



CS-208:Artificial Intelligence

Lectures-11

Game Playing

Game Playing

Games appeared to be a good domain to explore machine intelligence because of two reasons

- 1. Games provide a structured task in which it is very easy to measure success or failure.*
- 2. Games generally do not require large amount of knowledge*(this is true for simplest games).

In games, the player know what they have done and what they can do. Here we are interested in two players game with the objective that one of the two players has to win or at least draw the game.

The problem reduction approach can also be used to find a winning strategy through the process of proving that the game can be won.

Finding the Winning Strategy

Suppose we name the two players as PLUS and MINUS. Consider the problem of finding the strategy for the PLUS PLAYER starting with a given configuration represented by \mathbf{X}^t .

Where

t stands for either + or -.

\mathbf{X}^+ represents a configuration in which it is the PLUS turn to make the next move.

\mathbf{X}^- represents a configuration in which it is the MINUS turn to make the next move.

We would like to prove that PLUS can win from \mathbf{X}^t or at least PLUS can draw from \mathbf{X}^t and let us describe the problem of proving that PLUS can win from the configuration \mathbf{X}^t by the expression $W(\mathbf{X}^t)$.

Suppose it is PLUS turn to move next from the configuration \mathbf{X}^+ and there are N legal moves. It will result in configuration $W(X_1^-)$, $W(X_2^-)$, $W(X_3^-)$, . . . $W(X_N^-)$. The problem reduction operator will generate a game tree.

Illustration: Grundy's Game

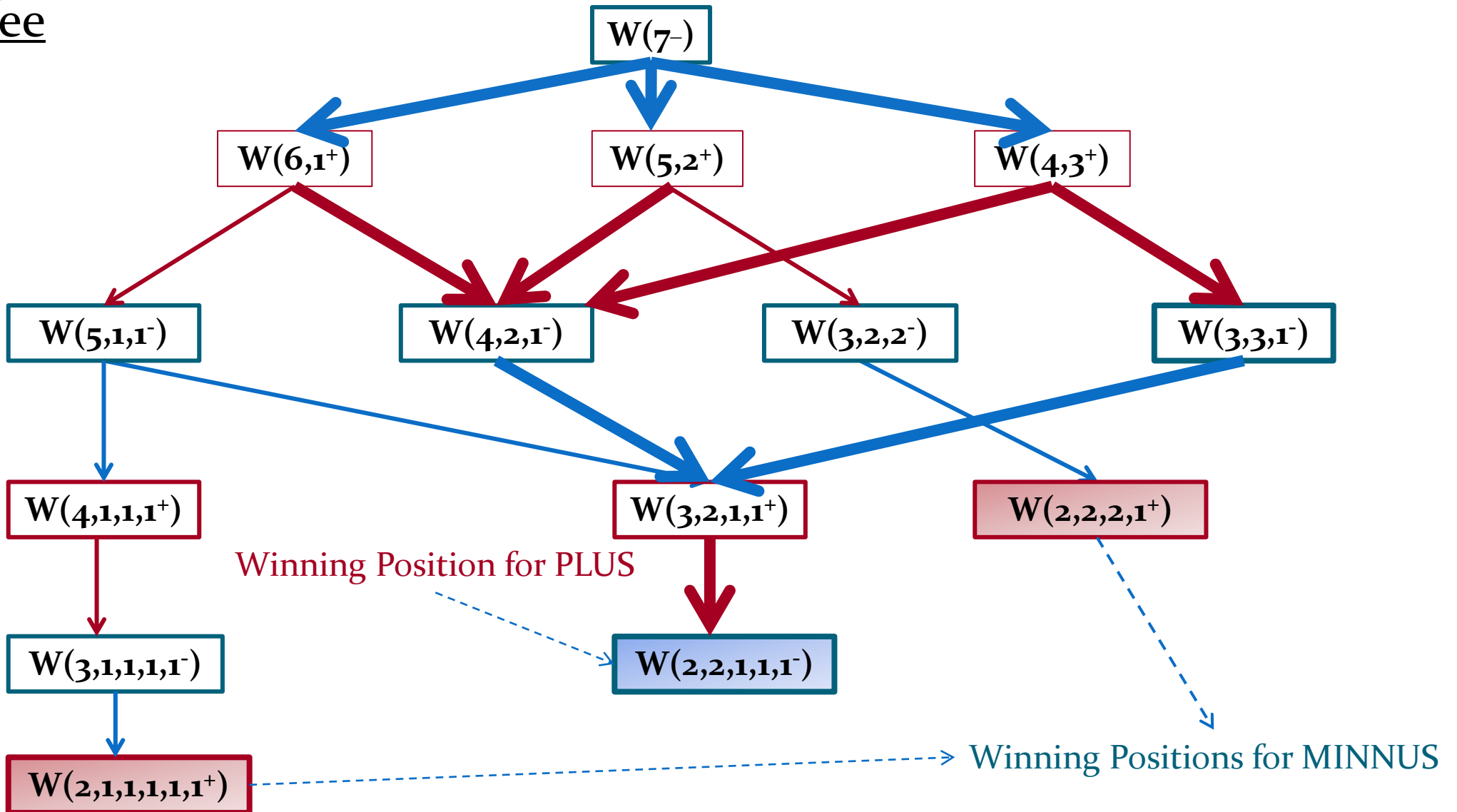
The Rules of the Grundy's Game: Two players have a single pile of objects say a stack of coins in front of them. The first player divides the original stacks in to two stacks that must be unequal. Each player alternatively does the same to some single stack until every stack has either just one penny or two. The player who first can not play is the lose

Problem: *Generate the Game Tree from the Starting Configuration $W(7^+)$, and Find the Winning strategy*

Two Things can be done to improve the effectiveness of the search based problem solving:

- I. Improve the generate procedure, so that only good move are generated.
- II. Improve the test procedure so that the best path will be recognized and explored.

Game Tree



Minimax Procedure

- It is a look ahead procedure
- It is a depth limited depth first procedure

Suppose a situation analyzer that converts overall judgments about a situation in to a single quality number. where

- +ve number by convention favours to one player
- -ve number by convention favours to other player
- The degree of favour goes with the absolute value of the number.

The procedure of determining the quality number is called *Static Evaluation*. At the end of limited exploration of move possibilities , the static evaluation score is produced by the situation analyzer called *Static Evaluator*. The player hoping for +ve number is called **maximizing player** and his opponent is the **minimizing player**.

Minimax Algorithm (It is a Recursive Procedure)

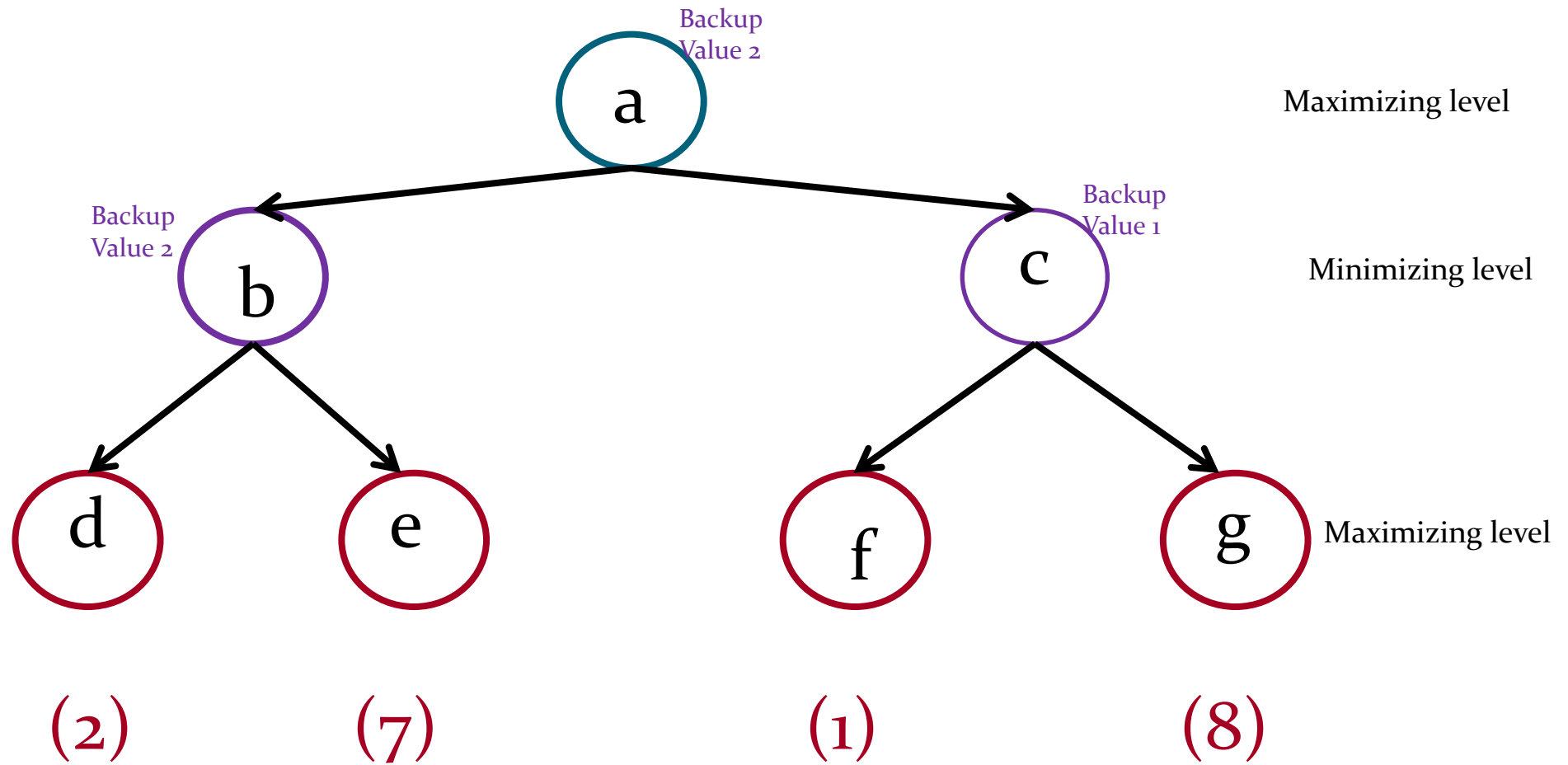
1. Determine if the limit of search has been reached **or**
If the current level is minimizing level **or**
If the current level is maximizing level
 - a) If the limit of search has been reached then
compute the static score of the current position relative to the appropriate player and report the result.
 - b) If the level is minimizing level then
*Apply **Minimax** on each successor of the current position and report the minimum of the results.*
 - c) If the level is maximizing level *then*
*Apply **Minimax** on each successor of the current position and report the maximum of the results.*

A good first move can be extracted by the Minimax procedure. If the maximizing player has to choose one among the tip nodes, the player would choose the node having the largest static score value. Therefore the parent of the tip nodes is assigned a back-up value equal to the maximum of the evaluations of the tip nodes

Minimax Algorithm (Refined Version)

1. If the current position is a final one then
 - a) Compute the static value of the current position.
 - b) Return the static value.
2. If Min is on move then
 - a) Generate the successors of the current position
 - b) Apply Minimax to each of these successors with Max to move next.
 - c) Return the minimum of the results.
3. If Max is on move then
 - a) Generate the successors of the current position
 - b) Apply Minimax to each of these successors with Min to move next.
 - c) Return the maximum of the results.

Example Game Tree



Minimax(Current_Node, Current_Player, Current_Level, Depth_Limit)

Minimax(a, Max, 1, 3)

Player=Max

Successors = {b, c}

Minimax (b, Min, 2, 3)

Player=Min

Successors = {d, e}

Minimax(d, Max, 3,3)

Report Static Score 2

Minimax(e, Max, 3,3)

Report Static Score 7

Return smallest Static Score (2, 7) = 2

Minimax (c, Min, 2, 3)

Player=Min

Successors = {f, g}

Minimax(f, Max, 3,3)

Report 1

Minimax(g, Max, 3,3)

Report 8

Return smallest Static Score (1, 8) = 1

Return Largest Reported value (2, 1) = 2

Minimax with Alpha-Beta Pruning

Minimax with Alpha-Beta Pruning Algorithm (It is also a Recursive Procedure)

1. Determine if the level is the top level **or**
if the limit of search has been reached **or**
if the current level is minimizing level **or**
if the current level is maximizing level
 - a) If the level is the top level then
Alpha be $-\infty$ and Beta be $+\infty$
 - b) If the limit of search has been reached then
Compute the static score of the current position relative to the appropriate player and report the result.

c) If the level is minimizing level then

Repeat the following steps until all the successors of the current position are examined with Minimax or α is greater than β

- i. Set β to the smaller of the current β value and the smallest value so far reported by Minimax working on the current position's successors*
- ii. Use Minimax on the next successor of the current position handling this new application of Minimax and current α β values*

Report β

d) If the level is maximizing level then

Repeat the following steps until all the successors of the current position are examined with Minimax or alpha is greater than beta


- i. Set Alpha to the larger of the current Alpha value and the biggest value so far reported by Minimax working on the current position's successor*
- ii. Use Minimax on the next successor of the current position handling this new application of Minimax and current Alpha Beta values*

Report Alpha

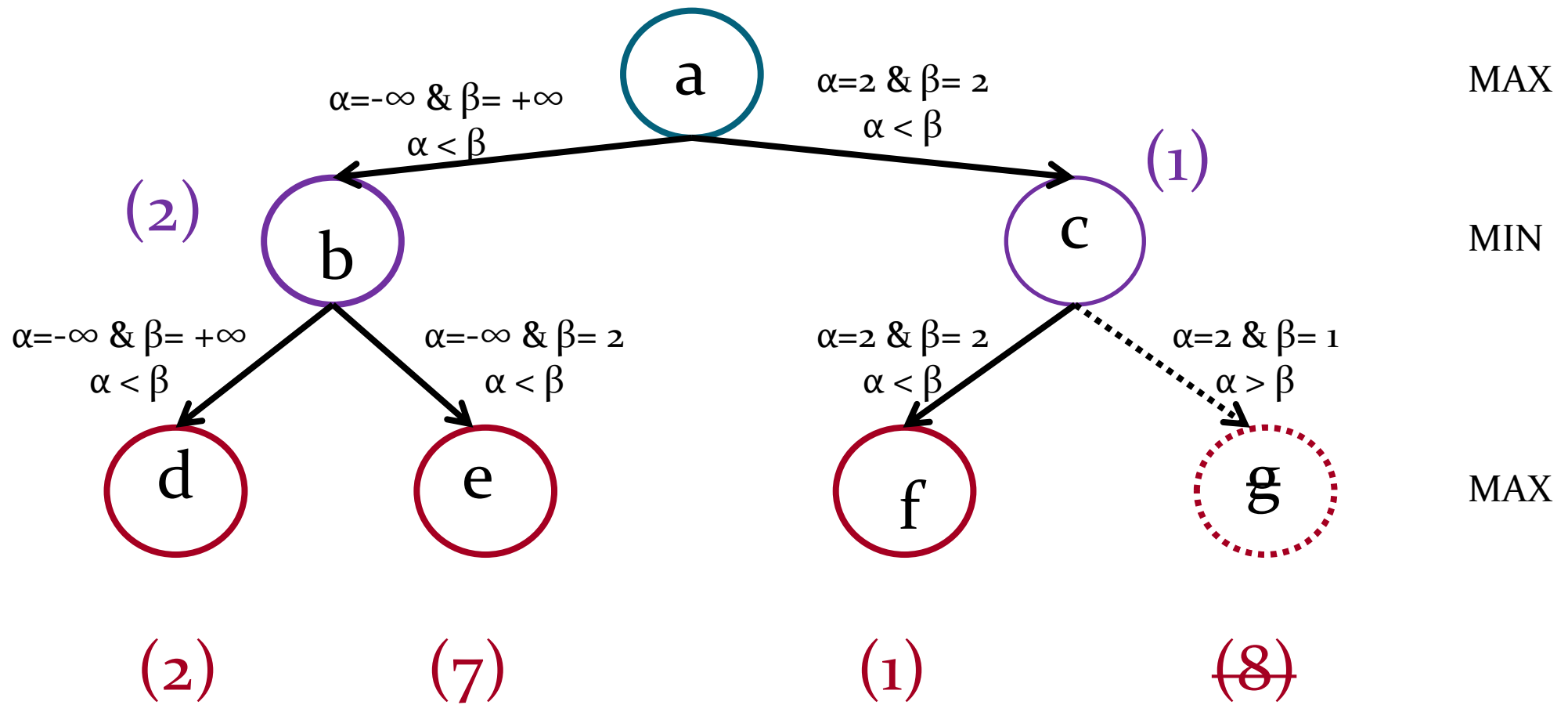
Minimax with Alpha-Beta Pruning (Refined Version)

AlphaBeta()

1. If current_level = top level then
 - $\alpha = -\infty$ and
 - $\beta = +\infty$
2. If current_level = limit_of_search then
Compute the static score of the current position relative to the appropriate player and return the result.
3. If Min is on Move then
 - a. Generate a list of successors of the current position.
 - b. If $(\alpha > \beta)$ or (successor_list = []) then
 - Terminate
 - Return β
 - c. $\beta_c = \text{Alpha Beta}(\text{successor_list}[1])$
 - d. $\beta = \text{Minimum}(\beta_c, \beta)$
 - e. Delete the first element from the successor_list and go back to 3b

- 
4. If Max is on Move then
 - a. Generate a list of successors of the current position.
 - b. If $(\alpha > \beta)$ or $(\text{successor_list} = [])$ then
 - Terminate
 - Return α
 - c. $\alpha_c = \text{Alpha Beta}(\text{successor_list}[1])$
 - d. $\alpha = \text{Maximum}(\alpha_c, \alpha)$
 - e. Delete the first element from the successor_list and go back to 4b

- Example Game Tree



AlphaBeta(Current_Node, Current_Player, Current_Level, Depth_Limit)

AlphaBeta((a, Max, 1, 3))

Player=Max

$\alpha = -\text{maxint}$

$\beta = +\text{maxint}$

Successors = {b, c}

AlphaBeta((b, Min, 2, 3))

Player=Min

Successors = {d, e}

AlphaBeta((d, Max, 3, 3))

Report Static Score 2

$\beta_c = 2$

$\beta = \min(+\text{maxint}, 2) = 2$

AlphaBeta((e, Max, 3, 3))

Report Static Score 7

$\beta_c = 7$

$\beta = \min(2, 7) = 2$

Return smallest Static Score = 2

$\alpha_c = 2$

$\alpha = \text{Max}(-\text{maxint}, 2) = 2$

AlphaBeta((c, Min, 2, 3))

Player=Min

Successors = {d, e}

AlphaBeta(f, Max, 3, 3)

Report Static Score 1

$\beta_c = 1$

$\beta = \min(2, 1) = 1$

→ Here $\alpha > \beta$ Pruning Take place

Return Static Score = 1

Return Largest Reported value (2, 1) = 2