AI SMPS 2024 Week 7 Algorithms

Prepared by S. Baskaran

GAME-PLAY(MAX)

```
1  while game not over
2      call k-ply search
3      make move
4      get MIN's move
```

MINIMAX(N)

```
1  if N is a terminal node
2      value ← eval(N)
3  else if N is a MAX node
4      value ← −LARGE
5      for each child C of N
6          value ← max(value, MINIMAX(C))
7  else value ← +LARGE
8      for each child C of N
9          value ← min(value, MINIMAX(C))
10 return value
```

BEST-MOVE(N)

```
1  bestNode ← null
2  bestValue ← −LARGE
3  for each child C of N
4      value ← MINIMAX(C)
5      if bestValue < value
6          bestValue ← value
7          bestNode ← C
8  return bestNode
```

CONSTRUCT-MAX-STRATEGY

```
1  traverse the tree starting at the root
2  for MAX nodes
3      choose ONE branch below it
4  for MIN nodes
5      choose ALL branches below it
6  return the subtree constructed
```

CONSTRUCT-BEST-MAX-STRATEGY

```
1  traverse the tree starting at the root
2  for MAX nodes
3      choose BEST branch below it
4  for MIN nodes
5      choose ALL branches below it
6  return the subtree constructed
```

```
ALPHA-BETA(N, α, β)
1  if N is a terminal node
2      return eval(N)
3  if N is a MAX node
4      for each child C of N
5          α ← max( α, ALPHA-BETA(C, α, β) )
6          if α ≥ β then return β
7      return α
8  if N is a MIN node
9      for each child C of N
10         β ← min( β, ALPHA-BETA(C, α, β) )
11         if α ≥ β then return α
12     return β
```

```
ALPHA-BETA(N, α, β)                                          ▷ α < β
1  if N is a terminal node
2      return eval(N)
3  if N is a MAX node
4      for each child C of N
5          eval_C ← ALPHA-BETA(C, α, β)                      ▷ α < β
6          if eval_C ≤ α < β then continue                   ▷ α < β
7          else if α < eval_C < β then α ← eval_C            ▷ α < β
8          else if α < β ≤ eval_C then return β              ▷ α < β
9      return α                                              ▷ α < β
10 if N is a MIN node
11     for each child C of N
12         eval_C ← ALPHA-BETA(C, α, β)                      ▷ α < β
13         if α < β ≤ eval_C then continue                   ▷ α < β
14         else if α < eval_C < β then β ← eval_C            ▷ α < β
15         else if eval_C ≤ α < β then return α              ▷ α < β
16     return β                                              ▷ α < β
```

▷ α and β are local variables.

SSS*(root)

```
1  OPEN ← empty priority queue
2  add ( root, LIVE, ∞ ) to OPEN
3  loop
4       ( N, status, h ) ← pop top element from OPEN
5       if N = root and status is SOLVED
6            return h

7       if status is LIVE
8            if N is a terminal node
9                 add ( N, SOLVED, min(h, eval(N)) ) to OPEN
10           else if N is a MAX node
11                for each child C of N
12                     add ( C, LIVE, h ) to OPEN
13           else if N is a MIN node
14                add ( first child of N, LIVE, h ) to OPEN

15      if status is SOLVED
16           P ← parent(N)
17           if N is a MAX node and N is the last child
18                add ( P, SOLVED, h ) to OPEN
19           else if N is a MAX node
20                add ( next child of P, LIVE, h ) to OPEN
21           else if N is a MIN node
22                add ( P, SOLVED, h ) to OPEN
23                remove all successors of P from OPEN
```