Ethics Warmup

Take a look at this description of RedZone, a navigation app that allows you to avoid high crime regions: https://www.androidauthority.com/redzone-navigation-app-avoids-high-crime-areas-686894/

1. If this app became commonly used, what impact could it have on businesses in the "red zone"?

   Businesses in zones the application considers to have high crime could lose customers due to decreased thoroughfare in their neighborhoods.

2. Yes or No - would you want this application to roll out to your hometown?

   There isn't one right answer to this question, but a positive impact of rolling the application out in your hometown could be:

   - Encouraging greater tourism/traffic to your hometown from people outside your area due to an increase in perceived safety, an economic benefit

   Some negative impacts could be:

   - Loss of business in "high crime areas", an economic harm
   - Creating/exacerbating income disparities between areas classified by the application as low crime and high crime, where businesses in high crime areas are more likely owned by and serve people of color.

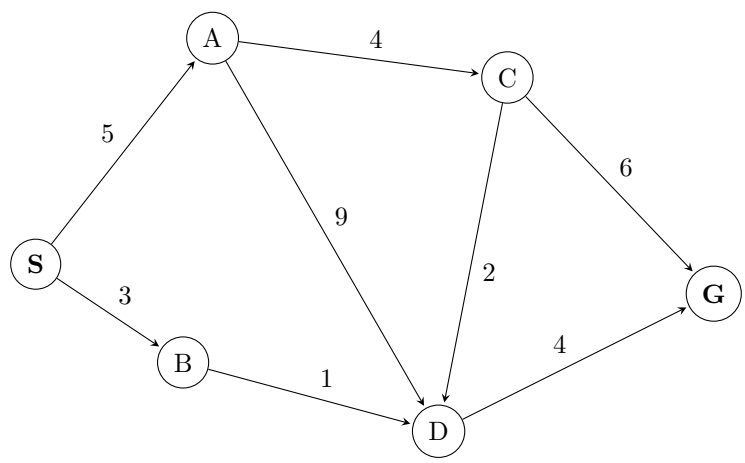
   There are many possible considerations, these are just provided as examples.

1 Designing & Understanding Heuristics

Today, we will be taking a closer look at how the performance of A^* is affected by the heuristics it uses. To do this, we'll be using the graph below. You may have noticed that no heuristic values have been provided (*Recall:* What is A^* without heuristic values?). This is because we'll be working together to come up with heuristics ourselves!

In groups, design both an admissible heuristic and a consistent heuristic for the graph below by annotating each node with a heuristic value. (Note: you do NOT need to find a closed form way to represent the heuristic function.)

When you have completed your heuristics, work together to answer the questions below.



(a) Write down the path found by running A^* using your heuristic on the graph above.

This will depend on the provided heuristic. Recall that A^* expands the node in its frontier that has the lowest $f(n) = g(n) + h(n)$ value. Some common examples are the zero heuristic or the exact heuristic.

(b) Work with your group to come up with a heuristic that's admissible but not consistent.

Heuristics will vary from team to team, and there may be many correct solutions. Feel free to come to OH or post on Diderot if you're unsure about a particular answer. A general strategy is to make the heuristic value "more" or "less" accurate on different nodes in a path to the goal, e.g. making one value the optimal heuristic value and the next on the path 0 (why does this work?).

(c) (Bonus) Explain why a consistent heuristic must also be admissible. You may assume that the heuristic value at a goal node is always 0.

Recall the definitions of admissibility and consistency.

A heuristic is admissible if it never overestimates the true cost to a nearest goal.
A heuristic is consistent if, when going from neighboring nodes a to b, the heuristic difference/step cost never overestimates the actual step cost. This can also be re-expressed as the triangle inequality mentioned in Lecture 3.

Let $h(n)$ be the heuristic value at node n and $c(n, n+1)$ be the cost from node n to $n+1$. We're given the assumption that $h(n_g) = 0$, where n_g is a goal node. Informally, we want to backtrack from the

goal node, showing that if we start with an admissible node (which $h(n_g) = 0$ is by default), its parent nodes will be admissible through the rule of consistency, and this pattern continues.

Consider some node n and assume the heuristic value at n is admissible. We want to show that any of its parent nodes, say $n-1$, will also be admissible by the definition of consistency.

By consistency, we get that:

$$h(n-1) - h(n) \leq c(n-1, n)$$
$$= h(n-1) \leq c(n-1, n) + h(n)$$

$c(n-1, n)$ is the actual path cost from $n-1$ to n. Since we know $h(n)$ is admissible, we know that $h(n)$ has to be less than or equal to the path cost from n to goal node n_g.

Thus, $h(n-1) \leq$ the path cost from ($n-1$ to n) $+$ $h(n)$.
Since $h(n)$ is less than or equal to path cost from (n to n_g),
$h(n-1) \leq$ path cost from ($n-1$ to n_g).

This by definition is admissible.

Formal proof (for students interested):
Assume we have some consistent heuristic h. Also assume $h(n_g) = 0$, where n_g is a goal node. By definition of consistency, $h(n) \leq c(n, n+1) + h(n+1)$ for all nodes n in the graph. We want to show that for all n, $h(n) \leq h^*(n)$ (the definition of admissibility) also holds.

Base Case: We begin by considering the $n_g - 1$th node in any path where n_g denotes the goal state.

$$h(n_g - 1) \leq c(n_g - 1, n_g) + h(n_g) \tag{1}$$

Because n_g is the goal state, by assumption, $h(n_g) = h^*(n_g)$. Therefore, we can rewrite the above as

$$h(n_g - 1) \leq c(n_g - 1, n_g) + h^*(n_g)$$

and given that $c(n_g - 1, n_g) + h^*(n_g) = h^*(n_g - 1)$, we can see:

$$h(n_g - 1) \leq h^*(n_g - 1)$$

as desired.

Inductive Hypothesis: Assume that for some arbitrary node (which lies on a path from start to goal) $n_g - k$ that $h(n_g - k) \leq h^*(n_g - k)$.

Inductive Step: To see if this is always the case, we consider the $n_g - k - 1$th node in any of the paths we considered above (e.g. where there is precisely one node between it and the goal state). The cost to get from this node to the goal state can be written as

$$h(n_g - k - 1) \leq c(n_g - k - 1, n_g - k) + h(n_g - k)$$

From our base case above, we know that

$$h(n_g - k - 1) \leq c(n_g - k - 1, n_g - k) + h(n_g - k) \leq c(n_g - k - 1, n_g - k) + h^*(n_g - k)$$
$$h(n_g - k - 1) \leq c(n_g - k - 1, n_g - k) + h^*(n_g - k)$$

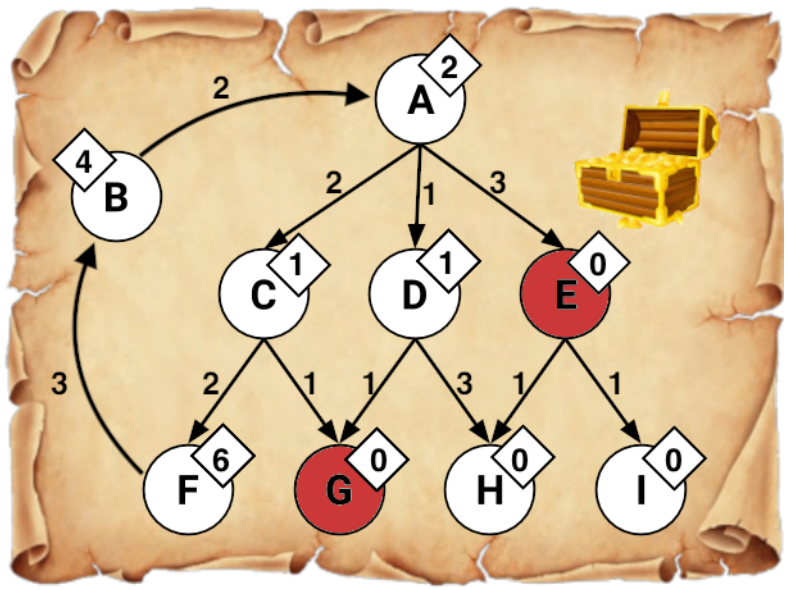
And again, we know that $c(n_g - k - 1, n_g - k) + h^*(n_g - k) = h^*(n_g - k - 1)$, so we can see:

$$h(n_g - k - 1) \leq h^*(n_g - k - 1)$$

By the inductive hypothesis, this holds for all nodes, proving that consistency does imply admissibility!

2 Treasure Hunting

We are lost at sea, trying to find a hidden treasure. We have a treasure map that tells us which paths we can take and approximately how far away the treasure is but it's not very accurate. We do know that the map will never overestimate the distance to the treasure. There is treasure on two different islands (but we only need to reach one of them).



A is the the island where we are currently and the shaded red states are the locations of the treasure. Arrows encode possible actions from each island, and numbers by the arrows represent action costs. Note that arrows are directed; for example, $A \to C$ is a valid action, but $C \to A$ is not. Numbers shown in diamonds are heuristic values that estimate the optimal (minimal) cost to get from that island to any treasure.

Run each of the following search algorithms with graph search and write down the nodes that are added to the explored set during the course of the search, as well as the final path returned and the corresponding cost of the final path, if applicable. When popping off of the frontier, assume that ties are broken alphabetically.

(a) **Depth-First Search**

Explored set:

A, C, F, B

Path returned:

A, C, G

(b) **Breadth-First Search**

Explored set:

A, C, D

Path returned:

A, E

(c) **Uniform-Cost Search**

Explored set:

A, D, C

Path returned and cost:

A, D, G with a cost of 2

(d) **Greedy Search**

Explored set:

A

Path returned and cost:

A, E with a cost of 3

(e) **A* Search**

Explored set:

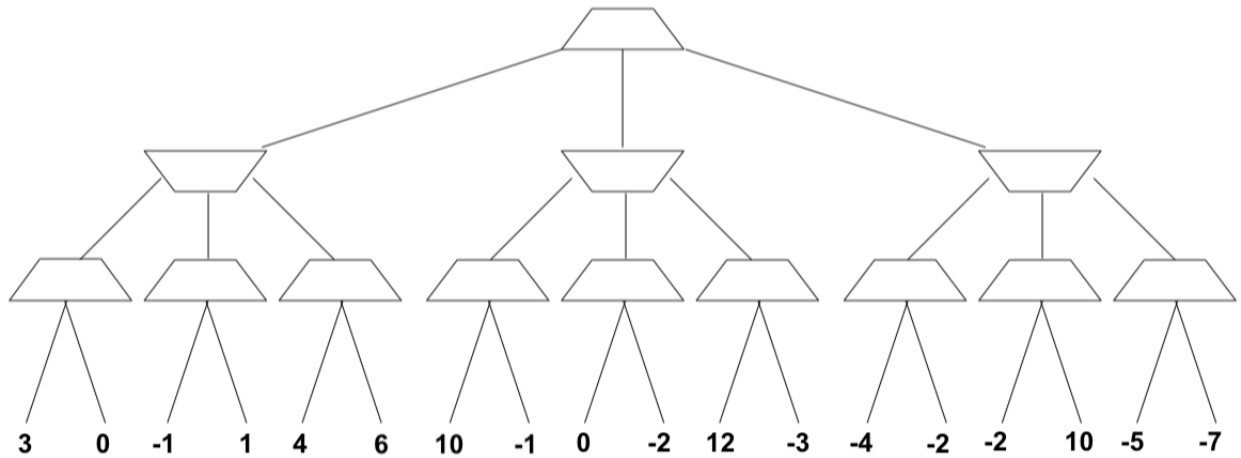
A, D

Path returned and cost:

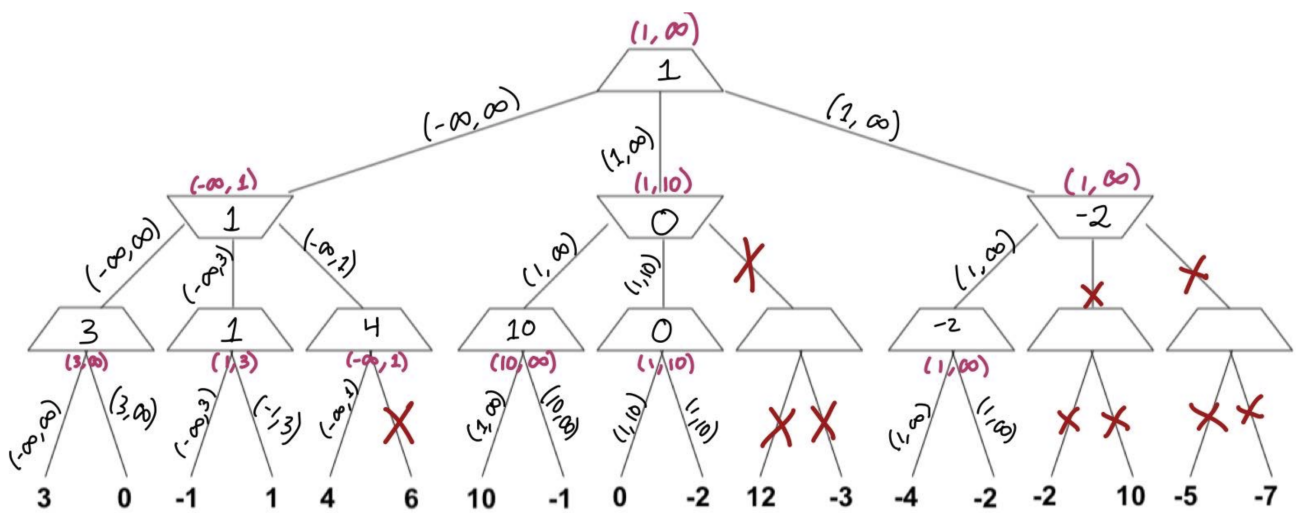
A, D, G with a cost of 2

3 Adversarial Search

Consider the following game tree, where the root node is a maximizer. Using alpha beta pruning and visiting successors from left to right, record the values of alpha and beta at each node. Furthermore, write the value being returned at each node inside the trapezoid. Put an 'X' through the edges that are pruned off.



The following picture shows the final (α, β) values at each node.



4 True/False Section

For each of the following questions, answer true or false and provide a brief explanation (or counterexample, if applicable).

(a) Depth-first search always expands at least as many nodes as A^* search with an admissible heuristic.

False: If the minimum goal path consists of d nodes, a lucky DFS might expand exactly those d nodes to reach the goal. A^* could potentially expand some other nodes before finding that path (those nodes would have a lower $f(n)$ value than the final goal's).

(b) Assume that for a single move, a rook can move any number of squares on a chessboard in a straight line, either vertically or horizontally, but cannot jump over other pieces. Manhattan distance is an admissible heuristic for the smallest number of moves to move the rook from square A to square B.

False: A rook can move from one corner to the opposite corner across a 4x4 board in two moves, although the Manhattan distance from start to finish is 6.

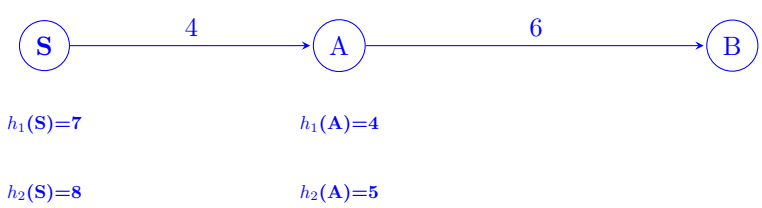
(c) Euclidean distance is an admissible heuristic for Pacman path-planning problems.

True: Euclidean distance is the minimum distance to travel between any two points. Thus, it will always be less than or equal to Pacman's (who only moves horizontally or vertically) actual cost to travel along some path, making it admissible ($0 \leq h(n) \leq h^*(n)$).

More specifically, if the goal is either vertical or horizontal to Pacman's current position, then the Euclidean distance will be the true distance. Otherwise, if the goal is diagonally away from Pacman's current position, then the Euclidean distance will be less than the true distance. In both scenarios, Euclidean distance is a lower bound on the true distance.

(d) The sum of several admissible heuristics is still an admissible heuristic.

False:

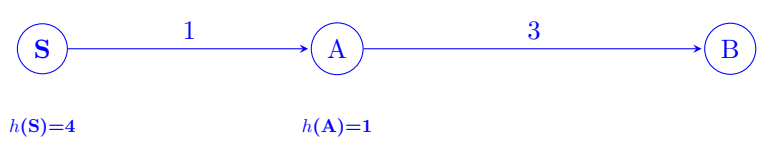


Both of these heuristics (h_1 and h_2) are admissible, but if we sum them, we find that $h_3(S) = 15$ and $h_3(A) = 9$. This is not admissible.

However, notice that taking the maximum of two admissible heuristics will result in an admissible heuristic.

(e) Admissibility of a heuristic for A^* search implies consistency as well.

False:

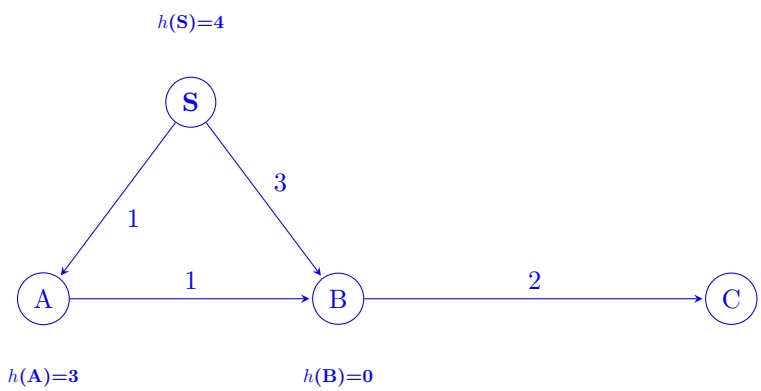


Heuristic h is admissible since it never overestimates the true cost to the goal. However, it is not consistent since the step cost (between states S and A) is overestimated by the heuristics. The estimated step cost is $h(S) - h(A) = 4 - 1 = 3$, while the true step cost between S and A is 1.

Another way of looking at it is $h(S) > cost(S, A) + h(A)$, which violates the triangle inequality.

(f) A^* with graph search is always optimal with an admissible heuristic.

False:



The optimal path from S to C is $S - A - B - C$ with a total cost of 4. However, with A* graph search:

- After starting from S, the options on the frontier are $S - A$ with a cost plus heuristic of 1+3=4, or $S - B$ with a cost plus heuristic of 3+0=3. So, we choose to explore $S - B$.
- Next, the options on the frontier are $S - A$ with a cost plus heuristic of 1+3=4, or $B - C$ with a backward path cost of 5. So, we choose to explore $S - A$.
- We've already reached B from the earlier path, so we don't consider edge $A - B$. Thus, our only option is to explore $B - C$ for a cost of 5.
- So, we will choose the path $S - B - C$ with a total cost of 5, which is not optimal.

Note that there is a distinction here between tree search and graph search: A^* with <u>tree search</u> is guaranteed to be optimal if the heuristic is admissible, but A^* with <u>graph search</u> is only guaranteed to be optimal if the heuristic is consistent.

For (g) and (h), consider an adversarial game tree where the root node is a maximizer, and the minimax value of the game (i.e., the value of the root node after running minimax search on the game tree) is V_M. Now, also consider an otherwise identical tree where every minimizer node is replaced with a chance node (with an arbitrary but known probability distribution). The expectimax value of the modified game tree is V_E.

(g) V_M is guaranteed to be less than or equal to V_E.

True: The maximizer is guaranteed to be able to achieve v_M if the minimizer acts optimally. They can potentially do better if the minimizer acts suboptimally (e.g. by acting randomly). The expectation at chance nodes can be no less than the minimum of the nodes' successors.

(h) Using the optimal minimax policy in the game corresponding to the modified (chance) game tree is guaranteed to result in a payoff of at least V_E.

False: In order to achieve v_E in the modified (chance) game, the maximizer may need to change their strategy. The minimax strategy may avoid actions where the minimum value is less than the expectation. Moreover, even if the maximizer followed the expectimax strategy, they are only guaranteed v_E *in expectation*.