FAI lecture 6: game trees

overview

"solved": the machine could always win

contingent plan, recommending a mvoe for every possible eventuality

- Game formulation:
  - Initial state: s_0
  - Players: Player(s) indicates whose move it is
  - Actions: Actions(s) for player on move
  - Transition model: Result(s,a)
  - Terminal test: Terminal-Test(s)
  - Terminal values: Utility(s,p) for player p
    - Or just Utility(s) for player making the decision at root

zero-sum

- opposite utilities:

  pure competision (one maximizes while the other minimizes)

- as comparison to general games:

  agents have independent utilities

(Single-agent) value of a state = the best achievable outcome (utility) from that state

minimax value:

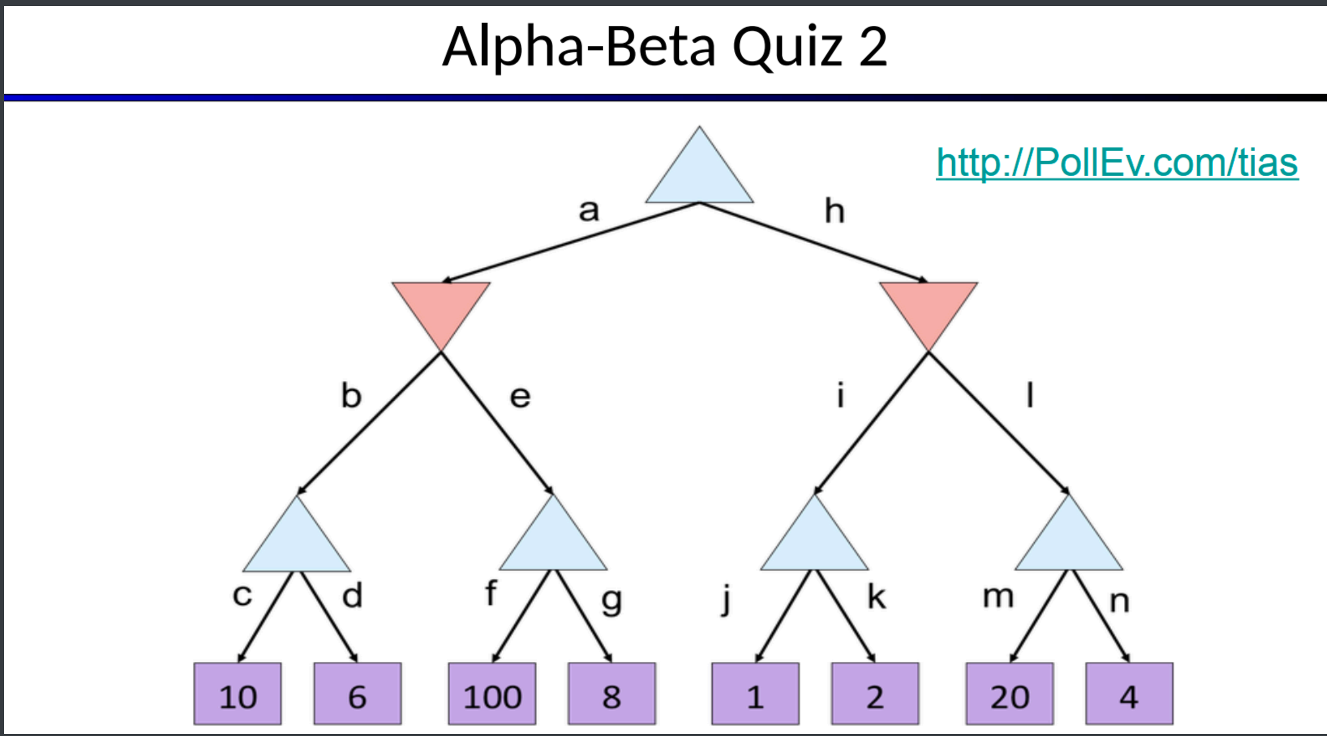
- MAX nodes: under agent's control, maximizing state value
- MIN nodes: under opponent's control, minimizing state value

game tree pruning

Alpha = best option so far fron any MAX node on path to root

! the order of generation matters !

- Alpha-beta pruning



alpha = MAX's best option on path to root

beta   = MIN's best option on path to root

Alpha-Beta Implementation

α: MAX's best option on path to root
β: MIN's best option on path to root

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β
            return v
        α = max(α, v)
    return v
```

```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α
            return v
        β = min(β, v)
    return v
```

- **Theorem**: no effect on minimax value computed for the root
- **time complexity**: ordering matters

  with perfect ordering time complexity ~ $O(b^{m/2})$

  (roughly half tiers)
- **optimal**?: yes

resource limits

- Cannot search to leaves in most scenarios
- bounded lookahead: depth limit

Example:
- Suppose we have 100 seconds, can explore 10K nodes / sec
- So can check 1M nodes per move
- Chess with alpha-beta, $35^{(8/2)}$ =~ 1M; depth 8 is good

- evaluation functions: for non-terminal positions

  typically weighted linear sum of features (num white queens - num black queens)

  or a more complex nonlinear function

  terminate search in quiescent positions