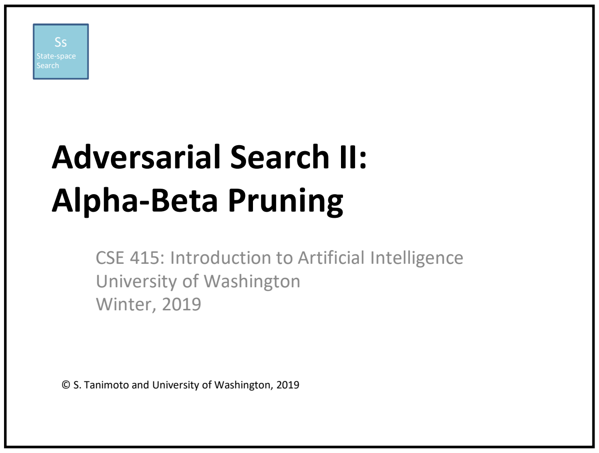
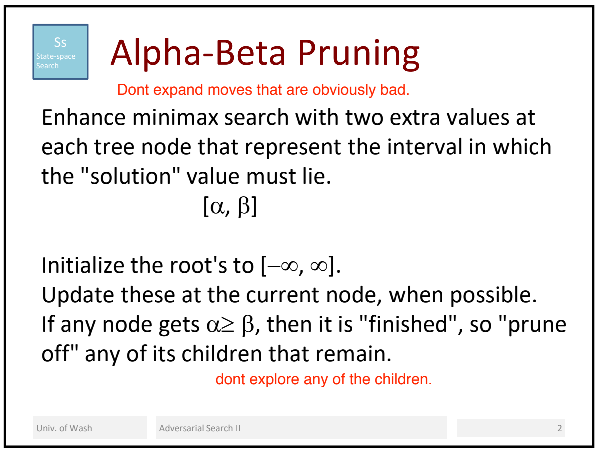
START MON JAN 28 1:05 pm



Ss State-space Search

Adversarial Search II: Alpha-Beta Pruning

CSE 415: Introduction to Artificial Intelligence
University of Washington
Winter, 2019



Ss State-space Search

Alpha-Beta Pruning

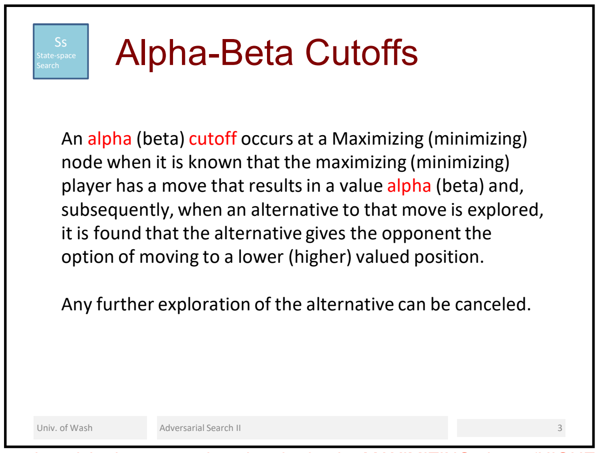
Dont expand moves that are obviously bad.

Enhance minimax search with two extra values at each tree node that represent the interval in which the "solution" value must lie.

$[\alpha, \beta]$

Initialize the root's to $[-\infty, \infty]$.
Update these at the current node, when possible.
If any node gets $\alpha \geq \beta$, then it is "finished", so "prune off" any of its children that remain.

dont explore any of the children.

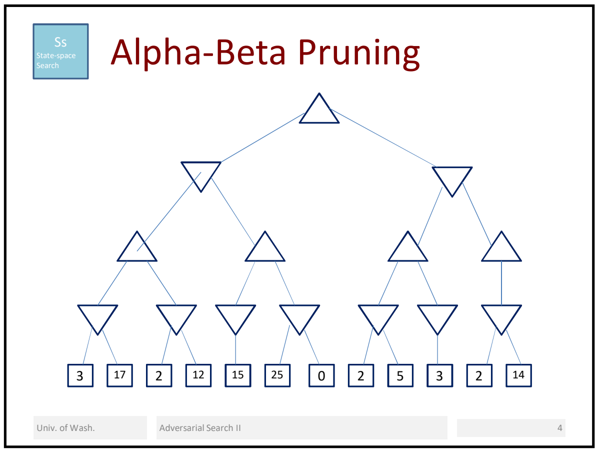


Ss State-space Search

Alpha-Beta Cutoffs

An alpha (beta) cutoff occurs at a Maximizing (minimizing) node when it is known that the maximizing (minimizing) player has a move that results in a value alpha (beta) and, subsequently, when an alternative to that move is explored, it is found that the alternative gives the opponent the option of moving to a lower (higher) valued position.

Any further exploration of the alternative can be canceled.



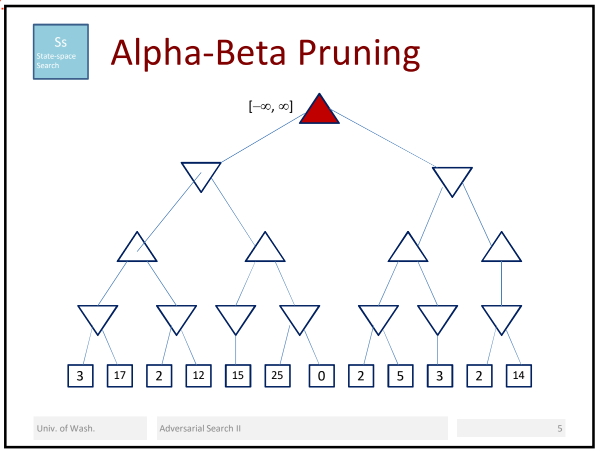
Ss State-space Search

Alpha-Beta Pruning

alpha represents the value of the best move found so far for the MAXIMIZING player (HIGHEST value), along the path from the root to the current node.
Beta represents the value of the best move found so far for the minimizing player (LOWEST value) along the path from the root to the current node

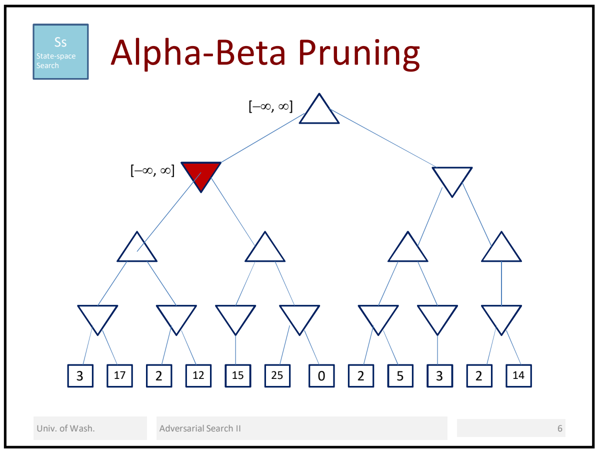
would NOT compute the static evaluations for each leaf right off the bat. We would compute when we reach each node.

Best move - a move that the player can “force’ if the search goes into the part of the tree explored so far.



Ss State-space Search

Alpha-Beta Pruning



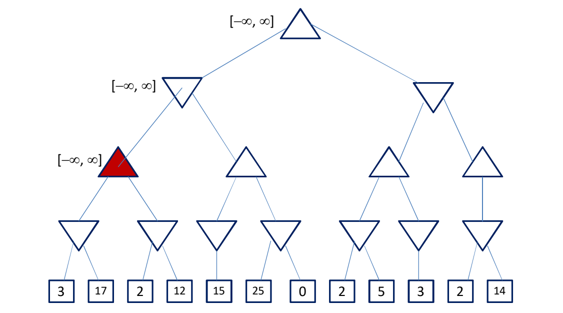
Ss State-space Search

Alpha-Beta Pruning

rather than simply doing a blind right to left search in alpha beta pruning, we will use some of the information we have gathered from moves previously to try to explore best nodes search (kind of like heuristics). Try to think about which move will lead to the best leaf FIRST rather than blindly exploring as this leads to a lot more cutoffs.

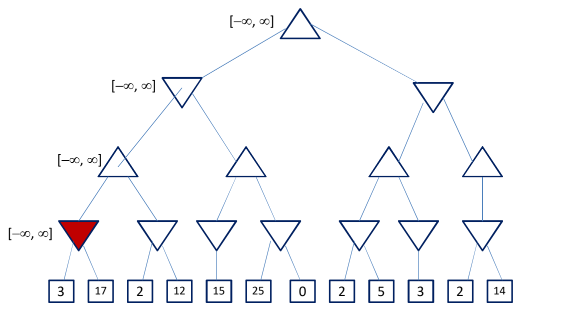
Ss State-space Search

Alpha-Beta Pruning



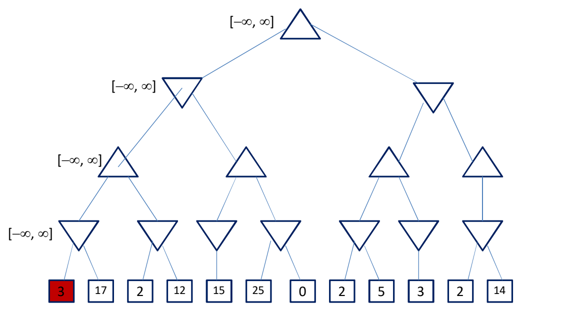
Ss State-space Search

Alpha-Beta Pruning



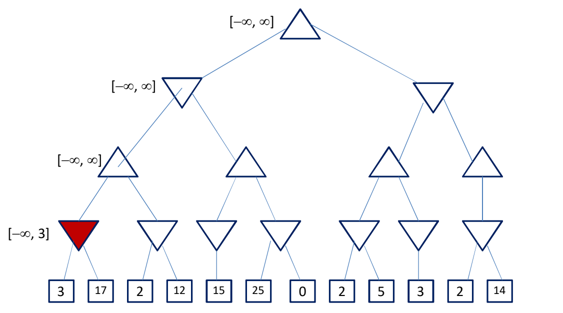
Ss State-space Search

Alpha-Beta Pruning



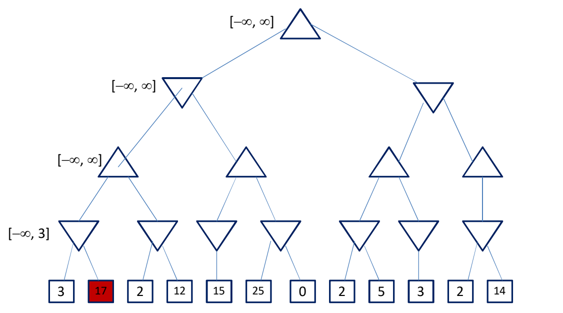
Ss State-space Search

Alpha-Beta Pruning



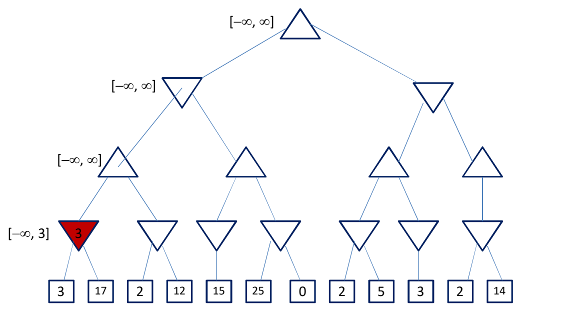
Ss State-space Search

Alpha-Beta Pruning

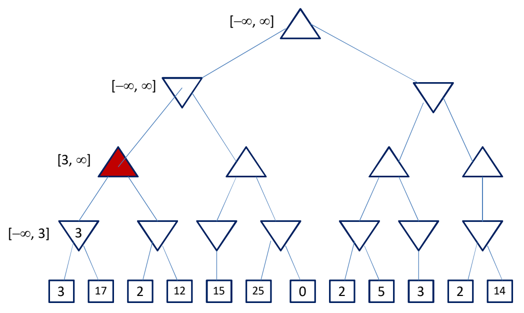


Ss State-space Search

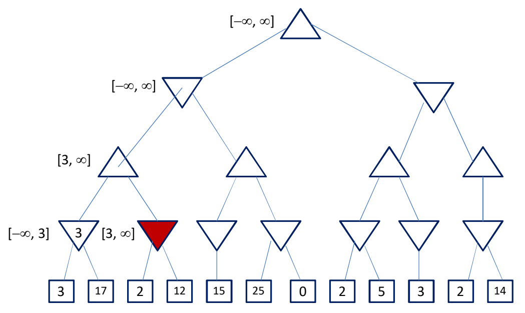
Alpha-Beta Pruning



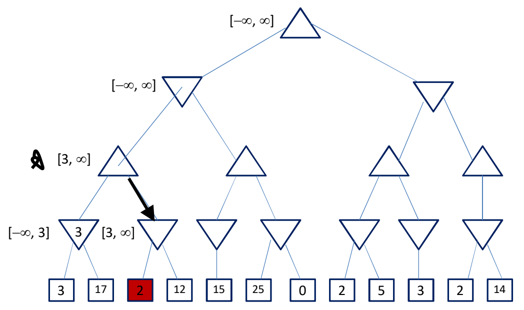
Alpha-Beta Pruning



Alpha-Beta Pruning

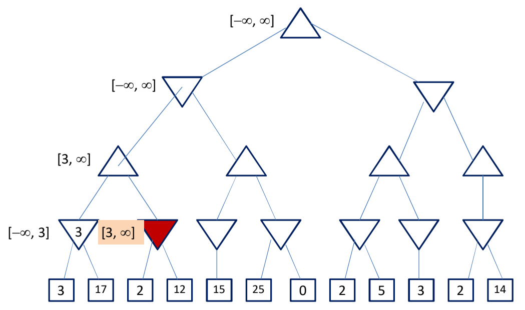


Alpha-Beta Pruning

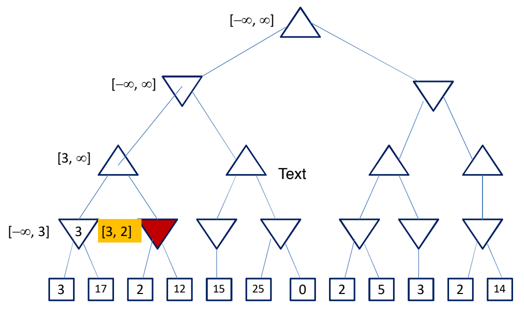


once we see the 2, we know we dont have to explore any of the other children in this root. NO WAY maximizing player at * will choose the move that leads to this two as it is certainly worse than the min of 3 he could get going the other way, so no need to bother considering other possible outcomes from this move (arrow above) as they could only be worse.

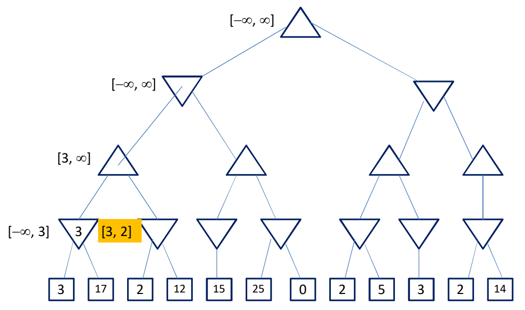
Alpha-Beta Pruning



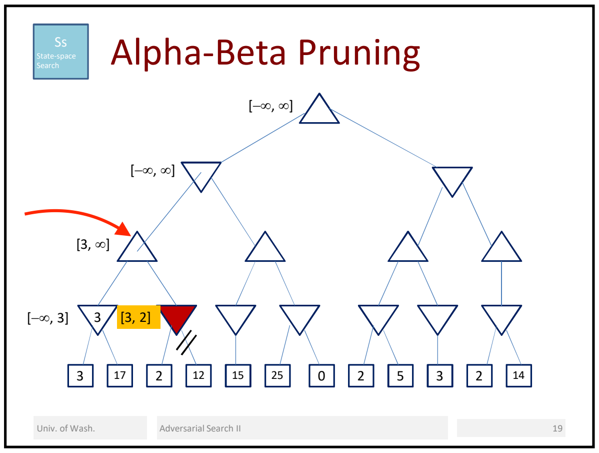
Alpha-Beta Pruning

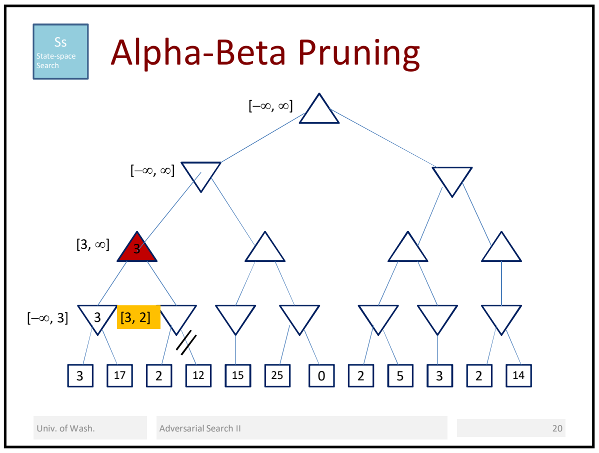


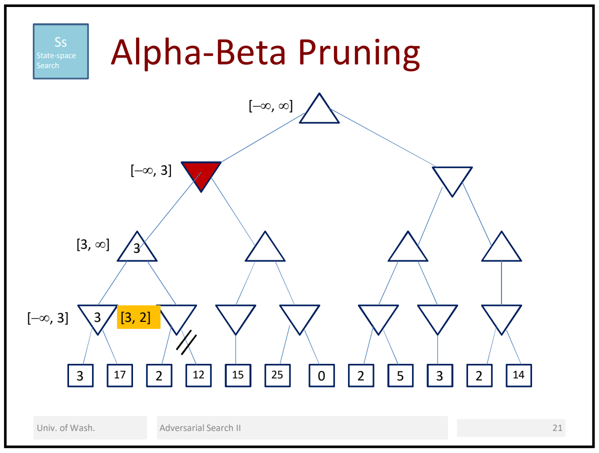
Alpha-Beta Pruning

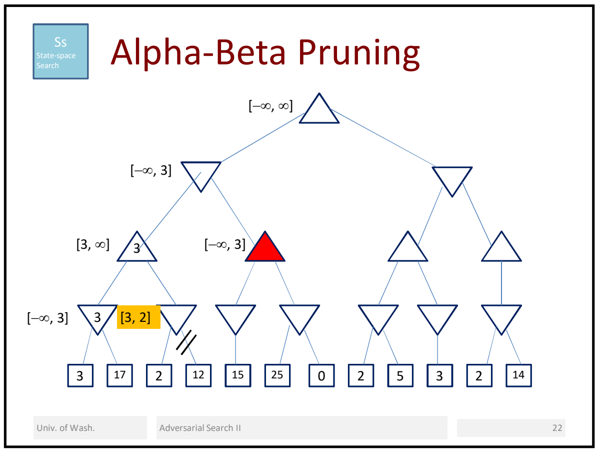


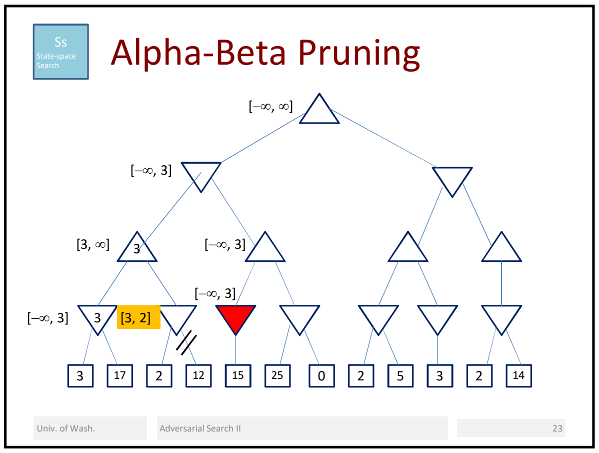
maximizer here would never choose right node because minimizer can force a lower value than what maximizer can force choosing an alternative path (i.e. the left subtree guarantees 3 for the maximizer)

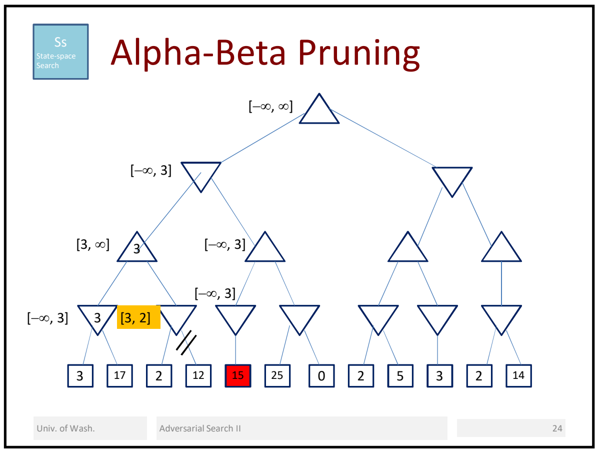




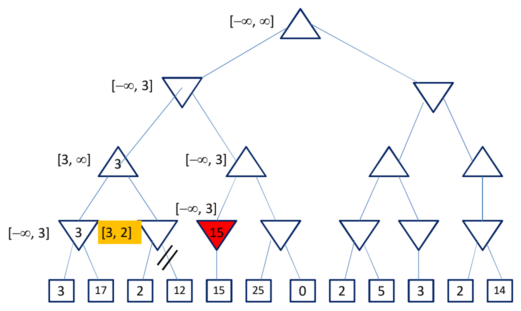




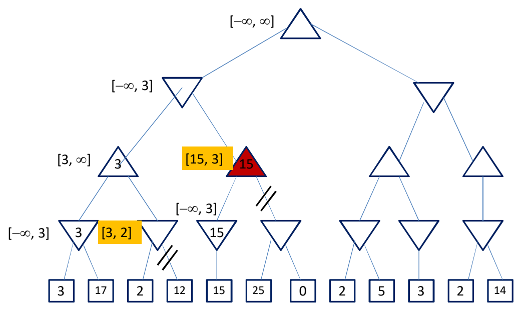




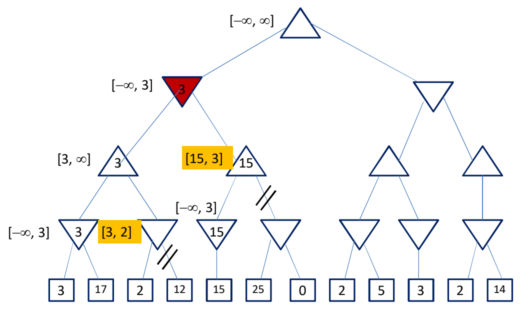
Alpha-Beta Pruning



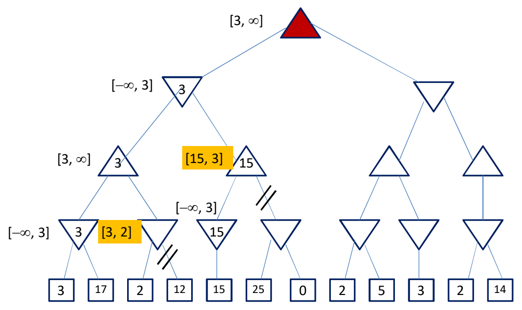
Alpha-Beta Pruning



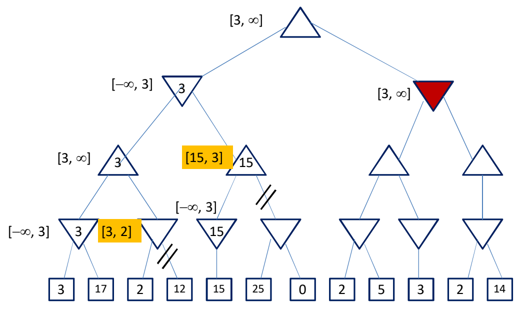
Alpha-Beta Pruning



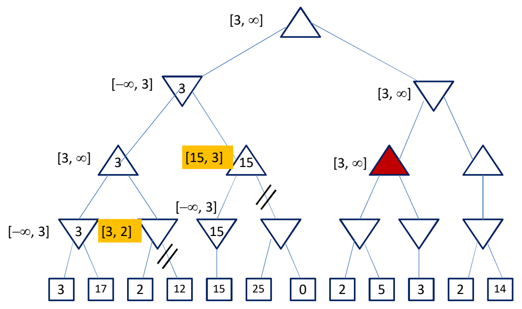
Alpha-Beta Pruning



Alpha-Beta Pruning

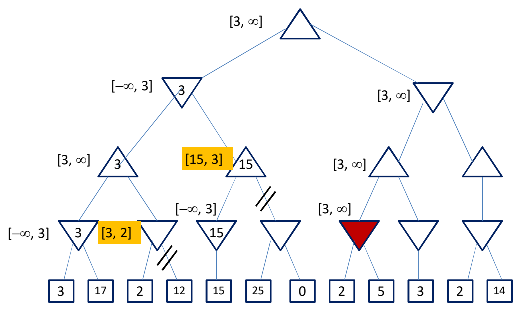


Alpha-Beta Pruning



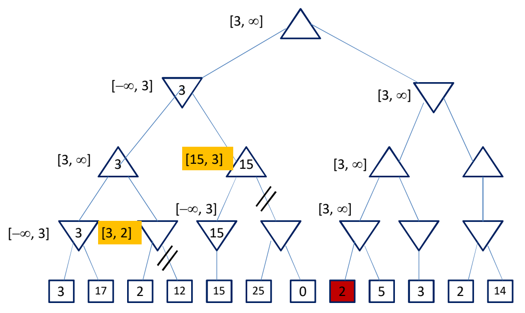
Ss State-space Search

Alpha-Beta Pruning



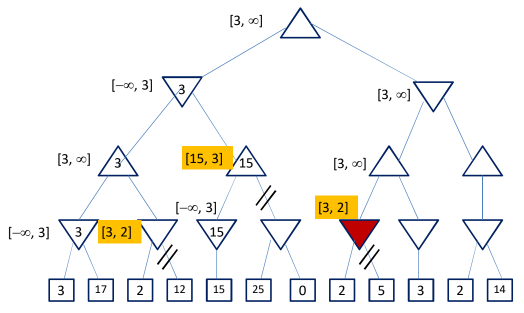
Ss State-space Search

Alpha-Beta Pruning



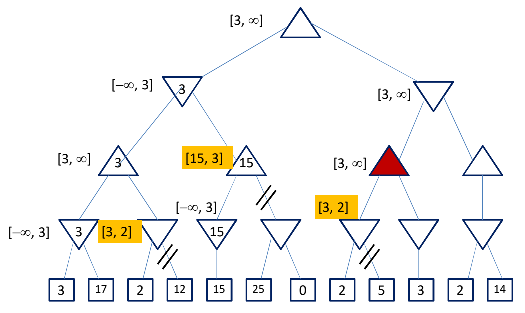
Ss State-space Search

Alpha-Beta Pruning



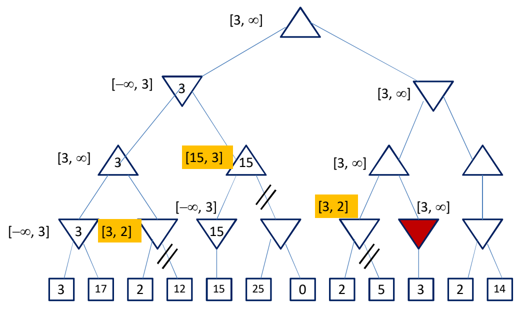
Ss State-space Search

Alpha-Beta Pruning



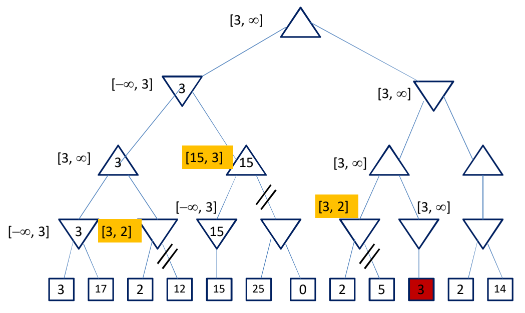
Ss State-space Search

Alpha-Beta Pruning

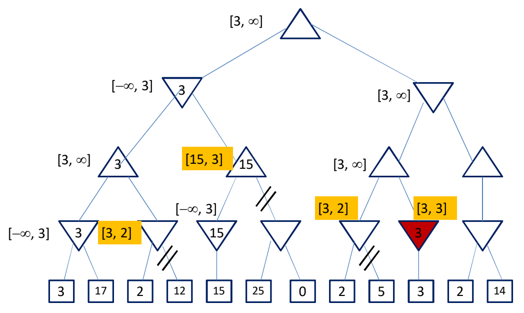


Ss State-space Search

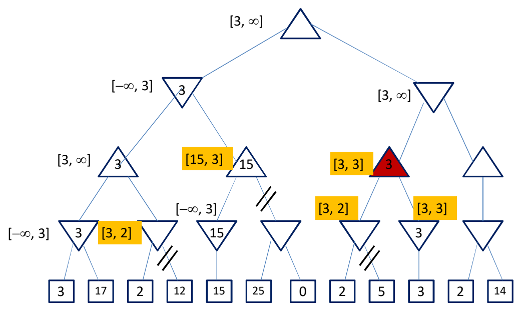
Alpha-Beta Pruning



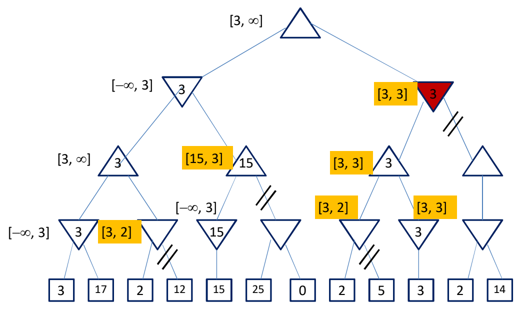
Alpha-Beta Pruning



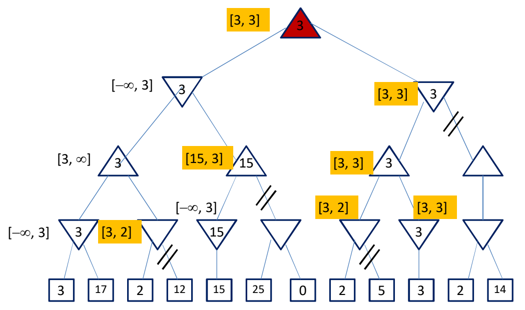
Alpha-Beta Pruning



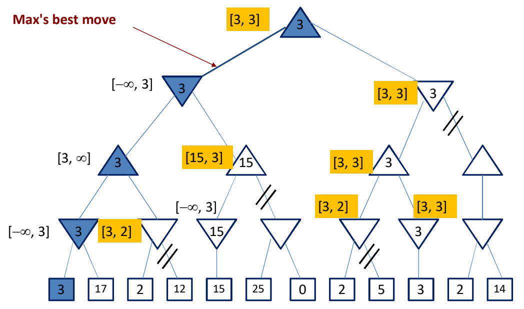
Alpha-Beta Pruning



Alpha-Beta Pruning



Alpha-Beta Pruning



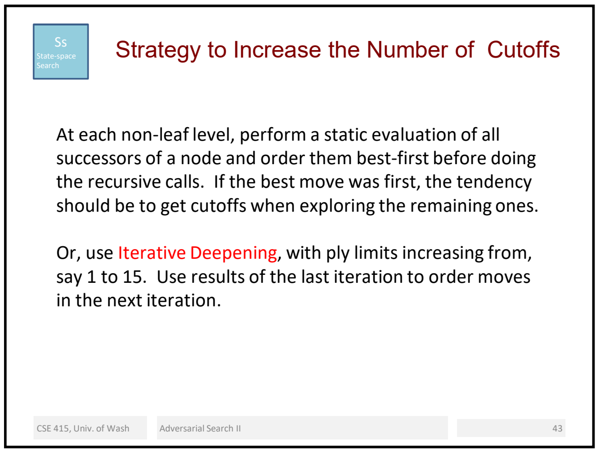
Strategy to Increase the Number of Cutoffs

At each non-leaf level, perform a static evaluation of all successors of a node and order them best-first before doing the recursive calls. If the best move was first, the tendency should be to get cutoffs when exploring the remaining ones.

Or, use Iterative Deepening, with ply limits increasing from, say 1 to 15. Use results of the last iteration to order moves in the next iteration.

gives us information about intermediate states; can help us choose which order to traverse at each level

In games like chess, α-β pruning typically allows searching ahead 2 times as deep. It tends to reduce the effective branching factor from d to approx. sqrt(d).



Ss
State-space
Search

Strategy to Increase the Number of Cutoffs

At each non-leaf level, perform a static evaluation of all successors of a node and order them best-first before doing the recursive calls. If the best move was first, the tendency should be to get cutoffs when exploring the remaining ones.

Or, use Iterative Deepening, with ply limits increasing from, say 1 to 15. Use results of the last iteration to order moves in the next iteration.