CSE 332: Data Structures and Parallelism

Section 8: P3 and Search Solutions

0. MiniMax

(a) Will the move returned by an Alpha-Beta search (always), or (never) be better than the move returned by a MiniMax search on the same board with the same search parameters?

Solution:

Never (just faster). Alpha-Beta only prunes branches which are guaranteed not to affect the final result.

(b) Explain why we *negate* the result of the recursive call in MiniMax.

Solution:

We do this in order to simulate the idea of switching between your perspective and your opponent's perspective. By using the assumption that your opponent is playing optimally, if you negate the result of the recursive call, then on your opponent's turn, they will choose the move with the lowest value for you. When the call returns, you choose the highest value from the choices you've simulated for your opponent.

(c) Why is MiniMax "naturally parallelizable", while Alpha-Beta is not?

Solution:

Alpha-Beta threads rely on information computed in sibling threads (the current values of `alpha` and `beta`), while MiniMax threads can determine their results independent of any of their siblings.

1. Cutoffs

Provide a short diagram or description to explain the following parameters from P3:

(a) `ply`

Solution:

The total number of levels ahead looked (so, the height of your search tree).

(b) `cutoff`

Solution:

The number of levels remaining in the tree when your search switches to a non-parallel algorithm.

(c) `divideCutoff`

Solution:

The maximum number of threads which should be forked sequentially when dividing-and-conquering a list of moves. Similar to the `sequentialCutoff` parameter from exercises.

(d) `PERCENTAGE_SEQUENTIAL`

Solution:

The maximum percentage of a list of moves which should be forked sequentially in order to determine reasonable values for `alpha` and `beta`.

2. Efficiency

Circle the **most efficient** option from each pair of possible implementation strategies for P3:

(a) To create threads for each move in a `List<M>` during Parallel Minimax:

Create threads in a `for` loop **OR** Create threads with divide-and-conquer

Solution:

Create threads with divide-and-conquer.

(b) To pass copies of boards to these threads:

Copy the board *inside* the thread **OR** Copy the board *before* passing it to the thread

Solution:

Copy the board *inside* the thread.

(c) To evaluate a list of moves using Alpha-Beta pruning:

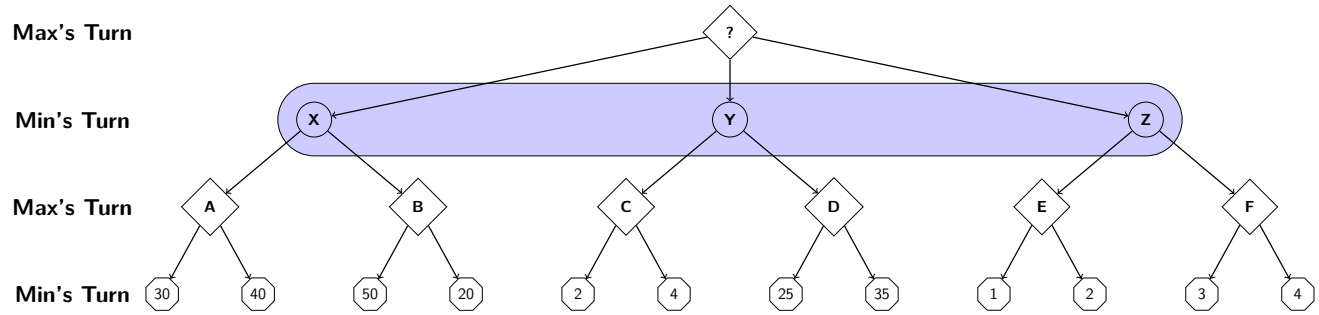
Evaluate the moves in the order provided **OR** Sort the moves best-first, then evaluate in sorted order

Solution:

Sort the moves. Sorting is fast, and allows us to prune more effectively by establishing tight Alpha/Beta bounds.

3. Alpha-Beta

Determine the value of the root node after running Alpha-Beta on the following tree (and cross out pruned branches/nodes):



Solution: