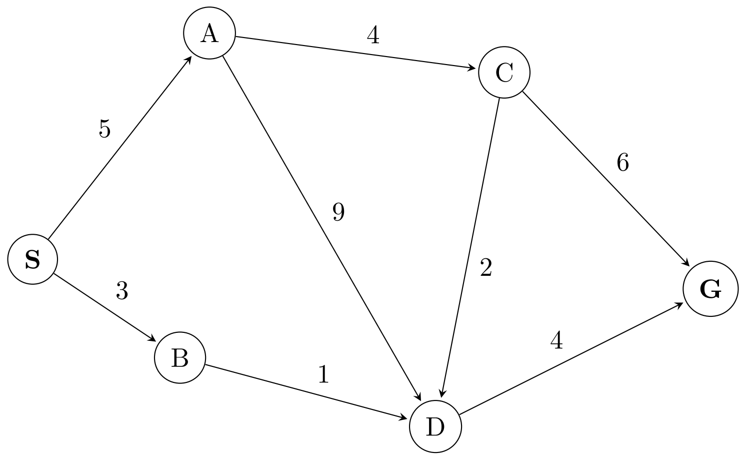
1 Designing & Understanding Heuristics

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.



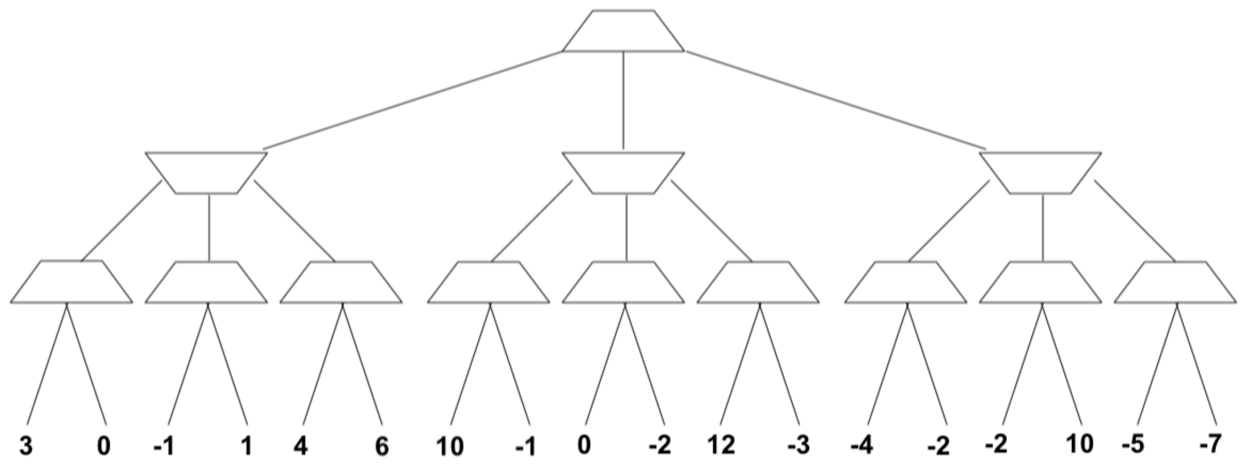
(a) Write down the path found by running A^* using your heuristic on the graph above.

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

(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.

2 Adversarial Search

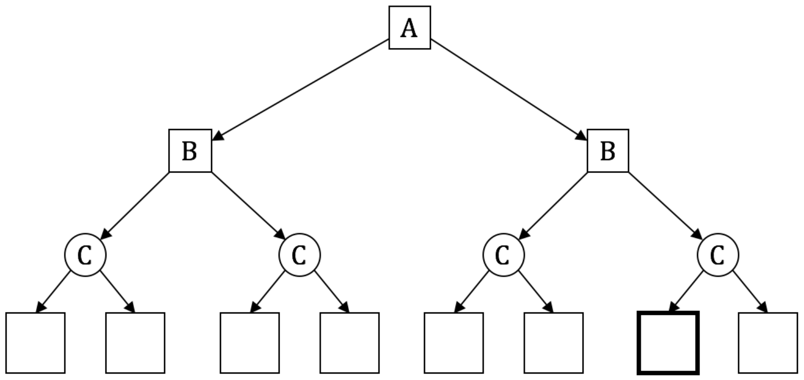
Consider the following game tree, where the root node is a maximizer. Using alpha beta pruning and visiting successors from left to right, record the values of alpha and beta at each node. Furthermore, write the value being returned at each node inside the trapezoid. Put an 'X' through the edges that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

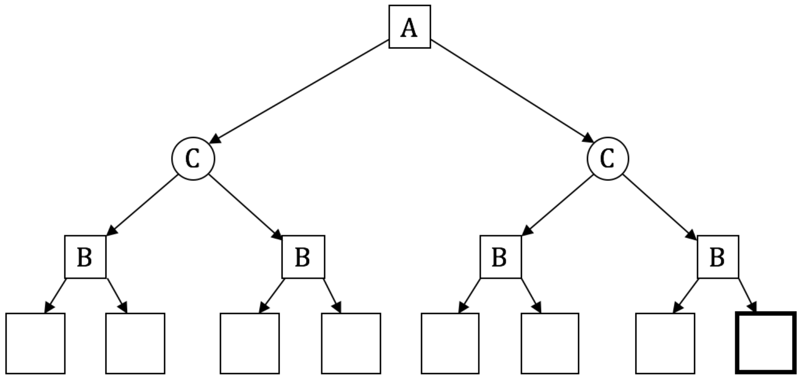


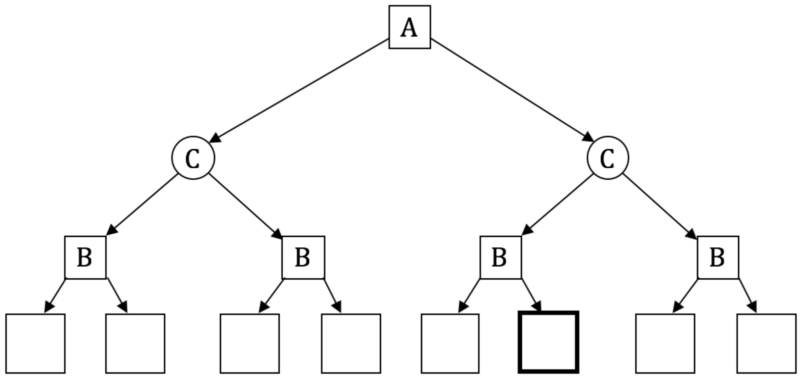
3 Alpha Beta Expectimax

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.







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.

(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.

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

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

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.

(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.