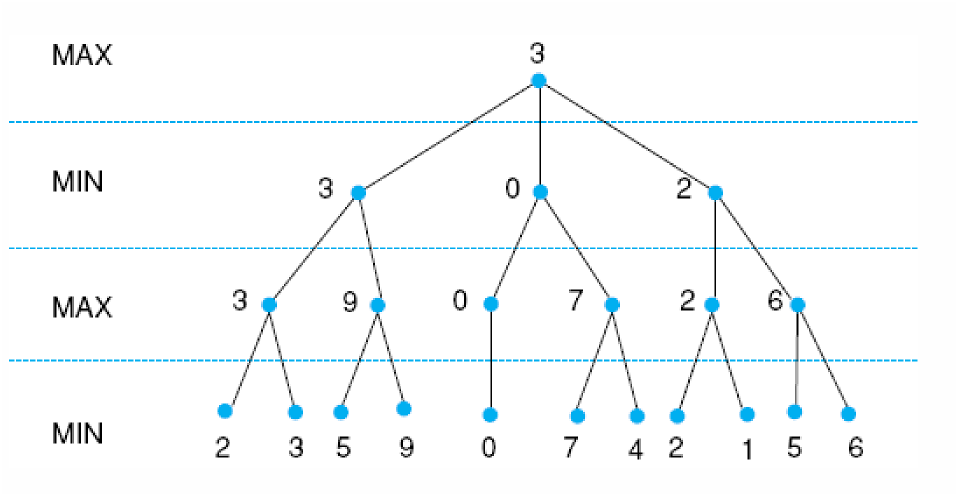
CSC 425/525: Extra Notes on Chapter 4

Here we examine the Minimax search and a variation called Alpha-Beta Pruning.

The Minimax heuristic strategy is associated with game playing. The idea in a normal search strategy is that you want to select the state that the heuristic tells you will most likely lead you to a solution. But what about in game playing against an opponent? When you make a move, you present your opponent with new move options. If, in selecting the best move for yourself, are you also leaving the board in a situation where your opponent has good moves? The Minimax strategy is based on the idea that while you want to maximize the heuristic value of the move you select, you also want to minimize the heuristic worth of the options that your opponent will get.

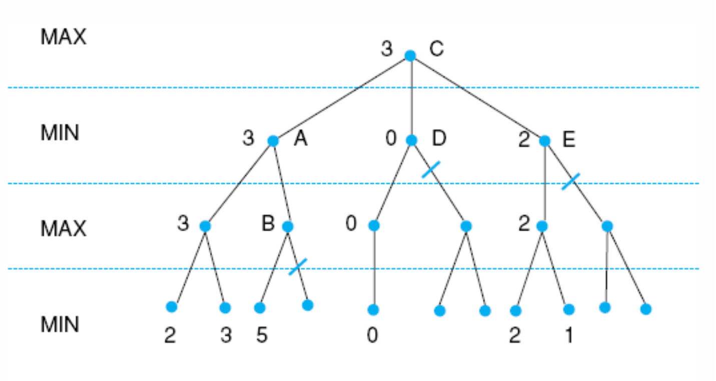
Consider the following game tree. You are looking 3 moves ahead. The tree shows you the heuristic worth of each of those moves. The best move would seem to select the first branch so that you could then select the right branch and then the right branch. Such a move will lead you to a state of 9. But if you select the left branch, would your opponent then select the right branch? If your opponent instead selected the left branch, you would be stuck with choice that yield heuristic values of only 2 or 3. Thus, the left branch may not be the best choice.



The minimax strategy is the same as a depth first search in that you look ahead many levels. However, unlike the brute force search of chapter 3, here you apply your heuristic to the states at the lowest level. Then, you apply minimax. This means that you group together the siblings and select the node that has the maximum worth. At the next level up, you select the minimum value assuming that your opponent will try to limit your best choice. At the next level up, you again select the maximum value because that would be your turn. And so on up the tree. This is done recursively. In the figure above, since you are looking 3 levels ahead and it is your turn, you would make two moves and your opponent one. So the levels will be to choose the "maximum of the minimum of the maximum". We start at the lowest level and select the max values. From 2 & 3, we would select 3. From 5 & 9, we would select 9. The next choice is only 0. From 7 & 4, we would select 7. From 2 & 1, we would select 2. From 5 & 6 we would choose 6. So you will see in the tree that at the level above the leaves (that is, the lowest level in terms of how far we looked ahead), we have max's of 3, 9, 0, 7, 2, 6. Now we select the minimum values assuming our opponent will leave us with the worst possible choices. Those are 3, 0, 2 respectively. Now, we would make our choice based on the maximums. So, while our best possible choice is 9, if the opponent is clever, in fact our best choice would be 3 (coincidentally down the same branch). NOTE: we make the assumption that our opponent is using the same heuristic that we are or else they may evaluate the leaf nodes in our search space differently.

The number of levels that we look ahead is known as the plies. This value indicates not only the depth from which we will look ahead, but also the complexity of the algorithm. If we look ahead with a ply of 3 (as in the figure above) and on average, each node has 4 children, then for each move, we are looking ahead 4^3, that is, at each of the levels, we have 4 choices. We would apply this search for every turn. If our game runs n turns, then the complexity is $n*4^3$. This is a significant improvement over n^m as we saw in chapter 3. Why is it better? Because we are eliminating great chunks of space with every move (if the average game say were to take k moves, and we are looking only 3 moves ahead at a time, then we are not looking at the depth of k, only a depth of 3). Unfortunately, by limiting our search to *plies* moves ahead, we also can make mistakes. We are eliminating potentially good moves. For instance, return to the tree above. Imagine that the middle branch, which had choices of 0 and 7, were extended two more levels. The heuristic worth of the children at that level were instead 10, 9, 8, 7, 6, 5. Then, the worst we would get by going down that branch would be a 5, an improvement over 3. This idea of perhaps missing a better choice is known as the *horizon effect*.

A variation of minimax is known as alpha-beta pruning. The idea is to prune away paths that appear futile because we have already found a minimum or maximum value down another branch. As an example, we return to the previous tree, but this time edited to show the prunings. Here, since we already know that the left branch yields a 3 and that at the middle level, the opponent will select the worst state for us, if we know by descending another level that we could obtain at least a 5, it is not worth exploring any further because our opponent would never choose that path. So, if we choose A because we are guaranteed at least a 3, our opponent would look and see "3" down the left branch, and "at least a 5" down the right branch, and would of course pick the left branch. So there is no point in exploring B's children any further once we know there is a 5 available. A Beta value is one which means that "there's no point searching further as we will not be given the choice to find a larger value" whereas an Alpha value means that "there's no point searching further because we already found a value worse than our current best choice". For instance, since the branch down E would give us an option of 2, we don't have to search E's right branch because anything greater than 2 would not be selected.



A has $\beta = 3$ (A will be no larger than 3)
B is β pruned, since $5 > 3$
C has $\alpha = 3$ (C will be no smaller than 3)
D is α pruned, since $0 < 3$
E is α pruned, since $2 < 3$
C is 3

What makes a useful heuristic? We saw 3 heuristics for the 8 puzzle, counting the number of out of place tiles, summing the distances of out of place tiles, and computing 2 * number of direct tile reversals needed. The first heuristic might be thought of as a “local heuristic” in that it tells us immediately about this state, but it does not take into account the idea that you might have to mess up a part of the puzzle to complete the entire thing. For instance, if you already had the first row in place entirely, you would most likely have to move some of those tiles to get the second and third rows in place. However, the first heuristic discourages you from moving correct tiles. The second heuristic is more of a “global heuristic” in that it takes into account what it might take to get the puzzle in its correct form. However, it too will discourage moving correct tiles if possible. A superior heuristic might reward you instead for correct blocks. For instance, if we have a state that gets us to having 1-2-4-5 for the upper left 2x2 portion, it would be better than a state that does not contain that block.

What about other types of games? What kind of heuristic might you have for say checkers? You might, as with the 8 puzzle, have a heuristic that merely sums up your pieces versus your opponent. This would encourage you to capture your opponent’s pieces, but it does not necessarily discourage moves that might allow your opponent to capture your pieces. The problem here is that you might make a move that captures 1 piece from your opponent only to have your opponent then capture 2 of your pieces in a single move. A heuristic for checkers should take into account how many pieces you can capture versus your opponent, and the positioning of those pieces. Another factor would be to encourage you obtaining king pieces while discouraging your opponent. Could you define a single heuristic that incorporates all of these ideas? Consider chess, certainly there the heuristic would comprise numerous functions that take into account the strength of your pieces versus your opponent’s, but also their locations.

Do non-game playing situations also have useful heuristics? One simple heuristic for a problem is: h(n) – how likely, if I choose state n, will it lead to a solution? But is it possible to write such a function that can predict the merit of a given choice? We will see later in the semester, an idea called genetic algorithms that promotes this kind of function.