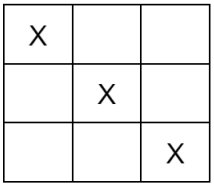
National University of Singapore
School of Computing
CS3243 Introduction to AI

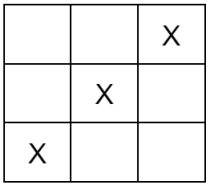
Midterm Examination Review (Solutions)

---

1. A magic square corresponds to a square array of size n (i.e., an $n \times n$ grid) that is filled with the sequence of positive integers from k to $k + n^2 - 1$. The constraints that make any such assignment a magic square are that the sum of each row, column, and both the primary and secondary diagonals (see below for an example of these diagonals over the 3×3 magic square) are equal.



Primary Diagonal



Secondary Diagonal

Define the search problem corresponding to an arbitrary magic square with parameters n and k. **Your search problem definition should ensure that the search space is efficient.**

Solution: First, we denote following:

(a) The magic-number (i.e., the value that each row, column, and diagonal must sum to) $m = [(n^2 + k - 1)(n^2 + k))/2 - ((k - 1)k)/2]/n$

(b) A sequence of n integers (without repeats) in the interval $[k, k + n^2 - 1]$, s. Note that there are many such sequences possible, which may be indexed as $s_1, \ldots, s_p$.

(c) The list of all possible unique sequences s, $S = \langle s_1, \ldots, s_p \rangle$, where the sum of the integers in each $s_i \in S$ is m.

The magic square problem may thus be defined as the following search problem.

(a) State representation: an array of length n, A, where each element in A corresponds to an assignment of some $s_i \in S$ to the row of the magic square (i.e., element i of the array corresponds to row $i + 1$ of the magic square). Thus, each state corresponds to the partial assignment of up to n unique $s_i \in S$, denoted by an array of indices (e.g., with $n = \langle 5, 3, 1 \rangle$ would correspond to s_5 being assigned to row 1 of the magic square, s_3 being assigned to row 2 of the magic square, and s_1 being assigned to row 3 of the magic square.

(b) Initial state: the empty array A

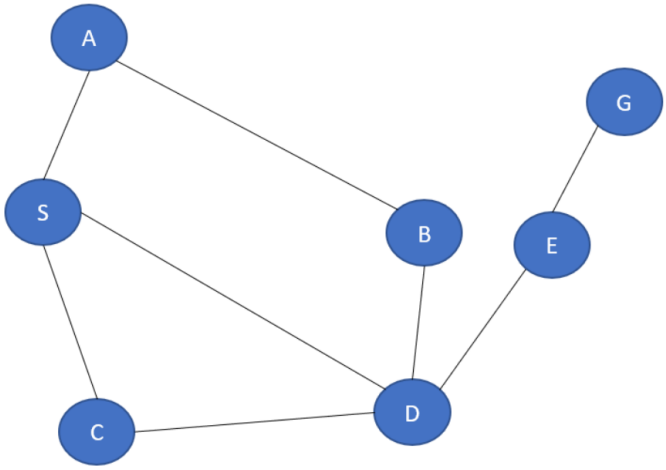
(c) Actions: assign each possible s_i to the next empty index in A; a possible assignment s_i is only allowed if each partial column and diagonal formed is itself a possible sequence in S that has not already been assigned.

(d) Goal test: if all the columns and diagonals based on the assignments sum to m.

(e) Transition model: assigning the specific index of some $s_i \in S$, i.e., i, to the array from the previous state.

(f) Cost function: costs are uniform (e.g., all 1).

2. (a) Consider the undirected graph depicted in the Figure below. Let S be the initial state and G be the goal state.



Trace the sequence of nodes **expanded** (i.e., popped off the frontier) by the **tree search** variant of the **Depth-limited Search** algorithm on the given graph. You are to assume that the maximum depth is 2.

Further, assume that the depth of the start node is 0 and that ties when **pushing to the frontier** are resolved based on **reverse alphabetical order** (e.g., B is pushed before A).

You should express your answer in the form X–Y–Z (i.e., no spaces, all uppercase letters, delimited by the dash (–) character), which, for example, corresponds to the order X, Y, and Z.

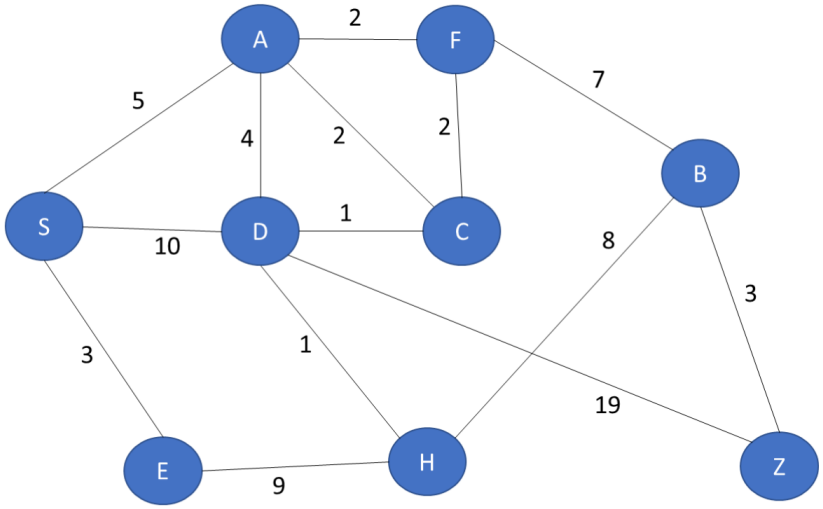
Solution:

DLS	**Stack (top = front)**	**Expanded**
-	$\langle S, 0\rangle$	-
Pop S	$\langle A, 1\rangle, \langle C, 1\rangle, \langle D, 1\rangle$	S
Pop A	$\langle B, 2\rangle, \langle S, 2\rangle, \langle C, 1\rangle, \langle D, 1\rangle$	S-A
Pop B	$\langle S, 2\rangle, \langle C, 1\rangle, \langle D, 1\rangle$	S-A-B
Pop S	$\langle C, 1\rangle, \langle D, 1\rangle$	S-A-B-S
Pop C	$\langle D, 2\rangle, \langle S, 2\rangle, \langle D, 1\rangle$	S-A-B-S-C
Pop D	$\langle S, 2\rangle, \langle D, 1\rangle$	S-A-B-S-C-D
Pop S	$\langle D, 1\rangle$	S-A-B-S-C-D-S
Pop D	$\langle B, 2\rangle, \langle C, 2\rangle, \langle E, 2\rangle, \langle S, 2\rangle$	S-A-B-S-C-D-S-D
Pop B	$\langle C, 2\rangle, \langle E, 2\rangle, \langle S, 2\rangle$	S-A-B-S-C-D-S-D-B
Pop C	$\langle E, 2\rangle, \langle S, 2\rangle$	S-A-B-S-C-D-S-D-B-C
Pop E	$\langle S, 2\rangle$	S-A-B-S-C-D-S-D-B-C-E
Pop S	-	S-A-B-S-C-D-S-D-B-C-E-S

Sequence of nodes explored is S-A-B-S-C-D-S-D-B-C-E-S.

The goal was not found.

(b) Consider the undirected graph depicted in the Figure below. Let S be the initial state. The cost of each action is as indicated.



You are to assume the use of the following heuristic.

State n	$h(n)$
S	16
A	12
B	3
C	2
D	11
E	15
F	9
H	10
Z	0

i. Let H be the goal state. Trace the sequence of nodes **expanded** (i.e., popped off the frontier) by the **limited-graph search** variant of the **Uniform-Cost Search** algorithm on the given graph.

ii. Let Z be the goal state. Trace the sequence of nodes **expanded** (i.e., popped off the frontier) by the **limited-graph search** variant of the $\boldsymbol{A^*}$ algorithm on the given graph.

For both parts, assume that ties when **enqueueing onto the frontier** are broken based on **alphabetical order**.

You should express your answer in the form X–Y–Z (i.e., no spaces, all uppercase letters, delimited by the dash (–) character), which, for example, corresponds to the order X, Y, and Z.

Solution:

i. Note that for the following trace, the Reached hash table is the same as Expanded without duplicates.

Expanded	**Priority Queue over** $f(n) = g(n)$
-	$\langle S, - \rangle$
S	$\langle E(S), 3 \rangle$, $\langle A(S), 5 \rangle$, $\langle D(S), 10 \rangle$
S-E	$\langle A(S), 5 \rangle$, $\langle D(S), 10 \rangle$, $\langle H(S, E), 12 \rangle$
S-E-A	$\langle C(S, A), 7 \rangle$, $\langle F(S, A), 7 \rangle$, $\langle D(S, A), 9 \rangle$, $\langle D(S), 10 \rangle$, $\langle H(S, E), 12 \rangle$
S-E-A-C	$\langle F(S, A), 7 \rangle$, $\langle D(S, A, C), 8 \rangle$, $\langle D(S, A), 9 \rangle$, $\langle F(S, A, C), 9 \rangle$, $\langle D(S), 10 \rangle$, $\langle H(S, E), 12 \rangle$
S-E-A-C-F	$\langle D(S, A, C), 8 \rangle$, $\langle D(S, A), 9 \rangle$, $\langle F(S, A, C), 9 \rangle$, $\langle D(S), 10 \rangle$, $\langle H(S, E), 12 \rangle$, $\langle B(S, A, F), 14 \rangle$
S-E-A-C-F-D	$\langle D(S, A), 9 \rangle$, $\langle F(S, A, C), 9 \rangle$, $\langle H(S, A, C, D), 9 \rangle$, $\langle D(S), 10 \rangle$, $\langle H(S, E), 12 \rangle$, $\langle B(S, A, F), 14 \rangle$, $\langle Z(S, A, C, D), 27 \rangle$
S-E-A-C-F-D-D	$\langle F(S, A, C), 9 \rangle$, $\langle H(S, A, C, D), 9 \rangle$, $\langle D(S), 10 \rangle$, $\langle H(S, A, D), 10 \rangle$, $\langle H(S, E), 12 \rangle$, $\langle B(S, A, F), 14 \rangle$, $\langle Z(S, A, C, D), 27 \rangle$, $\langle Z(S, A, D), 28 \rangle$
S-E-A-C-F-D-D-F	$\langle H(S, A, C, D), 9 \rangle$, $\langle D(S), 10 \rangle$, $\langle H(S, A, D), 10 \rangle$, $\langle H(S, E), 12 \rangle$, $\langle B(S, A, F), 14 \rangle$, $\langle B(S, A, C, F), 16 \rangle$, $\langle Z(S, A, C, D), 27 \rangle$, $\langle Z(S, A, D), 28 \rangle$
S-E-A-C-F-D-D-F-H	Goal Found

Sequence of nodes explored is S-E-A-C-F-D-D-F-H.
The goal was found via path S-A-C-D-H **with a path cost of 9**.

The following solution is also acceptable.

- S-E-A-C-F-D-H
  This applies to a trace where a check is performed before popping such that nodes in the reached/visited hash table are not popped (and thus not expanded).

ii. Note that for the following trace, the Reached hash table is the same as Expanded without duplicates.

Expanded	**Priority Queue over** $f(n) = g(n) + h(n)$
-	$\langle S, 16\rangle$
S	$\langle A(S), 17\rangle$, $\langle E(S), 18\rangle$, $\langle D(S), 21\rangle$
S-A	$\langle C(S, A), 9\rangle$, $\langle F(S, A), 16\rangle$, $\langle E(S), 18\rangle$, $\langle D(S, A), 20\rangle$, $\langle D(S), 21\rangle$
S-A-C	$\langle F(S, A), 16\rangle$, $\langle E(S), 18\rangle$, $\langle F(S, A, C), 18\rangle$, $\langle D(S, A, C), 19\rangle$, $\langle D(S, A), 20\rangle$, $\langle D(S), 21\rangle$
S-A-C-F	$\langle B(S, A, F), 17\rangle$, $\langle E(S), 18\rangle$, $\langle F(S, A, C), 18\rangle$, $\langle D(S, A, C), 19\rangle$, $\langle D(S, A), 20\rangle$, $\langle D(S), 21\rangle$
S-A-C-F-B	$\langle Z(S, A, F, B), 17\rangle$, $\langle E(S), 18\rangle$, $\langle F(S, A, C), 18\rangle$, $\langle D(S, A, C), 19\rangle$, $\langle D(S, A), 20\rangle$, $\langle D(S), 21\rangle$, $\langle H(S, A, F, B), 32\rangle$
S-A-C-F-B-Z	Goal Found

Sequence of nodes explored is S-A-C-F-B-Z.
The goal was found via path S-A-F-B-Z with a path cost of 17.

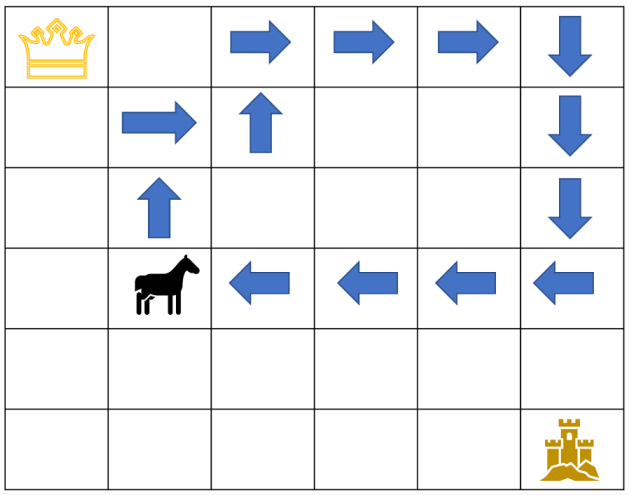
3. Once upon a time, in a kingdom far away, there lived a King. This King would often make trips and would require his advisors to find the most efficient routes for these trips. Suppose that one such trip is as follows.

   The King's position is currently quite far away from his palace. To find out the fastest route back to his palace, the King would like you, as one of his advisors, to model his commute as a search problem. His surroundings correspond to an $M \times N$ square grid, where he starts out in the top-most and left-most square. The goal is to find his way to his palace, which is located at the bottom-most and right-most square, in the least number of timesteps. He can only move left, right, up, or down, and he cannot move anywhere outside the grid.

   To speed up his commute, there is a single fixed horse route on the grid. On this fixed horse route, his horse travels in a cycle at a constant rate of three spaces per timestep. You are to assume that any square along this route is a valid horse stop for the King to mount or dismount.

   The King knows the route of the horse and if he is on the same square as the horse, he can mount the horse and travel at a rate of three spaces per timestep along the horse route. He can travel on the route for any number of timesteps and get off at any square adjacent to the path. However, as the horse will get tired, the King can only mount and dismount the horse at most once per commute.

   Below is an example start state on a 6 by 6 grid. The arrows represent the horse's route. Note that this is just an example route, and the horse route does not necessarily have to be in this pattern. For the following questions, consider the general case, not the specific example below. Also consider each part separately (i.e., do not carry over assumptions).



(a) **Determine if each of the following heuristics is admissible.**

If you feel that none are admissible, select only the option "None of the options are admissible". For each option, briefly, but clearly explain, why it is admissible or inadmissible.

Answers without rationale will not be awarded any marks.

i. The Manhattan distance between the King and his palace.

ii. The Manhattan distance between the King and his palace divided by 4.

iii. The Manhattan distance between the King and the his horse.

iv.

v. The Manhattan distance between (A) the closest horse stop to the King, and (B) his palace.

vi. None of the options are admissible.

(b) Suppose the King knows that there is at least one horse stop adjacent to his palace. **Determine if each of the following heuristics is admissible.**

If you feel that none are admissible, select only the option "None of the options are admissible". For each option, briefly, but clearly explain, why it is admissible or inadmissible.

Answers without rationale will not be awarded any marks.

i. The Manhattan distance between the King and the horse.

ii. The Manhattan distance between the horse and the stop next to the palace divided by 3.

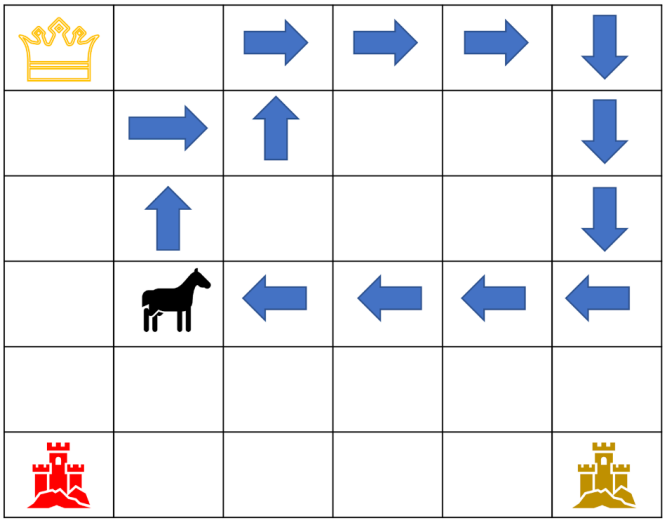
iii. $h = 1/3$ for all states.

iv. The minimum of the above three heuristics.

v. None of the options are admissible.

(c) Now, suppose that the King wishes to also visit the queen's palace, located in the bottom-most and left-most square of the grid (i.e., the red palace in the example figure below). Assume that it does not matter if he visits his own palace (i.e., the yellow palace) or the queen's place first. However, he must visit both locations.

Below is an example start state with the queen's palace in the bottom-most and left-most square, and the king's palace in the bottom-most and right-most square.



Design an admissible heuristic for this problem.

Your heuristic may not be $h(s) = 0$ for all states s, the (abstract) optimal heuristic, or a linear combination/simple function thereof. You may assume that this problem is solvable – i.e. there is some path to a goal state.

You must prove that your heuristic is admissible.

Solution:

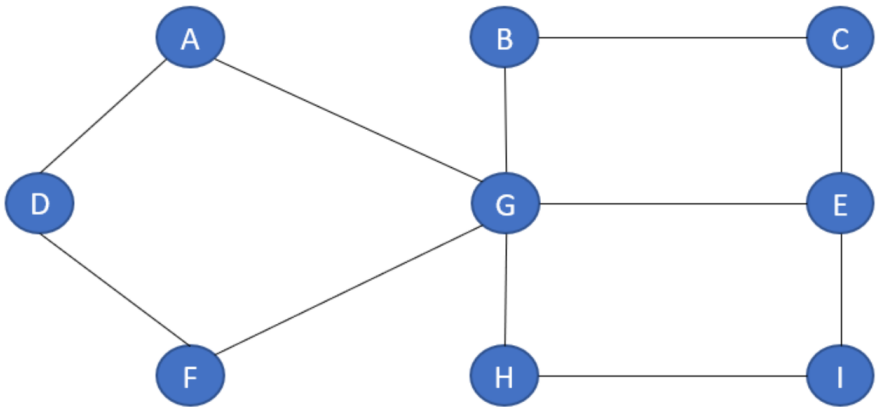
(a)
i. Not admissible: The King can reach his palace in less timesteps than the actual number of spaces between if he takes the horse.
ii. Admissible: At best, the King would be able to reach his palace by moving 3 spaces per timestep, assuming he rides the horse the whole way. Therefore, this heuristic is an underestimate of the true cost of reaching his palace.
iii. Not admissible: Consider the case where the King is already next to his palace; the best path does not necessarily involve the King's distance to the horse.
iv. Not admissible: Consider the case where the King is already next to his palace; the best path does not necessarily involve the King's distance to any horse stop.

(b)
i. Not admissible: The horse could be far away from the King, while the King could be right next to the palace.
ii. Not admissible: Same reason as (i); the horse travels in a fixed route, and its distance from his palace does not represent how close the King is to achieving his goal.
iii. Not admissible: The heuristic's value at the goal state must be 0 for admissibility, even though we underestimate everywhere else.
iv. Not admissible: The minimum would be admissible if one of the heuristics achieved 0 at the goal state, because the third heuristic underestimates at all states except the goal state; however, since none of them do, the minimum is still not admissible.

(c)
- Heuristic: The minimum of the Manhattan distances between the current position and the queen's palace, the queen's palace and the king's palace, and the current position and the king's palace, divided by 3.
- Explanation: From any space, the King is heading to either the queen's palace or his palace, so the minimal Manhattan distance between him and either (divided by 3 assuming he could take the horse the whole way) is less than or equal to the timesteps necessary to reach a goal state.

4. **Vertex Cover Problem.** In graph theory, a vertex cover (sometimes called node cover) of a graph is a set of vertices that includes at least one endpoint of every edge of the graph. A minimum vertex cover is a vertex cover having the smallest possible number of vertices for a given graph.

   Consider an instance of the vertex cover problem given in the Figure below.



   In the vertex cover problem we are given a graph $G = \langle V, E \rangle$. We say that a vertex v covers an edge $e \in E$ if v is incident on the edge e. We are interested in finding a vertex cover; this is a set of vertices $V' \subseteq V$ such that every edge is covered by some vertex in V'.

   (a) You are provided with a cost function associated with every state, given by:

   $$f(state) = \text{number of vertices in the current vertex cover}$$

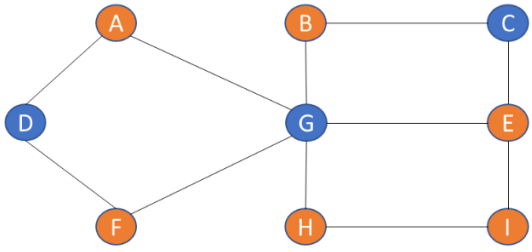
   You will take a step if and only if the cost of the next state is less than or equal to the cost of the current state. If there are different possible actions, you will always choose the action that leads to the state with the lowest cost. (If there are ties, you may decide any of the tied actions to take.)

   By using the hill-climbing (steepest descent) algorithm, describe the sequence of steps that would help you arrive to the global optima (minimum vertex cover). You are to assume that the initial state is the graph shown above.

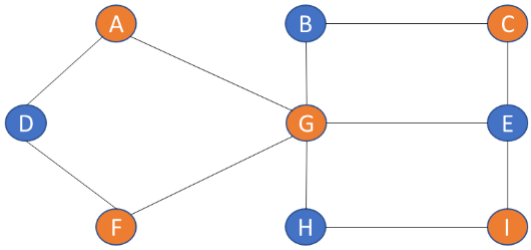
   (b) **State the nodes left in the minimum vertex cover.** You should express your answer in the form X–Y–Z (i.e., no spaces, all uppercase letters, delimited by the dash (–) character).

(c) The figures below are different states of a vertex cover of the graph, with the nodes coloured in orange being in the vertex cover, whilst the nodes in blue have been removed. **Identify the state(s) that correspond to a local optimum.**

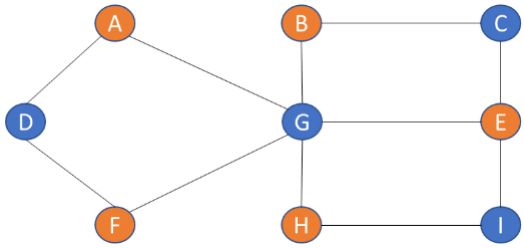
i.



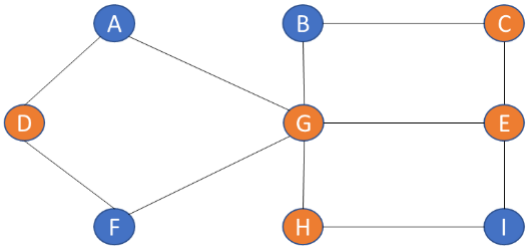
ii.



iii.



iv.

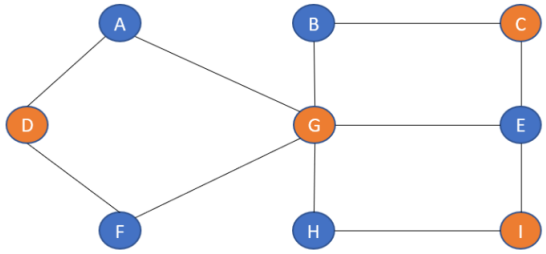


Solution:

(a) The local search trace is as follows.

Node Removed	f-score
-	9
A	8
F	7
B	6
H	5
E	4

(b) Remaining nodes: C, D, G, I



(c) Options (ii), (iii), and (iv).

5. Provide a proof that shows the A^* algorithm using an admissible heuristic h_1 will expand fewer or at most the same number of nodes as A^* using h_2, where $\forall n, h_2(n) = 0$. You may assume that the A^* algorithm is implemented using tree search in both cases, and that that $h_1 \neq h_2$.

Solution:

- We know, and thus assert, the following:
  (a) The A^* algorithm is optimal under tree search using an admissible heuristic.
  (b) All paths have increasing costs as they grow longer (i.e., traverse additional states).
- Let the optimal path be s_0, s_1, ..., s_k (where s_k is a goal node).
- Since h_1 is admissible, $\forall n, h_2(n) = 0$, and $\exists n, h_1(n) \neq h_2(n)$, we may assume that h_1 dominates h_2 - i.e., $\forall n, h_1(n) \geq h_2(n)$.
- Thus, we have:
  (c) $f_{h_2}(n) = h_2(n) + g(n) \leq f_{h_1}(n) = h_1(n) + g(n) \leq h^*(n) + g(n)$ – assert the admissibility of h_1 and h_2, and dominance of h_1 over h_2.
  (d) $f_{h_2}(s_k) = h_2(s_k) + g(s_k) = f_{h_1}(s_k) = h_1(s_k) + g(s_k) = h^*(s_k) + g(s_k) = g(s_k)$ – assert that s_k is a goal node - i.e., $h(s_k) = 0$.
  (e) All paths to n such that $f(n) < f(s_k) = g(s_k)$ will be expanded – assert A^* algorithm. And since both heuristics in question are admissible, no paths with path costs greater than $f(s_k) = g(s_k)$ will be expanded.
- Let N_{h_1} denote the set of nodes such that each $n_i \in N_{h_1}, f_{h_1}(n_i) \leq g(s_k)$ - i.e., N_{h_1} corresponds to all the nodes that will be expanded by A^* using h_1. This given (e).
- Let N_{h_2} denote the set of nodes such that each $n_i \in N_{h_2}, f_{h_2}(n_i) \leq g(s_k)$ - i.e., N_{h_2} corresponds to all the nodes that will be expanded by A^* using h_2. This given (e).
- We have: $N_{h_1} \setminus N_{h_2} = \{n_i | f_{h_1}(n_i) \leq g(s_k) < f_{h_2}(n_i)\}$.
- And, also have: $N_{h_2} \setminus N_{h_1} = \{n_i | f_{h_2}(n_i) \leq g(s_k) < f_{h_1}(n_i)\}$.
- However, we know that h_1 dominates h_2 since h_1 is admissible, but $\forall n, h_2(n) = 0$ and $h_1 \neq h_2$.
- This implies that it is possible that $\exists n_i, g(n_i) + h_2(n_i) \leq g(s_k) < g(n_i) + h_1(n_i)$, but it is impossible that $\exists n_i, g(n_i) + h_1(n_i) \leq g(s_k) < g(n_i) + h_2(n_i)$, since $\forall n, h_1(n_i) \geq h_2(n_i)$.
- This in turn implies that $\exists n_i, f_{h_2}(n_i) \leq g(s_k) < f_{h_1}(n_i)$ is possible, but $\exists n_i, f_{h_1}(n_i) \leq g(s_k) < f_{h_2}(n_i)$ is not.
- Consequently, we have $N_{h_1} \setminus N_{h_2} = \{n_i | f_{h_1}(n_i) > f_{h_2}(n_i) \leq g(s_k)\} = \emptyset$, but $N_{h_2} \setminus N_{h_1} = \{n_i | f_{h_2}(n_i) > f_{h_1}(n_i) \leq g(s_k)\} \neq \emptyset$

- Thus, it is possible that $|N_{h_2}| > |N_{h_1}|$, but impossible that $|N_{h_1}| > |N_{h_2}|$.
- This means that the effective branching factor of A^* using h_1 must be less than the effective branching factor of A^* using h_2 since effective branching factor is calculated by solving $b^0 + b^1 + ... + b^d = |N|$.