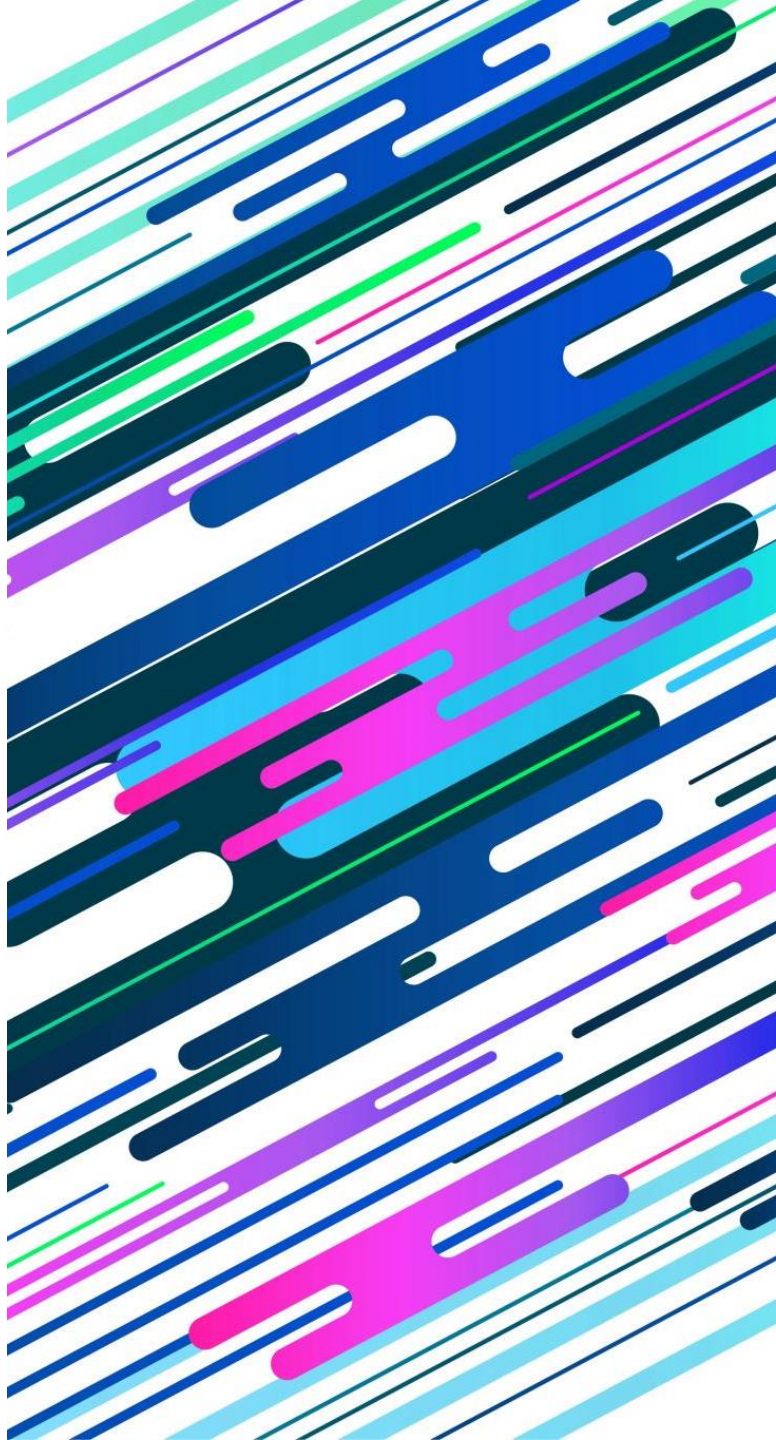
HEURISTICS AND A* SEARCH

Thomas Tiam-Lee, PhD



Informed VS Uninformed Search

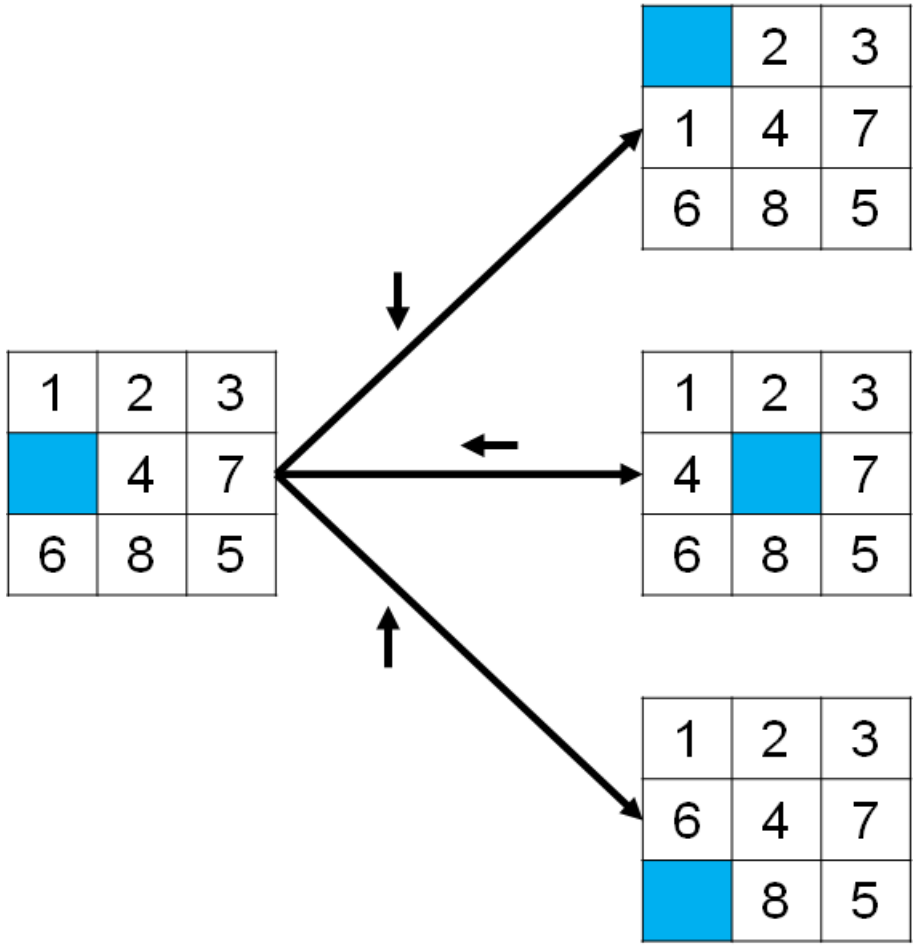
- **Uninformed Search**: Naively explore the state space without any additional domain knowledge about how to get nearer to the goal (also called **blind search**)
  - Backtracking, BFS, DFS, DFS-ID, and UCS are all uninformed search algorithms
- **Informed Search**: Exploit partial knowledge about the problem to guide the exploration of the state space (also known as **heuristic search**)

8-Puzzle

- Start with a random configuration.
- Each turn you can slide one of the adjacent tiles to the empty space into the empty space.
- Goal is to arrange it to the configuration shown on the right.

How Would a Human Do It?

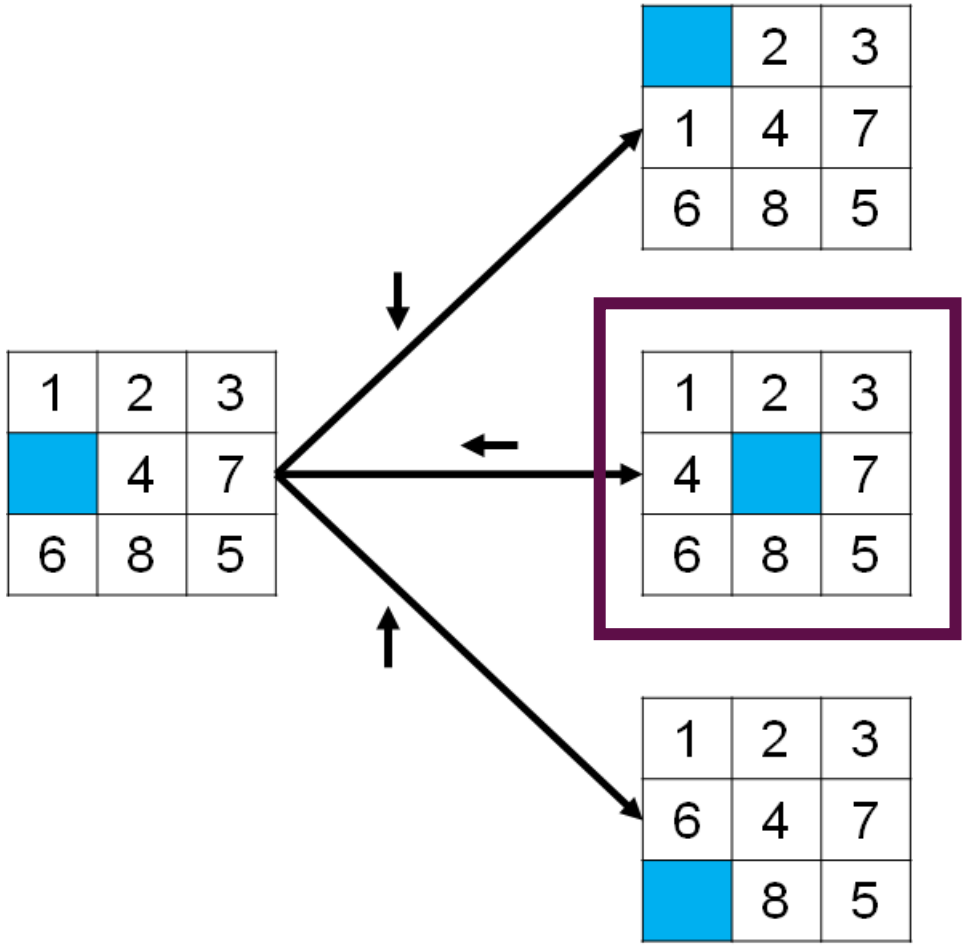
- If you were playing and you were at this state, **which of the three possible next moves would you try out first**? Why?



How Would a Human Do It?

- If you were playing and you were at this state, **which of the three possible next moves would you try out first**? Why?

- It makes sense to try the “left” move first because it puts more tiles in the correct spots!



Informed Search Intuition

- In choosing the next state to explore from the frontier, **take into consideration how close that state is to the goal state**!
- “How close” depends on domain knowledge about the problem.
- We need a **mathematical way to quantify how close we think a state is to the goal**.

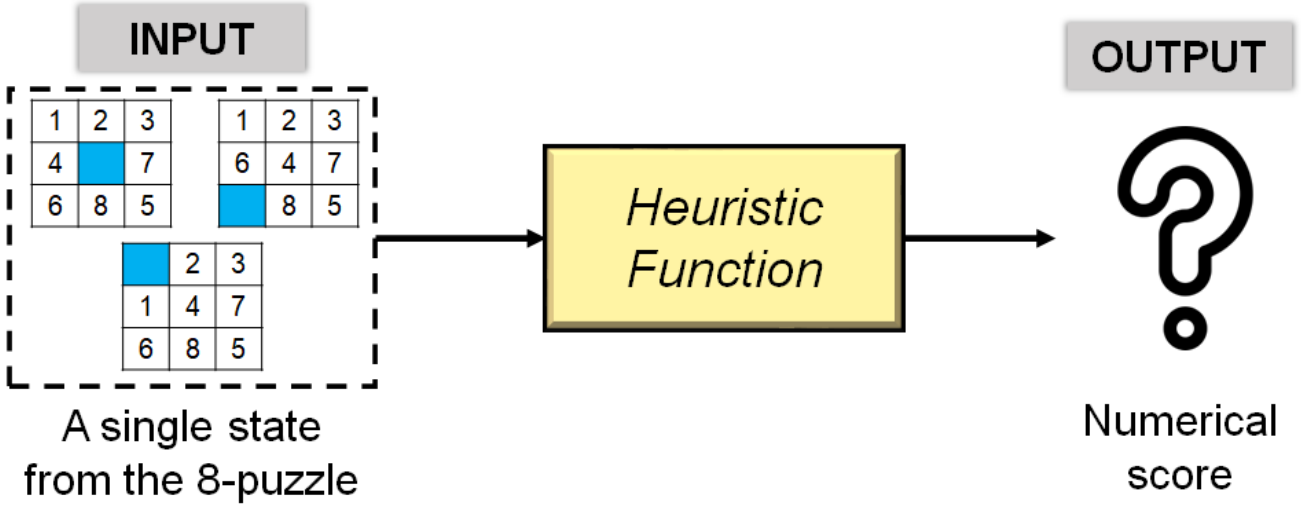
Heuristics

- Can be traced back to the Greek word *eurisco*, which means “I discover!”
- In AI, a heuristic is an **educated guess** on **which state is more likely to lead to a desirable outcome**.

Heuristic Function

- A heuristic function $h(s)$ takes a state s as a parameter and returns a **heuristic value** (a numerical score).
- The **smaller** the heuristic value, the **more desirable** the state is.

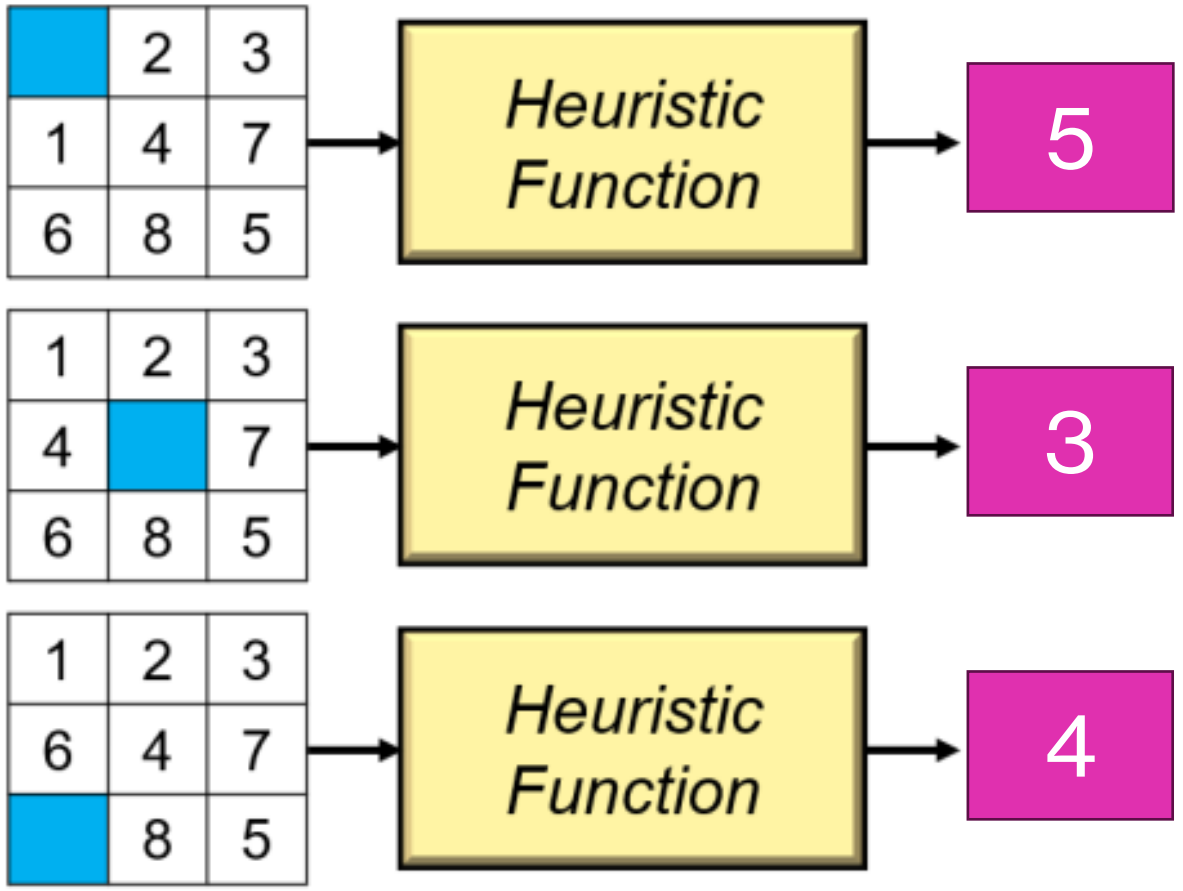
- $h(s)$ must be defined for any s.
- Need to consider time and space complexity.



What would be a possible heuristic for the 8-puzzle?

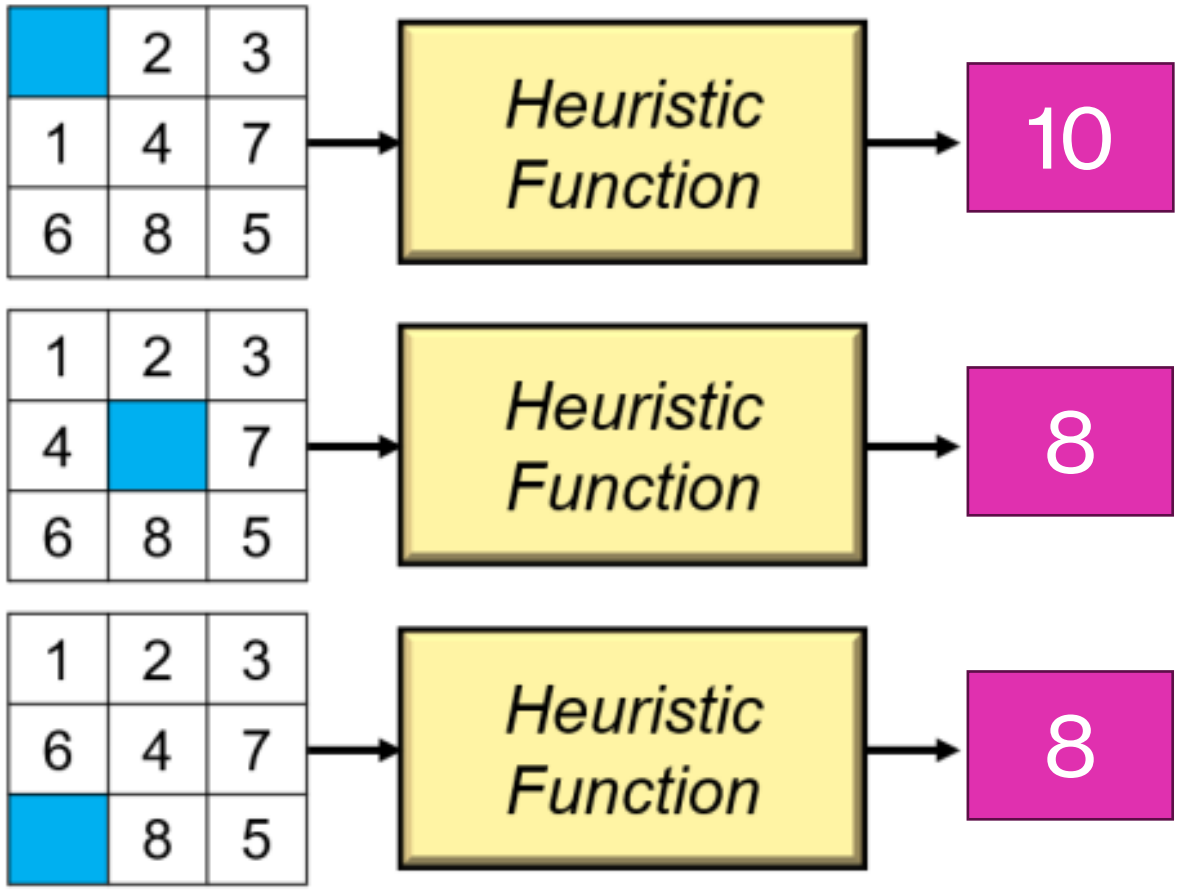
8-Puzzle Heuristic Function

- Possible heuristic function:
  - $h(s)$ = **number of tiles that are in the wrong place**
- Defined for any state
- Can be computed efficiently ($O(1)$).



Another 8-Puzzle Heuristic Function

- Possible heuristic function:
  - $h(s)$ = **total Manhattan Distance of each tile to its correct position**
- Defined for any state
- Can be computed efficiently ($O(1)$).



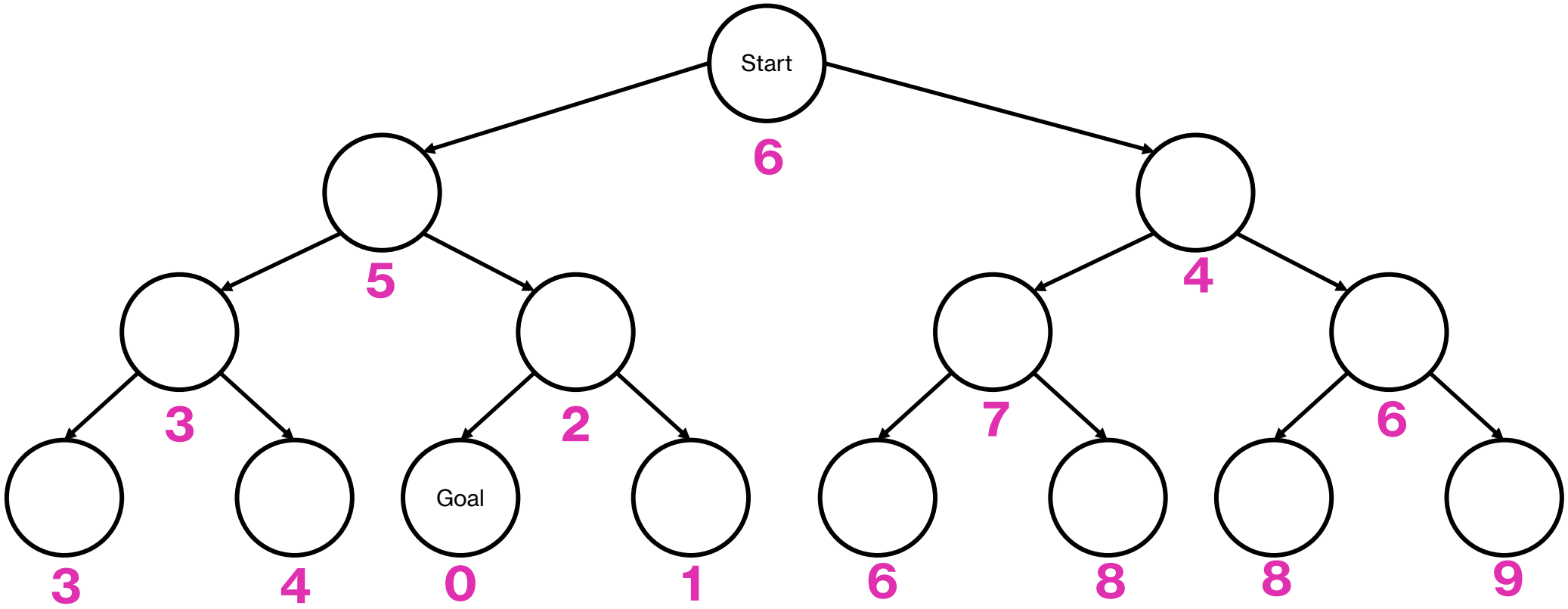
Informed Search Algorithms

- Greedy Best First Search
- A* Search

Greedy Best First Search

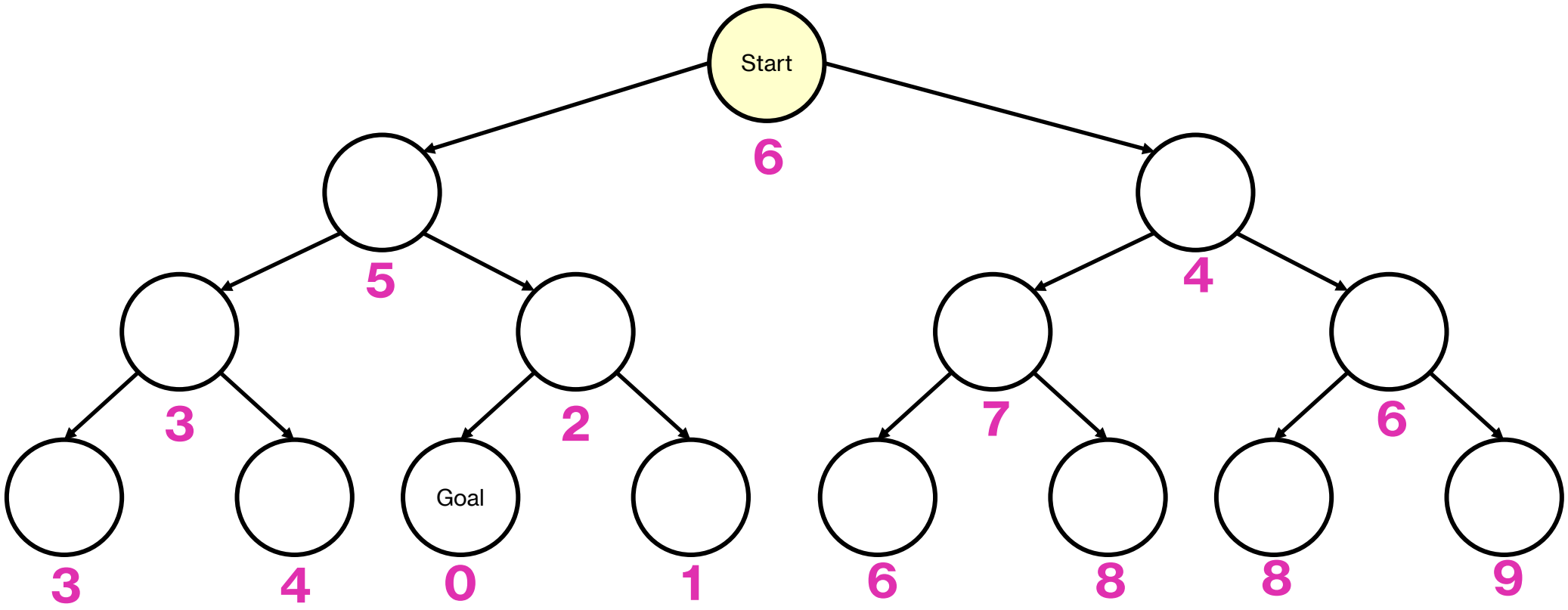
- **Key Idea:** In choosing the next state to explore from the frontier, **always choose the one with the smallest heuristic**!

Greedy Best First Search



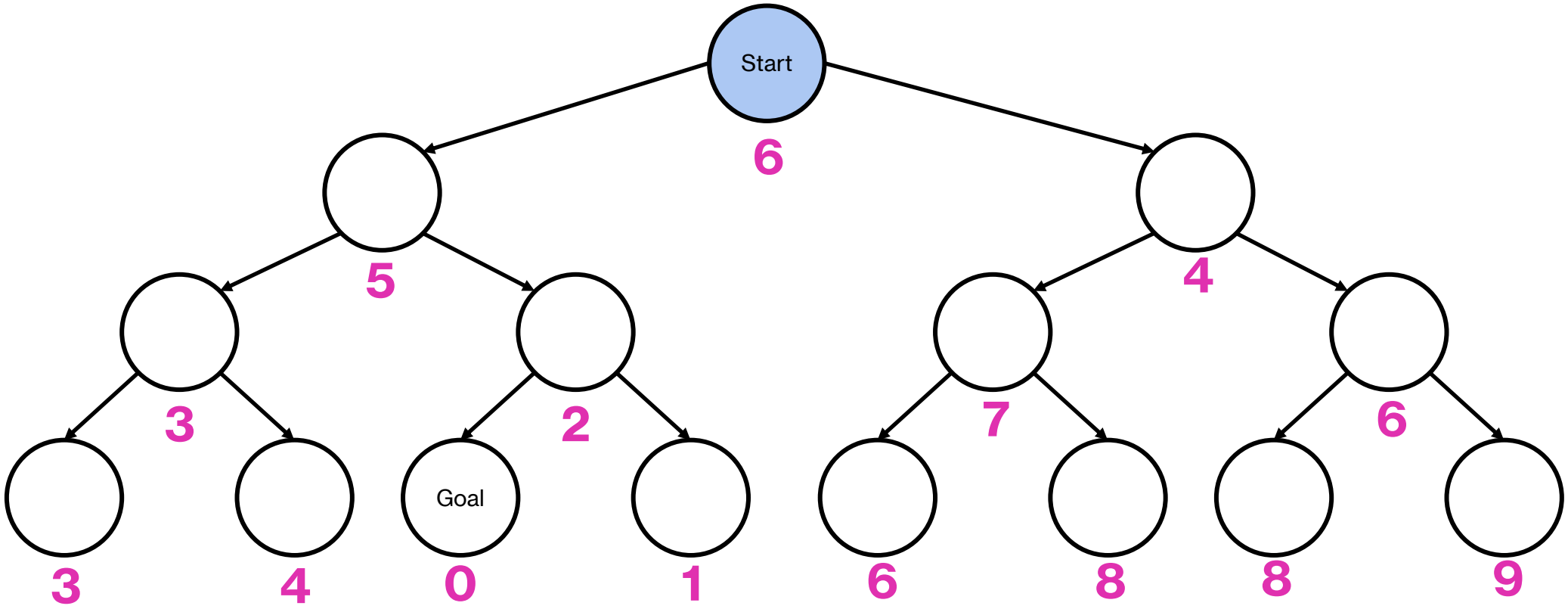
■ Heuristic value

Greedy Best First Search



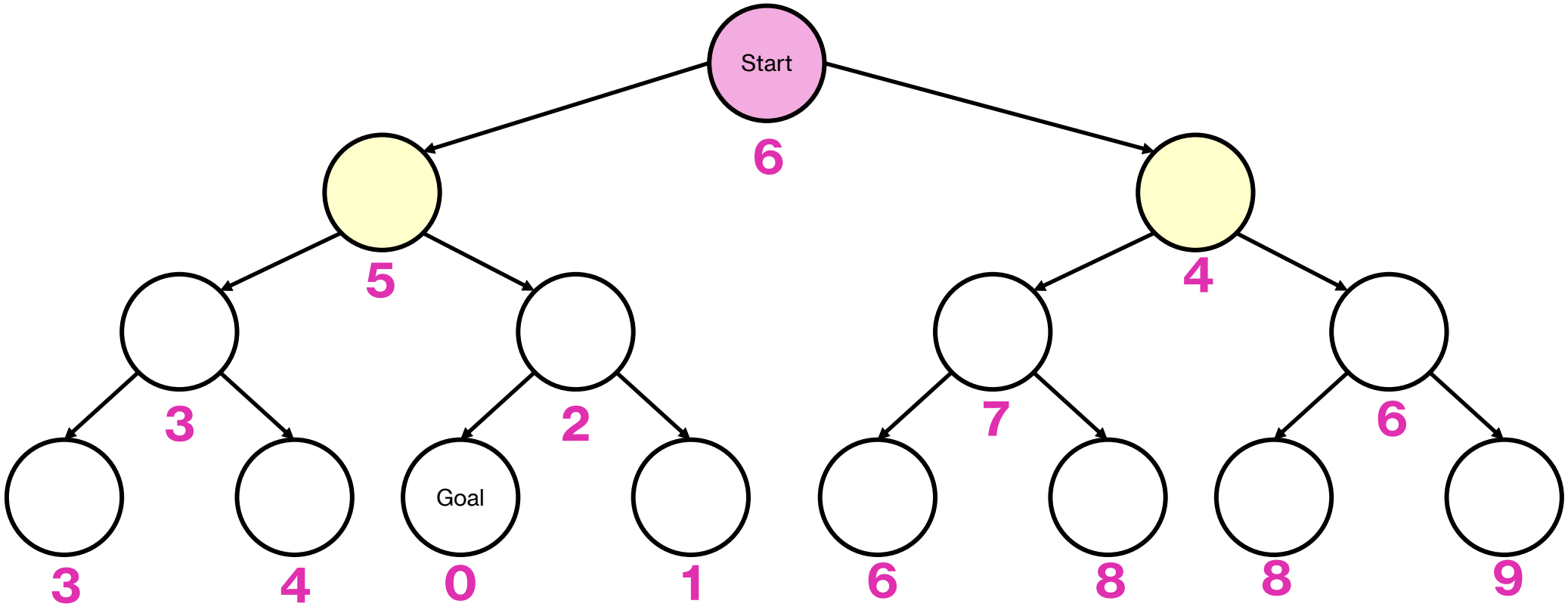
■ Heuristic value

Greedy Best First Search



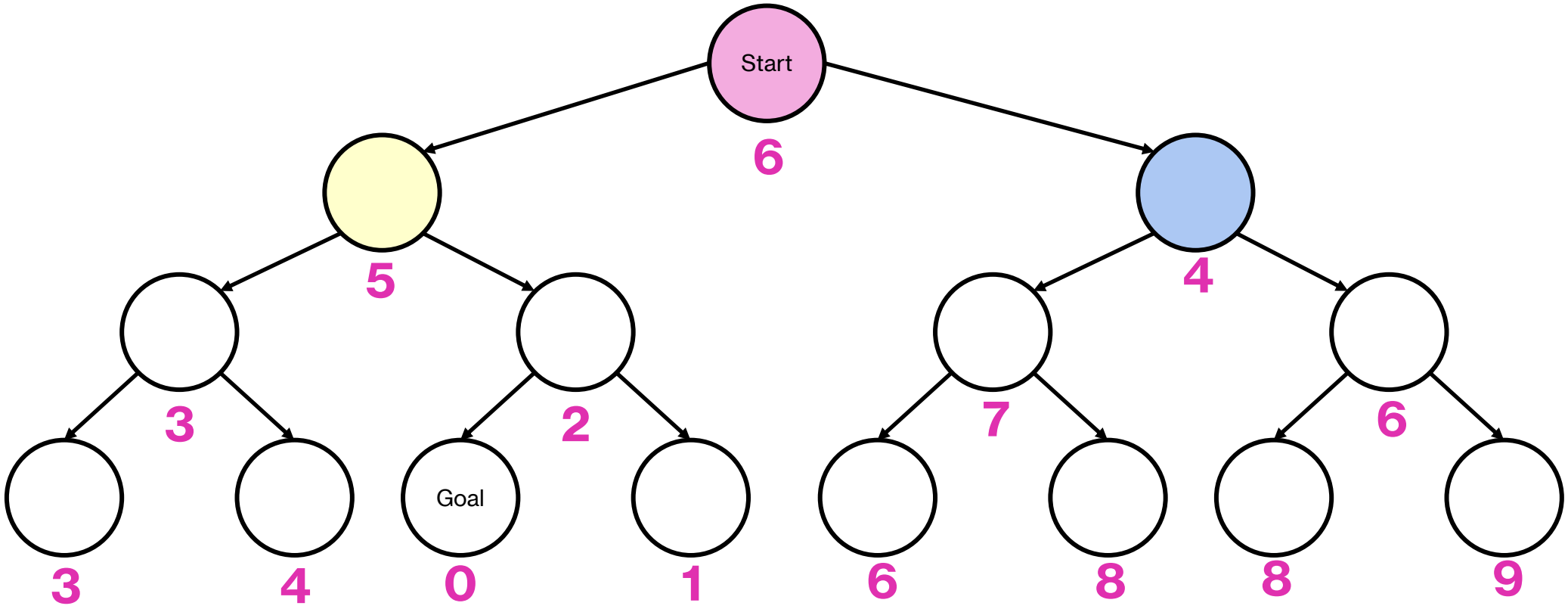
■ Heuristic value

Greedy Best First Search



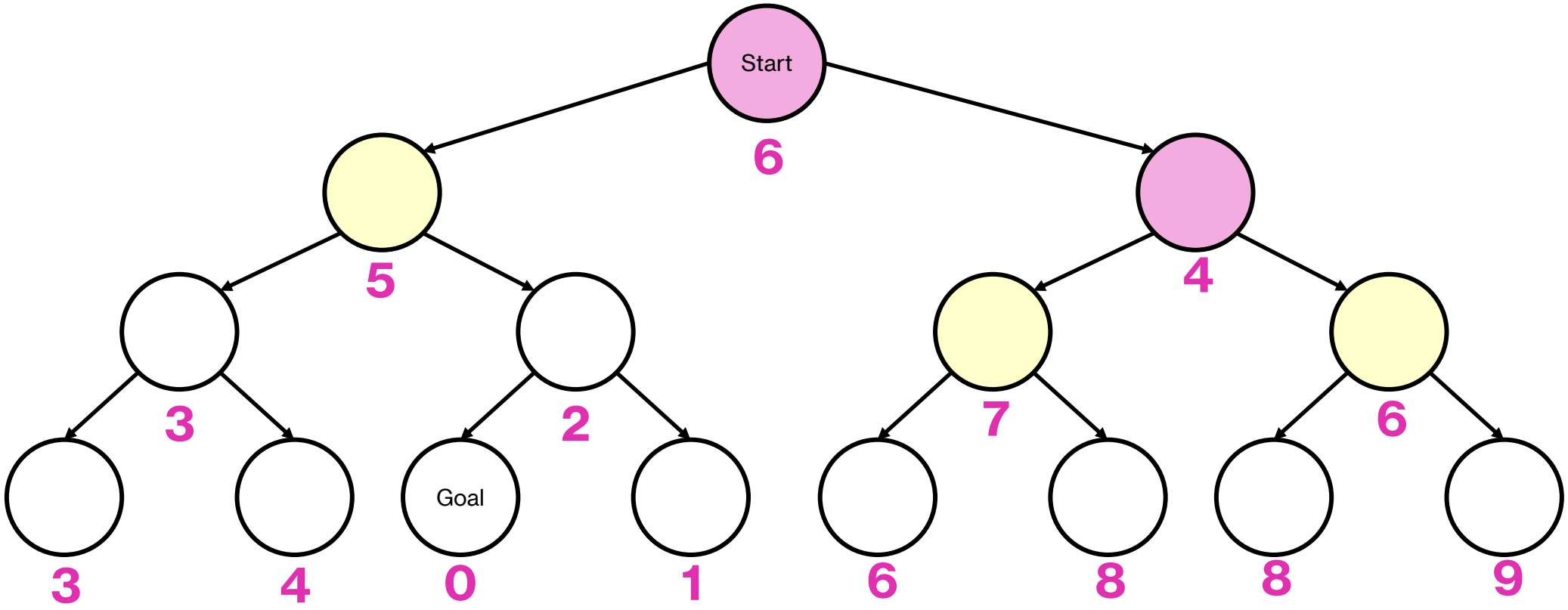
■ Heuristic value

Greedy Best First Search



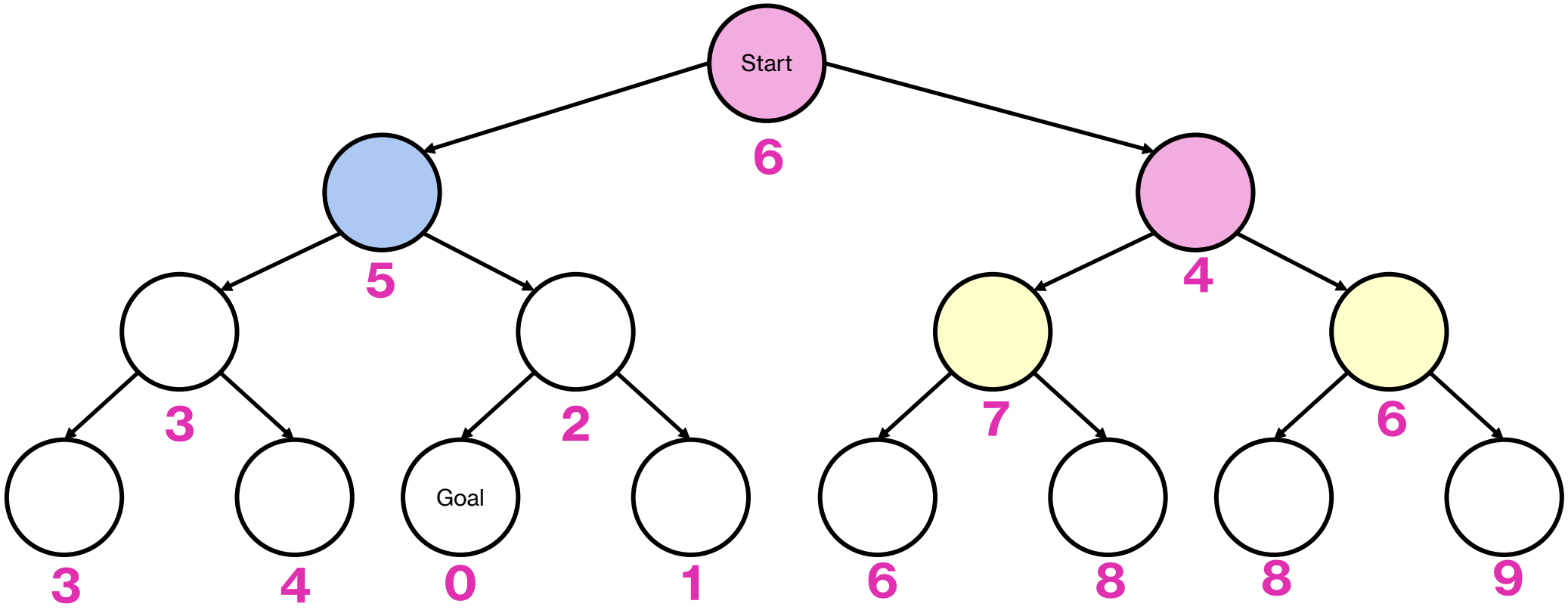
■ Heuristic value

Greedy Best First Search



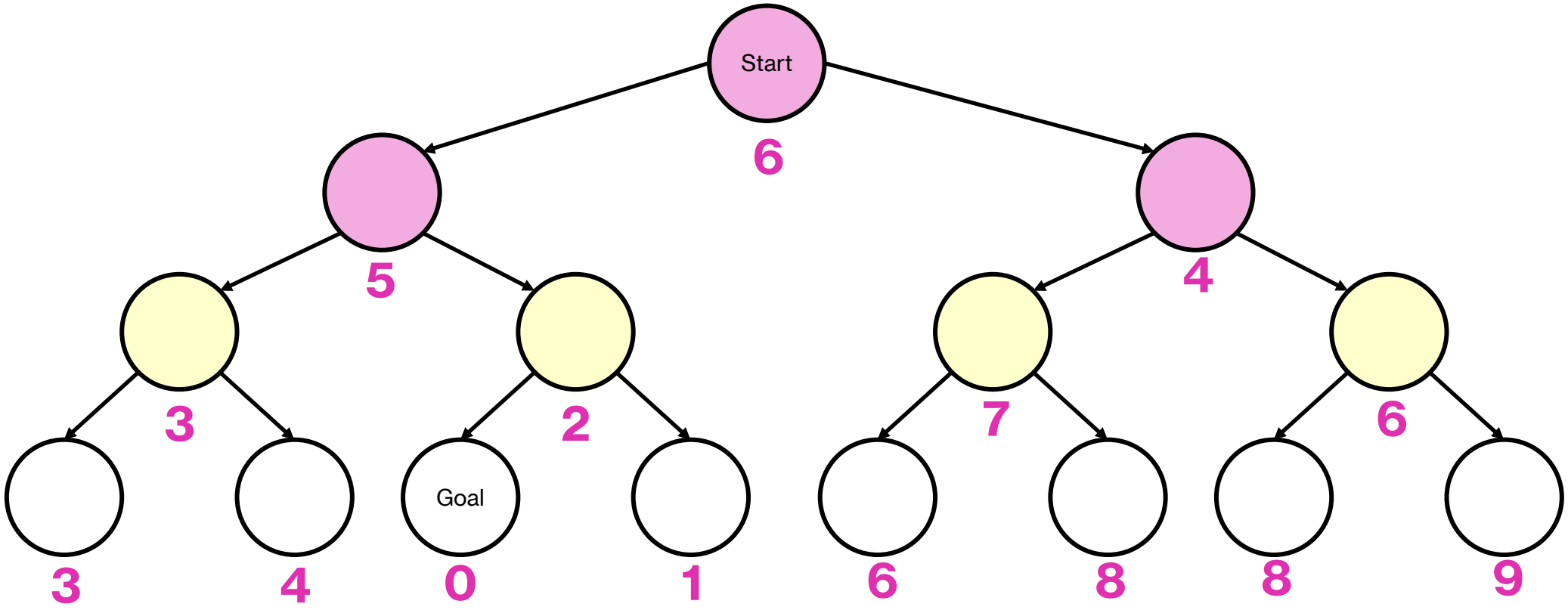
■ Heuristic value

Greedy Best First Search



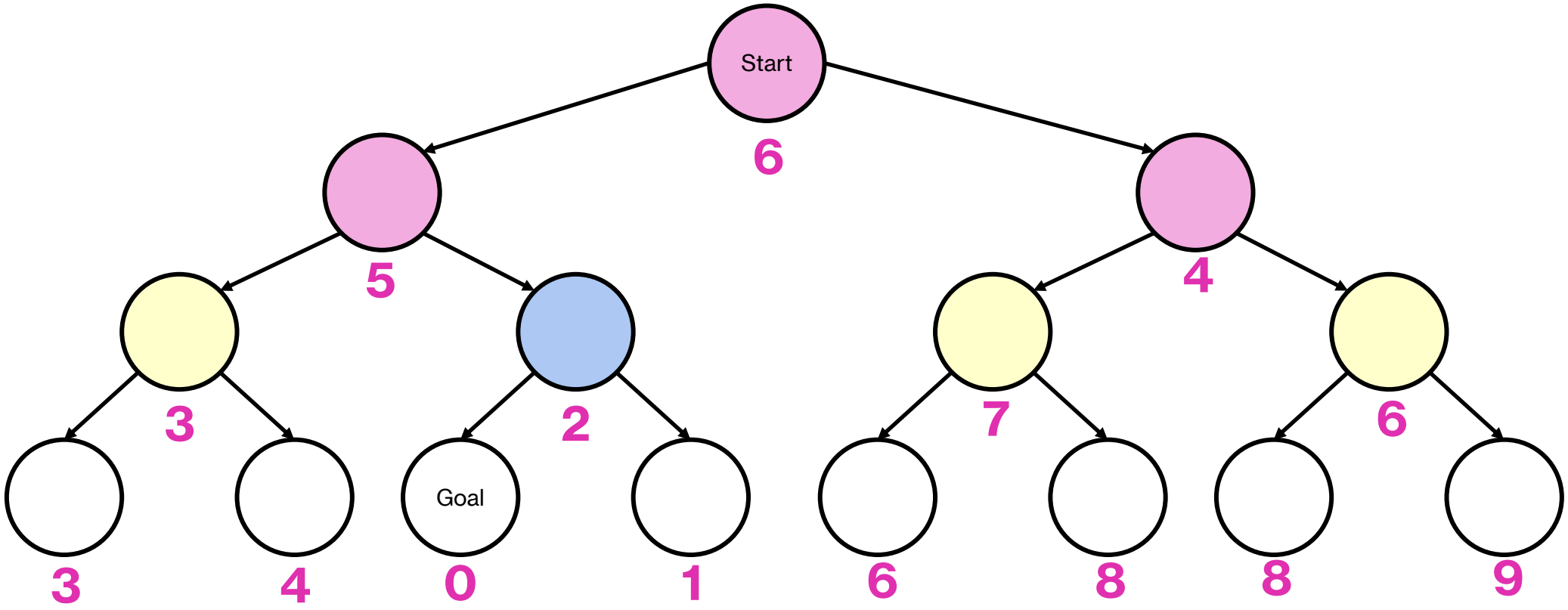
■ Heuristic value

Greedy Best First Search



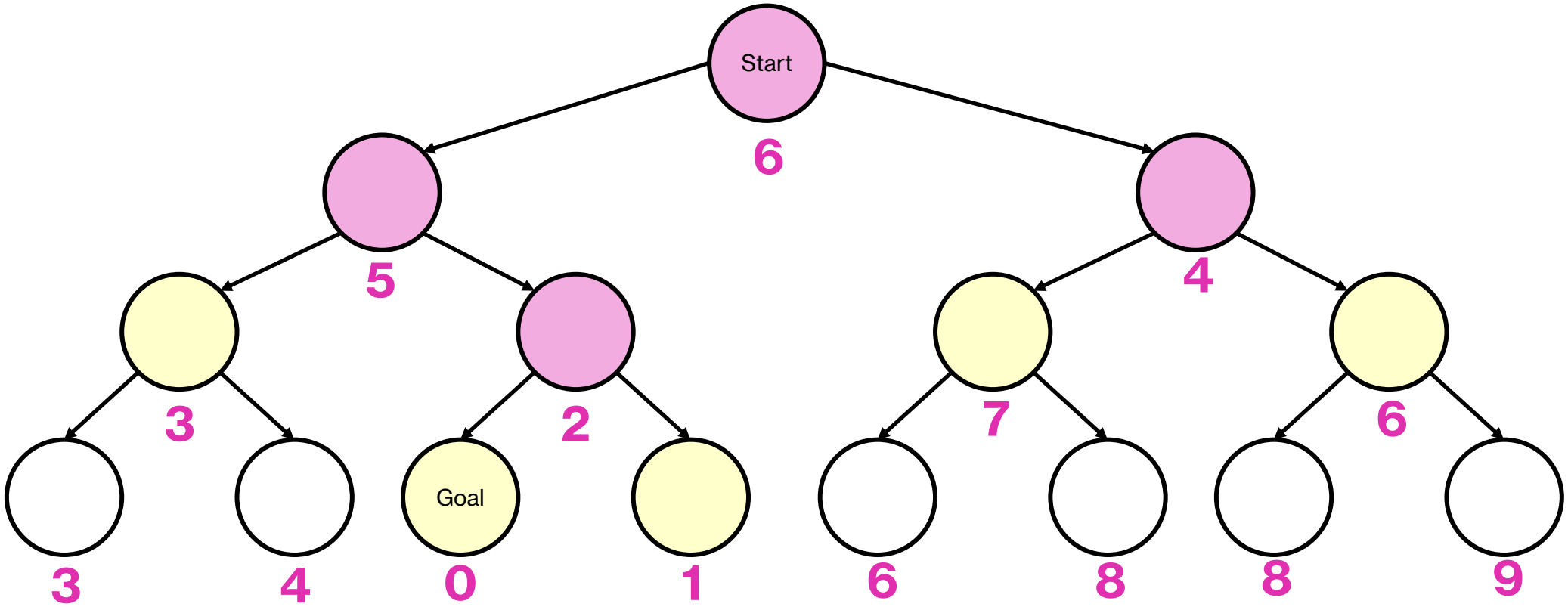
■ Heuristic value

Greedy Best First Search



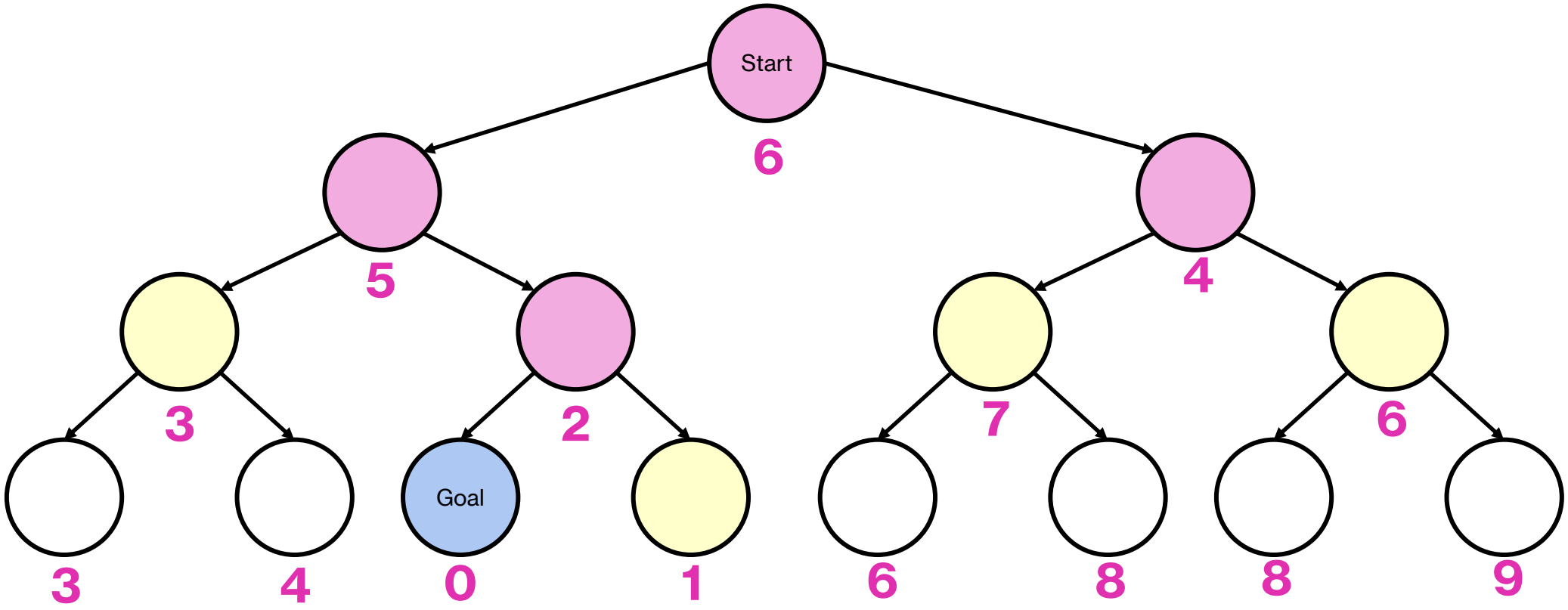
■ Heuristic value

Greedy Best First Search



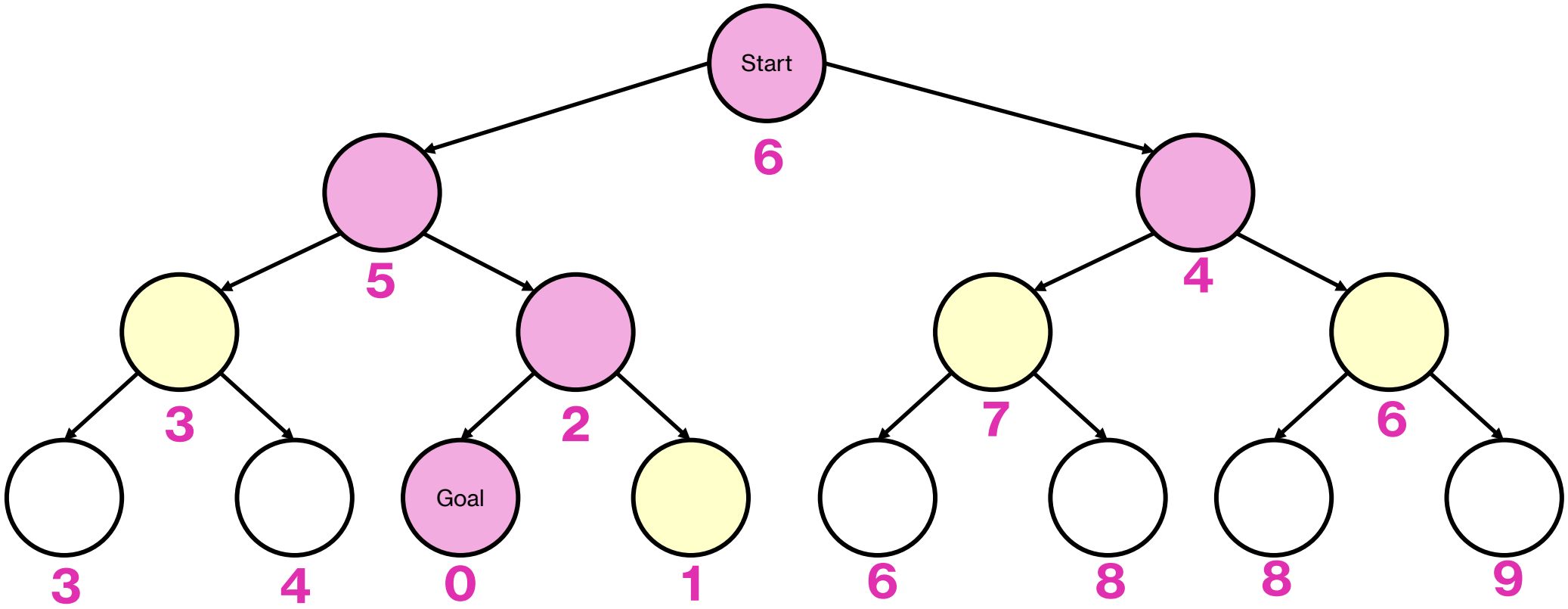
■ Heuristic value

Greedy Best First Search



■ Heuristic value

Greedy Best First Search



Characteristics of Greedy Best First Search

- If state space is finite, it is **complete**.
- It is **not optimal**.
- Effectiveness is **dependent on heuristic function** $h(s)$.

- What happens if $h(s) = 0$?

Characteristics of Greedy Best First Search

- If state space is finite, it is **complete**.
- It is **not optimal**.
- Effectiveness is **dependent on heuristic function** $h(s)$.

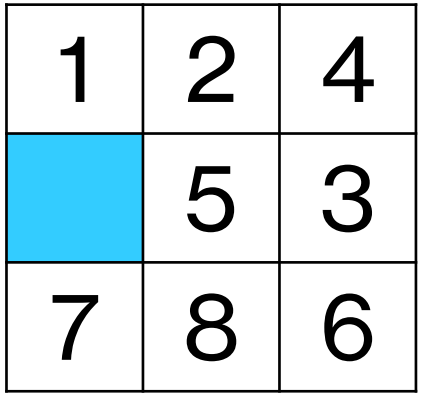
- What happens if $h(s) = 0$?
  - If $h(s) = 0$, greedy best first search **devolves into BFS or DFS**.

Why is GBFS Not Optimal?

1	2	4
	5	3
7	8	6

- How will you put 3 in the right place from this state?

Why is GBFS Not Optimal?



- How will you put 3 in the right place from this state?
- To slot 3 in the right place, the best approach is to **move 1 downwards and 2 leftwards** first to make room for the 3.
- However, these moves will result in worse heuristic values, so Greedy Best First will not prioritize those moves!

A* Search

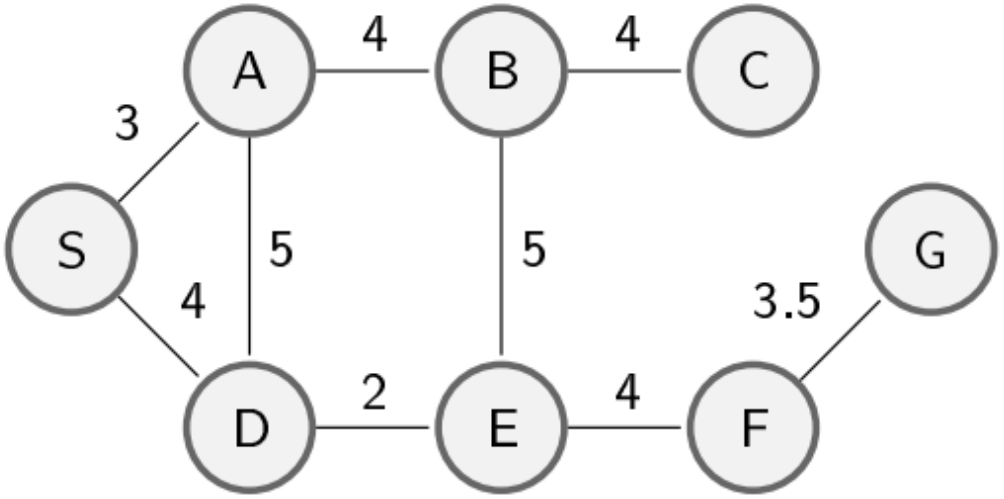
- **Key Idea:** In choosing the next state to explore from the frontier, consider both the total cost from the initial state, **and the estimated cost from the current state to the goal**.
- A* search is an extension of uniform cost search.

A* Search

ALGORITHM

Add s_{start} to FRONTIER with priority $h(s_{start})$.
While FRONTIER is not empty do
    Remove s with the lowest priority p from FRONTIER
    Let c be the cost up to s.
    if IsEnd(s) then
        return solution
    Add s to EXPLORED
    for each $a \in Actions(s)$ do
        Get successor $s' \leftarrow Succ(s, a)$
        if s' not yet in EXPLORED
            Update s' in FRONTIER with priority $c + Cost(s, a) + h(s')$

A* Search Example

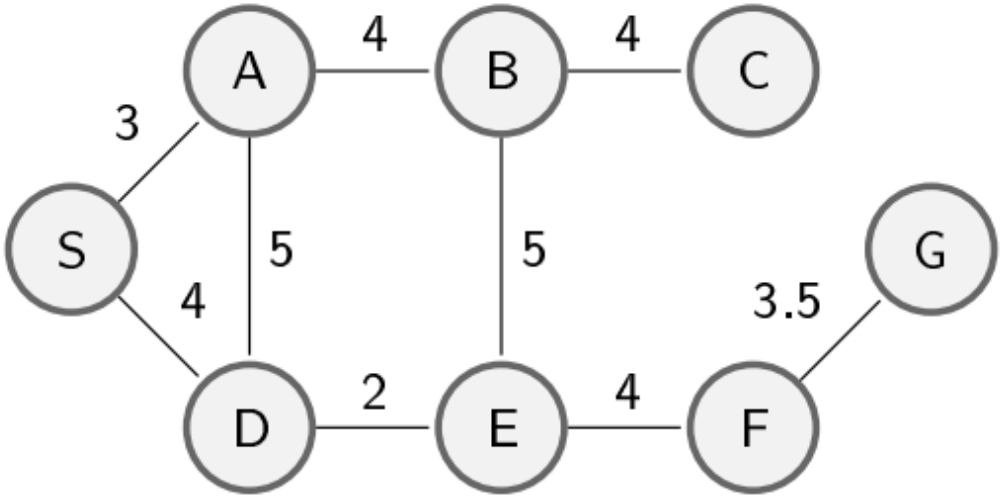


- Each node is a state.
- Start state: S
- Goal: G
- The h(s) table shows the heuristic value for each state.

$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

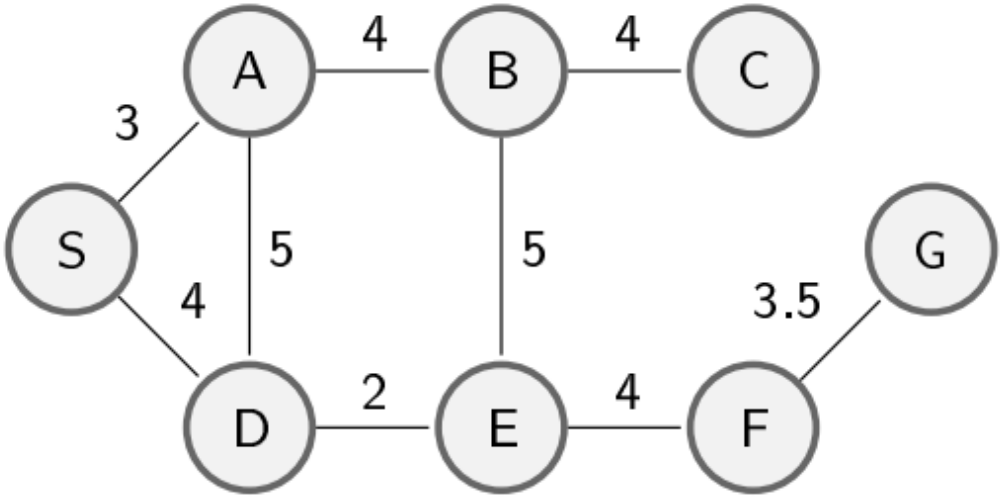


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

FRONTIER	EXPLORED
S (0/11.5)	

A* Search Example

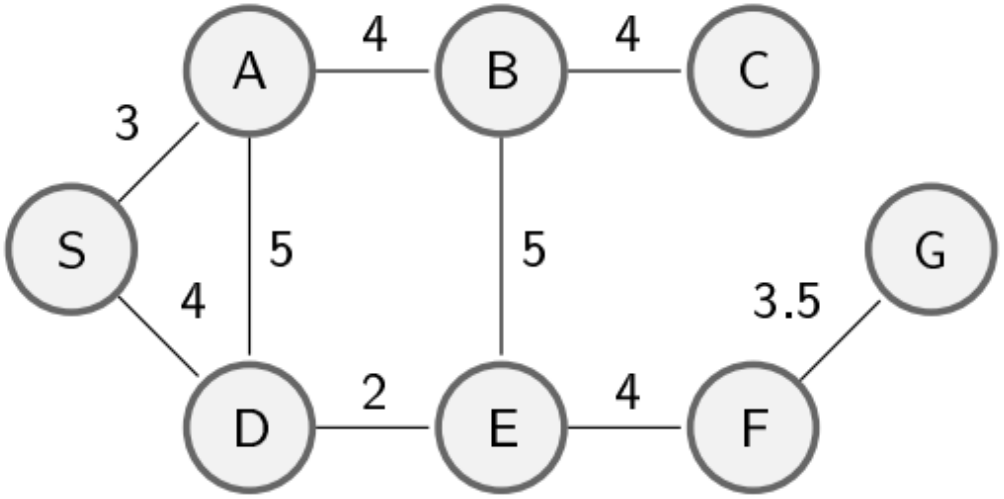


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

FRONTIER	EXPLORED
S (0/11.5)	

A* Search Example

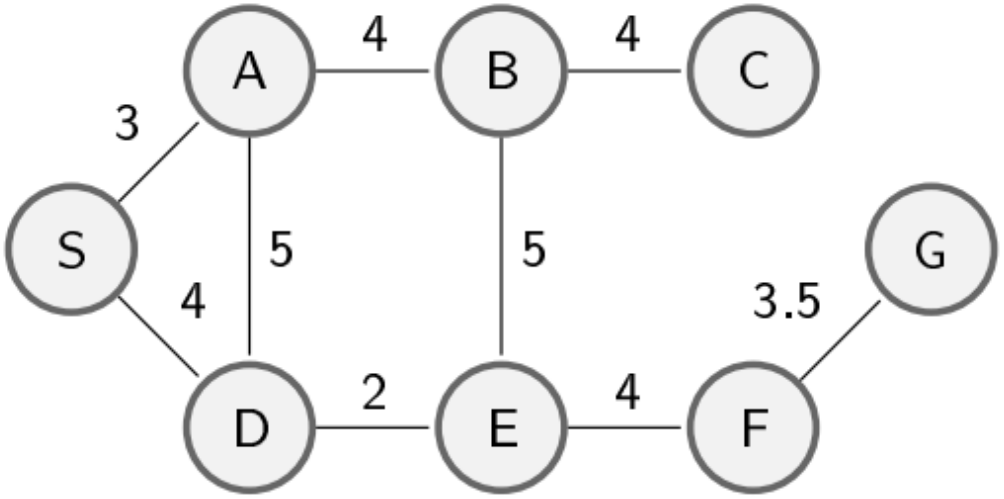


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

FRONTIER	EXPLORED
A (3/13.1) fr. S D (4/13.2) fr. S	S (0/11.5)

A* Search Example

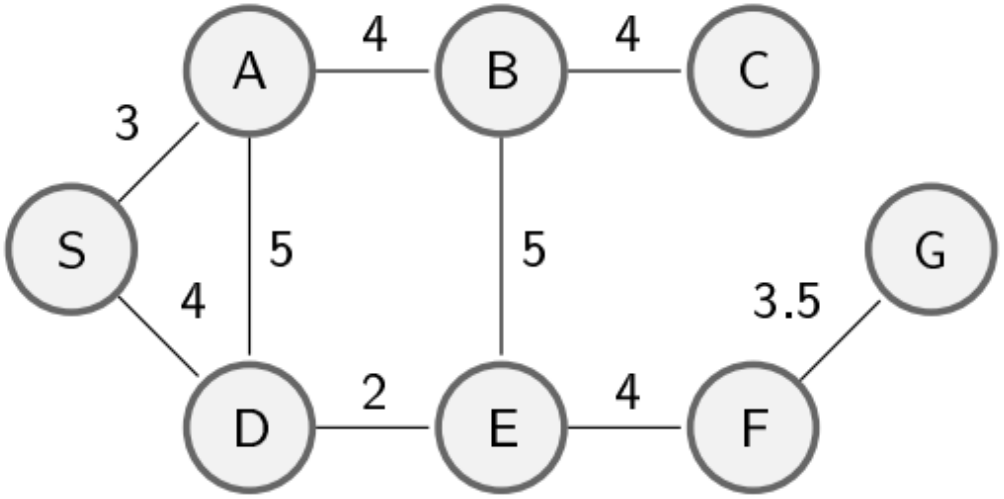


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

FRONTIER	EXPLORED
A (3/13.1) fr. S D (4/13.2) fr. S	S (0/11.5)

A* Search Example

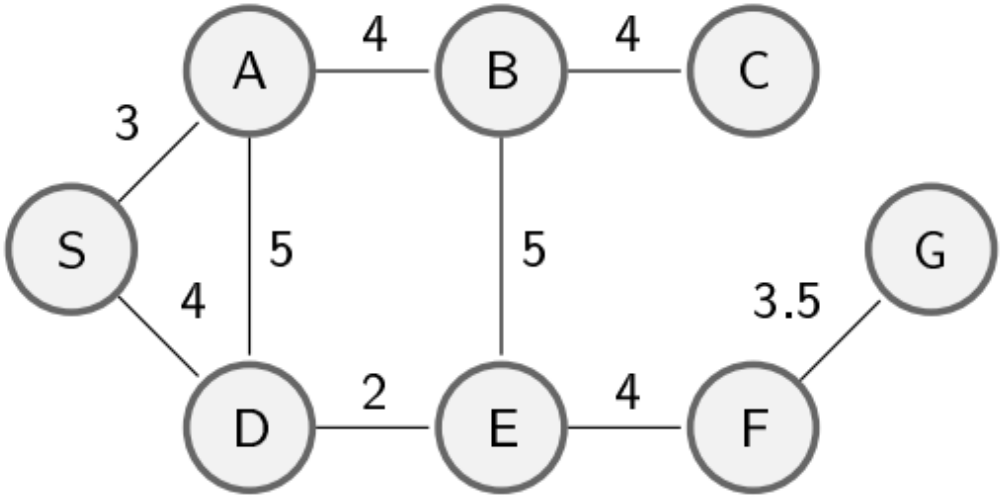


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

FRONTIER	EXPLORED
D (4/13.2) fr. S B (7/12.8) fr. A	S (0/11.5) A (3/13.5) fr. S

A* Search Example

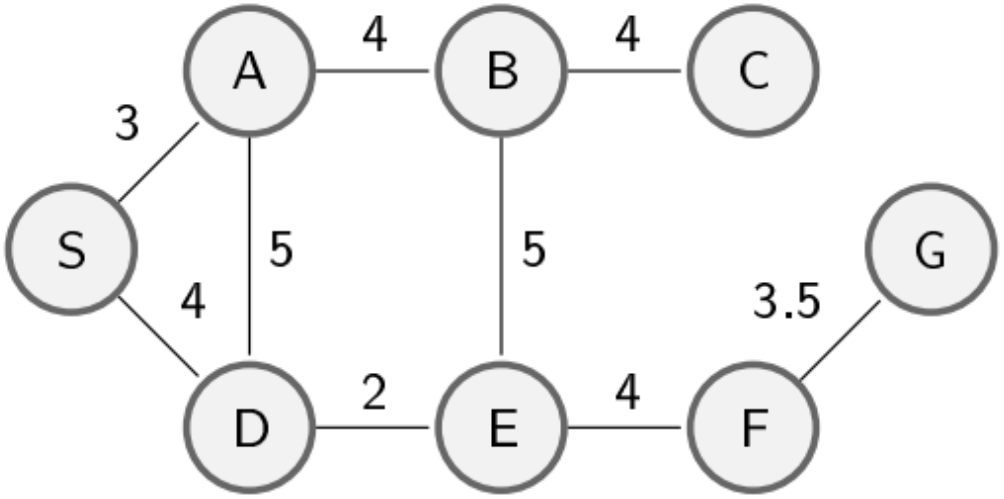


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

FRONTIER	EXPLORED
D (4/13.2) fr. S B (7/12.8) fr. A	S (0/11.5) A (3/13.5) fr. S

A* Search Example

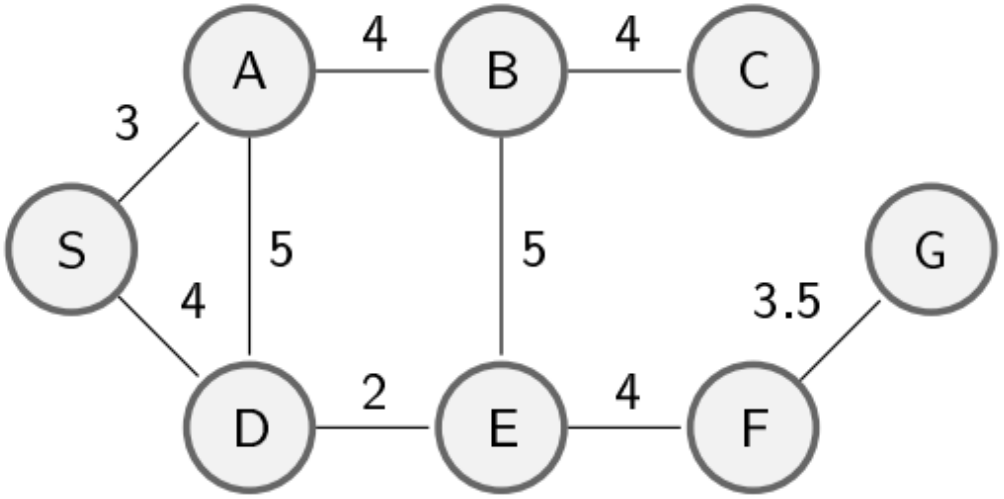


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

FRONTIER	EXPLORED
D (4/13.2) fr. S C (11/14.4) fr. B E (12/19.1) fr. B	S (0/11.5) A (3/13.5) fr. S B (7/12.8) fr. A

A* Search Example

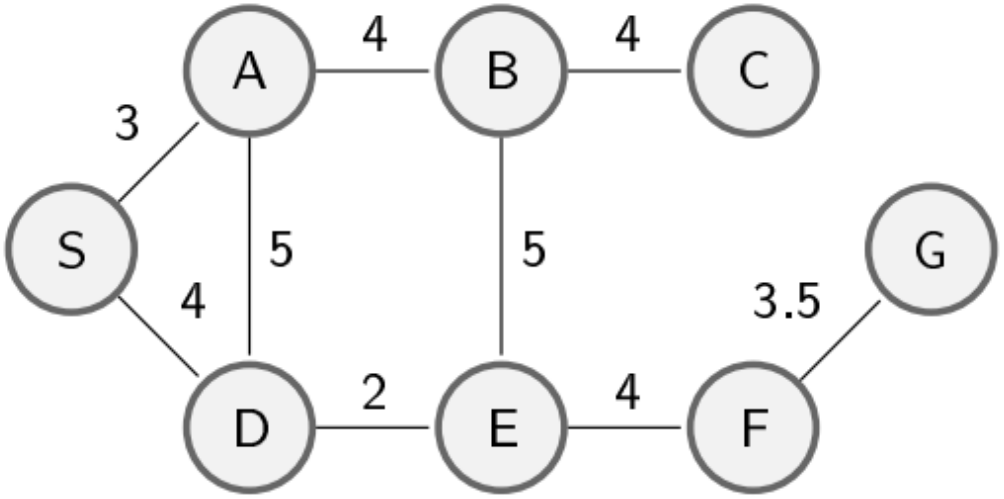


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

FRONTIER	EXPLORED
D (4/13.2) fr. S C (11/14.4) fr. B E (12/19.1) fr. B	S (0/11.5) A (3/13.5) fr. S B (7/12.8) fr. A

A* Search Example

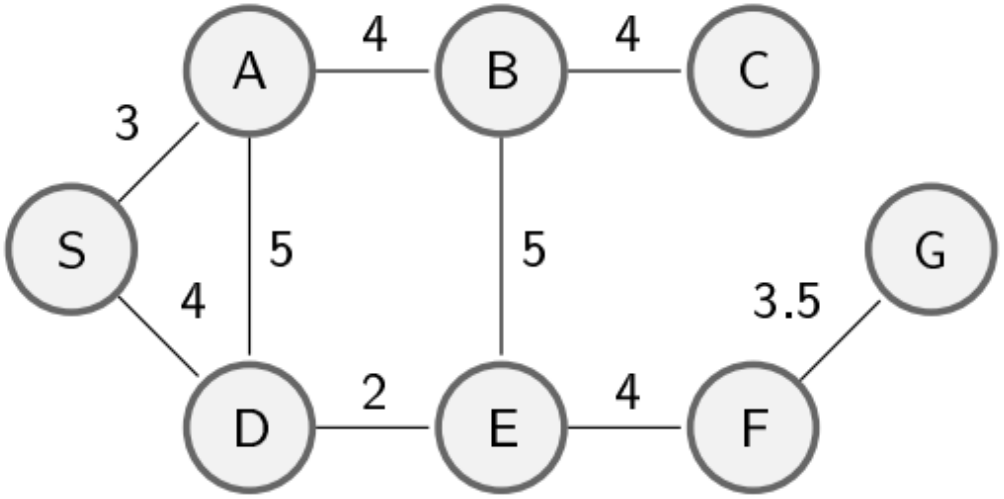


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

FRONTIER	EXPLORED
C (11/14.4) fr. B E (6/13.1) fr. D	S (0/11.5) A (3/13.5) fr. S B (7/12.8) fr. A D (4/13.2) fr. S

A* Search Example



$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

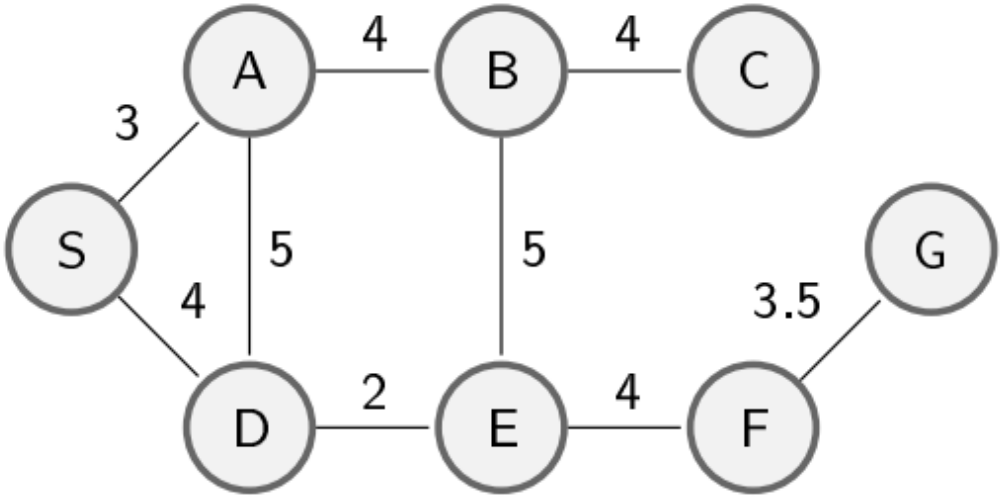
FRONTIER

C (11/14.4) fr. B
E (6/13.1) fr. D

EXPLORED

S (0/11.5)
A (3/13.5) fr. S
B (7/12.8) fr. A
D (4/13.2) fr. S

A* Search Example



$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

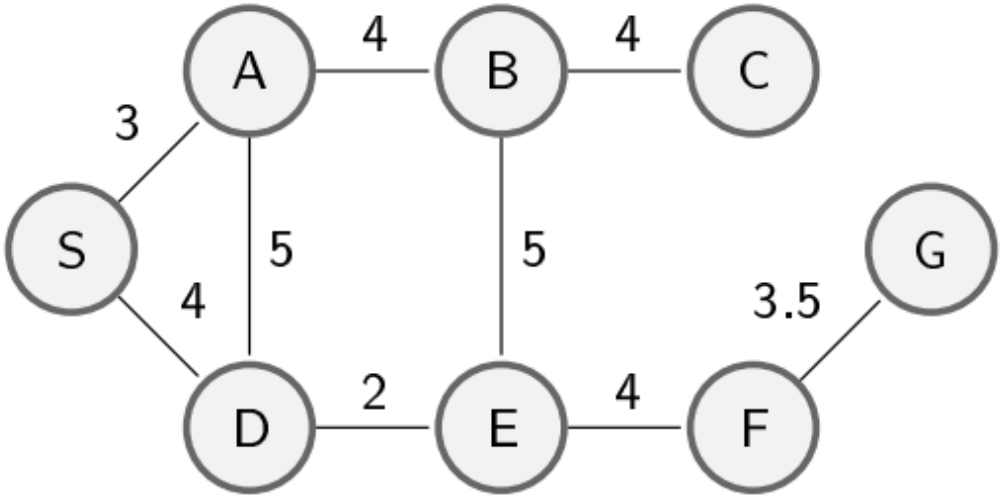
FRONTIER

C (11/14.4) fr. B
F (10/13.5) fr. E

EXPLORED

S (0/11.5)
A (3/13.5) fr. S
B (7/12.8) fr. A
D (4/13.2) fr. S
E (6/13.1) fr. D

A* Search Example



$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

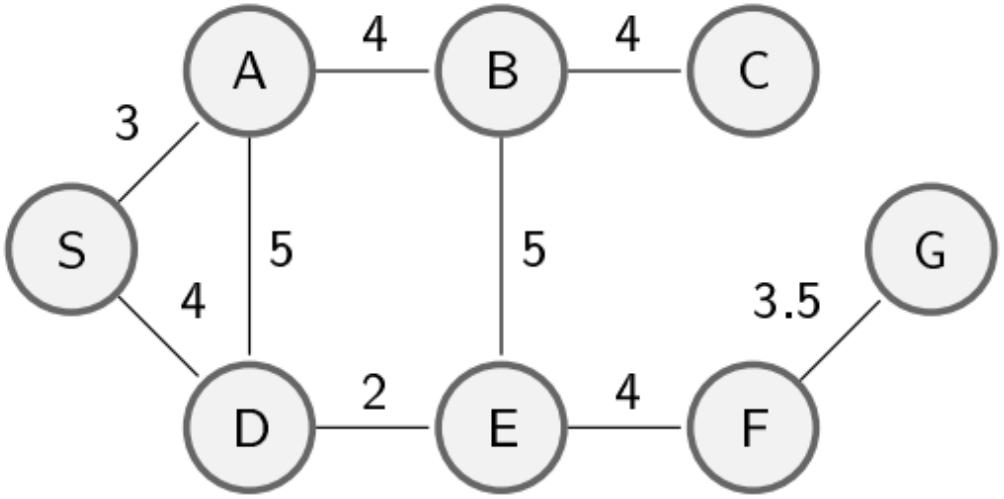
FRONTIER

C (11/14.4) fr. B
F (10/13.5) fr. E

EXPLORED

S (0/11.5)
A (3/13.5) fr. S
B (7/12.8) fr. A
D (4/13.2) fr. S
E (6/13.1) fr. D

A* Search Example



$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

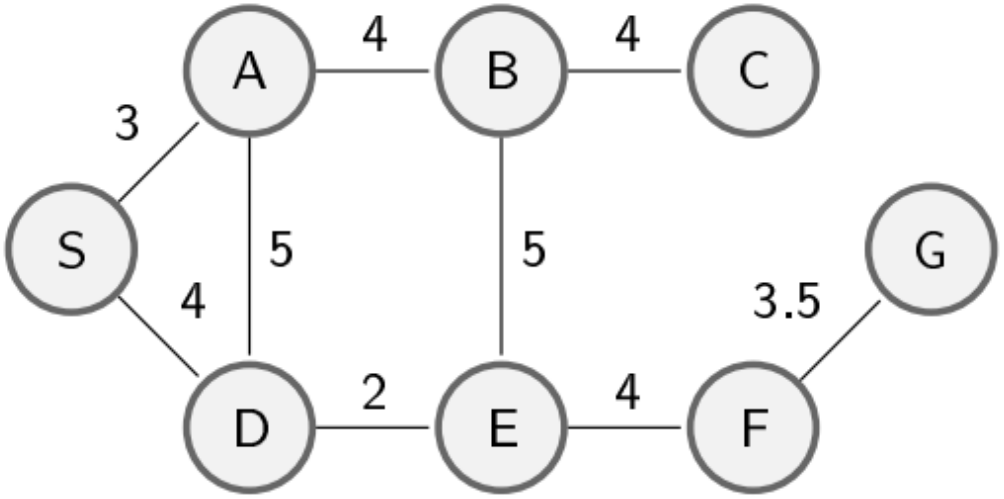
FRONTIER

C (11/14.4) fr. B
G (13.5/13.5) fr. F

EXPLORED

S (0/11.5)
A (3/13.5) fr. S
B (7/12.8) fr. A
D (4/13.2) fr. S
E (6/13.1) fr. D
F (10/13.5) fr. E

A* Search Example



$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

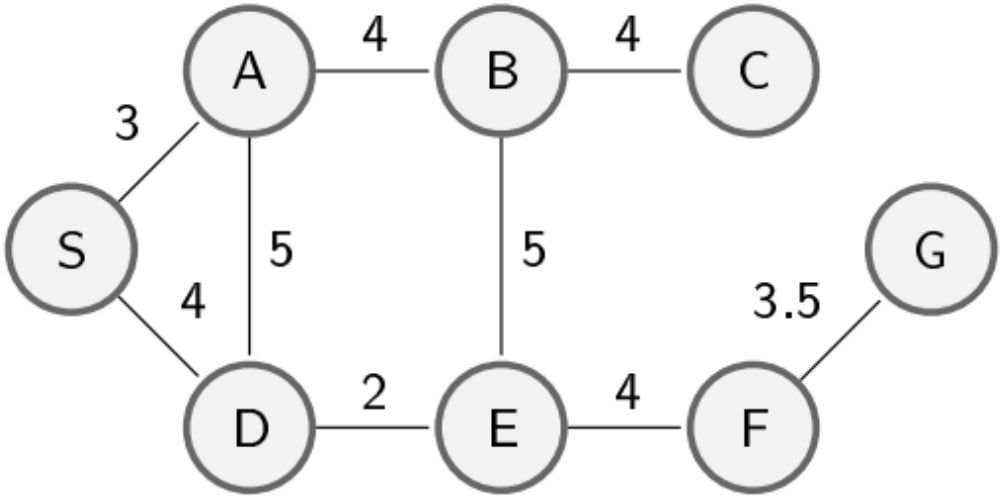
FRONTIER

C (11/14.4) fr. B
G (13.5/13.5) fr. F

EXPLORED

S (0/11.5)
A (3/13.5) fr. S
B (7/12.8) fr. A
D (4/13.2) fr. S
E (6/13.1) fr. D
F (10/13.5) fr. E

A* Search Example



$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

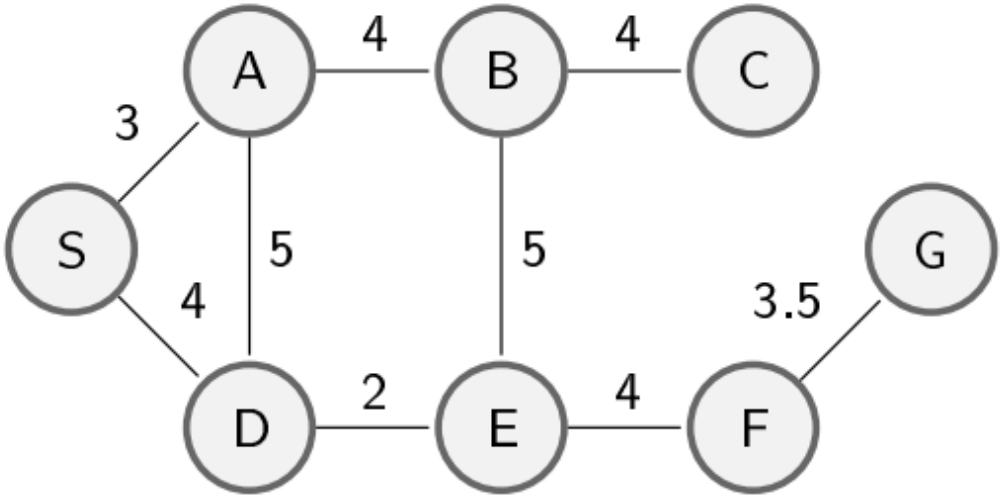
FRONTIER

C (11/14.4) fr. B

EXPLORED

S (0/11.5)
A (3/13.5) fr. S
B (7/12.8) fr. A
D (4/13.2) fr. S
E (6/13.1) fr. D
F (10/13.5) fr. E
G (13.5/13.5) fr. F

A* Search Example



$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

FRONTIER

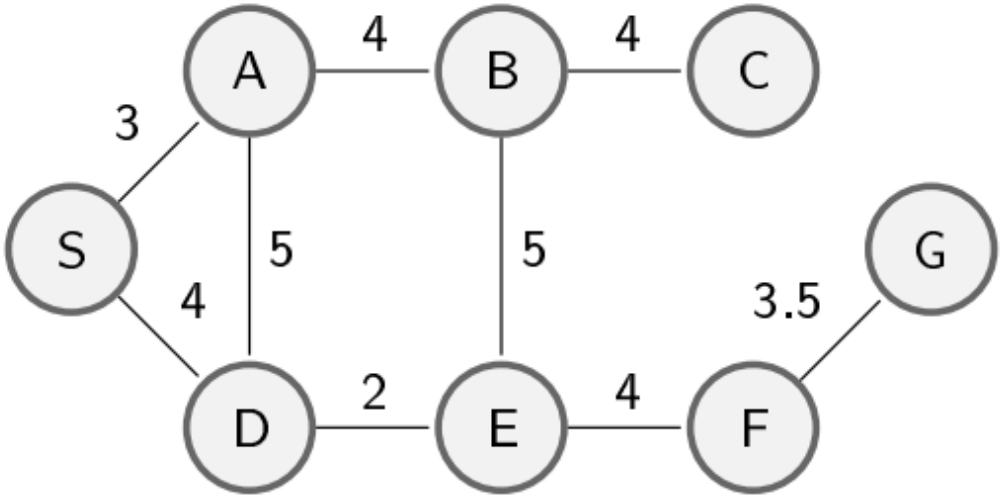
C (11/14.4) fr. B

EXPLORED

S (0/11.5)
A (3/13.5) fr. S
B (7/12.8) fr. A
D (4/13.2) fr. S
E (6/13.1) fr. D
F (10/13.5) fr. E
G (13.5/13.5) fr. F

A* Search Example

S (0/11.5)

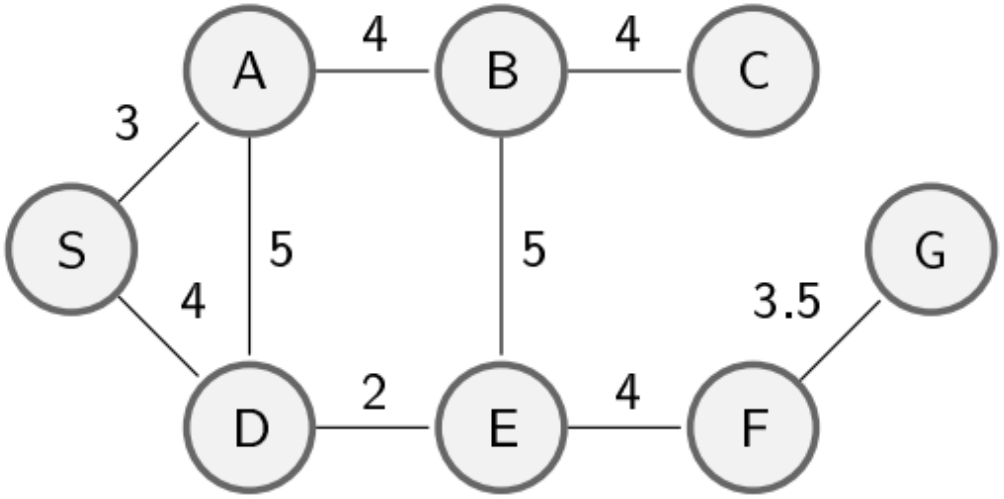


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

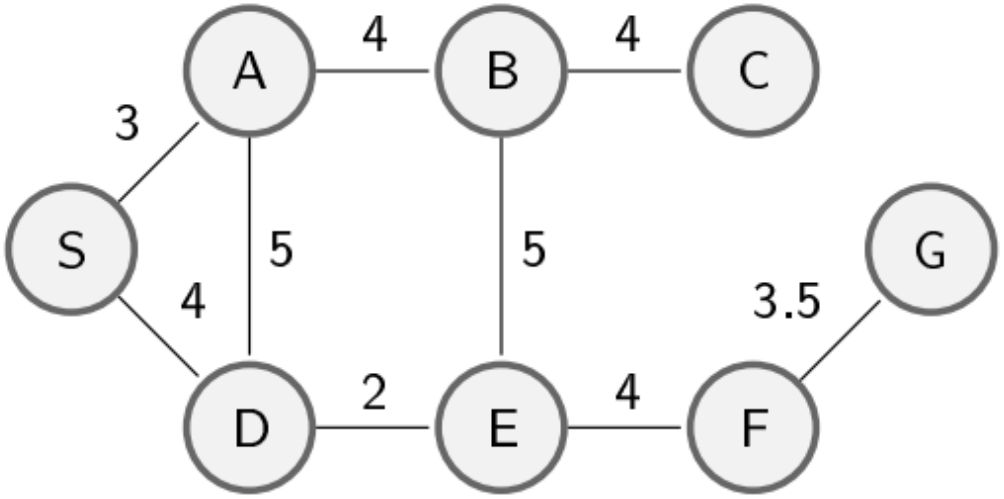
S (0/11.5)

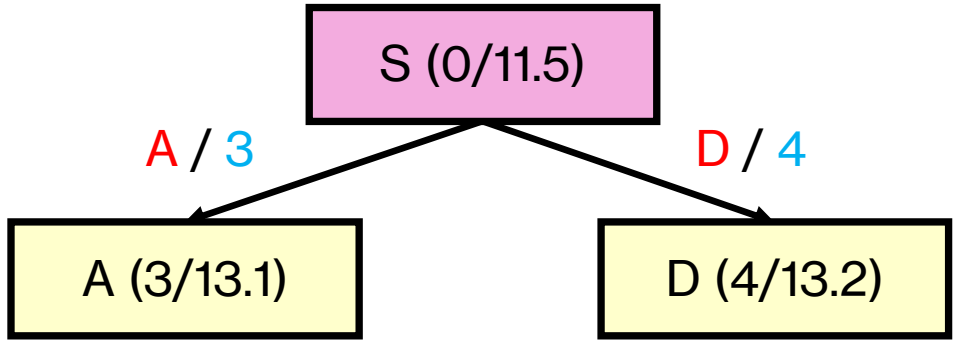


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

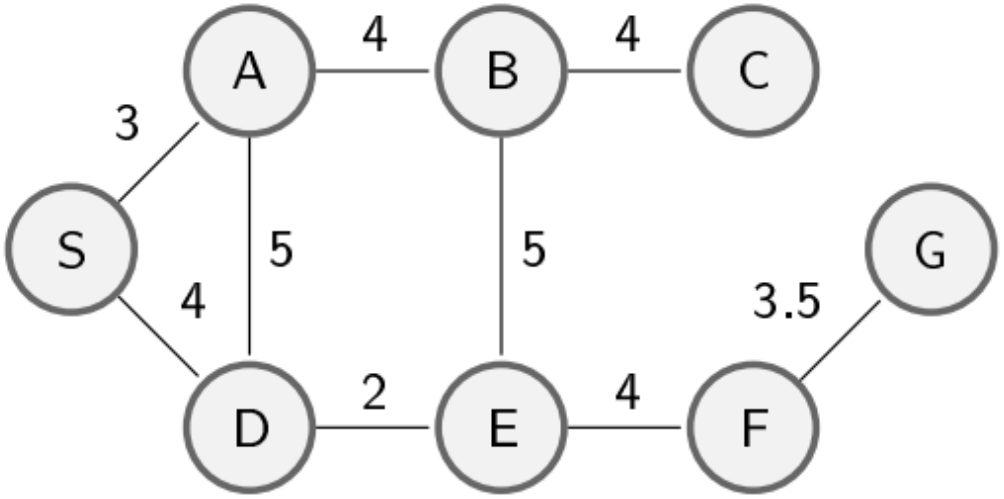


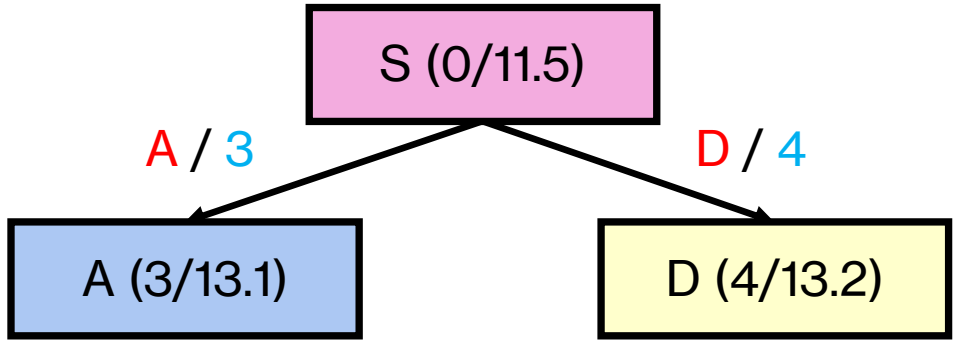


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

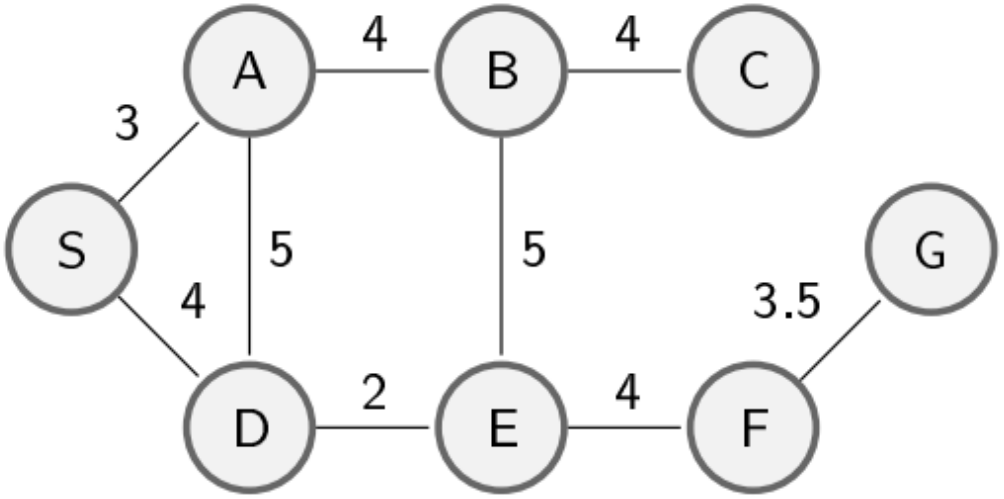


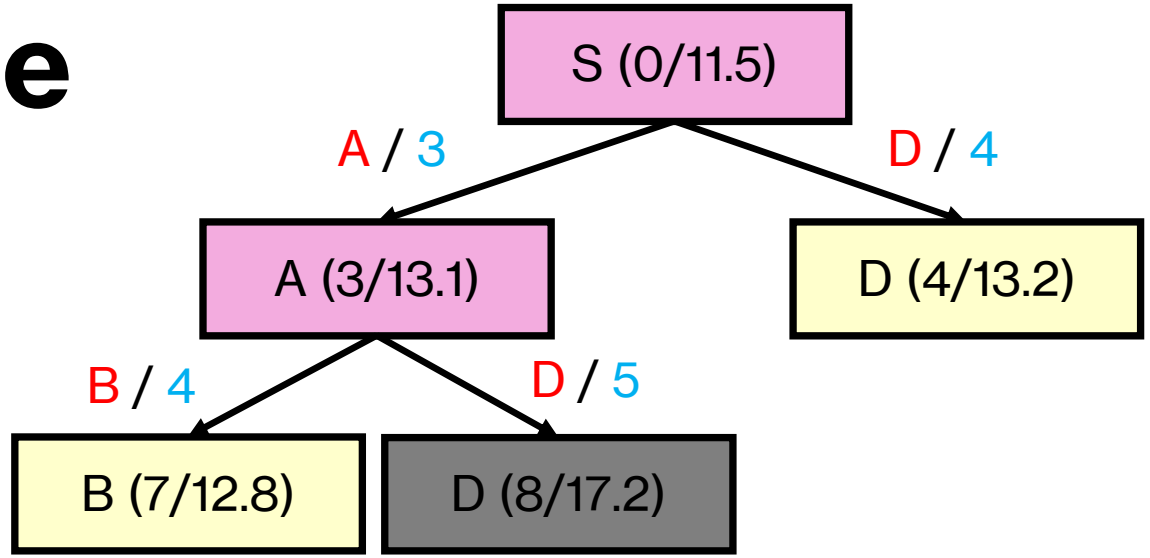


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

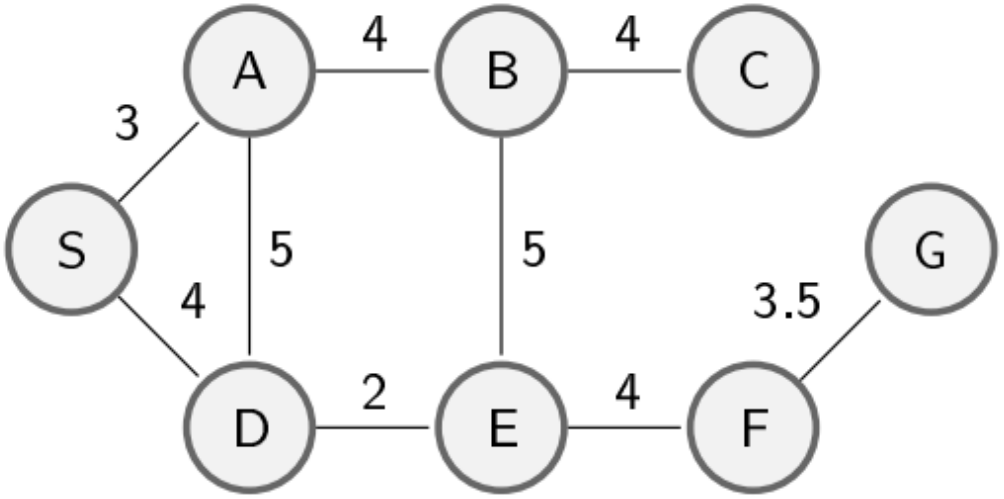


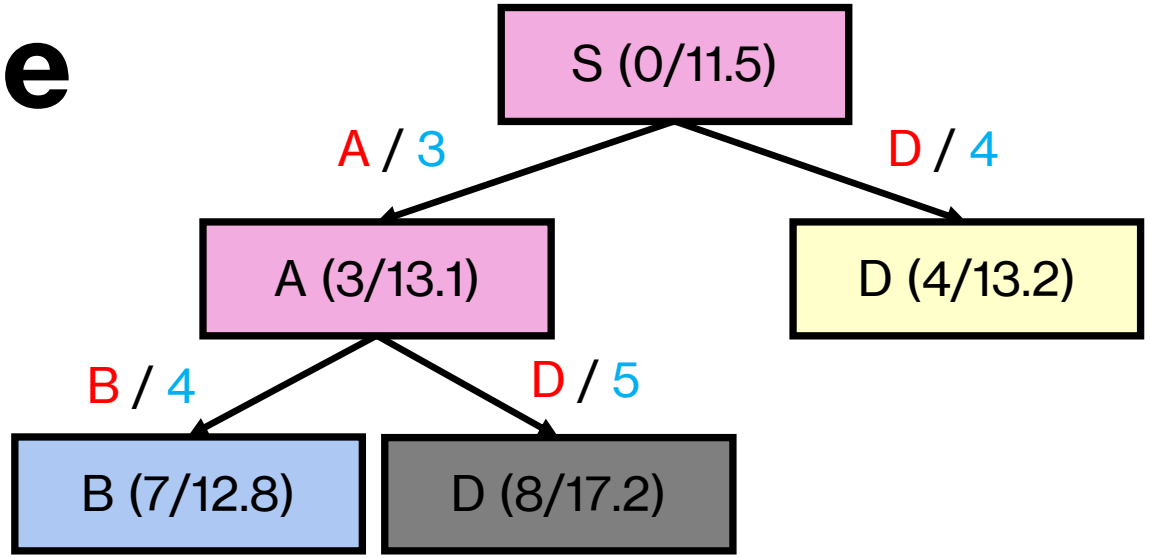


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

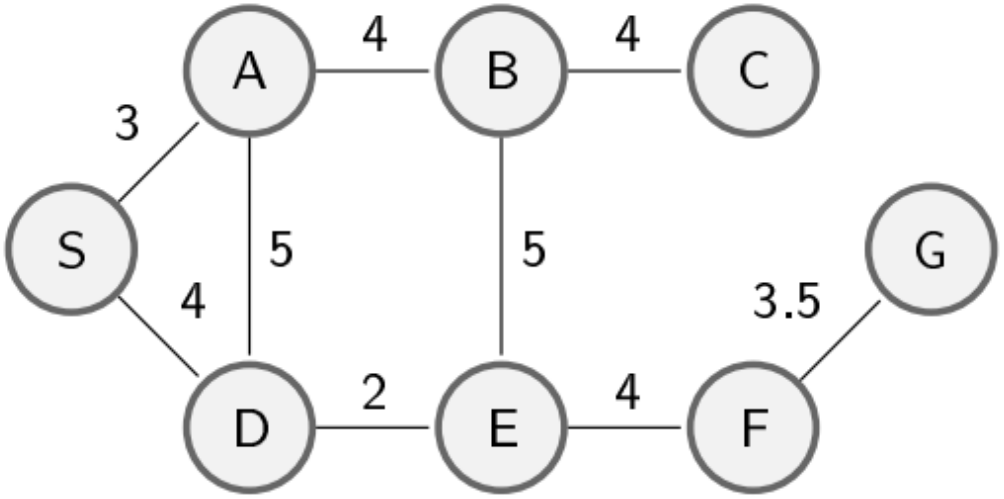


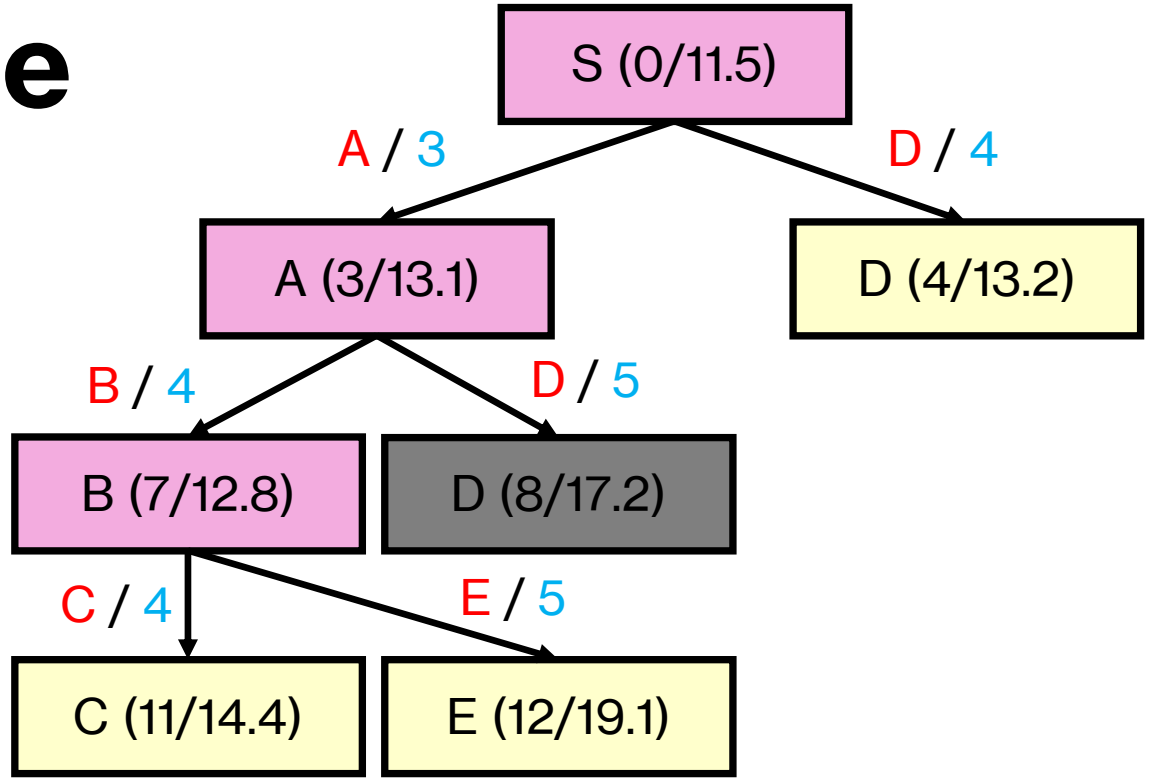


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

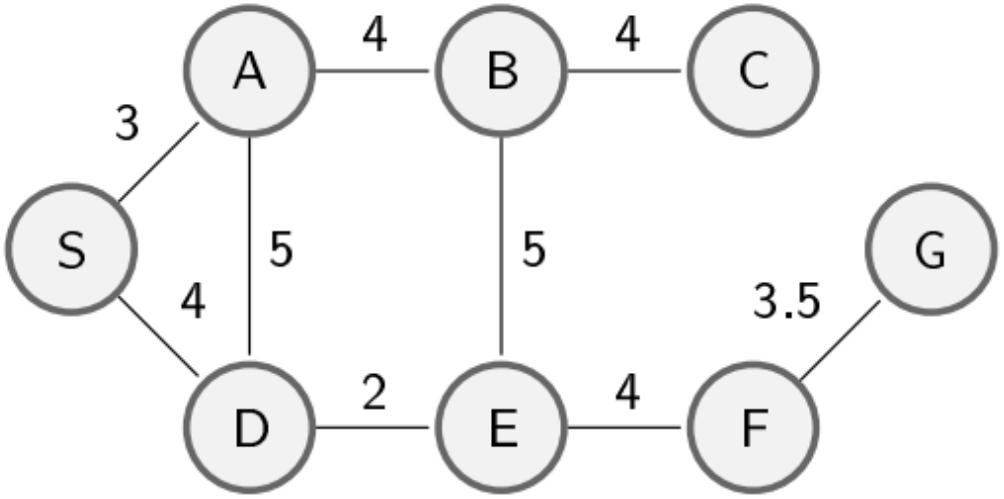


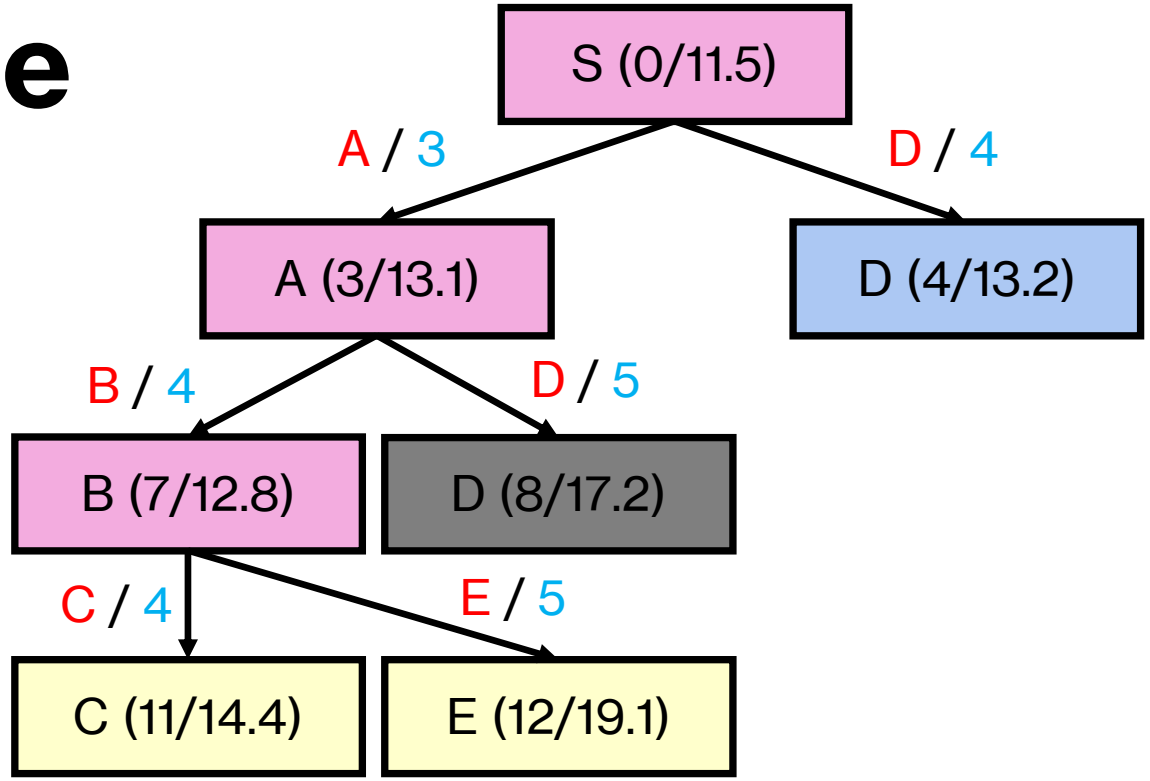


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

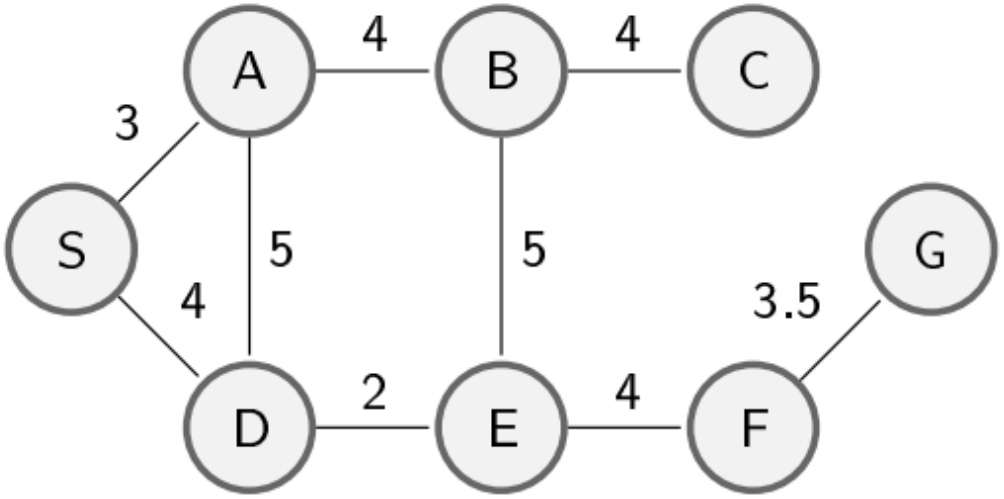


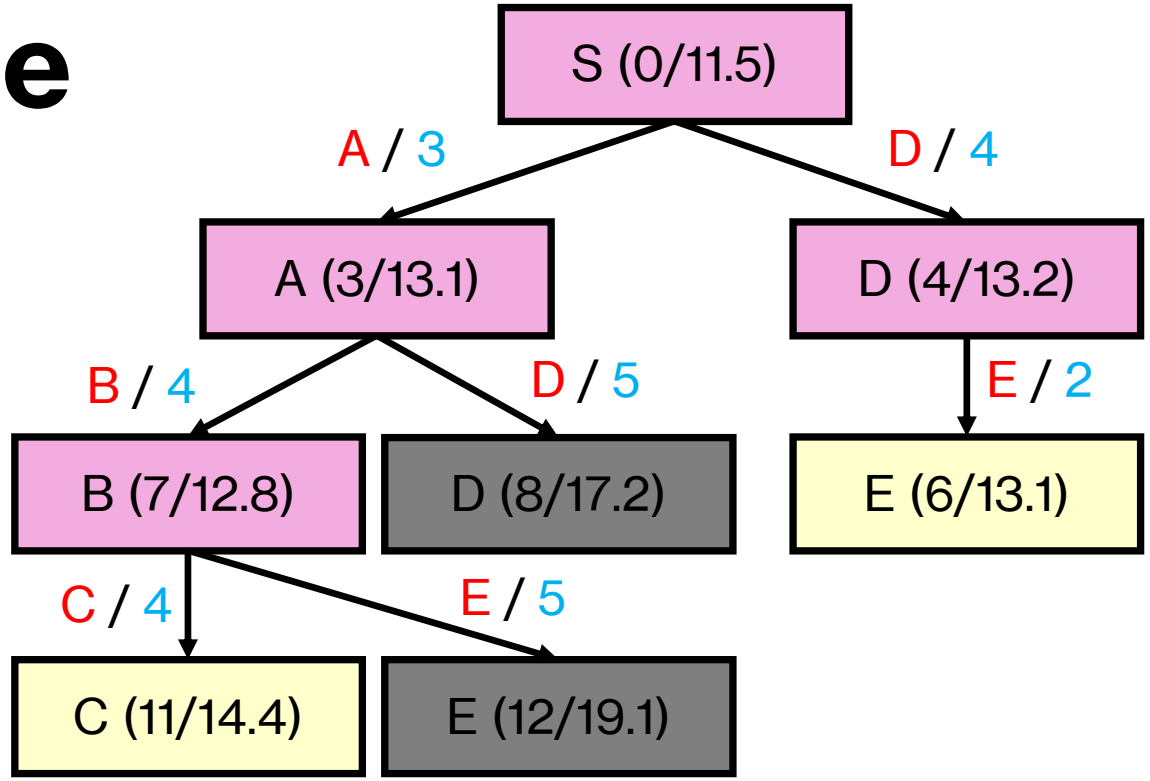


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

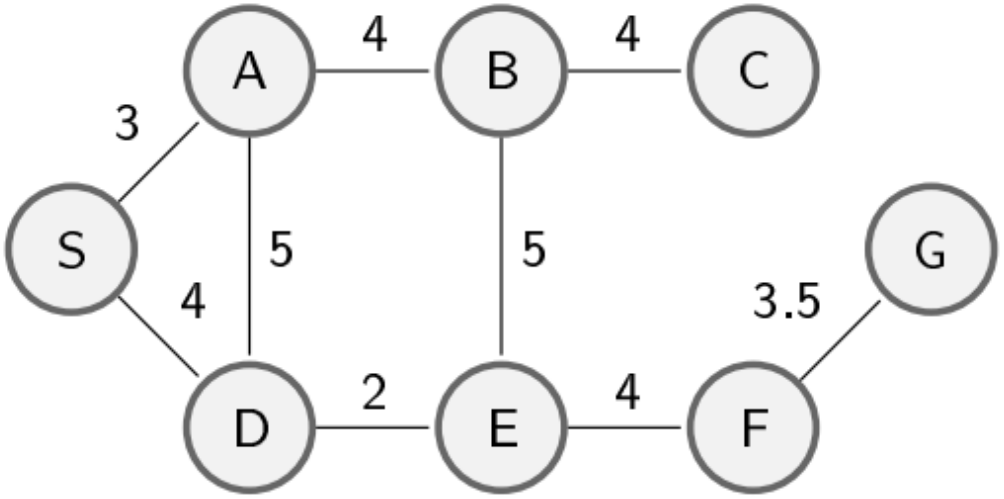


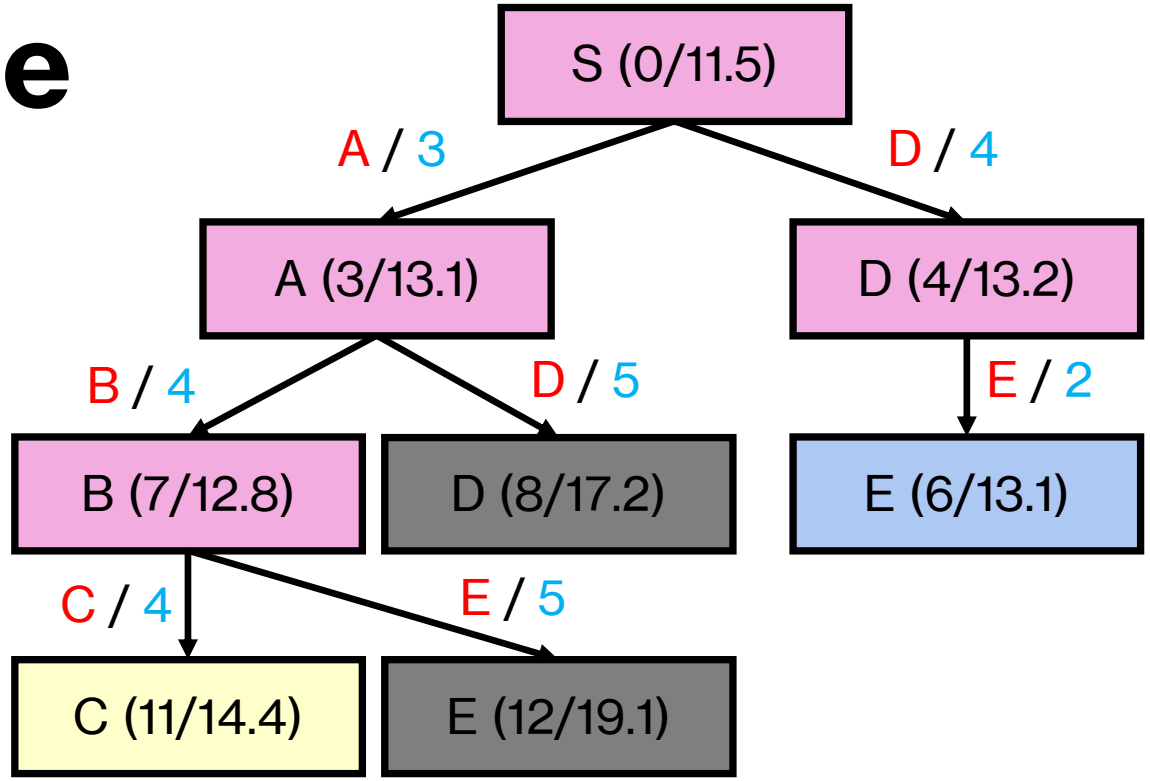


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

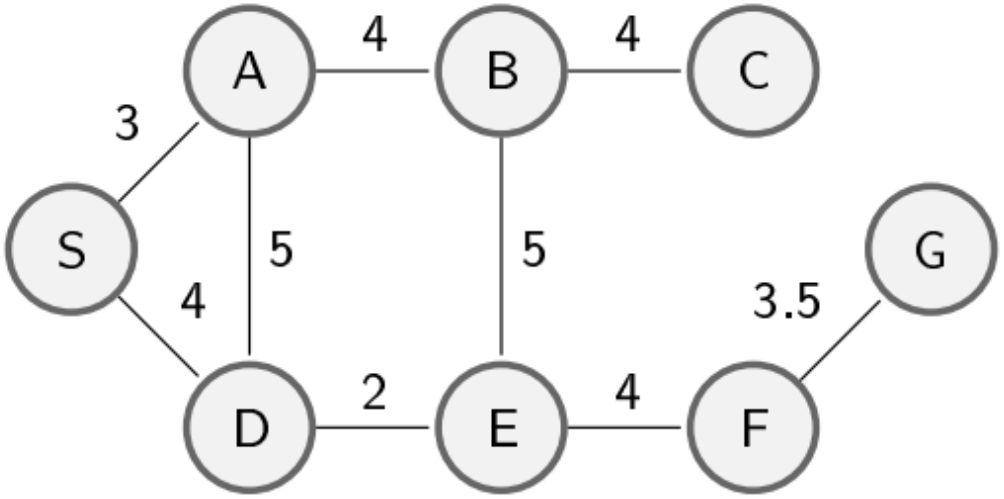


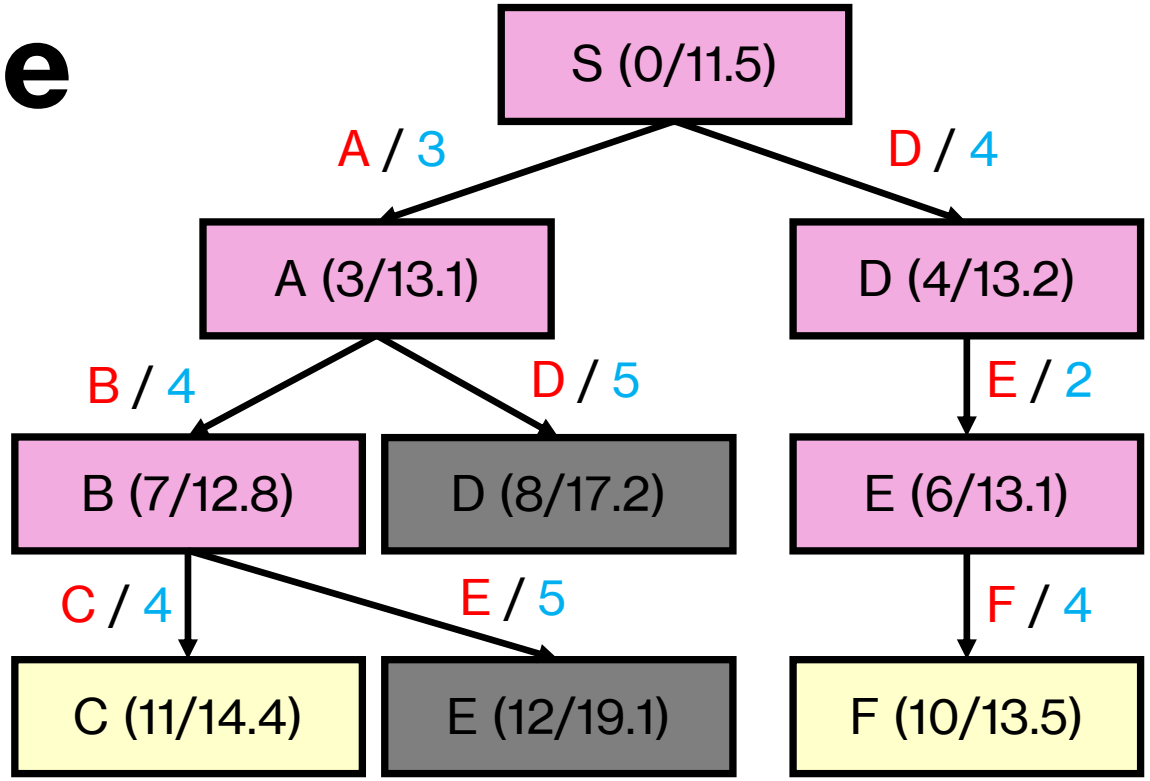


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

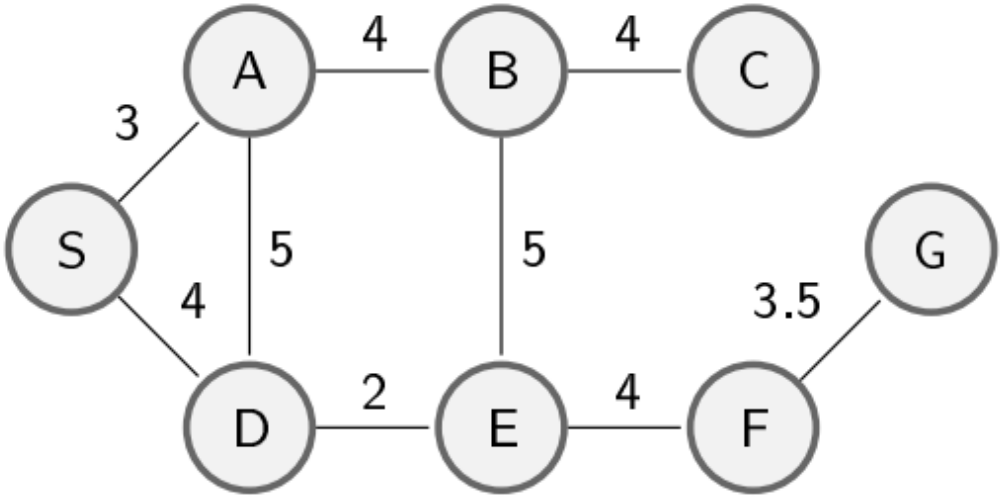


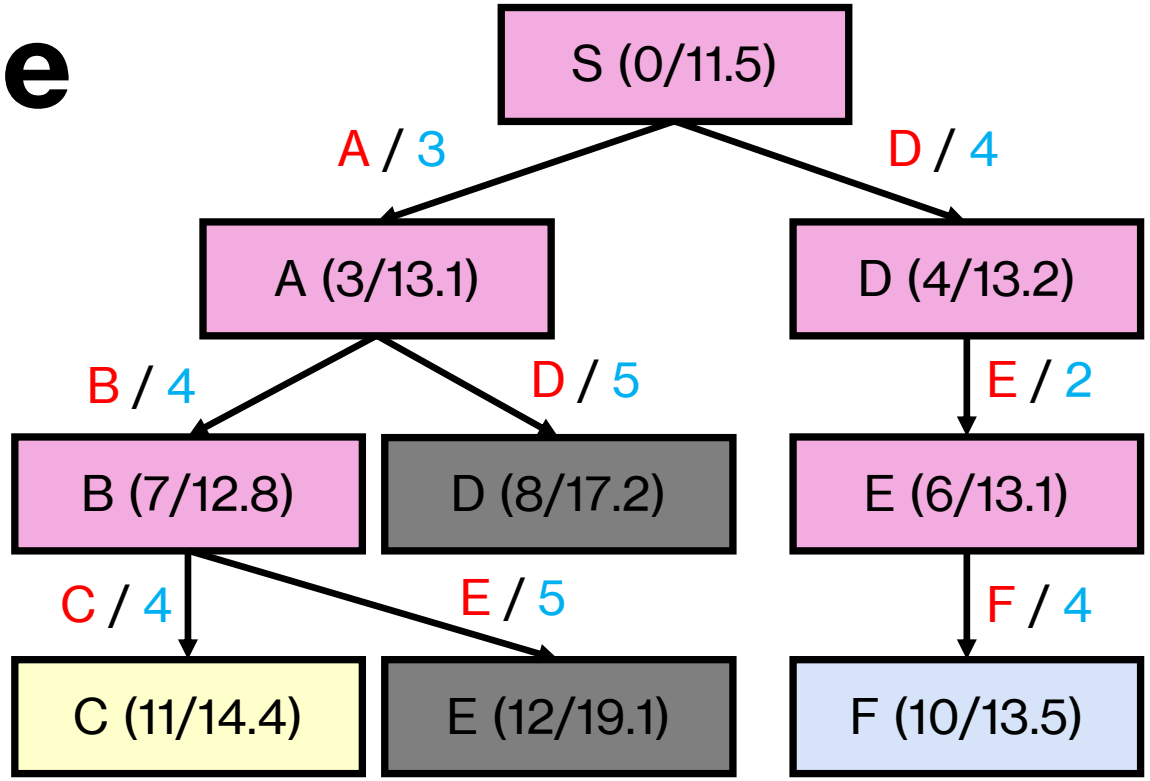


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

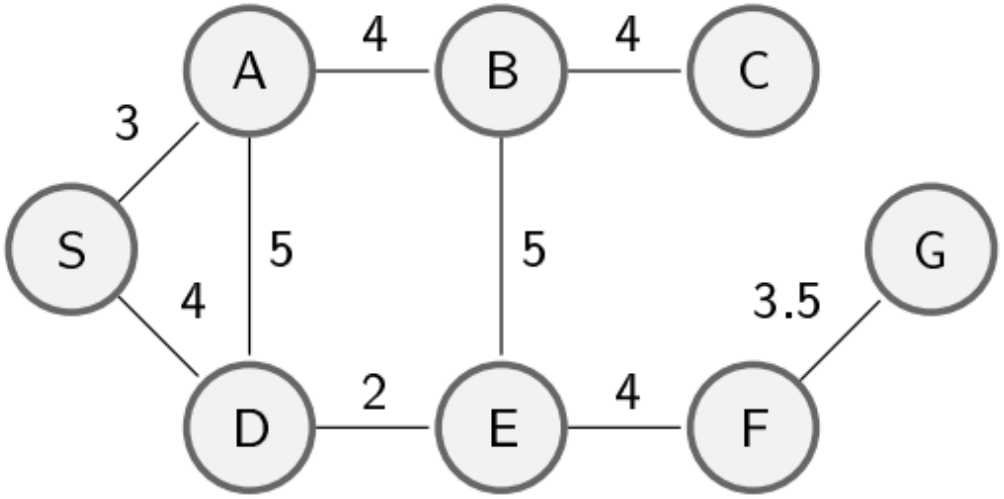


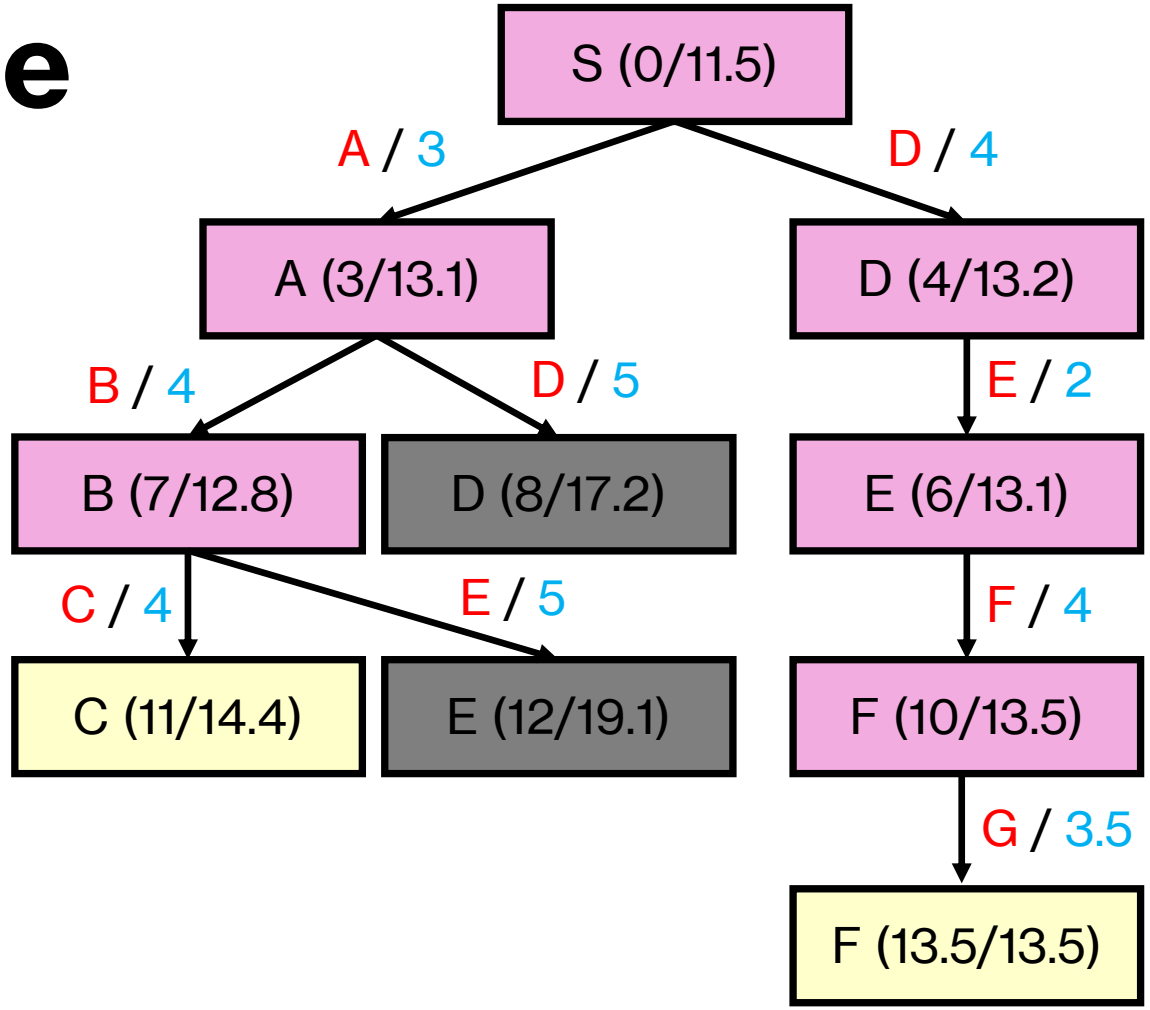


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

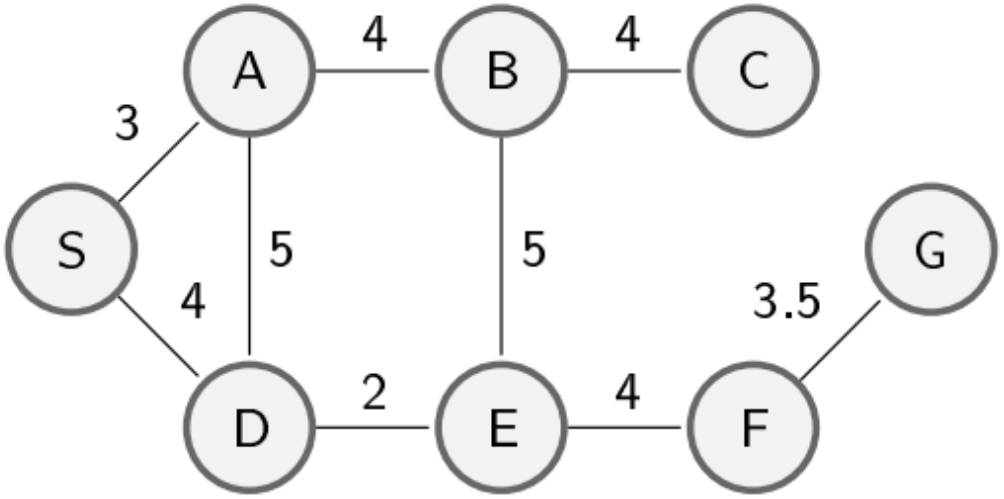


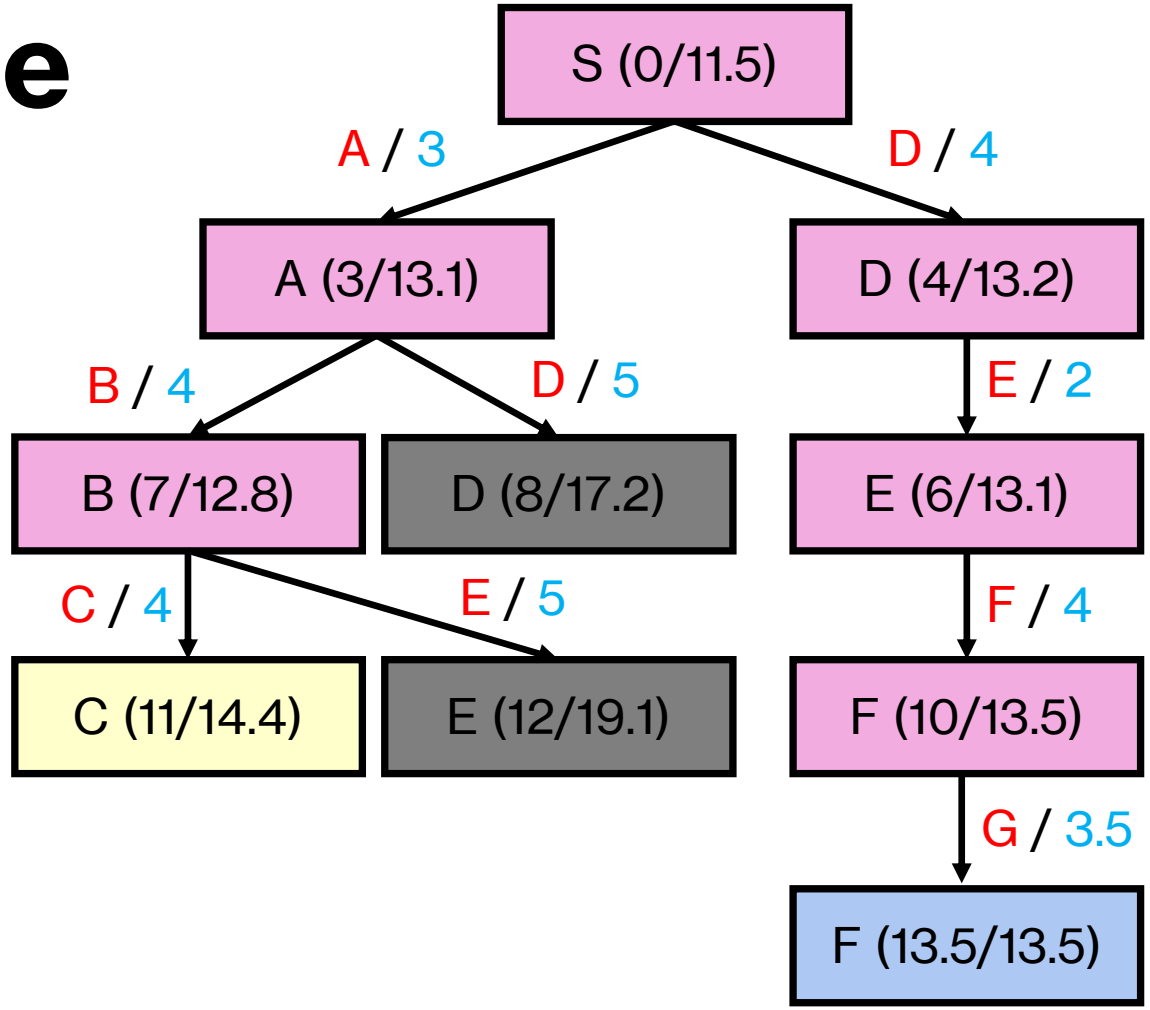


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example

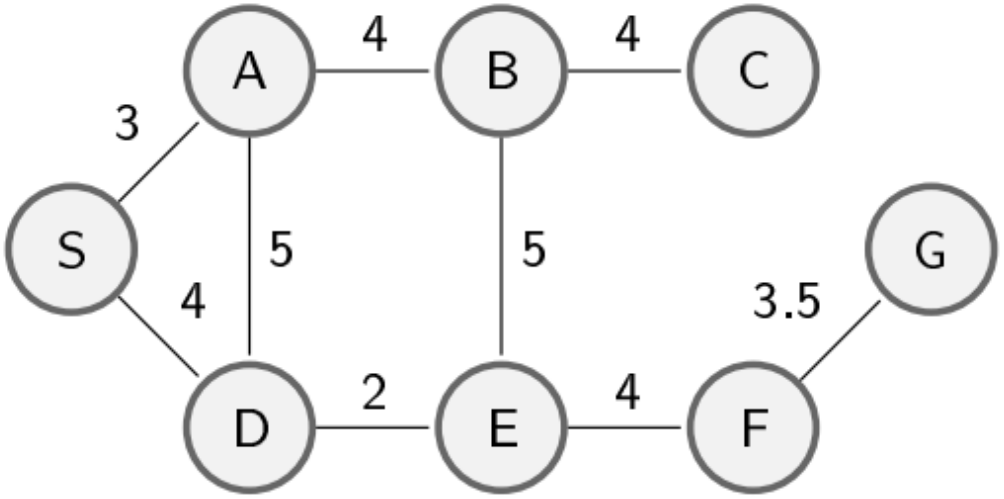


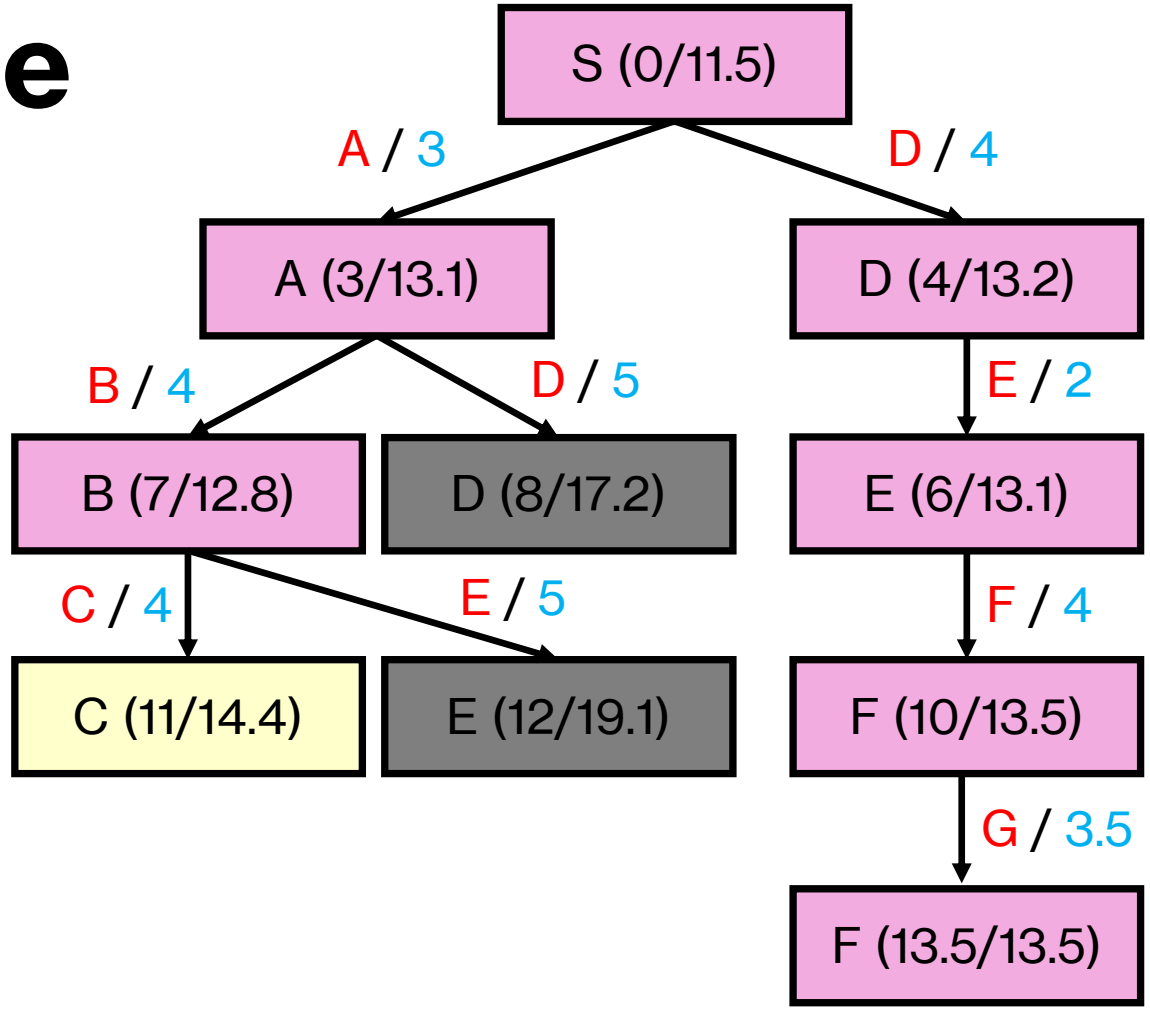


$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example



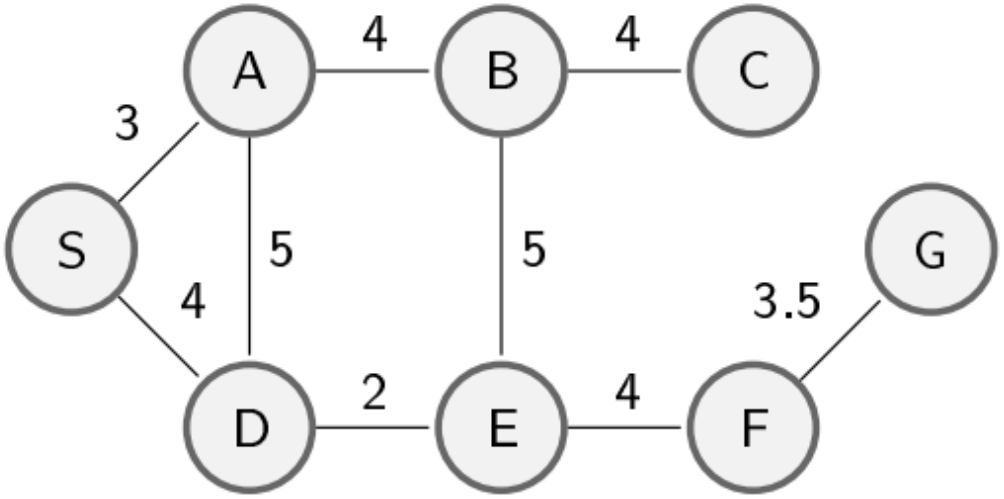


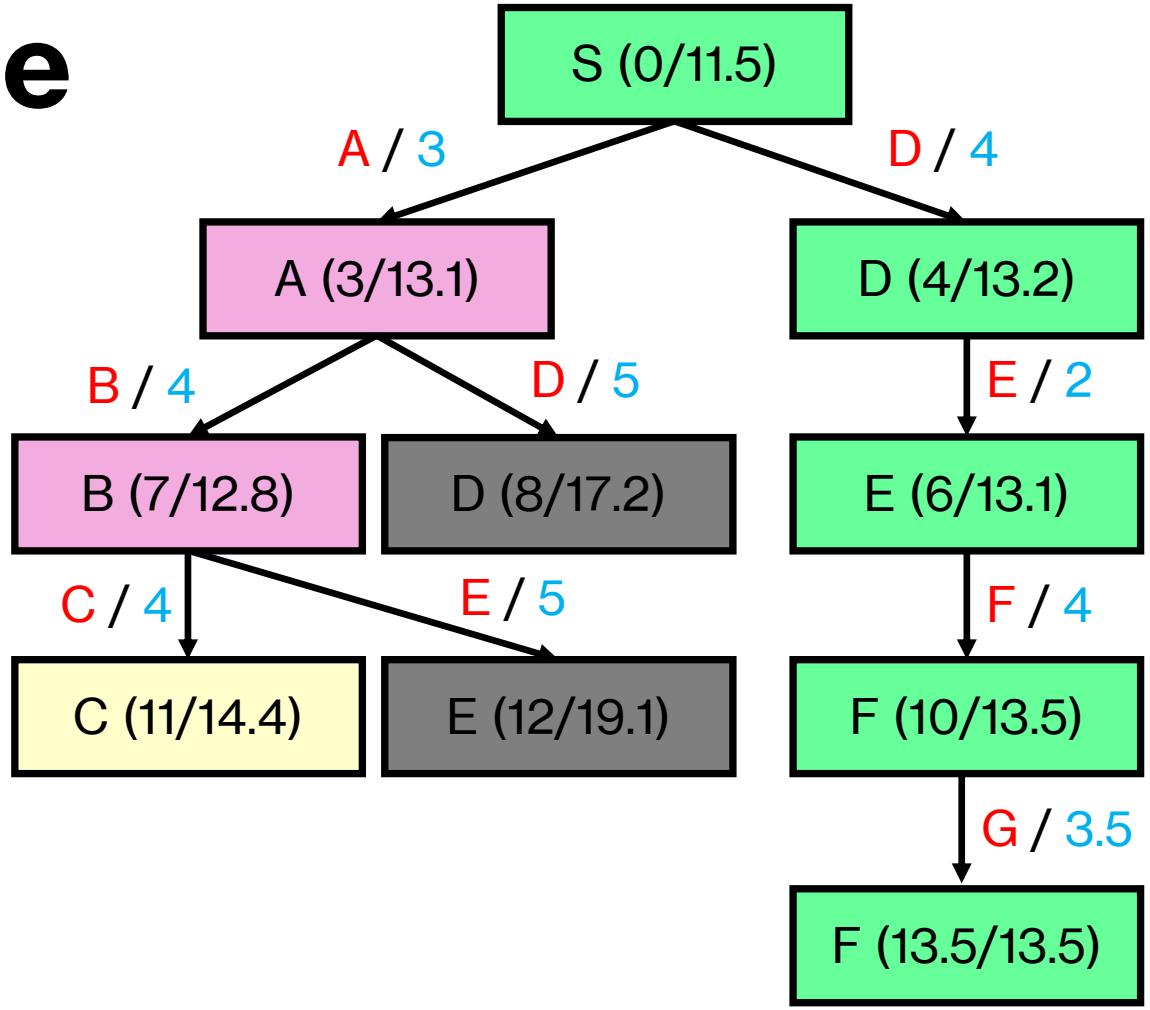
Solution Found!

$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

A* Search Example





Solution Found!

$h(s)$

S	A	B	C	D	E	F	G
11.5	10.1	5.8	3.4	9.2	7.1	3.5	0

Characteristics of A* Search

- If state space is finite, it is **complete**.
- It is **optimal** when the heuristic is **admissible** (more later).
- Performance is dependent on the heuristic function.

- What happens if $h(s) = 0$?

Characteristics of A* Search

- If state space is finite, it is **complete**.
- It is **optimal** when the heuristic is **admissible** (more later).
- Performance is dependent on the heuristic function.

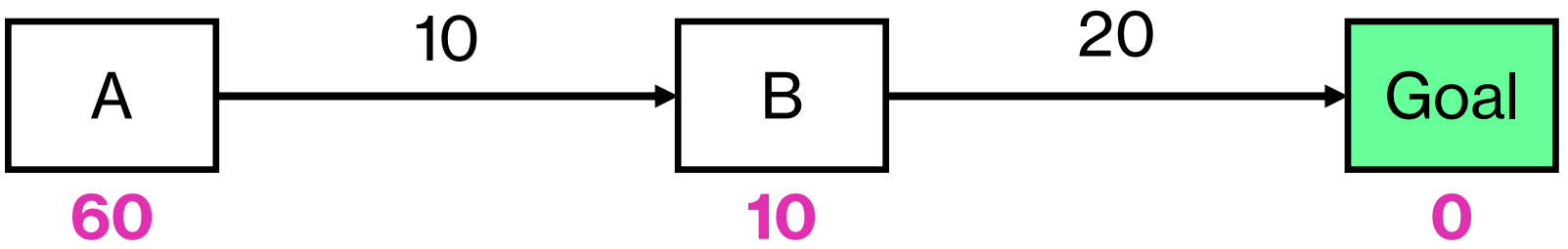
- What happens if $h(s) = 0$?
  - If $h(s) = 0$, A* search **devolves into uniform cost search**.

Admissible Heuristic

- A heuristic function $h(s)$ is **admissible** if:
  - for every node s, $h(s) \leq h^*(s)$, where $h^*(s)$ is the true optimal cost to reach the goal (actual cost).

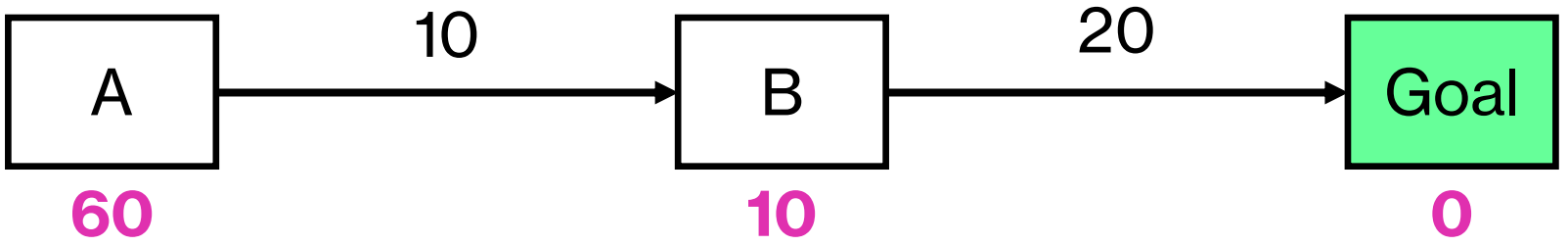
- **Intuition:** the heuristic should never overestimate the real optimal cost to the goal.

Admissible or Not Admissible?



■ Heuristic value

Admissible or Not Admissible?



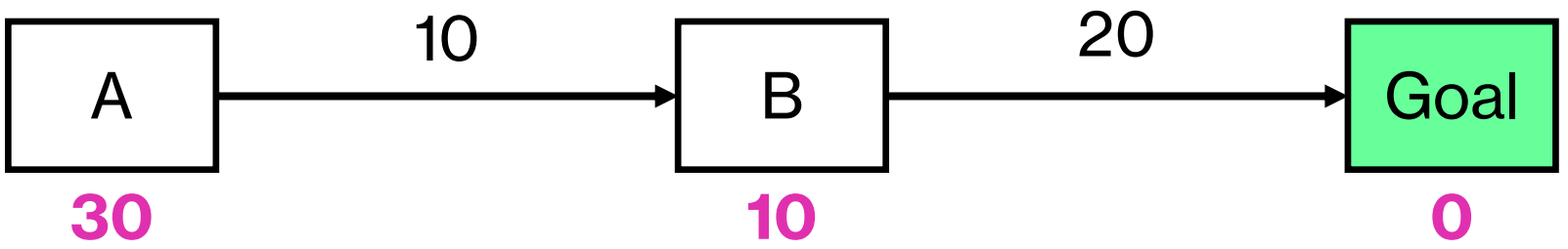
$h(s) \leq h^*(s)$

h(A) is 60 and is more than 10+20=30 → Not admissable

Not admissible! (from A)

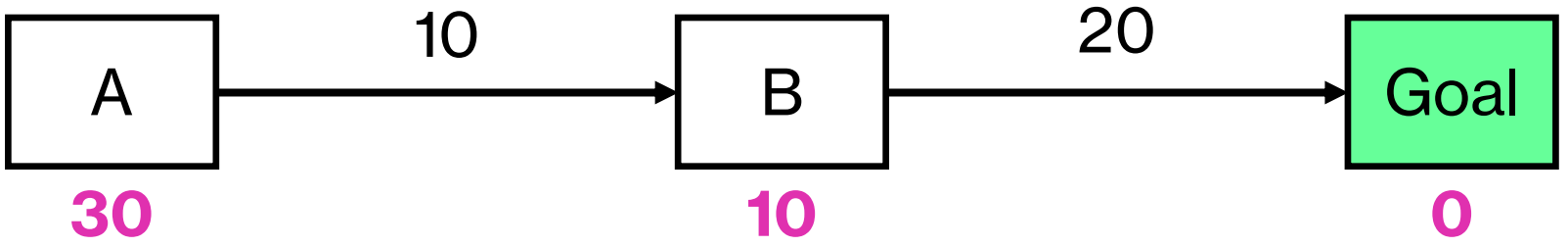
■ Heuristic value

Admissible or Not Admissible?



■ Heuristic value

Admissible or Not Admissible?



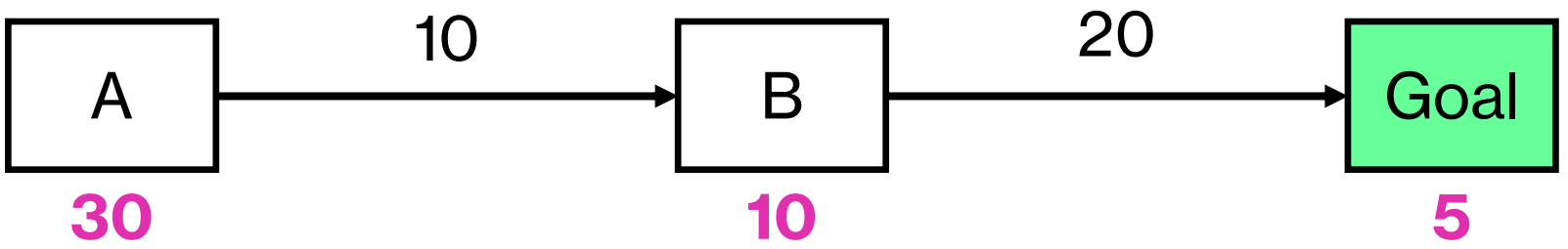
$h(s) \leq h^*(s)$

h(A) is 30 and is less than 10+20=30 → Admissable

Admissible!

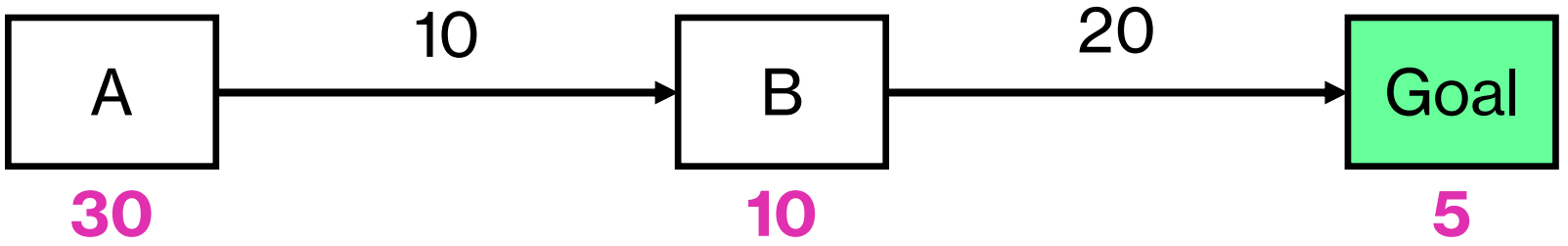
■ Heuristic value

Admissible or Not Admissible?



■ Heuristic value

Admissible or Not Admissible?



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

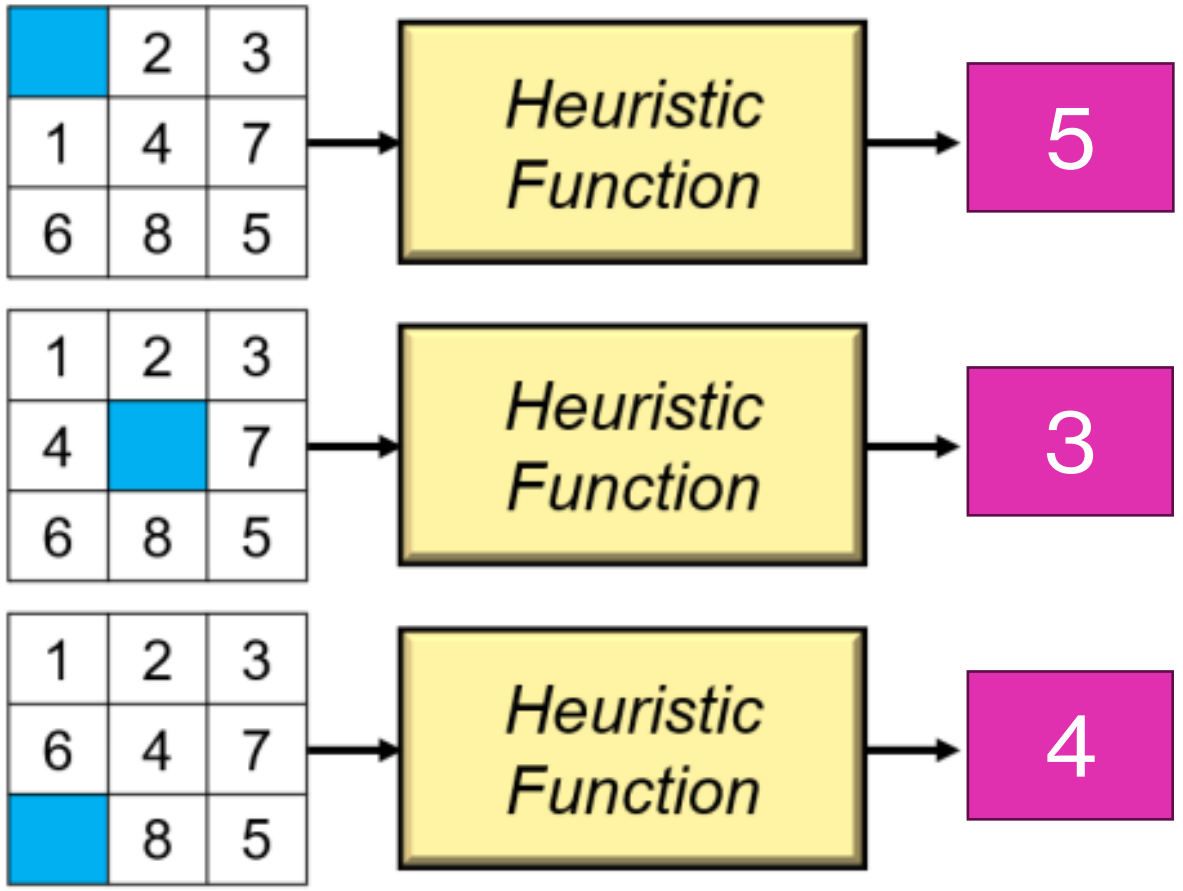
h(Goal) is 5 and is more than 0 → Not admissable

Not admissible! (from Goal)

■ Heuristic value

Admissible or Not Admissible?

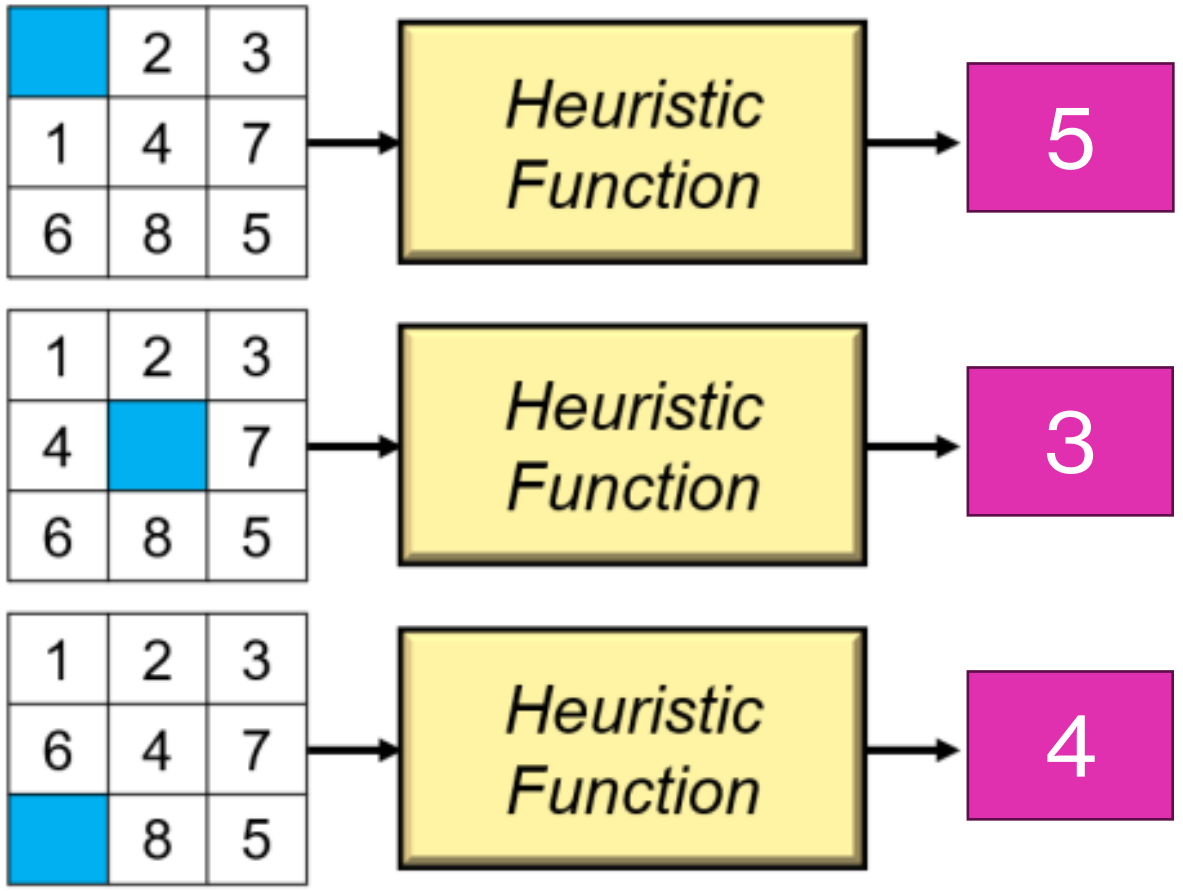
- $h(s)$ = number of tiles that are in the wrong place



Admissible or Not Admissible?

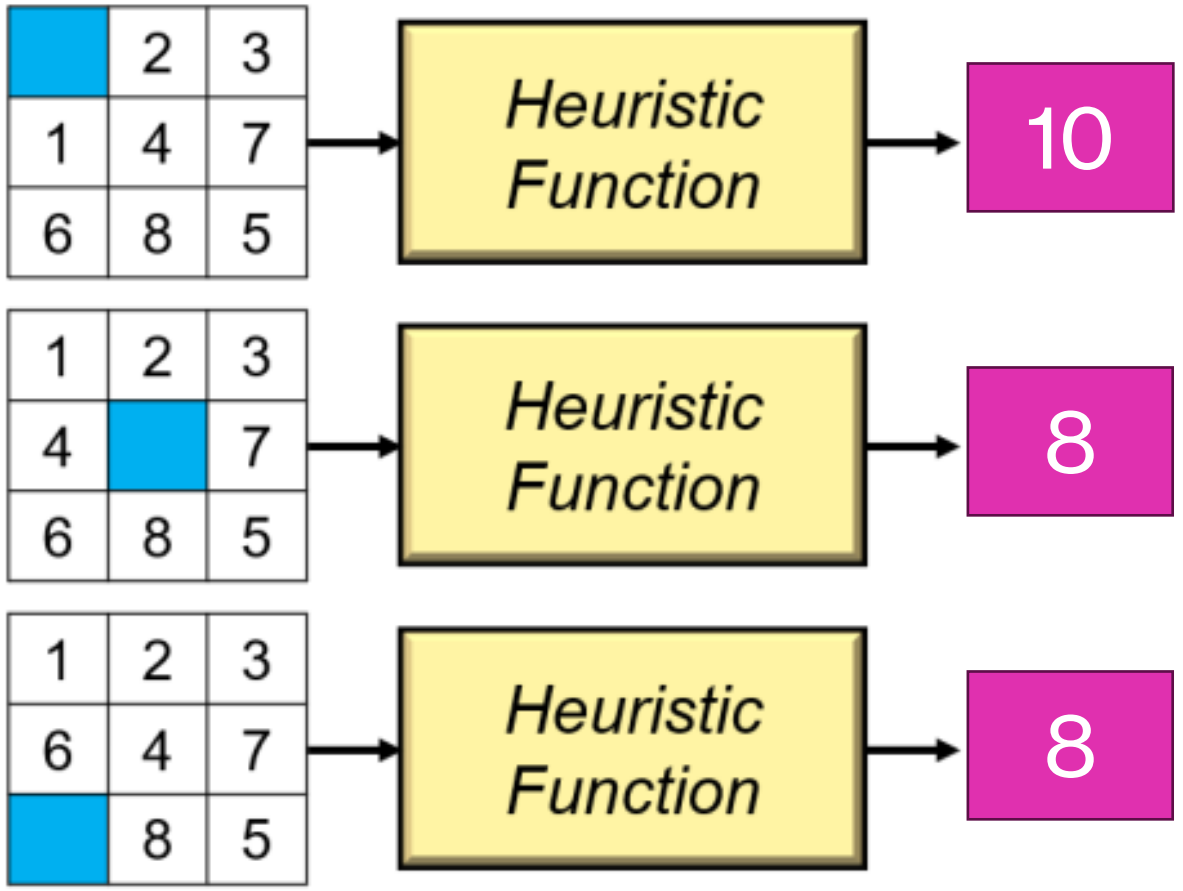
- **$h(s)$ = number of tiles that are in the wrong place**

- **Admissible!**
  - For each tile in the wrong place, we're sure that it needs at least one move to put it in the right place.



Admissible or Not Admissible?

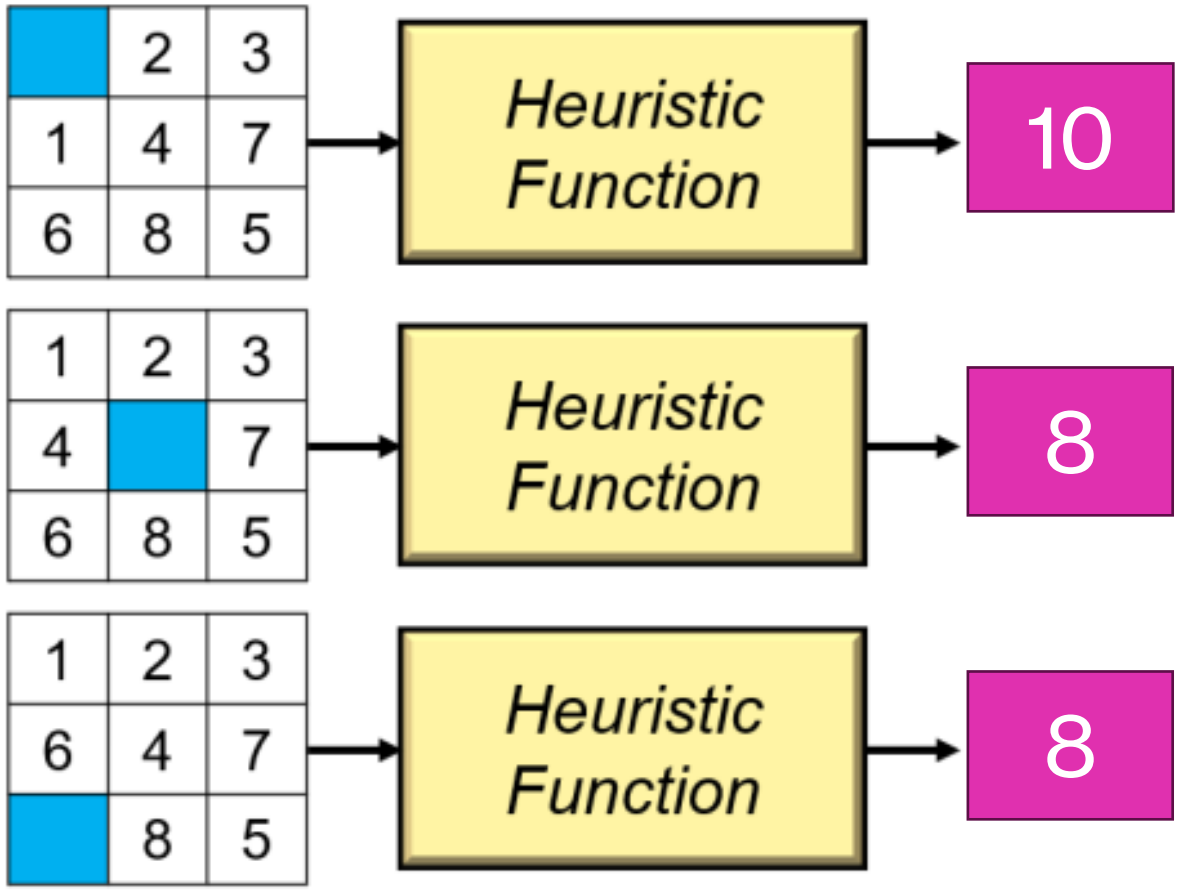
- $h(s)$ = total Manhattan Distance of each tile to its correct position



Admissible or Not Admissible?

- **$h(s)$ = total Manhattan Distance of each tile to its correct position**

- **Admissible!**
  - For each tile in the wrong place, we're sure that the number of moves to put it in the correct place will not be less than the Manhattan distance.



Why is Admissibility Important?

- A* is **only guaranteed to return an optimal solution if the heuristic is admissible**.

- To prove this, we will use the following notations:
  - $g(n)$: the cost of the optimal path from the start state to state n.
  - $h^*(n)$: the cost of the optimal path from state n to any goal.
  - $h(n)$: the heuristic value of state n.

Proof of A* Optimality

- Let $n_0, n_1, n_2, \ldots, n_k$ be the states in the optimal path, with n_k being the optimal goal.
- Let m be a non-optimal goal, such that $g(m) > g(n_k)$.
- Since m is a goal, $h^*(m) = 0$ and thus $h(m) = 0$.
- Every step of A* search selects the next n in such a way that:
  - $g(n_i) + h(n_i) \leq g(n_i) + h^*(n_i)$
  - $g(n_i) + h(n_i) \leq g(n_k)$ because $g(n_i) + h^*(n_i) \leq g(n_k)$
  - $g(n_i) + h(n_i) < g(m)$ because $g(n_k) < g(m)$
  - $g(n_i) + h(n_i) < g(m) + h(m)$ because $g(m) = g(m) + h(m)$
- But if $g(n_i) + h(n_i) < g(m) + h(m)$, then A* could not have selected m! Therefore, it is impossible to select m as a non-optimal goal.

Acknowledgments

- Stanford University CS221 Autumn 2021 course. Available online at: https://stanford-cs221.github.io/autumn2021
- Previous CSINTSY slides by the following instructors:
  - Raymund Sison, PhD
  - Judith Azcarraga, PhD
  - Merlin Suarez, PhD
  - Joanna Pauline Rivera