NUS | Computing
National University of Singapore

CS3243 Tutorial 2
Informed Search

Gu Zhenhao

February 1, 2023

Notes

1. Start working on project 1.
2. Tutorial assignment should be in-paper.
3. Check that you all get your TA1 feedback from me.

Remainders ...

... from tutorial 1

Tutorial 1, Problem 1.a

2	5			3		9		1
	1				4			
4		7				2		8
		5	2					
				9	8	1		
	4				3			
			3	6			7	2
	7							3
9		3				6		4

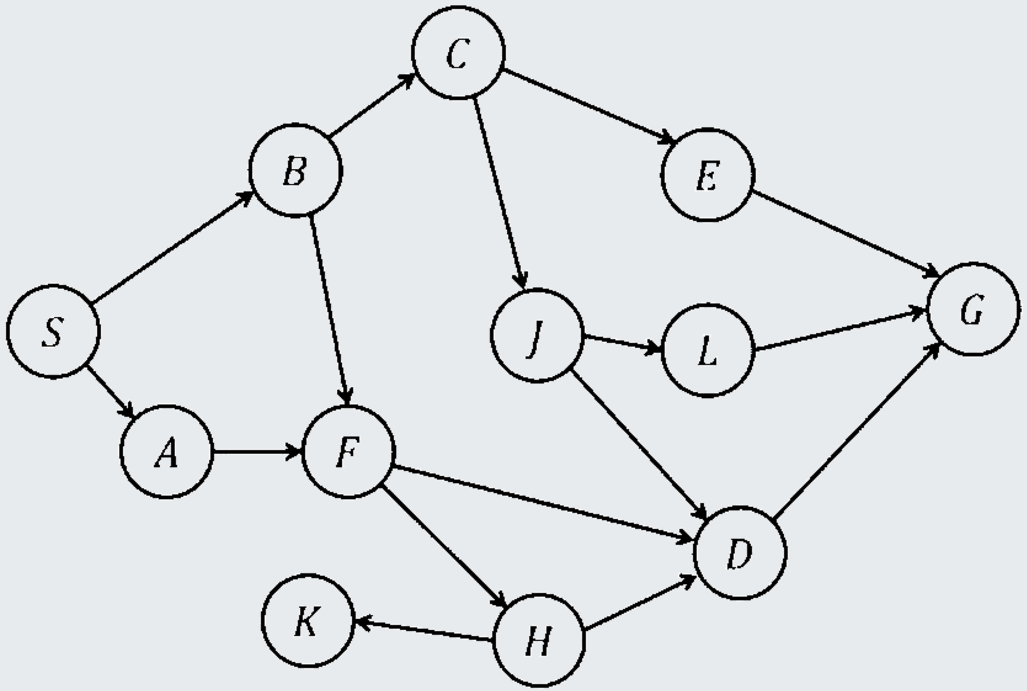
Figure. A Sudoku Puzzle.

Question: why can we view this environment as episodic?

Short answer: can be solved with simple/model-based **reflex** agent.

Action chosen is solely dependent on current state, not the previous actions!

Tutorial 1, Problem 5

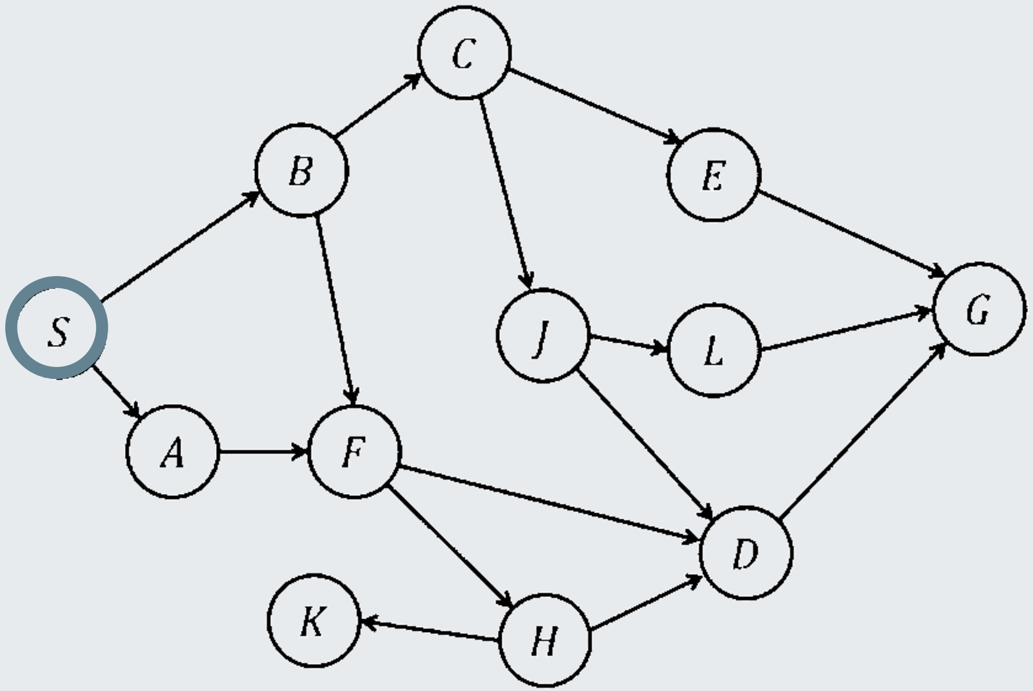


- **Goal**: Find path from S to G using DFS.

Visited:

Frontier: [*S(/)*]

Tutorial 1, Problem 5

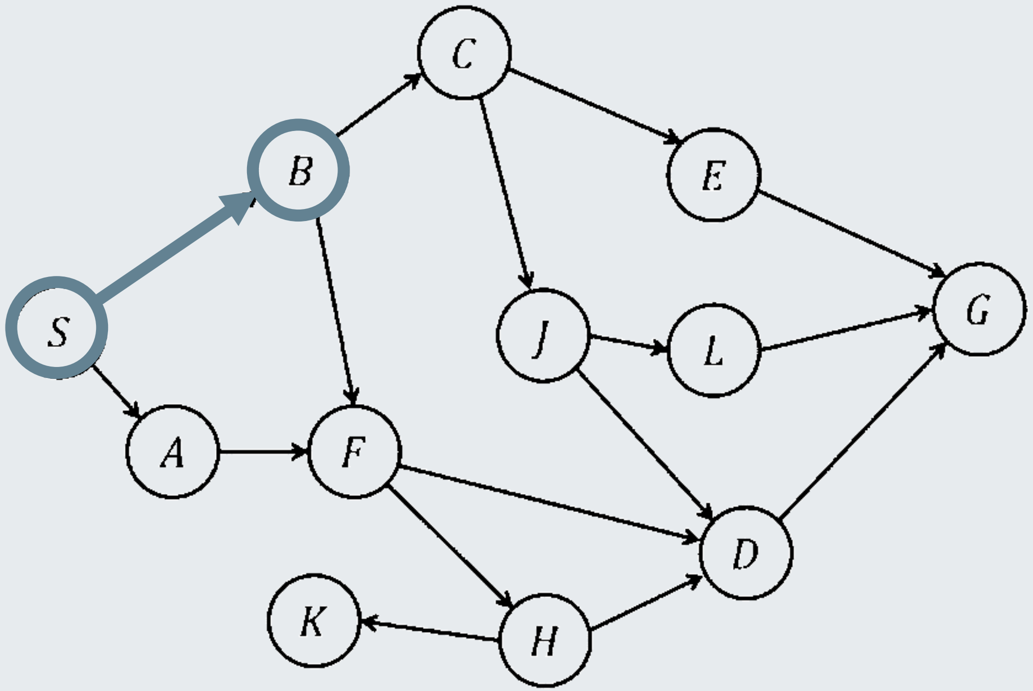


- **Goal**: Find path from S to G using DFS.

Visited: S

Frontier: [$A(S)$, $B(S)$]

Tutorial 1, Problem 5



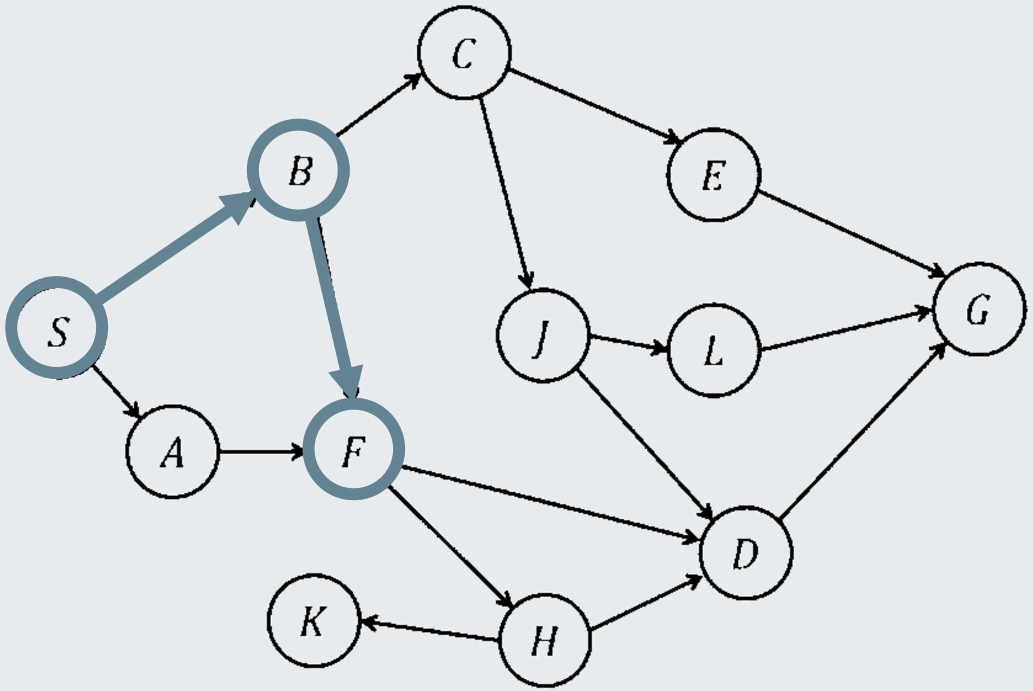
- **Goal**: Find path from S to G using DFS.

Visited: *S*, *B(S)*

Frontier: [*A(S)*, *C(B)*, *F(B)*]

Pop from the top of the stack.

Tutorial 1, Problem 5

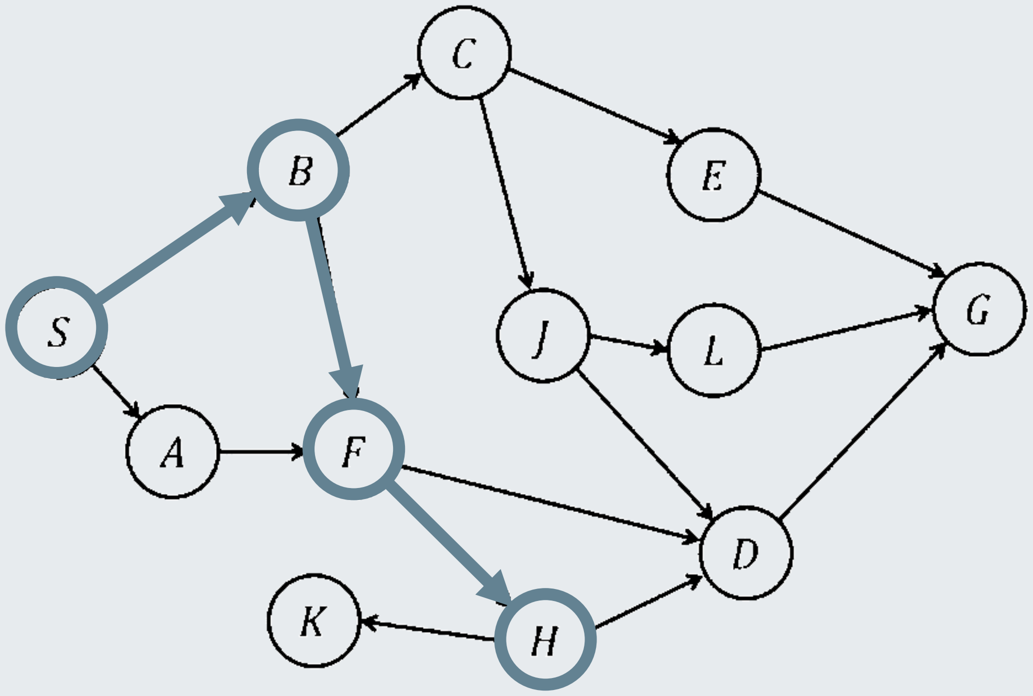


- **Goal**: Find path from S to G using DFS.

Visited: S, $B(S)$, $F(B)$

Frontier: [$A(S)$, $C(B)$, $D(F)$, $H(F)$]

Tutorial 1, Problem 5



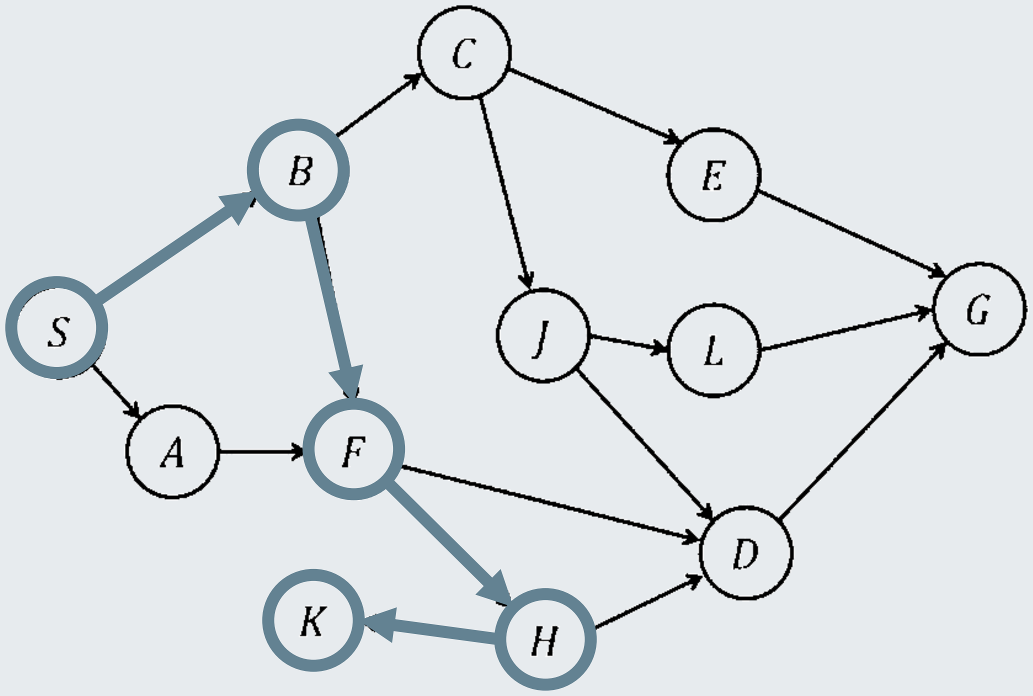
- **Goal**: Find path from S to G using DFS.

Visited: *S*, *B(S)*, *F(B)*, *H(F)*

Frontier: [*A(S)*, *C(B)*, *D(F)*, *D(H)*, *K(H)*]

pushing D again due to tree search.

Tutorial 1, Problem 5

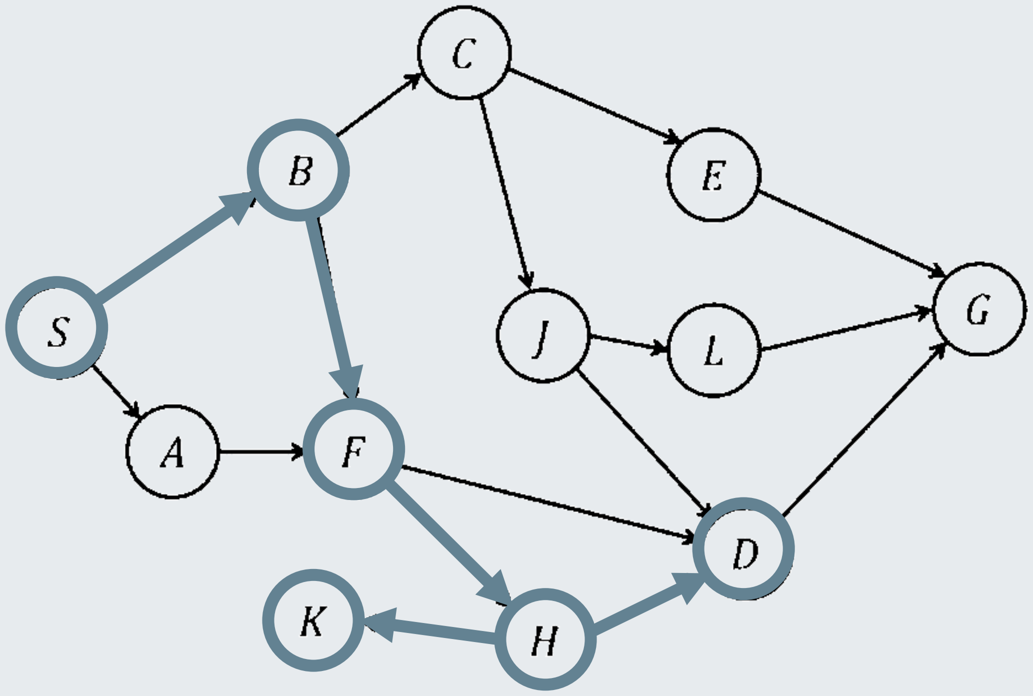


- **Goal**: Find path from S to G using DFS.

Visited: *S*, *B*(*S*), *F*(*B*), *H*(*F*), *K*(*H*)

Frontier: [*A*(*S*), *C*(*B*), *D*(*F*), *D*(*H*)]

Tutorial 1, Problem 5

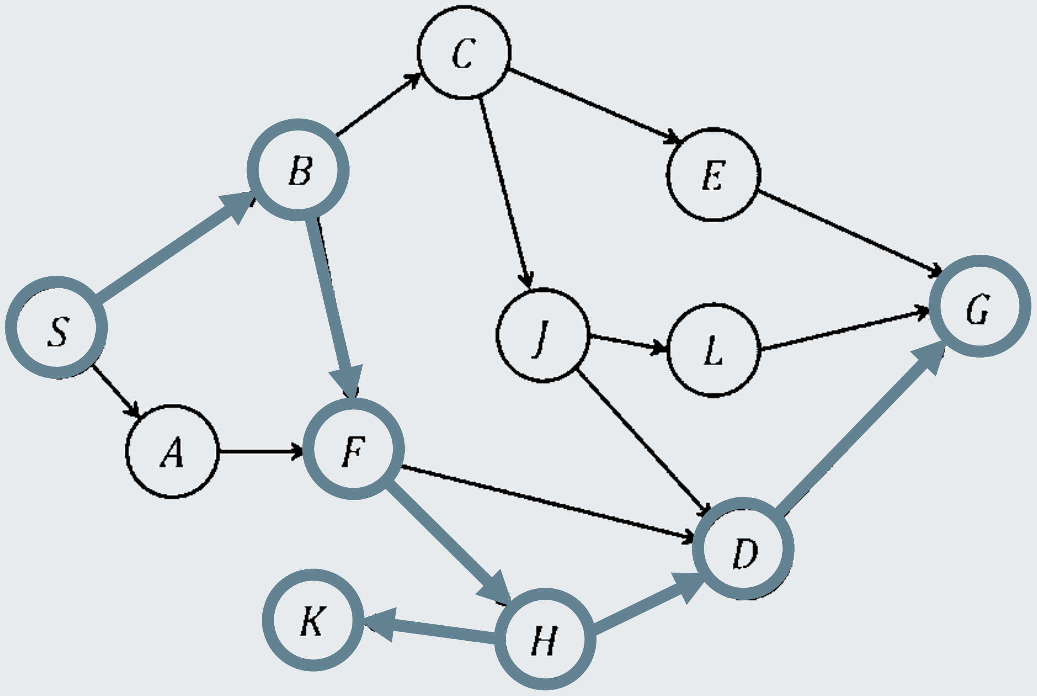


- **Goal**: Find path from S to G using DFS.

Visited: *S*, *B*(*S*), *F*(*B*), *H*(*F*), *K*(*H*), *D*(*H*)

Frontier: [*A*(*S*), *C*(*B*), *D*(*F*), *G*(*D*)]

Tutorial 1, Problem 5



- **Goal**: Find path from S to G using DFS.

Visited: *S*, *B(S)*, *F(B)*, *H(F)*, *K(H)*, *D(H)*, *G(D)*

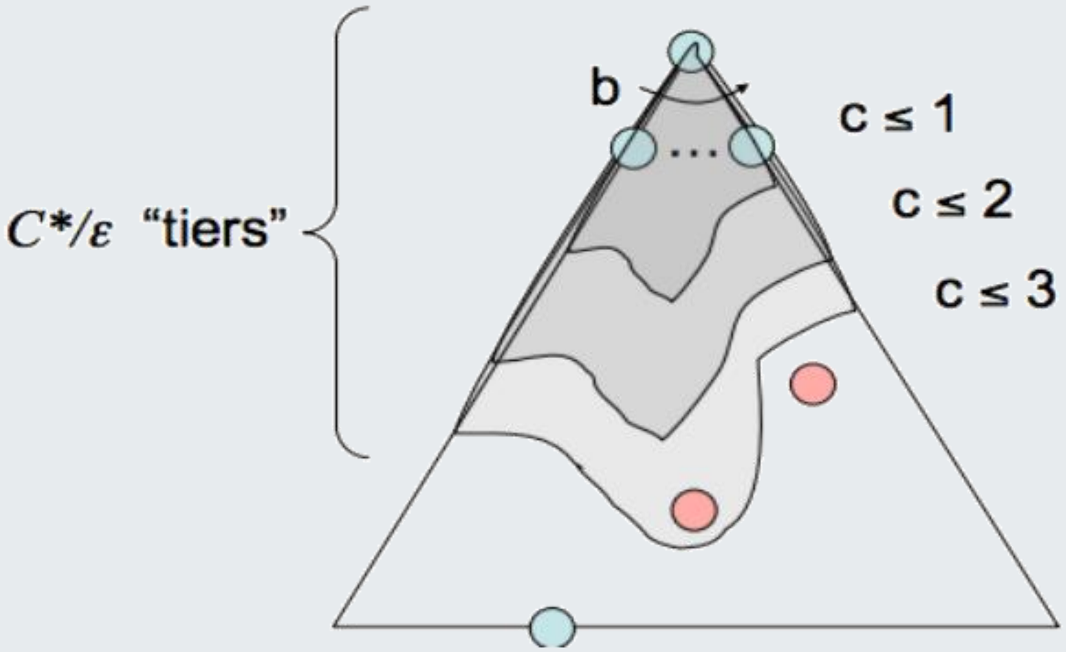
Frontier: [*A(S)*, *C(B)*, *D(F)*]

Final Path: *S-B-F-H-D-G*.

Informed Search

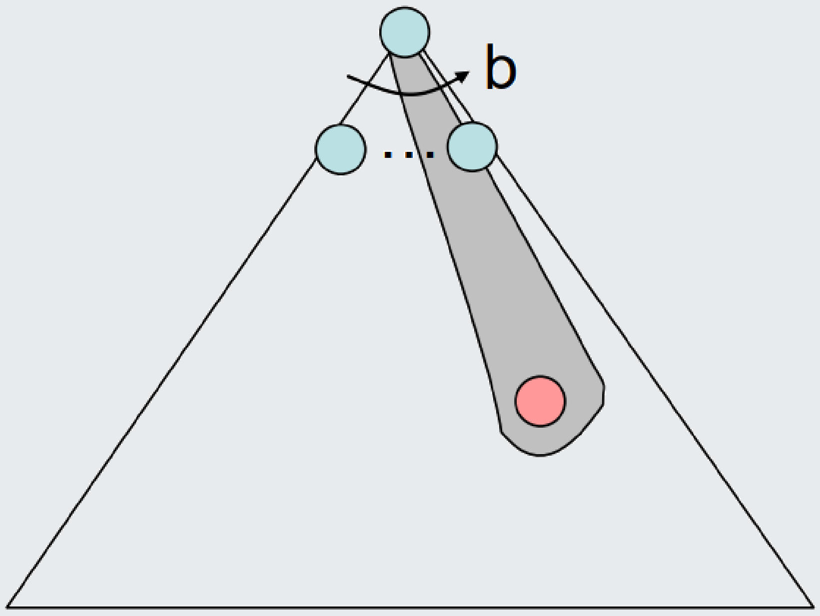
How can we utilize the information about the goals?

Uniform Cost Search



- **Intuition**: finds goal with smallest total action cost.
- **Pros**:
  - complete, and optimal for non-negative costs.
- **Cons**:
  - might also be memory-consuming.
  - wastes time searching in paths that don't lead to goal.

Greedy Best-First Search



- **Intuition**: go to nodes that ***looks*** closest to the goal, based on the heuristic $h(n)$.

Estimation of distance to goal.

- **Pros**:
  - If we are lucky, and the heuristic is good, we can find a solution quickly.
- **Cons**:
  - doesn't take the cost to get to the node into account.

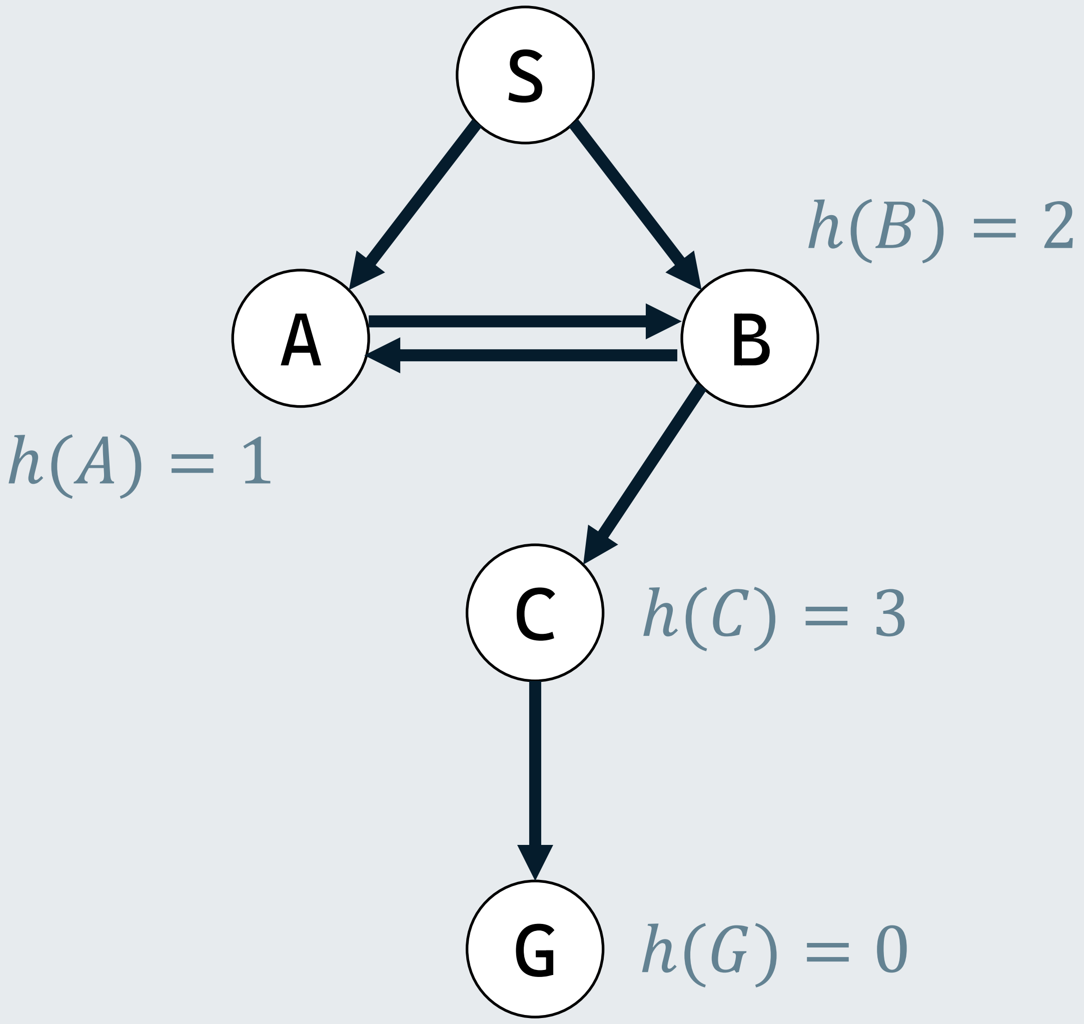
Question: is Greedy best-first search complete (given finite state space)?

Problem 1.a

Question: In what case, we will never explore node **G** in tree search version?

Idea:

- in the case when we are stuck in the loop **A–B–A–B–A**... and never pop **C** or **G** from the frontier.
- i.e. when **A** and **B** have smaller heuristic than **C** or **G**.

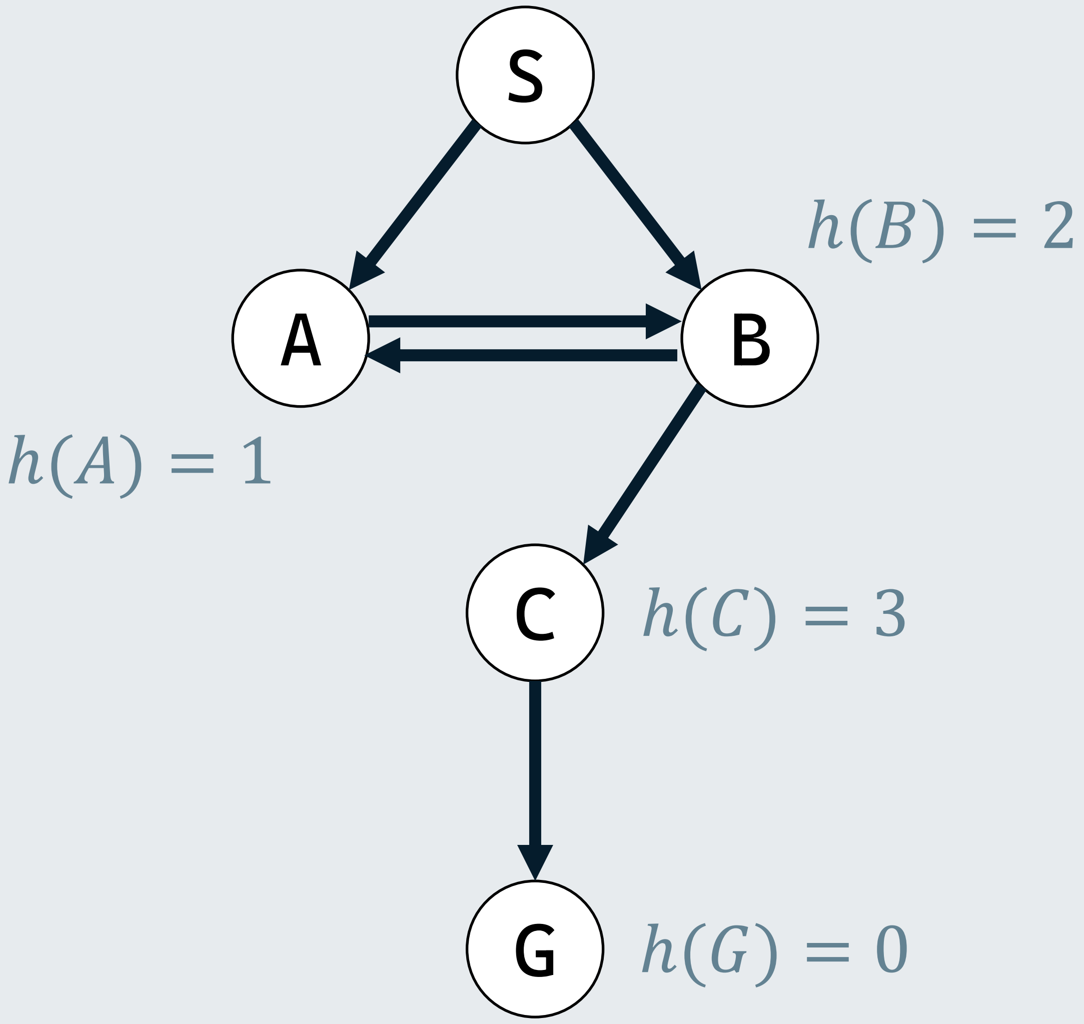


Problem 1.b

Question: In this case, is graph search version complete?

Yes!

- We will never visit a state twice.
- As long as the state space is finite, we will eventually visit all the states.

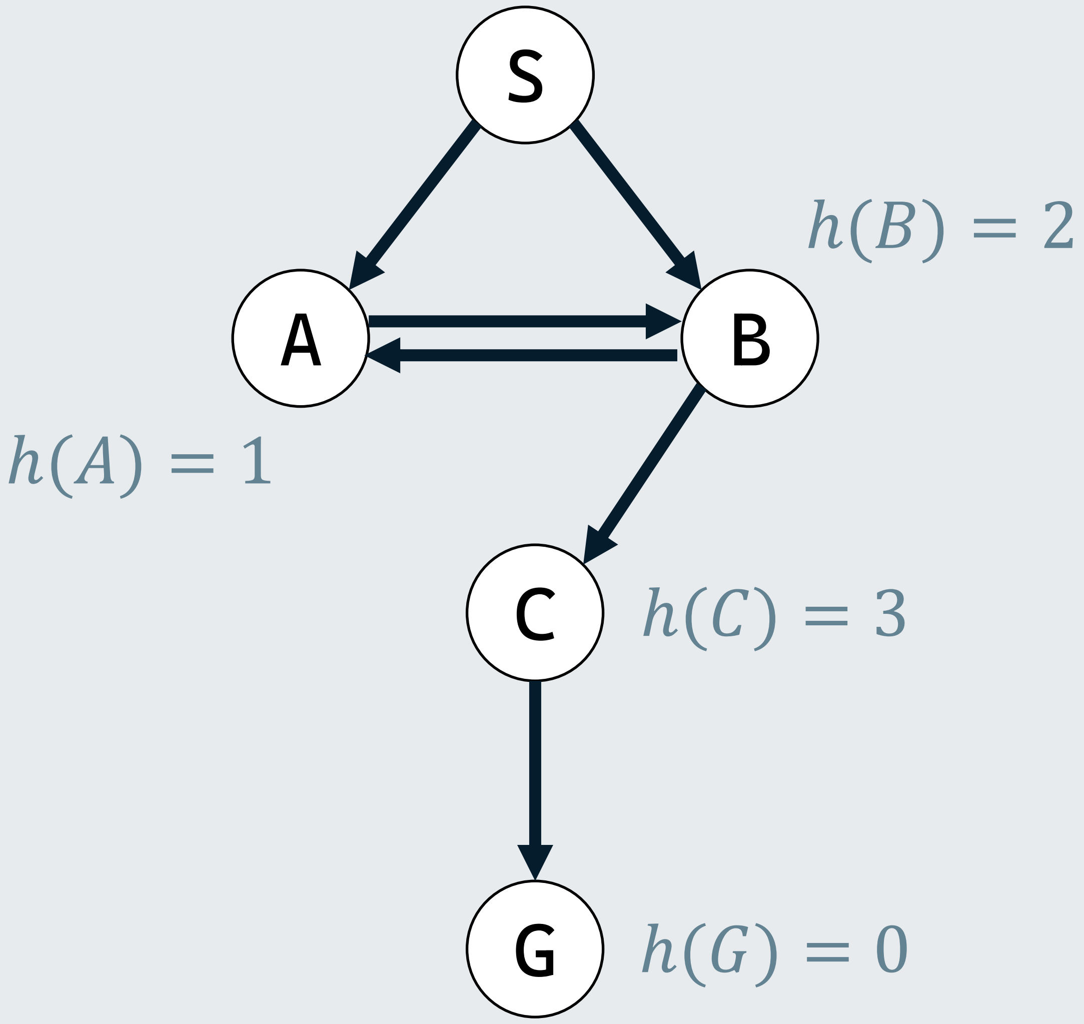


Problem 1.c

Question: When is greedy best-first search not optimal?

Two cases:

- When the heuristic h is badly defined, which leads us to sub-optimal goal or even incomplete.

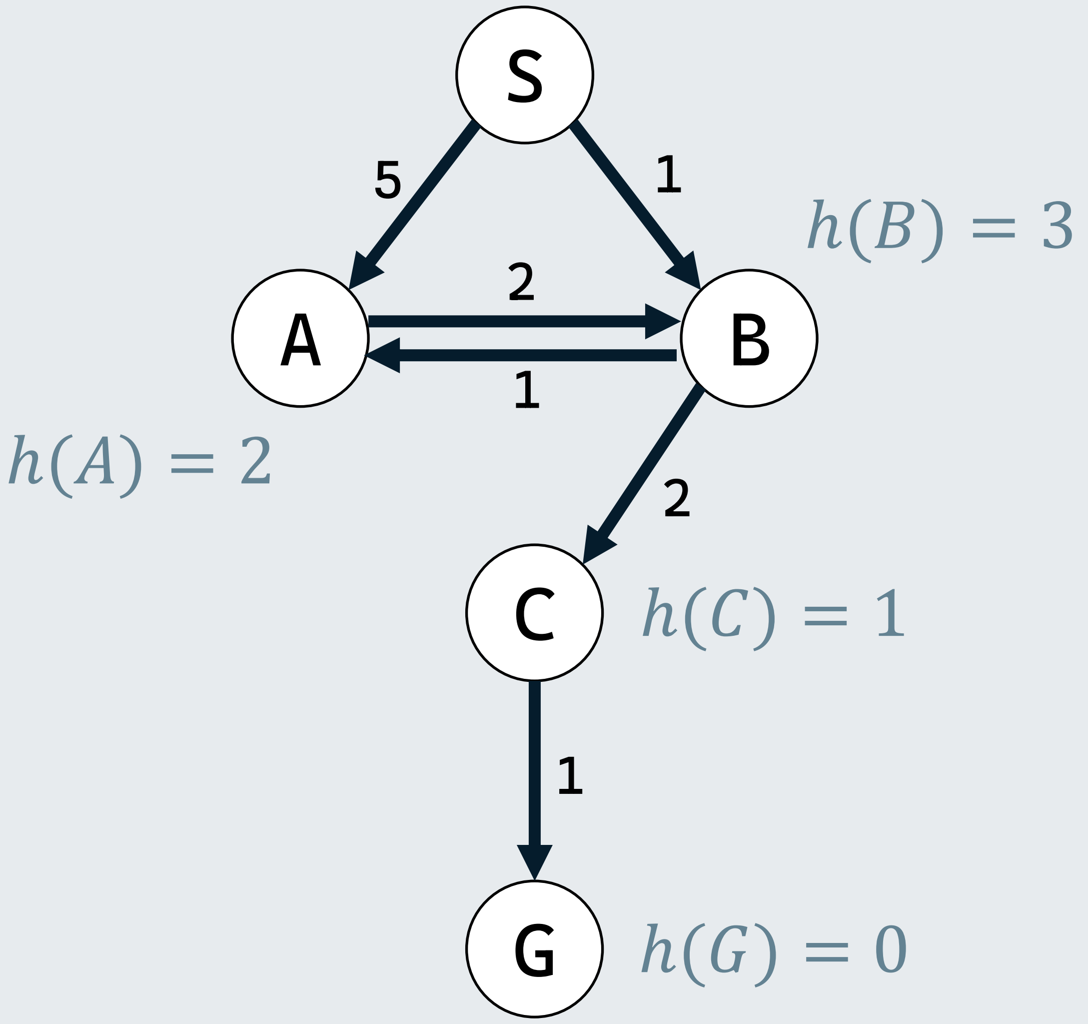


Problem 1.c

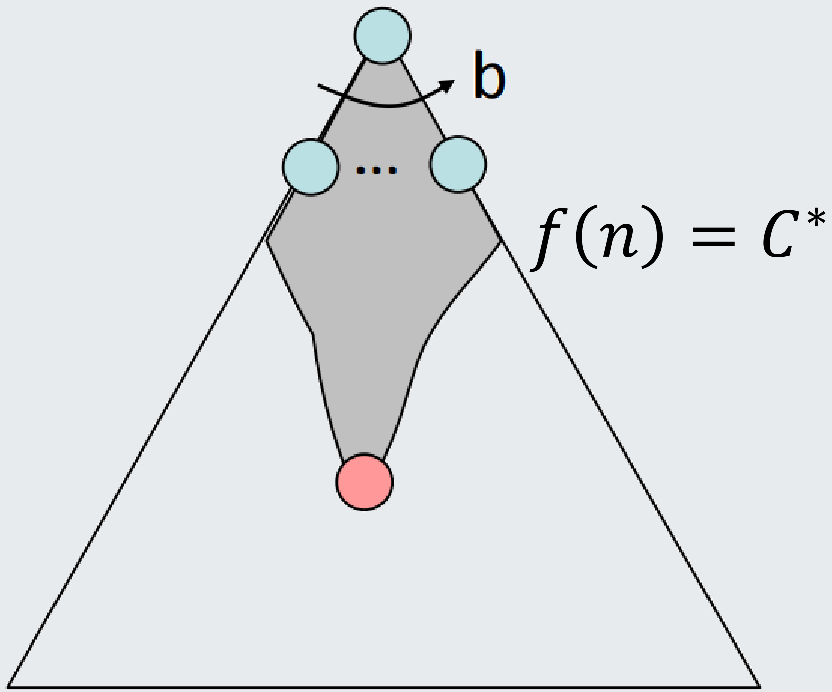
Question: When is greedy best-first search not optimal?

Two cases:

- When the heuristic h is badly defined, which leads us to sub-optimal goal or even incomplete.
- When the path cost to the "best" node is large.



A* Search



- **Intuition**: go to nodes that ***looks*** closest to both the start and the goal, based on $f(n) = g(n) + h(n)$.

  Estimation of total path cost.

- **Pros**:
  - Optimal while minimizing search space.
- **Challenges**:
  - Hard to find a "good" heuristic.

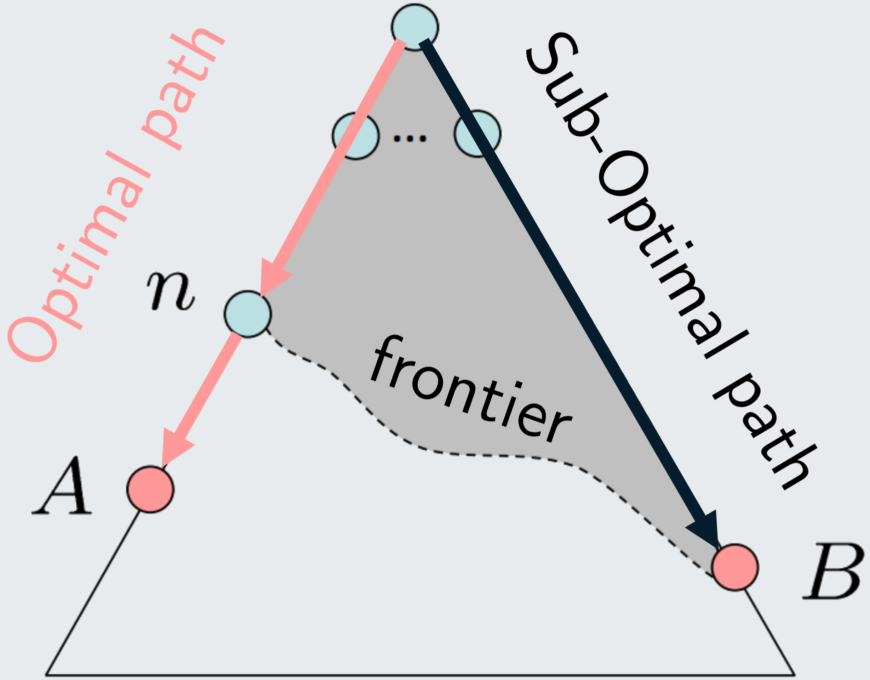
Question: What makes a heuristic function "good"?

Good Heuristics

True optimal cost to get to a goal.

- **admissible** (optimistic), if $h(n) \leq h^*(n)$ for all n,
- **consistent**, if $h(n) \leq c(n, a, n') + h(n')$ for all n and its successor n'.

Problem 2.a



Goal: show that if h is admissible, then A* tree search is optimal.

Claim 1: if n is ancestor of goal A, then $f(n) \leq f(A)$.

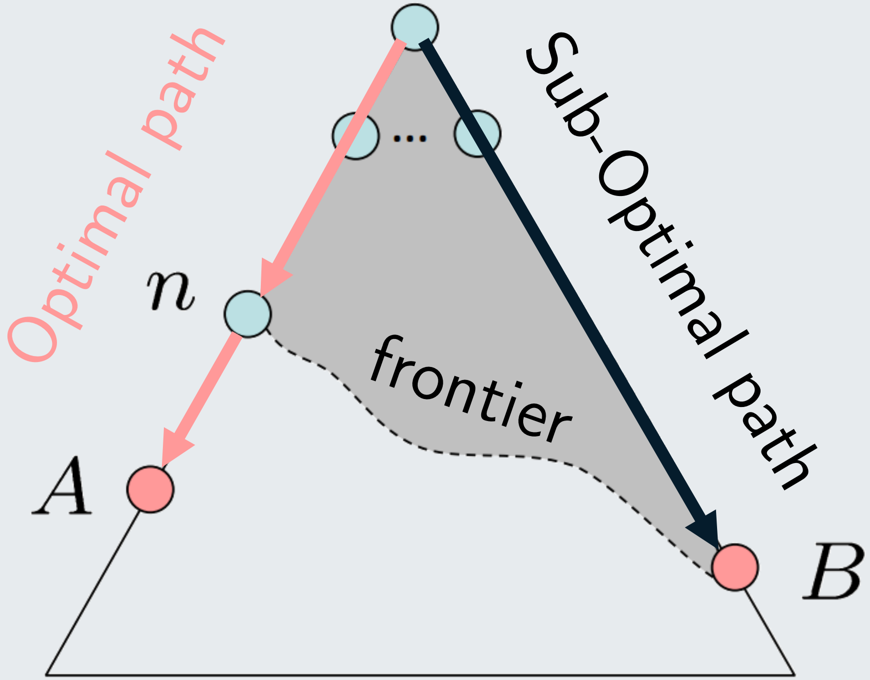
Proof.

$$
\begin{aligned}
f(n) &= g(n) + h(n) \\
&\leq g(n) + h^*(n) \\
&\leq g(A) \\
&= f(A)
\end{aligned}
$$

Admissibility

□

Problem 2.a



Goal: show that if h is admissible, then A* tree search is optimal.

Claim 1: if n is ancestor of goal A, then $f(n) \leq f(A)$.

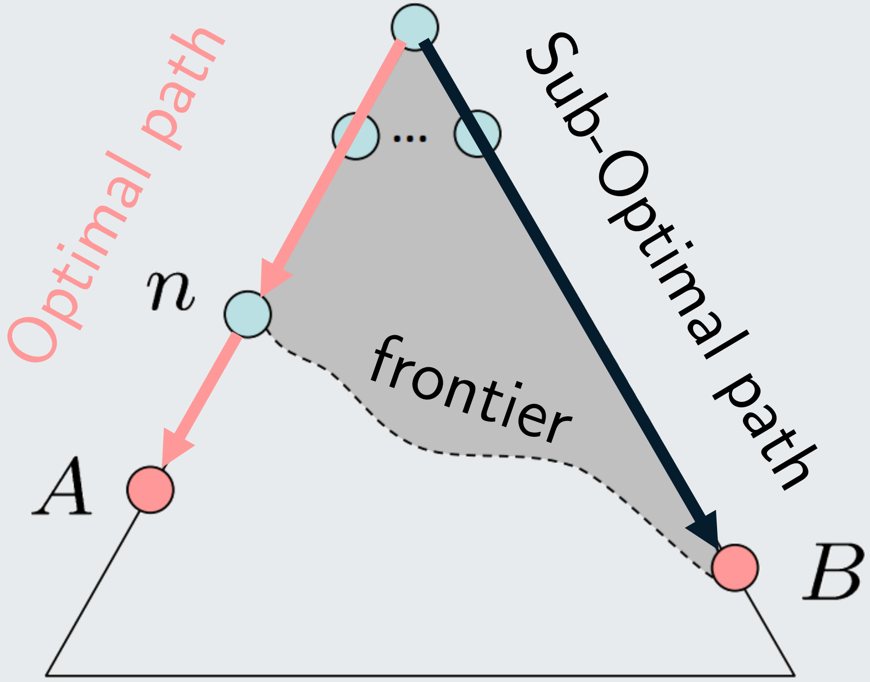
Claim 2: $f(A) \leq f(B)$.

Proof. Since A is the optimal goal,

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

□

Problem 2.a



Goal: show that if h is admissible, then A* tree search is optimal.

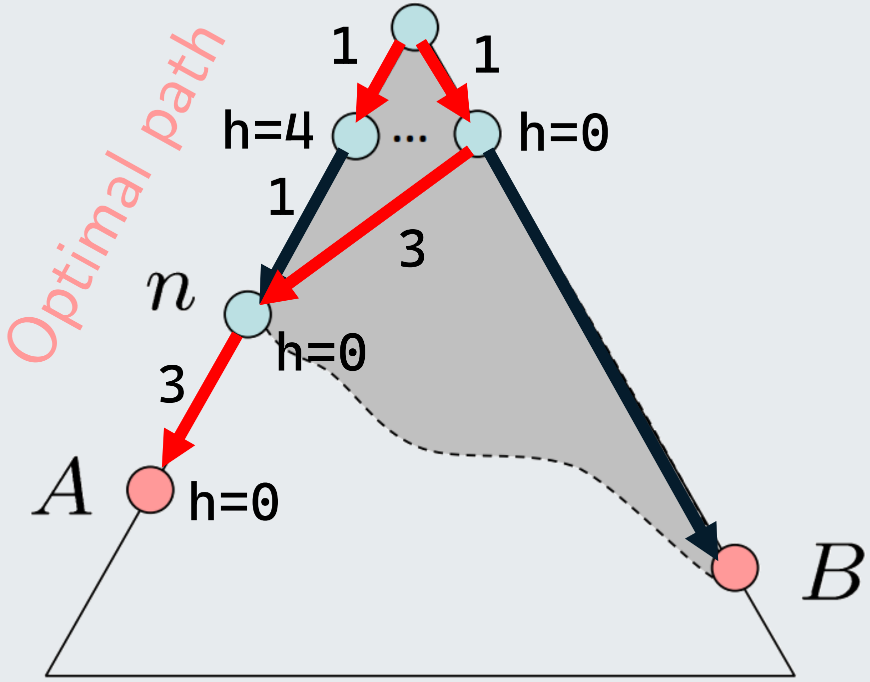
Claim 1: if n is ancestor of goal A, then $f(n) \leq f(A)$.

Claim 2: $f(A) \leq f(B)$.

Conclusion: $f(n) \leq f(B)$, which is true for all ancestors n of A (including A itself).

- All ancestors of A explored before B.
- The optimal path is explored before all other sub-optimal paths.

Problem 2.a



Question: Can we use the previous argument to prove optimality for *graph search (version 1 or 3)*?

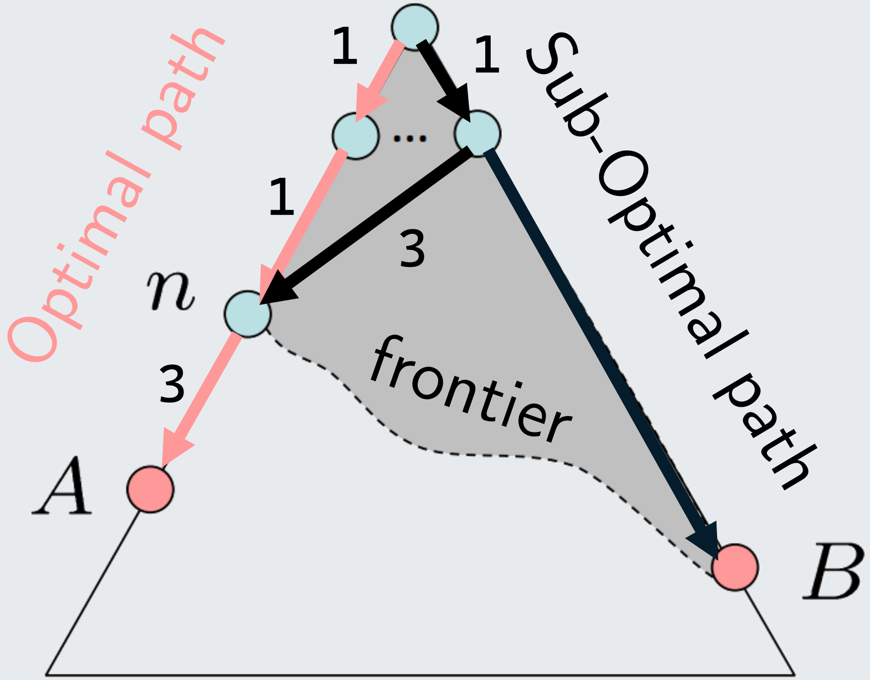
No!

- In graph search we won't visit a node that is *already popped from frontier*.
- It may happen that the nodes on optimal path is not explored in the correct order... and we won't return the optimal path.

Side Note for Graph Search

Graph Search	Characteristics	Pros	Cons
Version 1	Never push the same state into frontier twice.	Fast, guarantees $O(\lvert V\rvert + \lvert E\rvert)$ time.	Usually not optimal.
Version 2	May push the same state into frontier *if there is a more optimal path to it*.	Equivalent to tree search. Can find optimal paths.	Slower, as nodes are revisited.
Version 3	May push the same state into frontier *if it hasn't been popped from frontier yet*.	Faster than v2, and optimal if h is consistent.	Optimal path may be omitted for bad heuristics.

Problem 2.b



Goal: show that if h is consistent, then A* graph search is optimal.

Claim: along **all optimal paths to goals**, if n is the parent of n', then $f(n) \leq f(n')$.

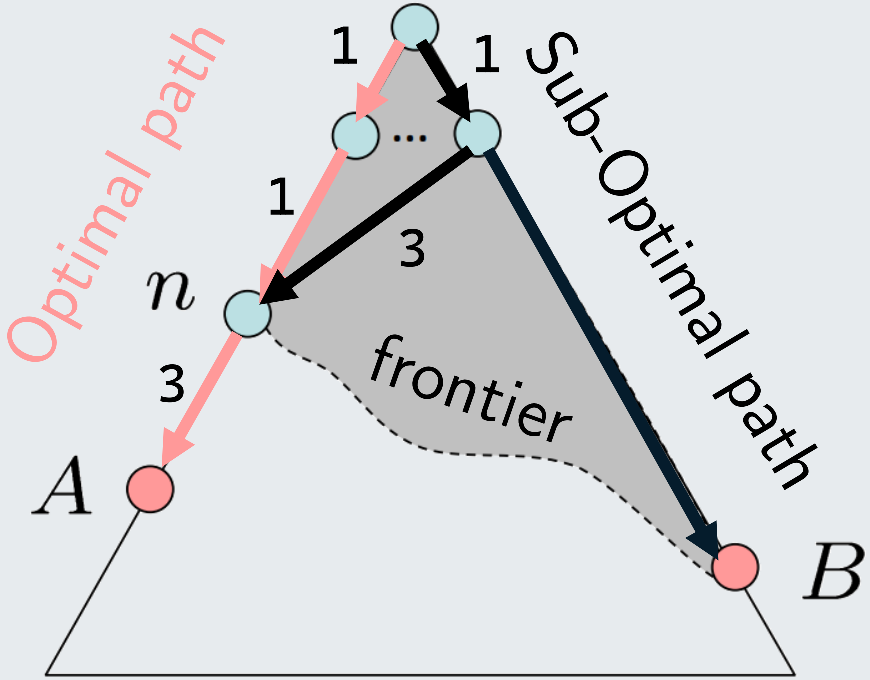
Proof.

$$
\begin{aligned}
f(n) &= g(n) + h(n) \\
&\leq \underbrace{g(n) + c(n, a, n')}_{=g(n')} + h(n') \\
&\leq f(n')
\end{aligned}
$$

Consistency

□

Problem 2.b



Goal: show that if h is consistent, then A* graph search is optimal.

Claim: along **all optimal paths to goals**, if n is the parent of n', then $f(n) \leq f(n')$.

$\Rightarrow$ f value *never decreases* along the path.

$\Rightarrow$ On the optimal path, *when a node n is popped*, all its ancestors must have been popped. i.e. *the optimal path to n is found*.

$\Rightarrow$ A* finds the optimal path to A.

Summary

- **admissible** (optimistic), $h(n) \leq h^*(n)$ for all n,
  - **Property**: for all states n on the path to goal G, $f(n) \leq f(G)$.
  - Tree search and Graph search (version 2) are optimal.
- **consistent**, $h(n) \leq c(n, a, n') + h(n')$ for all n and its successor n'.
  - **Property**: f value never decreases along an optimal path to a goal.
  - Graph search (version 2 & 3) is optimal.

Problem 3.a

Property: if $h(t) = 0$ for all goal state t, then h consistent $\Rightarrow$ h admissible.

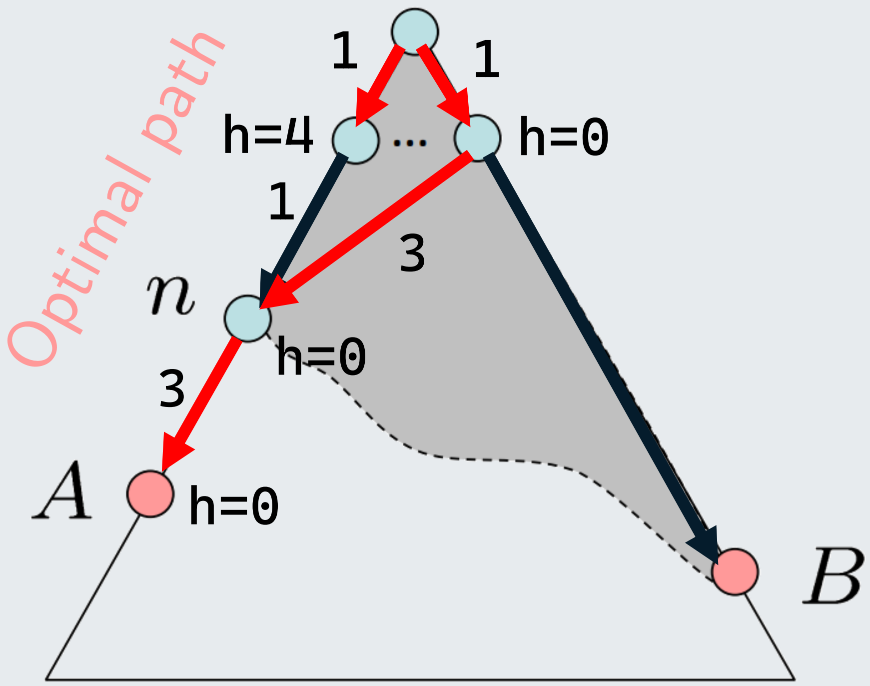
Proof. For all nodes n, suppose its optimal path to the goal is $n \to n_1 \to n_2 \to \cdots \to n_k \to t$, then

$$\begin{gathered} h(n) \leq c(n, a_0, n_1) + h(n_1) \\ h(n_1) \leq c(n_1, a_1, n_2) + h(n_2) \\ \vdots \\ h(n_k) \leq c(n_k, a_k, t) + h(t) \end{gathered}$$

Sum everything up, $h(n) \leq \text{optimal cost of path from } n \text{ to } t + h(t)$.

$\square$

Problem 3.b



Question: Is h admissible $\Rightarrow$ h consistent?

No! We have already gone through a counterexample.

Problem 4

Theorem: suppose

- $\mathcal{A}$ is a complete & deterministic graph search algorithm,
- $G = (V, E)$ is a finite graph.

there exists start node s_0 and goal node g such that $\mathcal{A}$ searches through all nodes.

Claim 1: For any start node s_0 and goal node g, $\mathcal{A}$ will visit g.

Claim 2: If we skip goal test, $\mathcal{A}$ will eventually visit all nodes. (proof by contradiction)

Problem 4

Theorem: suppose

- $\mathcal{A}$ is a complete & deterministic graph search algorithm,
- $G = (V, E)$ is a finite graph.

there exists start node s_0 and goal node g such that $\mathcal{A}$ searches through all nodes.

Idea:

- Fix s_0, run $\mathcal{A}$ without a goal test.
- Set the last visited node by $\mathcal{A}$ as goal g.

End of File

Thank you very much for your attention!

References

- D. Ler, "D. Ler, "Informed Search: Incorporating Domain Knowledge", 2023. [Online].
- S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," 3rd ed., Prentice Hall, 2010.
- N. Sharma. "Introduction to Artificial Intelligence", 2022. [Online]. Available:https://inst.eecs.berkeley.edu/~cs188/fa22/assets/notes/cs188-fa22-note02.pdf.