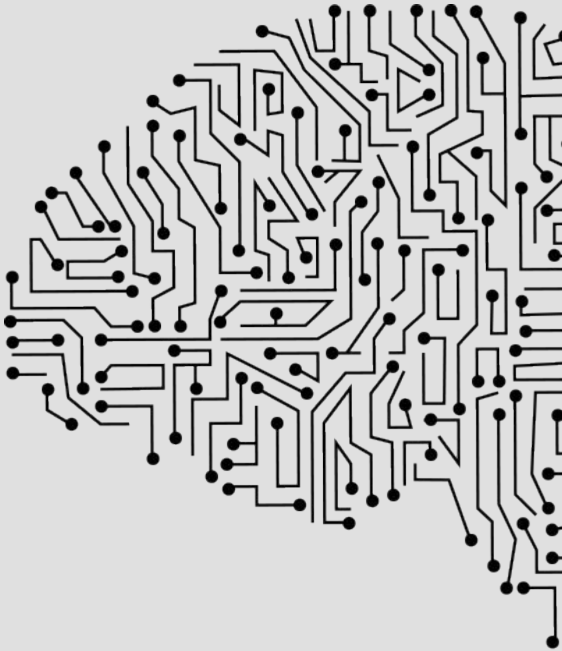
CS3243 INTRODUCTION TO ARTIFICIAL INTELLIGENCE

INFORMED SEARCH

Slides edited from Nicholas Teh

Kahoot!

Join at kahoot.it or with the Kahoot! app



KEY CONCEPTS

- **Heuristic**
  - Admissibility:
    - Given problem, come up with an admissible heuristic
    - Prove a heuristic is admissible
    - Show whether one heuristic dominates another
  - Consistency:
    - Prove a heuristic is consistent

KEY CONCEPTS

- **Informed Search Algorithms**
  - Best-First Search
  - Greedy Best-First Search
  - A* Search

HEURISTIC

- Informed search makes use of heuristics to make search faster by exploiting problem-specific knowledge. Order of node expansion still matters: which one more promising?
- [Definition] Heuristic: **guess of how far I am from the goal** and heuristic at every goal node should be 0.
  - *Trivial heuristics*: 0 for all nodes, infinity everywhere with 0 at the goal node
  - Actual distance/*Optimal heuristic* (seen problem before) is also a heuristic, but is unrealistic

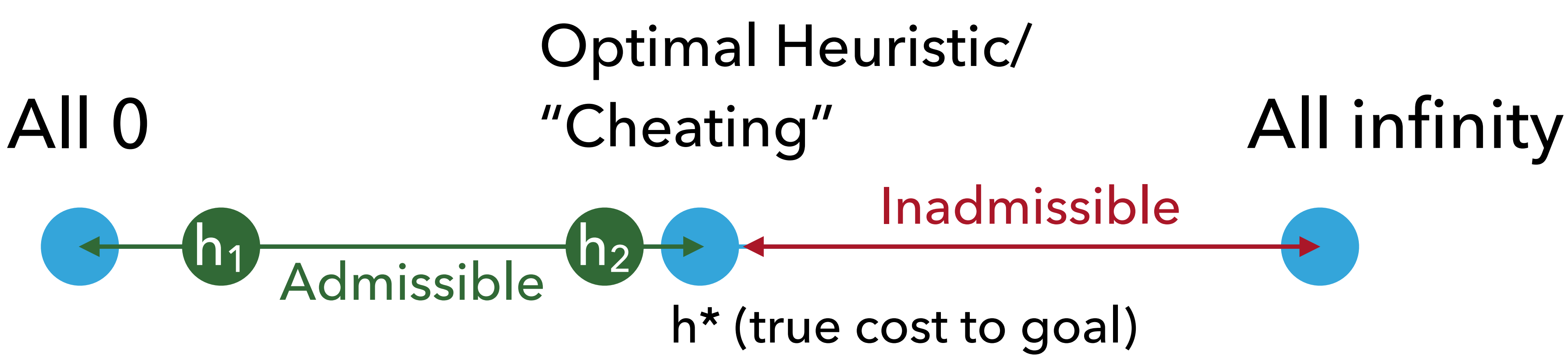
These 2 heuristics are prohibited in the quizzes/exams.

ADMISSIBILITY

- h(n) is admissible if **for all *n***, $h(n) \leq h^*(n)$
  - $h^*(n)$ is the true/optimal cost to reach goal state from *n*
  - <u>Never overestimates</u> the cost to reach goal state

DOMINANCE

- Usually defined (rather, more meaningful) for 2 admissible heuristics. But the same definition applies even if inadmissible heuristics are involved.
- If $h_2(n) \geq h_1(n)$ for all n, then h_2 dominates h_1. h_2 incurs <u>lower search cost</u> (underestimate less) than h_1, if h_1 and h_2 are admissible.
- Recall the line: if both are admissible, then h_2 is closer to the optimal heuristic than h_1.



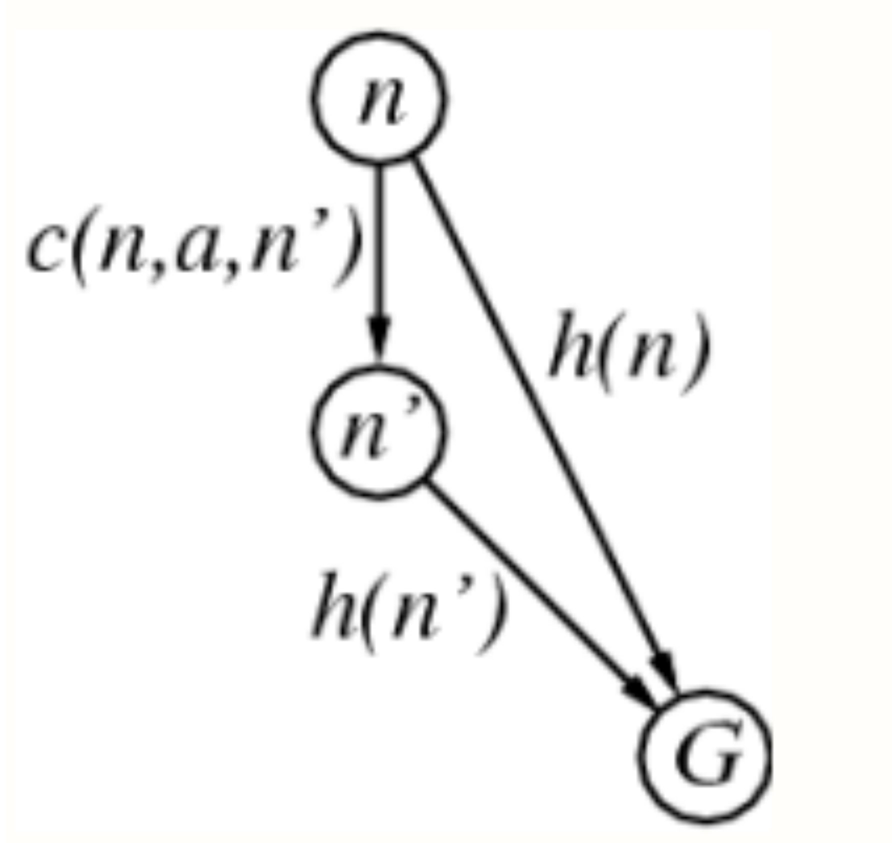
DOMINANCE

- **In A* search, a dominant heuristic leads to lower search costs.**
- A* expands nodes that have a lower $\boldsymbol{\hat{f}(n)} = \hat{g}(n) + h(n)$ value first
- The higher h(n) is, the less nodes A* expands, making it faster, and so lower search cost.

CONSISTENCY

- h(n) is consistent if for every node *n*, and every successor *n′* of *n* generated by action *a*,

  h(n) ≤ d(n, n′) + h(n′)



Basically the triangle inequality

CREATING ADMISSIBLE HEURISTICS (PROBLEM RELAXATION)

- A problem with fewer restrictions on actions is called a *relaxed problem* (easier to calculate). Think of it as bending the rules (e.g. instead of walking one step at a time, you can fly!).
- The more you bend, the more it underestimates the cost, because if you follow the rules, you have more restrictions, cost should be higher.

Properties:

- Any optimal solution in the original problem is also a solution in the relaxed problem.
- The cost of an optimal solution to the relaxed problem is an admissible heuristic for the original problem.
- Relax less is better (admissible with higher cost means closer to optimal!)

CREATING ADMISSIBLE HEURISTICS (PROBLEM RELAXATION)

- **An example: 8-puzzle (instantiation of k-puzzle!)**
- **Rules**: A tile can move from square A to square B if
  - (1) A is horizontally or vertically adjacent to B, and
  - (2) B is blank
- From this, we can generate three relaxed problems:

Bend/ignore rules!

  - a tile can move from A to B if A is adjacent to B (Manhattan distance) (ignore rule (2))
  - a tile can move from A to B if B is blank (ignore rule (1))
  - a tile can move from A to B (# misplaced tiles) (ignore rule (1) & (2))

INFORMED SEARCH ALGORITHMS

- **Best-First Search**
  - Use an evaluation function f(n) for each node n
  - Cost estimate: expand node with lowest f first.
  - Note special cases (different choices of f: greedy, A*, etc.)
- **Greedy Best-First Search**
  - Evaluation function $\hat{f}(n) = h(n)$ (heuristic function)
    = estimate cost of cheapest path from n to goal
  - At each stage, expands node that <u>appears</u> to be closest to goal
  - Note special cases (difference choices of h may yield similar algorithms to what we know)

Informed Search

What if there is more than one goal?

Data structure for frontier?

g(n)

Minimum path cost from s to n

ĝ(n)

Current best known path cost from s to n → Updated during execution!

h(n)

Heuristic function, h

- Approximate path cost from n to (the closest) goal g
- How much farther to go until the goal?
- Properties: admissibility and consistency of h

f(n)

Evaluation function adopted by a specific algorithm → Expand node with lowest $f(n)$ first (similar to UCS!)

Note: $f(n) = g(n) + h(n)$

Greedy Best-first Search: $f(n) = h(n)$

A* Search: $f(n) = \hat{g}(n) + h(n)$

TUTORIAL 3 QUESTION 2 (GREEDY BEST-FIRST SEARCH)

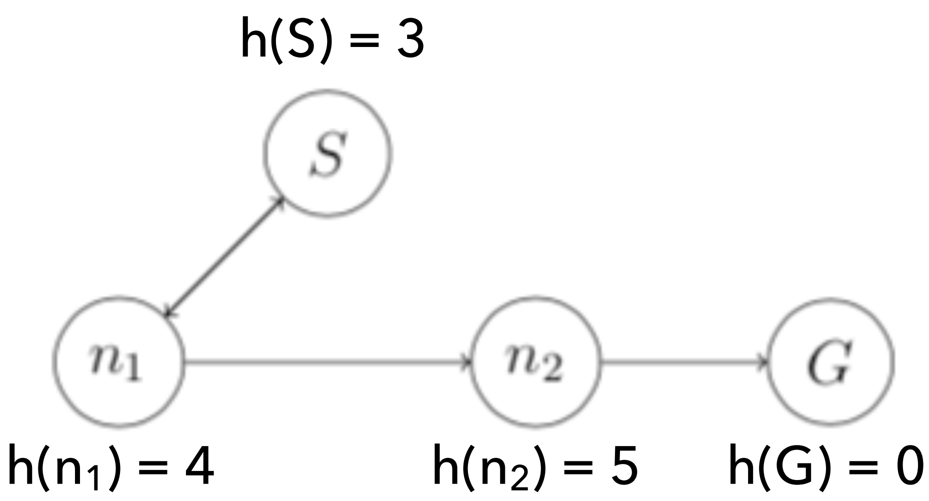
- **(a) Provide a counter-example to show that the tree-based variant for the greedy best-first search algorithm is incomplete.**

TUTORIAL 3 QUESTION 2 (GREEDY BEST-FIRST SEARCH)

- **(a) Provide a counter-example to show that the tree-based variant for the greedy best-first search algorithm is incomplete.**

ANSWER

- Stuck in an infinite loop because of short-sightedness
- Each time S is explored, we add n_1 to the front of the Frontier, and each time n_1 is explored, we add S to the front of the Frontier. n_2 is never at the front of the Frontier. This causes the greedy best-first search algorithm to continuously loop over S and n1.



TUTORIAL 3 QUESTION 2 (GREEDY BEST-FIRST SEARCH)

- **(b) Briefly explain why the graph-based variant of the greedy best-first search algorithm is complete.**

TUTORIAL 3 QUESTION 2 (GREEDY BEST-FIRST SEARCH)

- **(b) Briefly explain why the graph-based variant of the greedy best-first search algorithm is complete.**

ANSWER

- Assuming a finite branching factor, b, the graph-based variant of the greedy best-first search algorithm will **eventually visit all states** within the search space and thus find a goal state
- *(We always assume finite number of states in state space/nodes in search graph - not the same as finite depth* in a search tree)

TUTORIAL 3 QUESTION 2 (GREEDY BEST-FIRST SEARCH)

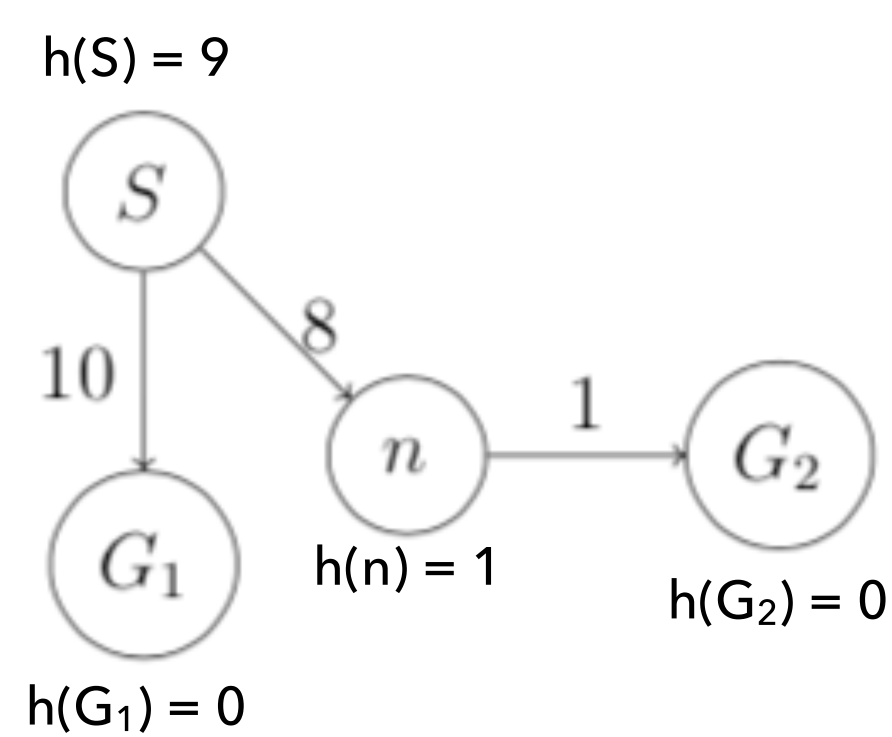
- **(c) Provide a counter-example to show that neither variant of the greedy best-first search algorithm is optimal.**

TUTORIAL 3 QUESTION 2 (GREEDY BEST-FIRST SEARCH)

- **(c) Provide a counter-example to show that neither variant of the greedy best-first search algorithm is optimal.**

ANSWER

- With either variant of the greedy best-first search algorithm, when S is explored, G_1 would be added to the front of the Frontier and then explored next, resulting in the algorithm returning the non-optimal S → G_1 path.



TUTORIAL 3 QUESTION 3 (TREE-BASED A* SEARCH)

- **Prove that the tree-based variant of the A* search algorithm is optimal when an admissible heuristic is utilised.**

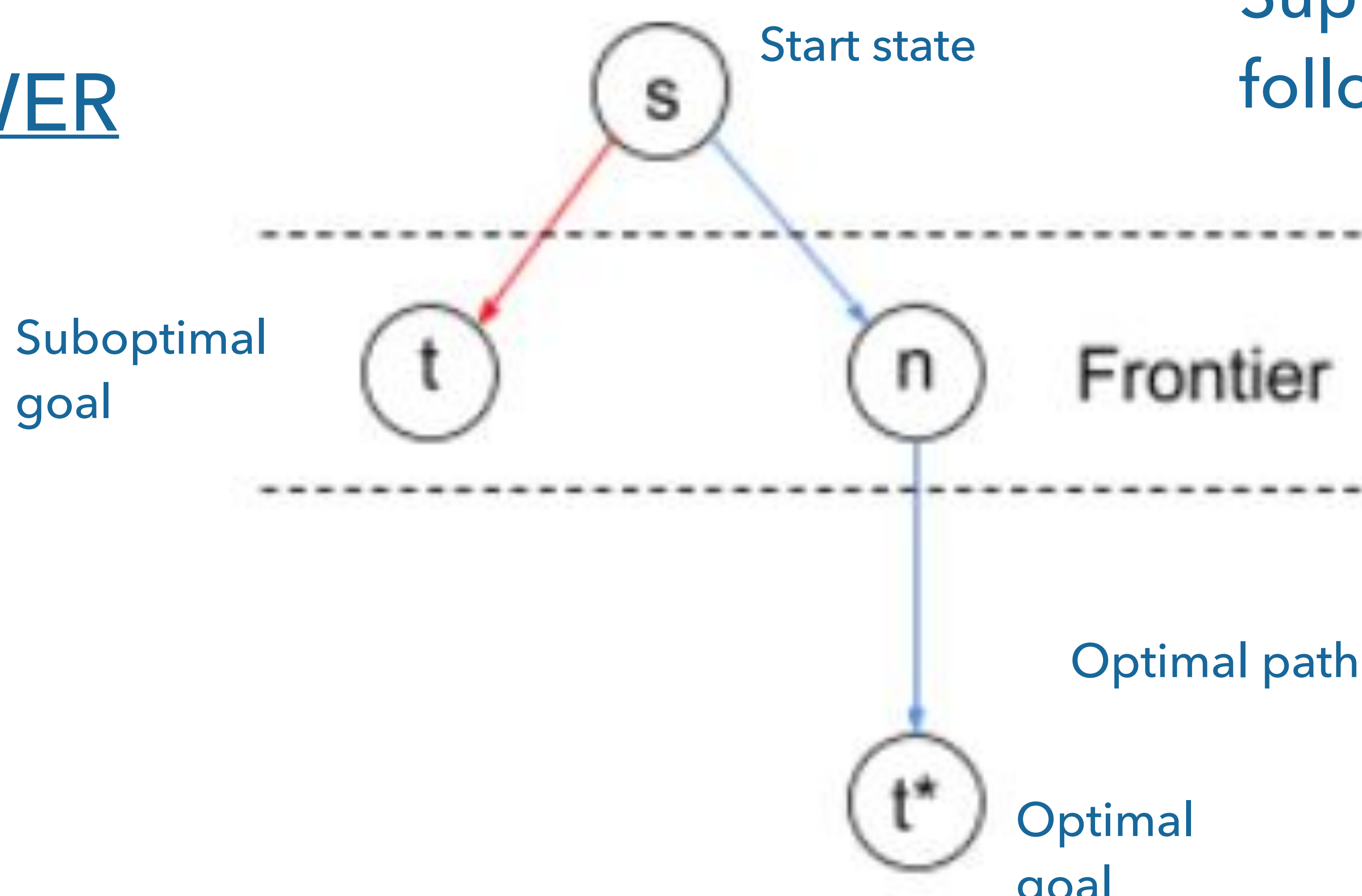
TUTORIAL 3 QUESTION 3 (TREE-BASED A* SEARCH)

- **Prove that the tree-based variant of the A* search algorithm is optimal when an admissible heuristic is utilised.**

Suppose we have the following search tree.

ANSWER

-



An optimal solution implies that **any intermediate node on the optimal path** *n* must be expanded before suboptimal goal *t*.

TUTORIAL 3 QUESTION 3 (TREE-BASED A* SEARCH)

ANSWER (Continued)

- We're going to prove it by contradiction.
- Assume, for a contradiction, that suboptimal goal *t* is expanded before any optimal path intermediate node *n*
- *Then* $f(t) \leq f(n)$, since A* uses f to determine expansion
- However, since *t* is not on the optimal path, and *t** is optimal, we have $f(t) > f(t^*) = g(t^*) + h(t^*)$.
- Since t* is a goal node, $h(t^*) = 0$, so we get $f(t) > g(t^*)$.
- $f(t) > g(t^*) = g(n) + d(n, t^*)$ where $d(n, t^*)$ is actual cost from n to t*

TUTORIAL 3 QUESTION 3 (TREE-BASED A* SEARCH)

ANSWER (Continued)

- $f(t) > g(t^*) = g(n) + d(n, t^*) = g(n) + h^*(n)$
  where $d(n, t^*)$ is actual cost from n to t*
- $f(t) > g(n) + \underline{h^*(n)} \geq g(n) + \underline{h(n)}$ because h(n) is admissible (question says an admissible heuristic is used)
- $\mathbf{f(t) >} g(n) + h(n) = \mathbf{f(n)}$
- Which contradicts $f(t) \leq f(n)$.
- *Note: we do not consider f(t) = f(n) since that will mean f(t) is equally optimal - we defined optimal goal t* and suboptimal goal t*

TUTORIAL 3 QUESTION 4 (GRAPH-BASED A* SEARCH)

- **Prove that the graph-based variant of the A* search algorithm is optimal when a consistent heuristic is utilised.**

TUTORIAL 3 QUESTION 4 (GRAPH-BASED A* SEARCH)

- **Prove that the graph-based variant of the A* search algorithm is optimal when a consistent heuristic is utilised.**

ANSWER Let n′ be a successor node of n by taking some action *a*.

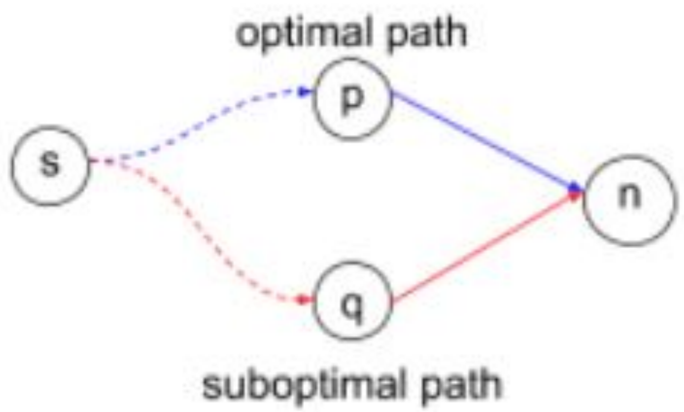
- A heuristic h(n) is consistent if for all *n*, $h(n) \leq d(n, n') + h(n')$
- LEMMA: $\hat{f}(n') = \underline{\hat{g}(n')} + h(n') = \underline{\hat{g}(n) + d(n, n')} + h(n')$

$$\geq \hat{g}(n) + h(n) \quad \text{by consistency}$$
$$= \hat{f}(n)$$

- So we get $\hat{f}(n') \geq \hat{f}(n)$. The evaluation function at a later node is always ≥ evaluation function at earlier node. Let's prove by contradiction.

ANSWER (Continued)

- What that also means is that A* search explores nodes in a non-decreasing order of $\hat{f}$ value;
  - Essentially, with each exploration, we may add a new contour (similar to how UCS explores nodes in a non-decreasing order of g value)
  - When A* expands n, the optimal path to *n* has been found (again, similar to UCS)



ANSWER (Continued)

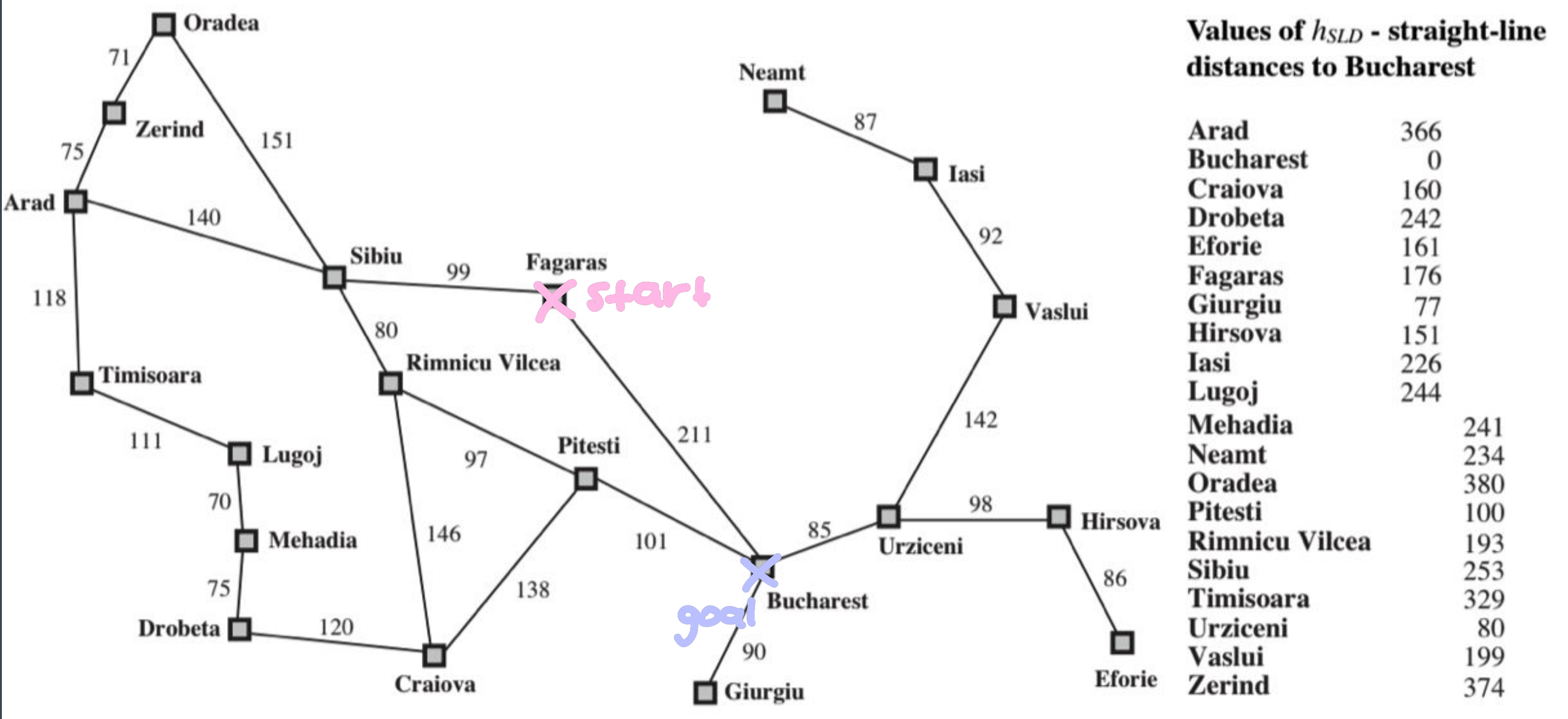
- Proof by contradiction:
  - Let *n* be the reached by a suboptimal path (via q) first rather than the optimal path. *n* is also on the optimal path though (via *p*)
  - On the suboptimal path, *n* is expanded before *p*, thus, we have: $\hat{f}(n) < \hat{f}(p)$
  - However, since *p* precedes *n* on the optimal path, $\hat{f}(p) < \hat{f}(n)$
  - Contradiction! Note that we don't consider case where $\hat{f}(p) = \hat{f}(n)$, otherwise either d(p, n) = 0 and p = n by definition of consistency.

Q5(a)

$$h(n) = |h_{SLD}(\text{Craiova}) - h_{SLD}(n)|$$

What do you need to keep track of?

(g, h, f) for each node

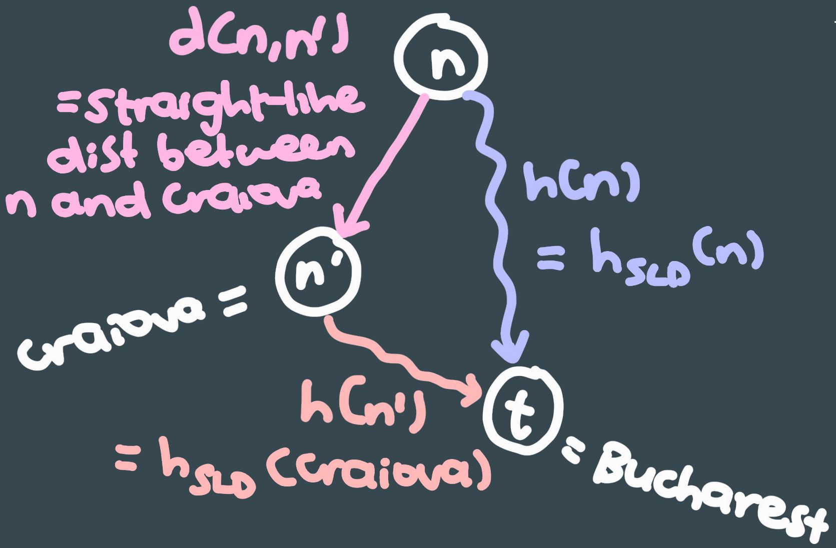


Q5(b) Prove that *h(n)* is admissible.

$$h(n) = |h_{SLD}(\text{Craiova}) - h_{SLD}(n)|$$

Note that the goal node is <u>Craiova</u>, not Bucharest!

Let *n'* denote Craiova and *t* denote Bucharest.



Using the triangle inequality, we have

(1) $d(n, Craiova) + h_{SLD}(n) \geq h_{SLD}(Craiova)$

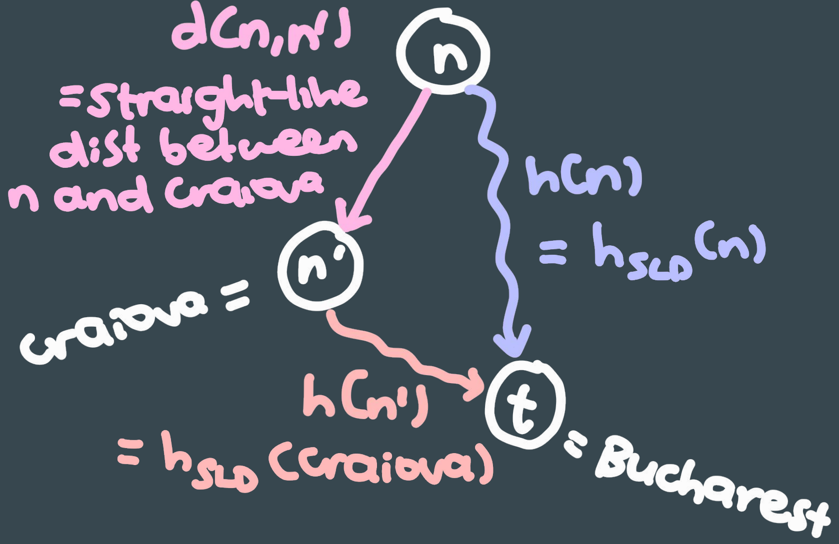
$\Rightarrow\ d(n, Craiova) \geq h_{SLD}(Craiova) - h_{SLD}(n)$

(2) $d(n, Craiova) + h_{SLD}(Craiova) \geq h_{SLD}(n)$

$\Rightarrow\ d(n, Craiova) \geq h_{SLD}(n) - h_{SLD}(Craiova)$

Q5(b) (cont.)

$$h(n) = |h_{SLD}(\text{Craiova}) - h_{SLD}(n)|$$



$d(n, \mathit{Craiova}) \geq h_{SLD}(\mathit{Craiova}) - h_{SLD}(n)$
$d(n, \mathit{Craiova}) \geq h_{SLD}(n) - h_{SLD}(\mathit{Craiova})$

$\Rightarrow d(n, \mathit{Craiova}) \geq |h_{SLD}(n) - h_{SLD}(\mathit{Craiova})| = h(n)$

Now, we let $h^*(n)$ be the true cost of reaching Craiova and $D(n) = d(n, \mathit{Craiova})$.

We have $h^*(n) \geq D(n) \geq h(n)$.
$\Rightarrow h(n)$ is admissible!

TUTORIAL 3 QUESTION 6 (PROVE ADMISSIBLE)

- **For 2 admissible heuristics h_1 and h_2, and where h_2 dominates h_1, we define:**

  $$h_3 = (h_1 + h_2)/2 \qquad\qquad h_4 = h_1 + h_2$$

  Are h_3 and h_4 admissible? If they are, compare their dominance with respect to h_1 and h_2.

TUTORIAL 3 QUESTION 6 (PROVE ADMISSIBLE)

ANSWER

- If I can show the heuristic is dominated by an admissible heuristic, then I can prove it's admissible.
- Since h_2 dominates h_1, $h_1(s) \leq h_2(s)$ for all *n*,

$$\underbrace{h_3(n) =}_{\text{By the definition of } h_3} \frac{h_1(n) + h_2(n)}{2} \underbrace{\leq}_{\text{Since } h_2 \text{ dominates } h_1} \frac{h_2(n) + h_2(n)}{2} \underbrace{=}_{\text{Simple arithmetic}} h_2(n) \underbrace{\leq}_{\text{Because } h_2 \text{ is admissible}} h^*(n)$$

So h_3 is admissible

TUTORIAL 3 QUESTION 6 (PROVE ADMISSIBLE)

ANSWER (Continued)

- h_4 is not admissible (in general), and in this specific scenario if we were to talk about 8-puzzle.
  - h_1 is Number of misplaced tiles
  - h_2 is Manhattan distance

Consider a board/state *n* where moving one tile will reach the goal. Then both heuristics will give 1, and $h_4(s)$ will give 2, not admissible

TUTORIAL 3 QUESTION 7A (CONSISTENT → ADMISSIBLE)

- **If a heuristic is consistent, it is also admissible. Prove it.**

TUTORIAL 3 QUESTION 7A (CONSISTENT → ADMISSIBLE)

- **If a heuristic is consistent, it is also admissible. Prove it.**

ANSWER

- Consistency: $h(n) \leq d(n, n') + h(n')$ for all n, n′ (n′ is successor of n)
- So do induction/ start from the end
  $h(n_k) \leq d(n_k, G) + h(G) = h^*(n_k)$
  $h(n_{k-1}) \leq d(n_{k-1}, n_k) + d(n_k, G) + h(G) = h^*(n_{k-1})$
  $h(n_{k-2}) \leq d(n_{k-2}, n_{k-1}) + .. + d(n_k, G) + h(G) = h^*(n_{k-2})$
  ...
  $h(n_1) \leq ..... = h^*(n_1)$
- So the heuristic is admissible for all nodes *n*!

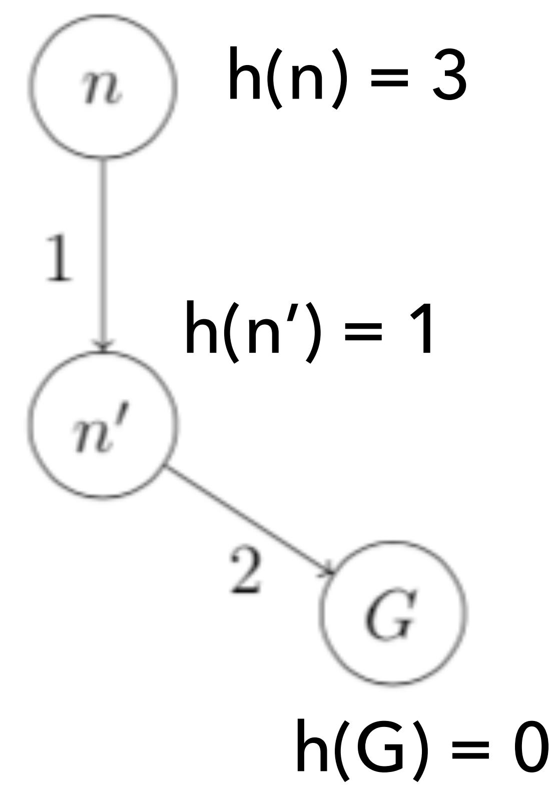
Note: $h^*(n)$ is defined as the cost of optimal path from n to G

TUTORIAL 3 QUESTION 7B (ADMISSIBLE DOESN'T → CONSISTENT)

- **Give an example of an admissible heuristic function that is not consistent.**

TUTORIAL 3 QUESTION 7B (ADMISSIBLE DOESN'T → CONSISTENT)

- **Give an example of an admissible heuristic function that is not consistent.**



ANSWER

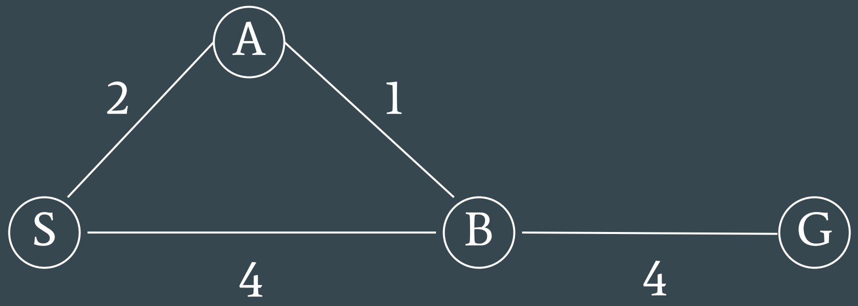
- Then, *h* is admissible, since
  h(n) = 3 ≤ h*(n) = 1 + 2 = 3
  h(n′) = 1≤ h*(n) = 2
- But *h* is not consistent because
  3 = h(n) > d(n, n′) + h(n′) = 2

Q8

There are several possible solutions one might consider.

First, let us set

$$h(S) = 7$$
$$h(B) = 0$$
$$h(A) = 3$$
$$h(G) = 0$$



By observation, h is indeed admissible.

Q8

At the start,

Priority Queue: [(S, 7)]

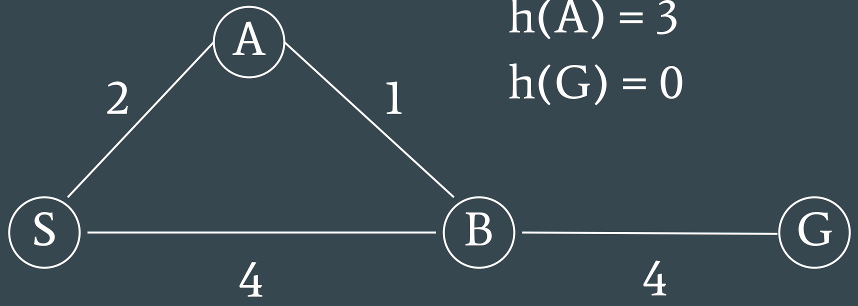
- $f(S) = g(S) + h(S) = 0 + 7$

$h(S) = 7$

$h(B) = 0$

$h(A) = 3$

$h(G) = 0$



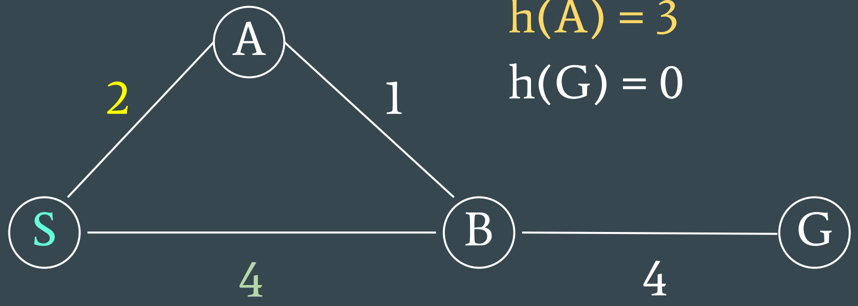
Q8

Explore (S, 7), Add A and B into PQ

Priority Queue: [(B, 4), (A, 5)]

- $f(A) = g(A) + h(A) = 2 + 3 = 5$
- $f(B) = g(B) + h(B) = 4 + 0 = 4$

$h(S) = 7$
$h(B) = 0$
$h(A) = 3$
$h(G) = 0$

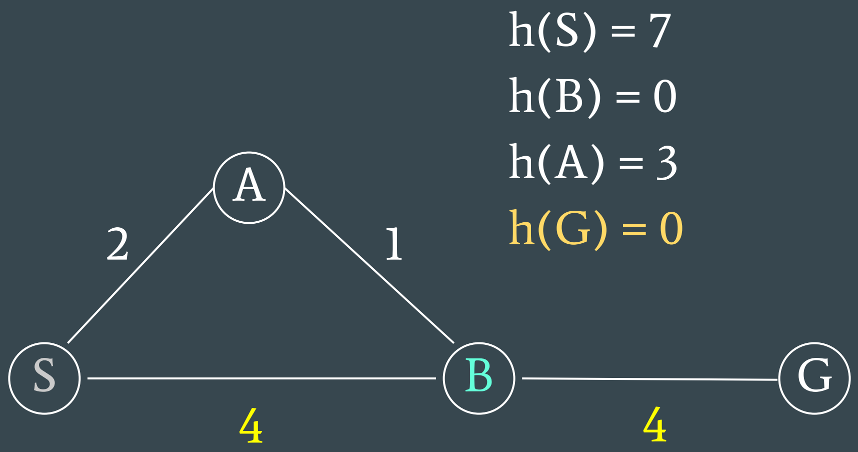


Q8

Explore (B, 4), Add G into PQ

Priority Queue: [(A, 5), (G, 8)]

- $f(G) = g(G) + h(G) = 8 + 0 = 8$

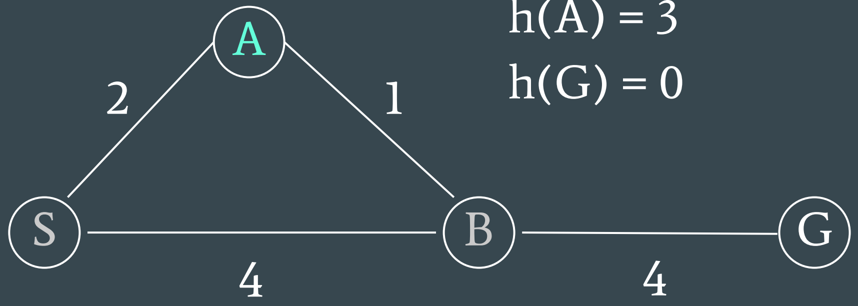


Q8

Explore (A, 5), does not add any neighbours because this is graph search

Priority Queue: [(G, 8)]

$h(S) = 7$
$h(B) = 0$
$h(A) = 3$
$h(G) = 0$

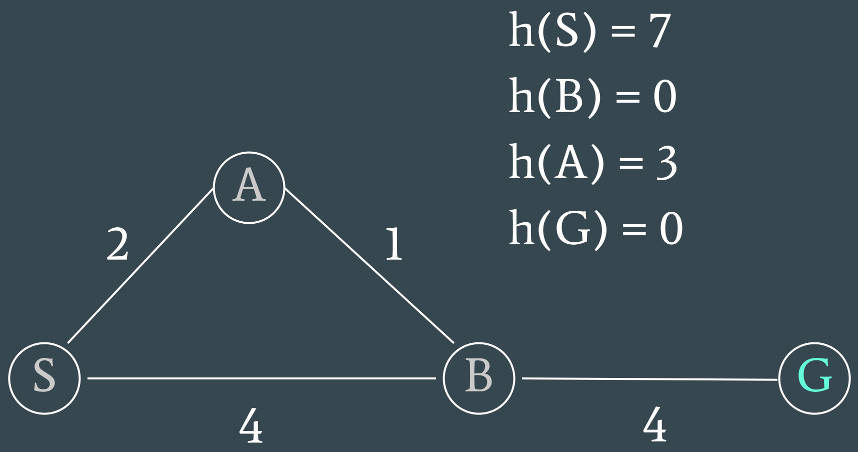


Q8

Explore (G, 8), end up with a suboptimal solution :(

- optimal is (G, 7)

Priority Queue: [ ]



TUTORIAL 3 QUESTION 9 (ALSO AY19/20 SEM 2 MIDTERM EXAM)

- **PROVE/DISPROVE: Suppose that the A* search algorithm utilises $f(n) = w \times g(n) + (1 - w) \times h(n)$, where $0 \leq w \leq 1$ (instead of $f(n) = g(n) + h(n)$). For any value of w, an optimal solution will be found whenever h is a consistent heuristic.**

TUTORIAL 3 QUESTION 9 (ALSO AY19/20 SEM 2 MIDTERM EXAM)

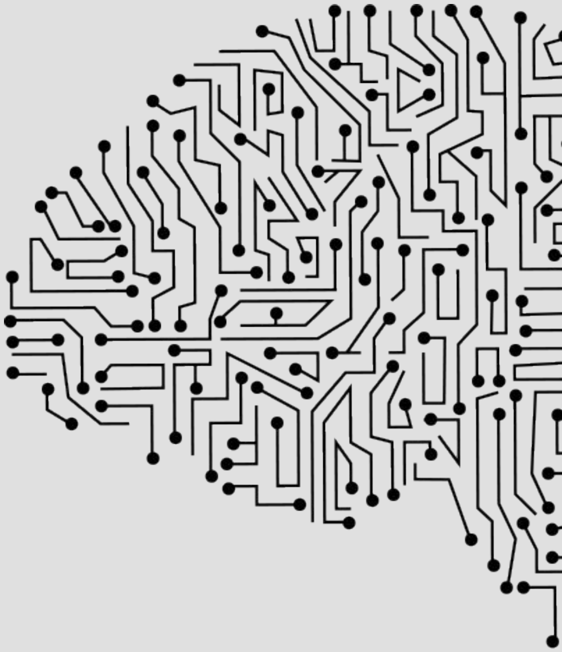
- **PROVE/DISPROVE: Suppose that the A* search algorithm utilises $f(n) = w \times g(n) + (1 - w) \times h(n)$, where $0 \leq w \leq 1$ (instead of $f(n) = g(n) + h(n)$). For any value of w, an optimal solution will be found whenever h is a consistent heuristic.**

ANSWER

- False. When $w = 0$, we get greedy best-first search, which is suboptimal (as proven in Question 2c)
- You do not need to prove greedy best first search if you quote the property that it is suboptimal as proven during in tutorials.
- Because this question does not explicitly ask you to prove it.

Questions?

https://forms.gle/McAnYhnJ17g8Fi28A



Thank you!