Can we do better?

max

min
\(\alpha-\beta\) Pruning Example
α-β Pruning

- **General configuration**
  - α is the best value that MAX can get at any choice point along the current path
  - If n becomes worse than α, MAX will avoid it, so can stop considering n’s other children
  - Define β similarly for MIN
Alpha-Beta Pruning Example

α is MAX’s best alternative here or above
β is MIN’s best alternative here or above
Alpha-Beta Pseudocode

**inputs:** \( state \), current game state
\( \alpha \), value of best alternative for MAX on path to \( state \)
\( \beta \), value of best alternative for MIN on path to \( state \)

**returns:** a utility value

**function** MAX-VALUE\((state, \alpha, \beta)\)

- if TERMINAL-TEST\((state)\) then
  - return UTILITY\((state)\)
- \( v \leftarrow -\infty \)
- for \( a, s \) in SUCCESSORS\((state)\) do
  - \( v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s, \alpha, \beta)) \)
  - if \( v \geq \beta \) then return \( v \)
  - \( \alpha \leftarrow \text{MAX}(\alpha, v) \)
- return \( v \)

**function** MIN-VALUE\((state, \alpha, \beta)\)

- if TERMINAL-TEST\((state)\) then
  - return UTILITY\((state)\)
- \( v \leftarrow +\infty \)
- for \( a, s \) in SUCCESSORS\((state)\) do
  - \( v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s, \alpha, \beta)) \)
  - if \( v \leq \alpha \) then return \( v \)
  - \( \beta \leftarrow \text{MIN}(\beta, v) \)
- return \( v \)
Alpha-Beta Pruning Example

α is MAX’s best alternative here or above
β is MIN’s best alternative here or above
α is MAX’s best alternative here or above
β is MIN’s best alternative here or above
Alpha-Beta Pruning Properties

- This pruning has no effect on final result at the root.
- Values of intermediate nodes might be wrong!
  - but, they are bounds
- Good child ordering improves effectiveness of pruning
- With “perfect ordering”:
  - Time complexity drops to $O(b^{m/2})$
  - Doubles solvable depth!
  - Full search of, e.g. chess, is still hopeless…
Resource Limits

- Cannot search to leaves
- Depth-limited search
  - Instead, search a limited depth of tree
  - Replace terminal utilities with an eval function for non-terminal positions
    - e.g., $\alpha-\beta$ reaches about depth 8 – decent chess program
- Guarantee of optimal play is gone
- Evaluation function matters
  - It works better when we have a greater depth look ahead
Depth Matters

depth 2
Depth Matters

[Diagram of Pac-Man game with a score of 0 and depth 10]
Evaluation Functions

- Function which scores non-terminals

- Ideal function: returns the utility of the position
- In practice: typically weighted linear sum of features:
  - e.g. $f_1(s) = (\text{num white queens} - \text{num black queens})$, etc.
What features would be good for Pacman?

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$
Evaluation Function
Evaluation Function
Bad Evaluation Function
Why Pacman Starves

- He knows his score will go up by eating the dot now
- He knows his score will go up just as much by eating the dot later on
- There are no point-scoring opportunities after eating the dot
- Therefore, waiting seems just as good as eating
Which algorithm?

$\alpha - \beta$, depth 4, simple eval fun
Which algorithm?

α-β, depth 4, better eval fun
Minimax Example

Suicidal agent
Expectimax

- Uncertain outcomes are controlled by chance not an adversary
- Chance nodes are new types of nodes (instead of Min nodes)