Game Playing
Summary

Yingyu Liang

yliang@cs.wisc.edu

Computer Sciences Department

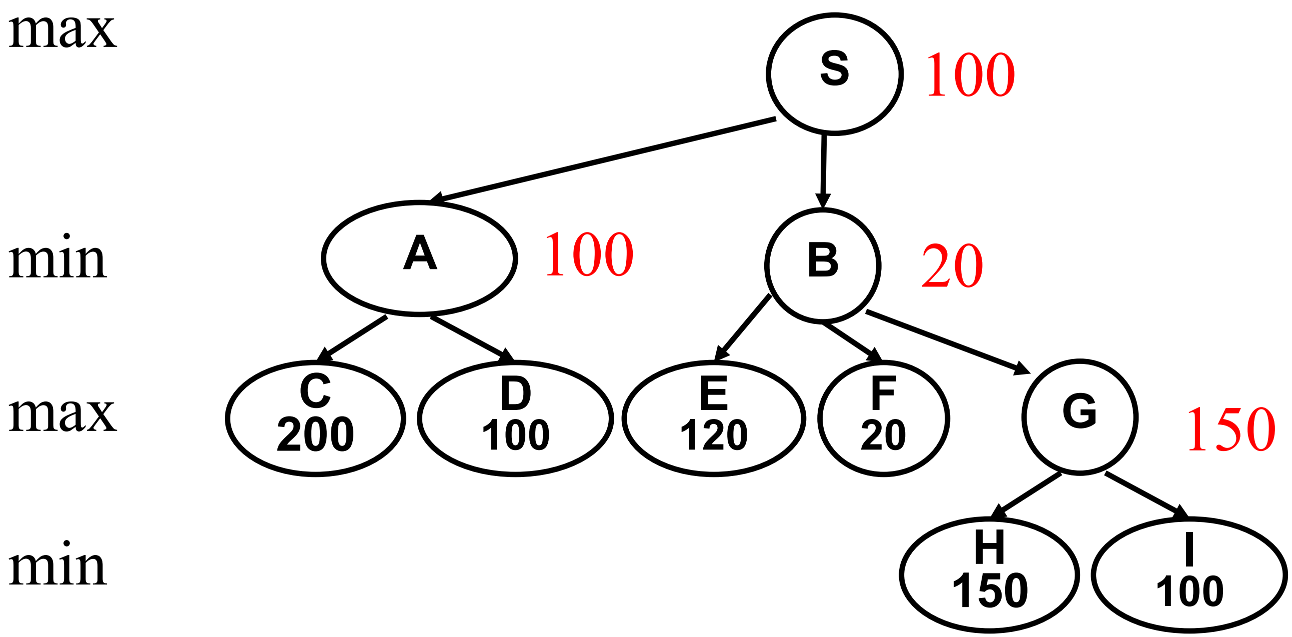
University of Wisconsin, Madison

[based on slides from A. Moore, C. Dyer, J. Skrentny, Jerry Zhu]

The Kind of Games Focused

- Two-player
- Zero-sum
- Discrete finite
- Deterministic
- Perfect information

Key Concept: Game Theoretic Value



The bottom-up approach: needs the whole game tree.
The minimax algorithm: a variant of DFS to save space.

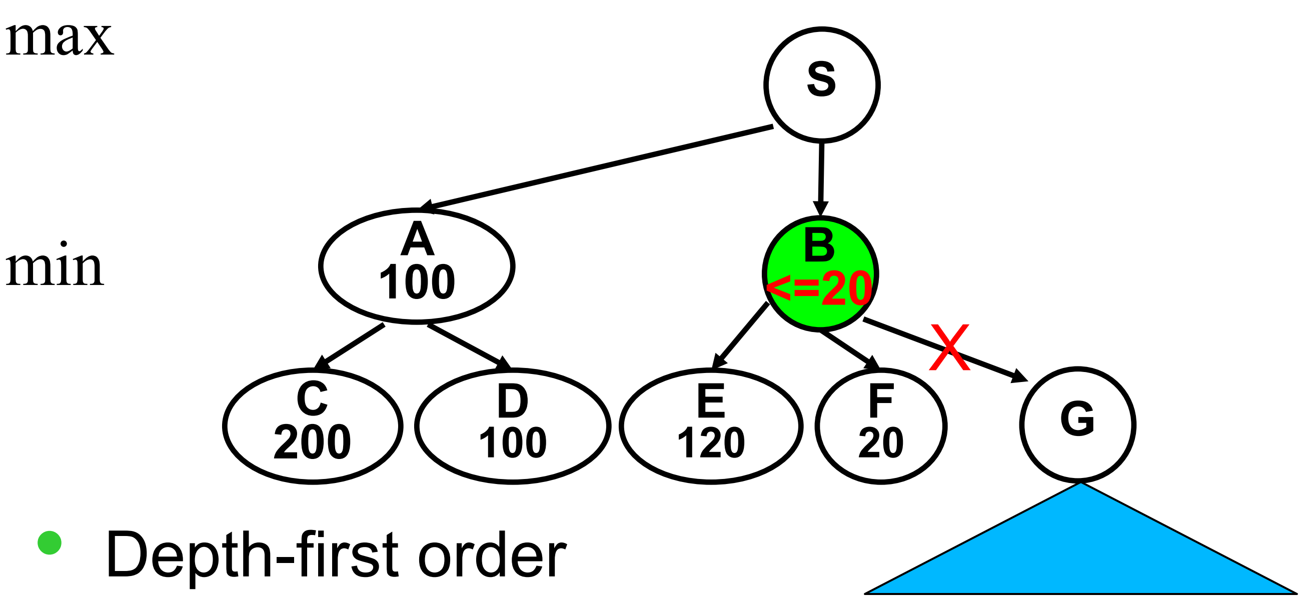
Key Algorithm: Minimax algorithm

```
function Max-Value(s)
inputs:
    s: current state in game, Max about to play
output: best-score (for Max) available from s

    if ( s is a terminal state )
    then return ( terminal value of s )
    else
            α := – ∞
            for each s' in Succ(s)
                α := max( α , Min-value(s'))
    return α
```

```
function Min-Value(s)
output: best-score (for Min) available from s

    if ( s is a terminal state )
    then return ( terminal value of s)
    else
            β := ∞
            for each s' in Succs(s)
                β := min( β , Max-value(s'))
    return β
```

- Time complexity $O(b^m)$
- Space complexity $O(bm)$

Alpha-Beta Pruning Intuition



- Depth-first order
- After returning from A, Max can get at least 100 at S
- After returning from F, Max can get at most 20 at B
- At this point, Max losts interest in B
- There is no need to explore G. The subtree at G is pruned. Saves time.

Key Algorithm: Alpha-beta pruning

```
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
output: min(β , best-score (for Max) available from s)

    if ( s is a terminal state )
    then return ( terminal value of s )
    else for each s' in Succ(s)
      α := max( α , Min-value(s',α,β))
      if ( α ≥ β ) then return β   /* alpha pruning */
    return α
```

```
function Min-Value(s,α,β)
output: max(α , best-score (for Min) available from s )

    if ( s is a terminal state )
    then return ( terminal value of s)
    else for each s' in Succs(s)
      β := min( β , Max-value(s',α,β))
      if (α ≥ β ) then return α   /* beta pruning */
    return β
```

Starting from the root:
Max-Value(root, $-\infty$, $+\infty$)