AI SMPS 2024 Week 7 Algorithms

Prepared by S. Baskaran

Game-Play(max)

```
while game not over
    call k-ply search
    make move
    get min's move
```

Minimax(N)

```
if N is a terminal node
    value $\leftarrow$ eval(N)

else if N is a max node
    value $\leftarrow -$large
    for each child C of N
        value $\leftarrow$ max(value, Minimax(C))

else value $\leftarrow +$large
    for each child C of N
        value $\leftarrow$ min(value, Minimax(C))

return value
```

Best-Move(N)

```
bestNode $\leftarrow$ null
bestValue $\leftarrow -$large
for each child C of N
    value $\leftarrow$ Minimax(C)
    if bestValue $<$ value
        bestValue $\leftarrow$ value
        bestNode $\leftarrow$ C
return bestNode
```

Construct-Max-Strategy

```
traverse the tree starting at the root

for max nodes
    choose ONE branch below it

for min nodes
    choose ALL branches below it

return the subtree constructed
```

Construct-Best-Max-Strategy

```
traverse the tree starting at the root

for max nodes
    choose BEST branch below it

for min nodes
    choose ALL branches below it

return the subtree constructed
```

```
ALPHA-BETA(N, $\alpha$, $\beta$)
if N is a terminal node
    return eval(N)

if N is a MAX node
    for each child C of N
        $\alpha \leftarrow$ max( $\alpha$,  ALPHA-BETA(C, $\alpha$, $\beta$) )
        if $\alpha \geq \beta$ then return $\beta$
    return $\alpha$

if N is a MIN node
    for each child C of N
        $\beta \leftarrow$ min( $\beta$,  ALPHA-BETA(C, $\alpha$, $\beta$) )
        if $\alpha \geq \beta$ then return $\alpha$
    return $\beta$
```

```
ALPHA-BETA(N, $\alpha$, $\beta$)                                          ▷ $\alpha < \beta$
if N is a terminal node
    return eval(N)

if N is a MAX node
    for each child C of N
        $\text{eval}_C \leftarrow$ ALPHA-BETA(C, $\alpha$, $\beta$)                  ▷ $\alpha < \beta$
        if $\text{eval}_C \leq \alpha < \beta$ then continue                   ▷ $\alpha < \beta$
        else if $\alpha < \text{eval}_C < \beta$ then $\alpha \leftarrow \text{eval}_C$         ▷ $\alpha < \beta$
        else if $\alpha < \beta \leq \text{eval}_C$ then return $\beta$          ▷ $\alpha < \beta$
    return $\alpha$                                                 ▷ $\alpha < \beta$

if N is a MIN node
    for each child C of N
        $\text{eval}_C \leftarrow$ ALPHA-BETA(C, $\alpha$, $\beta$)                  ▷ $\alpha < \beta$
        if $\alpha < \beta \leq \text{eval}_C$ then continue                   ▷ $\alpha < \beta$
        else if $\alpha < \text{eval}_C < \beta$ then $\beta \leftarrow \text{eval}_C$         ▷ $\alpha < \beta$
        else if $\text{eval}_C \leq \alpha < \beta$ then return $\alpha$          ▷ $\alpha < \beta$
    return $\beta$                                                  ▷ $\alpha < \beta$
```

▷ α and β are local variables.

```
SSS*(root)
OPEN ← empty priority queue
add ( root, LIVE, $\infty$ ) to OPEN
loop
    ( N, status, h ) ← pop top element from OPEN
    if N = root and status is SOLVED
        return h

    if status is LIVE
        if N is a terminal node
            add ( N, SOLVED, min(h, eval(N)) ) to OPEN
        else if N is a MAX node
            for each child C of N
                add ( C, LIVE, h ) to OPEN
        else if N is a MIN node
            add ( first child of N, LIVE, h ) to OPEN

    if status is SOLVED
        P ← parent(N)
        if N is a MAX node and N is the last child
            add ( P, SOLVED, h ) to OPEN
        else if N is a MAX node
            add ( next child of P, LIVE, h ) to OPEN
        else if N is a MIN node
            add ( P, SOLVED, h ) to OPEN
            remove all successors of P from OPEN
```