



# CS 540 Introduction to Artificial Intelligence

## **Search II: Informed Search**

University of Wisconsin-Madison  
Spring 2026 Sections 1 & 2

# Announcements

- **Homework:**

- HW7 due tomorrow, **April 8<sup>th</sup> at 11:59 PM**
- HW8 will also be released tomorrow

- **Class roadmap:**

Search II: Informed search

Game – Part I

Game – Part II

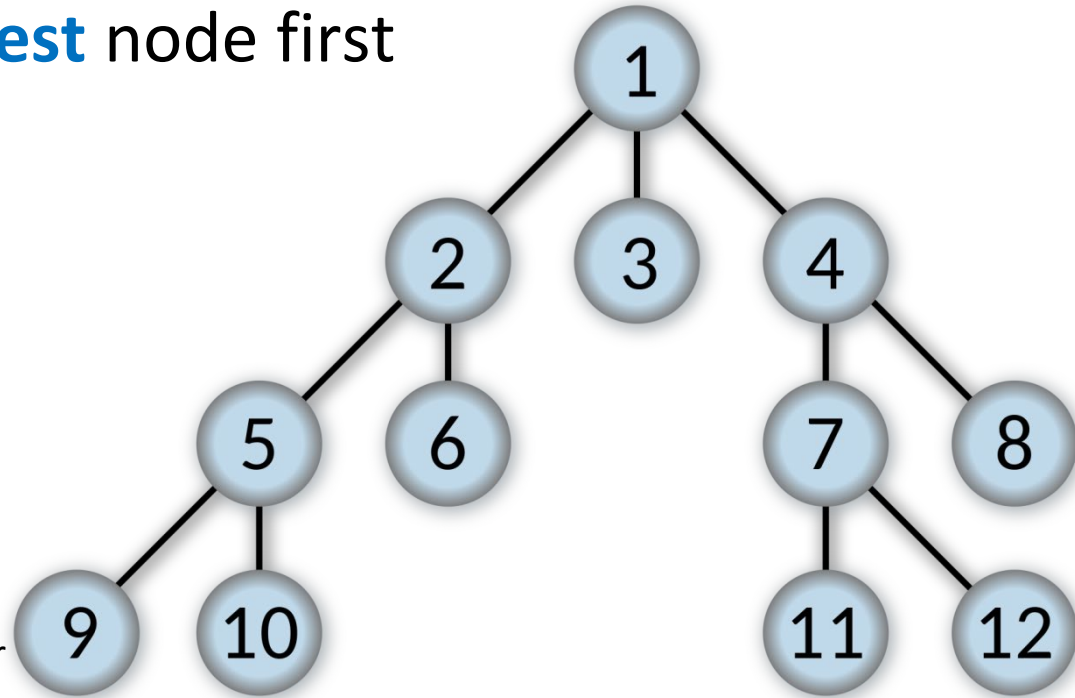
# Today's Goals

- Review of uninformed search strategies.
- Understand the difference between uninformed and informed search.
- Introduce A\* Search
  - Heuristic properties, stopping rules, analysis
- Extensions: Beyond A\*
  - Iterative deepening, beam search

# Review: Breadth-First Search

Recall: expand **shallowest** node first

- Data structure: queue
- **Properties:**
  - Complete
  - Optimal (if edge cost 1)
  - Time  $O(b^d)$ 
    - ← Depth
    - ← Branching Factor
  - Space  $O(b^d)$



# Review: Uniform Cost Search

Like BFS, but keeps track of cost

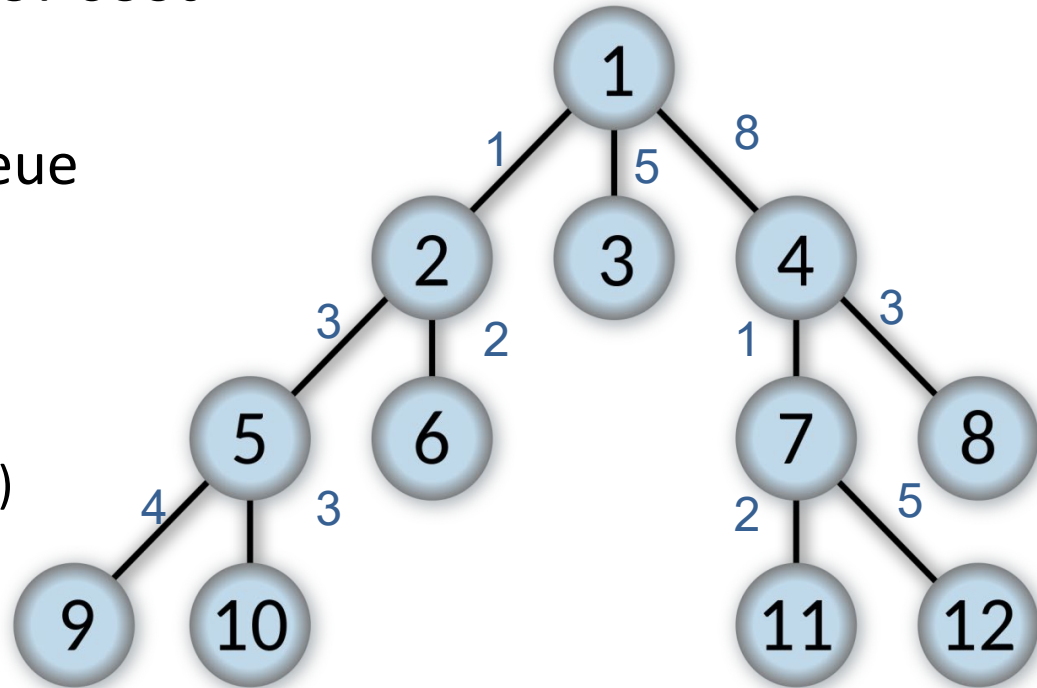
- Expand least cost node
- Data structure: priority queue

- **Properties:**

- Complete
- Optimal  
(if weight lower bounded by  $\epsilon$ )
- Time  $O(b^{C^*/\epsilon})$
- Space  $O(b^{C^*/\epsilon})$

$C^*$  is optimal path cost to goal.

$\epsilon$  is cost of edge with smallest cost.

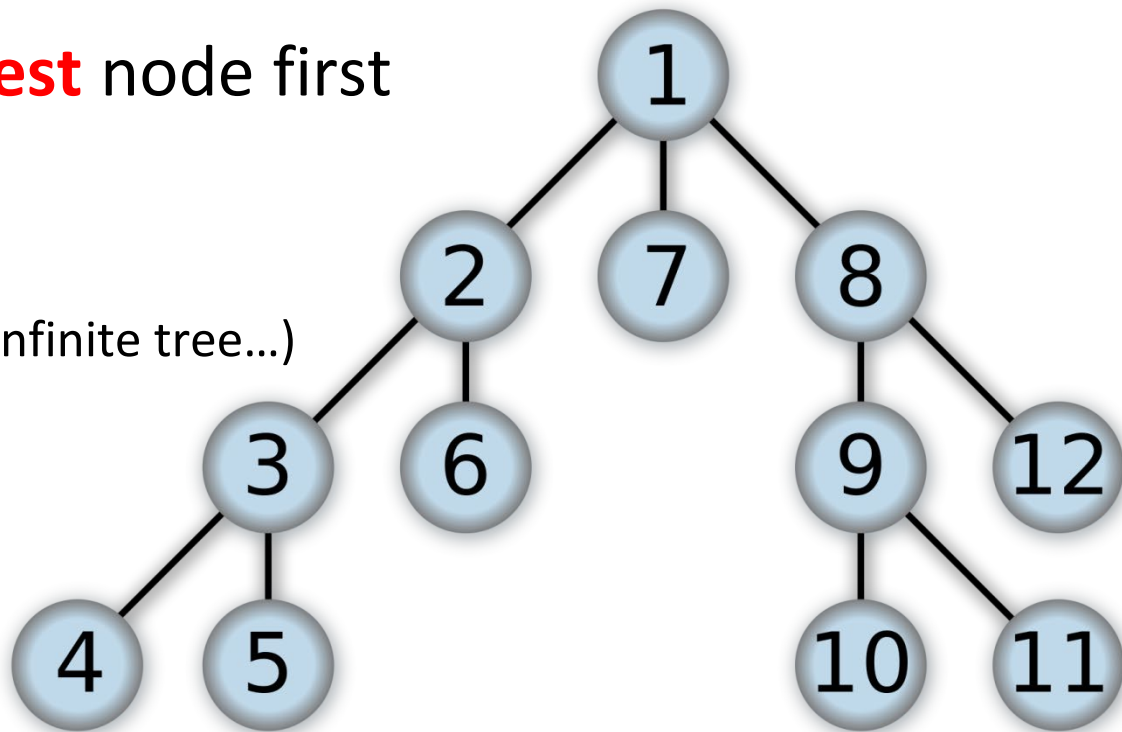


# Review: Depth-First Search

Recall: expand **deepest** node first

- Data structure: stack
- **Properties:**
  - Incomplete (stuck in infinite tree...)
  - Suboptimal
  - Time  $O(b^m)$

Max Depth



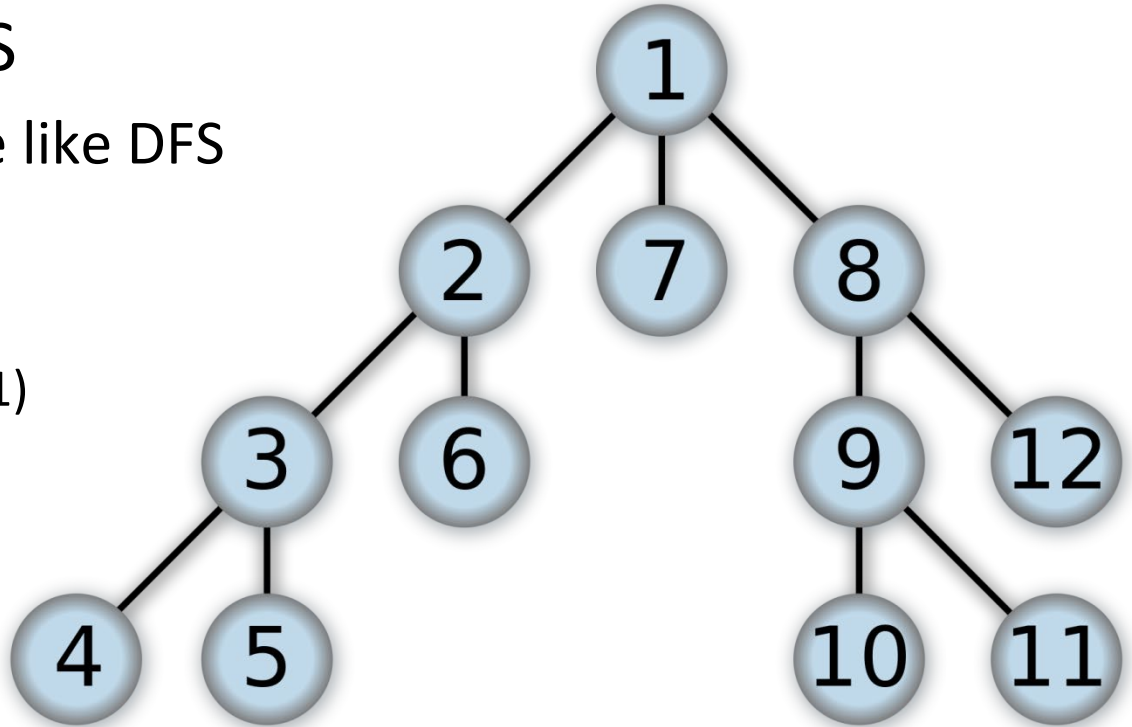
- Space  $O(bm)$

# Review: Iterative Deepening DFS

## Repeated limited DFS

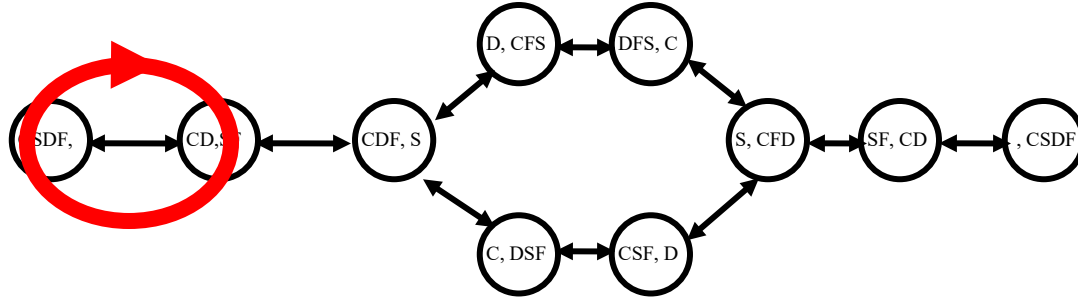
- Search like BFS, fringe like DFS
- **Properties:**
  - Complete
  - Optimal (if edge cost 1)
  - Time  $O(b^d)$
  - Space  $O(bd)$

**A good option!**



# Review: If state space graph is not a tree

- The problem: repeated states



- Ignore the danger of repeated states: wasteful (BFS) or impossible (DFS). Can you see why?
- How to prevent it?

# Review: If state space graph is not a tree

- We have to remember already-expanded states (**CLOSED**).
- When we take out a state from the fringe (OPEN), check whether it is in CLOSED (already expanded).
  - If yes, throw it away.
  - If no, expand it (add successors to OPEN), and move it to CLOSED.



# Informed Search

# Uninformed vs Informed Search

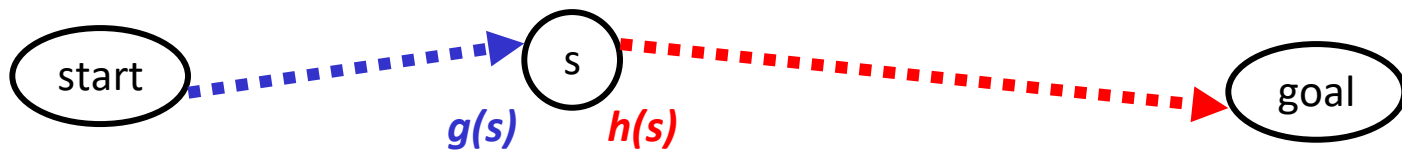
Uninformed search (all of what we saw). Know:

- Path cost  $g(s)$  from start to state  $s$ .
- Successors.



Informed search. Know:

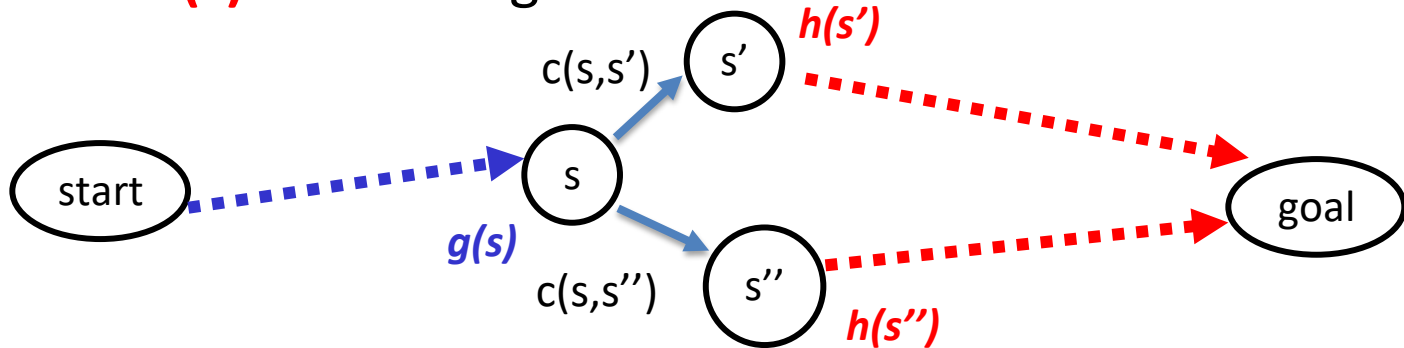
- All uninformed search properties, plus
- Heuristic  $h(s)$  from  $s$  to goal.



# Informed Search

Informed search. Know:

- All uninformed search properties, plus
- Heuristic  $h(s)$  from  $s$  to goal.

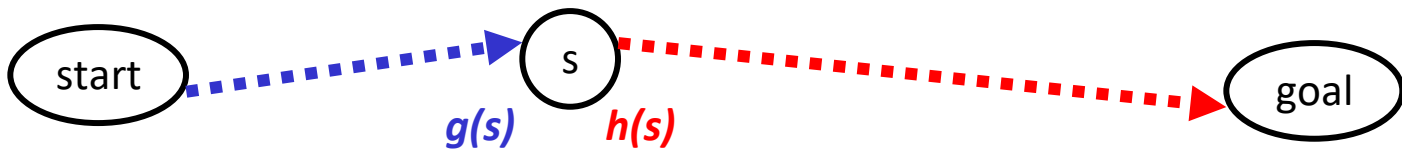


- Goal: **speed up search.**

# Using the Heuristic

Recall uniform-cost search

- We store potential next states with a priority queue
- Expand the state with the smallest  $g(s)$ 
  - $g(s)$  “first-half-cost”

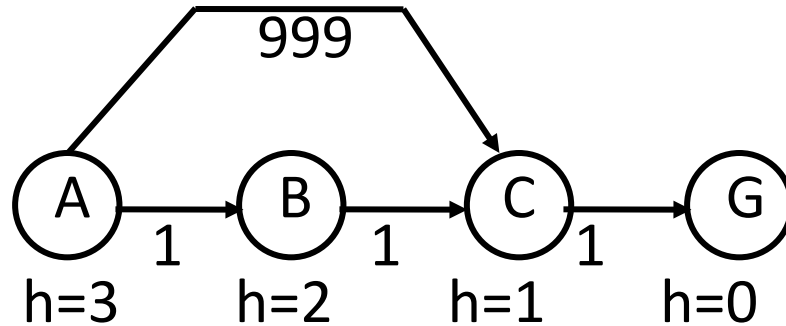


- Now let's use the heuristic (“second-half-cost”)
  - Several possible approaches: let's see what works

# Attempt 1: Best-First Greedy

One approach: just use  $h(s)$  alone

- Specifically, expand the state with smallest  $h(s)$
- This isn't a good idea. Why?

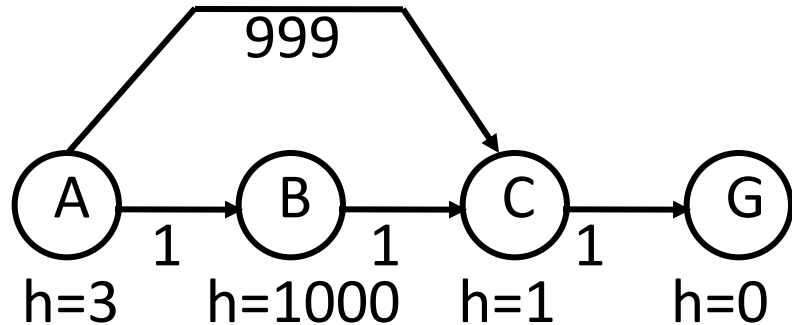


- Not optimal! **Get**  $A \rightarrow C \rightarrow G$ . **Want:**  $A \rightarrow B \rightarrow C \rightarrow G$

# Attempt 2: A Search

Next approach: use both  $g(s)$  +  $h(s)$

- Specifically, expand state with smallest  $g(s)$  +  $h(s)$
- Again, use a priority queue
- Called “A” search



- **Still not optimal!** (Does work for former example).

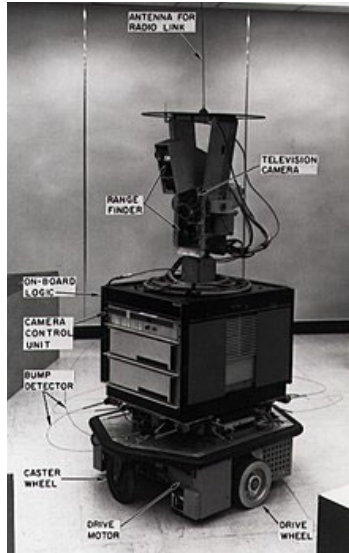
# Attempt 3: A\* Search

Same idea, use  $g(s) + h(s)$ , with one **requirement**

- Demand that  $h(s) \leq h^*(s)$  where  $h^*(s)$  is true cost from  $s$  to goal.
- If heuristic has this property, it is called “admissible”
  - Optimistic! Never over-estimates
- Still need  $h(s) \geq 0$ 
  - Negative heuristics can lead to strange behavior
- This is **A\* search**

# Attempt 3: A\* Search

## Origins: robots and planning



Shakey the Robot,  
1960's

Credit: Wiki



**Animation:** finding a path  
around obstacle

Credit: Wiki

# Admissible Heuristic Functions

Have to be careful to ensure admissibility (**optimism!**)

- Example: **8 Game**

Example State

1		5
2	6	3
7	4	8

Goal State

1	2	3
4	5	6
7	8	

- One useful approach: **relax constraints**
  - $h(s)$  = number of tiles in wrong position
    - allows tiles to fly to destination in a single step

# Break & Quiz

**Q 1.1:** Consider finding the fastest driving route from one US city to another. Measure cost as the number of hours driven when driving at the speed limit. Let  $h(s)$  be the number of hours needed to ride a bike from city  $s$  to your destination.  $h(s)$  is

- A. An admissible heuristic
- B. Not an admissible heuristic

# Break & Quiz

**Q 1.1:** Consider finding the fastest driving route from one US city to another. Measure cost as the number of hours driven when driving at the speed limit. Let  $h(s)$  be the number of hours needed to ride a bike from city  $s$  to your destination.  $h(s)$  is

- A. An admissible heuristic
- **B. Not an admissible heuristic**

# Break & Quiz

**Q 1.1:** Consider finding the fastest driving route from one US city to another. Measure cost as the number of hours driven when driving at the speed limit. Let  $h(s)$  be the number of hours needed to ride a bike from city  $s$  to your destination.  $h(s)$  is

- A. An admissible heuristic **No: riding your bike takes longer.**
- **B. Not an admissible heuristic**

# Break & Quiz

**Q 1.2:** Which of the following are admissible heuristics?

(i)  $h(s) = h^*(s)$

(ii)  $h(s) = \max(2, h^*(s))$

(iii)  $h(s) = \min(2, h^*(s))$

(iv)  $h(s) = h^*(s) - 2$

(v)  $h(s) = \text{sqrt}(h^*(s))$

- A. All of the above
- B. (i), (iii), (iv)
- C. (i), (iii)
- D. (i), (iii), (v)

# Break & Quiz

**Q 1.2:** Which of the following are admissible heuristics?

(i)  $h(s) = h^*(s)$

(ii)  $h(s) = \max(2, h^*(s))$

(iii)  $h(s) = \min(2, h^*(s))$

(iv)  $h(s) = h^*(s) - 2$

(v)  $h(s) = \text{sqrt}(h^*(s))$

- A. All of the above
- B. (i), (iii), (iv)
- **C. (i), (iii)**
- D. (i), (iii), (v)

# Break & Quiz

**Q 1.2:** Which of the following are admissible heuristics?

(i)  $h(s) = h^*(s)$

(ii)  $h(s) = \max(2, h^*(s))$       No:  $h(s)$  might be too big

(iii)  $h(s) = \min(2, h^*(s))$

(iv)  $h(s) = h^*(s) - 2$       No:  $h(s)$  might be negative

(v)  $h(s) = \text{sqrt}(h^*(s))$       No: if  $h^*(s) < 1$  then  $h(s)$  is bigger

- A. All of the above
- B. (i), (iii), (iv)
- **C. (i), (iii)**
- D. (i), (iii), (v)

# Heuristic Function Tradeoffs

Dominance:  $h_2$  dominates  $h_1$  if for all states  $s$ ,

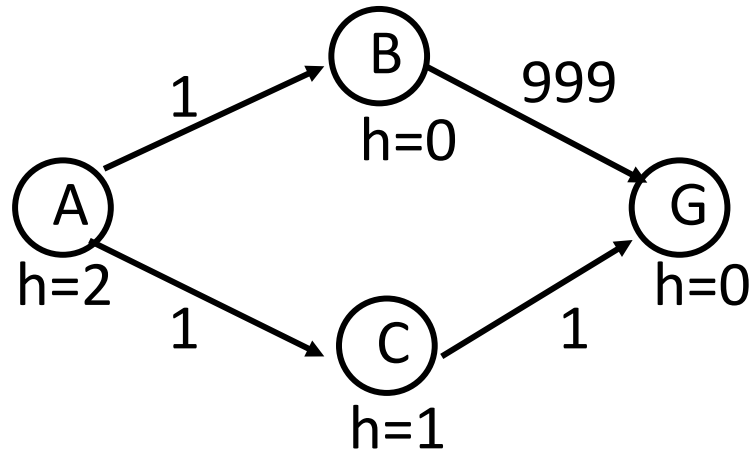
$$h_1(s) \leq h_2(s) \leq h^*(s)$$

- **Idea:** we want to be as close to  $h^*$  as possible
  - But not over! **Must under-estimate true cost.**
- **Tradeoff:** being very close might require a very complex heuristic, expensive computation
  - Might be better off with cheaper heuristic & expand more nodes.

# A\* Termination

When should A\* **stop**?

- One idea: as soon as we reach goal state?

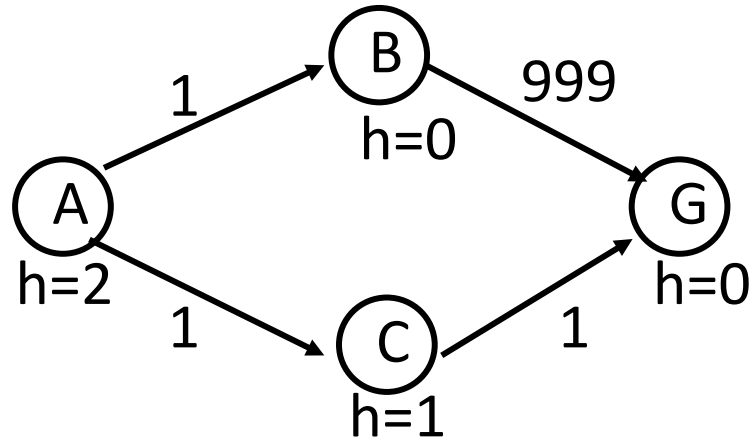


- ***h*** is admissible, but note that we get  $A \rightarrow B \rightarrow G$  (**cost 1000**)!

# A\* Termination

When should A\* **stop**?

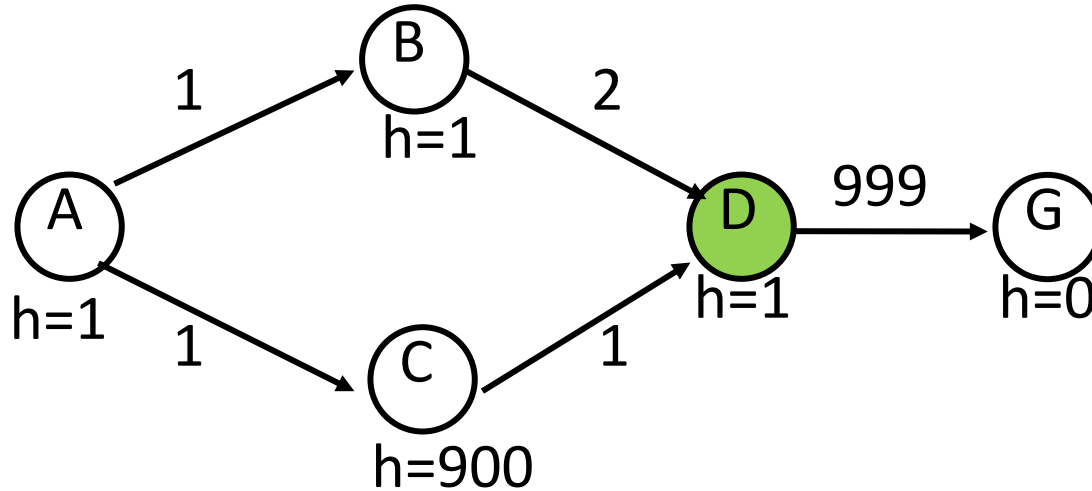
- **Rule:** terminate **when a goal is popped** from queue.



- Note: taking  $h=0$  reduces to uniform cost search rule.

# A\* Revisiting Expanded States

Possible to revisit an expanded state, get a shorter path:



- Put D back into priority queue, smaller  $g+h$ .
- **Note:** uninformed search methods will not revisit expanded states.

# A\* Full Algorithm

1. Put the start state  $S$  on the priority queue. We call the priority queue **OPEN**
2. If **OPEN** is empty, exit with failure
3. Remove from **OPEN** and place on **CLOSED** a node  $n$  for which  $f(n)$  is minimum (note that  $f(n)=g(n)+h(n)$ )
4. If  $n$  is a goal node, exit (recover path by tracing back pointers from  $n$  to  $S$ )
5. Expand  $n$ , generating all successors and attach to pointers back to  $n$ .  
For each successor  $n'$  of  $n$ 
  1. If  $n'$  is not already on **OPEN** or **CLOSED** compute  $h(n')$ ,  $g(n')=g(n)+c(n,n')$ ,  $f(n')=g(n')+h(n')$ , and place it on **OPEN**.
  2. If  $n'$  is already on **OPEN** or **CLOSED**, then check if  $g(n')$  is lower for the new version of  $n'$ .  
If so, then:
    1. Redirect pointers backward from  $n'$  along path yielding lower  $g(n')$ .
    2. If ( $n'$  is already on **OPEN**) then update  $n'$  on **OPEN**; else add  $n'$  to **OPEN**
  3. If  $g(n')$  is not lower for the new version, do nothing.
6. Goto 2.

States we have already expanded



# A\* Analysis

Some properties:

- Terminates!
- A\* can use **lots of memory**:
  - $O(\# \text{ states})$ .
- Will run out on large problems.
- Next, we will consider some alternatives to deal with this.

# Break & Quiz

**Q 2.1:** Consider two heuristics for the 8 puzzle problem.  $h_1$  is the number of tiles in wrong position.  $h_2$  is the  $l_1$ /Manhattan distance between the tiles and the goal location. How do  $h_1$  and  $h_2$  relate?

- A.  $h_2$  dominates  $h_1$
- B.  $h_1$  dominates  $h_2$
- C. Neither dominates the other

# Break & Quiz

**Q 2.1:** Consider two heuristics for the 8 puzzle problem.  $h_1$  is the number of tiles in wrong position.  $h_2$  is the  $l_1$ /Manhattan distance between the tiles and the goal location. How do  $h_1$  and  $h_2$  relate?

- **A.  $h_2$  dominates  $h_1$**
- B.  $h_1$  dominates  $h_2$
- C. Neither dominates the other

# Break & Quiz

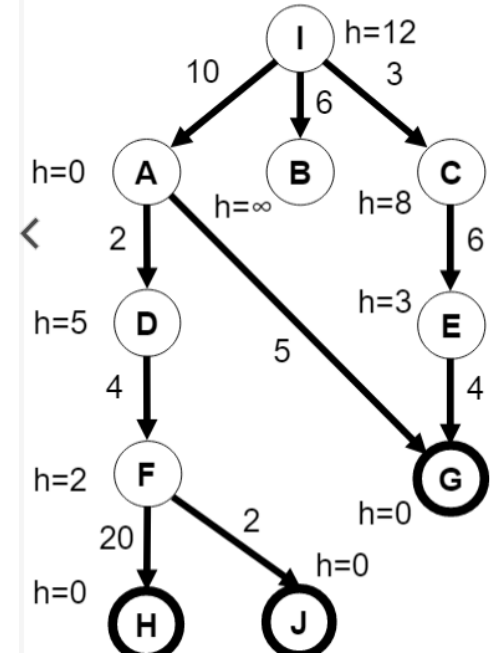
**Q 2.1:** Consider two heuristics for the 8 puzzle problem.  $h_1$  is the number of tiles in wrong position.  $h_2$  is the  $l_1$ /Manhattan distance between the tiles and the goal location. How do  $h_1$  and  $h_2$  relate?

- **A.  $h_2$  dominates  $h_1$**
- **B.  $h_1$  dominates  $h_2$  (No:  $h_1$  is a distance where each entry is at most 1,  $h_2$  can be greater)**
- C. Neither dominates the other

# Break & Quiz

**Q 2.2:** Consider the state space graph below. Goal states have bold borders.  $h(s)$  is shown next to each node. What node will be expanded by A\* after the initial state I?

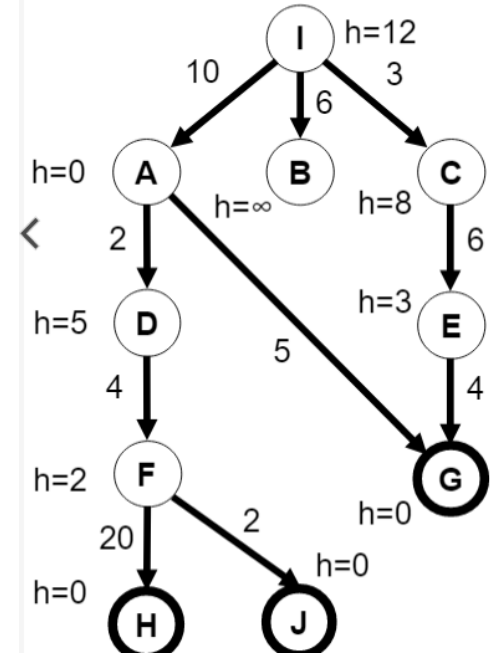
- A. A
- B. B
- C. C



# Break & Quiz

**Q 2.2:** Consider the state space graph below. Goal states have bold borders.  $h(s)$  is show next to each node. What node will be expanded by A\* after the initial state I?

- **A. A**
- B. B
- C. C



# IDA\*: Iterative Deepening A\*

Similar idea to our earlier iterative deepening.

- Bound the memory in search.
- At each phase, don't expand any node with  $g(s) + h(s) > k$ ,
  - Assuming integer costs, do this for  $k=0$ , then  $k=1$ , then  $k=2$ , and so on
- Complete + optimal, might be costly time-wise
  - Revisit many nodes
- Lower memory use than A\*

# IDA\*: Properties

How many restarts do we expect?

- With integer costs, optimal solution  $C^*$ , at most  $C^*$

What about non-integer costs?

- Initial threshold  $k$ . Use the same rule for non-expansion
- Set new  $k$  to be the min  $g(s) + h(s)$  for non-expanded nodes
- Worst case: restarted for each state

# Beam Search

General approach (beyond A\* too)

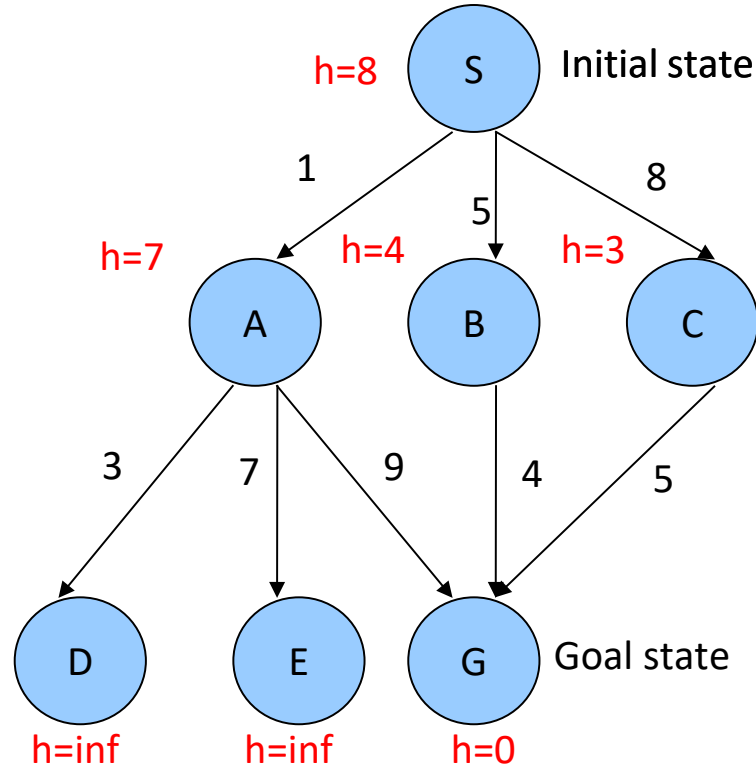
- Priority queue with fixed size  $k$ ; beyond  $k$  nodes, **discard!**
- **Upside**: good memory efficiency
- **Downside**: not complete or optimal

Variation:

- Priority queue with nodes that **are at most  $\epsilon$  worse** than best node.

# Recap and Examples

**Example for A\*:**



# Recap and Examples

## Example for A\*:

OPEN

S(0+8)

A(1+7) B(5+4) C(8+3)

B(5+4) C(8+3) D(4+inf) E(8+inf) G(10+0)

C(8+3) D(4+inf) E(8+inf) G(9+0)

C(8+3) D(4+inf) E(8+inf)

CLOSED

-

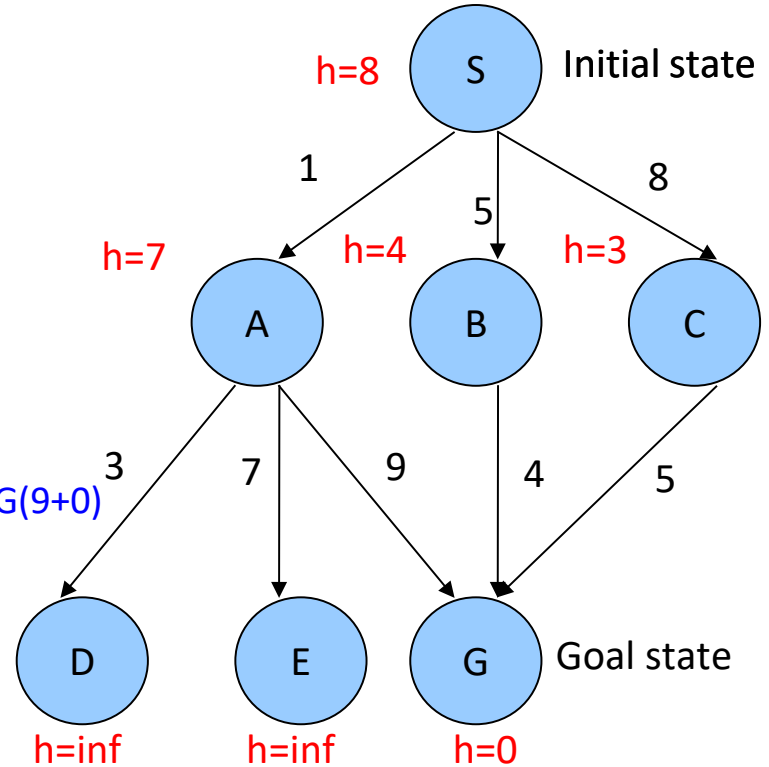
S(0+8)

S(0+8) A(1+7)

S(0+8) A(1+7) B(5+4)

S(0+8) A(1+7) B(5+4) G(9+0)

G → B → S



# Recap and Examples

**Example for IDA\*:**

**Threshold = 8**

PATH PREFIX

-

S

SA

SAH

SAHF

SAD

OPEN

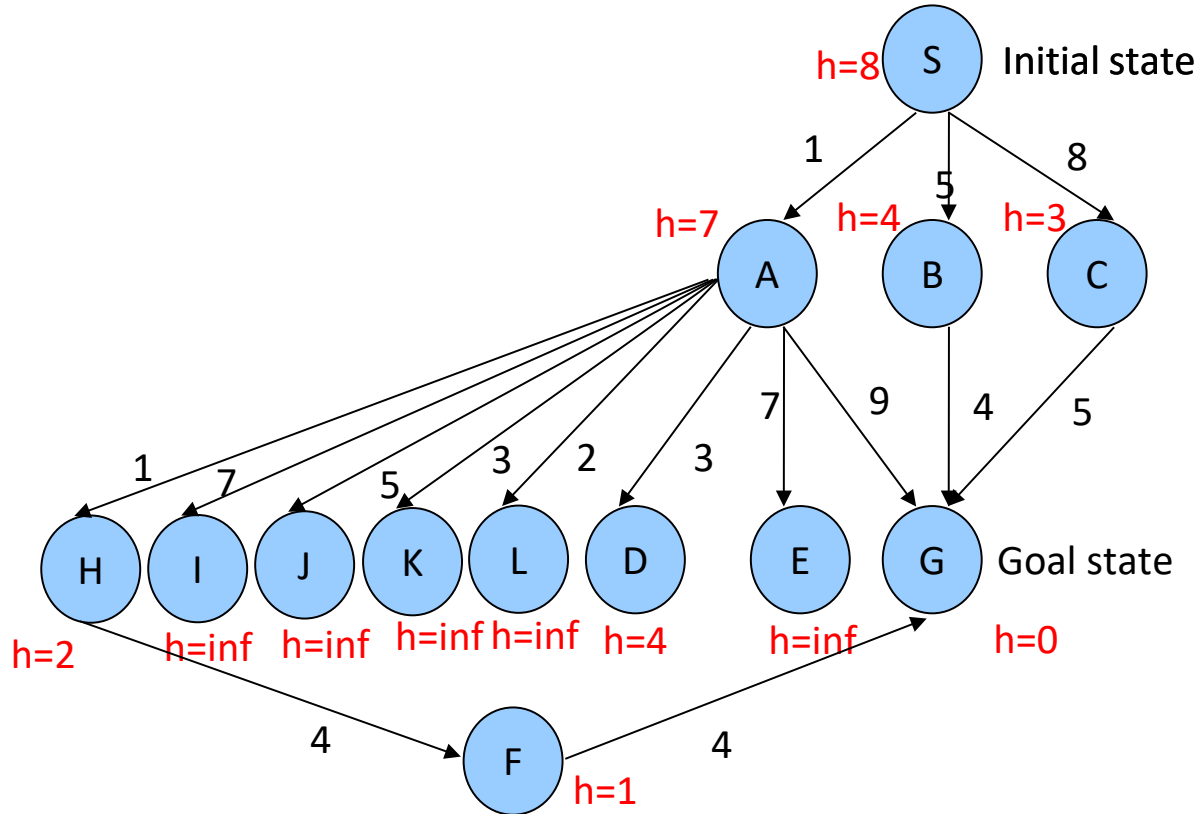
S(0+8)

A(1+7)

H(2+2) D(4+4)

D(4+4) F(6+1)

D(4+4)

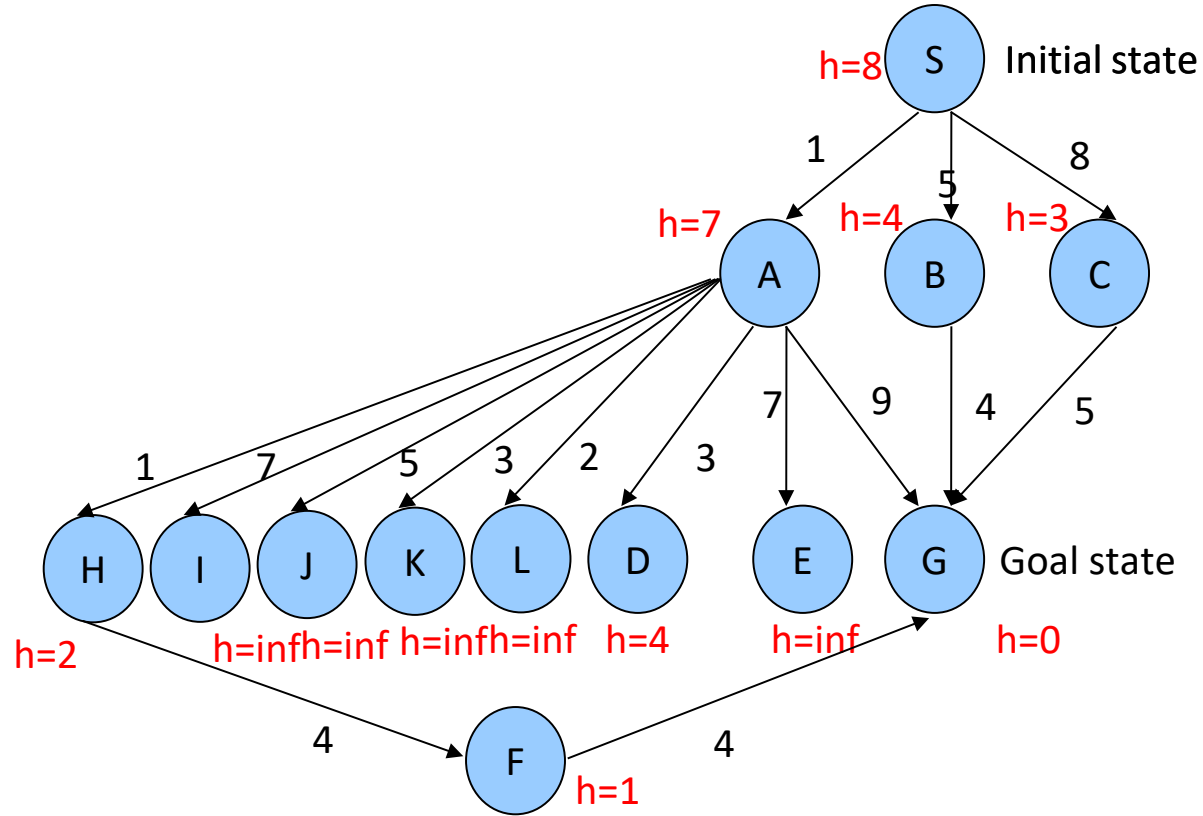


# Recap and Examples

**Example for IDA\*:**

**Threshold = 9**

PREFIX	OPEN
-	S(0+8)
S	A(1+7) B(5+4)
SA	B(5+4) H(2+2) D(4+4)
SAH	B(5+4) D(4+4) F(6+1)
SAHF	B(5+4) D(4+4)
SAD	B(5+4)
SB	G(9+0)
SBG	



# Recap and Examples

## Example for Beam Search: $k=2$

CURRENT

-

S

A

H

F

D

G

OPEN

S(0+8)

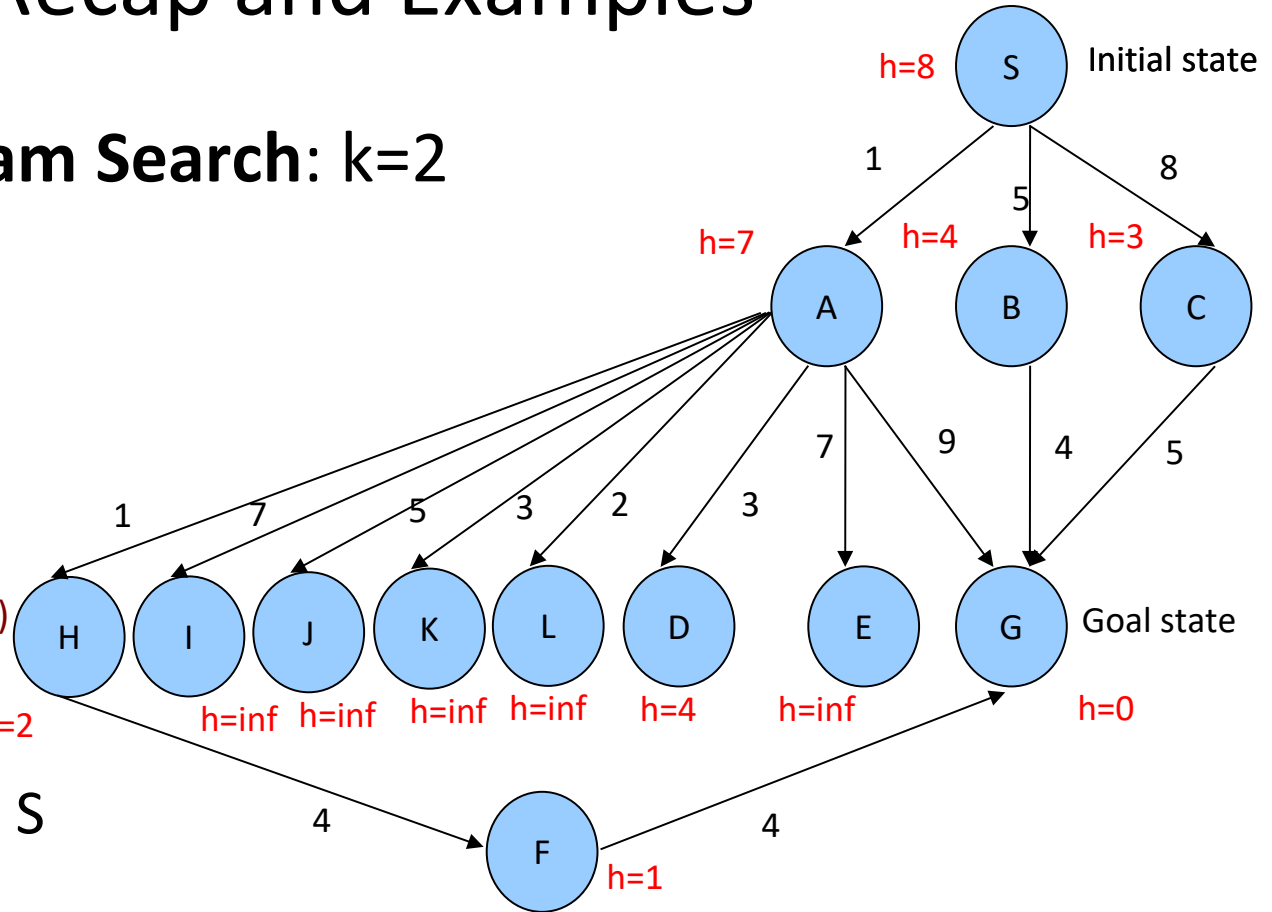
A(1+7) B(5+4)

H(2+2) D(4+4)

D(4+4) F(6+1)

D(4+4) G(10+0)

G(10+0)



$G \rightarrow F \rightarrow H \rightarrow A \rightarrow S$

Not optimal!

# Summary

- Informed search: introduce heuristics
  - Not all approaches work: best-first greedy is bad
- A\* algorithm
  - Properties of A\*, idea of admissible heuristics
- Beyond A\*
  - IDA\*, beam search. Ways to deal with space requirements.

# Suggested Readings

Textbook: Artificial Intelligence: A Modern Approach (4th edition).

Stuart Russell and Peter Norvig. Pearson, 2020.

- Section 3.5