Deep Blue - search algorithms

Prof. Pushpak Bhattacharya
Group 2: Neel Shah, Pallav Vasa and Roshan Prizak

CS 621: Artificial Intelligence

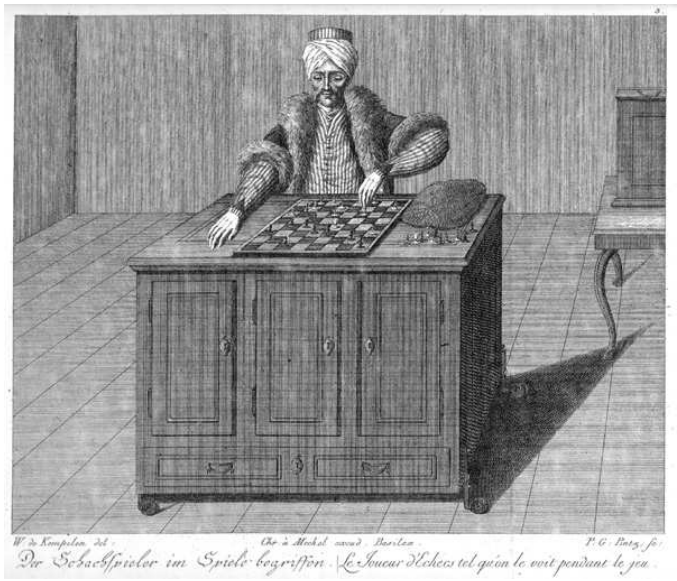
November 14, 2011

Outline

- Motivation and History
- Deep Blue
- Game trees and the Minimax algorithm
- Alpha Beta pruning and Iterative deepening
- Chess Complexities

The roots

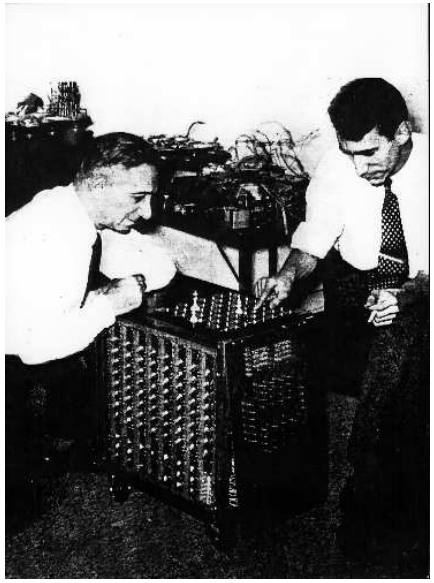
- Carl Sagan once said, "Exploration is in our nature. We began as wanderers, and we are wanderers still." The Enlightenment revived an interest in automaton



Borrowed from [Wikipedia]

The foundation

- Claude Shannon wrote "Programming a computer for playing chess" in 1950
- Ideas from that paper still used today



Borrowed from [Computer History Museum], Shannon 1950

The prediction

- Herbert Simon, in 1957, predicted that in 10 years, a computer would defeat a human champion in chess

Borrowed from [Computer History Museum]

The controversial win

- The prophecy turned true but delayed only by 30 more years
- After the world saw many chess machines and programs, finally Deep Blue defeated Gary Kasparov in 1997
- Primary reasons: highly specialized hardware, massive parallel processing abilities and **some good chess-intuition in the search algorithms**
- Controversial, but an inspiration for excellent computer chess
- The Big Question: Is Deep Blue intelligent? Or at least, chess-intelligent?

Deep Blue

- Descendant of Deep Thought, which was named after the computer in the Hitchhikers Guide to the Galaxy which comes up with the answer to Life, the Universe and Everything
- A parallel, RS/6000 SP-Based computer designed to play chess at grandmaster level
- 30 such processors meant for software search and 480 custom VLSI chess processors that performed move generation (including move ordering) hardware search
- Searches 126 million nodes per second at an average, with peak of 330 million nodes per second

---

information retrieved from [IBM Research]

Deep Blue

- A opening book of about 4000 positions was used, as well as a database of 700,000 grandmaster games from which consensus recommendations could be extracted
- **Iterative deepening alpha-beta minimax algorithm with transposition tables, singular extensions and other chess-specific heuristics** and a historical database of opening and end games was used
- Generates 30 billion positions per move with average depth of 14 routinely

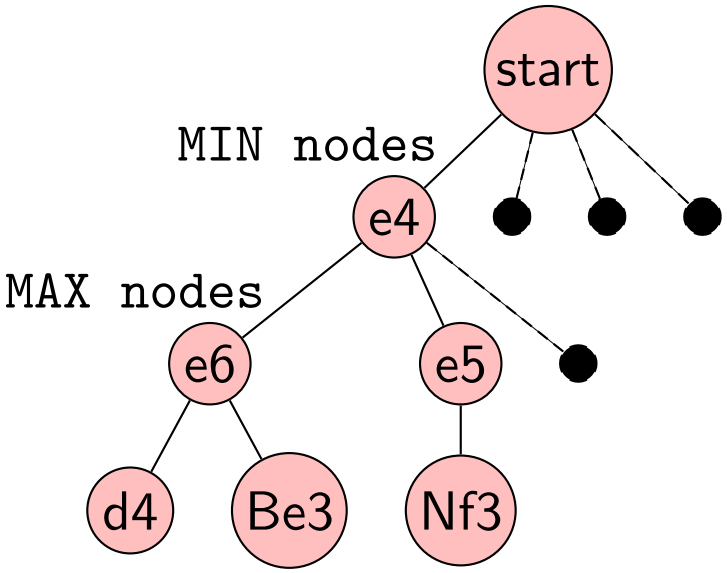
Adversarial search and chess

- Generic problem: plan ahead in an environment consisting of other agents (rational and selfish: adversarial) actively planning against us
- Success of my move depends on the opponent's move and so on
- Need to look ahead and consider various alternatives at deeper 'plies'
- Zero-sum: minimize opponent's payoff iff maximize my payoff
- Underlying philosophy - prepare for the worst

Game (2 player) - formal definition

- A : finite set of states of the game - chess: a particular setting on the board
- S : nonempty set of start states ($\subset A$) - starting position of chess
- E : nonempty set of end states ($\subset A$) - any checkmate or stalemate or other position which is a win/draw
- $f : A \rightarrow 2^A$ - transition function, $f(x)$ is the set of possible states from state x
- $v : E \rightarrow [-\infty, \infty]^2$, payoff/utility function of both players at the end state - if I win, I get $+\infty$ and the opponent gets $-\infty$

Chess game tree



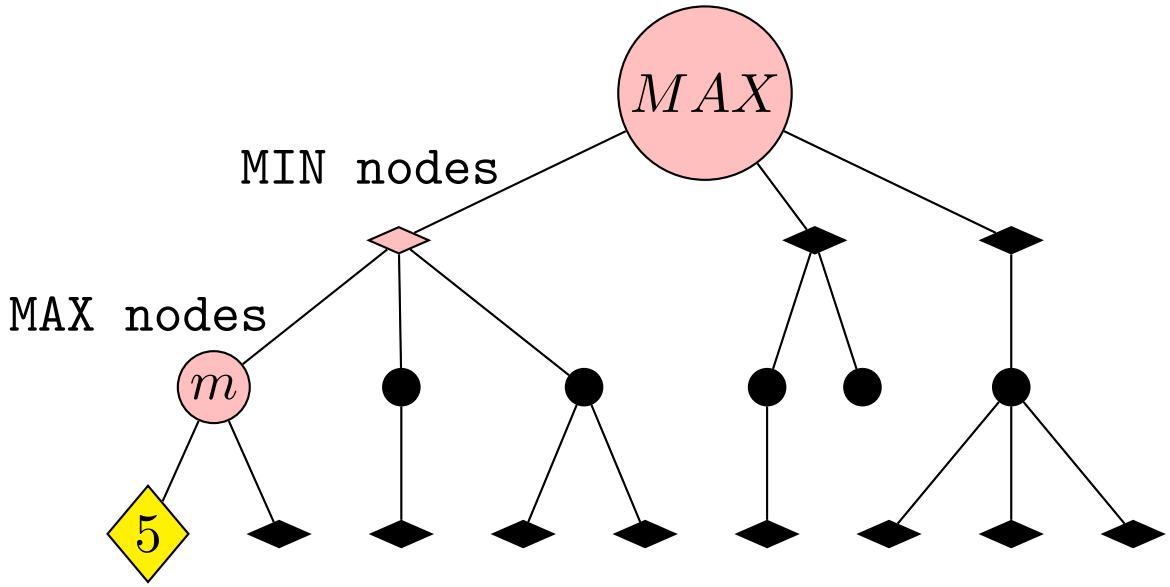
Minimax

- Maximizing the worst-case outcome (MIN chooses this if it plays optimally) for MAX is the best thing MAX can do. If MIN doesn't play optimally, this strategy does even better

- Minimax value (with respect to MAX player) of a state $s \in A=$
  - $v(s)$ if $s \in E$ is an end state
  - maximum of successors minimax values if s is a MAX node
  - minimum of successors minimax values if a is a MIN node

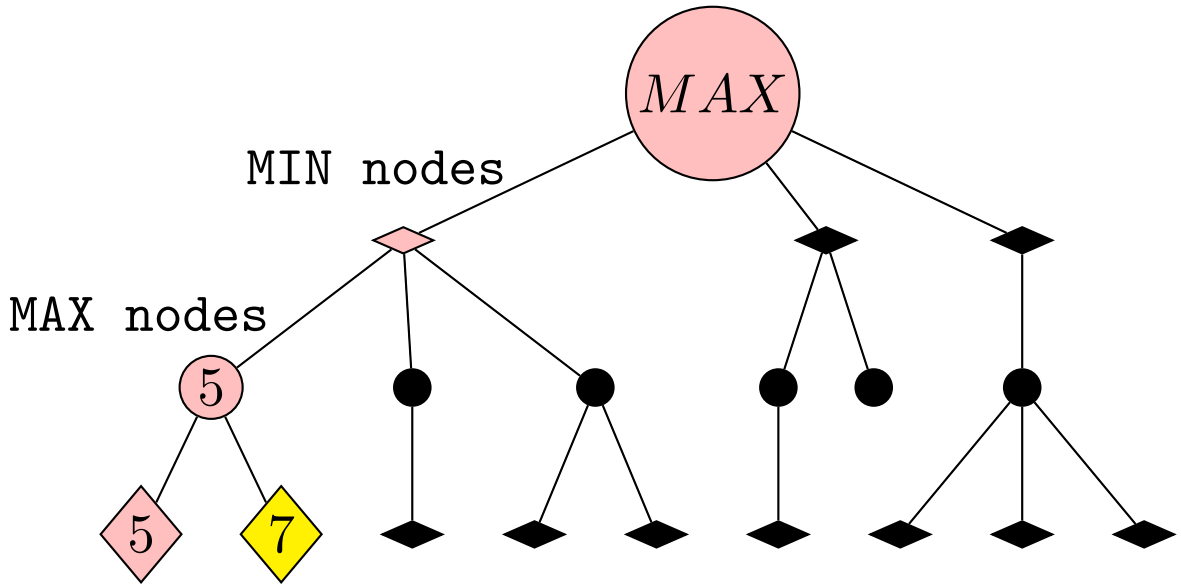
- At a node, MAX would pick the move which gives the highest minimax value and MIN would pick the node which gives the lowest minimax value (thereby giving MIN the highest utility)

Minimax example



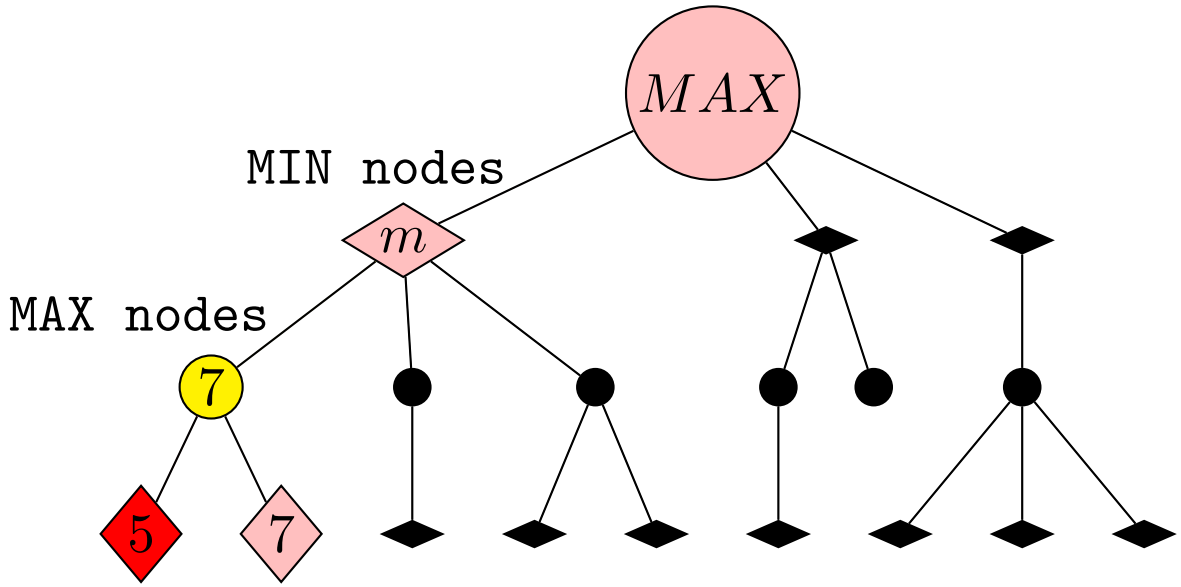
The minimax value of m is taken to be 5 because that is the only child we have explored whose minimax value we know.

Minimax example



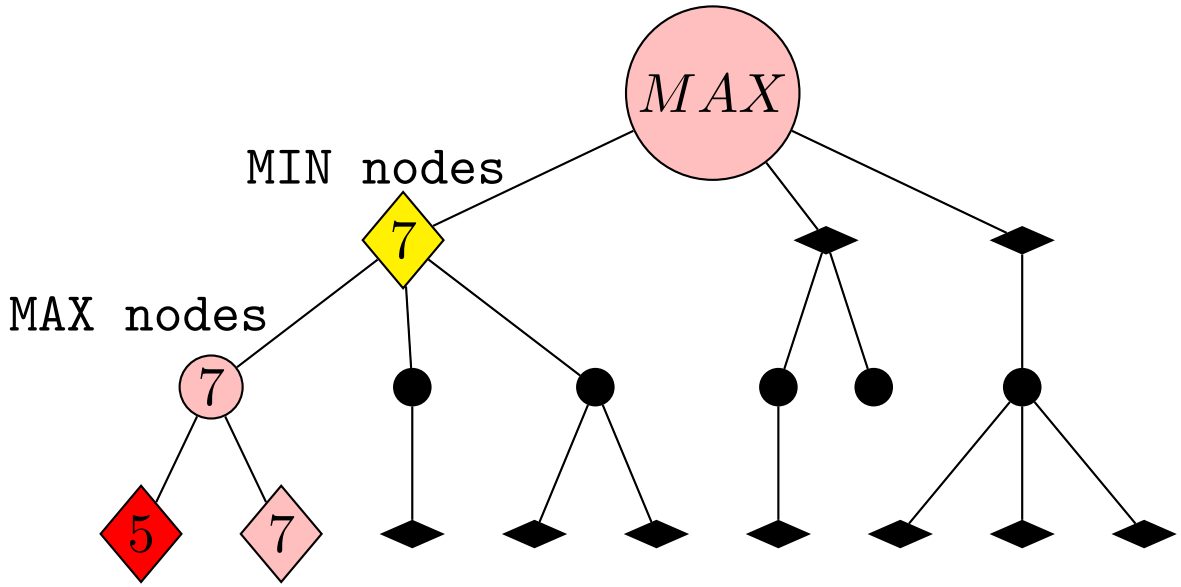
A new end state is explored which has a minimax value of 7. Compare this with the parent's minimax value and change if needed. Here, the circular node is a MAX node and hence, its minimax value is the maximum of its children's values = 7.

Minimax example



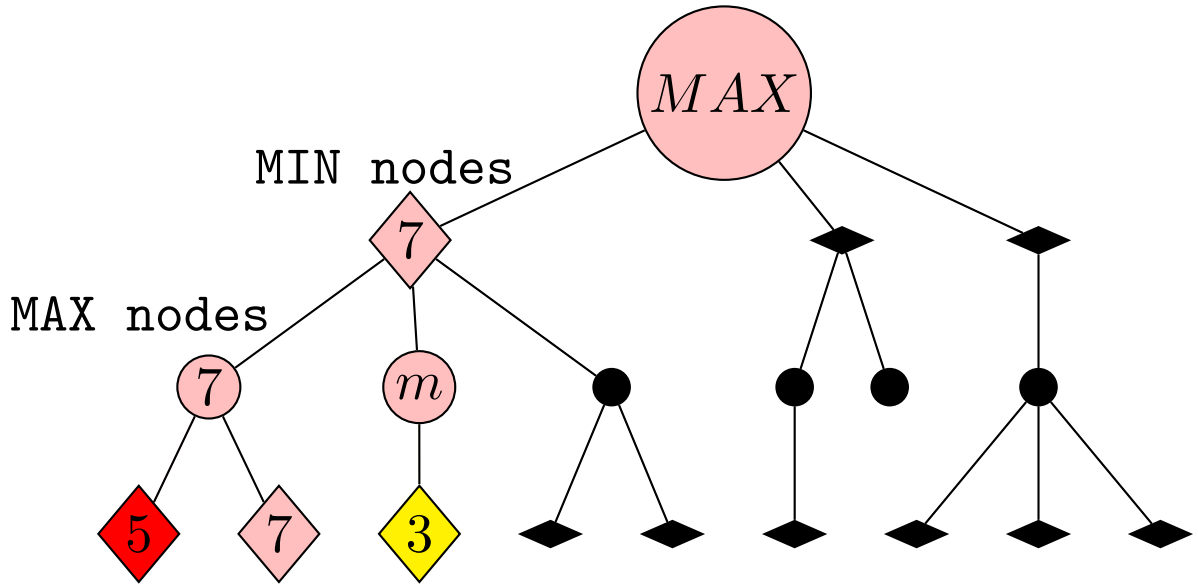
The yellow node has a known minimax value of 7 and from this, the minimax value of node m is taken to be 7, from whatever information is known .

Minimax example



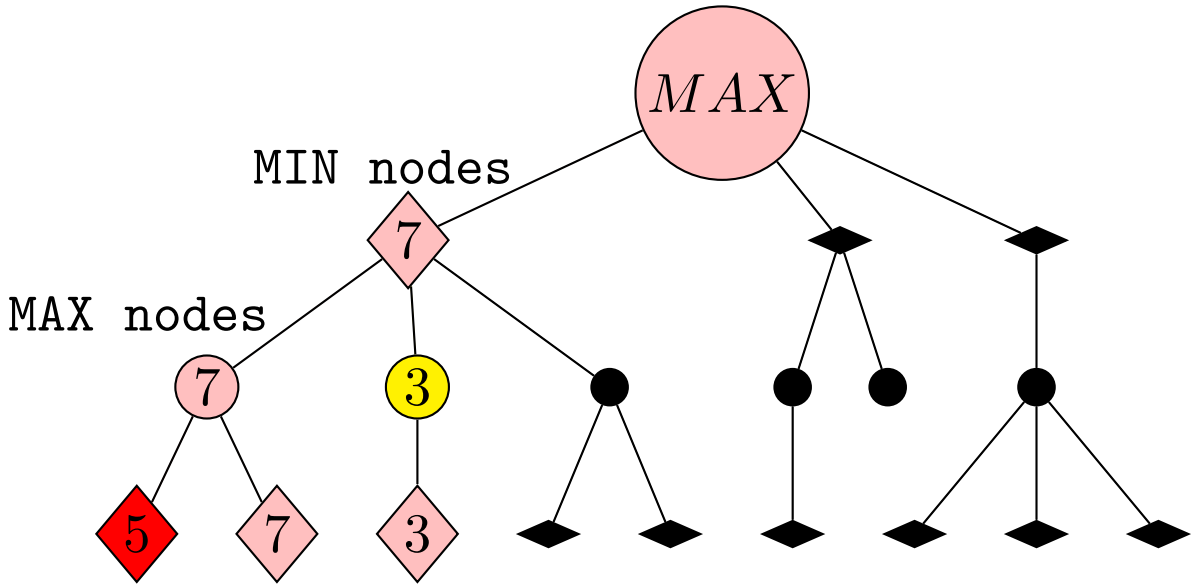
Now we move on to the other children of the yellow node and see if its minimax value needs to be updated.

Minimax example



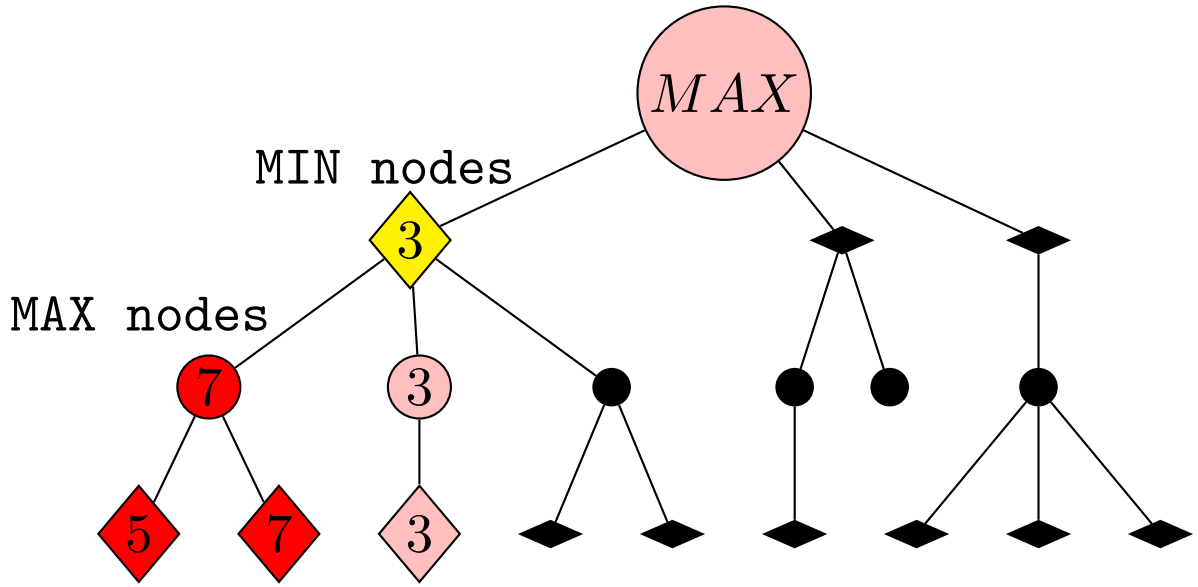
The yellow node is an end state and has a minimax value of 3. Now, m needs to be updated to 3.

Minimax example



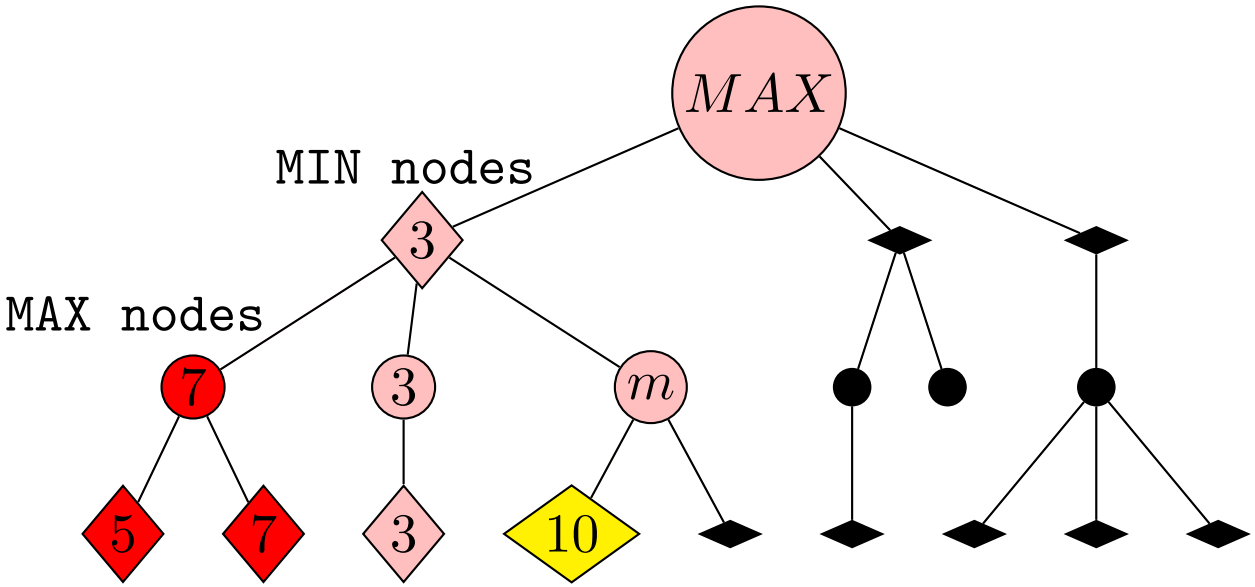
The diamond node with minimax 7 is updated to 3, as it is a minimum node.

Minimax example

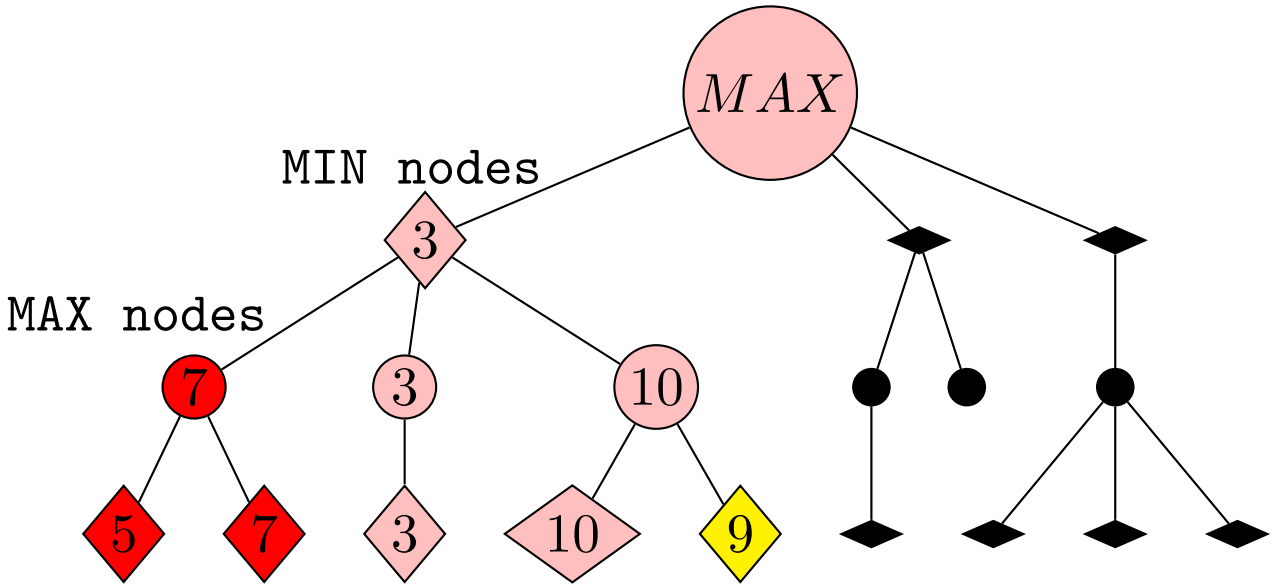


The red portion is ruled out as the yellow diamond node has a circular node whose minimax value is 3, less than that of the red portion's. We move on to explore the other children now.

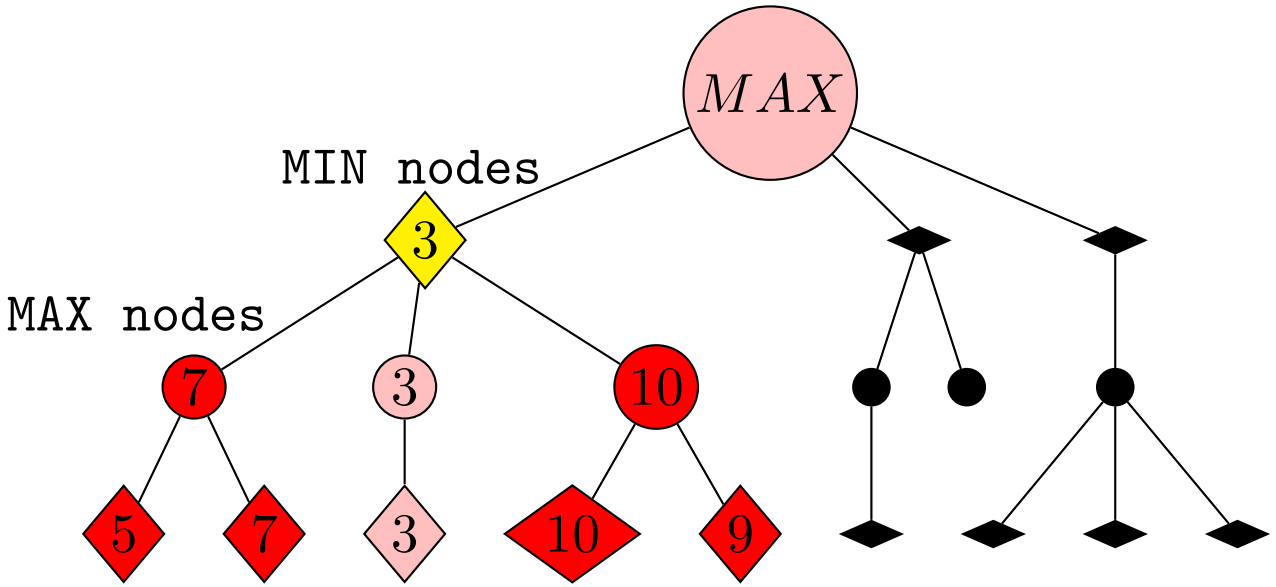
Minimax example



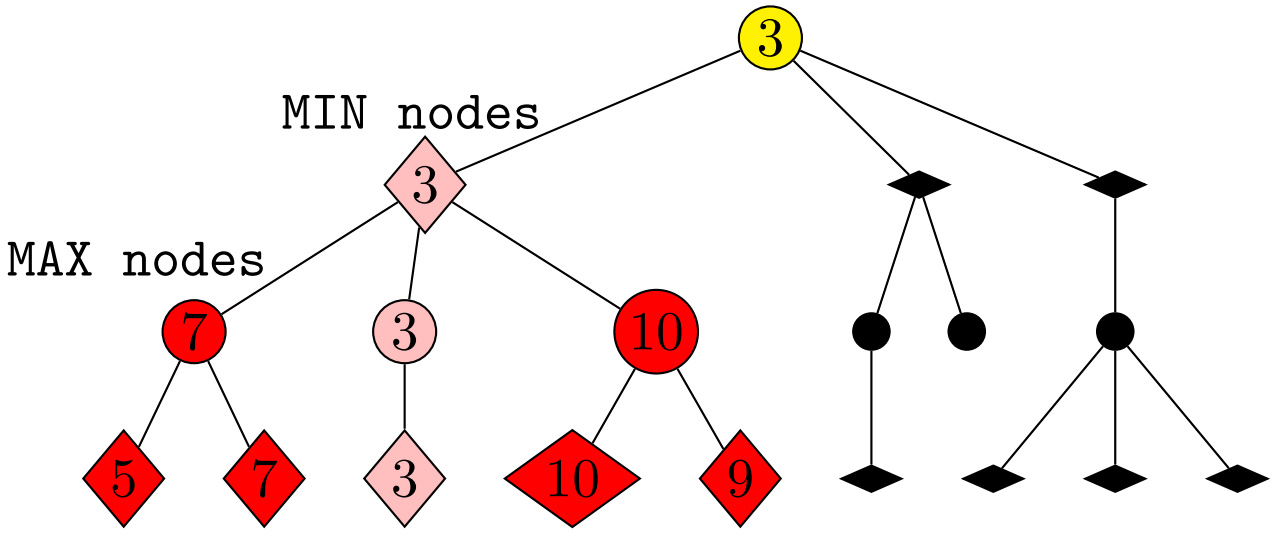
Minimax example



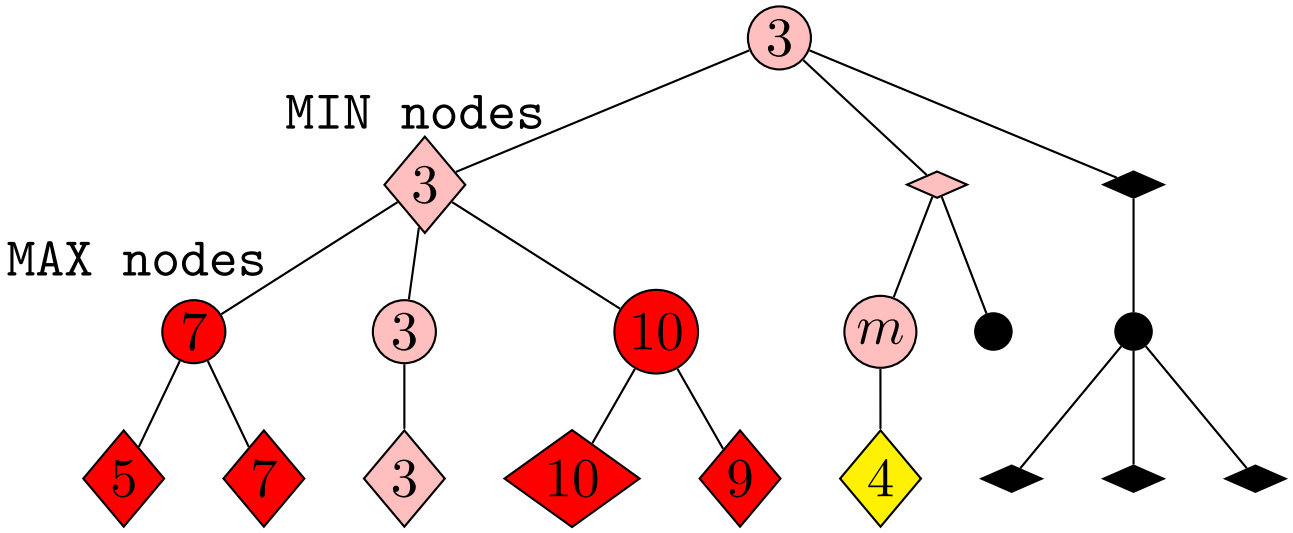
Minimax example



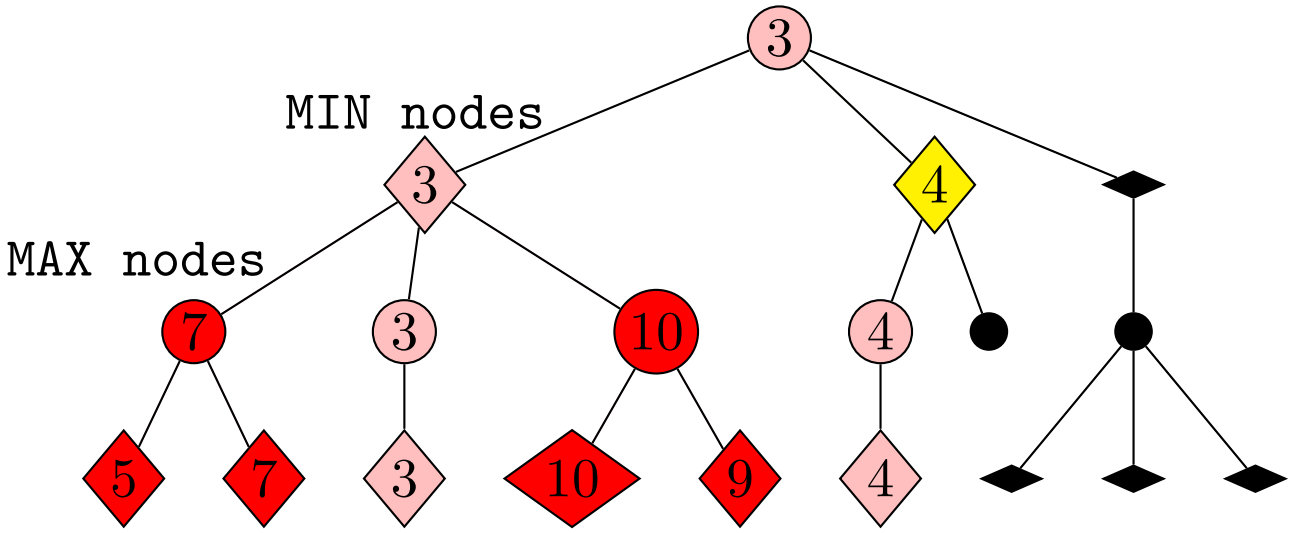
Minimax example



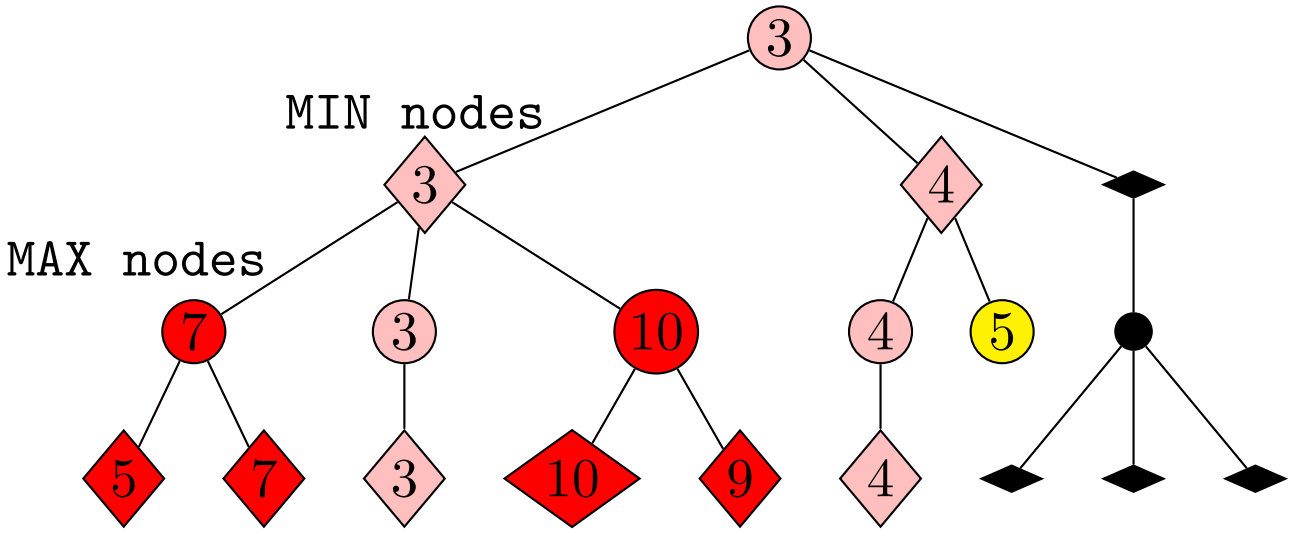
Minimax example



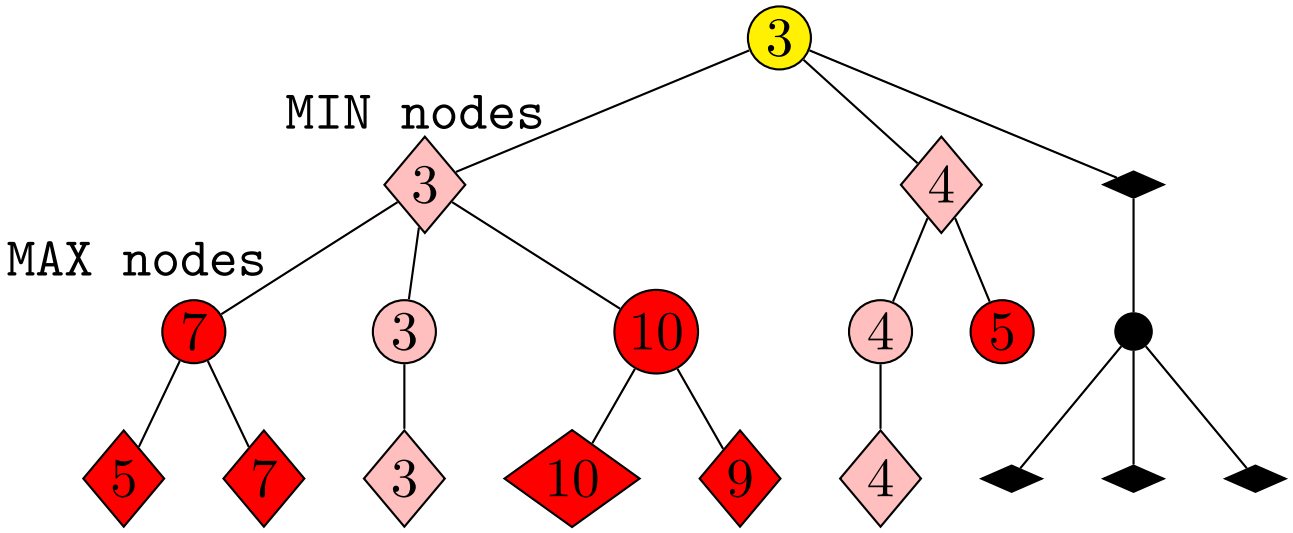
Minimax example



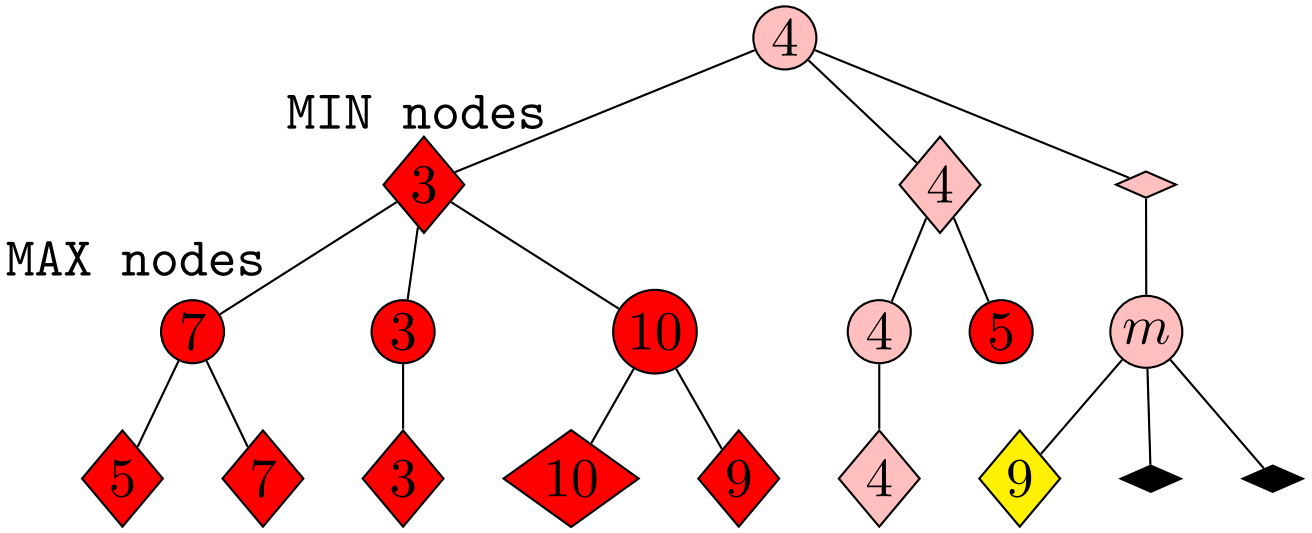
Minimax example



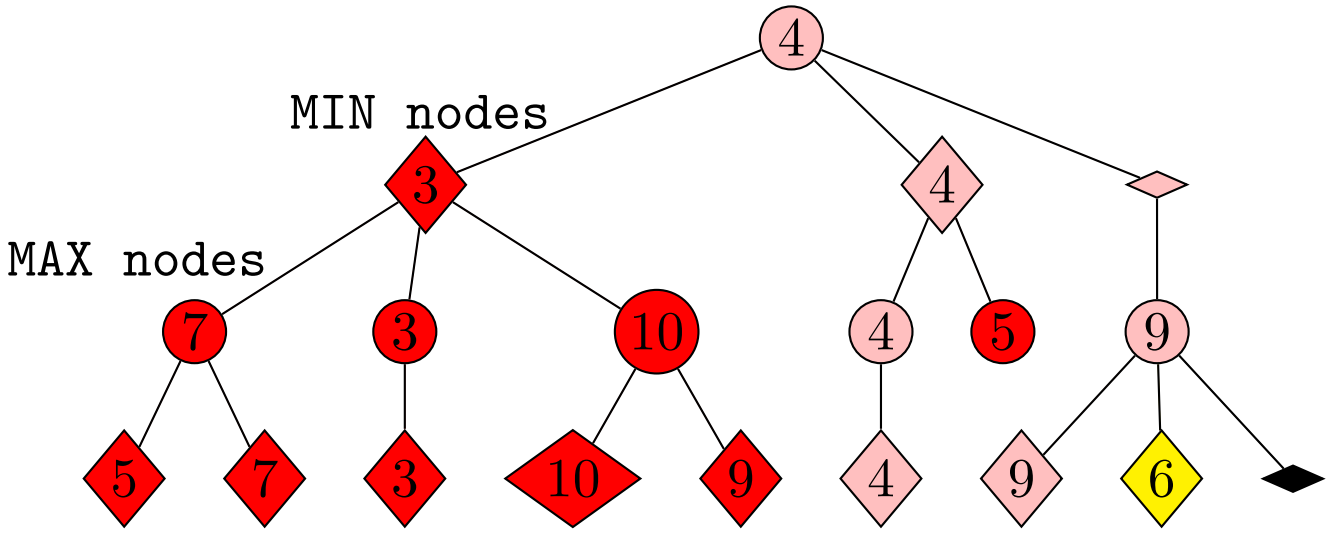
Minimax example



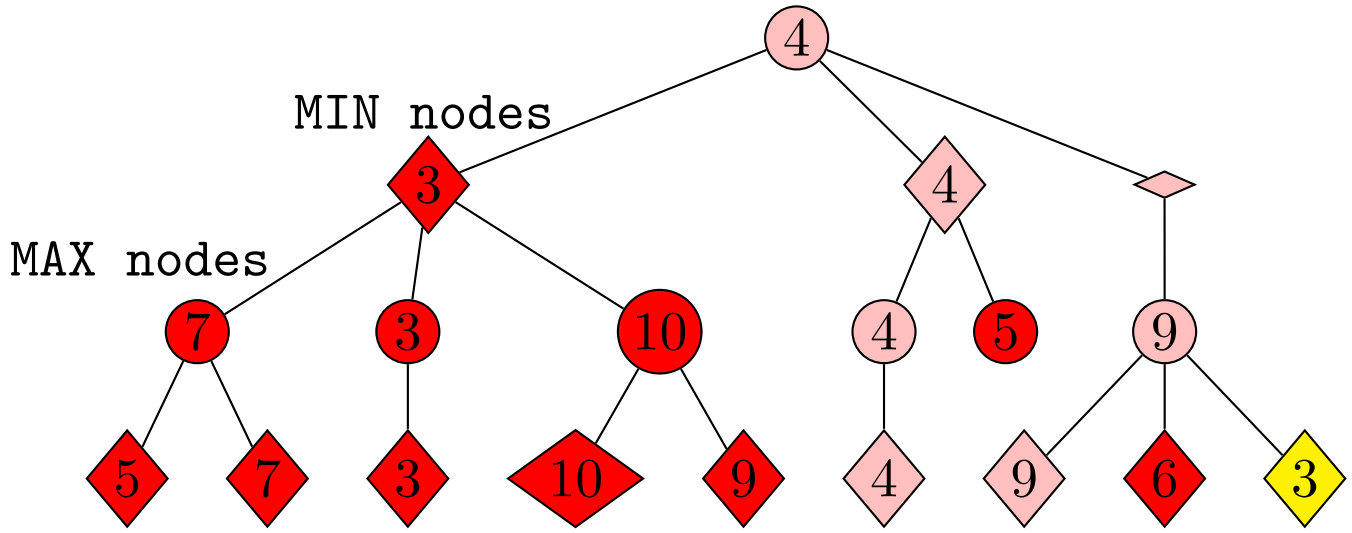
Minimax example



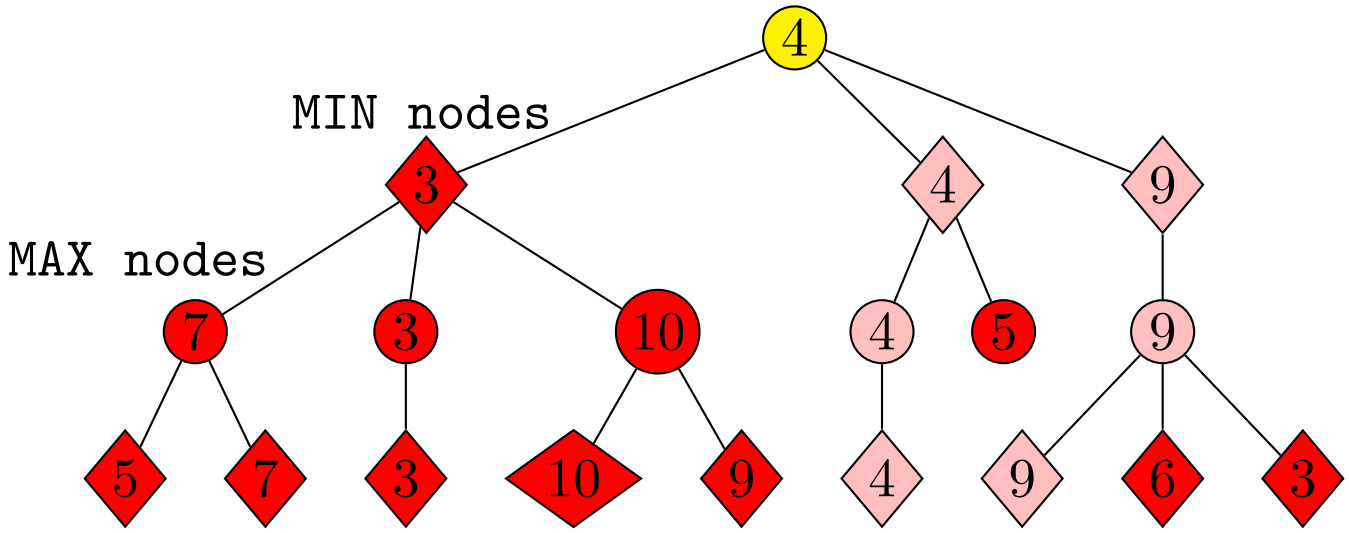
Minimax example



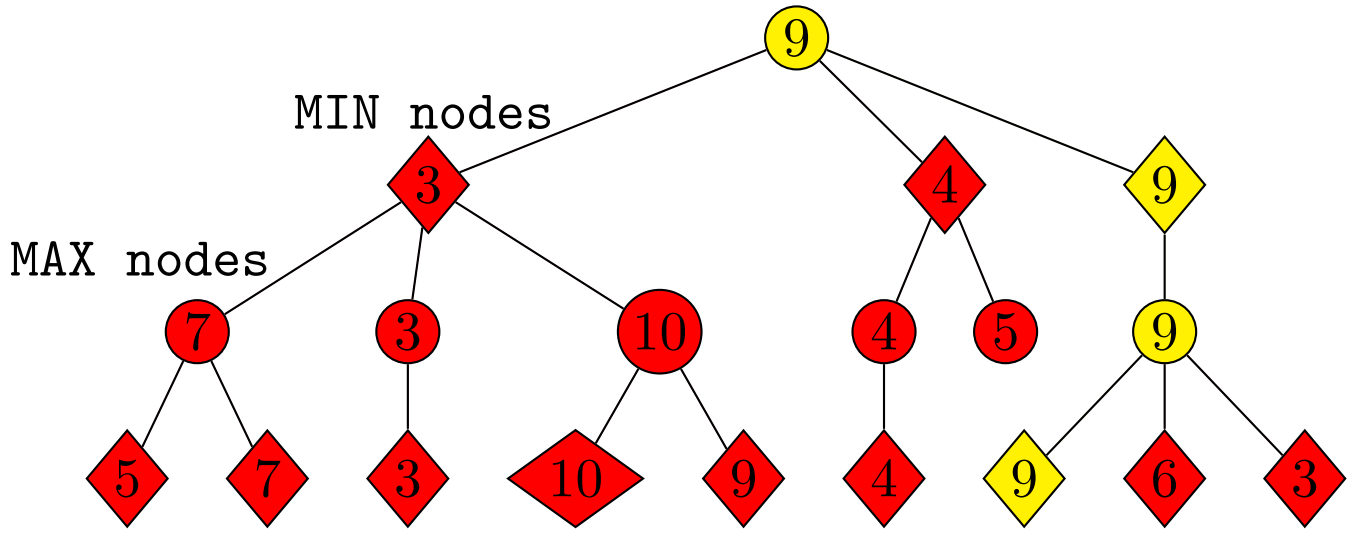
Minimax example



Minimax example



Minimax example



This is the best path that MAX evaluates now. Best move is the third one.

Complexity for Chess

- b: branching factor, average number of children to a node, ≈ 35 in chess
- n: number of moves by both players, ≈ 100 in chess
- Game tree time complexity: $O(b^n) \approx 10^{154}$ in chess
- Searching entire tree is impractical, making optimal decisions in finite time and space constraints is important
- Heuristics for assigning utilities to midgame positions (to consider them as end states of a particular search routine)

Heuristic evaluation function

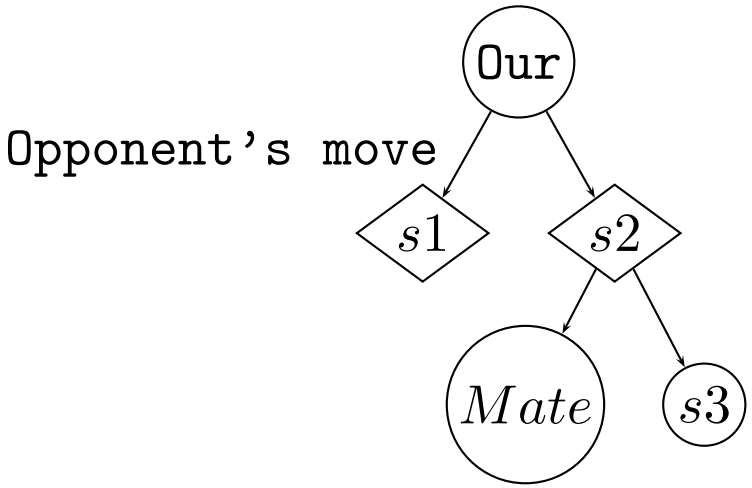
- Cut the depth of search to a certain level - consider some midgame states as potential end states for the sake of algorithm
- Attach utilities to different configurations - how advantageous a position is compared to opponent
- Easily computable, compatible with end state utilities, should reflect the actual state of the game

Heuristic evaluation function

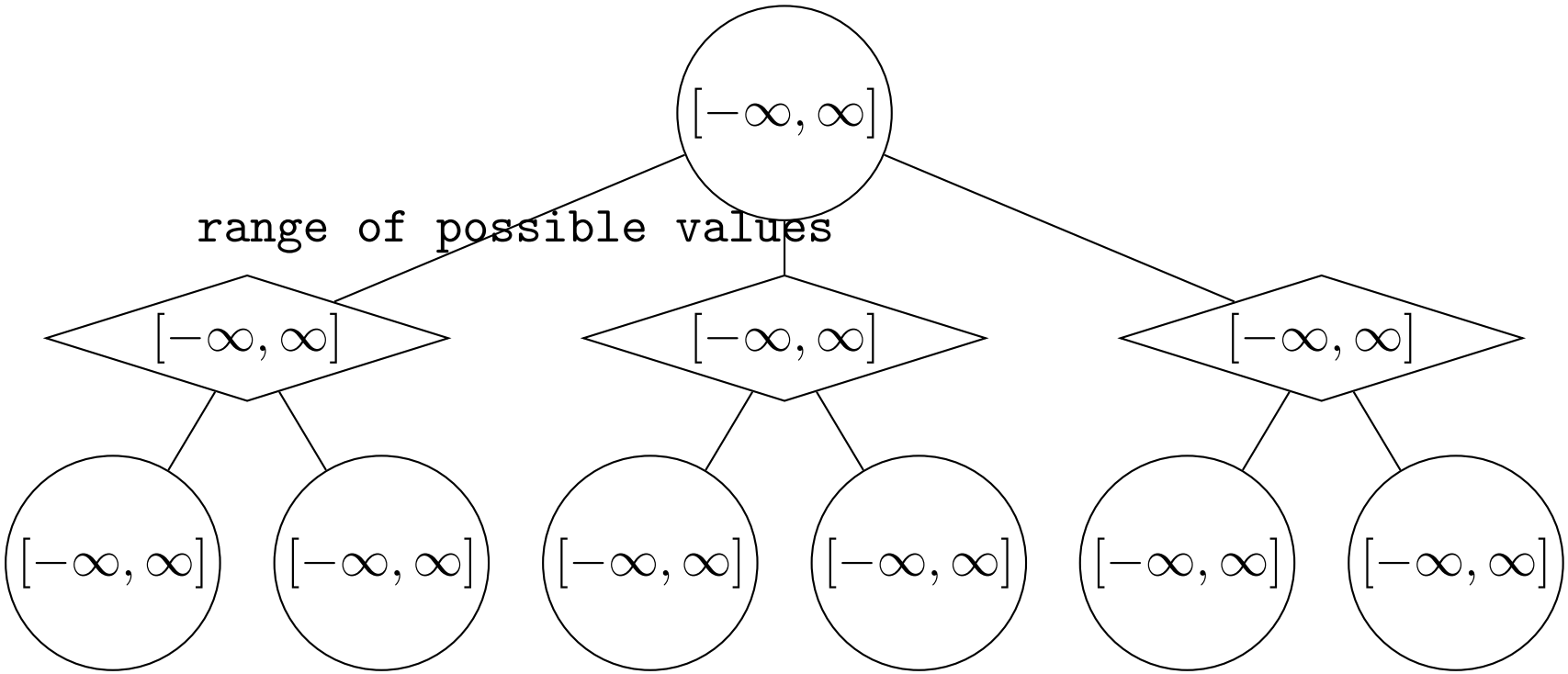
- Each piece has an intrinsic material value which is static, which is a naive way of doing things
- Advanced players analyze groups of pieces (patterns); these are also given values
- e.g. knight/queen/pawn traps (positive utility), wasted pawns (negative utility)
- Dynamic values are also allotted, based on how central and mobile a piece is
- Final heuristic value evaluated as a linear combination of the intrinsic material values, pattern values (features) and dynamic values
- Constrain the search space by cutting off the tree and treating the lowest level nodes as leaves whose utilities are given by this function

Pruning

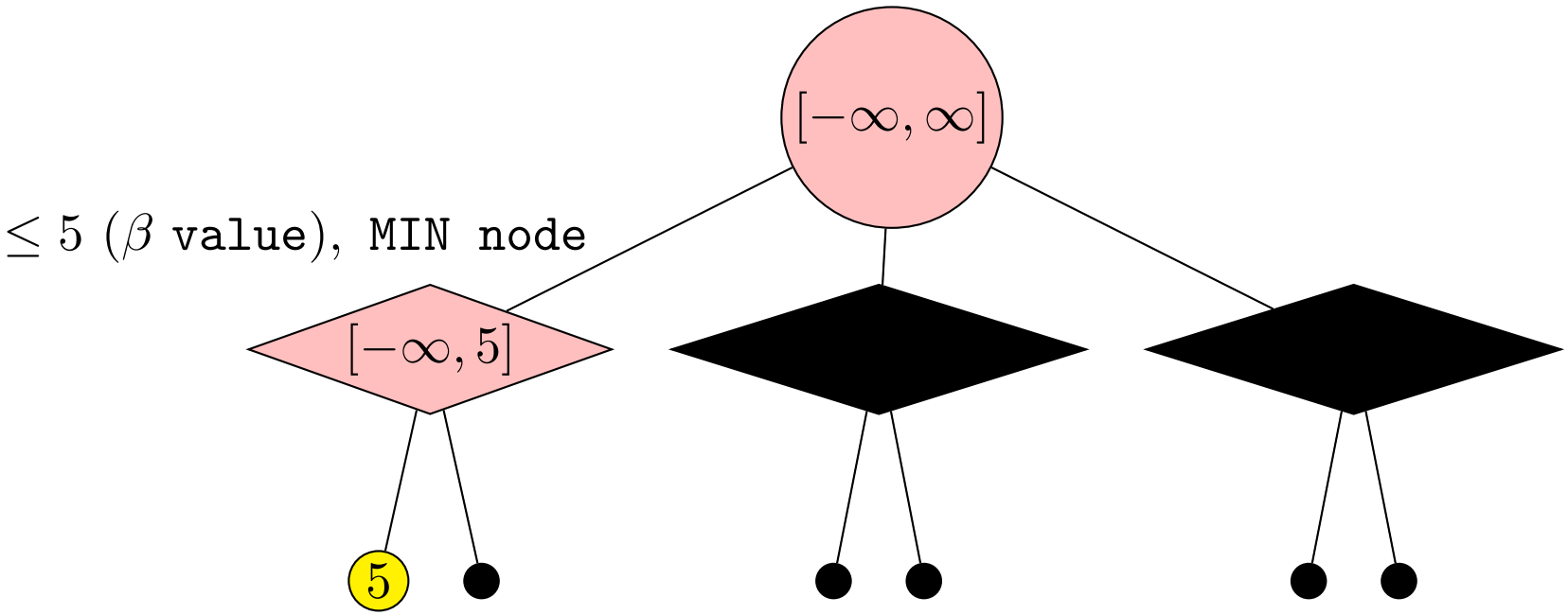
- Even after cutting off the tree and using heuristics, time complexity is large
- If we play $s2$, opponent is going to mate us for sure. Without examining $s3$, we can conclude that $s1$ is better than $s2$. This is the idea behind pruning.



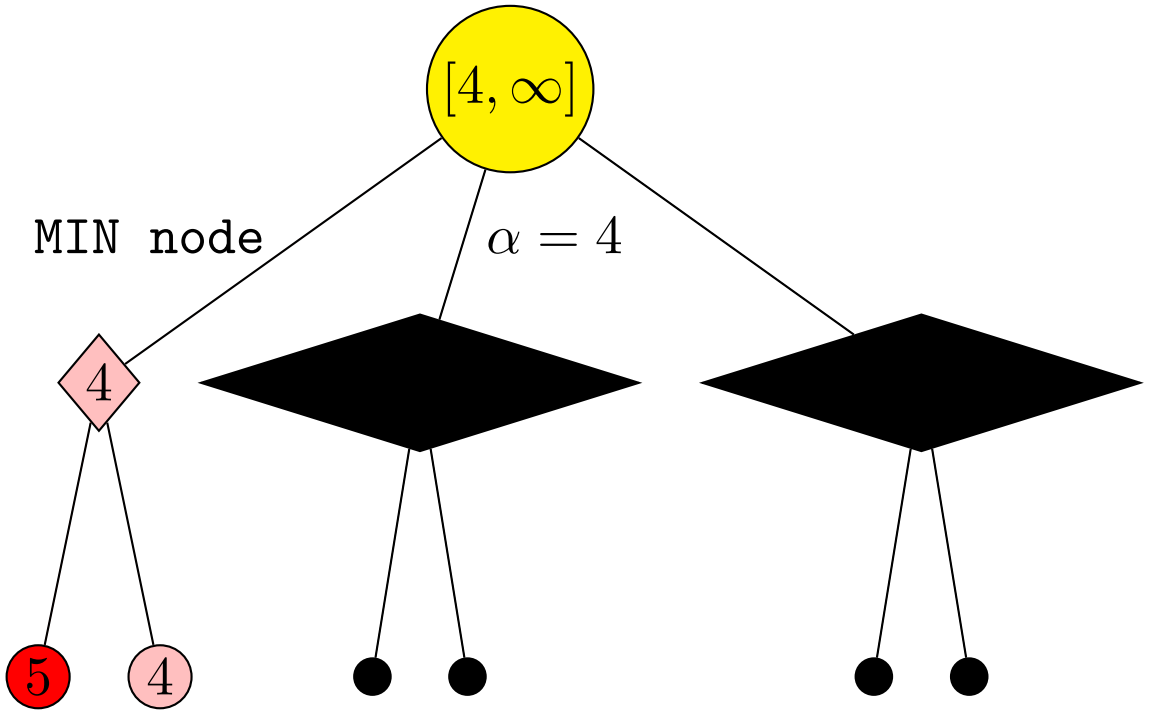
$\alpha\beta$ pruning



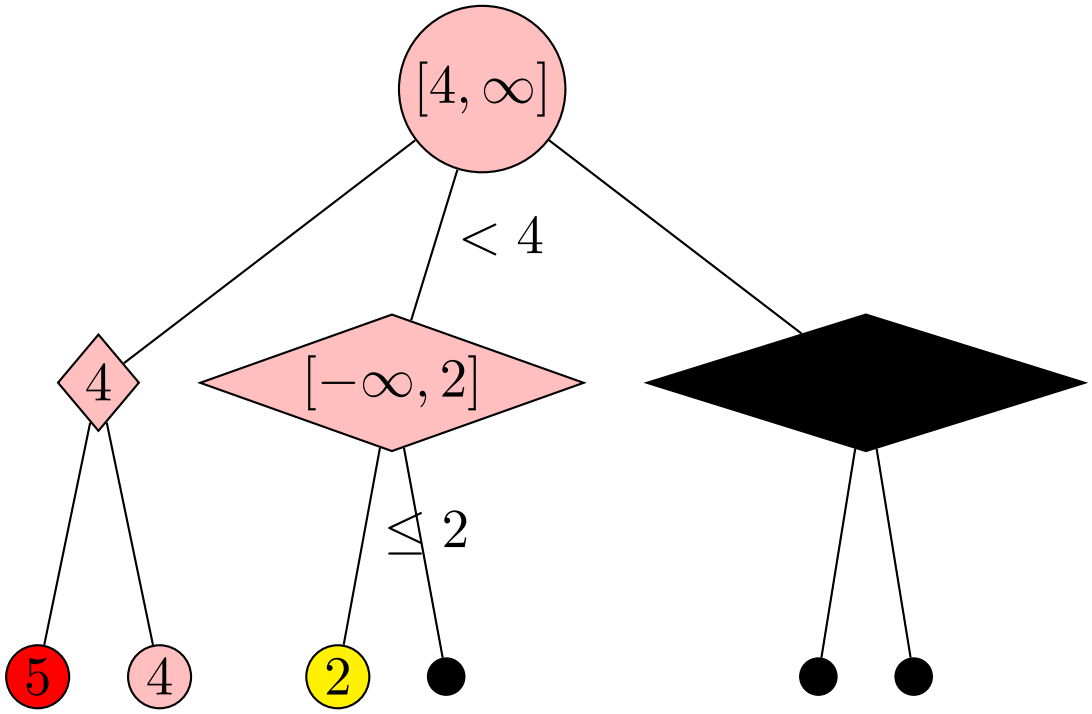
$\alpha\beta$ pruning



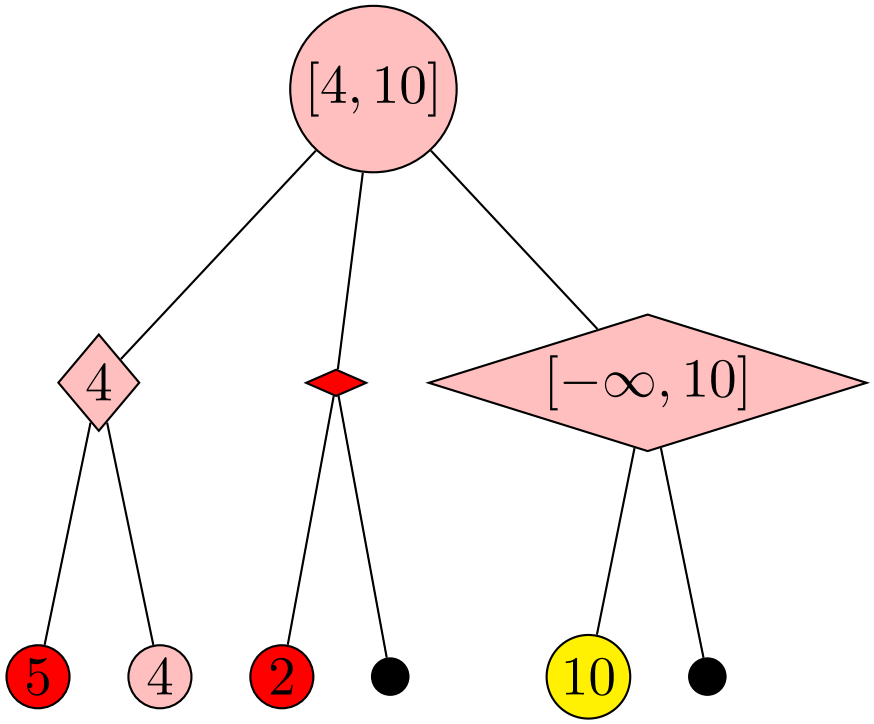
$\alpha\beta$ pruning



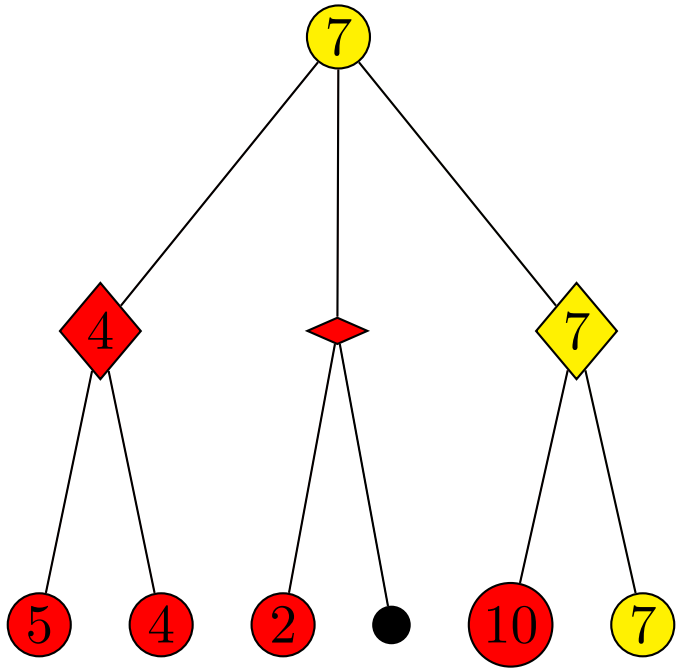
$\alpha\beta$ pruning



$\alpha\beta$ pruning



$\alpha\beta$ pruning



$\alpha\beta$ pruning

- Comes from the branch and bound family of algorithms
- Keep track of the highest minimax value α for MAX and the lowest minimax value β for MIN found so far, along all the paths search via DFS
- For a node s, if its minimax value is known to be worse than α for a MAX node (or β for MIN), prune all other branches of s
- In complex games such as chess which have a large branching factor, pruning saves a lot of time
- If there is a strict time constraint, DFS might give bad results. Iterative deepening - breadth first with increasing depth - would help here.

$\alpha\beta$ pruning

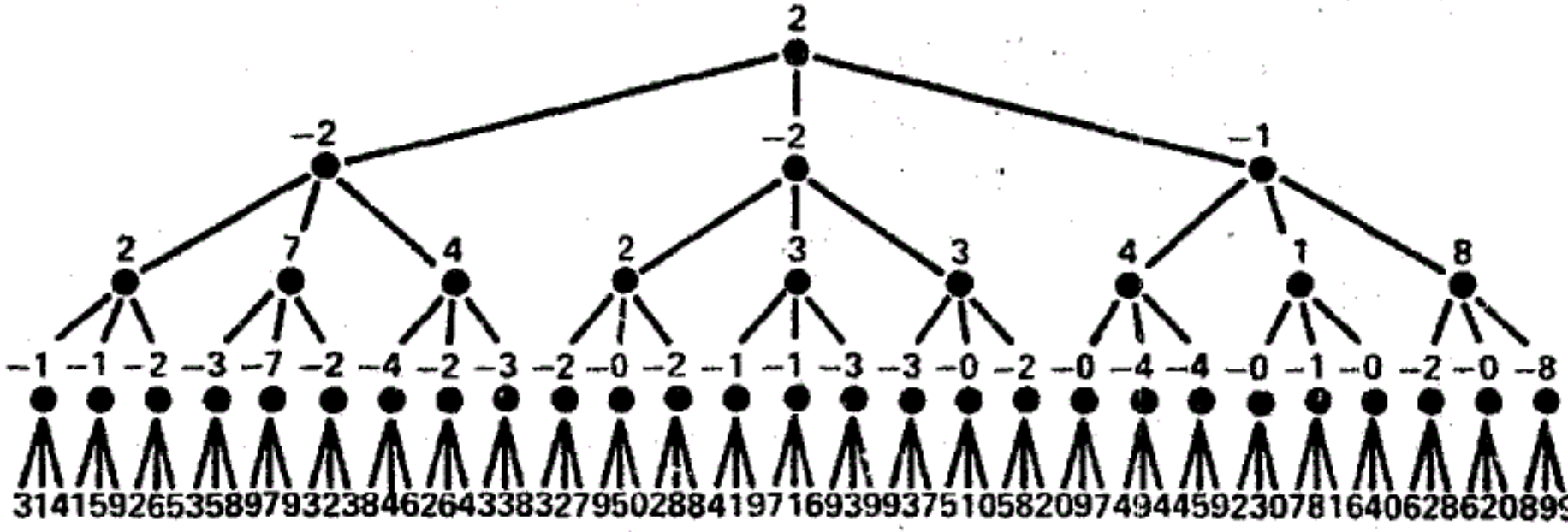


FIG. 1. Complete evaluation of a game tree.

Borrowed from [Knuth and Moore 1975]

Alpha beta pruning

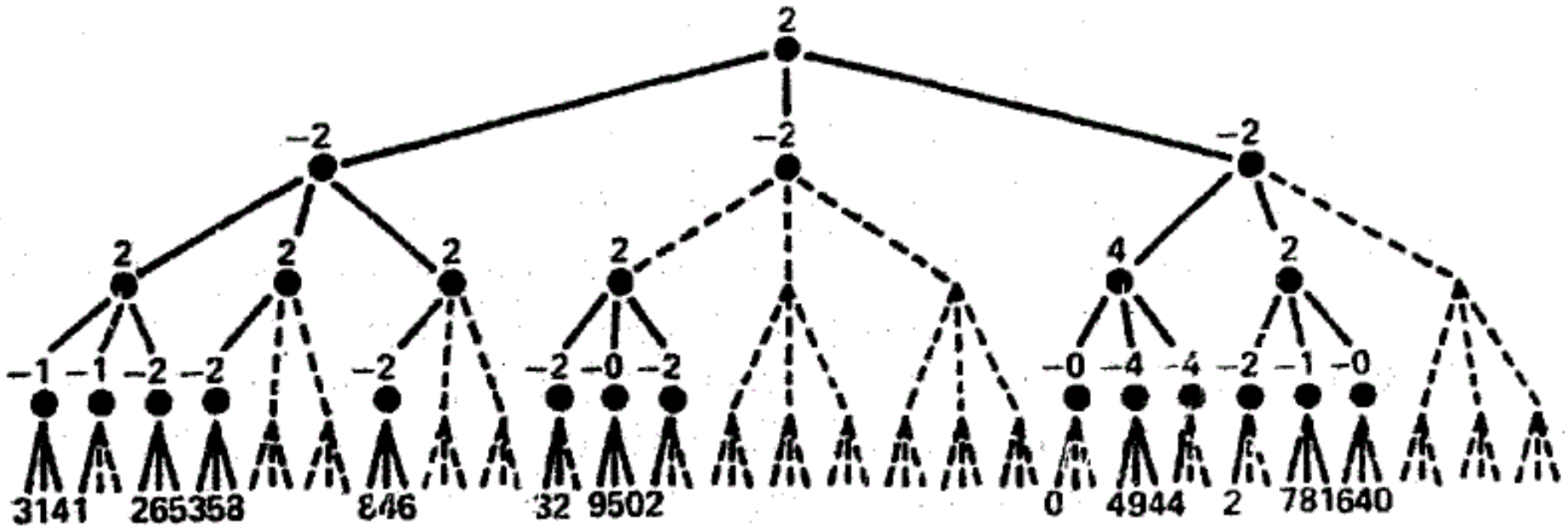


FIG. 3. The game tree of Fig. 1 evaluated with procedure *F2* (alpha-beta strategy).

Borrowed from [Knuth and Moore 1975]

$\alpha\beta$ pruning- Observations

- Sorting of successors can make a huge difference, if we encounter a bound first, we can prune more quickly
- In the best case, we can get the same optimal solution as Minimax in $O(b^{d/2})$ instead of $O(b^d)$ [Knuth and Moore 1975]
- Equivalently, we can reduce the effective branching factor to $\sqrt{b}$ from b
- Static ordering systems for successors - try captures first, followed by threats, forward moves and then, backward moves
- Dynamic ordering systems - Killer Heuristic, History Heuristic - based on assumption that a previously found good move will be a good move again in the present position [Schaeffer 1989]

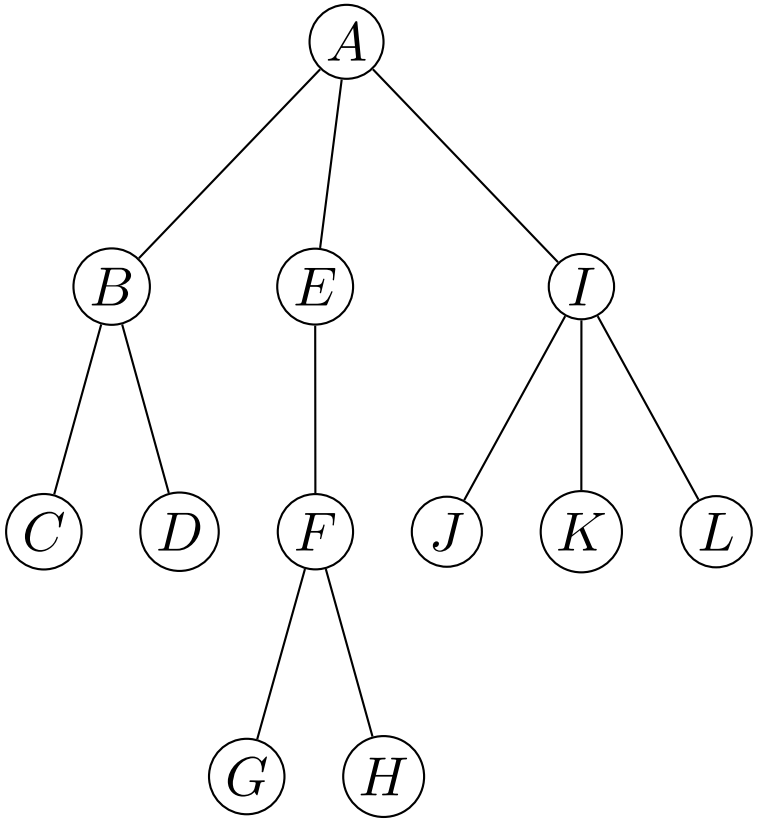
$\alpha\beta$ pruning- Observations

- Smaller range of $[\alpha, \beta]$ can lead to faster pruning
- Iterative Deepening can give a rich history of good moves with increasing confidence in each iteration
- Killer Moves - ones that cause a lot of cut-offs
- Transposition table uses hash tables to remember the information obtained in a particular search now, and use it later in the search of the next move [Schaeffer 1989]

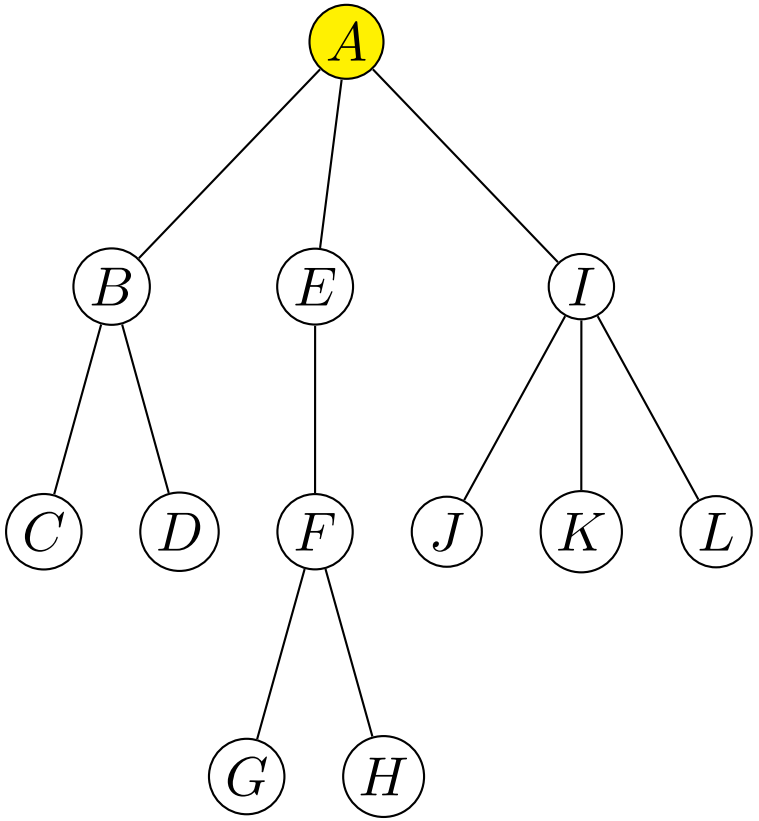
Iterative Deepening

- BFS - Guarantees optimal path, time and space complexities - $O(b^d)$
- DFS - Non-optimal, can find longer path to goal first, might loop in graph search, space complexity - $O(d)$
- Depth First Iterative Deepening - best of both worlds
- Do sequential DFS's for increasing depth starting from 1
- Redundant at first sight, but asymptotically optimal among brute force searches in terms of time, space and length of path found

Iterative Deepening Example

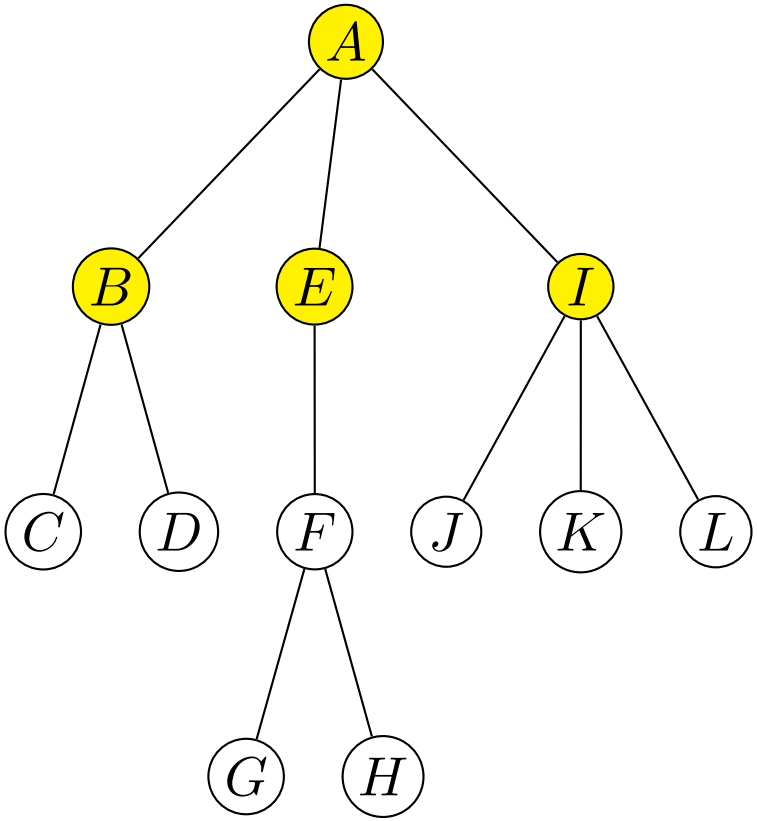


Iterative Deepening Example



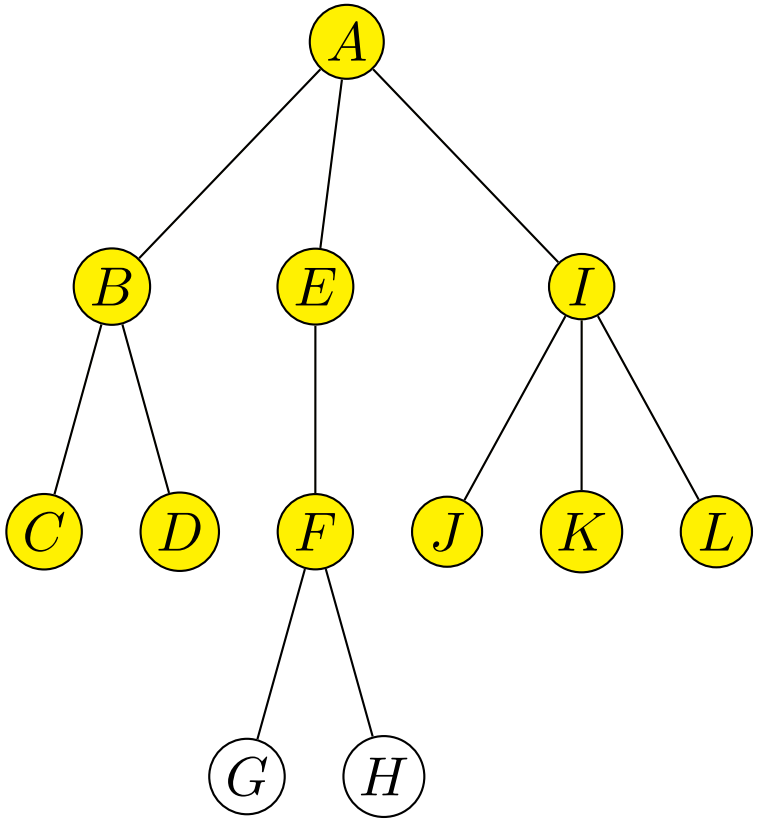
Iteration 1 - A

Iterative Deepening Example



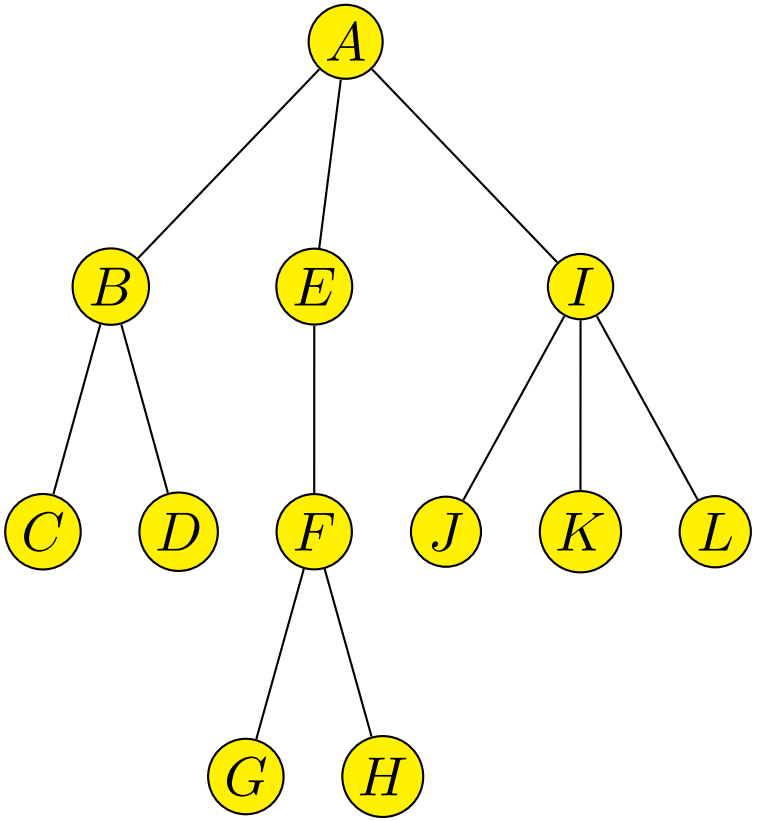
Iteration 2 - ABEI

Iterative Deepening Example



Iteration 3 - ABCDEFIJKL

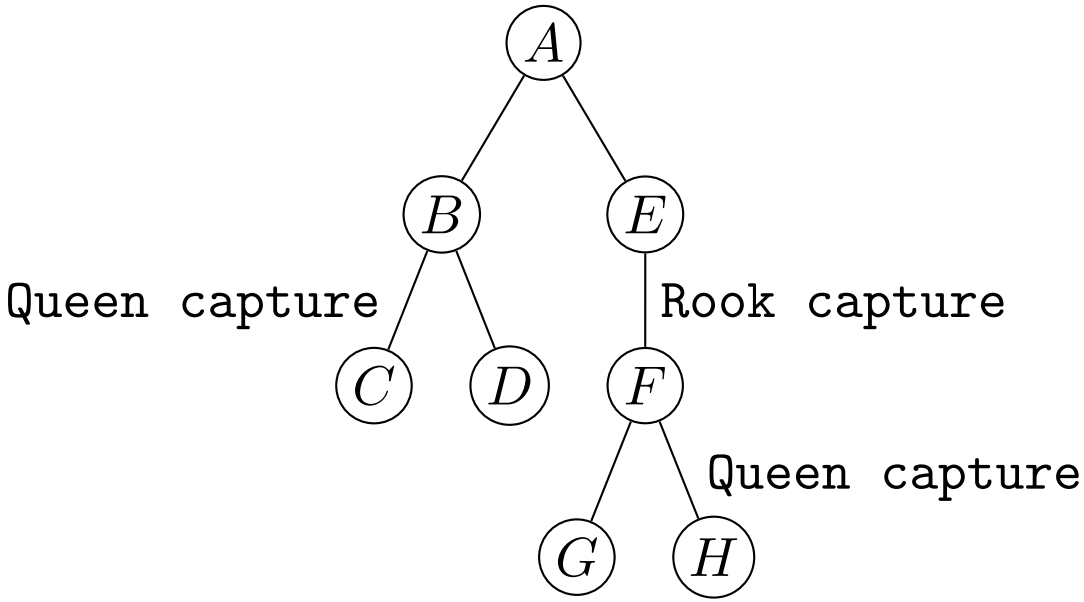
Iterative Deepening Example



Iteration 4 - ABCDEFGHIJKL

Horizon effect

- The program might make a move which is $actually$ a bad one, but it is not able to see the true nature of the move as its consequences show up at a depth greater than what the program can search for



Horizon effect and quiescence

- Heuristic estimates may miss out on complex dependencies, not perfect $\Rightarrow$ determining the depth or time at which to truncate search in game trees a delicate matter
- Emulate what humans do: *intuitively* explore promising moves and forget the dull ones
- Evaluate utilities only at stable positions which reflect the actual situation

Quiescence search

- Searching past the terminal search nodes and testing all the interesting moves (piece capture, pawn advance, midboard acquisition, piece trapping, any other judgement criterion etc.) until the situation becomes "calm" [Hsu 1999]

- Enables the system to detect long capture sequences and calculate whether or not they are worth initiating - a crucial aspect of human chess intelligence

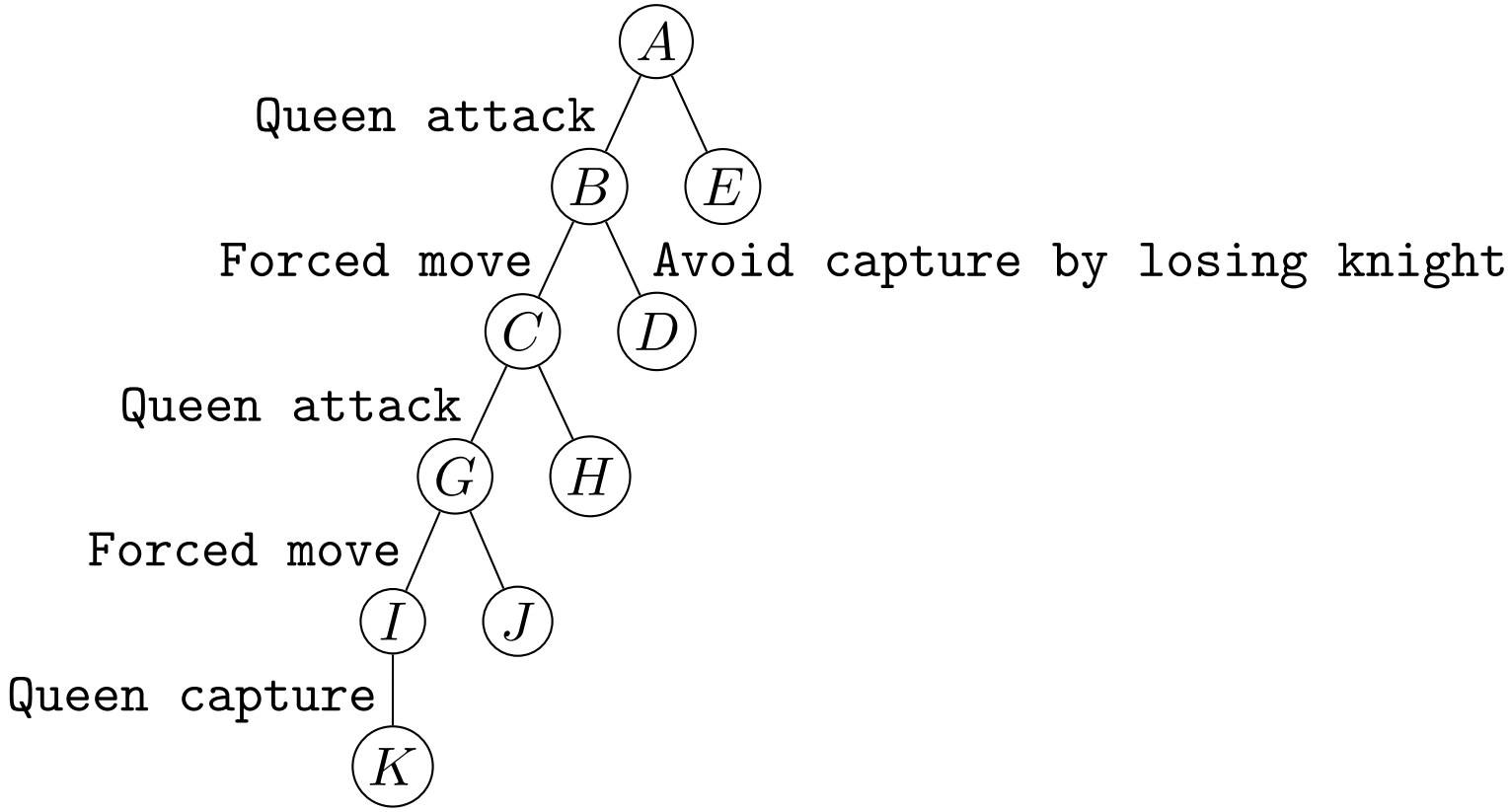
Quiescence and computation

- More than half the time spent on exploring the levels opened up by quiescence
- Deep Blue, through its special VLSI chess chips can generate the interesting moves - forcing and capture moves - really quickly

Forcing moves

- White has a definite winning thread from the current position but black can play numerous delaying moves (checks, mate threats, attacks on important pieces, etc.)
- Eventually Black runs out on any more delaying moves till which precise responses are needed by White
- An extended search is the only sure shot way out of this

Forced moves



Singular extensions

- Going deeper in the search tree is by probing deeper into specific subtrees, especially forced moves [Anantharaman et al. 1990]
- If a particular node is significantly better than other alternative moves at that level, that calls for a review as a mistake would cost much
- This might be an indication that our evaluation function did not give us good values or that we haven't reached to sufficient depth

Experience and knowledge

- Opening book, end game book, extended book (from Grand Master games) [Hsu 1999]
- Randomly chosen opening game, appended by moves from the extended book
- A particular position is allotted a high heuristic utility if it is found to be successful in the Grand Master games database
- A substitute for human experience and knowledge

Conclusions

- Given infinite space and time capability, in principle, a computer chess program would always defeat a human - but the point of a chess game is lost
- Computer programs have an upper hand at space and time complexity management, while humans have an advantage at using the right heuristics for evaluation function

Conclusions

- Deep Blue and other chess programs have tried to emulate an experienced human's heuristic way of thought by using a historical database and nonuniform search, but the actual structure of the heuristic evaluation function also matters

- Is intelligence in the algorithm that one exploits in the wake of insensitive knowledge, or in the procedure that comes up with that (in)sensitive knowledge?

References

- Stuart J. Russell, Peter Norvig (2003), **"Artificial intelligence: A modern approach"**, *Upper Saddle River, N.J: Prentice Hall/Pearson Education*

- Claude E. Shannon (1950), **"Programming a computer for playing chess"**, *Philosophical Magazine*, Ser.7, Vol. 41, No. 314

- Donald E. Knuth, Ronald W. Moore (1975), **"An analysis of alpha-beta pruning"**, *Artificial Intelligence*, Vol. 6, No. 4, pp. 293-326

- J. Schaeffer (1989), **"The history heuristic and alpha-beta search enhancements in practice"**, *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-11, No. 11, pp. 1203-1212

- **Research, IBM**, information about Deep Blue retrieved from http://www.research.ibm.com/deepblue/

- Thomas Anantharaman, Murray S. Campbell, Feng H. Hsu (1990), **"Singular extensions: Adding selectivity to brute-force searching"**, *Artificial Intelligence*, Vol. 43, pp. 99-109

- Feng H. Hsu (1999), **"IBM's Deep Blue Chess Grandmaster Chips"**, *IEEE Micro*, Vol. 19, No. 2, pp. 70-81

- Feng H. Hsu, Murray S. Campbell, Joseph A. Hoane, **"Deep Blue system overview"**, in ICS '95: *Proceedings of the* 9^{th} *international conference on Supercomputing (1995)*, pp. 240-244

- Wikipedia
  - **"Minimax"**, in *Wikipedia*, retrieved November 1, 2011, from http://en.wikipedia.org/wiki/Minimax

- **"Alpha-beta pruning"**, in *Wikipedia*, retrieved November 1, 2011, from http://en.wikipedia.org/wiki/Alpha-beta_pruning
- Image of **The Turk**, from Karl Gottlieb von Windisch's 1784 book *Inanimate Reason*, extracted from http://en.wikipedia.org/wiki/The_Turk

- The Computer History Museum
  - **Shannon and Lasker at Shannon's chess machine**, (1950), *Gift of Monroe Newborn*, retrieved November 10, 2011, from http://www.computerhistory.org
  - **Herbert Simon**, (1958), *Courtesy of Carnegie Mellon University*, retrieved November 10, 2011, from http://www.computerhistory.org

- Chess diagrams generated using the **Chess Diagram Generator** script from http://www.chessvideos.tv