),  $\dots$ ,  $a_n$ ) is goal}  
 (no overestimating)

$h$  consistent

$\Leftrightarrow \forall s \in \text{state space}$ , applicable action  $a$ ,  
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (triangle inequality)

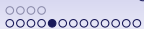
Claim  $h$  consistent  $\Rightarrow h$  admissible

Consider minimal distance  $d$  from  $s$  to goal

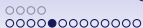
Observe  $d = 0$  no action,  $h(s) = 0$  ✓

Observe  $d = 1$   $h(s) \leq \text{cost}(a) + h(s') = \text{cost}(a)$  ✓

Observe  $d > 1$  consider one step and  $d - 1$  by induction ✓



# A\* Search with Consistent Heuristics



# A\* Search with Consistent Heuristics

**Claim** A\* search with a consistent  $h$  is **optimal**



# A\* Search with Consistent Heuristics

**Claim** A\* search with a consistent  $h$  is **optimal**

**Remember**

$$\underbrace{f}_{\text{evaluation}} = \underbrace{g}_{\text{cost}} + \underbrace{h}_{\text{heuristic}}$$



# A\* Search with Consistent Heuristics

**Claim** A\* search with a consistent  $h$  is **optimal**

**Remember**

$$\underbrace{f}_{\text{evaluation}} = \underbrace{g}_{\text{cost}} + \underbrace{h}_{\text{heuristic}}$$

**Sub-Claim 1** along any path  $f$ -values are non-decreasing



# A\* Search with Consistent Heuristics

**Claim** A\* search with a consistent  $h$  is **optimal**

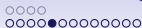
**Remember**

$$\underbrace{f}_{\text{evaluation}} = \underbrace{g}_{\text{cost}} + \underbrace{h}_{\text{heuristic}}$$

**Sub-Claim 1** along any path  $f$ -values are non-decreasing

**Proof**

**Consider** state  $s$ , action  $a$ , state  $s' = \text{transition-model}(s, a)$



# A\* Search with Consistent Heuristics

**Claim** A\* search with a consistent  $h$  is **optimal**

**Remember**

$$\underbrace{f}_{\text{evaluation}} = \underbrace{g}_{\text{cost}} + \underbrace{h}_{\text{heuristic}}$$

**Sub-Claim 1** along any path  $f$ -values are non-decreasing

**Proof**

**Consider** state  $s$ , action  $a$ , state  $s' = \text{transition-model}(s, a)$   
 $f(s')$



# A\* Search with Consistent Heuristics

**Claim** A\* search with a consistent  $h$  is **optimal**

**Remember**

$$\underbrace{f}_{\text{evaluation}} = \underbrace{g}_{\text{cost}} + \underbrace{h}_{\text{heuristic}}$$

**Sub-Claim 1** along any path  $f$ -values are non-decreasing

**Proof**

**Consider** state  $s$ , action  $a$ , state  $s' = \text{transition-model}(s, a)$

$$\underbrace{f(s')}_{\text{definition}} = g(s') + h(s')$$



# A\* Search with Consistent Heuristics

**Claim** A\* search with a consistent  $h$  is **optimal**

**Remember**

$$\underbrace{f}_{\text{evaluation}} = \underbrace{g}_{\text{cost}} + \underbrace{h}_{\text{heuristic}}$$

**Sub-Claim 1** along any path  $f$ -values are non-decreasing

**Proof**

**Consider** state  $s$ , action  $a$ , state  $s' = \text{transition-model}(s, a)$

$$f(s') \underbrace{=}_{\text{definition}} g(s') + h(s') \underbrace{=}_{\text{by transition}} g(s) + \text{cost}(a) + h(s')$$

# A\* Search with Consistent Heuristics

**Claim** A\* search with a consistent  $h$  is **optimal**

**Remember**

$$\underbrace{f}_{\text{evaluation}} = \underbrace{g}_{\text{cost}} + \underbrace{h}_{\text{heuristic}}$$

**Sub-Claim 1** along any path  $f$ -values are non-decreasing

**Proof**

**Consider**

state  $s$ , action  $a$ , state  $s' = \text{transition-model}(s, a)$

$$f(s') \underbrace{=}_{\text{definition}} g(s') + h(s') \underbrace{=}_{\text{by transition}} g(s) + \text{cost}(a) + h(s')$$

$$\underbrace{\geq}_{\text{consistency}} g(s) + h(s)$$

# A\* Search with Consistent Heuristics

**Claim** A\* search with a consistent  $h$  is **optimal**

**Remember**

$$\underbrace{f}_{\text{evaluation}} = \underbrace{g}_{\text{cost}} + \underbrace{h}_{\text{heuristic}}$$

**Sub-Claim 1** along any path  $f$ -values are non-decreasing

**Proof**

**Consider**

$$\begin{aligned} \text{state } s, \text{ action } a, \text{ state } s' &= \text{transition-model}(s, a) \\ f(s') &\underbrace{=}_{\text{definition}} g(s') + h(s') \underbrace{=}_{\text{by transition}} g(s) + \text{cost}(a) + h(s') \\ &\underbrace{\geq}_{\text{consistency}} g(s) + h(s) \underbrace{=}_{\text{definition}} f(s) \checkmark \end{aligned}$$

# A\* Search with Consistent Heuristics

**Claim** A\* search with a consistent  $h$  is **optimal**

**Remember**

$$\underbrace{f}_{\text{evaluation}} = \underbrace{g}_{\text{cost}} + \underbrace{h}_{\text{heuristic}}$$

**Sub-Claim 1** along any path  $f$ -values are non-decreasing

**Proof**

**Consider**

state  $s$ , action  $a$ , state  $s' = \text{transition-model}(s, a)$

$$f(s') \underbrace{=}_{\text{definition}} g(s') + h(s') \underbrace{=}_{\text{by transition}} g(s) + \text{cost}(a) + h(s')$$

$$\underbrace{\geq}_{\text{consistency}} g(s) + h(s) \underbrace{=}_{\text{definition}} f(s) \checkmark$$

for each edge, thus for each path  $\square$



## A\* Search with Consistent Heuristics (continued)

**Claim** A\* search with a consistent  $h$  is **optimal**

**Sub-Claim 1** along any path  $f$ -values are non-decreasing ✓



## A\* Search with Consistent Heuristics (continued)

**Claim** A\* search with a consistent  $h$  is **optimal**

**Sub-Claim 1** along any path  $f$ -values are non-decreasing ✓

**Sub-Claim 2** A\* selects  $v$  for expansion  
⇒ optimal path to  $v$  found



# A\* Search with Consistent Heuristics (continued)

**Claim** A\* search with a consistent  $h$  is **optimal**

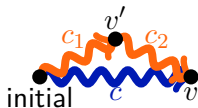
**Sub-Claim 1** along any path  $f$ -values are non-decreasing ✓

**Sub-Claim 2** A\* selects  $v$  for expansion  
 $\Rightarrow$  optimal path to  $v$  found

**Proof**

**Assume**

**otherwise**  $\exists$



with  $c > c_1 + c_2$



# A\* Search with Consistent Heuristics (continued)

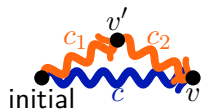
**Claim** A\* search with a consistent  $h$  is **optimal**

**Sub-Claim 1** along any path  $f$ -values are non-decreasing ✓

**Sub-Claim 2** A\* selects  $v$  for expansion  
 $\Rightarrow$  optimal path to  $v$  found

**Proof**

**Assume**

otherwise  $\exists$   with  $c > c_1 + c_2$

**Observe**  $f(v) \geq f(v')$  (due to Sub-Claim 1)

# A\* Search with Consistent Heuristics (continued)

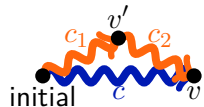
**Claim** A\* search with a consistent  $h$  is **optimal**

**Sub-Claim 1** along any path  $f$ -values are non-decreasing ✓

**Sub-Claim 2** A\* selects  $v$  for expansion  
 $\Rightarrow$  optimal path to  $v$  found

**Proof**

**Assume**

otherwise  $\exists$   with  $c > c_1 + c_2$

**Observe**  $f(v) \geq f(v')$  (due to Sub-Claim 1)

**Thus**  $v'$  expanded prior to  $v$



# A\* Search with Consistent Heuristics (continued)

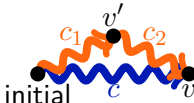
**Claim** A\* search with a consistent  $h$  is **optimal**

**Sub-Claim 1** along any path  $f$ -values are non-decreasing ✓

**Sub-Claim 2** A\* selects  $v$  for expansion  
 $\Rightarrow$  optimal path to  $v$  found

**Proof**

**Assume**

otherwise  $\exists$   with  $c > c_1 + c_2$

**Observe**  $f(v) \geq f(v')$  (due to Sub-Claim 1)

**Thus**  $v'$  expanded prior to  $v$

**Thus** optimal route found prior to expanding  $v$  ✗ □



## A\* Search with Consistent Heuristics (continued)

Claim A\* search with a consistent  $h$  is **optimal**

Sub-Claim 1 along any path  $f$ -values are non-decreasing ✓

Sub-Claim 2 A\* selects  $v$  for expansion  
⇒ optimal path to  $v$  found ✓



## A\* Search with Consistent Heuristics (continued)

**Claim** A\* search with a consistent  $h$  is **optimal**

**Sub-Claim 1** along any path  $f$ -values are non-decreasing ✓

**Sub-Claim 2** A\* selects  $v$  for expansion  
⇒ optimal path to  $v$  found ✓

**Observation** A\* expands nodes in non-decreasing  $f$ -order  
**follows** from Sub-Claim 1 and Sub-Claim 2

## A\* Search with Consistent Heuristics (continued)

**Claim** A\* search with a consistent  $h$  is **optimal**

**Sub-Claim 1** along any path  $f$ -values are non-decreasing ✓

**Sub-Claim 2** A\* selects  $v$  for expansion  
 $\Rightarrow$  optimal path to  $v$  found ✓

**Observation** A\* expands nodes in non-decreasing  $f$ -order  
**follows** from Sub-Claim 1 and Sub-Claim 2

**Consider** first considered **goal node**  $v$

## A\* Search with Consistent Heuristics (continued)

**Claim** A\* search with a consistent  $h$  is **optimal**

**Sub-Claim 1** along any path  $f$ -values are non-decreasing ✓

**Sub-Claim 2** A\* selects  $v$  for expansion  
 $\Rightarrow$  optimal path to  $v$  found ✓

**Observation** A\* expands nodes in non-decreasing  $f$ -order  
**follows** from Sub-Claim 1 and Sub-Claim 2

**Consider** first considered **goal node**  $v$

**Observation**  $f(v) = g(v)$  (true cost)  
 since  $h(v) = 0$  for all goal nodes (by definition)

## A\* Search with Consistent Heuristics (continued)

**Claim** A\* search with a consistent  $h$  is **optimal**

**Sub-Claim 1** along any path  $f$ -values are non-decreasing ✓

**Sub-Claim 2** A\* selects  $v$  for expansion  
 $\Rightarrow$  optimal path to  $v$  found ✓

**Observation** A\* expands nodes in non-decreasing  $f$ -order  
**follows** from Sub-Claim 1 and Sub-Claim 2

**Consider** first considered **goal node**  $v$

**Observation**  $f(v) = g(v)$  (true cost)  
 since  $h(v) = 0$  for all goal nodes (by definition)

**Consequence**

- first considered goal node has minimal cost ✓

## A\* Search with Consistent Heuristics (continued)

Claim A\* search with a consistent  $h$  is **optimal**

Sub-Claim 1 along any path  $f$ -values are non-decreasing ✓

Sub-Claim 2 A\* selects  $v$  for expansion  
 $\Rightarrow$  optimal path to  $v$  found ✓

Observation A\* expands nodes in non-decreasing  $f$ -order  
**follows** from Sub-Claim 1 and Sub-Claim 2

Consider first considered **goal node**  $v$

Observation  $f(v) = g(v)$  (true cost)  
 since  $h(v) = 0$  for all goal nodes (by definition)

Consequence

- first considered goal node has minimal cost ✓
- A\* is optimal ✓



# A\* Search and Uniform-Cost Search

Remember uniform-cost search



# A\* Search and Uniform-Cost Search

**Remember** uniform-cost search  
use **priority queue** and sort by cost  $g$



# A\* Search and Uniform-Cost Search

**Remember** uniform-cost search  
use **priority queue** and sort by cost  $g$   
 $\hat{=} f = g$



# A\* Search and Uniform-Cost Search

Remember uniform-cost search  
use **priority queue** and sort by cost  $g$   
 $\hat{f} = g$

Remember A\* search  $\hat{f} = g + h$



# A\* Search and Uniform-Cost Search

**Remember** uniform-cost search  
 use **priority queue** and sort by cost  $g$   
 $\hat{=} f = g$

**Remember** A\* search  $\hat{=} f = g + h$

**Observation**  $(f = g) \equiv (f = g + 0)$



# A\* Search and Uniform-Cost Search

**Remember** uniform-cost search  
 use **priority queue** and sort by cost  $g$   
 $\hat{=} f = g$

**Remember** A\* search  $\hat{=} f = g + h$

**Observation**  $(f = g) \equiv (f = g + 0)$   
 thus uniform-cost search  $\hat{=} \text{A}^*$  search with  $h = 0$   
 (A\* search without heuristic information)

# A\* Search and Uniform-Cost Search

**Remember** uniform-cost search  
 use **priority queue** and sort by cost  $g$   
 $\hat{=} f = g$

**Remember** A\* search  $\hat{=} f = g + h$

**Observation**  $(f = g) \equiv (f = g + 0)$   
 thus uniform-cost search  $\hat{=} A^*$  search with  $h = 0$   
 (A\* search without heuristic information)  
 'without heuristic information'  
 $\hat{=} \text{'with trivial heuristic information } 0\text{'}$

# A\* Search and Uniform-Cost Search

**Remember** uniform-cost search  
use **priority queue** and sort by cost  $g$   
 $\hat{=} f = g$

**Remember** A\* search  $\hat{=} f = g + h$

**Observation**  $(f = g) \equiv (f = g + 0)$   
**thus** uniform-cost search  $\hat{=} \text{A}^*$  search with  $h = 0$   
(A\* search without heuristic information)  
'without heuristic information'  
 $\hat{=} \text{'with trivial heuristic information } 0\text{'}$

**Remember** uniform-cost search is **optimal**  
A\* with consistent heuristic is **optimal**

## A\* Search and Uniform-Cost Search

**Remember** uniform-cost search  
use **priority queue** and sort by cost  $g$   
 $\hat{=} f = g$

**Remember** A\* search  $\hat{=} f = g + h$

**Observation**  $(f = g) \equiv (f = g + 0)$   
**thus** uniform-cost search  $\hat{=} \text{A}^*$  search with  $h = 0$   
(A\* search without heuristic information)  
'without heuristic information'  
 $\hat{=} \text{'with trivial heuristic information } 0\text{'}$

**Remember** uniform-cost search is **optimal**  
A\* with consistent heuristic is **optimal**

Is 0 consistent heuristic information?

## Consistency of 0 Information

**Remember** uniform-cost search is **optimal**  
A\* with consistent heuristic is **optimal**

Is 0 consistent heuristic information?

## Consistency of 0 Information

Remember uniform-cost search is **optimal**  
 A\* with consistent heuristic is **optimal**

Is 0 consistent heuristic information?

Remember  $h$  consistent  
 $\Leftrightarrow \forall s \in \text{state space, applicable action } a,$   
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (**triangle inequality**)

## Consistency of 0 Information

**Remember** uniform-cost search is **optimal**  
 A\* with consistent heuristic is **optimal**

Is 0 consistent heuristic information?

**Remember**  $h$  consistent  
 $\Leftrightarrow \forall s \in \text{state space, applicable action } a,$   
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (**triangle inequality**)

**Observation** with  $h = 0$  we have  $\forall v: h(v) = 0$

## Consistency of 0 Information

**Remember** uniform-cost search is **optimal**  
 A\* with consistent heuristic is **optimal**

Is 0 consistent heuristic information?

**Remember**  $h$  consistent  
 $\Leftrightarrow \forall s \in \text{state space, applicable action } a,$   
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (**triangle inequality**)

**Observation** with  $h = 0$  we have  $\forall v: h(v) = 0$   
 thus We **need**  $0 \leq \text{cost}(a) + 0$

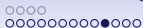
## Consistency of 0 Information

**Remember** uniform-cost search is **optimal**  
 A\* with consistent heuristic is **optimal**

Is 0 consistent heuristic information?

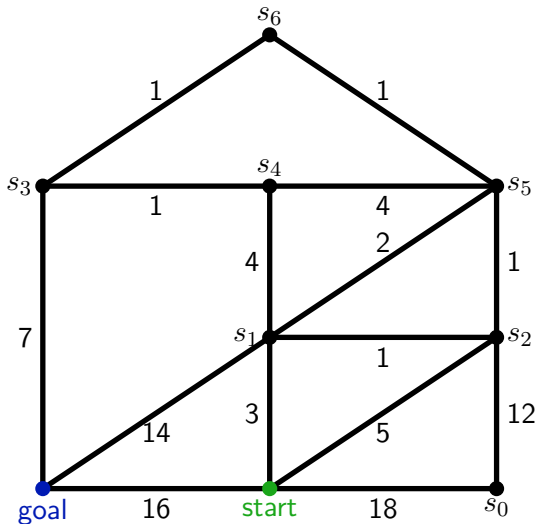
**Remember**  $h$  consistent  
 $\Leftrightarrow \forall s \in \text{state space, applicable action } a,$   
 $s' = \text{transition-model}(s, a): h(s) \leq \text{cost}(a) + h(s')$   
 (triangle inequality)

**Observation** with  $h = 0$  we have  $\forall v: h(v) = 0$   
 thus We **need**  $0 \leq \text{cost}(a) + 0$   
**holds** since we **assume** non-negative cost,  
 $\forall a: \text{cost}(a) \geq 0$  ✓

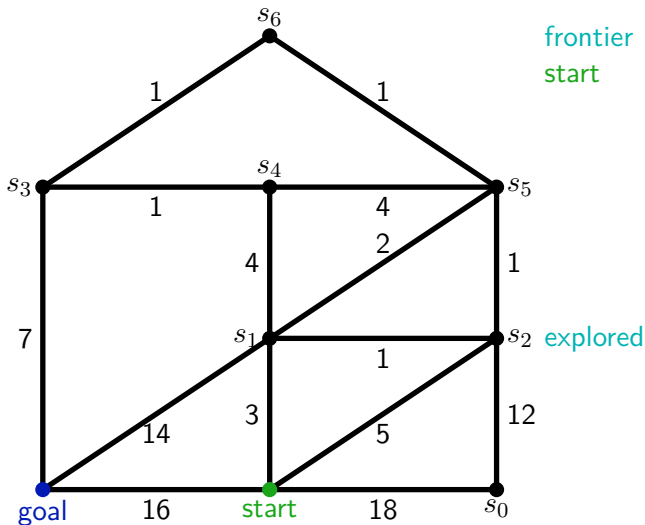


# Example 'Shortest Path' Without Heuristic Information

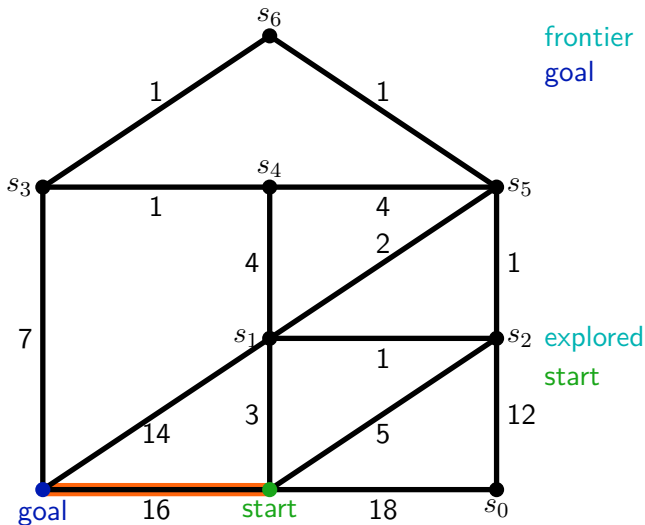
## Example 'Shortest Path' Without Heuristic Information



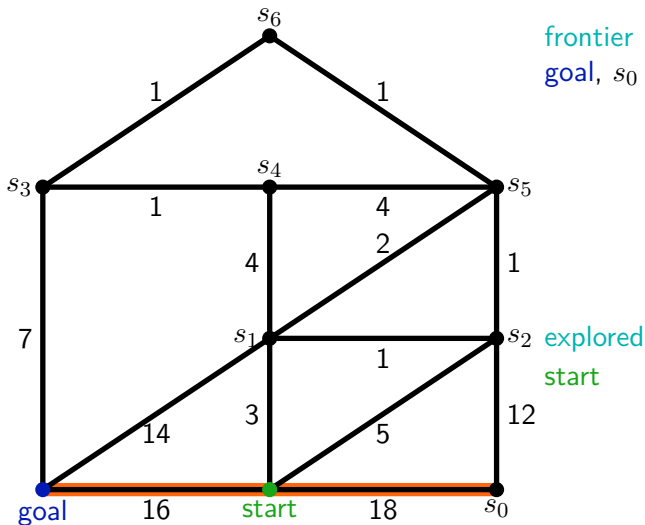
# Example 'Shortest Path' Without Heuristic Information



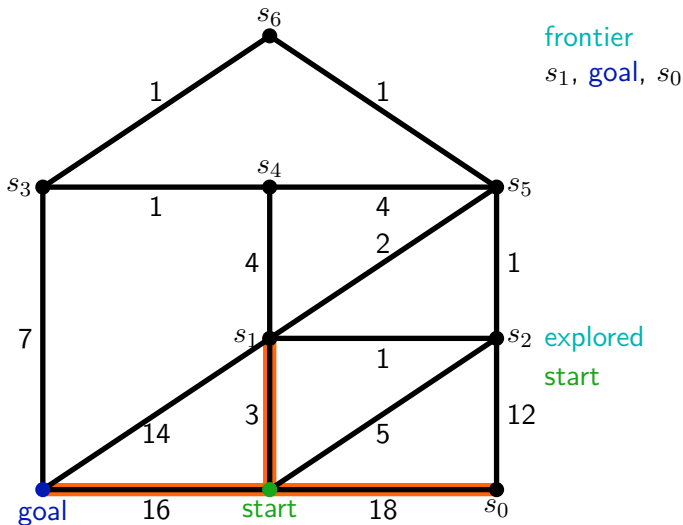
# Example 'Shortest Path' Without Heuristic Information



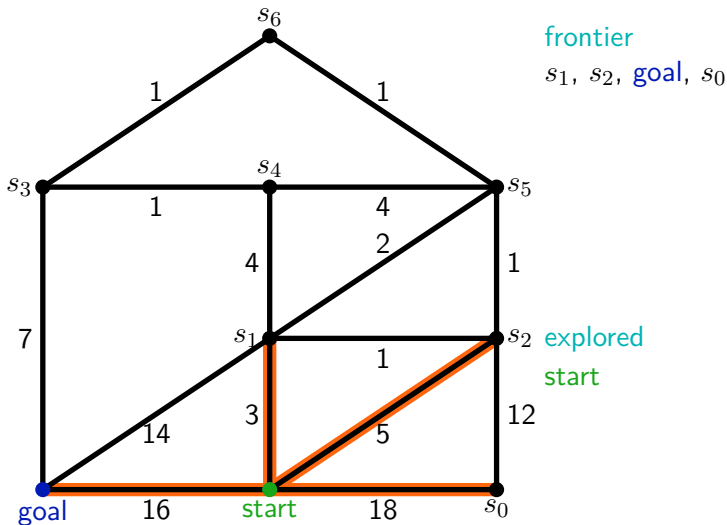
# Example 'Shortest Path' Without Heuristic Information



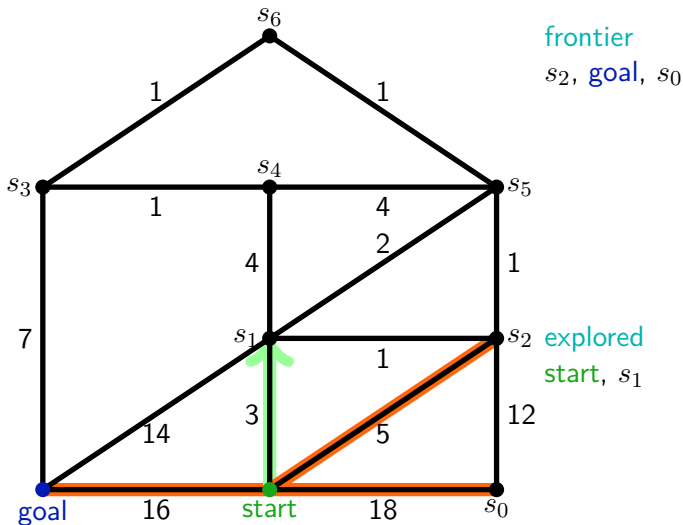
# Example 'Shortest Path' Without Heuristic Information



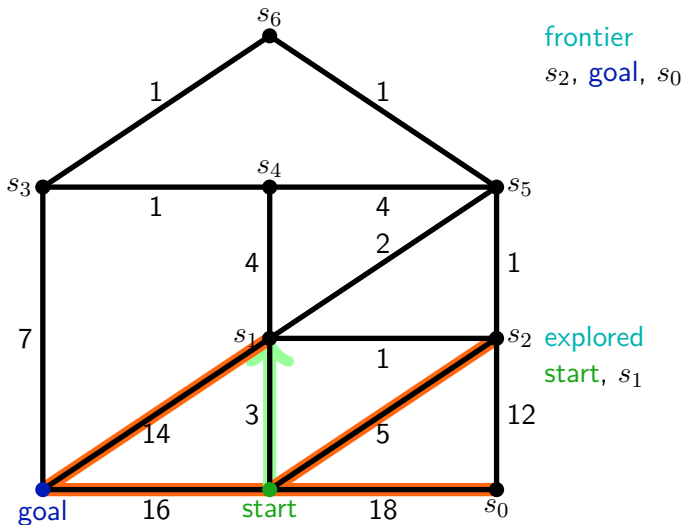
# Example 'Shortest Path' Without Heuristic Information



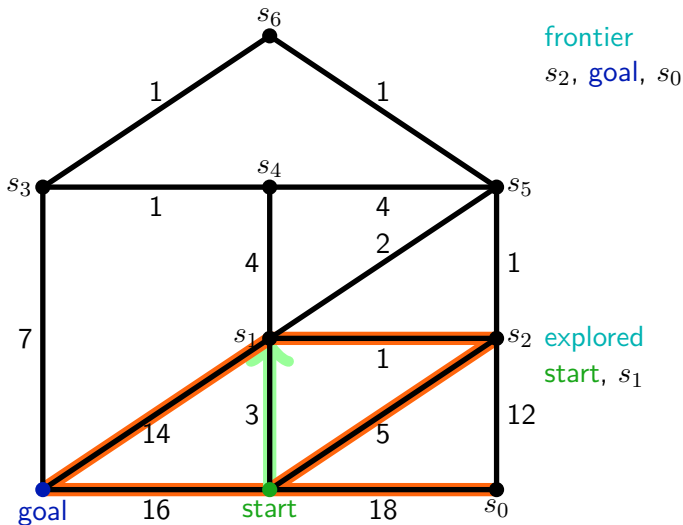
# Example 'Shortest Path' Without Heuristic Information



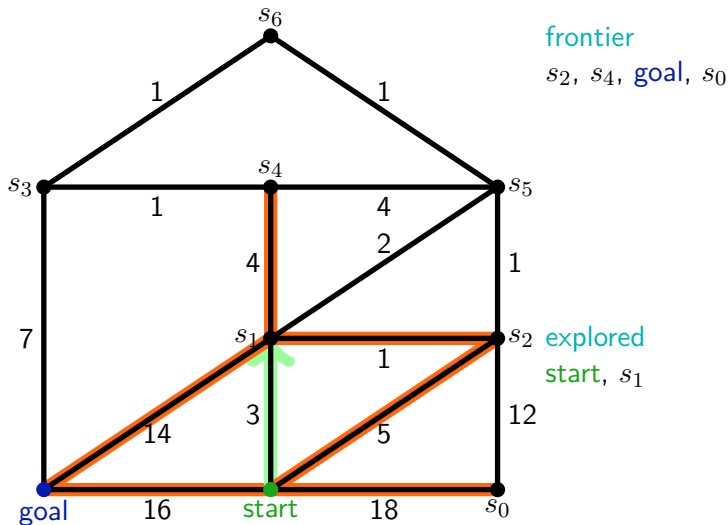
# Example 'Shortest Path' Without Heuristic Information



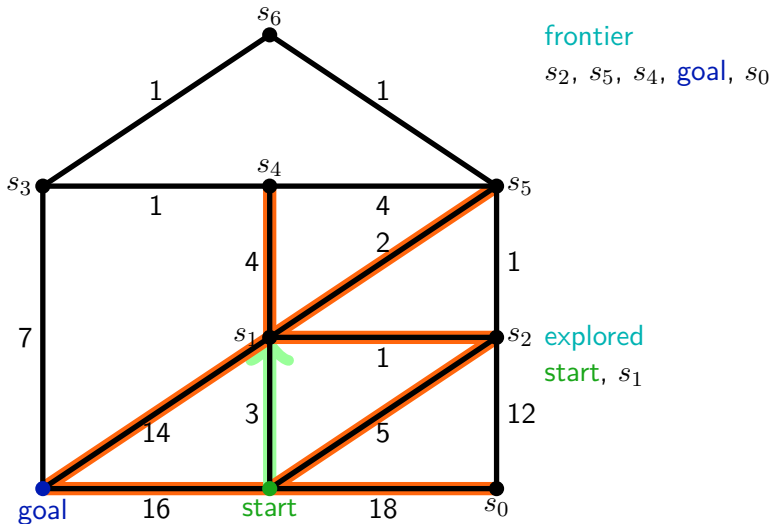
# Example 'Shortest Path' Without Heuristic Information



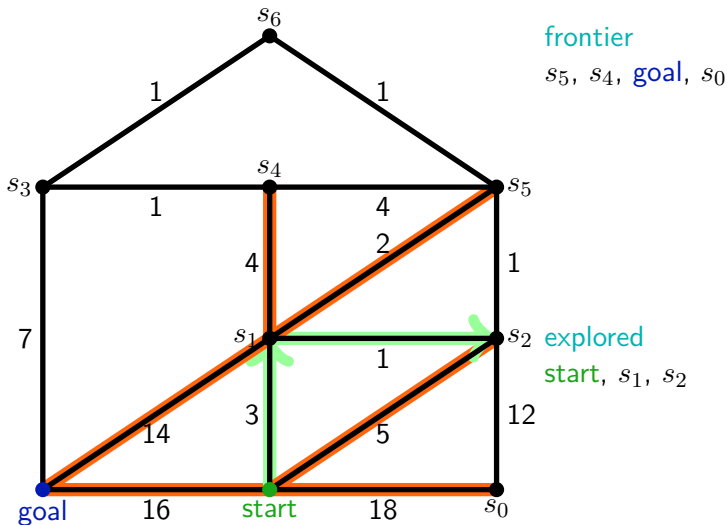
# Example 'Shortest Path' Without Heuristic Information



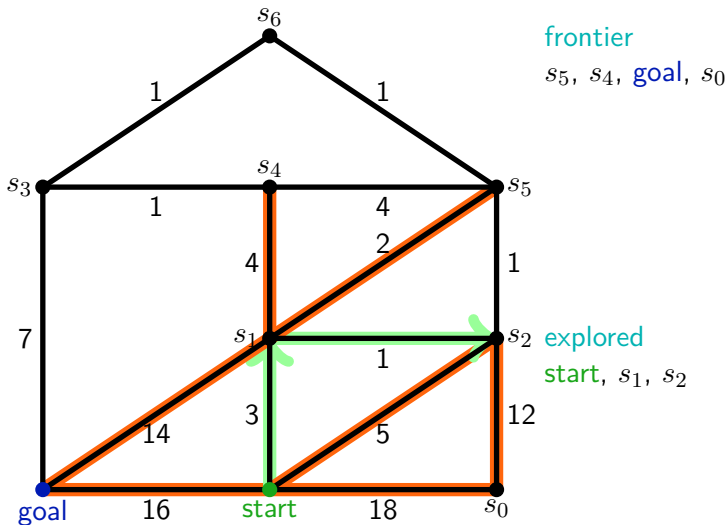
# Example 'Shortest Path' Without Heuristic Information



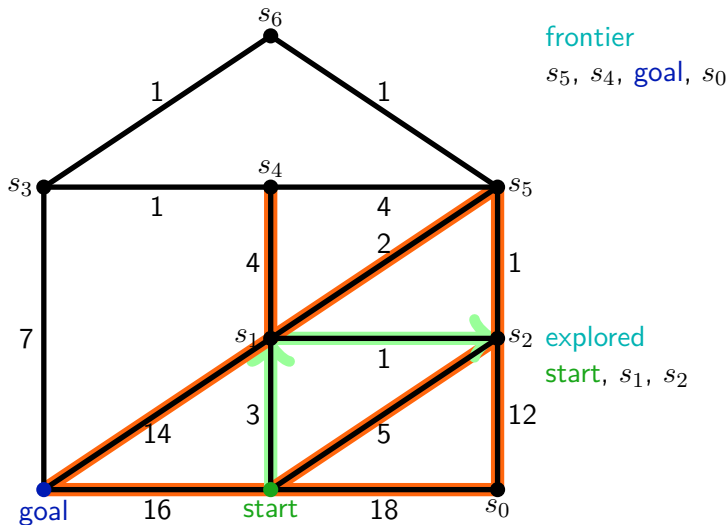
# Example 'Shortest Path' Without Heuristic Information



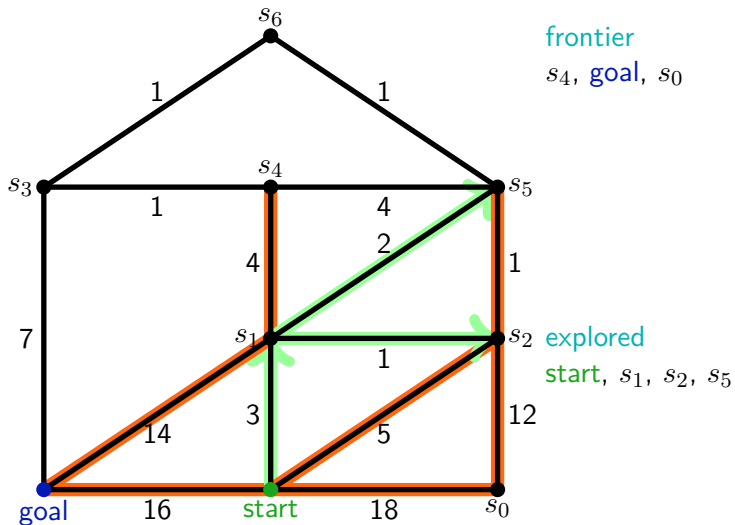
# Example 'Shortest Path' Without Heuristic Information



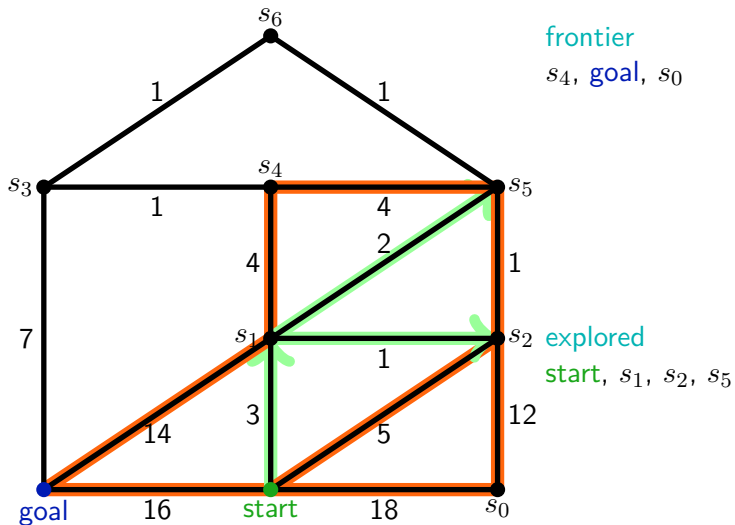
# Example 'Shortest Path' Without Heuristic Information



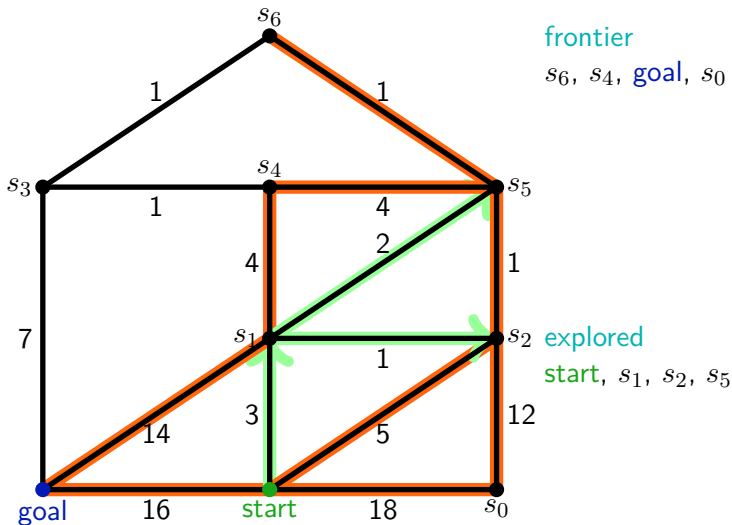
# Example 'Shortest Path' Without Heuristic Information



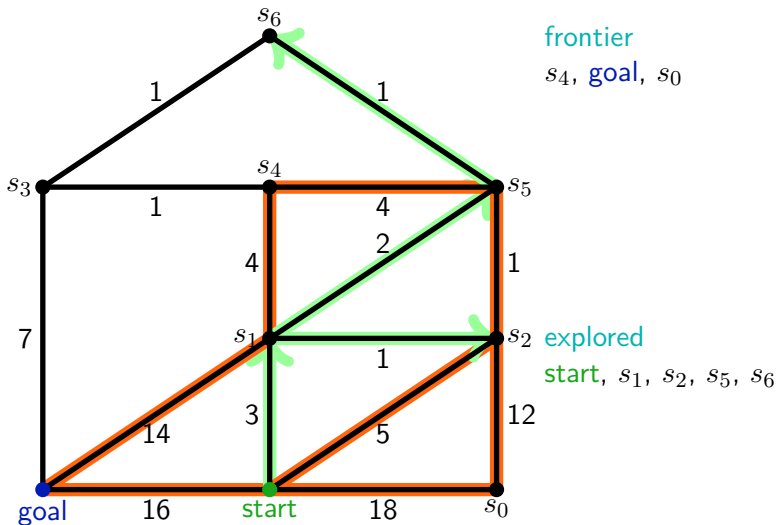
# Example 'Shortest Path' Without Heuristic Information



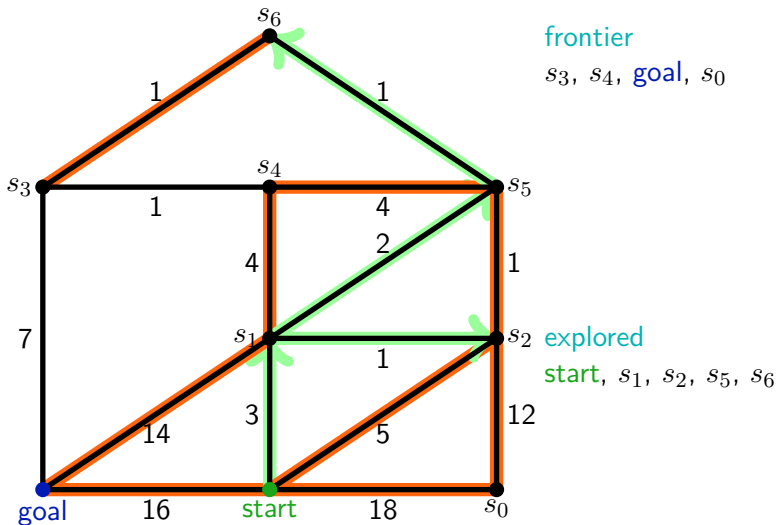
# Example 'Shortest Path' Without Heuristic Information



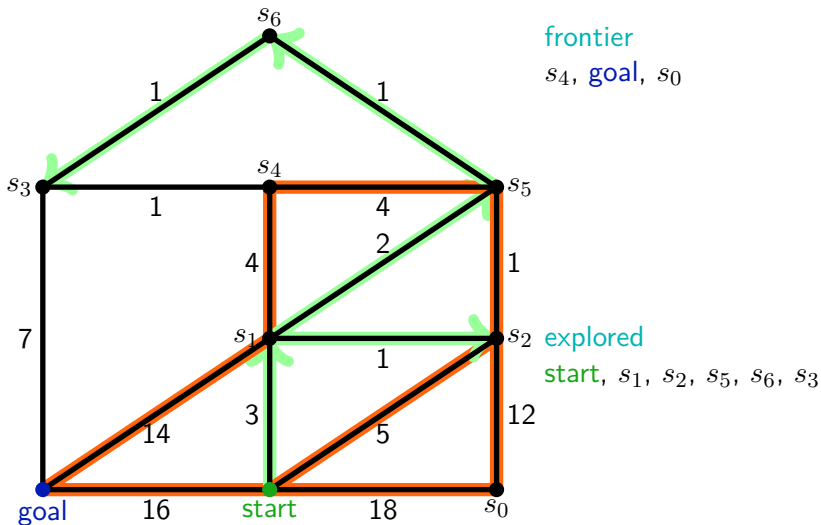
# Example 'Shortest Path' Without Heuristic Information



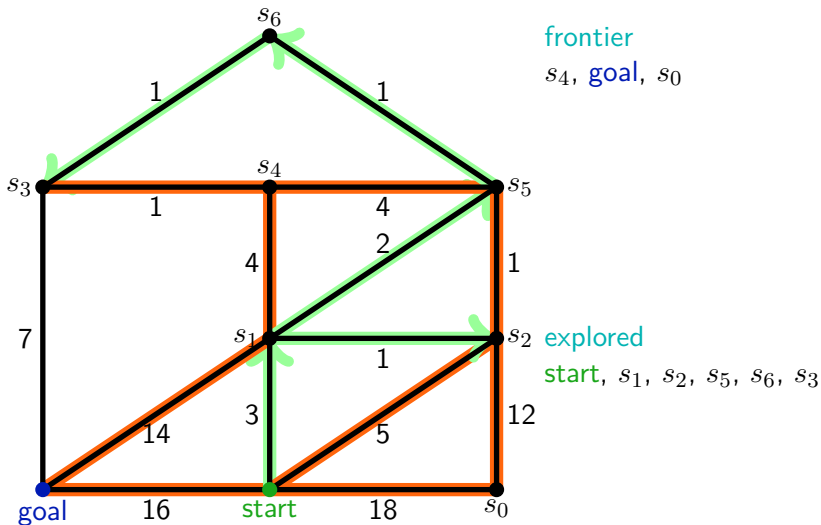
# Example 'Shortest Path' Without Heuristic Information



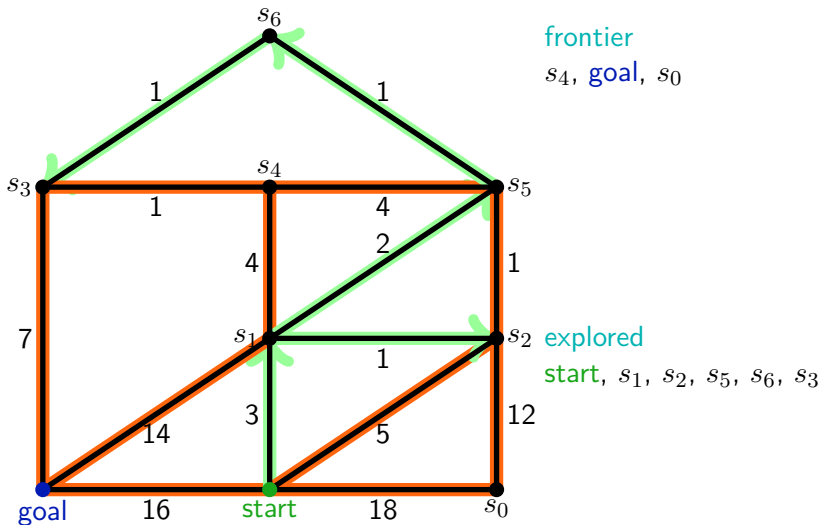
# Example 'Shortest Path' Without Heuristic Information



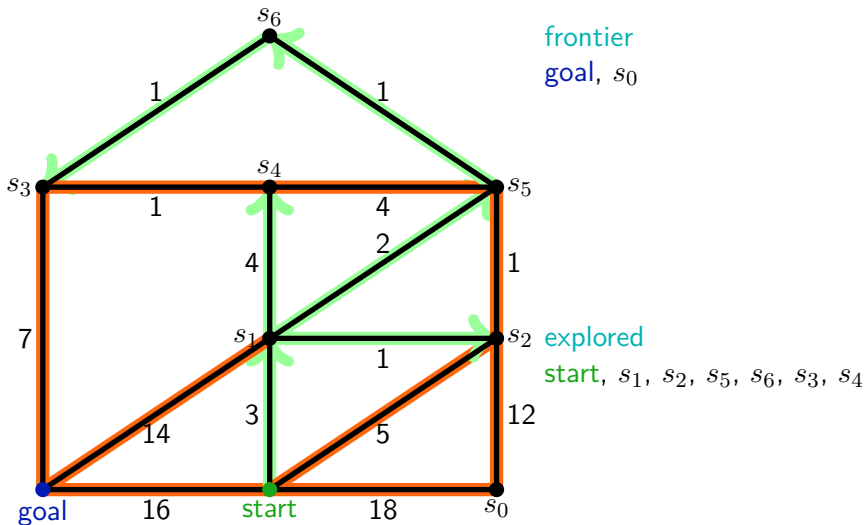
# Example 'Shortest Path' Without Heuristic Information



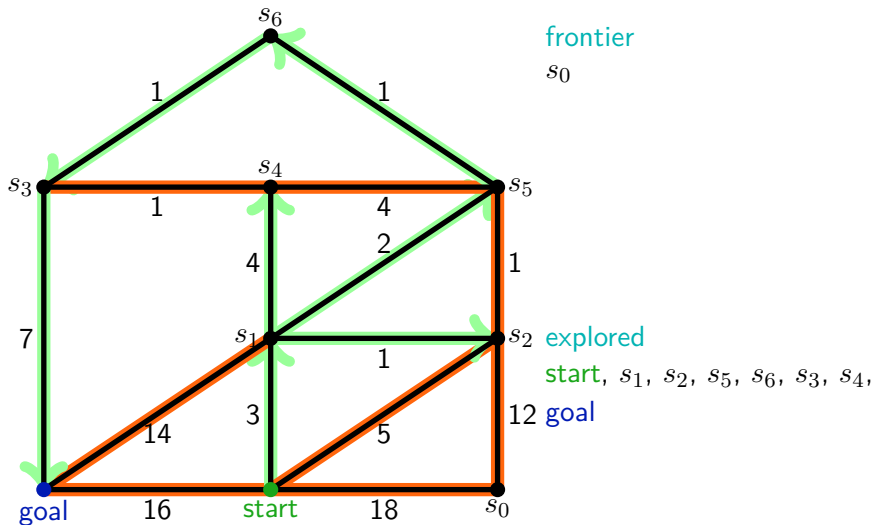
# Example 'Shortest Path' Without Heuristic Information

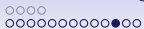


# Example 'Shortest Path' Without Heuristic Information

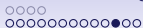


# Example 'Shortest Path' Without Heuristic Information





# Contours in State Space



# Contours in State Space

Consider **metric** in state space  
given by actual minimum path cost



## Contours in State Space

Consider **metric** in state space  
given by actual minimum path cost

**Observation**  $A^*$  search with  $h = 0$  ( $\hat{=}$  uniform-cost search)  
explores state space in **circles** around initial state

## Contours in State Space

Consider **metric** in state space  
given by actual minimum path cost

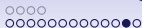
**Observation** A\* search with  $h = 0$  ( $\hat{=}$  uniform-cost search)  
explores state space in **circles** around initial state

**Observation** A\* search with  $h \neq 0$   
explores state space according to  $g + h$   
 $\rightsquigarrow$  circles 'bend' towards promising regions  
(promising according to  $h$ )



## Contours in State Space

- Consider **metric** in state space  
given by actual minimum path cost
- Observation** A\* search with  $h = 0$  ( $\hat{=}$  uniform-cost search)  
explores state space in **circles** around initial state
- Observation** A\* search with  $h \neq 0$   
explores state space according to  $g + h$   
 $\rightsquigarrow$  circles 'bend' towards promising regions  
(promising according to  $h$ )
- Observation** A\* search implements **biased search**  
with search bias defined by  $h$



# Properties of A\* Search

Consider A\* search with consistent  $h$   
in problem with optimal solution with cost  $c_{\text{opt}}$

# Properties of A\* Search

Consider A\* search with consistent  $h$   
in problem with optimal solution with cost  $c_{\text{opt}}$

- A\* is **complete**  
if there are finitely many states with cost  $\leq c_{\text{opt}}$



## Properties of A\* Search

Consider A\* search with consistent  $h$   
in problem with optimal solution with cost  $c_{\text{opt}}$

- A\* is **complete**  
if there are finitely many states with cost  $\leq c_{\text{opt}}$
- A\* is **optimal**  
if it is complete

# Properties of A\* Search

Consider A\* search with consistent  $h$   
in problem with optimal solution with cost  $c_{\text{opt}}$

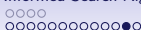
- A\* is **complete**  
if there are finitely many states with cost  $\leq c_{\text{opt}}$
- A\* is **optimal**  
if it is complete
- A\* **prunes** parts of the search space  
it never expands nodes with cost  $> c_{\text{opt}}$



## Properties of A\* Search

Consider A\* search with consistent  $h$   
in problem with optimal solution with cost  $c_{\text{opt}}$

- A\* is **complete**  
if there are finitely many states with cost  $\leq c_{\text{opt}}$
- A\* is **optimal**  
if it is complete
- A\* **prunes** parts of the search space  
it never expands nodes with cost  $> c_{\text{opt}}$
- A\* has **optimal efficiency**  
no complete, optimal search algorithm can guarantee to explore less nodes



## Properties of A\* Search

Consider A\* search with consistent  $h$   
in problem with optimal solution with cost  $c_{\text{opt}}$

- A\* is **complete**  
if there are finitely many states with cost  $\leq c_{\text{opt}}$
- A\* is **optimal**  
if it is complete
- A\* **prunes** parts of the search space  
it never expands nodes with cost  $> c_{\text{opt}}$
- A\* has **optimal efficiency**  
no complete, optimal search algorithm can guarantee to explore less nodes
- A\* is often **too inefficient** to solve problems in practice



## Properties of A\* Search

Consider A\* search with consistent  $h$   
in problem with optimal solution with cost  $c_{\text{opt}}$

- A\* is **complete**  
if there are finitely many states with cost  $\leq c_{\text{opt}}$
- A\* is **optimal**  
if it is complete
- A\* **prunes** parts of the search space  
it never expands nodes with cost  $> c_{\text{opt}}$
- A\* has **optimal efficiency**  
no complete, optimal search algorithm can guarantee to explore less nodes
- A\* is often **too inefficient** to solve problems in practice
- A\* encounters **space issues** before **time issues**  
since it needs to store all expanded nodes

# Memory-Bounded Heuristic Search

**Aim** at solving the space complexity issue of A\* search



## Memory-Bounded Heuristic Search

**Aim** at solving the space complexity issue of  $A^*$  search

- **iterative-deepening  $A^*$  search (IDA\*)**  
use  $f = g + v$ -value as cutoff measure (not depth): cutoff is min.  $f$ -value of a node exceeding the previous cutoff value



# Memory-Bounded Heuristic Search

**Aim** at solving the space complexity issue of  $A^*$  search

- **iterative-deepening  $A^*$  search (IDA\*)**

use  $f = g + v$ -value as cutoff measure (not depth): cutoff is min.  $f$ -value of a node exceeding the previous cutoff value

**problematic** for real-valued functions  $f$

# Memory-Bounded Heuristic Search

**Aim** at solving the space complexity issue of  $A^*$  search

- **iterative-deepening  $A^*$  search (IDA\*)**

use  $f = g + v$ -value as cutoff measure (not depth): cutoff is min.  $f$ -value of a node exceeding the previous cutoff value

**problematic** for real-valued functions  $f$

**re-considers** large parts of the search space many times



## Memory-Bounded Heuristic Search

**Aim** at solving the space complexity issue of  $A^*$  search

- **iterative-deepening  $A^*$  search (IDA\*)**  
 use  $f = g + v$ -value as cutoff measure (not depth): cutoff is min.  $f$ -value of a node exceeding the previous cutoff value  
**problematic** for real-valued functions  $f$   
**re-considers** large parts of the search space many times
- **recursive best-first search (RBFS)**  
 restrict best-first search by value of best-so-far alternative solution, backtrack if limit reached, keep track of branches with potential solutions



## Memory-Bounded Heuristic Search

**Aim** at solving the space complexity issue of  $A^*$  search

- **iterative-deepening  $A^*$  search (IDA\*)**  
use  $f = g + v$ -value as cutoff measure (not depth): cutoff is min.  $f$ -value of a node exceeding the previous cutoff value  
**problematic** for real-valued functions  $f$   
**re-considers** large parts of the search space many times
- **recursive best-first search (RBFS)**  
restrict best-first search by value of best-so-far alternative solution, backtrack if limit reached, keep track of branches with potential solutions  
still **re-considers** large parts of the search space many times

## Memory-Bounded Heuristic Search

**Aim** at solving the space complexity issue of  $A^*$  search

- **iterative-deepening  $A^*$  search (IDA\*)**  
use  $f = g + v$ -value as cutoff measure (not depth): cutoff is min.  $f$ -value of a node exceeding the previous cutoff value  
**problematic** for real-valued functions  $f$   
**re-considers** large parts of the search space many times
- **recursive best-first search (RBFS)**  
restrict best-first search by value of best-so-far alternative solution, backtrack if limit reached, keep track of branches with potential solutions  
still **re-considers** large parts of the search space many times
- **simplified memory-bounded  $A^*$  search (SMA\*)**  
work like  $A^*$  until memory-limit is reached, then discard worst leaf memorising its value, re-evaluate discarded sub-trees if all other sub-trees look worse  
**re-consideration** can increase time complexity drastically



## On Heuristic Functions

**Remember** performance of  $A^*$  search with (consistent) heuristic  $h$  depends very much on  $h$

## On Heuristic Functions

**Remember** performance of  $A^*$  search with (consistent) heuristic  $h$  depends very much on  $h$

Can we characterise the influence of  $h$ ?



## On Heuristic Functions

**Remember** performance of  $A^*$  search with (consistent) heuristic  $h$  depends very much on  $h$

Can we characterise the influence of  $h$ ?

**Remember** example 8-puzzle

start state

7	2	4
5		6
8	3	1

goal state

	1	2
3	4	5
6	7	8

transitions, e. g.,

7	2	4
5		6
8	3	1

 $\xrightarrow{\text{up}}$ 

7	2	4
5	3	6
8		1



# On 8-Puzzle

## General Properties



# On 8-Puzzle

## General Properties

- number of different states

# On 8-Puzzle

## General Properties

- number of different states  $\leq 9! = 362880$



# On 8-Puzzle

## General Properties

- number of different states  $\leq 9! = 362880$   
manageable for any non-repeating search algorithm



# On 8-Puzzle

## General Properties

- number of different states  $\leq 9! = 362880$   
manageable for any non-repeating search algorithm
- average branching factor



# On 8-Puzzle

## General Properties

- number of different states  $\leq 9! = 362\,880$   
manageable for any non-repeating search algorithm
- average branching factor =  $1.4$

# On 8-Puzzle

## General Properties

- number of different states  $\leq 9! = 362880$   
manageable for any non-repeating search algorithm
- average branching factor =  $1 \cdot 4 + 4 \cdot 3$



# On 8-Puzzle

## General Properties

- number of different states  $\leq 9! = 362880$   
manageable for any non-repeating search algorithm
- average branching factor  $= 1 \cdot 4 + 4 \cdot 3 + 4 \cdot 2 = \frac{24}{9} \approx 2.7$

# On 8-Puzzle

## General Properties

- number of different states  $\leq 9! = 362880$   
manageable for any non-repeating search algorithm
- average branching factor =  $1 \cdot 4 + 4 \cdot 3 + 4 \cdot 2 = \frac{24}{9} \approx 2.7$
- comparable to 15-puzzle (well studied, interesting)



# On 8-Puzzle

## General Properties

- number of different states  $\leq 9! = 362880$   
manageable for any non-repeating search algorithm
- average branching factor =  $1 \cdot 4 + 4 \cdot 3 + 4 \cdot 2 = \frac{24}{9} \approx 2.7$
- comparable to 15-puzzle (well studied, interesting)

**Idea** empirically compare effects of different heuristics on A\*'s search behaviour

# On 8-Puzzle

## General Properties

- number of different states  $\leq 9! = 362880$   
manageable for any non-repeating search algorithm
- average branching factor  $= 1 \cdot 4 + 4 \cdot 3 + 4 \cdot 2 = \frac{24}{9} \approx 2.7$
- comparable to 15-puzzle (well studied, interesting)

**Idea** empirically compare effects of different heuristics on A\*'s search behaviour

**Need** at least two different heuristic functions  $h_1, h_2$  preferably consistent



# Heuristics for A\* Search

# Heuristics for A\* Search

- $h_1(v) = \#$ misplaced tiles

# Heuristics for A\* Search

- $h_1(v) = \#$ misplaced tiles  
Admissible?



## Heuristics for A\* Search

- $h_1(v) = \#$ misplaced tiles

Admissible?

Remember **yes**, if never overestimating

## Heuristics for A\* Search

- $h_1(v) = \#$ misplaced tiles

Admissible?

Remember **yes**, if never overestimating

**yes**, since per misplaced tile at least one move necessary ✓

## Heuristics for A\* Search

- $h_1(v) = \#$ misplaced tiles

Admissible?

Remember **yes**, if never overestimating

**yes**, since per misplaced tile at least one move necessary ✓

Consistent?

## Heuristics for A\* Search

- $h_1(v) = \#$ misplaced tiles

Admissible?

Remember **yes**, if never overestimating

**yes**, since per misplaced tile at least one move necessary ✓

Consistent?

Remember **yes**, if  $h(s) \leq h(\text{transition-model}(s, a)) + \text{cost}(a)$

## Heuristics for A\* Search

- $h_1(v) = \#$ misplaced tiles

Admissible?

Remember **yes**, if never overestimating

**yes**, since per misplaced tile at least one move necessary ✓

Consistent?

Remember **yes**, if  $h(s) \leq h(\text{transition-model}(s, a)) + \text{cost}(a)$

**yes**, since each move changes number of misplaced tiles by

$\leq 1$  and  $\forall a: \text{cost}(a) = 1$  ✓



## Heuristics for A\* Search

- $h_1(v) = \#$ misplaced tiles

Admissible?

Remember **yes**, if never overestimating

**yes**, since per misplaced tile at least one move necessary ✓

Consistent?

Remember **yes**, if  $h(s) \leq h(\text{transition-model}(s, a)) + \text{cost}(a)$

**yes**, since each move changes number of misplaced tiles by

$\leq 1$  and  $\forall a: \text{cost}(a) = 1$  ✓

- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$



## Heuristics for A\* Search

- $h_1(v) = \#$ misplaced tiles

Admissible?

Remember **yes**, if never overestimating

**yes**, since per misplaced tile at least one move necessary ✓

Consistent?

Remember **yes**, if  $h(s) \leq h(\text{transition-model}(s, a)) + \text{cost}(a)$

**yes**, since each move changes number of misplaced tiles by  $\leq 1$  and  $\forall a: \text{cost}(a) = 1$  ✓

- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

Admissible?

## Heuristics for A\* Search

- $h_1(v) = \#$ misplaced tiles

Admissible?

Remember **yes**, if never overestimating

**yes**, since per misplaced tile at least one move necessary ✓

Consistent?

Remember **yes**, if  $h(s) \leq h(\text{transition-model}(s, a)) + \text{cost}(a)$

**yes**, since each move changes number of misplaced tiles by  $\leq 1$  and  $\forall a: \text{cost}(a) = 1$  ✓

- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

Admissible? **yes**, since each move decreases Manhattan distance by at most 1 ✓

## Heuristics for A\* Search

- $h_1(v) = \#$ misplaced tiles

Admissible?

Remember **yes**, if never overestimating

**yes**, since per misplaced tile at least one move necessary ✓

Consistent?

Remember **yes**, if  $h(s) \leq h(\text{transition-model}(s, a)) + \text{cost}(a)$

**yes**, since each move changes number of misplaced tiles by  $\leq 1$  and  $\forall a: \text{cost}(a) = 1$  ✓

- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

Admissible? **yes**, since each move decreases Manhattan distance by at most 1 ✓

Consistent?

## Heuristics for A\* Search

- $h_1(v) = \#$ misplaced tiles

Admissible?

Remember yes, if never overestimating

yes, since per misplaced tile at least one move necessary ✓

Consistent?

Remember yes, if  $h(s) \leq h(\text{transition-model}(s, a)) + \text{cost}(a)$

yes, since each move changes number of misplaced tiles by  $\leq 1$  and  $\forall a: \text{cost}(a) = 1$  ✓

- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

Admissible? yes, since each move decreases Manhattan distance by at most 1 ✓

Consistent? yes, since each move changes Manhattan distance by  $\leq 1$  and  $\forall a: \text{cost}(a) = 1$  ✓



# Applying Heuristics $h_1$ and $h_2$

## Remember

- $h_1(v) = \#$ misplaced tiles
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$



## Applying Heuristics $h_1$ and $h_2$

### Remember

- $h_1(v) = \# \text{misplaced tiles}$
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

### Example

- $h_1 \left( \begin{array}{|c|c|c|} \hline 7 & 2 & 4 \\ \hline 5 & & 6 \\ \hline 8 & 3 & 1 \\ \hline \end{array} \right) =$



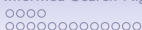
## Applying Heuristics $h_1$ and $h_2$

### Remember

- $h_1(v) = \# \text{misplaced tiles}$
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

### Example

- $h_1 \left( \begin{array}{|c|c|c|} \hline 7 & 2 & 4 \\ \hline 5 & & 6 \\ \hline 8 & 3 & 1 \\ \hline \end{array} \right) = 8$



# Applying Heuristics $h_1$ and $h_2$

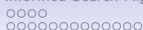
## Remember

- $h_1(v) = \# \text{misplaced tiles}$
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

## Example

- $h_1 \left( \begin{pmatrix} 7 & 2 & 4 \\ 5 & & 6 \\ 8 & 3 & 1 \end{pmatrix} \right) = 8$

- $h_2 \left( \begin{pmatrix} 7 & 2 & 4 \\ 5 & & 6 \\ 8 & 3 & 1 \end{pmatrix} \right) =$



# Applying Heuristics $h_1$ and $h_2$

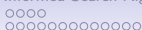
## Remember

- $h_1(v) = \# \text{misplaced tiles}$
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

## Example

- $h_1 \left( \begin{pmatrix} 7 & 2 & 4 \\ 5 & & 6 \\ 8 & 3 & 1 \end{pmatrix} \right) = 8$

- $h_2 \left( \begin{pmatrix} 7 & 2 & 4 \\ 5 & & 6 \\ 8 & 3 & 1 \end{pmatrix} \right) = 3$



## Applying Heuristics $h_1$ and $h_2$

### Remember

- $h_1(v) = \# \text{misplaced tiles}$
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

### Example

- $h_1 \left( \begin{pmatrix} 7 & 2 & 4 \\ 5 & & 6 \\ 8 & 3 & 1 \end{pmatrix} \right) = 8$

- $h_2 \left( \begin{pmatrix} 7 & 2 & 4 \\ 5 & & 6 \\ 8 & 3 & 1 \end{pmatrix} \right) = 3 + 1$



# Applying Heuristics $h_1$ and $h_2$

## Remember

- $h_1(v) = \# \text{misplaced tiles}$
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

## Example

- $h_1 \left( \begin{pmatrix} 7 & 2 & 4 \\ 5 & & 6 \\ 8 & 3 & 1 \end{pmatrix} \right) = 8$

- $h_2 \left( \begin{pmatrix} 7 & 2 & 4 \\ 5 & & 6 \\ 8 & 3 & 1 \end{pmatrix} \right) = 3 + 1 + 2$



# Applying Heuristics $h_1$ and $h_2$

## Remember

- $h_1(v) = \# \text{misplaced tiles}$
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

## Example

- $h_1 \left( \begin{pmatrix} 7 & 2 & 4 \\ 5 & & 6 \\ 8 & 3 & 1 \end{pmatrix} \right) = 8$

- $h_2 \left( \begin{pmatrix} 7 & 2 & 4 \\ 5 & & 6 \\ 8 & 3 & 1 \end{pmatrix} \right) = 3 + 1 + 2 + 2$



## Applying Heuristics $h_1$ and $h_2$

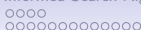
### Remember

- $h_1(v) = \#$ misplaced tiles
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

### Example

- $h_1 \left( \begin{array}{|c|c|c|} \hline 7 & 2 & 4 \\ \hline 5 & & 6 \\ \hline 8 & 3 & 1 \\ \hline \end{array} \right) = 8$

- $h_2 \left( \begin{array}{|c|c|c|} \hline 7 & 2 & 4 \\ \hline 5 & & 6 \\ \hline 8 & 3 & 1 \\ \hline \end{array} \right) = 3 + 1 + 2 + 2 + 2$



# Applying Heuristics $h_1$ and $h_2$

## Remember

- $h_1(v) = \#$ misplaced tiles
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

## Example

- $h_1 \left( \begin{array}{|c|c|c|} \hline 7 & 2 & 4 \\ \hline 5 & & 6 \\ \hline 8 & 3 & 1 \\ \hline \end{array} \right) = 8$

- $h_2 \left( \begin{array}{|c|c|c|} \hline 7 & 2 & 4 \\ \hline 5 & & 6 \\ \hline 8 & 3 & 1 \\ \hline \end{array} \right) = 3 + 1 + 2 + 2 + 2 + 3$



# Applying Heuristics $h_1$ and $h_2$

## Remember

- $h_1(v) = \# \text{misplaced tiles}$
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

## Example

- $h_1 \left( \begin{pmatrix} 7 & 2 & 4 \\ 5 & & 6 \\ 8 & 3 & 1 \end{pmatrix} \right) = 8$

- $h_2 \left( \begin{pmatrix} 7 & 2 & 4 \\ 5 & & 6 \\ 8 & 3 & 1 \end{pmatrix} \right) = 3 + 1 + 2 + 2 + 2 + 3 + 3$



## Applying Heuristics $h_1$ and $h_2$

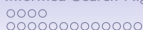
### Remember

- $h_1(v) = \# \text{misplaced tiles}$
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

### Example

- $h_1 \left( \begin{array}{|c|c|c|} \hline 7 & 2 & 4 \\ \hline 5 & & 6 \\ \hline 8 & 3 & 1 \\ \hline \end{array} \right) = 8$

- $h_2 \left( \begin{array}{|c|c|c|} \hline 7 & 2 & 4 \\ \hline 5 & & 6 \\ \hline 8 & 3 & 1 \\ \hline \end{array} \right) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2$



# Applying Heuristics $h_1$ and $h_2$

## Remember

- $h_1(v) = \# \text{misplaced tiles}$
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

## Example

- $h_1 \left( \begin{array}{|c|c|c|} \hline 7 & 2 & 4 \\ \hline 5 & & 6 \\ \hline 8 & 3 & 1 \\ \hline \end{array} \right) = 8$

- $h_2 \left( \begin{array}{|c|c|c|} \hline 7 & 2 & 4 \\ \hline 5 & & 6 \\ \hline 8 & 3 & 1 \\ \hline \end{array} \right) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$

# Applying Heuristics $h_1$ and $h_2$

## Remember

- $h_1(v) = \# \text{misplaced tiles}$
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

## Example

- $h_1 \left( \begin{array}{|c|c|c|} \hline 7 & 2 & 4 \\ \hline 5 & & 6 \\ \hline 8 & 3 & 1 \\ \hline \end{array} \right) = 8$

- $h_2 \left( \begin{array}{|c|c|c|} \hline 7 & 2 & 4 \\ \hline 5 & & 6 \\ \hline 8 & 3 & 1 \\ \hline \end{array} \right) = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2 = 18$

- **Info** true cost  $\left( \begin{array}{|c|c|c|} \hline 7 & 2 & 4 \\ \hline 5 & & 6 \\ \hline 8 & 3 & 1 \\ \hline \end{array} \right) = 26$



# Evaluating Heuristics for 8-Puzzle

# Evaluating Heuristics for 8-Puzzle

What do we compare?



# Evaluating Heuristics for 8-Puzzle

## What do we compare?

- number of generated nodes (for solution at given depth)



# Evaluating Heuristics for 8-Puzzle

## What do we compare?

- number of generated nodes (for solution at given depth)
- effective branching factor (for solution at given depth)  
for a uniform search tree with  $n + 1$  nodes with branching factor  $b$  we have

$$n + 1 = \sum_{i=0}^d b^i$$



# Evaluating Heuristics for 8-Puzzle

## What do we compare?

- number of generated nodes (for solution at given depth)
- effective branching factor (for solution at given depth)  
for a uniform search tree with  $n + 1$  nodes with branching factor  $b$  we have

$$n + 1 = \sum_{i=0}^d b^i = (b^{d+1} - 1) / (b - 1)$$

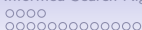
# Evaluating Heuristics for 8-Puzzle

## What do we compare?

- number of generated nodes (for solution at given depth)
  - effective branching factor (for solution at given depth)
- for a uniform search tree with  $n + 1$  nodes with branching factor  $b$  we have

$$n + 1 = \sum_{i=0}^d b^i = (b^{d+1} - 1) / (b - 1)$$

$d$	number of generated nodes			effective branching factor		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$



## Evaluating Heuristics for 8-Puzzle

### What do we compare?

- number of generated nodes (for solution at given depth)
- effective branching factor (for solution at given depth)  
for a uniform search tree with  $n + 1$  nodes with branching factor  $b$  we have

$$n + 1 = \sum_{i=0}^d b^i = (b^{d+1} - 1) / (b - 1)$$

$d$	number of generated nodes			effective branching factor		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79



# Evaluating Heuristics for 8-Puzzle

## What do we compare?

- number of generated nodes (for solution at given depth)
- effective branching factor (for solution at given depth)  
for a uniform search tree with  $n + 1$  nodes with branching factor  $b$  we have

$$n + 1 = \sum_{i=0}^d b^i = (b^{d+1} - 1) / (b - 1)$$

$d$	number of generated nodes			effective branching factor		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45



# Evaluating Heuristics for 8-Puzzle

## What do we compare?

- number of generated nodes (for solution at given depth)
  - effective branching factor (for solution at given depth)
- for a uniform search tree with  $n + 1$  nodes with branching factor  $b$  we have

$$n + 1 = \sum_{i=0}^d b^i = (b^{d+1} - 1) / (b - 1)$$

$d$	number of generated nodes			effective branching factor		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30



# Evaluating Heuristics for 8-Puzzle

## What do we compare?

- number of generated nodes (for solution at given depth)
- effective branching factor (for solution at given depth)  
for a uniform search tree with  $n + 1$  nodes with branching factor  $b$  we have

$$n + 1 = \sum_{i=0}^d b^i = (b^{d+1} - 1) / (b - 1)$$

$d$	number of generated nodes			effective branching factor		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24



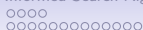
# Evaluating Heuristics for 8-Puzzle

## What do we compare?

- number of generated nodes (for solution at given depth)
  - effective branching factor (for solution at given depth)
- for a uniform search tree with  $n + 1$  nodes with branching factor  $b$  we have

$$n + 1 = \sum_{i=0}^d b^i = (b^{d+1} - 1) / (b - 1)$$

$d$	number of generated nodes			effective branching factor		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6 384	39	25	2.80	1.33	1.24
10	47 127	93	39	2.79	1.38	1.22



# Evaluating Heuristics for 8-Puzzle

## What do we compare?

- number of generated nodes (for solution at given depth)
  - effective branching factor (for solution at given depth)
- for a uniform search tree with  $n + 1$  nodes with branching factor  $b$  we have

$$n + 1 = \sum_{i=0}^d b^i = (b^{d+1} - 1) / (b - 1)$$

$d$	number of generated nodes			effective branching factor		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6 384	39	25	2.80	1.33	1.24
10	47 127	93	39	2.79	1.38	1.22
12	3 644 035	227	73	2.78	1.42	1.24



# Evaluating Heuristics for 8-Puzzle

## What do we compare?

- number of generated nodes (for solution at given depth)
  - effective branching factor (for solution at given depth)
- for a uniform search tree with  $n + 1$  nodes with branching factor  $b$  we have

$$n + 1 = \sum_{i=0}^d b^i = (b^{d+1} - 1) / (b - 1)$$

$d$	number of generated nodes			effective branching factor		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6 384	39	25	2.80	1.33	1.24
10	47 127	93	39	2.79	1.38	1.22
12	3 644 035	227	73	2.78	1.42	1.24

Observation  $h_2$  dominates  $h_1$



## Domination of Heuristics

Remember  $h_2$  dominates  $h_1$

$$h_1(v) = \# \text{misplaced tiles}$$

$$h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$$



## Domination of Heuristics

Remember  $h_2$  dominates  $h_1$

$$h_1(v) = \# \text{misplaced tiles}$$

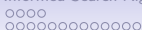
$$h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$$

Remember  $A^*$  never expands nodes  $v$  with  $f(v) > c_{\text{opt}}$



## Domination of Heuristics

- Remember**  $h_2$  **dominates**  $h_1$
- $h_1(v) = \# \text{misplaced tiles}$
- $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$
- Remember**  $A^*$  never expands nodes  $v$  with  $f(v) > c_{\text{opt}}$
- $A^*$  expands each node  $v$  with  $f(v) < c_{\text{opt}}$



## Domination of Heuristics

**Remember**  $h_2$  **dominates**  $h_1$

$h_1(v) = \# \text{misplaced tiles}$

$h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

**Remember**  $A^*$  never expands nodes  $v$  with  $f(v) > c_{\text{opt}}$

$A^*$  expands each node  $v$  with  $f(v) < c_{\text{opt}}$

$A^*$  expands each node  $v$  with  $g(v) + h(v) < c_{\text{opt}}$

## Domination of Heuristics

**Remember**  $h_2$  **dominates**  $h_1$

$h_1(v) = \# \text{misplaced tiles}$

$h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

**Remember**  $A^*$  never expands nodes  $v$  with  $f(v) > c_{\text{opt}}$

$A^*$  expands each node  $v$  with  $f(v) < c_{\text{opt}}$

$A^*$  expands each node  $v$  with  $g(v) + h(v) < c_{\text{opt}}$

$A^*$  expands each node  $v$  with  $h(v) < c_{\text{opt}} - g(v)$



## Domination of Heuristics

**Remember**  $h_2$  **dominates**  $h_1$   
 $h_1(v) = \# \text{misplaced tiles}$   
 $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

**Remember**  $A^*$  never expands nodes  $v$  with  $f(v) > c_{\text{opt}}$   
 $A^*$  expands each node  $v$  with  $f(v) < c_{\text{opt}}$   
 $A^*$  expands each node  $v$  with  $g(v) + h(v) < c_{\text{opt}}$   
 $A^*$  expands each node  $v$  with  $h(v) < c_{\text{opt}} - g(v)$

**Observation**  $\forall v: h_2(v) \geq h_1(v)$

## Domination of Heuristics

**Remember**  $h_2$  **dominates**  $h_1$   
 $h_1(v) = \# \text{misplaced tiles}$   
 $h_2(v) = \sum_{i=1}^8 \text{Manhattan distance}(\text{tile } i, \text{goal position})$

**Remember**  $A^*$  never expands nodes  $v$  with  $f(v) > c_{\text{opt}}$   
 $A^*$  expands each node  $v$  with  $f(v) < c_{\text{opt}}$   
 $A^*$  expands each node  $v$  with  $g(v) + h(v) < c_{\text{opt}}$   
 $A^*$  expands each node  $v$  with  $h(v) < c_{\text{opt}} - g(v)$

**Observation**  $\forall v: h_2(v) \geq h_1(v)$

**Consequence**  $A^*(h_2)$  will not explore more nodes than  $A^*(h_1)$   
 (and most likely much less)  
 $\hat{=} h_2$  **dominates**  $h_1$



# Summary & Take Home Message

Things to remember



# Summary & Take Home Message

## Things to remember

- informed search: evaluation f.  $f$ , path cost  $g$ , heuristic f.  $h$



# Summary & Take Home Message

## Things to remember

- informed search: evaluation f.  $f$ , path cost  $g$ , heuristic f.  $h$
- greedy best-first search ( $f = h$ ), A\* search ( $f = g + h$ )



# Summary & Take Home Message

## Things to remember

- informed search: evaluation f.  $f$ , path cost  $g$ , heuristic f.  $h$
- greedy best-first search ( $f = h$ ), A\* search ( $f = g + h$ )
- heuristic functions: admissibility (no overestimation), consistency (triangle inequality)



# Summary & Take Home Message

## Things to remember

- informed search: evaluation  $f$ ,  $f$ , path cost  $g$ , heuristic  $f$ ,  $h$
- greedy best-first search ( $f = h$ ), A\* search ( $f = g + h$ )
- heuristic functions: admissibility (no overestimation), consistency (triangle inequality)
- A\* variants: iterative-deepening A\* (IDA\*), recursive best-first (RBFS), simplified memory-bounded A\* (SMA\*)



# Summary & Take Home Message

## Things to remember

- informed search: evaluation  $f$ ,  $f$ , path cost  $g$ , heuristic  $f$ .  $h$
- greedy best-first search ( $f = h$ ), A\* search ( $f = g + h$ )
- heuristic functions: admissibility (no overestimation), consistency (triangle inequality)
- A\* variants: iterative-deepening A\* (IDA\*), recursive best-first (RBFS), simplified memory-bounded A\* (SMA\*)
- heuristics for A\*: domination



# Summary & Take Home Message

## Things to remember

- informed search: evaluation  $f$ ,  $f$ , path cost  $g$ , heuristic  $f$ ,  $h$
- greedy best-first search ( $f = h$ ), A\* search ( $f = g + h$ )
- heuristic functions: admissibility (no overestimation), consistency (triangle inequality)
- A\* variants: iterative-deepening A\* (IDA\*), recursive best-first (RBFS), simplified memory-bounded A\* (SMA\*)
- heuristics for A\*: domination

## Take Home Message



## Summary & Take Home Message

### Things to remember

- informed search: evaluation  $f$ ,  $f$ , path cost  $g$ , heuristic  $f$ .  $h$
- greedy best-first search ( $f = h$ ), A\* search ( $f = g + h$ )
- heuristic functions: admissibility (no overestimation), consistency (triangle inequality)
- A\* variants: iterative-deepening A\* (IDA\*), recursive best-first (RBFS), simplified memory-bounded A\* (SMA\*)
- heuristics for A\*: domination

### Take Home Message

- Informed search can hugely excel over uninformed search.



# Summary & Take Home Message

## Things to remember

- informed search: evaluation  $f$ ,  $f$ , path cost  $g$ , heuristic  $f$ .  $h$
- greedy best-first search ( $f = h$ ), A\* search ( $f = g + h$ )
- heuristic functions: admissibility (no overestimation), consistency (triangle inequality)
- A\* variants: iterative-deepening A\* (IDA\*), recursive best-first (RBFS), simplified memory-bounded A\* (SMA\*)
- heuristics for A\*: domination

## Take Home Message

- Informed search can hugely excel over uninformed search.
- A\* search is very powerful but not always practical.

# Summary & Take Home Message

## Things to remember

- informed search: evaluation  $f$ ,  $f$ , path cost  $g$ , heuristic  $f$ .  $h$
- greedy best-first search ( $f = h$ ), A\* search ( $f = g + h$ )
- heuristic functions: admissibility (no overestimation), consistency (triangle inequality)
- A\* variants: iterative-deepening A\* (IDA\*), recursive best-first (RBFS), simplified memory-bounded A\* (SMA\*)
- heuristics for A\*: domination

## Take Home Message

- Informed search can hugely excel over uninformed search.
- A\* search is very powerful but not always practical.
- Variants of A\* search tackling some of A\*'s problems exist.

# Summary & Take Home Message

## Things to remember

- informed search: evaluation f.  $f$ , path cost  $g$ , heuristic f.  $h$
- greedy best-first search ( $f = h$ ), A\* search ( $f = g + h$ )
- heuristic functions: admissibility (no overestimation), consistency (triangle inequality)
- A\* variants: iterative-deepening A\* (IDA\*), recursive best-first (RBFS), simplified memory-bounded A\* (SMA\*)
- heuristics for A\*: domination

## Take Home Message

- Informed search can hugely excel over uninformed search.
- A\* search is very powerful but not always practical.
- Variants of A\* search tackling some of A\*'s problems exist.
- Heuristics are crucial for the performance of A\* search.