

Adversarial Search II: Alpha-Beta Pruning

CSE 415: Introduction to Artificial Intelligence
University of Washington
Winter, 2018

© S. Tanimoto and University of Washington, 2018

1

Alpha-Beta Pruning

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.

Univ. of Wash

Adversarial Search II

2

2

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.

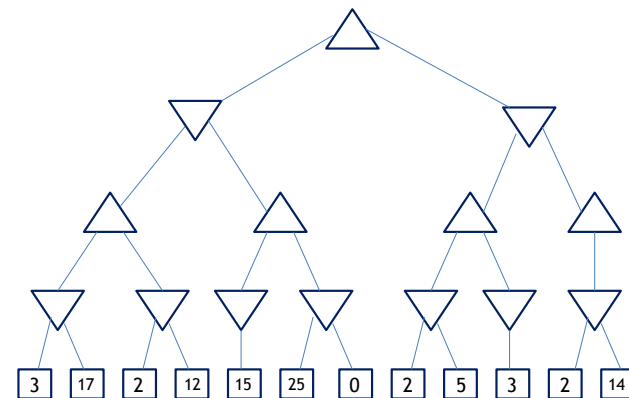
Univ. of Wash

Adversarial Search II

3

3

Alpha-Beta Pruning



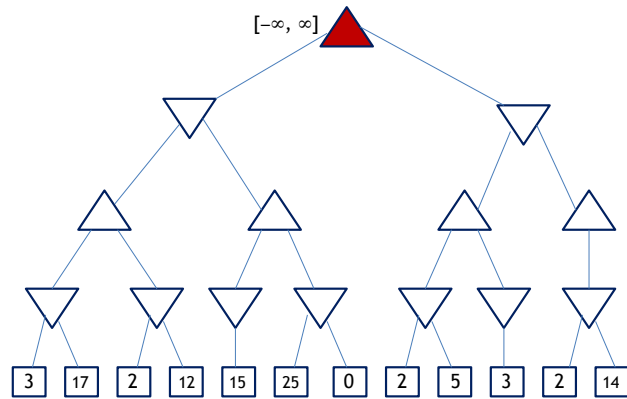
Univ. of Wash.

Adversarial Search II

4

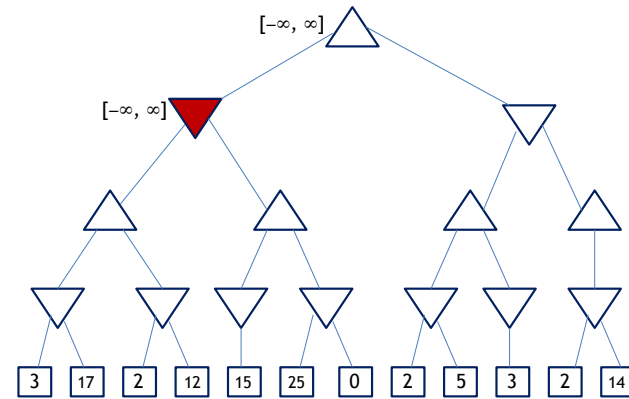
4

Alpha-Beta Pruning



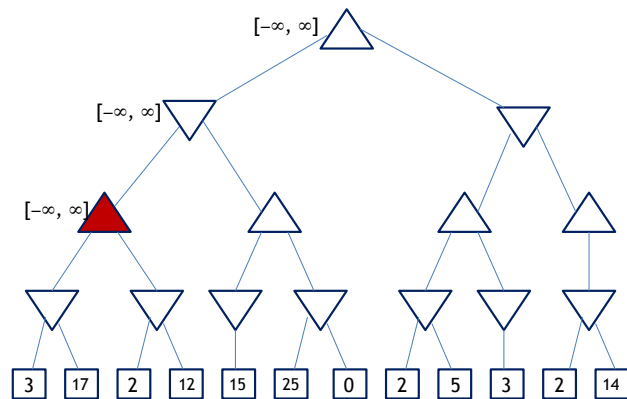
5

Alpha-Beta Pruning



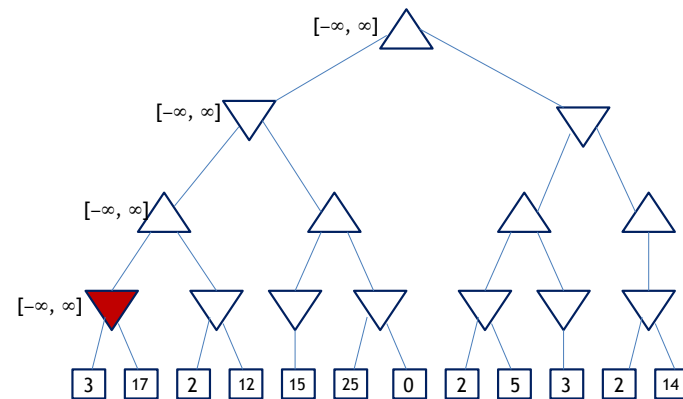
6

Alpha-Beta Pruning



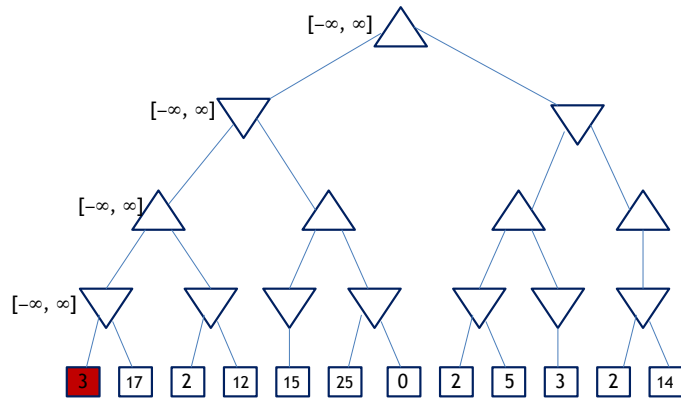
7

Alpha-Beta Pruning



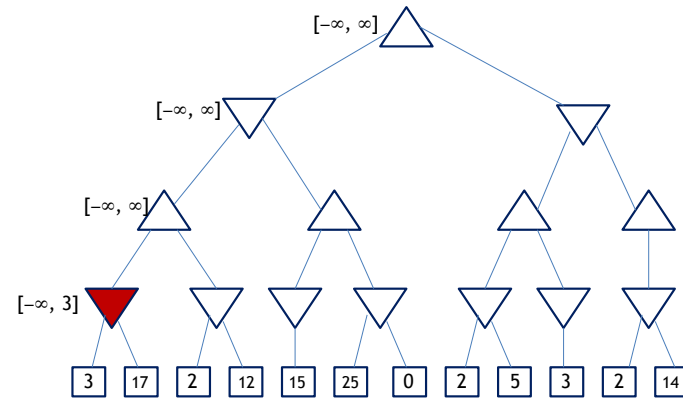
8

Alpha-Beta Pruning



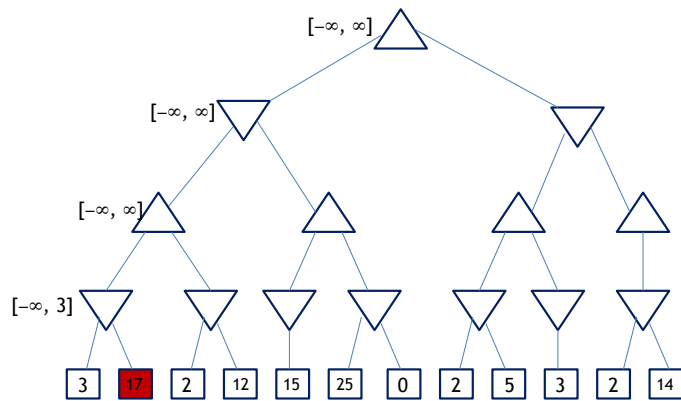
9

Alpha-Beta Pruning



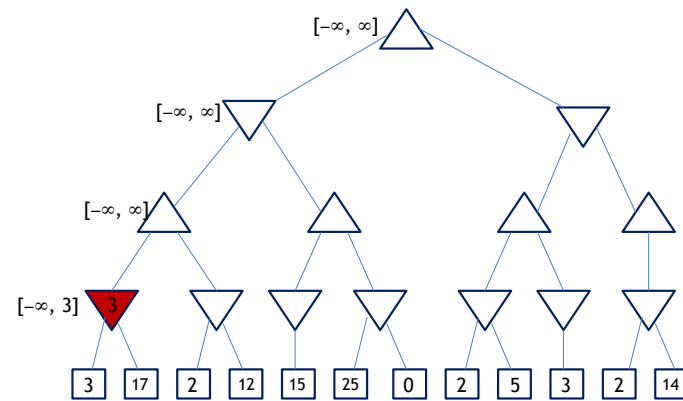
10

Alpha-Beta Pruning



11

Alpha-Beta Pruning



12

Alpha-Beta Pruning

Ss
State-space
Search

[-∞, ∞]

[-∞, ∞]

[3, ∞]

[-∞, 3]

3 17 2 12 15 25 0 2 5 3 2 14

Univ. of Wash. Adversarial Search II 13

13

[illegible]

14

[illegible]

15

[illegible]

16

Alpha-Beta Pruning

Univ. of Wash. Adversarial Search II 17

17

[illegible]

18

Alpha-Beta Pruning

Univ. of Wash. Adversarial Search II 19

19

Alpha-Beta Pruning

State-space Search

Alpha-Beta Pruning

Diagram illustrating a search tree for Alpha-Beta Pruning. The tree structure shows nodes and their associated values and ranges:

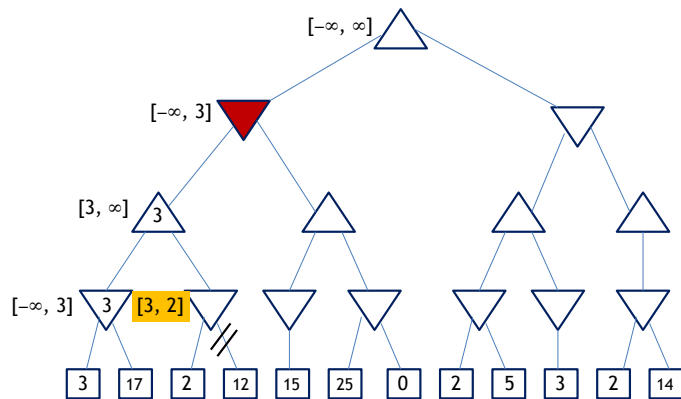
- Root node (Max): $[-\infty, \infty]$
- Left child (Min): $[-\infty, \infty]$
- Left child of left child (Max): $[3, \infty]$ (highlighted in red, value 3)
- Right child of left child (Min): $[-\infty, 3]$ (value range $[3, 2]$ highlighted in yellow)
- Leaf nodes (values): 3, 17, 2, 12, 15, 25, 0, 2, 5, 3, 2, 14

The diagram demonstrates the pruning process, where the search is terminated early when the current node's value is already less than or equal to the best value found so far (indicated by the double slash //).

Univ. of Wash. Adversarial Search II 20

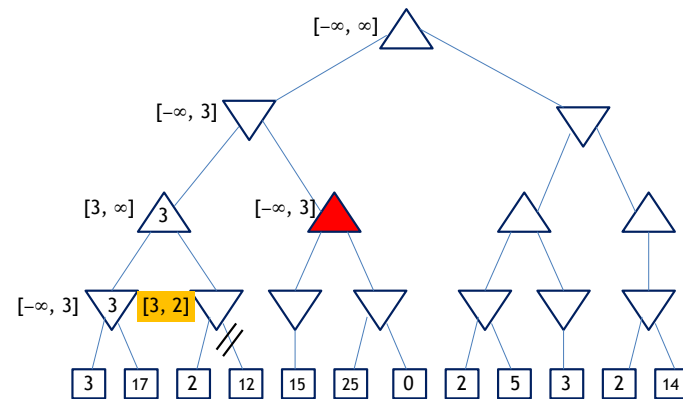
20

Alpha-Beta Pruning



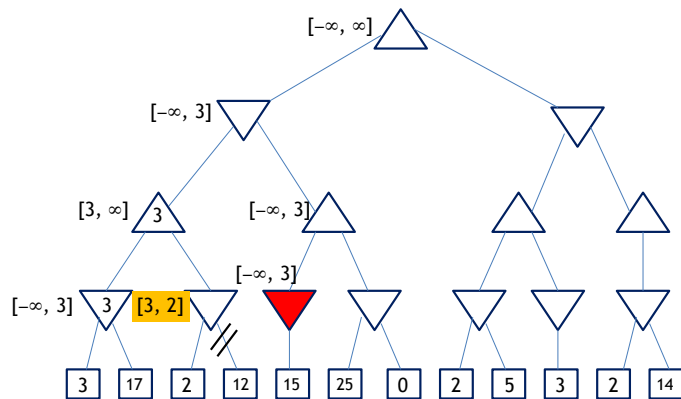
21

Alpha-Beta Pruning



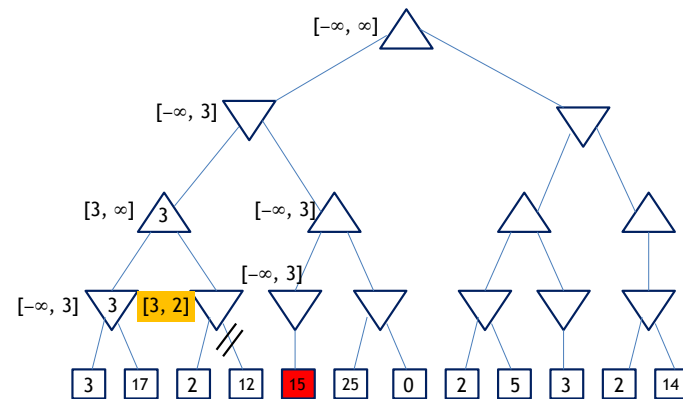
22

Alpha-Beta Pruning



23

Alpha-Beta Pruning



24

Univ. of Wash. Adversarial Search II 25

State-space Search

Alpha-Beta Pruning

Diagram illustrating a minimax search tree with Alpha-Beta Pruning. The root node is a Max node with a value range of $[-\infty, \infty]$. It branches to two Min nodes. The left Min node has a value range of $[-\infty, 3]$ and branches to two Max nodes. The left Max node has a value range of $[3, \infty]$ and a value of 3. The right Max node has a value range of $[15, 3]$ and a value of 15, which is highlighted in red. The right Min node has a value range of $[-\infty, 3]$ and branches to two Max nodes. The left Max node has a value range of $[-\infty, 3]$ and a value of 3. The right Max node has a value range of $[3, 2]$ and a value of 2, which is highlighted in yellow. The tree continues to have more levels, with values 3, 17, 2, 12, 15, 25, 0, 2, 5, 3, 2, 14 at the bottom. Pruning is indicated by double slashes (//) on the branches leading to the red node and the yellow node.

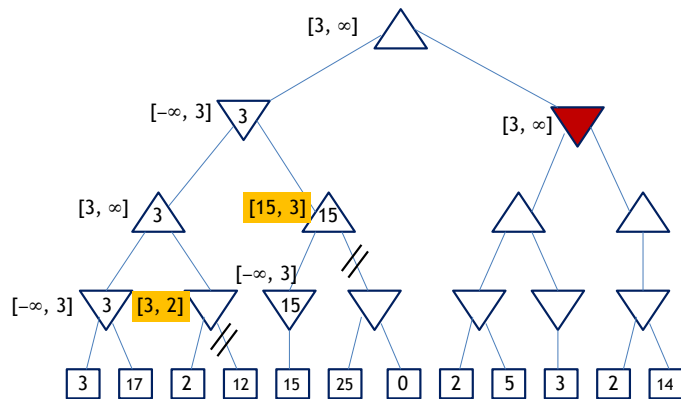
Univ. of Wash. Adversarial Search II 26

[illegible]

Alpha-Beta Pruning

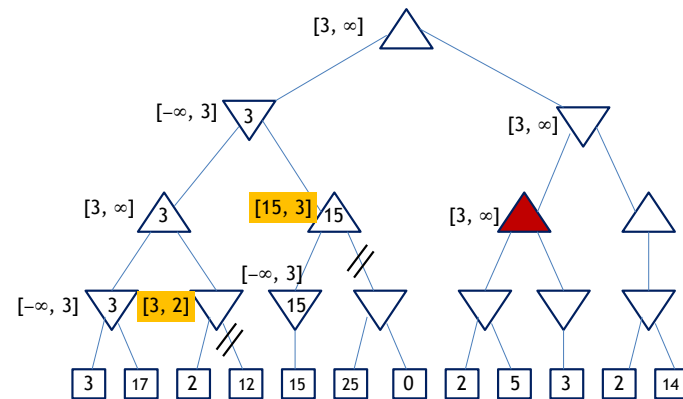
The diagram illustrates a minimax search tree for Alpha-Beta Pruning. The root node is a red triangle labeled $[3, \infty]$. It branches into two white triangle nodes. The left branch leads to a node labeled $[-\infty, 3]$ containing the value 3. This node further branches into two white triangle nodes. The left child of this node is labeled $[3, \infty]$ and contains the value 3. Its right child is labeled $[15, 3]$ and contains the value 15. The right child of the root is also a white triangle. The left child's left child (the node with value 3) has two children: a white triangle labeled $[-\infty, 3]$ containing 3, and a white triangle labeled $[3, 2]$ containing 2. The right child of the node with value 3 is a white triangle labeled $[-\infty, 3]$ containing 15. The left child of the node with value 15 is a white triangle labeled $[15, 3]$ containing 15. The right child of the node with value 15 is a white triangle. The leaf nodes are represented by blue squares. The leaf nodes under the node with value 3 are 3, 17, 2, and 12. The leaf nodes under the node with value 15 are 15, 25, 0, and 2. The leaf nodes under the right child of the root are 5, 3, 2, and 14. Double slashes (//) indicate pruning at the nodes with values 2 and 15.

Alpha-Beta Pruning



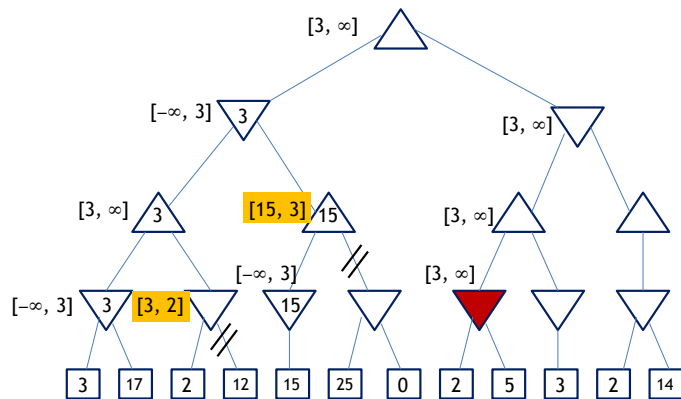
29

Alpha-Beta Pruning



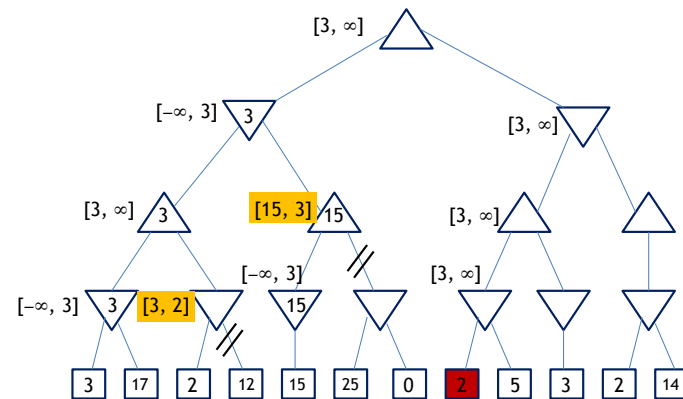
30

Alpha-Beta Pruning



31

Alpha-Beta Pruning



32

Alpha-Beta Pruning



Alpha-Beta Pruning



Alpha-Beta Pruning

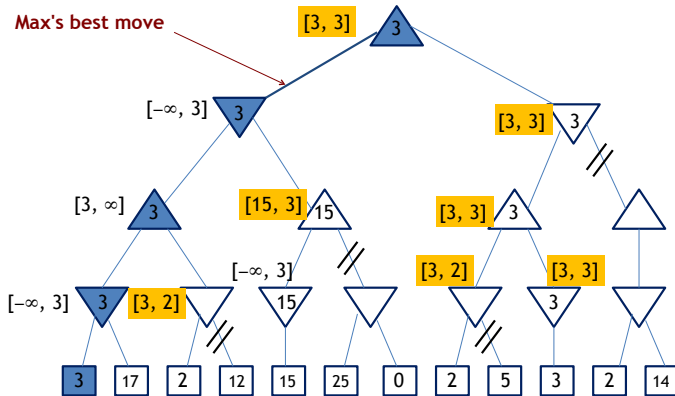


Alpha-Beta Pruning



40

Alpha-Beta Pruning



41

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.

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} .

42

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.

43