Section 8: Minimax & Alpha Beta Pruning

CSE 332 19Au

University of Washington

November 14, 2019

A Game algorithm: Minimax

Some backgrounds on the game

- Let's assume that our opponent plays optimally
- Let's assume that we evaluate the game using **positive** values, and opponent does so using **negative** values (zero-sum)

A Game algorithm: Minimax

Some backgrounds on the game

- Let's assume that our opponent plays optimally
- Let's assume that we evaluate the game using **positive** values, and opponent does so using **negative** values (zero-sum)

Game strategies

- My gain is my opponent's loss (and vice versa)
- If the position value is 50 for me, it should be -50 for my opponent.
- If I reach $+\infty$, I win; if my opponent reaches $-\infty$, he/she wins.
- So I want to **MAXIMIZE** my score, while my opponent wants to **MINIMIZE** the score
- Thus, Minimax.

Introducing the players!

For the following slides, assume:

- It's blue's turn!
- **MIN** wants to minimize the value
- **MAX** wants to maximize the value

Minimax, Code


```
int minimax(Position p, boolean is_max) {
  if (p is a leaf) {
    // always position value of MAX
    return p.evaluate();
  }
  if (is_max) { // MAX
    int bestValue = −∞
    for (move in p.getMoves()) {
      p.applyMove();
      int value = minimax(p, is_max);
      p.undoMove();
      if (value > bestValue) {
        bestValue = value;
      }
    }
  } else { // MIN
    int bestValue = ∞
    for (move in p.getMoves()) {
      p.applyMove();
      int value = minimax(p, is_max);
      p.undoMove();
      if (value < bestValue) {
        bestValue = value;
      }
    }
  }
}
```

The highlighted parts are the only differences!

How do we simplify Minimax?

A fact

$$\max(a, b) = -\min(-a, -b)$$

How do we simplify Minimax?

A fact

$$\max(a, b) = -\min(-a, -b)$$

Change Minimax Code

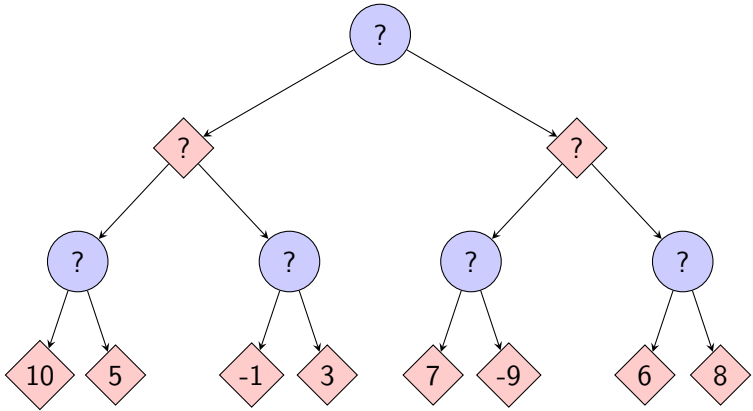
Then...

- For MAX player's turn, we negate the negative values returned by MIN, and find max
- For MIN player's turn, we negate the positive values returned by MAX, and find max, which is equivalent to find min.
- Now both players are maximizing, we can use the same piece of code.

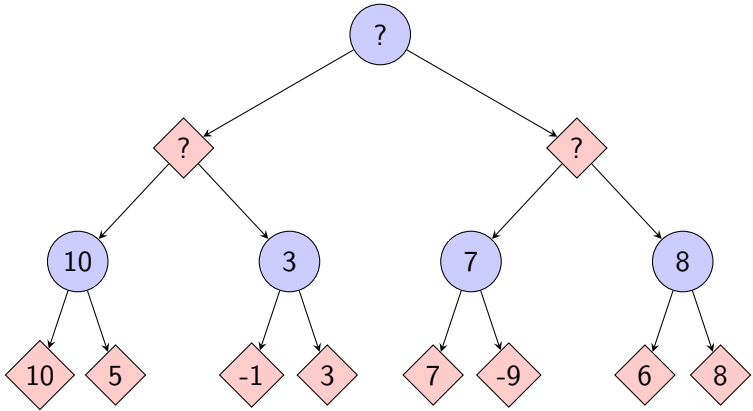
Code from your Game handout:

```
int minimax(Position p) {
  if (p is a leaf) {
    // position value of current player
    return p.evaluate();
  }
  int bestValue = −∞
  for (move in p.getMoves()) {
    p.applyMove();
    int value = -minimax(p);
    p.undoMove();
    if (value > bestValue) {
      bestValue = value;
    }
  }
}
```

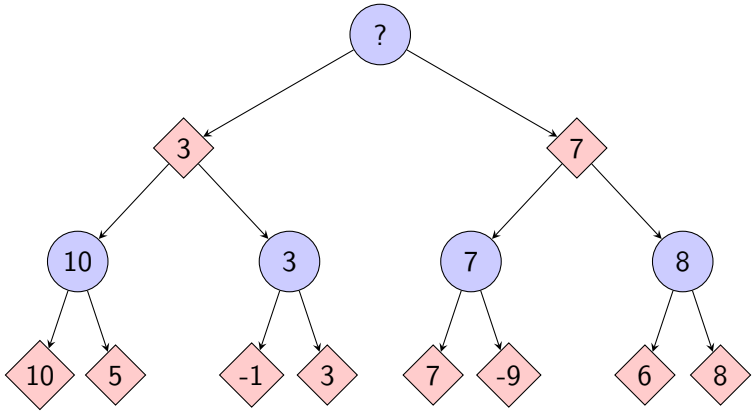
Minimax Example



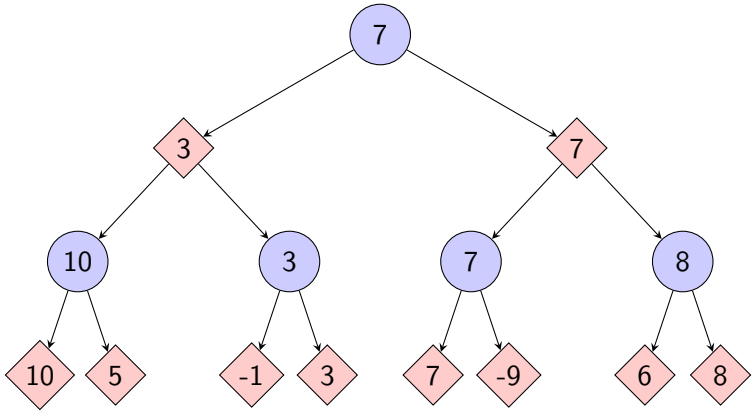
Minimax Example



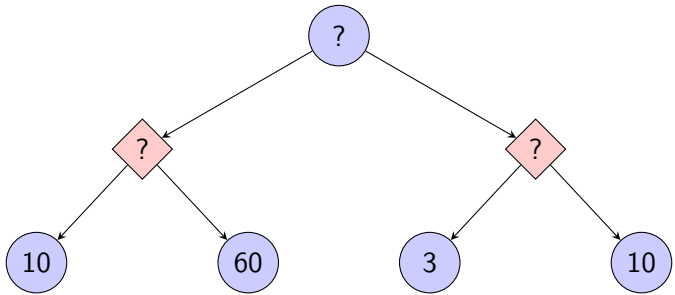
Minimax Example



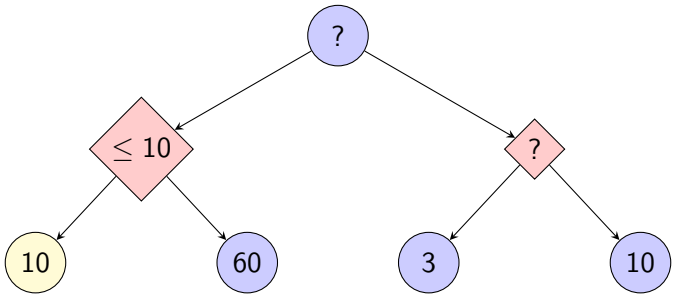
Minimax Example



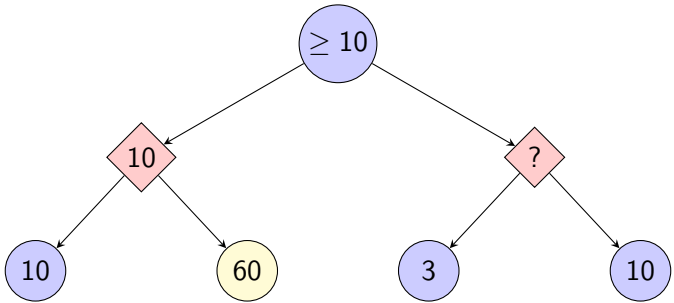
Did we need to look at every leaf node?



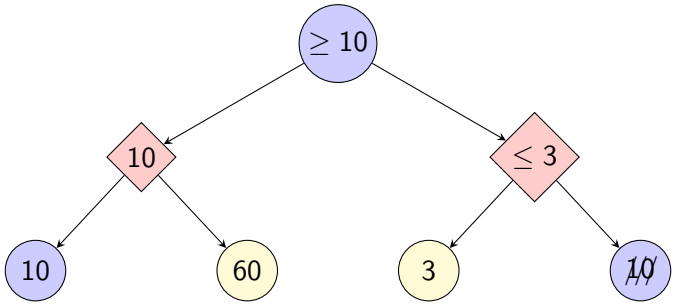
Did we need to look at every leaf node?



Did we need to look at every leaf node?



Did we need to look at every leaf node?



Now, without looking at 10, we know that the minimizer will give a score that is ≤ 3, yet the root maximizer already has a ≥ 10 guarantee. So we don't need to look at 10 really.

To formalize this process...

Alpha beta pruning

We are going to use two helper values:

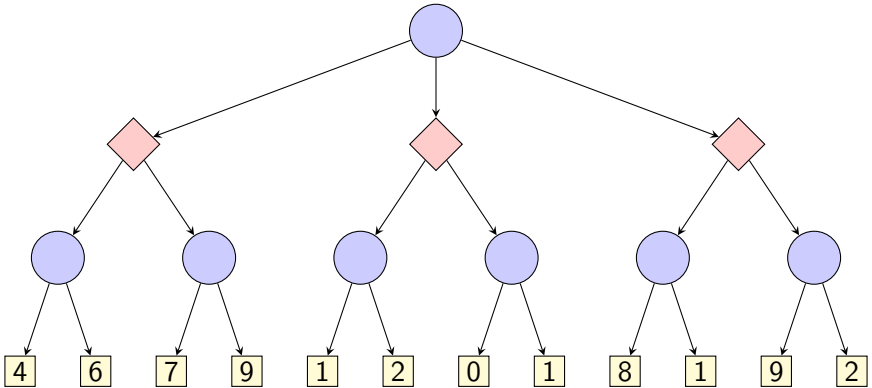
- α: best option **along the path to the root** for MAX
- β: best option **along the path to the root** for MIN

Pruning when...

- If the value of a MAX node is larger than β, or
- if the value of a MIN node is smaller than α

Overall, this means when α is larger than β, we prune the children of the current node.

Alpha beta demo[1]



[1] https://www.youtube.com/watch?v=xBXHtz4Gbdo&t=614s