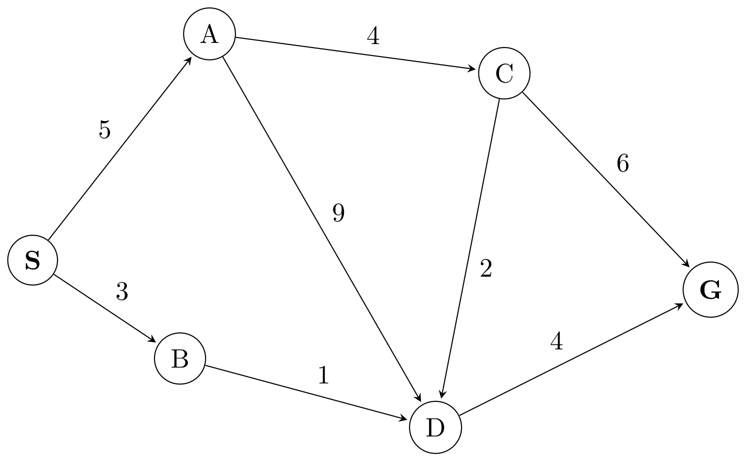
1 Designing & Understanding Heuristics

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 in pairs to come up with heuristics ourselves!

Please find someone next to you to work with, and decide between yourselves who will design an admissible heuristic and who will design a consistent heuristic. Then, *independently* create these heuristics for the given graph by annotating each node with a heuristic value.

When you have completed your heuristic, exchange papers with your partner, and 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.

(b) Work with your partner 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 Piazza if you're unsure about a particular answer.

(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 $h(g) = 0$. Informally, we want to backtrack from the goal node, showing that if we start with an admissible node (which $h(g) = 0$ 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 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 g),
$h(n-1) \leq$ path cost from ($n-1$ to g).

This by definition is admissible.

Formal proof (for students interested):
Assume we have some consistent heuristic h. Also assume $h(g) = 0$, where 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 $g-1$th node in any path where g denotes the goal state.

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

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

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

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

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

as desired.

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

Inductive Step: To see if this is always the case, we consider the $g-k-1$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

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

From our base case above, we know that

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

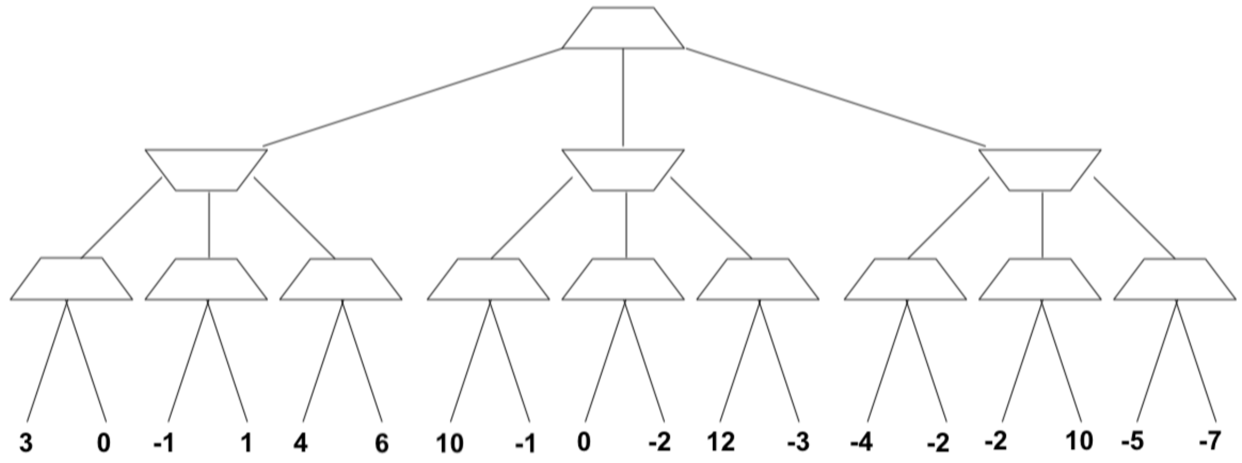
And again, we know that $c(g-k-1, g-k) + h^*(g-k) = h^*(g-k-1)$, so we can see:

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

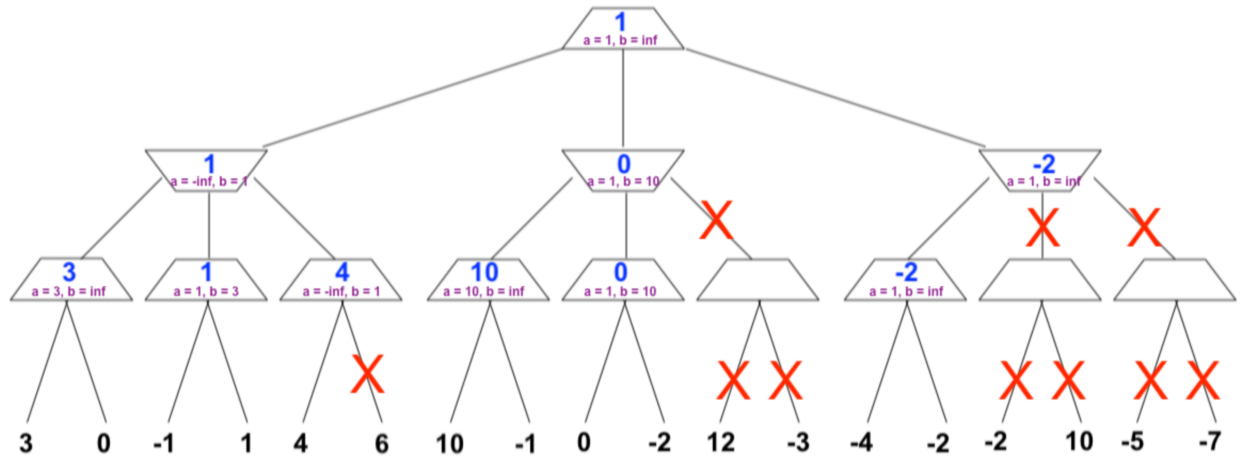
By the inductive hypothesis, this holds for all nodes, proving that consistency does imply admissibility!

2 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.



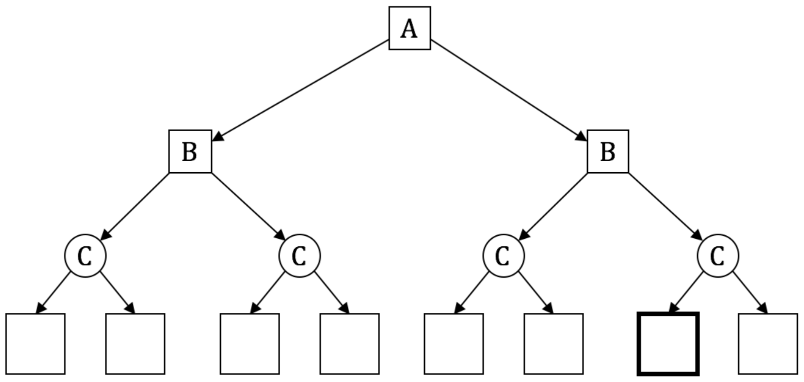
The following picture shows the final α and β (denoted a and b) values at each node.

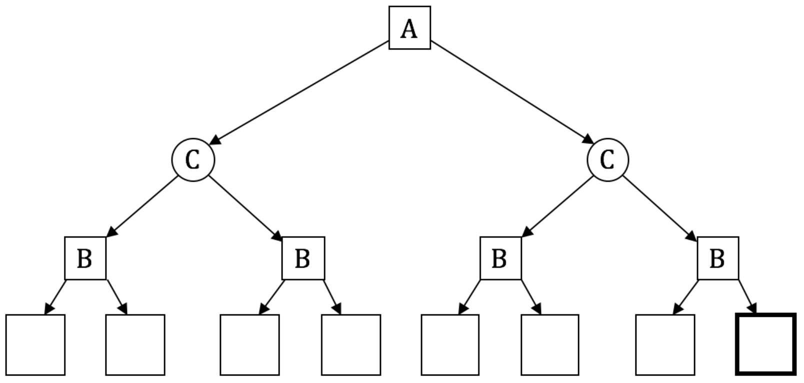


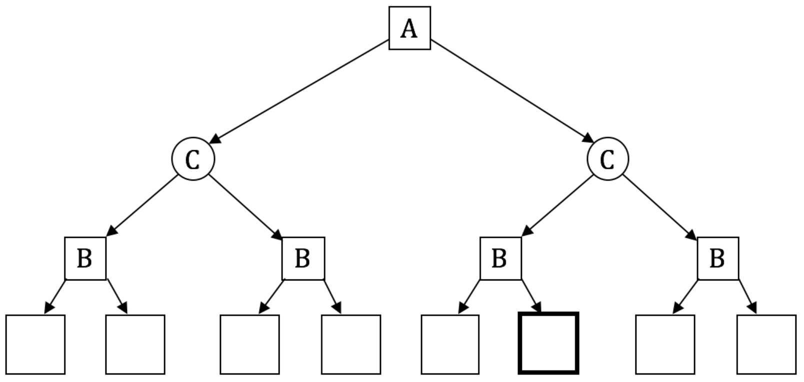
3 Alpha Beta Expectimax

In this question, player A is a minimizer, player B is a maximizer, and C represents a chance node. All children of a chance node are equally likely. Consider a game tree with players A, B, and C. In lecture, we considered how to prune a minimax game tree - in this question, you will consider how to prune an expectimax game tree (like a minimax game tree but with chance nodes). Assume that the children of a node are visited left to right.

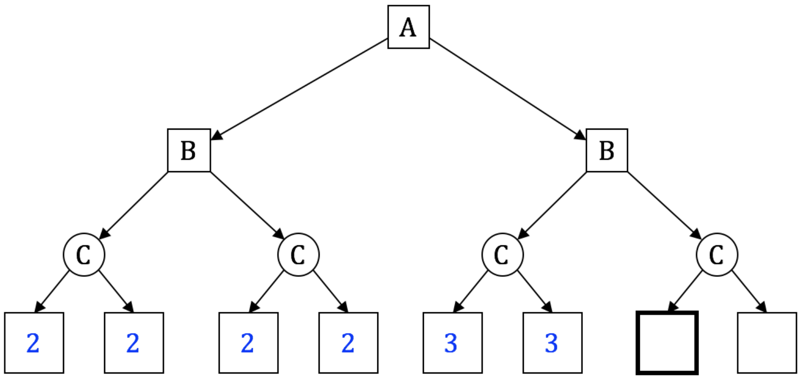
For each of the following game trees, give an assignment of terminal values to the leaf nodes such that the bolded node can be pruned (it doesn't matter if you prune more nodes), or write "not possible" if no such assignment exists. You may give an assignment where an ancestor of the bolded node is pruned (since then the bolded node will never be visited). You should not prune on equality, and your terminal values must be finite.

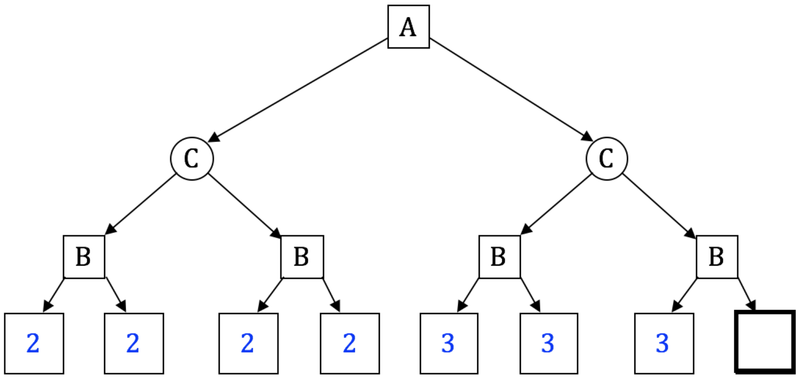


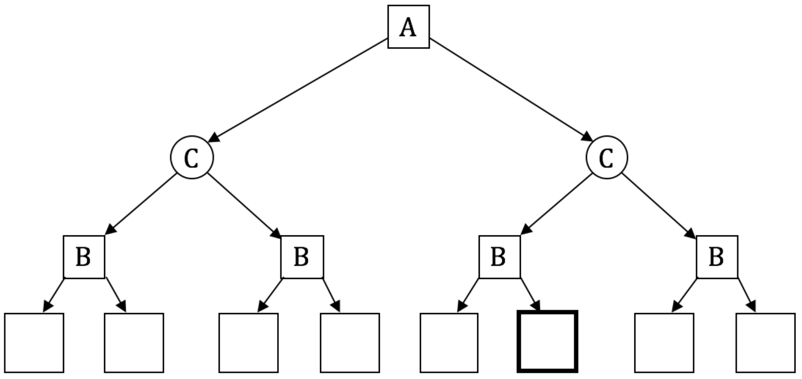




Answers may vary.







Not possible. At the bolded node, the minimizer can guarantee a score of at most the value of its left subtree. However, since we have not visited all the children of C, there is no bound on the value that C could attain. So, we need to continue exploring nodes until we can put a bound on C's value, which means we must explore the bolded 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 a rook can move on a chessboard any number of squares in a straight line, vertically or horizontally, but cannot jump over other pieces. Manhattan distance is an admissible heuristic for the problem of moving the rook from square A to square B in the smallest number of moves.

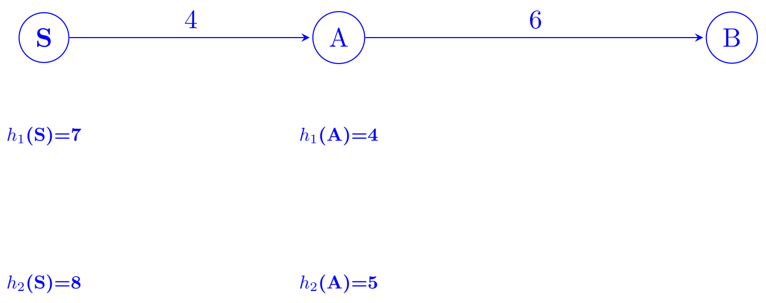
False: A rook can move from one corner to the opposite corner across a 4x4 board in two moves, although the Manhattan distance from start to finish is 6.

(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 $(0 \leq h(n) \leq h^*(n))$.

(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 $h3(s) = 15$ and $h3(a) = 9$. However, this is not admissible.

For (e) and (f), 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$_E$.

(e) 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.

(f) 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*.