Alpha-Beta Algorithm

```
function Max-Value (s, α, β)
  inputs:
    s: current state in game, Max about to play
    α: best score (highest) for Max along path to s
    β: best score (lowest) for Min along path to s
  if ( s is a terminal state )
    then return ( SBE value of s )
  else for each s' in Successors(s)
    α := max( α, Min-Value(s', α, β))
    if ( α ≥ β ) then return α  /* prune remaining children of Max */
  return α

function Min-Value(s, α, β)
  if ( s is a terminal state )
    then return ( SBE value of s)
  else for each s' in Successors(s)
    β := min( β, Max-Value(s', α, β))
    if (α ≥ β ) then return β  /* prune remaining children of Min */
  return β
```

Starting from the root:
Max-Value(root, -∞, +∞)
Alpha-Beta Example

\[ \alpha = -\infty, \beta = +\infty \]
Alpha-Beta Example
Alpha-Beta Example
Alpha-Beta Example
Alpha-Beta Example

\[
\alpha(F) = 4, \text{ maximum seen so far}
\]
Alpha-Beta Example

Call Stack

max

min

max

min

brown: terminal state

알파-베타 예시

호출 스택
Alpha-Beta Example

Call Stack

brown: terminal state (depth limit)
Alpha-Beta Example

\[ \text{beta}(0) = -3, \text{ minimum seen so far} \]
Alpha-Beta Example

O's beta ≤ F's alpha: stop expanding O (alpha cutoff)
Why? Smart opponent will choose W or worse, thus O's upper bound is –3. So computer shouldn't choose O:-3 since N:4 is better.
Alpha-Beta Example
Alpha-Beta Example

\[ \beta(B) = 4, \text{ minimum seen so far} \]
Alpha-Beta Example
\( \beta(B) = -5 \), updated to minimum seen so far

\( \alpha = -\infty \)

Call Stack

\( \max \)

\( \min \)

brown: terminal state
Alpha-Beta Example

$\alpha(A) = -5$, maximum seen so far
Alpha-Beta Example

Call Stack

max

min

max

min

brown: terminal state
Alpha-Beta Example

Call Stack

min

max

min

brown: terminal state
Alpha-Beta Example

\[ \beta(C) = 3, \text{ minimum seen so far} \]
Alpha-Beta Example

Diagram showing a tree of nodes with values and labels, illustrating the Alpha-Beta pruning algorithm.
Alpha-Beta Example

beta(C) not changed (minimizing)

Call Stack

A

max

B

min

F

min

N

max

G

H

I

J

P

Q

R

min

O

W

X

alpha = -5, beta = 3

beta(C) not changed (minimizing)

brown: terminal state
Alpha-Beta Example

Call Stack

brown: terminal state
Alpha-Beta Example

max

min

min

max

brown: terminal state

Call Stack

A

α=-5

B

α=4, β=-5

C

α=-5, β=3

D

E

α=-5

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

α=-5, β=3

α=-5, β=3

α=-5, β=3
Alpha-Beta Example

alpha(J) = 9
Alpha-Beta Example

J's alpha ≥ C's beta: stop expanding J (beta cutoff)
Alpha-Beta Example

Why? Computer should choose P or better, thus J's lower bound is 9. So smart opponent won't take J:9 since H:3 is worse.
Alpha-Beta Example

beta(C) not changed (minimizing)

Call Stack

brown: terminal state
alpha(A) = 3, updated to maximum seen so far
Alpha-Beta Example

Diagram showing a part of a game tree with nodes labeled with values and alpha-beta pruning.

Call Stack:

- A (max, alpha=3)
- B (max, beta=-5)
  - F (min, alpha=4)
    - G (min, beta=-5)
    - H (max, beta=3)
      - I (min, alpha=9)
        - K (min, beta=-7)
          - S (max, beta=3)
          - T (max, beta=5)
            - U (min, beta=-7)
            - V (max, beta=-9)
  - J (max, beta=3)
    - C (min, beta=-5)
      - O (min, beta=-3)
        - N (max, alpha=4)
          - L (min, alpha=9)
            - M (max, alpha=9)

Brown: terminal state
Alpha-Beta Example

alpha(A) not updated (maximizing)

max

min

max

min

brown : terminal state
How does the algorithm finish the search tree?
Alpha-Beta Example

E's beta $\leq$ A's alpha: stop expanding E (alpha cutoff)
Why? Smart opponent will choose L or worse, thus E's upper bound is 2. So computer shouldn't choose E:2 since C:3 is better path.
Final result: Computer chooses move C