6.034: Lecture 5

Adversarial search & games

Can computers play chess?

Professor Robert C. Berwick

Chess (Yes, Chess) Is Now a Streaming Obsession

Viewers are flocking to games during the pandemic, entranced by a charismatic grandmaster and his lightning-fast play.

Hikaru Nakamura

AI Methods

- **Problem solving**
  - G+T, *search, optimal search, games, constraint satisfaction*
- **Inference**
  - *rule-based systems, Bayesian inference*
- **Machine learning**
  - *k-nearest neighbors, id trees, neural nets*, deep neural nets, *support vector machines*, genetic algorithms, near miss/one-shot
- **Communication, perception, action**
  - natural language processing, vision, robotics

Menu for today

Games: Adversarial search

- ❑ Intelligence & Chess
- ❑ Ways to Play Chess
- ❑ How to search: Minimax algorithm
- ❑ Improving search: Alpha-beta pruning
- ❑ Progressive Deepening
- ❑ Reflections on contemporary chess computers & Gold Star ideas

A very brief history

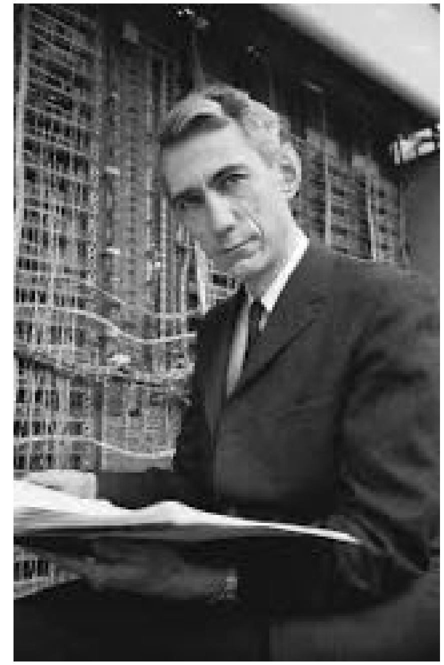
Philosophical Magazine, Ser.7, Vol. 41, No. 314 - March 1950.

XXII. Programming a Computer for Playing Chess[1]

By CLAUDE E. SHANNON

Bell Telephone Laboratories, Inc., Murray Hill, N.J.[2]

[Received November 8, 1949]



Claude Shannon
(1916-2001)

1. INTRODUCTION

This paper is concerned with the problem of constructing a computing routine or "program" for a modern general purpose computer which will enable it to play chess. Although perhaps of no practical importance, the question is of theoretical interest, and it is hoped that a satisfactory solution of this problem will act as a wedge in attacking other problems of a similar nature and of greater significance. Some possibilities in this direction are: -

> The chess machine is an ideal one to start with, since: (1) the problem is sharply defined both in allowed operations (the moves) and in the ultimate goal (checkmate); (2) it is neither so simple as to be trivial nor too difficult for satisfactory solution; (3) chess is generally considered to require "thinking" for skilful play; a solution of this problem will force us either to admit the possibility of a mechanized thinking or to further restrict our concept of "thinking"; (4) the discrete structure of chess fits well into the digital nature of modern computers.

ALCHEMY AND ARTIFICIAL INTELLIGENCE

Hubert L. Dreyfus

December 1965

remains unimproved. Burton Bloom at M.I.T. has made the latest attempt to write a chess program; like all the others, it plays a stupid game. In fact, in the nine years since the Los Alamos program beat a weak player, in spite of a great investment of time, energy, and ink, the only improvement seems to be that a machine now plays poorly on an eight-by-eight rather than a six-by-six board. According to Newell, Shaw, and Simon themselves, evaluating the Los Alamos, the IBM, and the NSS programs: "All three programs play roughly the same quality of chess (mediocre) with roughly the same amount of computing time" [20:14]. Still no chess program can play even amateur chess, and the world championship tournament is only two years away.

The Greenblatt chess program

by *RICHARD D. GREENBLATT,*

DONALD E. EASTLAKE, III,

and

STEPHEN D. CROCKER

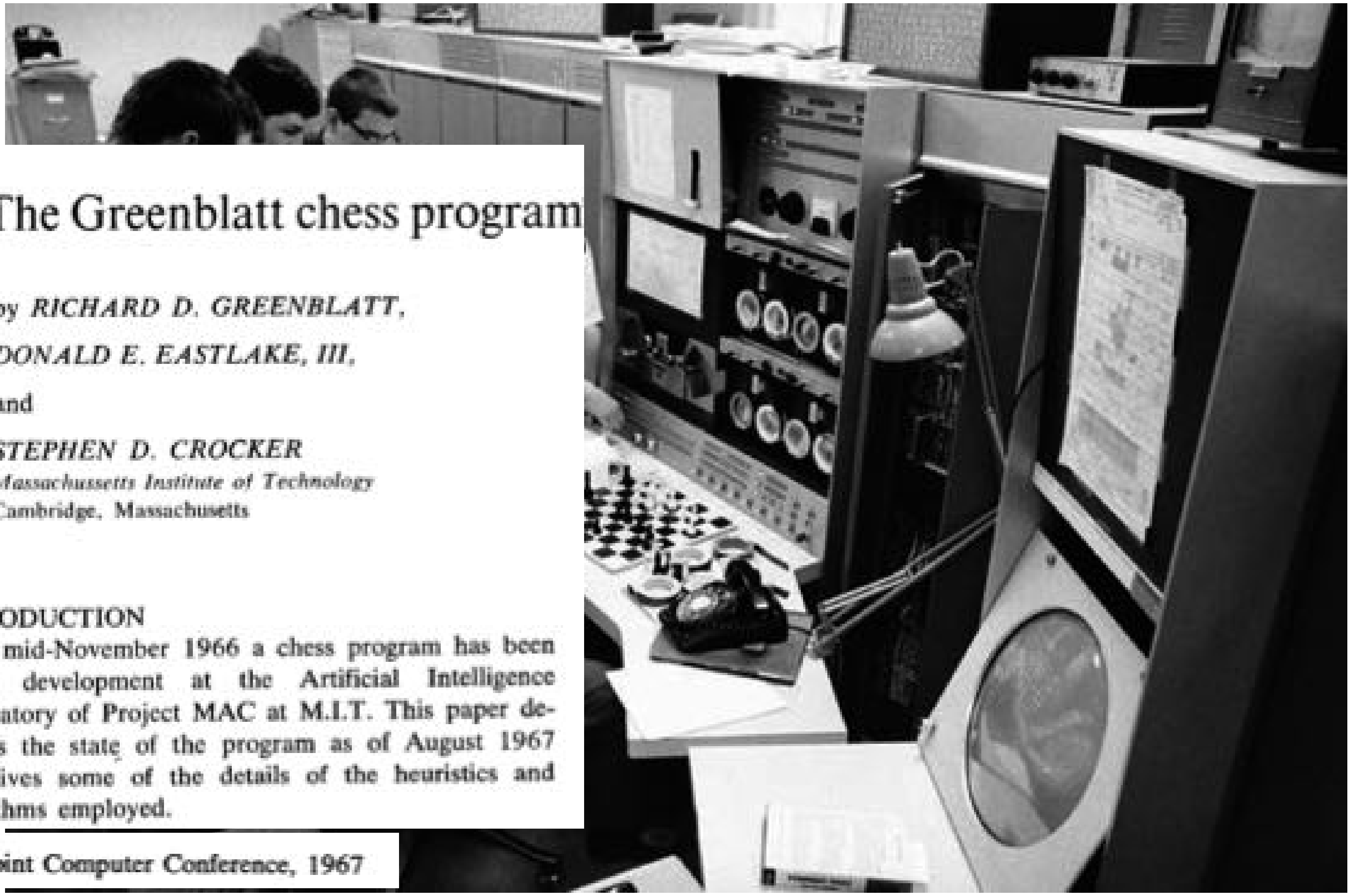
Massachussetts Institute of Technology

Cambridge, Massachusetts

INTRODUCTION

Since mid-November 1966 a chess program has been under development at the Artificial Intelligence Laboratory of Project MAC at M.I.T. This paper describes the state of the program as of August 1967 and gives some of the details of the heuristics and algorithms employed.

Fall Joint Computer Conference, 1967



MIT AI Lab MacHack IV

Richard Greenblatt

H. Dreyfus vs. MacHack IV

Dreyfus vs. "Robert Q"
1967

Prof. Dreyfus loses after 38 moves

THE ARTIFICIAL INTELLIGENCE OF HUBERT L. DREYFUS

A Budget of Fallacies

by

Seymour Papert

January 1968.

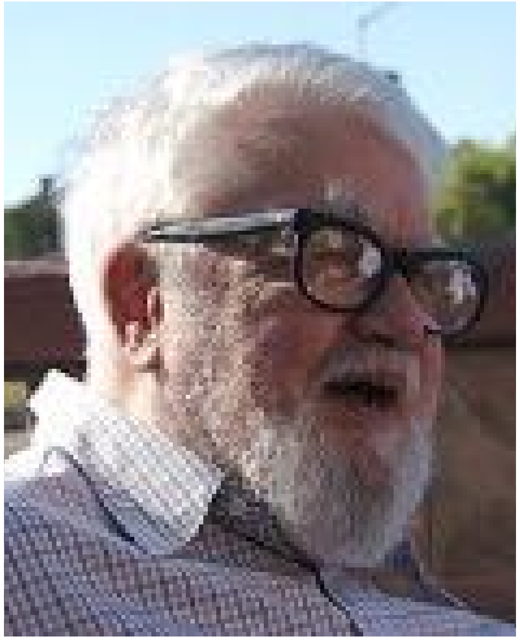
1.5 Computers Can't Play Chess.

1.5.1 Nor Can Dreyfus.

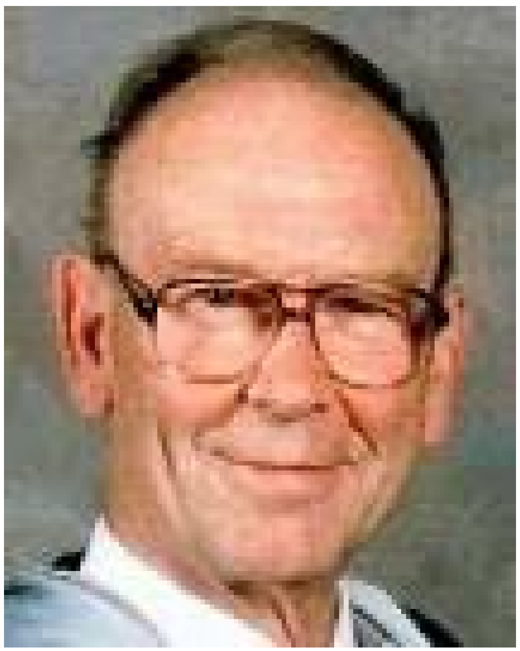
I hope readers will pardon a digression to get a debating point off my chest. They will surely understand why I find it irresistible.

In *Alchemy* and *Phenomenology* Dreyfus discusses the weakness of chess-playing programs. He plainly gives the impression that they can typically be defeated by human novices. His twice recounted story of how a ten-year-old child defeated a program constructed by Newell, Shaw and Simon was gleefully quoted by the *New Yorker* and other popular magazines as demonstrating the futility of Artificial Intelligence. While *Phenomenology* was in press I had the pleasure of arranging for Dreyfus to play against Richard Greenblatt's chess program at M.I.T. and seeing him very roundly trounced. The newsletter SIGART reprinted the game with no comment beyond one phrase from *Alchemy:*

"... no chess program can play even amateur chess." (p. 10)



John McCarthy



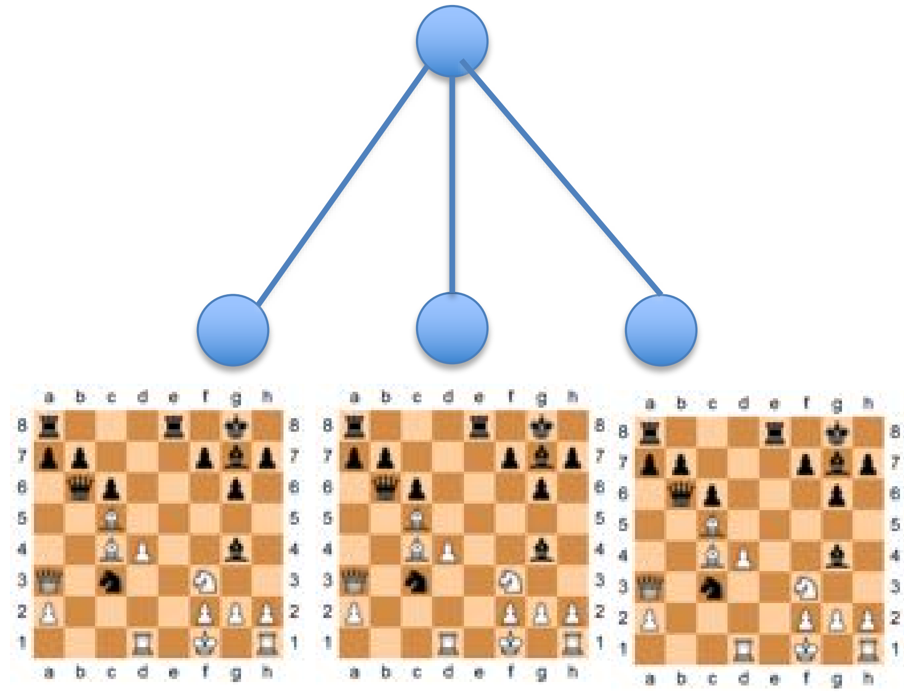
Donald Michie

Versus.

Chess Grandmaster David Levy
Bet in 1968, 1250 pounds

How could a computer play chess?

1. Human like : Describe board via pawn structure, King safety, good time to castle….use tactics, strategy then make a move
2. If-then rules – make most plausible move
3. Look ahead and evaluate



$$S = g(f_1, f_2, \ldots, f_n)$$

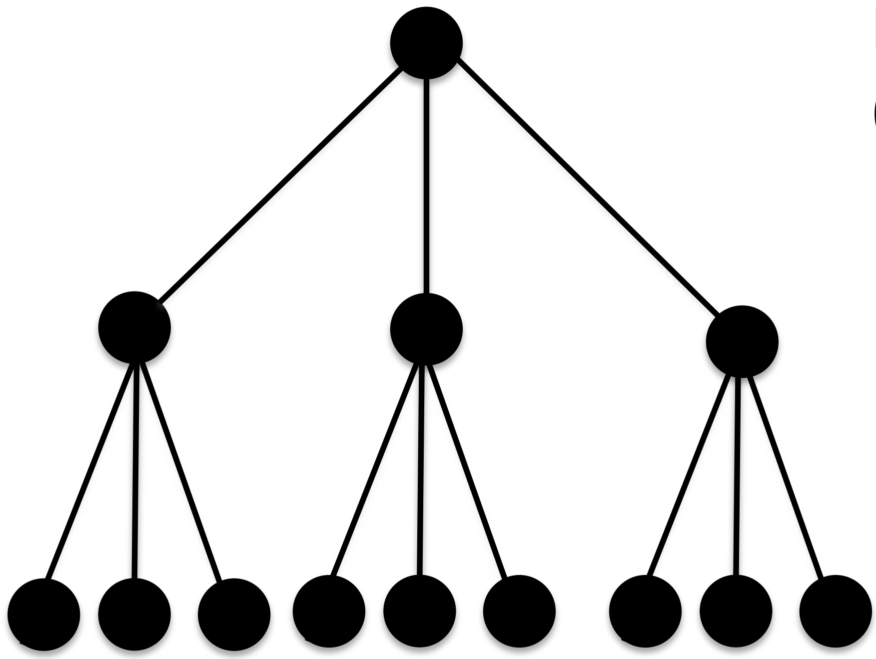
$$= c_1 f_1 + c_2 f_2 + \ldots + c_n f_n$$

Linear
Scoring
Polynomial

How could a computer play chess?

1. Human like
2. If-then rules: look & make most plausible move
3. Look ahead and evaluate
4. British Museum algorithm (exhaustive search)?

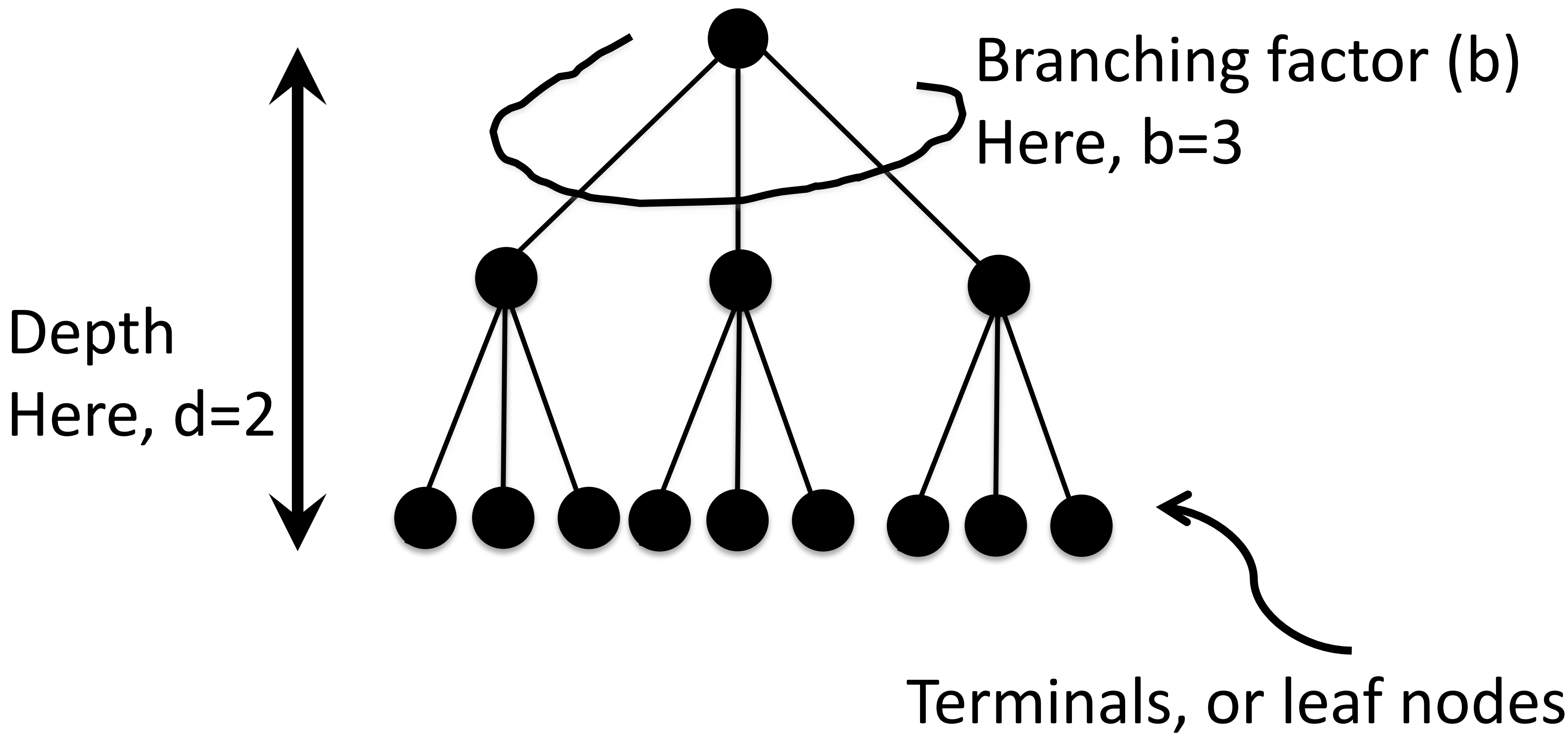
Vocabulary for game trees



Player 1 Moves
(one turn or 1 “ply”)

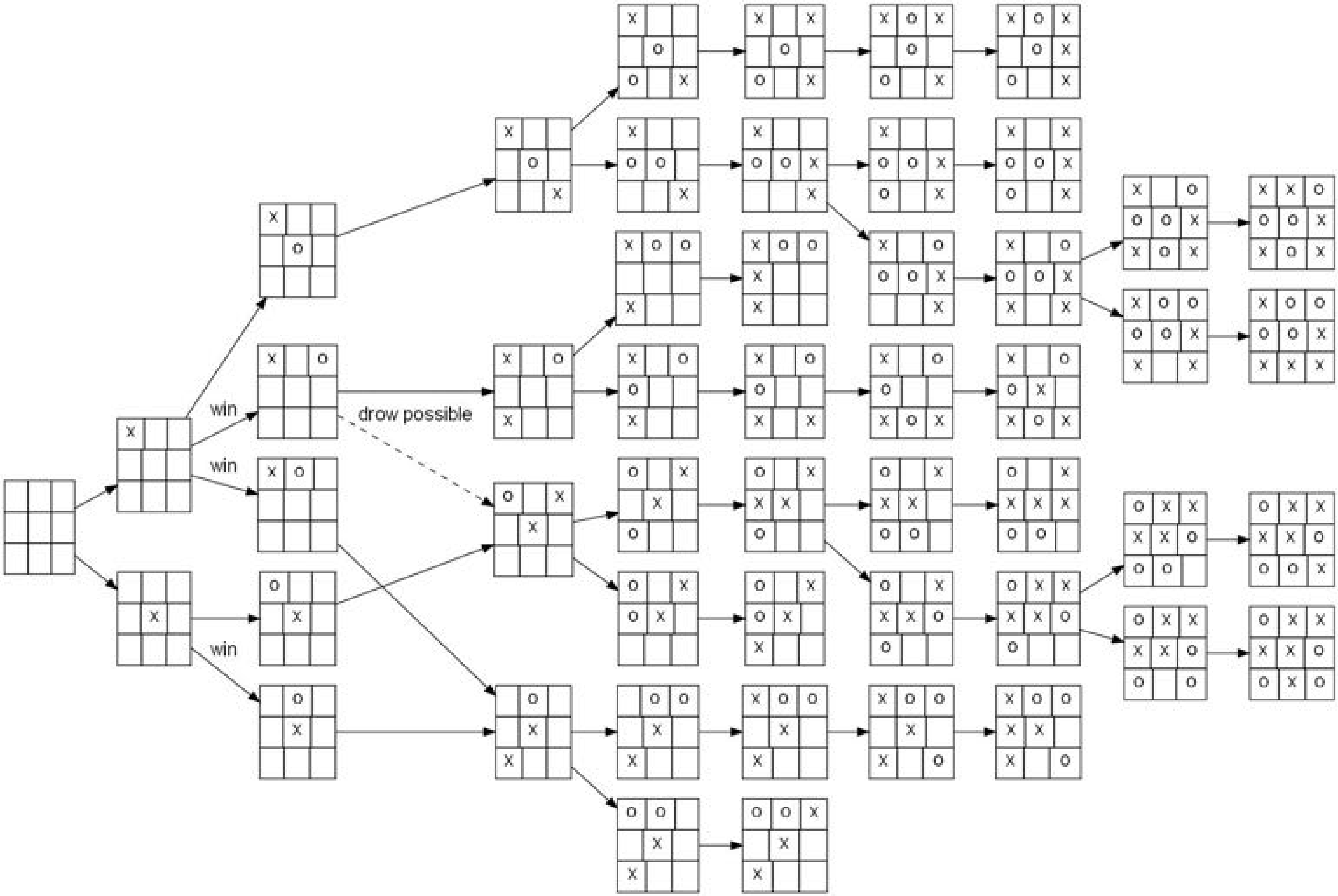
Player 2 Moves

Vocabulary for game trees



$b^d = 3^2 = 9$

Tic-tac-toe: complete game tree space



Could we use cloud computing to evaluate all possible chess moves in the same way?

10^{120} moves

Could we use cloud computing to evaluate all possible chess moves in the same way?

10^{120} moves

10^{80}	atoms in universe
$\pi \times 10^{7}$	seconds/year
10^{9}	nanoseconds/second
10^{10}	years since Big Bang

$\approx 10^{26}$ nanoseconds x 10^{80} atoms

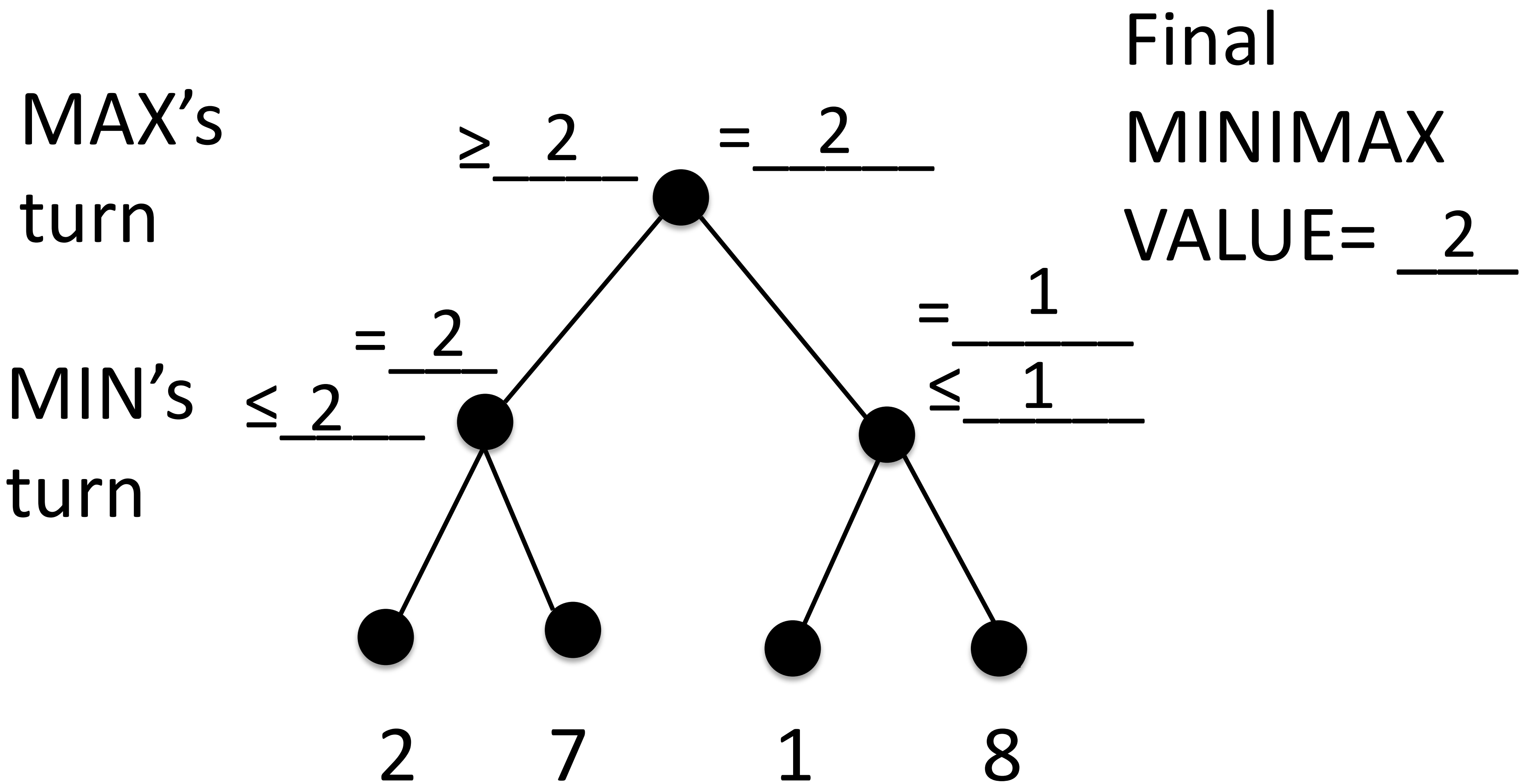
$\approx 10^{106}$ ops, each atom operating @ nanosecond speeds

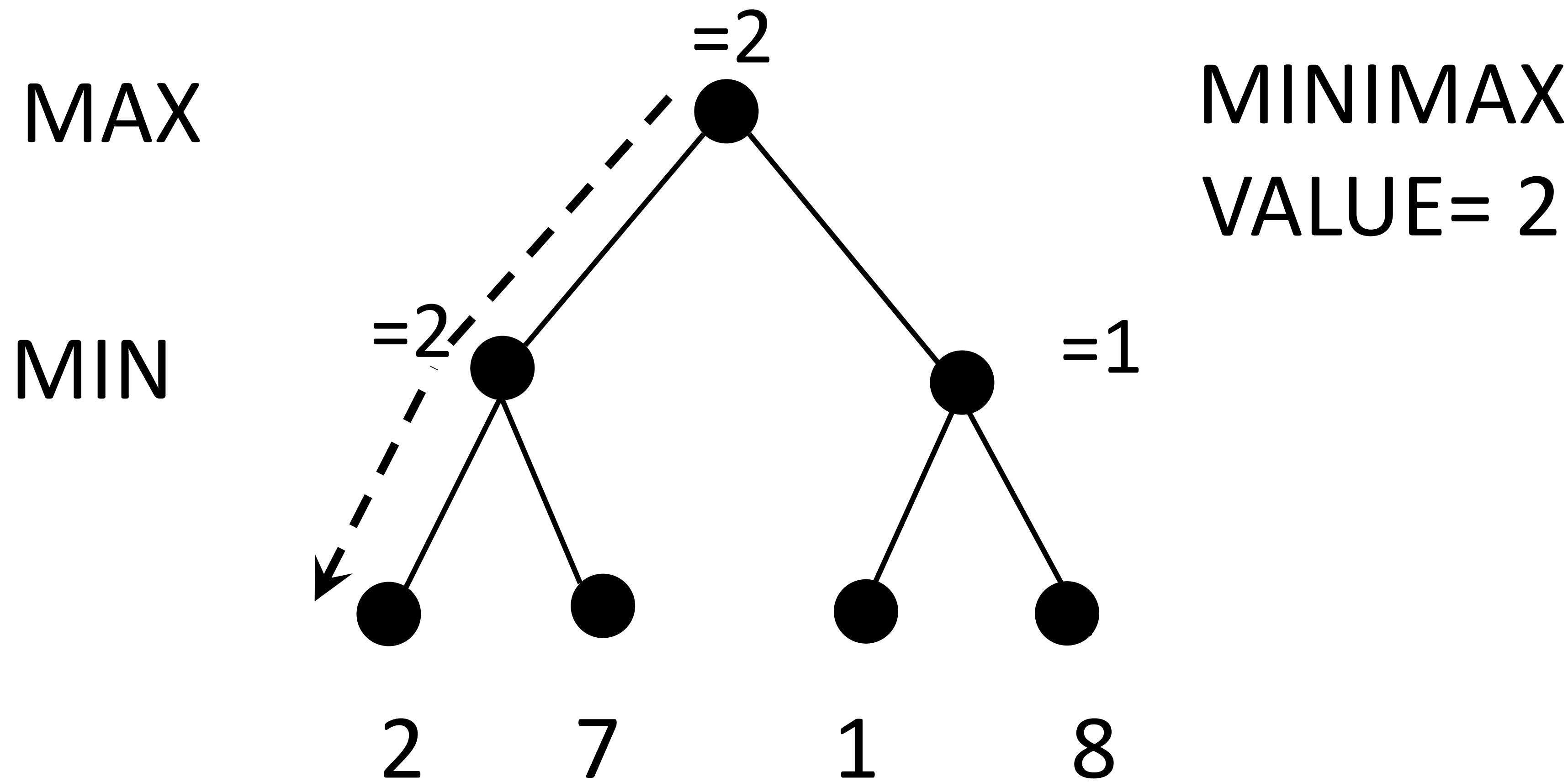
Not enough compute power for 10^{120} moves!

How could a computer play chess?

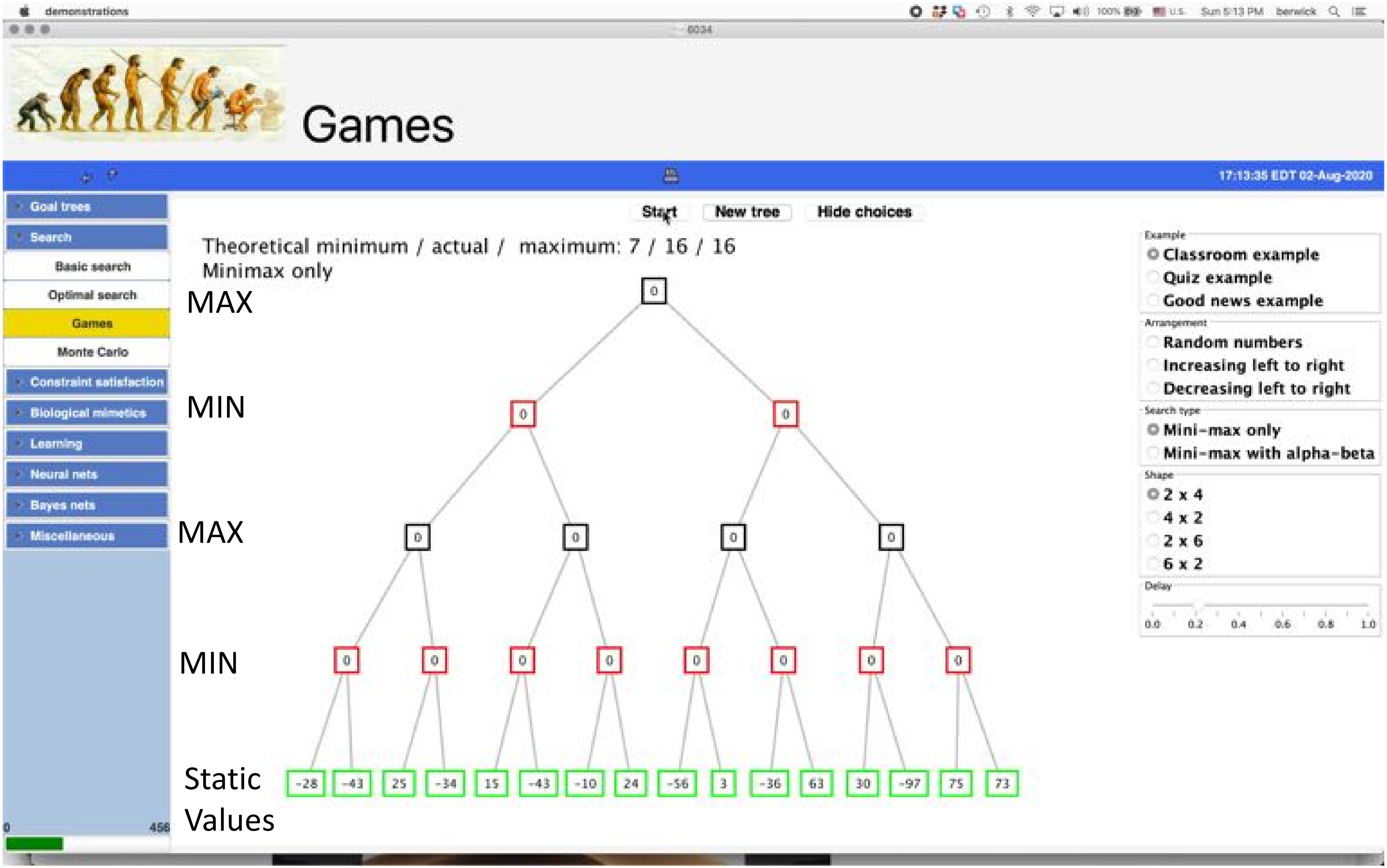
1. Human like
2. If-then– make most plausible move
3. Look ahead and evaluate
4. British Museum algorithm (exhaustive search)
5. Look ahead as far as possible

Minimax algorithm for searching game tree to find optimal move sequence



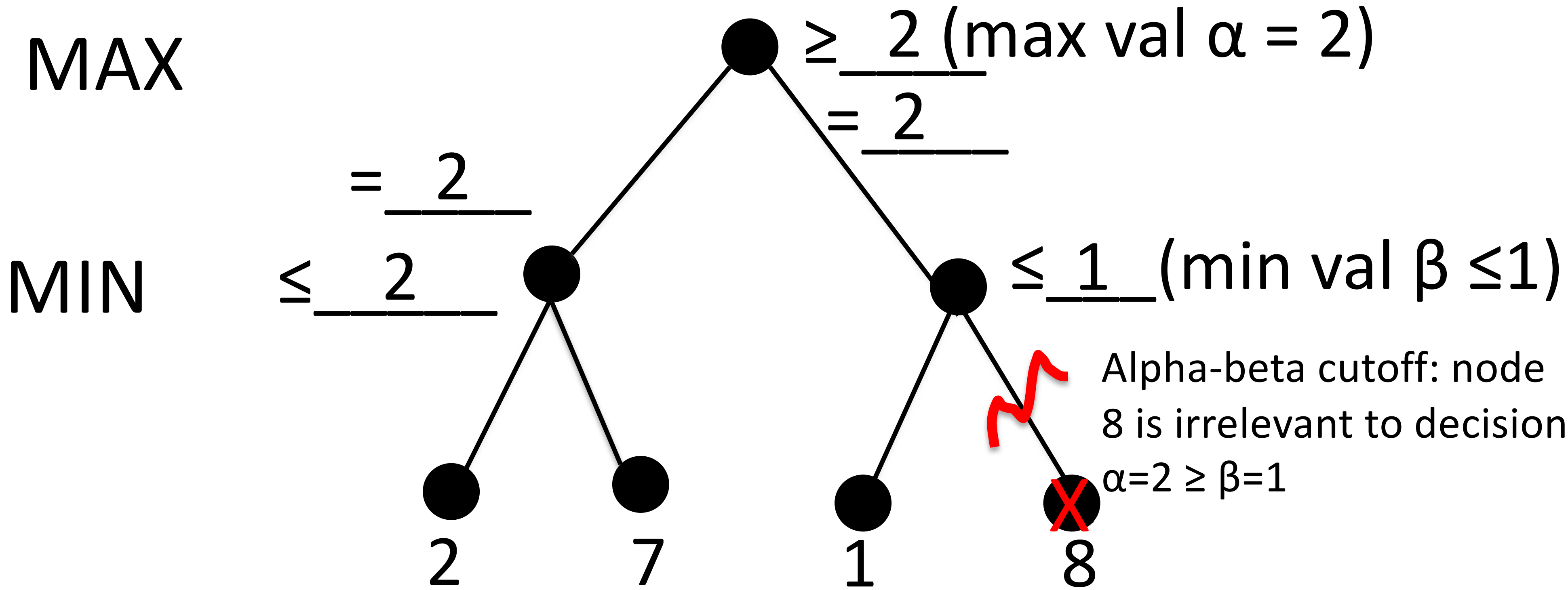


Note that the best minimax path is preserved under any transform of the leaf scores that preserves their rank ordering



demonstrations
Games
17:13:35 EDT 02-Aug-2020
Goal trees
Search
Basic search
Optimal search
Games
Monte Carlo
Constraint satisfaction
Biological mimetics
Learning
Neural nets
Bayes nets
Miscellaneous
Start
New tree
Hide choices
Theoretical minimum / actual / maximum: 7 / 16 / 16
Minimax only
MAX
MIN
MAX
MIN
Static Values
-28 -43 25 -34 15 -43 -10 24 -56 3 -36 63 30 -97 75 73
Example
Classroom example
Quiz example
Good news example
Arrangement
Random numbers
Increasing left to right
Decreasing left to right
Search type
Mini-max only
Mini-max with alpha-beta
Shape
2 x 4
4 x 2
2 x 6
6 x 2
Delay
0.0 0.2 0.4 0.6 0.8 1.0

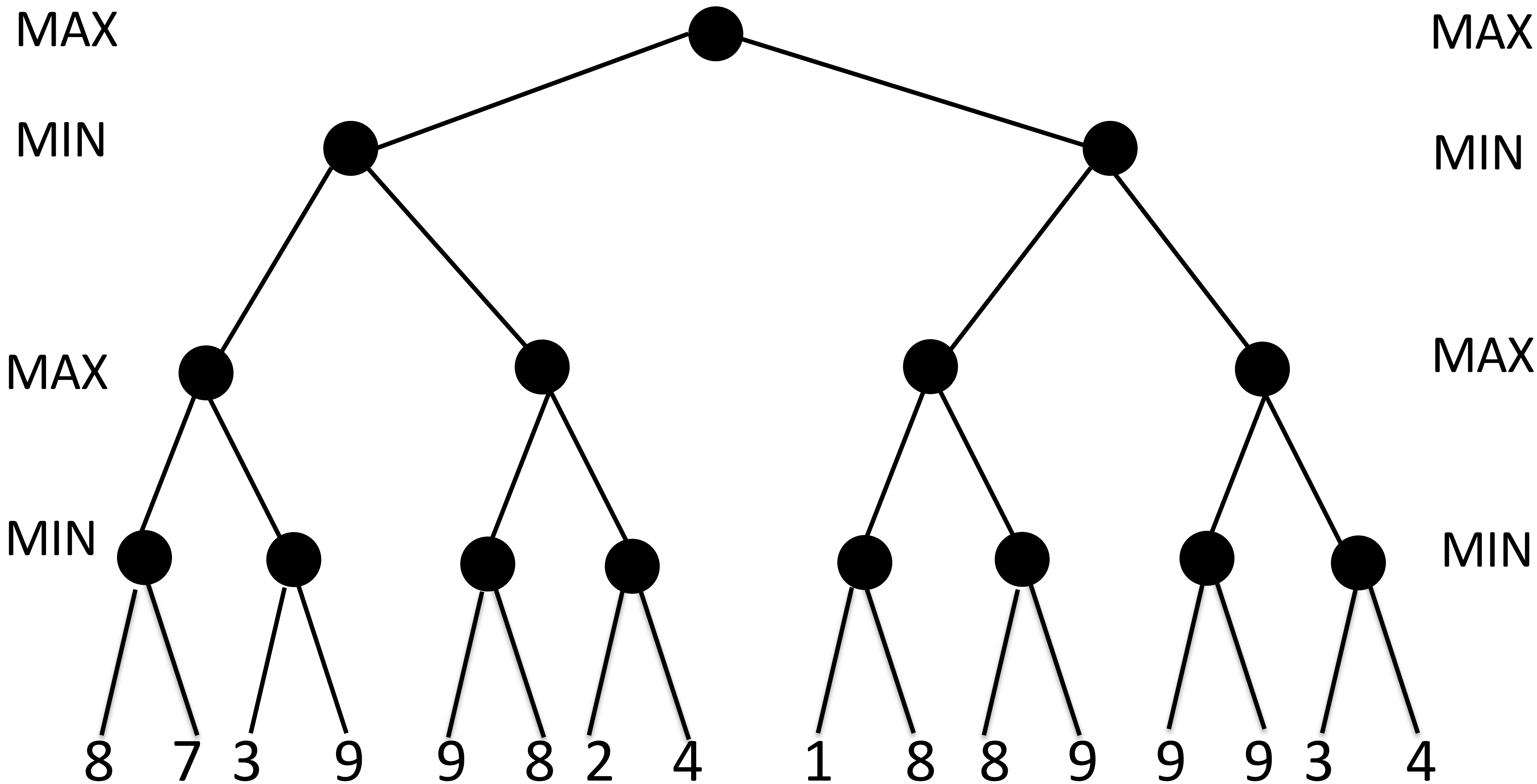
Minimax with alpha-beta (α-β) pruning



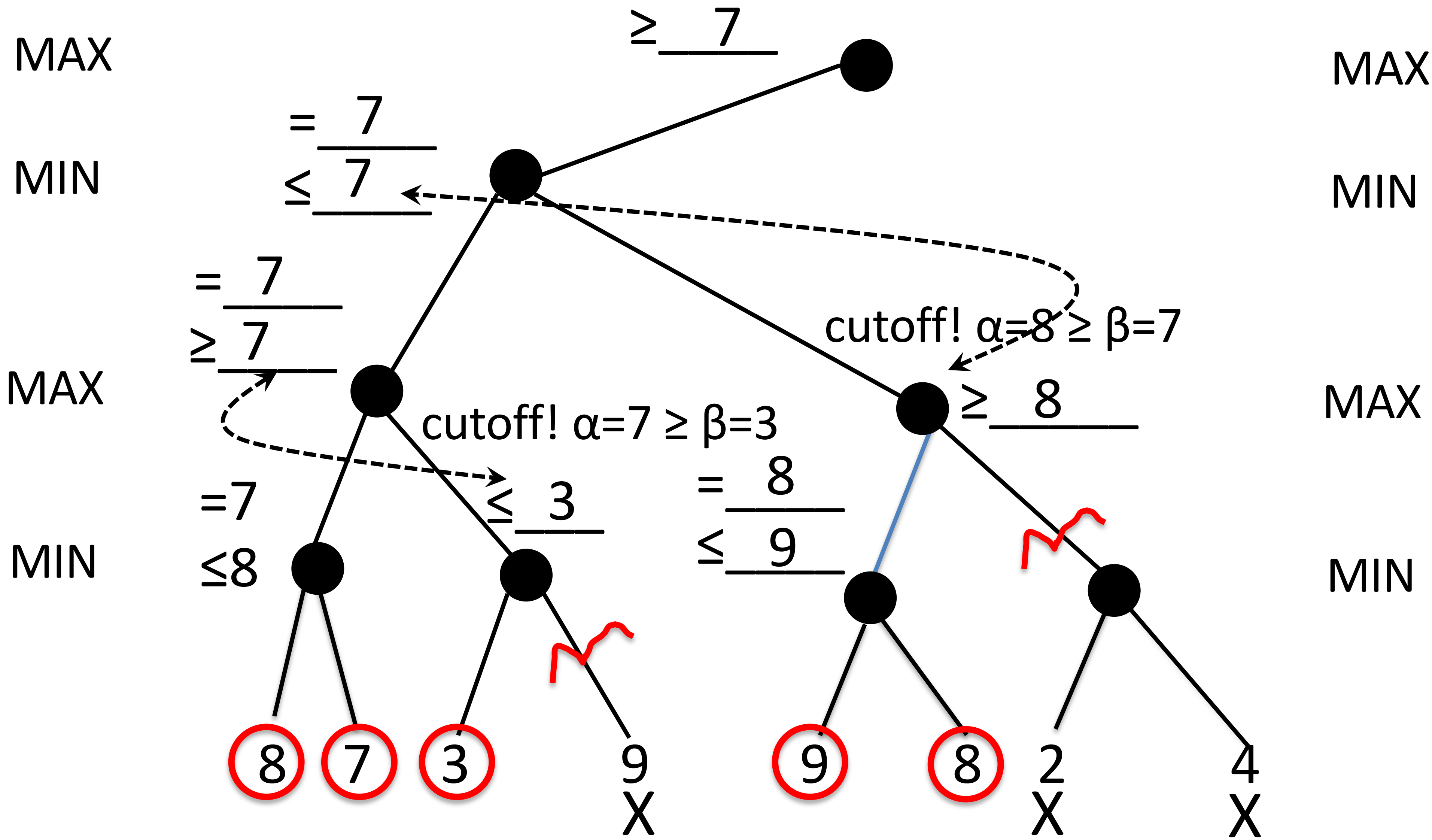
α value = MAX is guaranteed to gain at least this much, or ≥ α; a floor on MAX's gain.
β value = MIN is guaranteed to lose at most this much, or ≤ β; a ceiling on MIN's loss.
Note: [α < β], or floor < ceiling. Why?

Q: Alpha-Beta pruning value = Minimax value w/o pruning?
Ans: 2

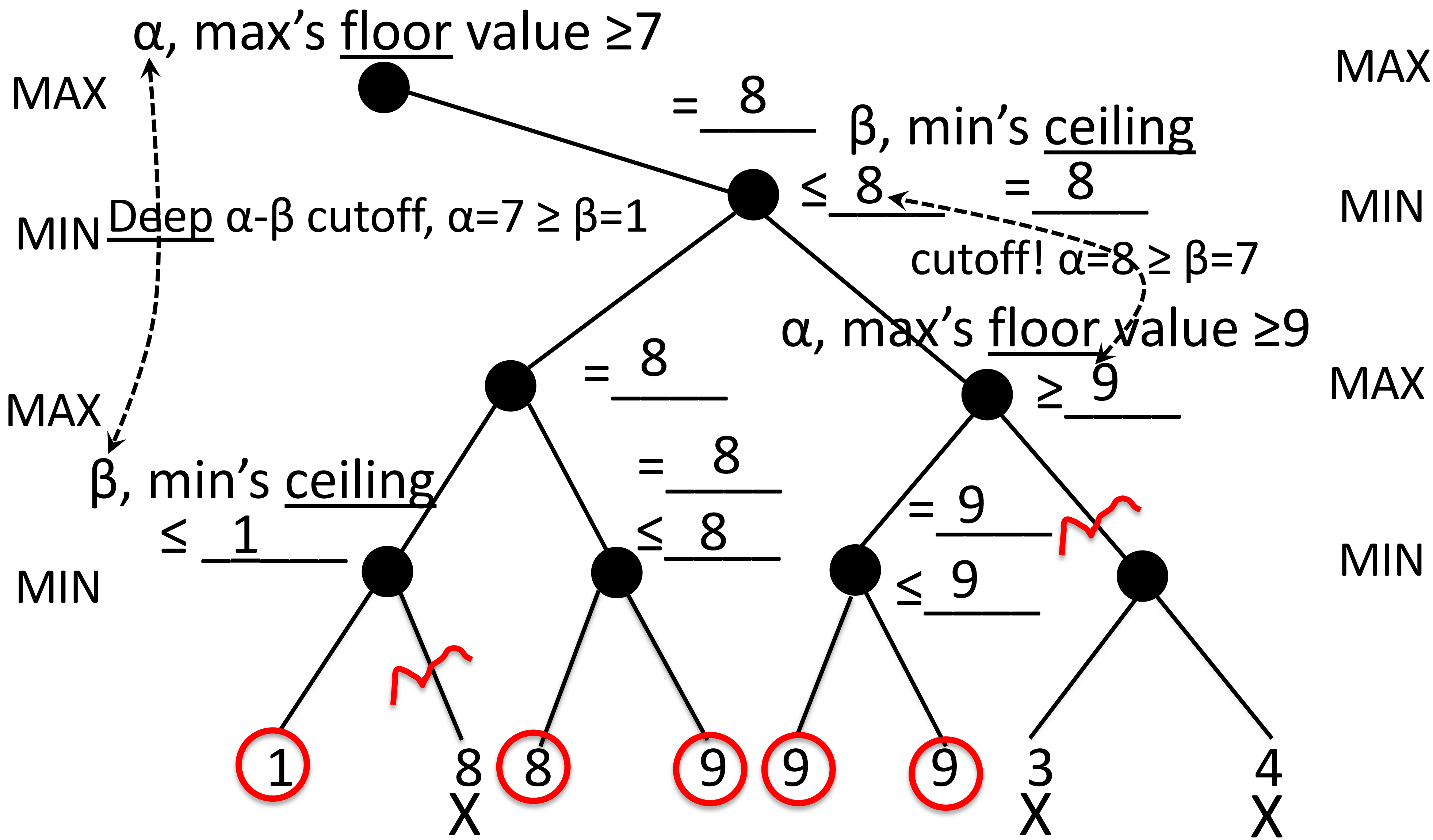
A deeper game tree illustrates how much pruning alpha-beta can do (b = 2, d= 3)



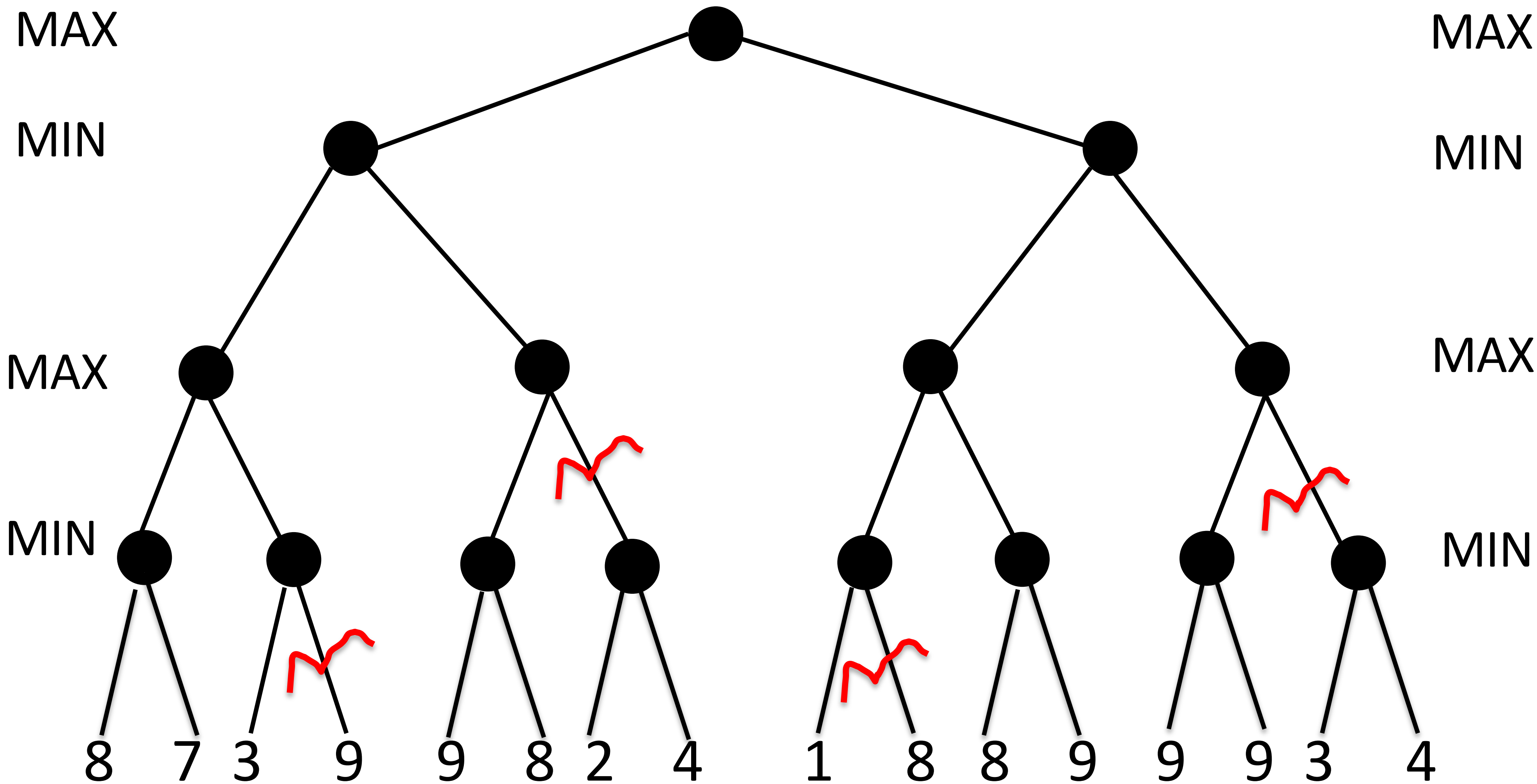
A deeper game tree illustrates how much pruning alpha-beta can do
First half of tree (max-α values; min-β values)



Second half of tree: cutoff whenever α≥ β

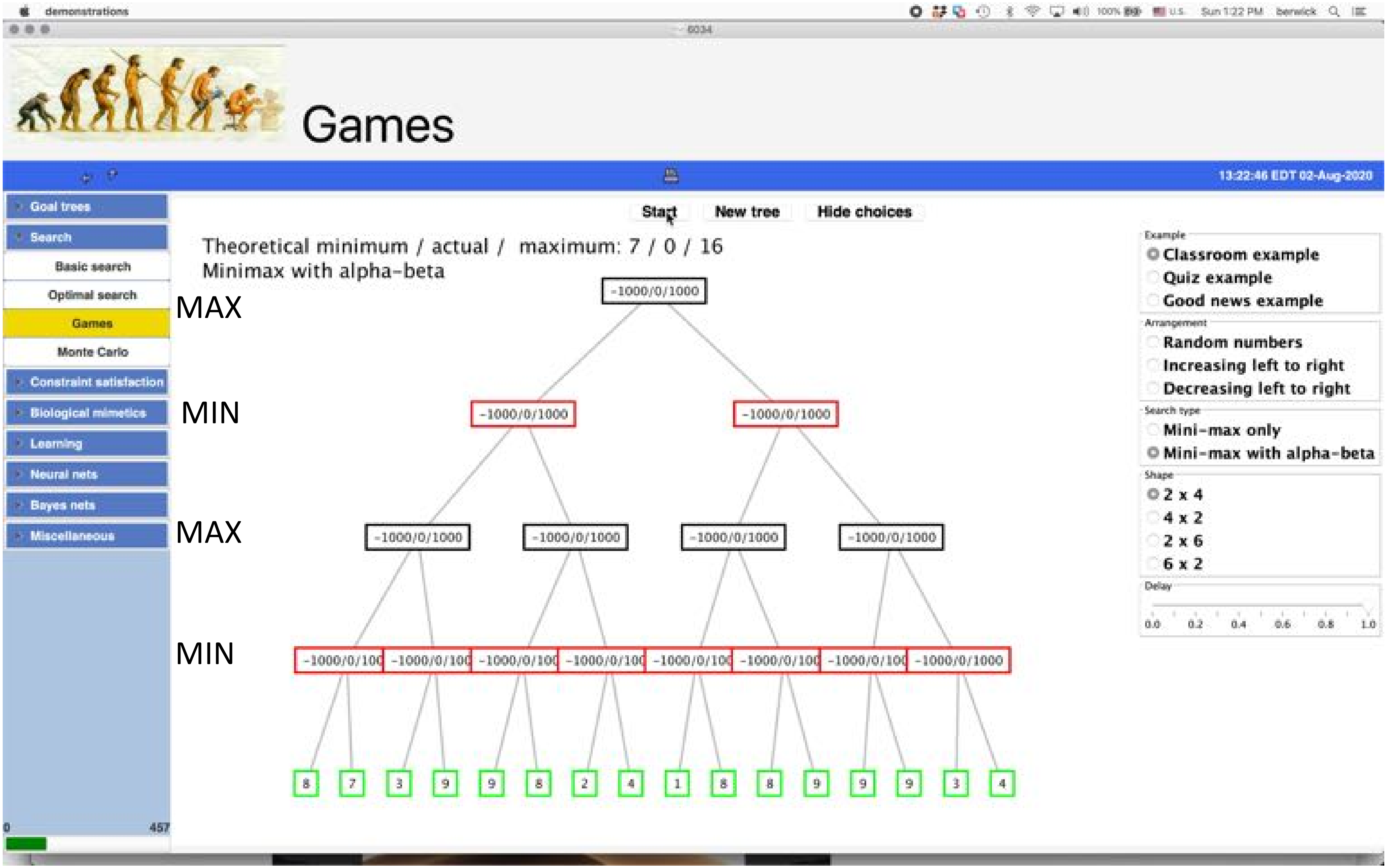


A deeper game tree



6 out of 16 static evaluations are <u>not</u> made

Dead horse principle (principle AWP)

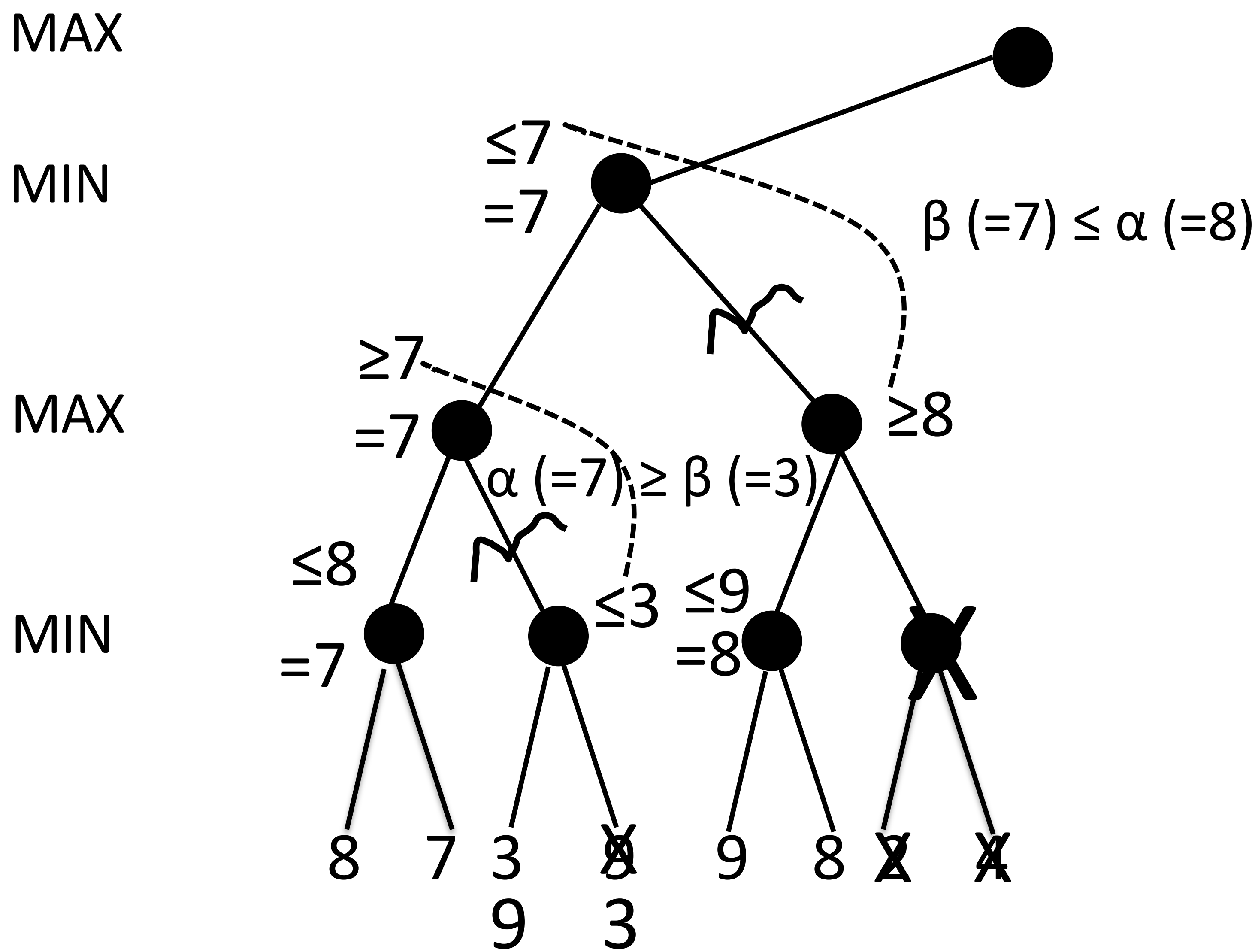


Games
13:22:46 EDT 02-Aug-2020
Goal trees
Search
Basic search
Optimal search
Games
Monte Carlo
Constraint satisfaction
Biological mimetics
Learning
Neural nets
Bayes nets
Miscellaneous
Start
New tree
Hide choices
Theoretical minimum / actual / maximum: 7 / 0 / 16
Minimax with alpha-beta
MAX
MIN
MAX
MIN
-1000/0/1000
8
7
3
9
9
8
2
4
1
8
8
9
9
9
3
4
Example
Classroom example
Quiz example
Good news example
Arrangement
Random numbers
Increasing left to right
Decreasing left to right
Search type
Mini-max only
Mini-max with alpha-beta
Shape
2 x 4
4 x 2
2 x 6
6 x 2
Delay
0.0
0.2
0.4
0.6
0.8
1.0

Ordering of static values can <u>greatly</u> affect alpha-beta pruning

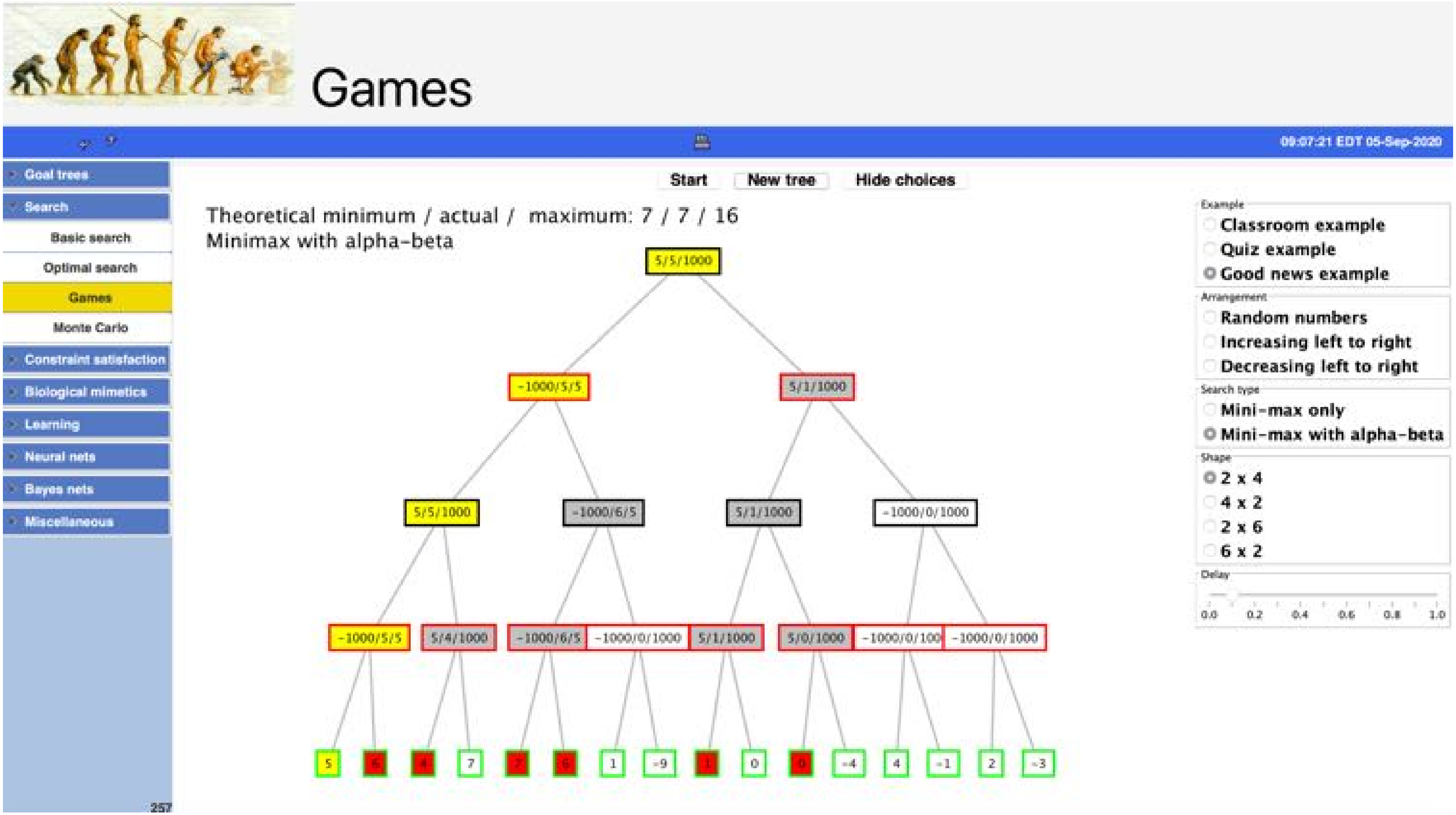
- If the most favorable successor nodes for <u>both</u> MAX and MIN are on the <u>LEFT</u> so we explore them <u>FIRST</u>, then this leads to maximal pruning
- If the most favorable successor nodes for <u>both</u> MAX and MIN are on the <u>RIGHT</u> so they are explored <u>LAST</u>, then there is less pruning, possibly none at all
- Maximal pruning (in terms of branching factor b and tree depth d is approx: $\underline{2\ b^{d/2}}$, so can search down 2x as far using α-β search in this "good news" optimal case, compared to std minimax

Prune whenever α ≥ β



What would happen to pruning if nodes 3, 9 reversed? 3 Not pruned!

Optimal (good news) α-β game tree pruning



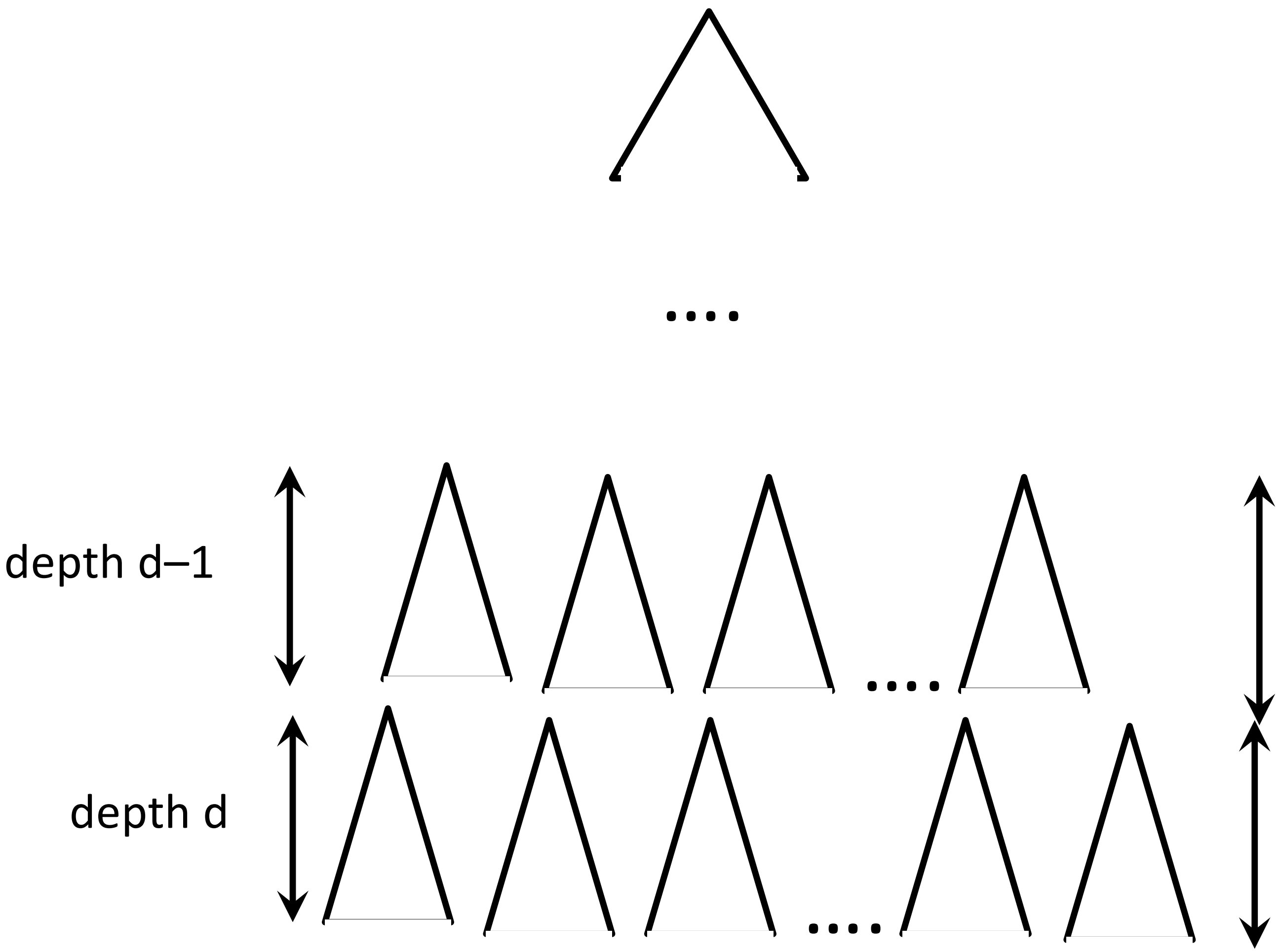
$\approx 2b^{d/2}$ $\approx$ maximum savings using α-β <u>if</u> optimally ordered game tree

But suppose we run out of compute time?
The branching factor depends on the game & board state – won't know for sure how deep we can go in 1 minute….
Suppose we are playing blitz chess?
Will we always have a (good) move at hand?
Let's combine this thought with the idea of optimally ordering the node evaluations…

Anytime algorithm

Martial arts principle

How not to run out of time



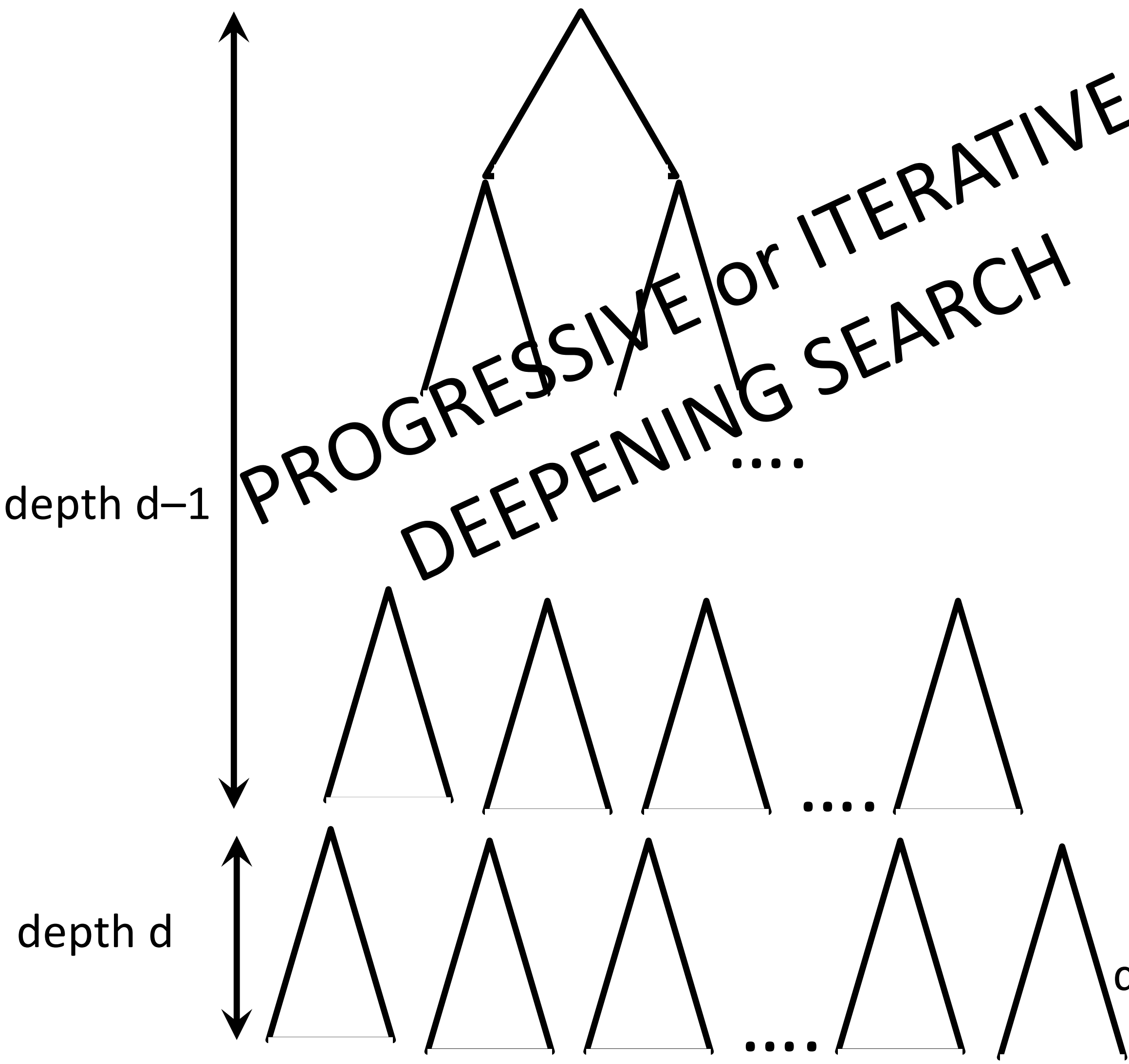
b^{d-1} nodes, or $1/b^{th}$ as much as bottom level (e.g., if branching factor is 15, only $1/15^{th}$ as much)

b^{d} nodes

So, we can have a move in hand at level d–1 for only $1/b^{th}$ of computation cost required to go all the way down to level d

We can do this recursively to great benefit…

How not to run out of time (ignoring α-β for now)



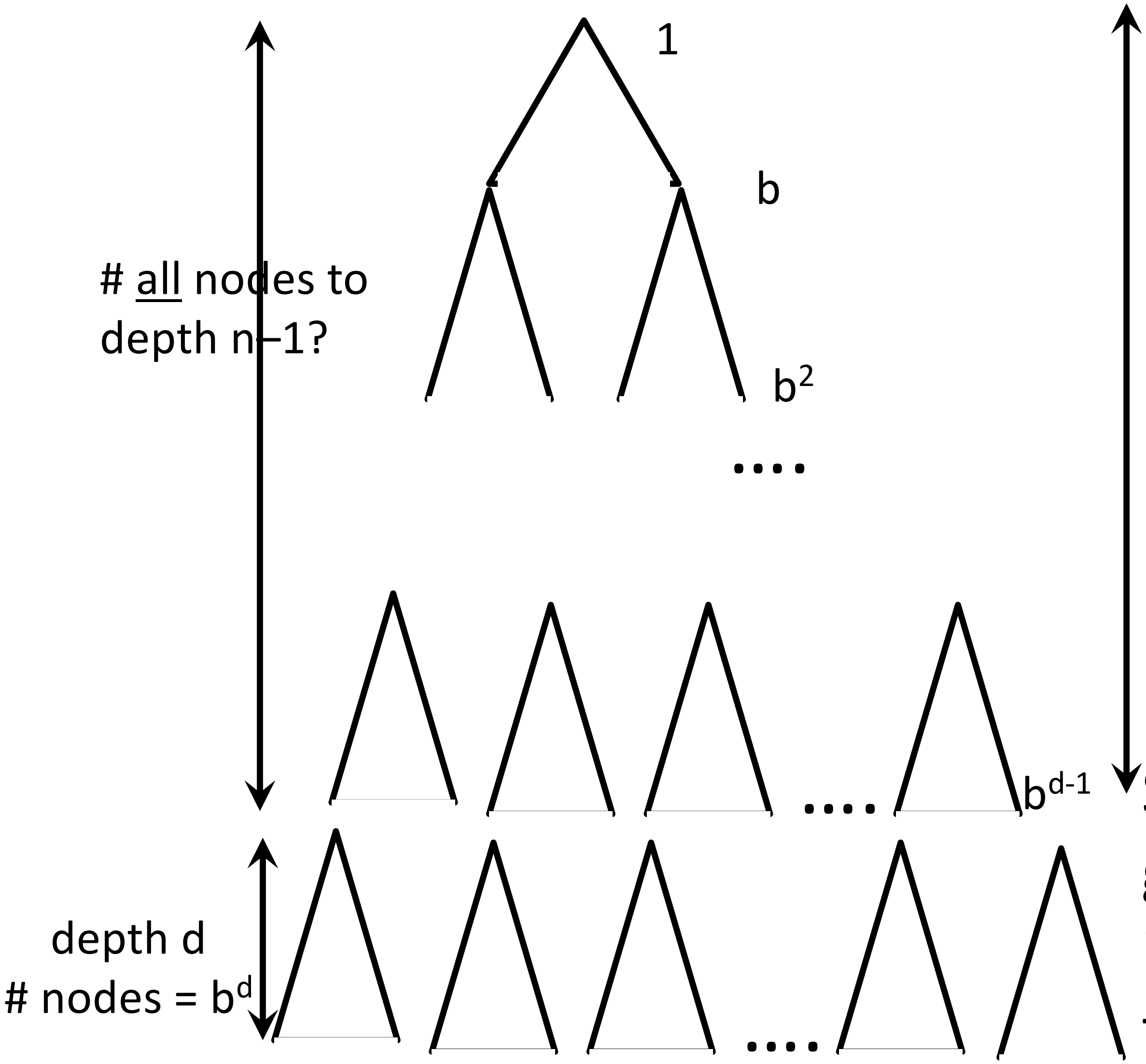
1. α-β EXPLORE tree to depth 1, or b nodes, and use the result to sort the best moves first; if there's time…
2. α-β EXPLORE tree to depth 2, or b^2 nodes, revisiting depth 1; sort the moves; if any time remains…

....

α-β EXPLORE tree to depth d–1, or b^{d-1} nodes, revisiting all depths to d–2; sort the moves; if any time remains…

α-β EXPLORE tree to depth d, or b^d nodes

Computational cost of progressive deepening?
How many total nodes S in game tree to depth d–1?



$$S=1+b+b^2+\ldots+b^{d-1}$$

$$bS= \quad b+b^2+\ldots+b^d$$

$$\Rightarrow bS-S= b^d-1$$

$$S(b-1)= b^d-1$$

$$S= \frac{b^d-\cancel{1}}{b-\cancel{1}} \approx b^{d-1}$$

So, search through <u>all</u> the game tree up thru depth d–1 is <u>only</u> time b^{d-1}, or $1/b^{th}$ of time to search to <u>full</u> depth d

Progressive (Iterative) deepening search

- Optimal DFS method (redundant use of space, but doesn’t lose on arbitrary depth **d** trees)
- Earlier searches tend to improve the commonly used heuristics, so that a more accurate estimate of the score of various nodes at the final depth search can occur
- Because early iterations use small values for **d** they execute extremely quickly. This allows the algorithm to supply early indications of the result almost immediately, followed by refinements as **d** increases

Deep Blue, IBM, circa 1997.
≈ 10^8 static evaluations per sec

Deep Blue = minimax + alpha-beta + progressive deepening+ parallel computing + opening book + end game + secret sauce =

uneven tree development


Gary Kasparov, then current world champion,
vs. Deep Blue, 1997
Is this intelligence?
Is this human intelligence?

Donald Byrne vs Bobby Fischer (black) 1956, move 17, Queen sacrifice

Gold star ideas today

- Dead horse principle, aka “AWP”: α-β search
- Martial arts principle – use adversary’s strength against them: progressive deepening
- Anytime algorithms: progressive deepening
- Simple ≠ Trivial: sometimes, bulldozers work