

# COMP3211 Tutorial 3: Search

---

Fengming ZHU

Feb. 26&29, 2024

Department of CSE  
HKUST

# Outline

---

Search

Before Formulations

Formulation

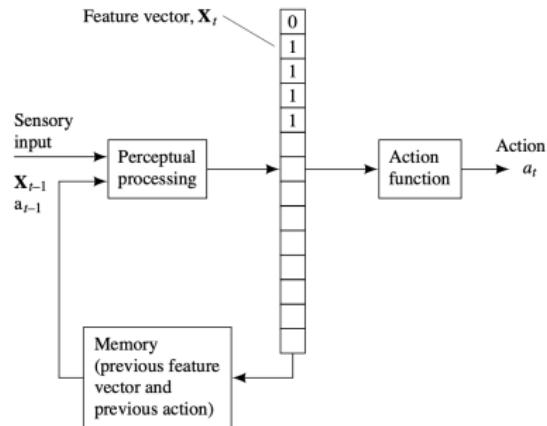
Search Diagram

Exercise

# Search

---

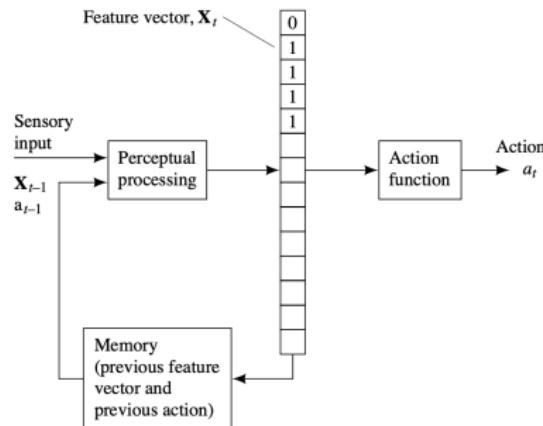
# Simple Agents



© 1998 Morgan Kaufman Publishers

**Figure 1:** Simple agents

# Simple Agents



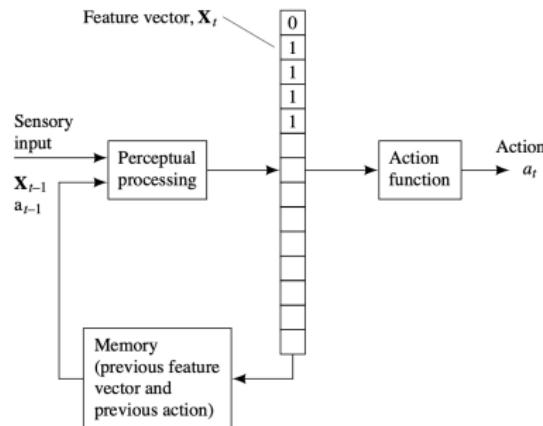
© 1998 Morgan Kaufman Publishers

## Key points:

- Respond to the environment,

**Figure 1:** Simple agents

# Simple Agents



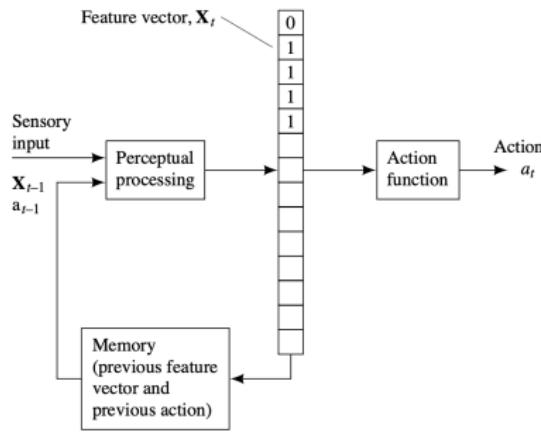
© 1998 Morgan Kaufman Publishers

## Key points:

- Respond to the environment,
- Consider how the world IS, or HAVE BEEN,

**Figure 1:** Simple agents

# Simple Agents



© 1998 Morgan Kaufman Publishers

**Figure 1:** Simple agents

## Key points:

- Respond to the environment,
- Consider how the world IS, or HAVE BEEN,
- Cannot imagine how the world WOULD BE.

# Formulation

## Notations:

- A set of states  $\mathcal{S}$
  - An initial state  $I \in \mathcal{S}$
  - A goal state  $G \in \mathcal{S}$  (sometimes a goal test)
  - A set of actions  $\mathcal{A}$
  - Deterministic transitions  $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$
  - Cost function  $c : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$
  - A solution (path) is a sequence of actions from  $I$  to  $G$ .
- function*  
*graph*

## Follow-up Questions

**Question #1:**

Can reactive agents or state machines do search?

## Follow-up Questions

### Question #1:

Can reactive agents or state machines do search?

- Neither, should be agents that can plan ahead (transitions matter).

## Follow-up Questions

### Question #1:

Can reactive agents or state machines do search?

- Neither, should be agents that can plan ahead (transitions matter).

### Question #2:

For agent 1, who can compute a feasible plan (path), if you extract her plan and deploy to agent 2 under the same setting, who has no sensing ability and no computing power, can she successfully reach the goal?

## Follow-up Questions

### Question #1:

Can reactive agents or state machines do search?

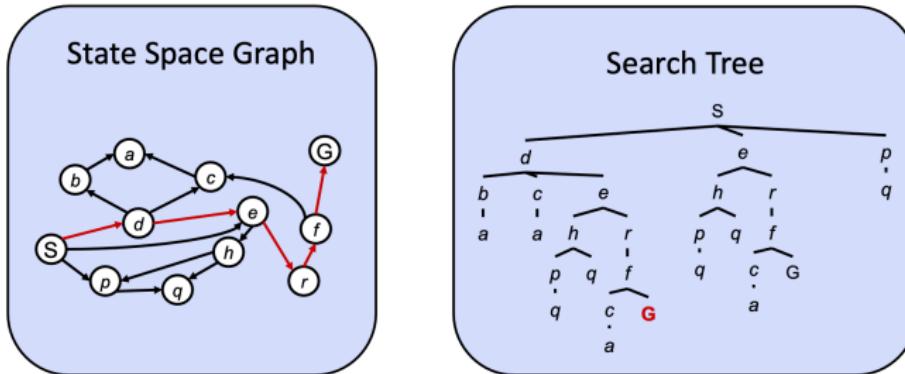
- Neither, should be agents that can plan ahead (transitions matter).

### Question #2:

For agent 1, who can compute a feasible plan (path), if you extract her plan and deploy to agent 2 under the same setting, who has no sensing ability and no computing power, can she successfully reach the goal?

- Yes, once computed, just blindly execute it.

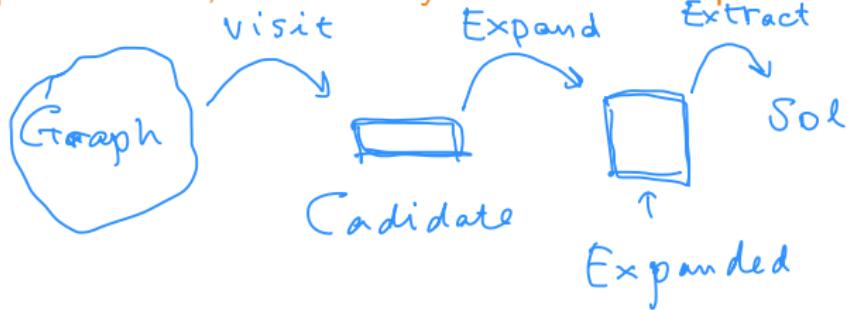
# General Idea: Graph/Tree Search



A node can only be expanded once, while it may be visited multiple times!

Key concepts:

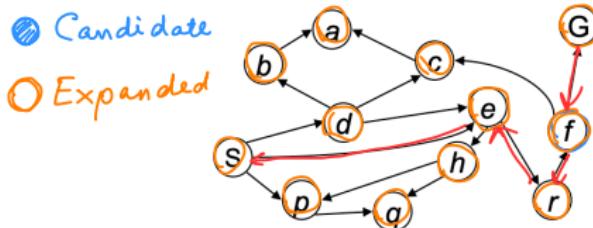
- Fringe (frontier)
- Expansion
- Exploration strategy



# Breadth-First Search

Strategy: expand a shallowest node first

Implementation: Fringe is a FIFO queue



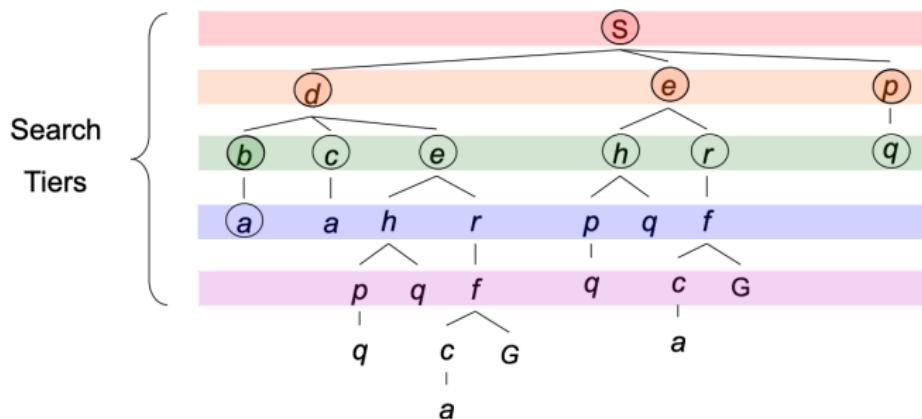
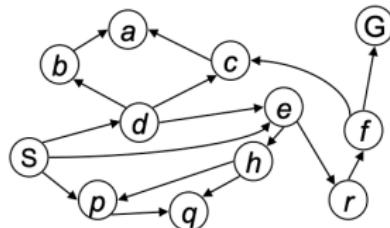
1 dep  
2 e p b c e d  
3 p b c e h r d e  
4 b c e h r g d e p  
5 c e h r g a d e p b  
6 e h r g a a d e p b c  
7 x h r g a a d e p b c  
8 r g a a g d e p b c h  
9 g a a g f d e p b c h r

10 a a g f d e p b c h r g  
" a g f d e p b c h r g a  
" x g f d e p b c h r g a  
12 x f d e p b c h r g a  
✓ ① G d e p b c h r g a f  
② d e p b c h r g a f G  
S → e → r → f → G

# Breadth-First Search

*Strategy: expand a shallowest node first*

*Implementation: Fringe is a FIFO queue*



**Figure 2:** Breadth-First Search

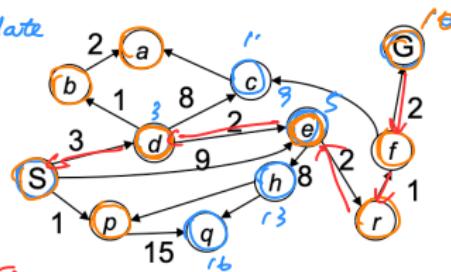
# Uniform Cost Search WEIGHTED - BFS

Strategy: expand a cheapest node first:

Fringe is a priority queue  
(priority: cumulative cost)

○ Candidate

○ Expand



1	p	d <sup>3</sup>	e <sup>9</sup>		①
2	d <sup>3</sup>	e <sup>9</sup>	g <sup>16</sup>	p	
3	b <sup>4</sup>	e <sup>5</sup>	e <sup>9</sup>	c <sup>"</sup> q <sup>16</sup>	p d
7	e <sup>5</sup>	a <sup>6</sup>	e <sup>9</sup>	c <sup>"</sup> g <sup>16</sup>	p d b
5	a <sup>6</sup>	r <sup>7</sup>	e <sup>9</sup>	c <sup>"</sup> h <sup>13</sup>	g <sup>16</sup> p d b e
6	r <sup>7</sup>	e <sup>9</sup>	c <sup>"</sup> h <sup>13</sup>	g <sup>16</sup>	p d b e a
7	f <sup>8</sup>	e <sup>9</sup>	c <sup>"</sup> h <sup>13</sup>	g <sup>16</sup>	p d b e a r

8 e<sup>9</sup> G<sup>10</sup> c<sup>"</sup> h<sup>13</sup> g<sup>16</sup> p d b e a r f  
9 x G<sup>10</sup> c<sup>"</sup> h<sup>13</sup> g<sup>16</sup> p d b e a r f  
c<sup>"</sup> h<sup>13</sup> g<sup>16</sup> p d b e a r f G

S → d → e → r → f → G

# Uniform Cost Search

Strategy: expand a cheapest node first:

Fringe is a priority queue  
(priority: cumulative cost)

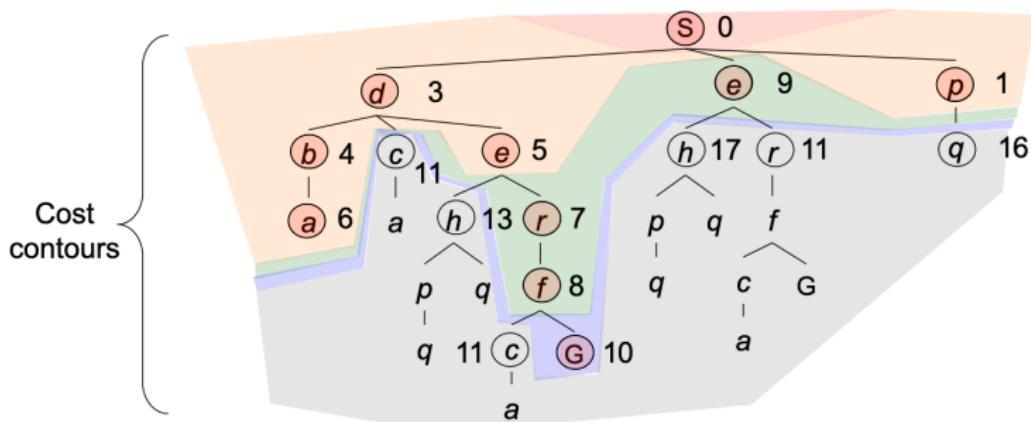
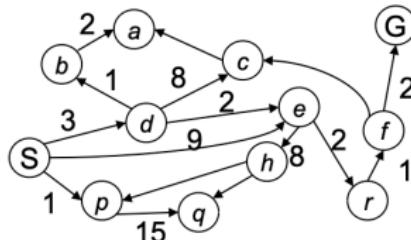
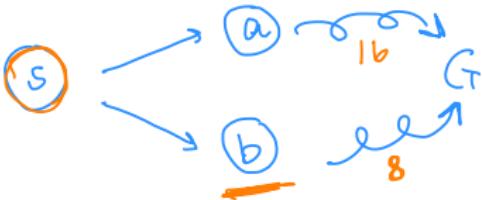


Figure 3: Uniform Cost Search

# Greedy Search



- Strategy: expand a node that you think is closest to a goal state.
- Best case: every time you make a perfect guess.
- Worst case: turn around until you get into a dead end.

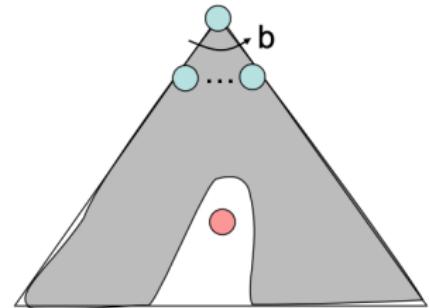
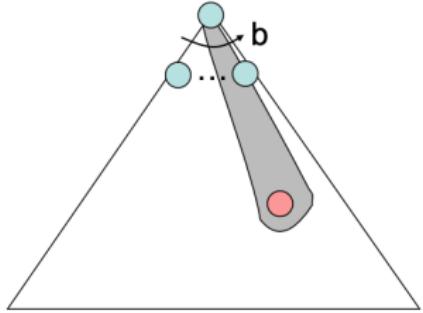
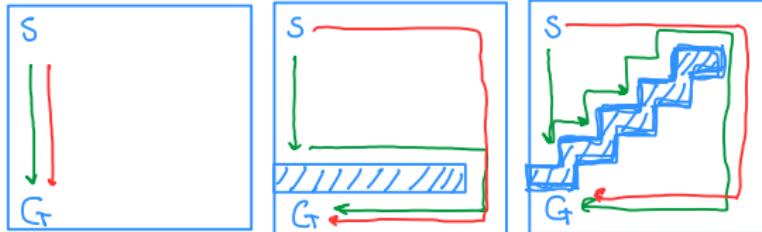


Figure 4: ~~Uniform Cost Search~~  
Greedy search

## $A^*$ Search

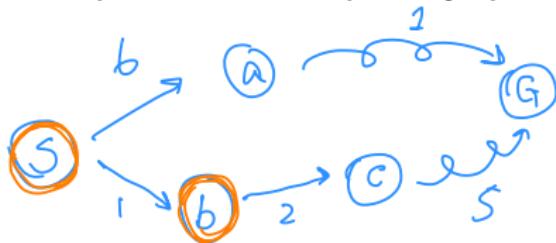
- Uniform-cost orders by path cost, or backward cost  $g(n)$
- Greedy orders by goal proximity, or forward cost  $h(n)$

## $A^*$ Search

- Uniform-cost orders by path cost, or backward cost  $g(n)$
- Greedy orders by goal proximity, or forward cost  $h(n)$
- Strategy: expand a node that is best so far.

# A\* Search

- Uniform-cost orders by path cost, or backward cost  $g(n)$
- Greedy orders by goal proximity, or forward cost  $h(n)$
- Strategy: expand a node that is best so far.
- Implementation: priority queue,  $f(n) = g(n) + h(n)$ .



BFS:  $c^{l+2} < a^b$

Greedy:  $a^1 < c^5$

$A^*$ :  $a^{b+1} < c^{l+2+5}$

## $A^*$ Search

- Uniform-cost orders by path cost, or backward cost  $g(n)$
- Greedy orders by goal proximity, or forward cost  $h(n)$
- Strategy: expand a node that is best so far.
- Implementation: priority queue,  $f(n) = g(n) + h(n)$ .
- Admissible heuristic:  $h(n) \leq cost(n, G)$ .

# A\* Search

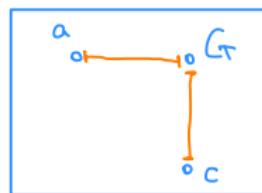
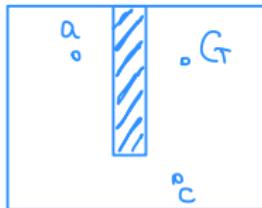
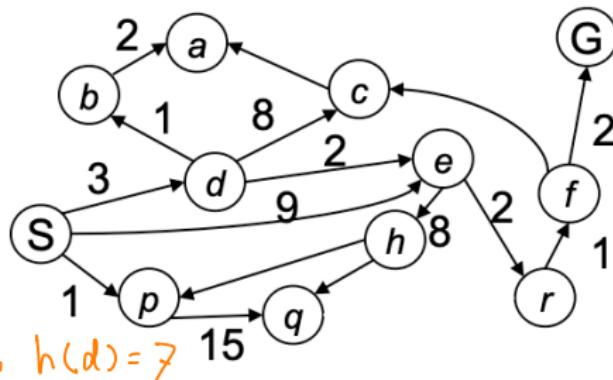
- Uniform-cost orders by path cost, or backward cost  $g(n)$
  - Greedy orders by goal proximity, or forward cost  $h(n)$
  - Strategy: expand a node that is best so far.
  - Implementation: priority queue,  $f(n) = g(n) + h(n)$ .
- Admissible heuristic:  $h(n) \leq \text{cost}(n, G)$ .

Eg 1. For d.

$$\text{Cost}^*(d, G) = ?$$

$$h(d) \triangleq 2 < ?$$

$$f(d) = g(d) + h(d) \\ = 5$$

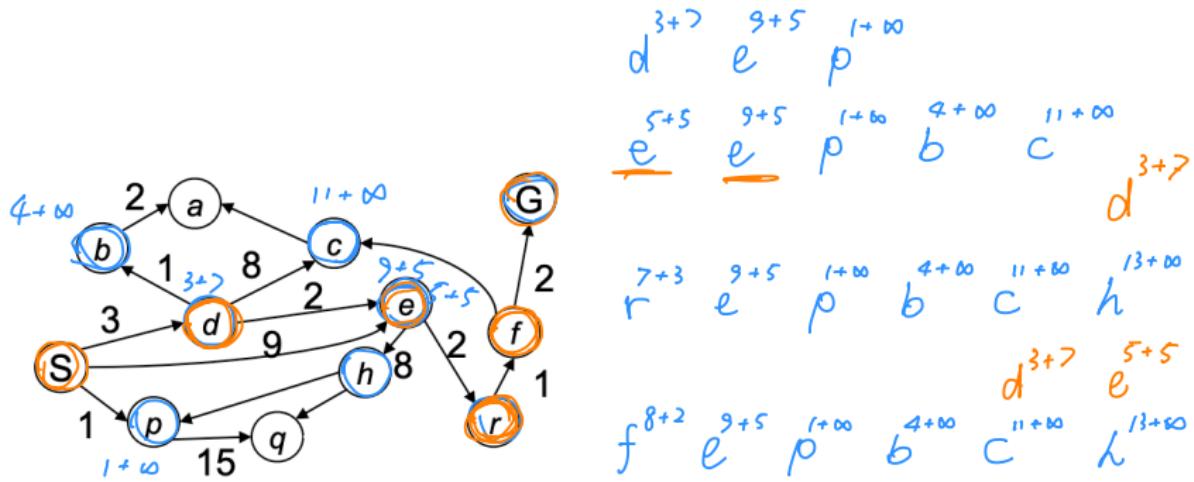


Eg 2.  $h(\cdot) = \text{cost}^*$ ,  $h(d) = ?$

Eg 3.  $h(\cdot) = 0$

Figure 5:  $h(n) = \text{shortest\_path\_length}(n, G)$

# A\* Search

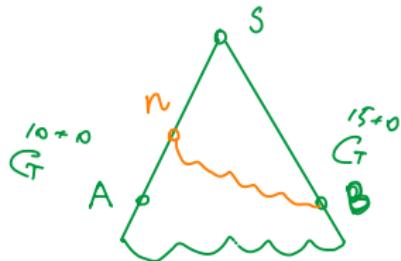


**Figure 6:**  
 $h(n) = \text{shortest\_path\_length}(n, G)$

$h:$   
 $\begin{matrix} G & f & r & e & d & s \\ \infty & \infty & \infty & \infty & \infty & \infty \end{matrix}$

$\xrightarrow{①} \begin{matrix} G & e & p & b & c & h \\ 10+\infty & 9+5 & 1+\infty & 4+\infty & 11+\infty & 13+\infty \end{matrix}$   
 $\xrightarrow{②} \begin{matrix} d & e & r & f \\ 3+7 & 5+5 & 7+3 & 8+2 \end{matrix}$   
 $\begin{matrix} G & d & e & r & f \\ 10+\infty & 3+7 & 5+5 & 7+3 & 8+2 \end{matrix}$

## Optimality of A\*:



A: Opt Goal state

B: sub-opt Goal state

⇒ Prove: A will always be expanded before B ?

$\exists n . n \in \text{Ancestors}(A), n \& B \in \text{Fringe}(\text{Open})$ .

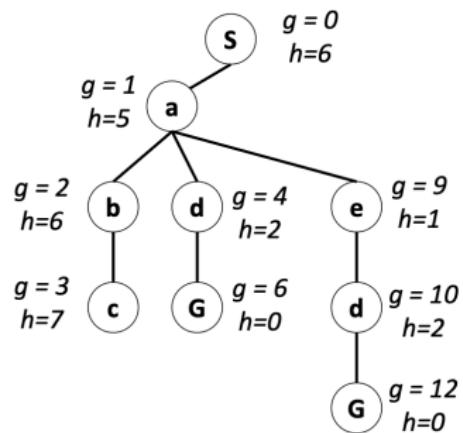
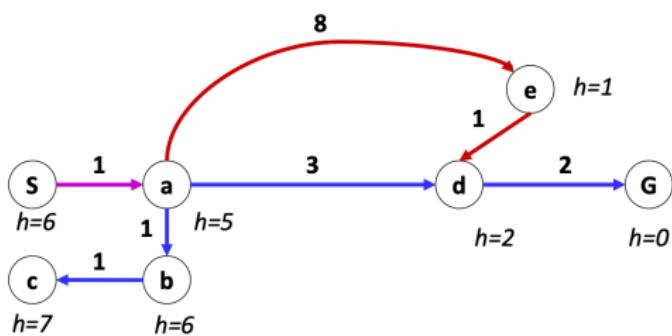
$$f(n) = g(n) + h(n) \leq g(n) + \text{Cost}^*(n, A)$$

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

$$f(A) = g(A) + 0 \leq g(B) + 0 = f(B)$$

⇒  $f(n) < f(B)$  Expand n ahead of B

# $A^*$ Search – another example



## $A^*$ Search – demo

- Uniform-cost orders by path cost, or backward cost  $g(n)$
- Greedy orders by goal proximity, or forward cost  $h(n)$
- Strategy: expand a node that is best so far.
- Implementation: priority queue,  $f(n) = g(n) + h(n)$ .
- Admissible heuristic:  $h(n) \leq cost(n, G)$ .
- Live demo: <https://www.movingai.com/SAS/index.html>

# Exercise

## *A\** search:

- An offline process – when the map changes: need to replan the whole solution.

## Exercise

### *A\** search:

- An offline process – when the map changes: need to replan the whole solution.
- Can we reuse any historical data to be more efficient?

## Exercise

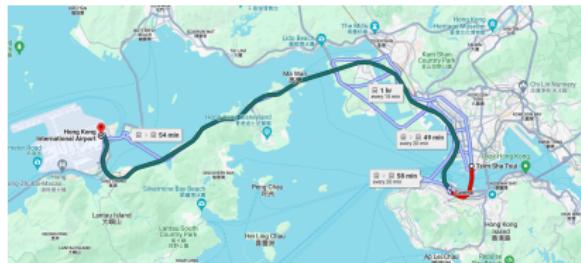
### *A\* search:*

- An offline process – when the map changes: need to replan the whole solution.
- Can we reuse any historical data to be more efficient?

# Exercise

## *A\** search:

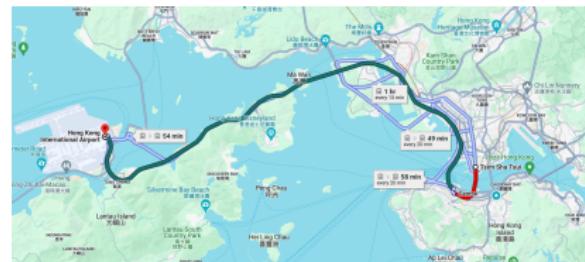
- An offline process – when the map changes: need to replan the whole solution.
- Can we reuse any historical data to be more efficient?



# Exercise

## A\* search:

- An offline process – when the map changes: need to replan the whole solution.
- Can we reuse any historical data to be more efficient?



# Exercise

## *A\** search:

- An offline process – when the map changes: need to replan the whole solution.
- Can we reuse any historical data to be more efficient?



**Figure 7:** Plan backwards from the goal to the start: *D\** Lite

*Thanks!*