Adversarial Search

Chapter 6

Section 1 – 4

Outline

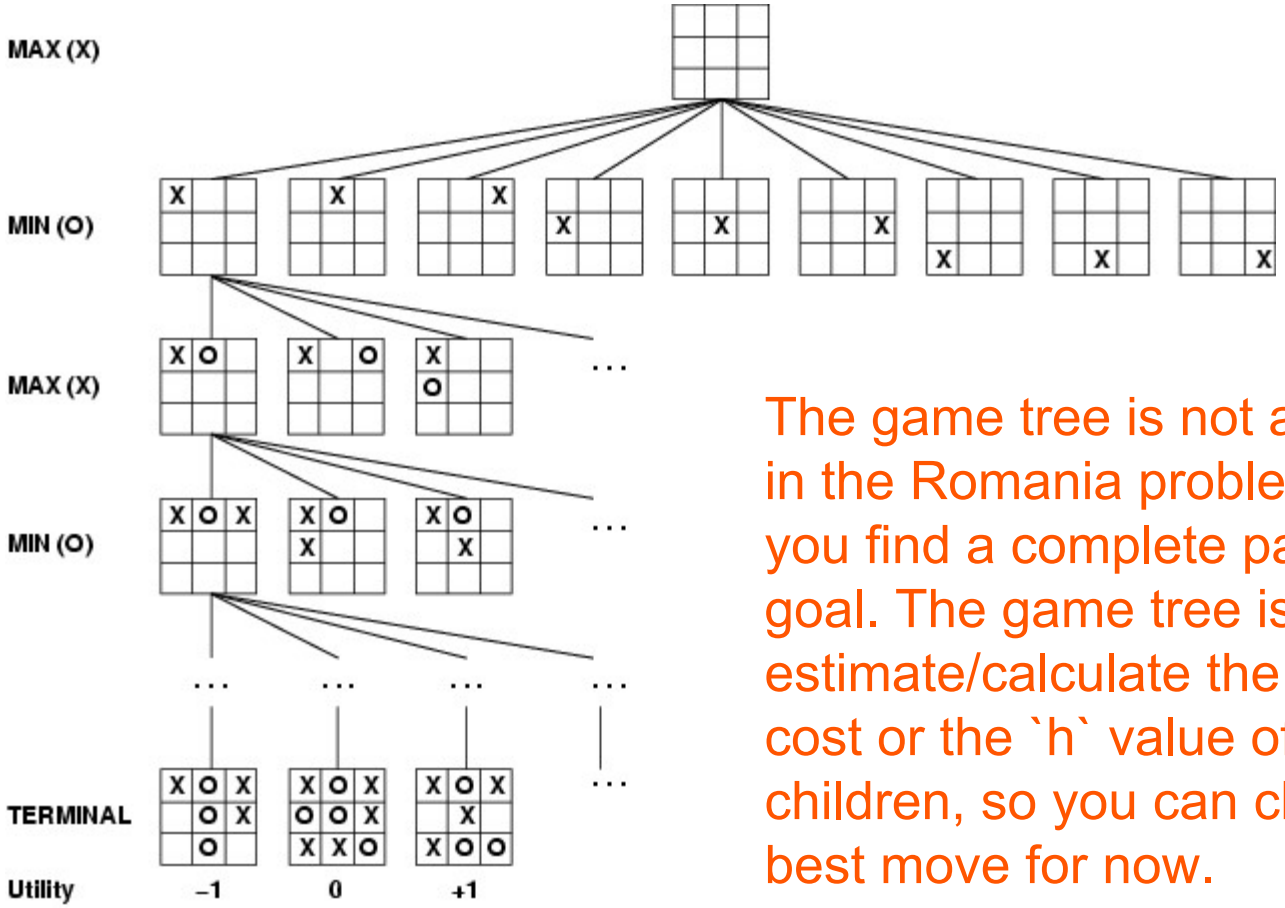
- Optimal decisions
- α-β pruning
- Imperfect, real-time decisions

Games vs. search problems

- "Unpredictable" opponent → specifying a move for every possible opponent reply
-
- Time limits → unlikely to find goal, must approximate
-

In Romania problem, you need to find a complete PATH to the goal; in a gaming problem, you simply cannot find a complete PATH to win since you can only move one step and wait for your opponent's move; so the decision is really about ONE move.

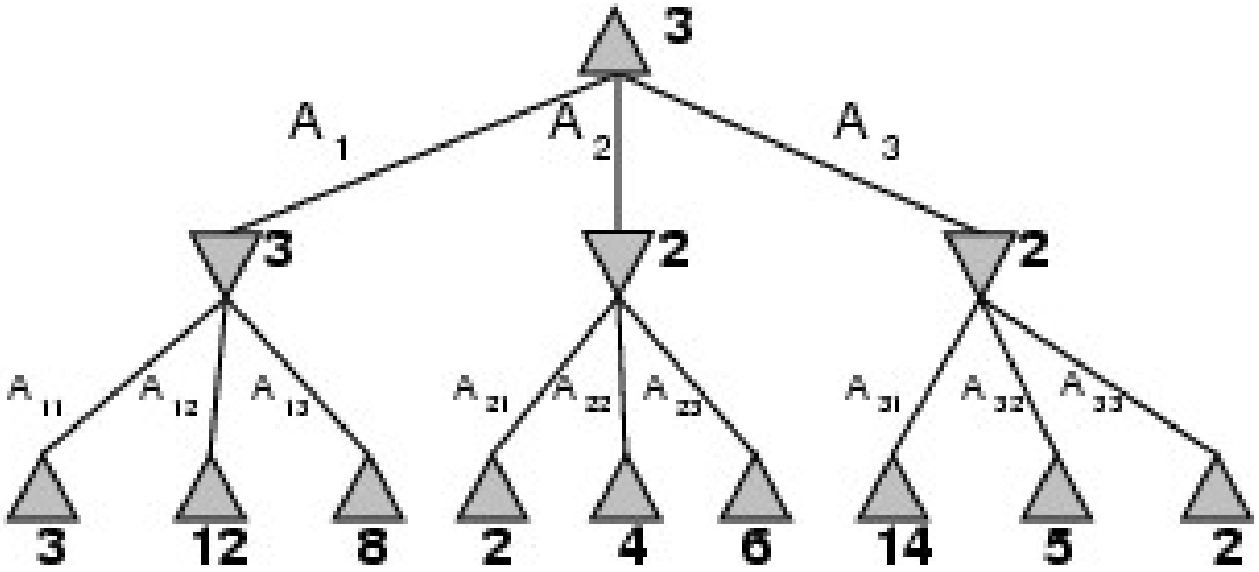
Game tree (2-player, deterministic, turns)



The game tree is not a search tree in the Romania problem, where you find a complete path to the goal. The game tree is a device to estimate/calculate the forward-cost or the `h` value of each direct children, so you can choose the best move for now.

Minimax

- Perfect play for deterministic games
-
- Idea: choose move to position with highest minimax value
  = best achievable payoff against best play
- MAX
- E
- MIN



```
def pow(x,n): #n is a positive integer
    return x if n==1 else x*pow(x,n-1)
pow(x,3)=x*pow(x,2)=x*x*pow(x,1)=x*x*x
```

Minimax algorithm

function MINIMAX-DECISION(*state*) **returns** *an action*

  $v \leftarrow$ MAX-VALUE(*state*)
  return the *action* in SUCCESSORS(*state*) with value v

---

function MAX-VALUE(*state*) **returns** *a utility value* i.e., the terminal reward

  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(s))
  return v

---

function MIN-VALUE(*state*) **returns** *a utility value*

  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow \infty$
  for a, s in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(s))
  return v

```
def successors(s):
    return [(a,result(s,a)) for a in actions(s) ]
```

`actions`: return the set of actions for the input state s

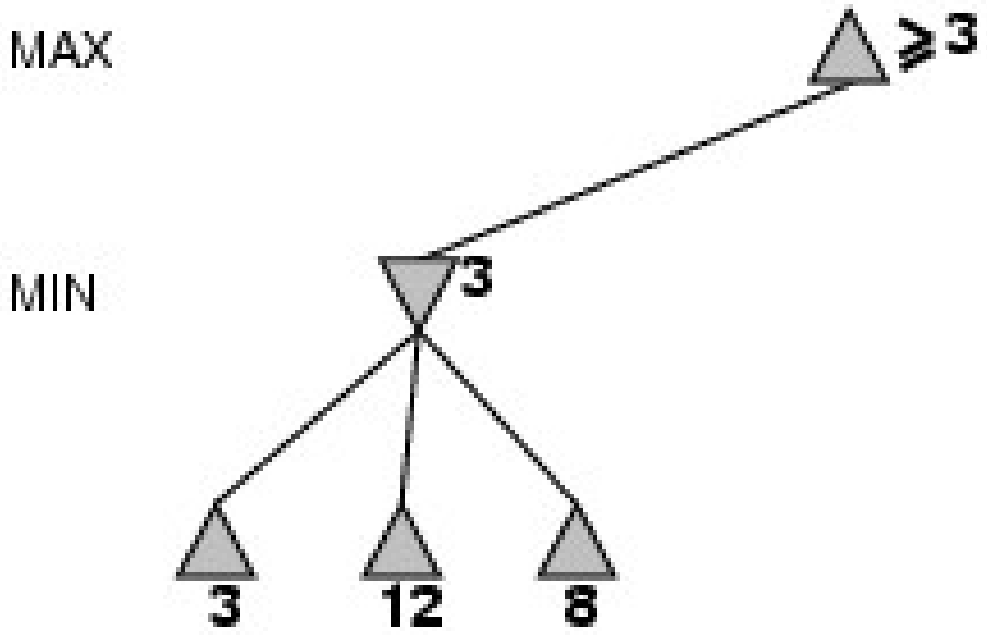
`result`: return the new state as a result of taking action `a` at a state `s`

Properties of minimax

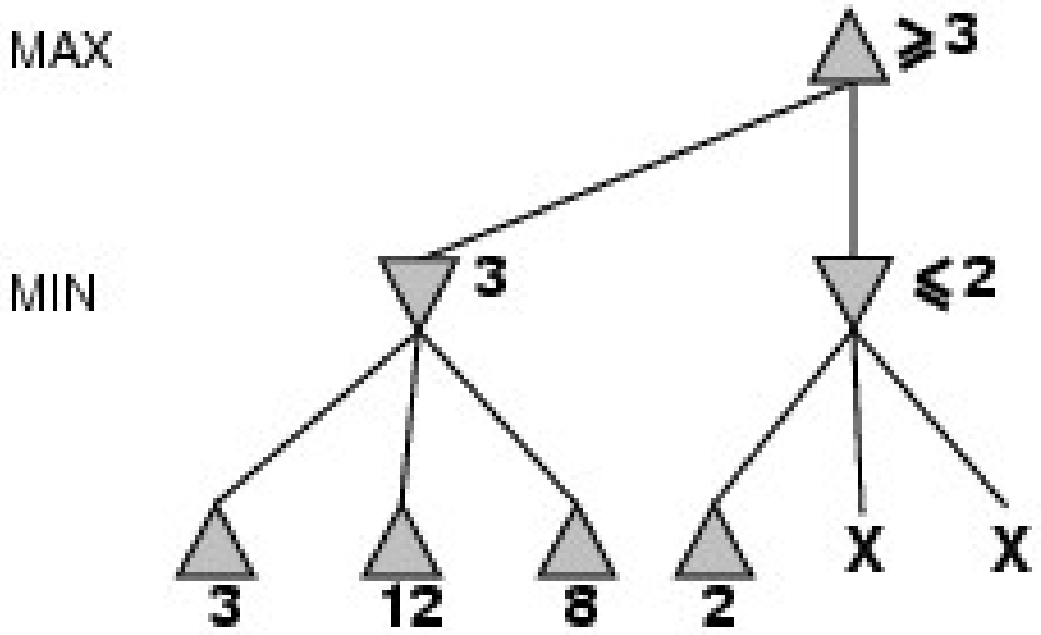
- Complete? Yes (if tree is finite)
-
- Optimal? Yes (against an optimal opponent)
-
- Time complexity? $O(b^m)$
-
- Space complexity? O(bm) (depth-first exploration)
-

- For chess, b ≈ 35, m ≈100 for "reasonable" games
  → exact solution completely infeasible
-

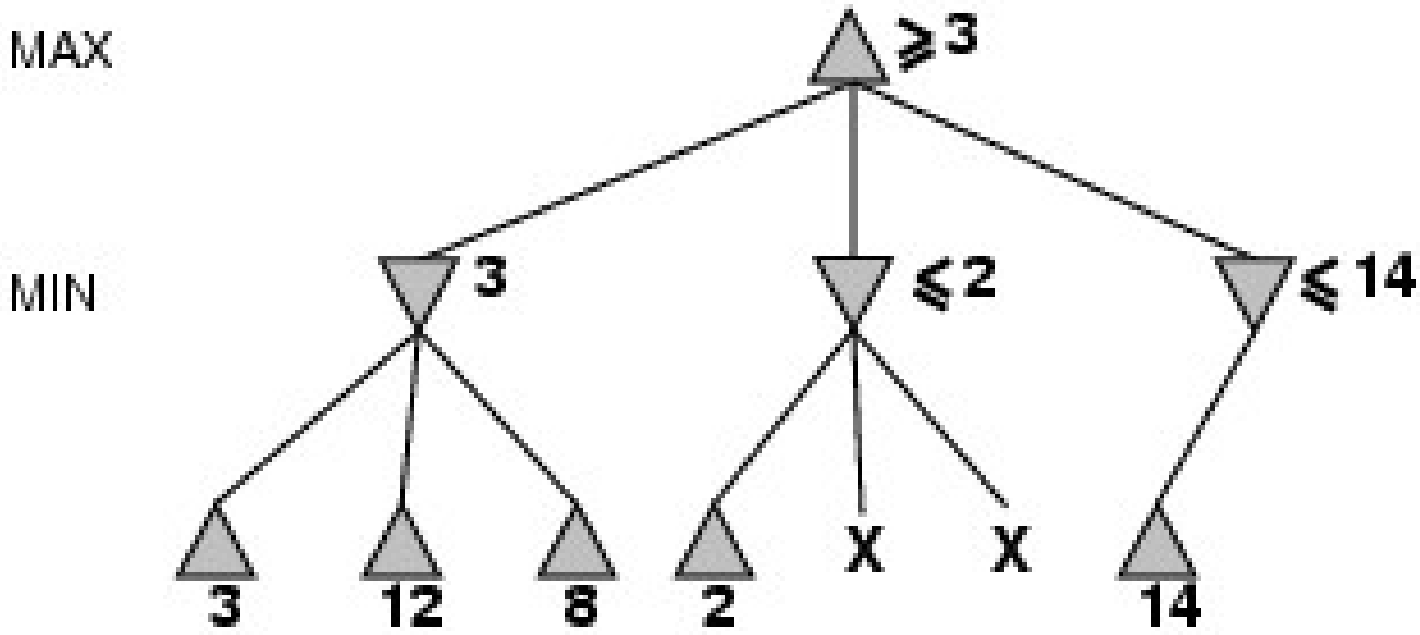
α-β pruning example



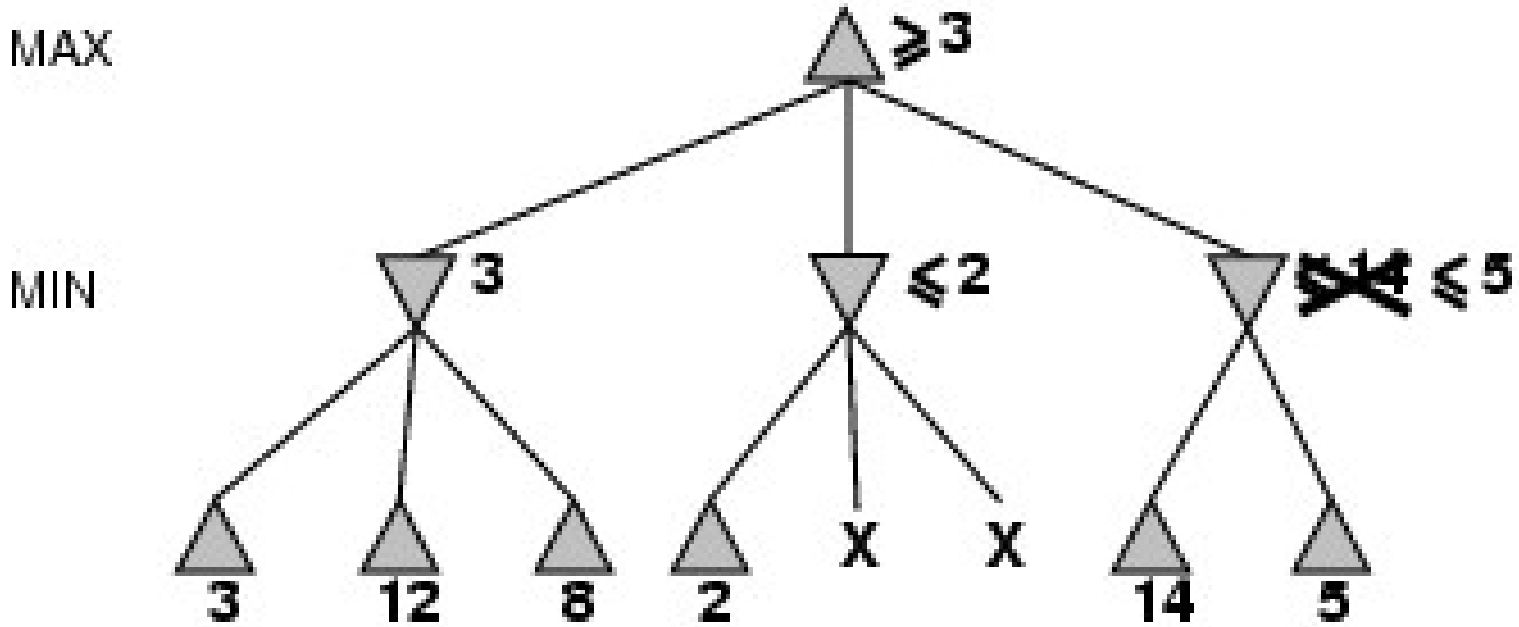
α-β pruning example



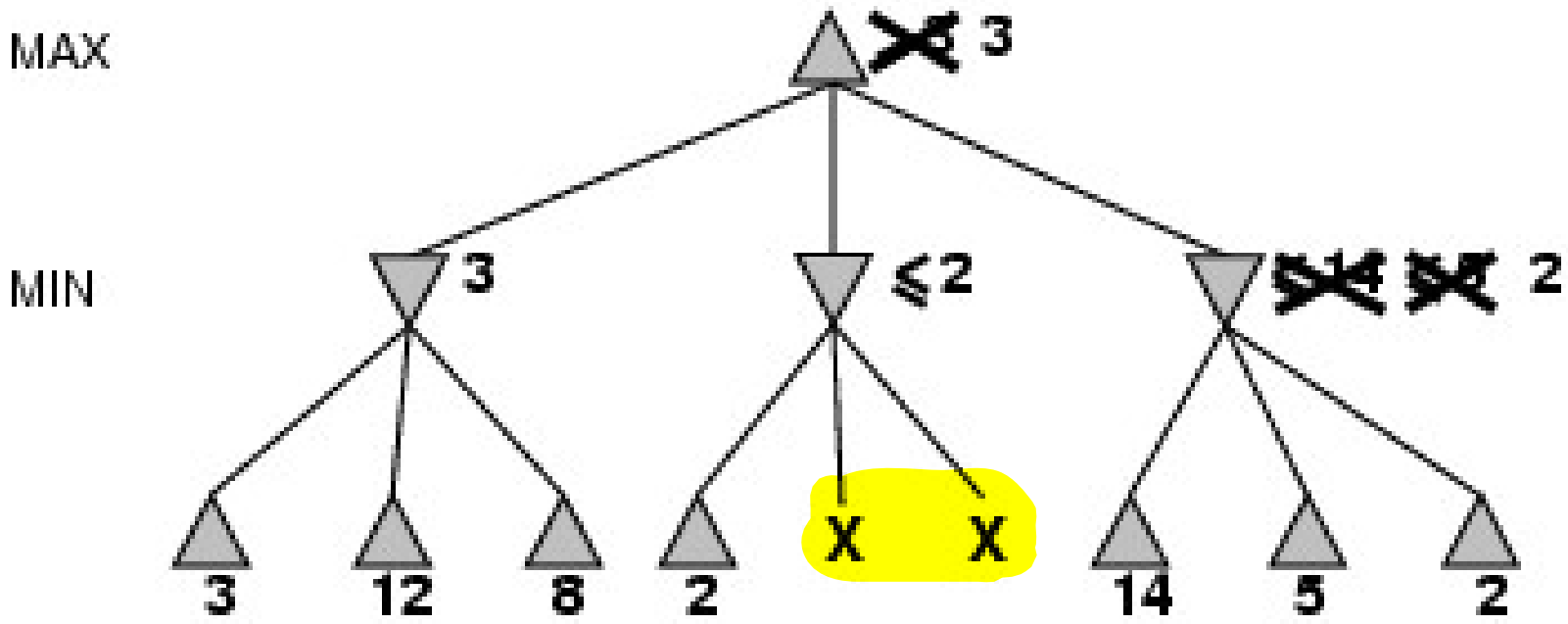
α-β pruning example



α-β pruning example



α-β pruning example



No need to try the rest actions for MIN since MAX won't choose this branch at all.

Since MAX sees MIN could play this nasty action `2`, MAX won't court danger at all. Hence it is a waste of time to assume MAX will choose this middle branch, and explore the full set responses of MIN.

Properties of α-β

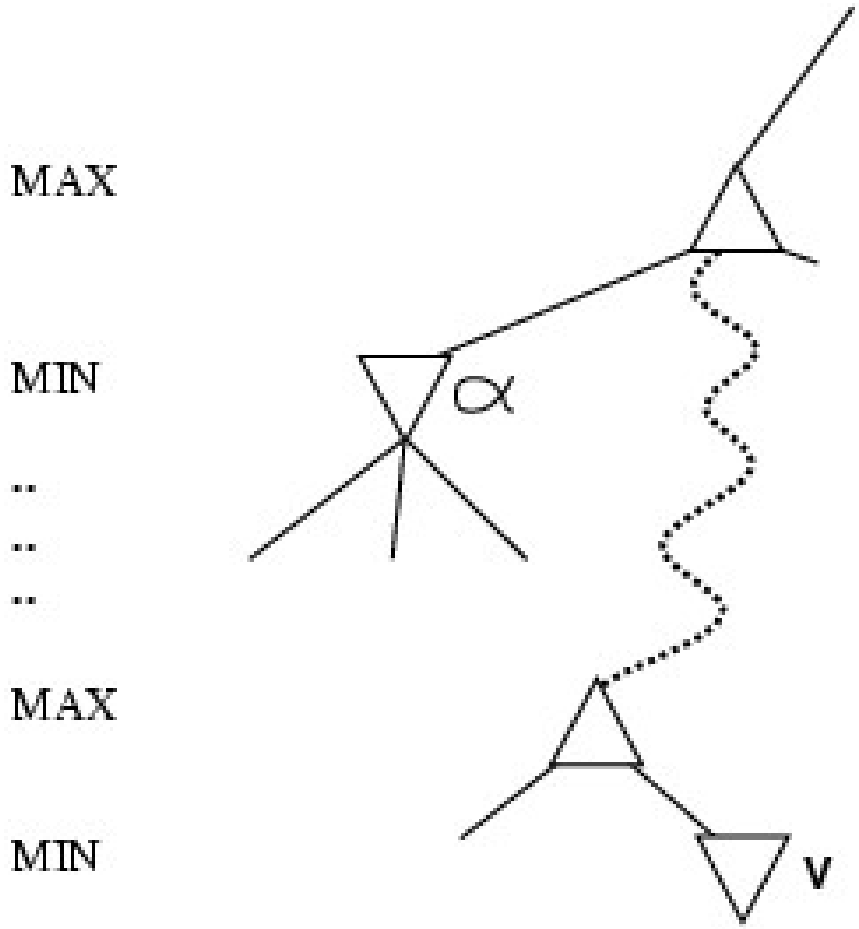
- Pruning does not affect final result
-
- Good move ordering improves effectiveness of pruning
-
- With "perfect ordering," time complexity = $O(b^{m/2})$
  → doubles depth of search
- A simple example of the value of reasoning about which computations are relevant (a form of metareasoning)
-

Why is it called α-β?

- α is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for *max*
-
- If *v* is worse than α, *max* will avoid it
-

  → prune that branch

- Define β similarly for



The α-β algorithm

```
function ALPHA-BETA-SEARCH(state) returns an action
    inputs: state, current state in game

    v ← MAX-VALUE(state, −∞, +∞)
    return the action in SUCCESSORS(state) with value v

---

function MAX-VALUE(state, α, β) returns a utility value
    inputs: state, current state in game
            α, the value of the best alternative for MAX along the path to state
            β, the value of the best alternative for MIN along the path to state

    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s, α, β))
        if v ≥ β then return v
        α ← MAX(α, v)
    return v
```

No need to try the rest of successors anymore
since MIN won't choose this branch at all in the previous step.

The α-β algorithm

```
function MIN-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for MAX along the path to state
          β, the value of the best alternative for MIN along the path to state

  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for a, s in SUCCESSORS(state) do
    v ← MIN(v, MAX-VALUE(s, α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

No need to try the rest since MAX won't choose this branch at all in the previous step since MAX know MIN will play this nasty `a`

Resource limits

Suppose we have 100 secs, explore 10^4 nodes/sec

→ 10^6 nodes per move

Standard approach:

- cutoff test:

  e.g., depth limit (perhaps add quiescence search)

- evaluation function

Evaluation functions

- For chess, typically linear weighted sum of features

$$\mathit{Eval}(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

- e.g., $w_1 = 9$ with

  $f_1(s)$ = (number of white queens) – (number of black queens), etc.

Cutting off search

MinimaxCutoff is identical to *MinimaxValue* except

1. *Terminal?* is replaced by *Cutoff?*
2. *Utility* is replaced by *Eval*
3.

Does it work in practice?

$b^m = 10^6$, b=35 → m=4

4-ply lookahead is a hopeless chess player!

– 4-ply ≈ human novice

Deterministic games in practice

- Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used a precomputed endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 444 billion positions.
  »
  »
- Chess: Deep Blue defeated human world champion Garry Kasparov in a six-game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.
-
- Othello: human champions refuse to compete against computers, who are too good.
-
- Go: human champions refuse to compete against computers, who are too bad. In go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves.
-

Summary

- Games are fun to work on!
-
- They illustrate several important points about AI
-
- perfection is unattainable → must approximate
- good idea to think about what to think about