Die Digitale Revolution

Schachprogrammierung

Internet

3D-Drucker

Quants

Singularität

Freie Universität Berlin

Schachprogrammierung

27.02.14

Johannes Polster

Das Spiel der Könige

- Sehr altes Spiel: Entstehung vor 1500 Jahren
- Weltberühmt
- Strategisches Spiel
  - Kein Glück, Intelligenz, Kreativität, Phantasie, Erfahrung und Intuition
- Schachspielende Computer → Intelligente Computer?
- Herbert Simon: *„if one could devise a successful chess machine, one would seem to have penetrated to the core of human intellectual endeavor.“*

Drosophila der KI

- Klare, einfache Regeln, klares Ziel
- Trotzdem komplexes Spiel
- Bekanntes, beliebtes Spiel
- Vorhandene Infrastruktur:
  - Schachturniere
  - Elorang
  - Aufzeichnungen vergangerner Partien

Geschichte der Schachprgrammierung

- 1769 - 1836: Schachtürke: Automat
- 1890: Machine die Turm, König vs König Endspiele lösen konnte
- 1942 – 1945: Konrad Zuse
- 1950 Claude Shannon
- 1951: Alan Turing
- 1957: Wette von Herbert Simon
- 1966: Mac Hack
- 1968: Wette von Levy
- 1976: Microchess
- 1997: Deep blue

Kasparov vs IBM Deep Blue

- 1996: erster Schachcomputer der amiterenden Weltmeister schlägt
- 1997: Turnier über 6 Partien, 3.5 zu 2.5 für Deep Blue
- 100 Millionen – 200 Millionen Stellungen pro Sekunde
- Kosten: 5 Millionen Dollar
- IBM lehnt rematch ab

Schachprogrammierung

- Endliche Anzahl an möglichen Partien → Problem gelößt?
- 42 Züge pro Spieler: 84 Züge
- 38 mögliche Züge pro Zug (Annahme)
- → $38^{84} = 10^{134}$ Züge müssen ausgewertet werden
- Typ A Lösung: nicht so weit Vorrausschauen (brute force)
- Typ B Lösung: nur bestimmte Äste verfolgen (menschlicher)

Aufbau von Schachprogrammen

- Zuggenerator
  - Brettrepräsentation
- Bewertung
  - Evaluation der Position

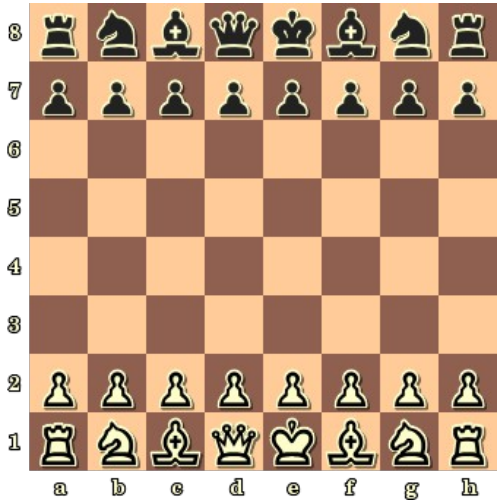
Suche

- Finden des besten Weges im Spielbaum (Mini Max Algorithmus)

Zuggenerator

- Benötigt Repräsentation des Bretts
- **Bitboards:** Brett als 64 bit binäre Zahl
- Wenn Figur auf Feld 1 ansonsten 0
- 0000000011111111000000000000000000000000000000000000000000000000

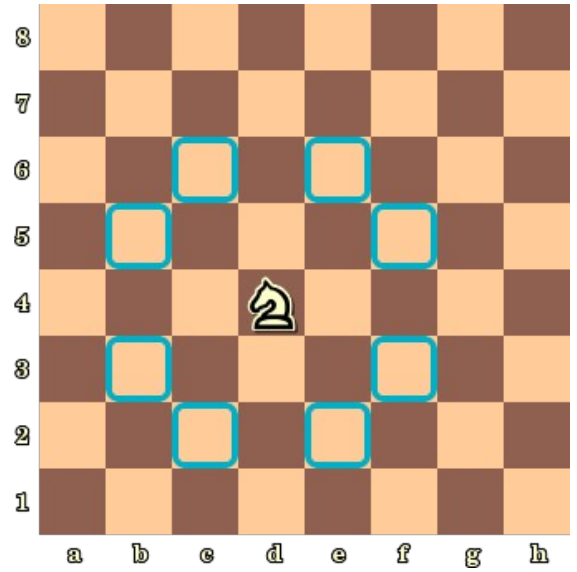
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0



Beispiel Zugberechnung Springer

- Bitboard mit mögliche Zielfelder (Z)
- Bitboard mit alle Felder die nicht von weiß besetzt sind (NW)
  - NW = not (B1 or B2 or …or B6)
- Z and NW

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0
0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0

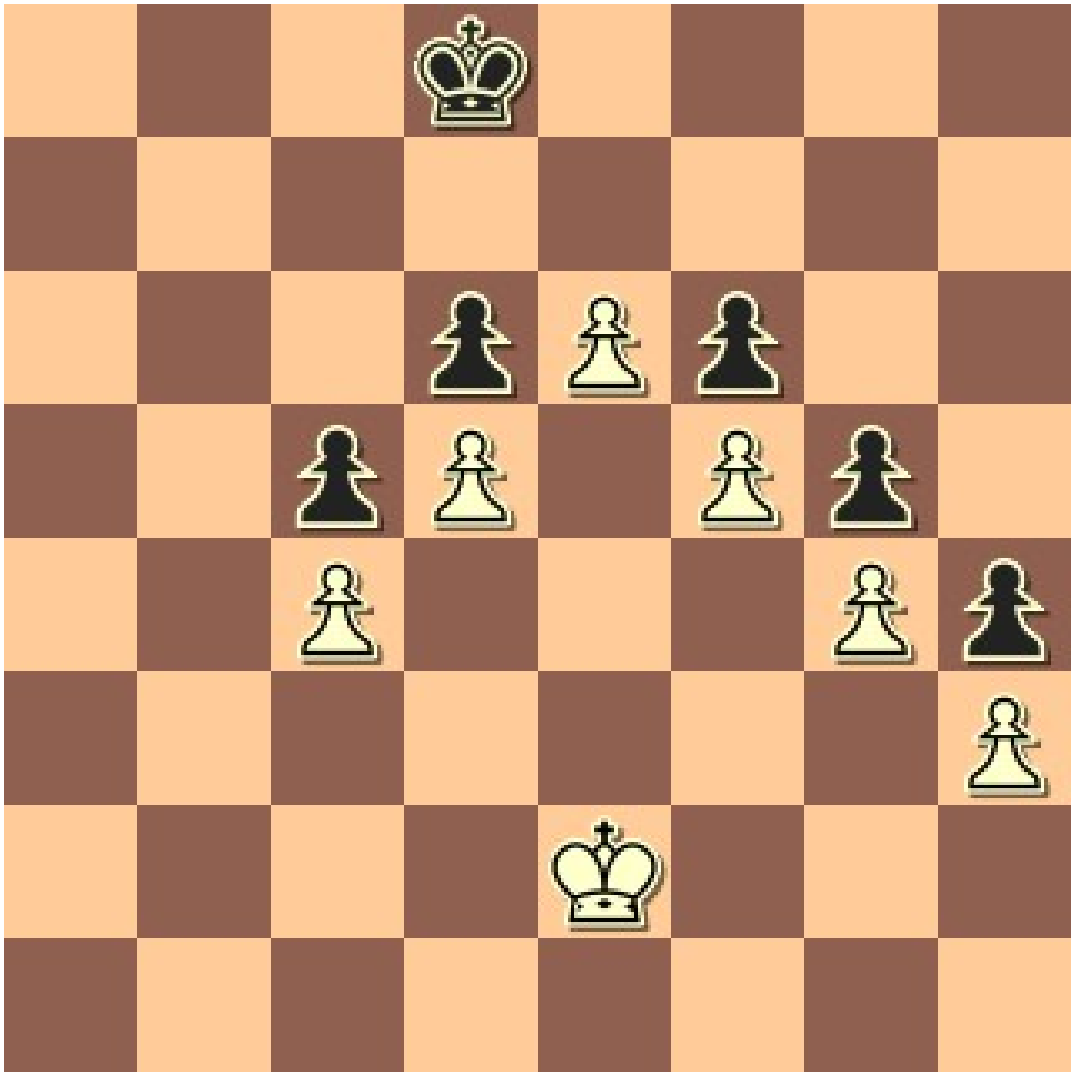


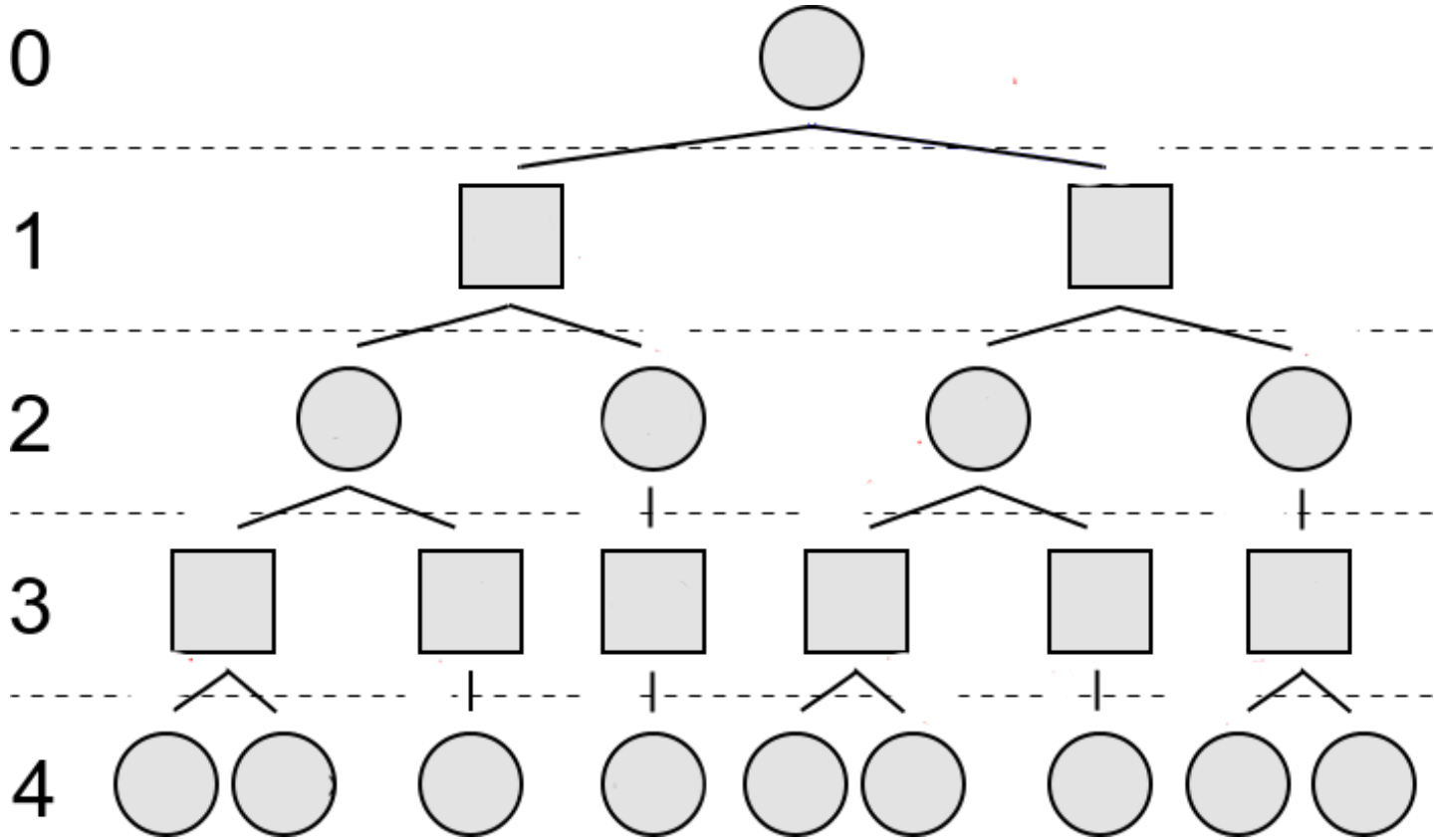
Bewertungsfunktion

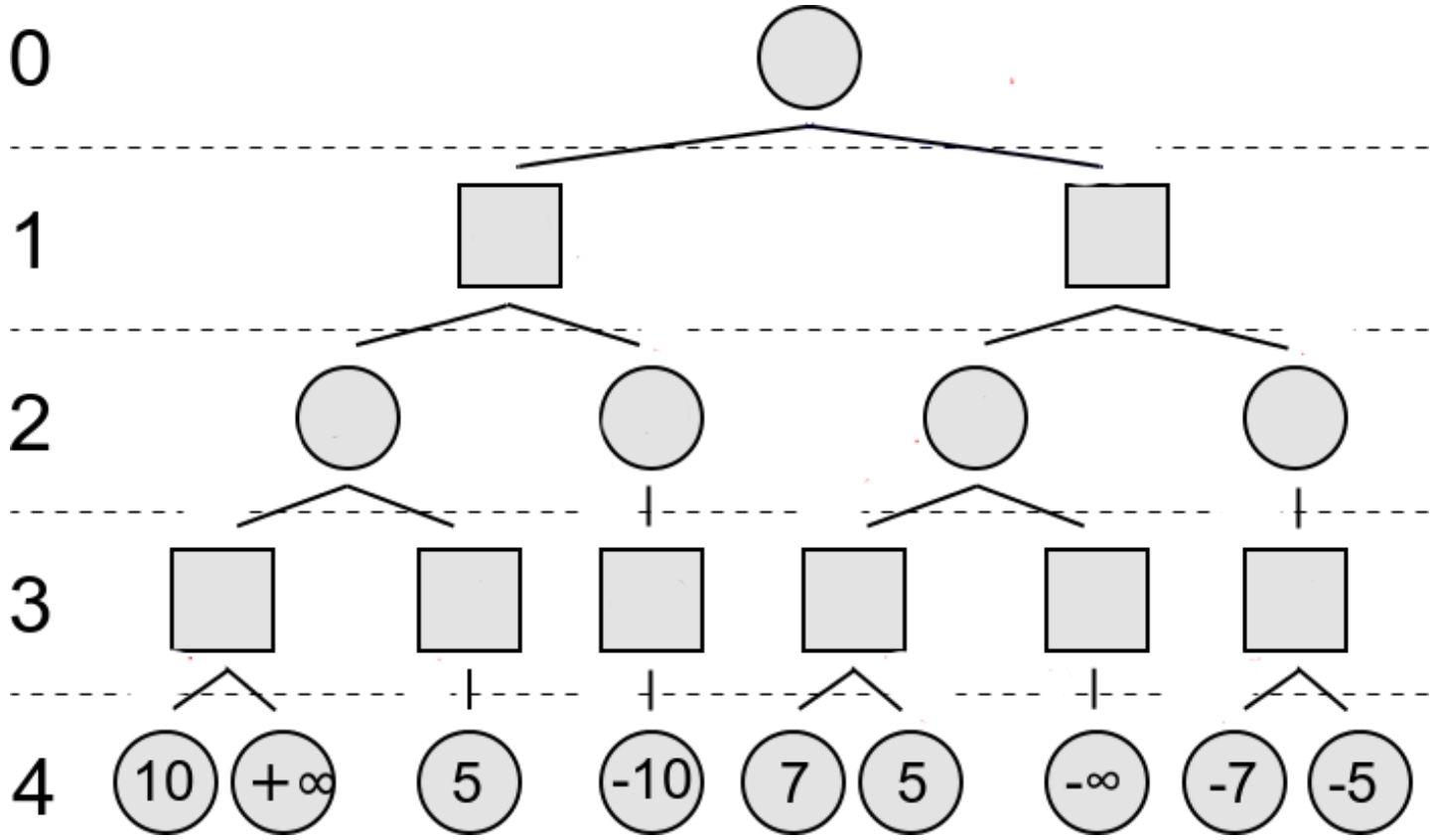
- Versuchen zu Berechnen wie gut eine gegebene Stellung ist
- Jede Figur hat einen Wert
- Bewegungsspielraum der Figur
- Position
- Königssicherheit
- Deckung durch Bauern...
- Boni bzw. Mali aufsummieren und beide Werte von einander abziehen
- Bsp: Weiß: 8482 Scharz: 8497 → 8482 – 8497 = -15 → Vorteil für Schwarz!

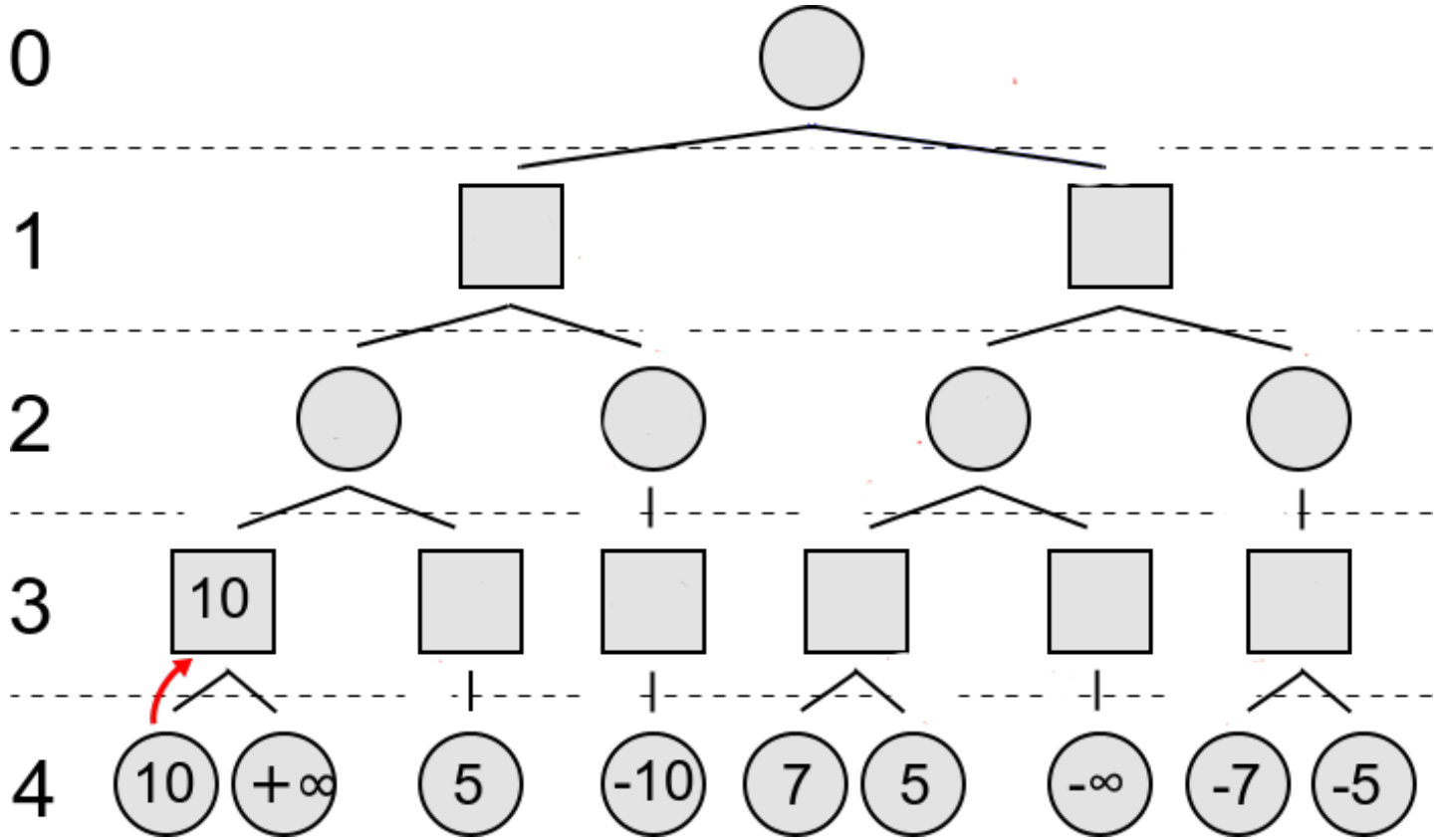
Suche: Mini Max

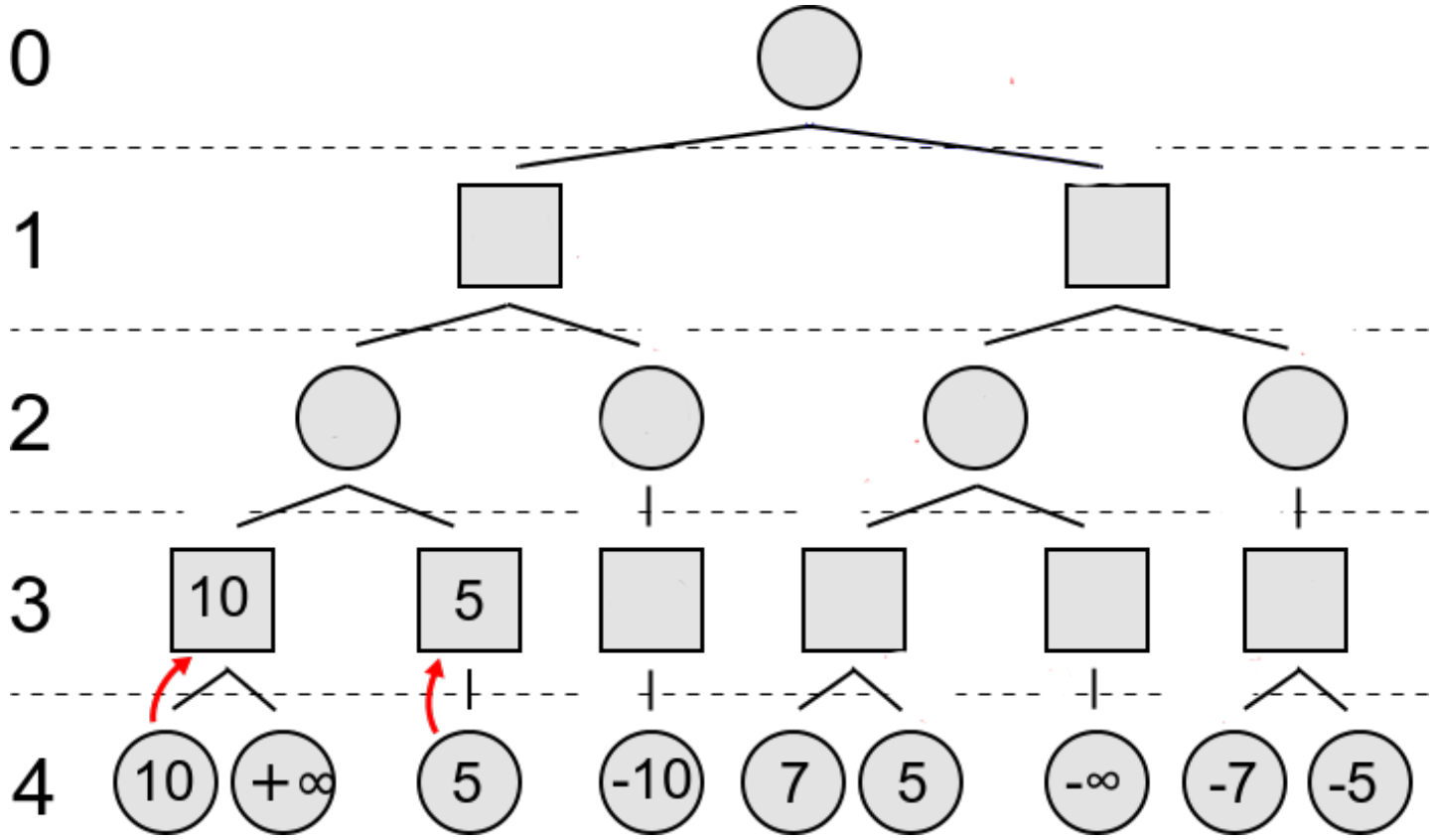
- Finden des besten Zuges (Annahme: Gegner zieht auch am besten)
- Wird in allen erfolgreichen Schachprogrammen eingesetzt
- Zusammen mit Alpha-Beta Pruning

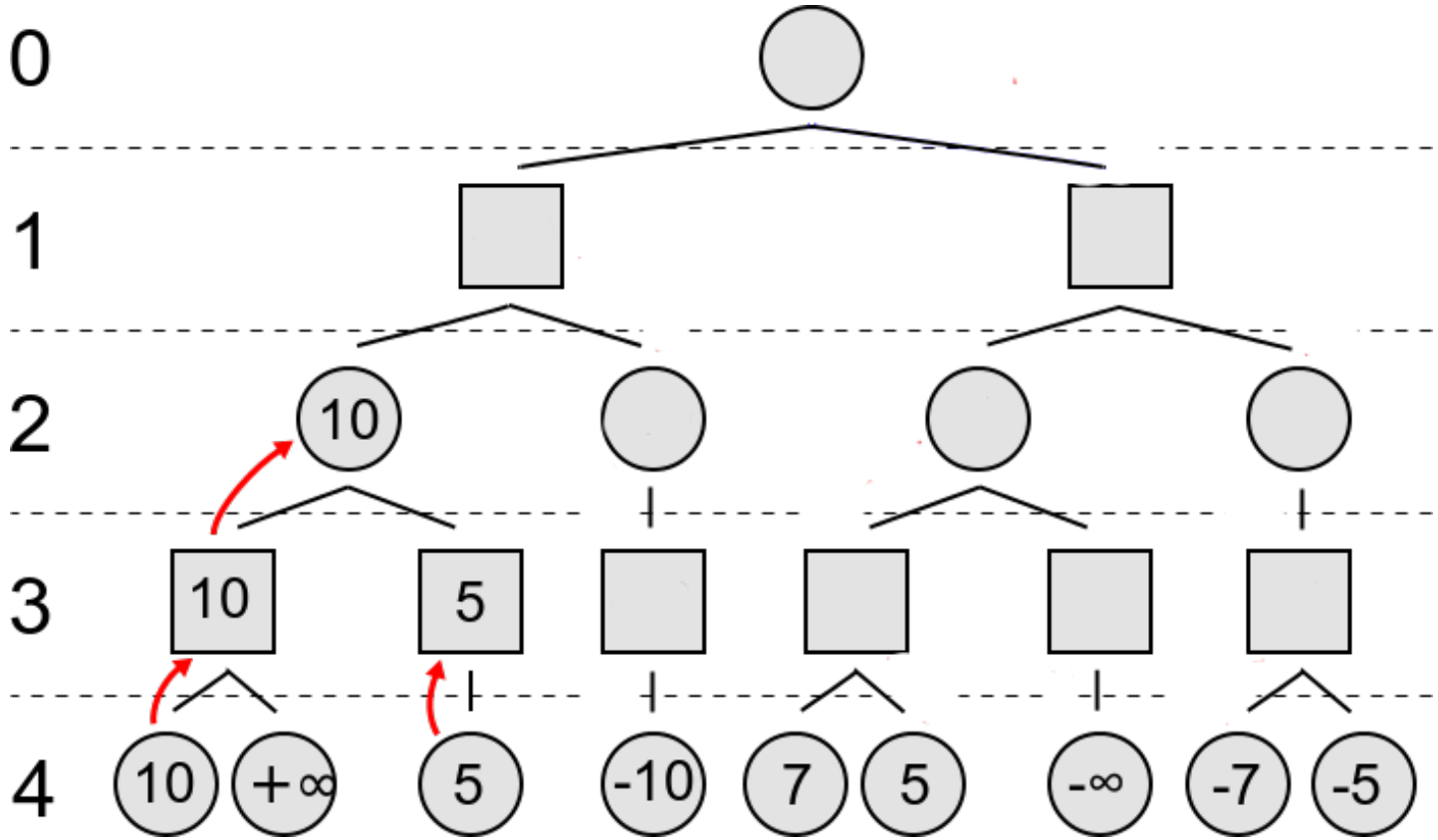


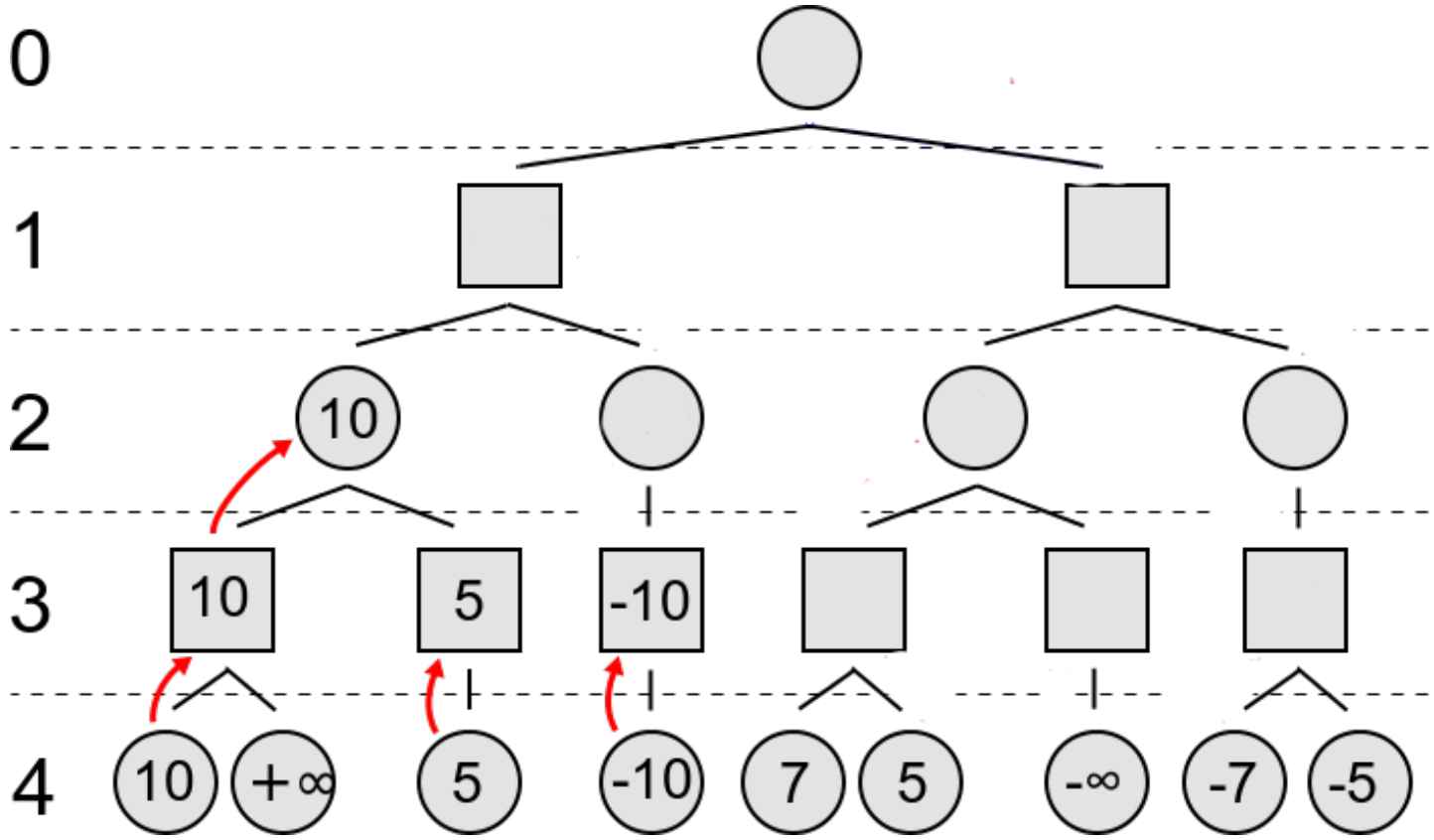

0
1
2
3
4

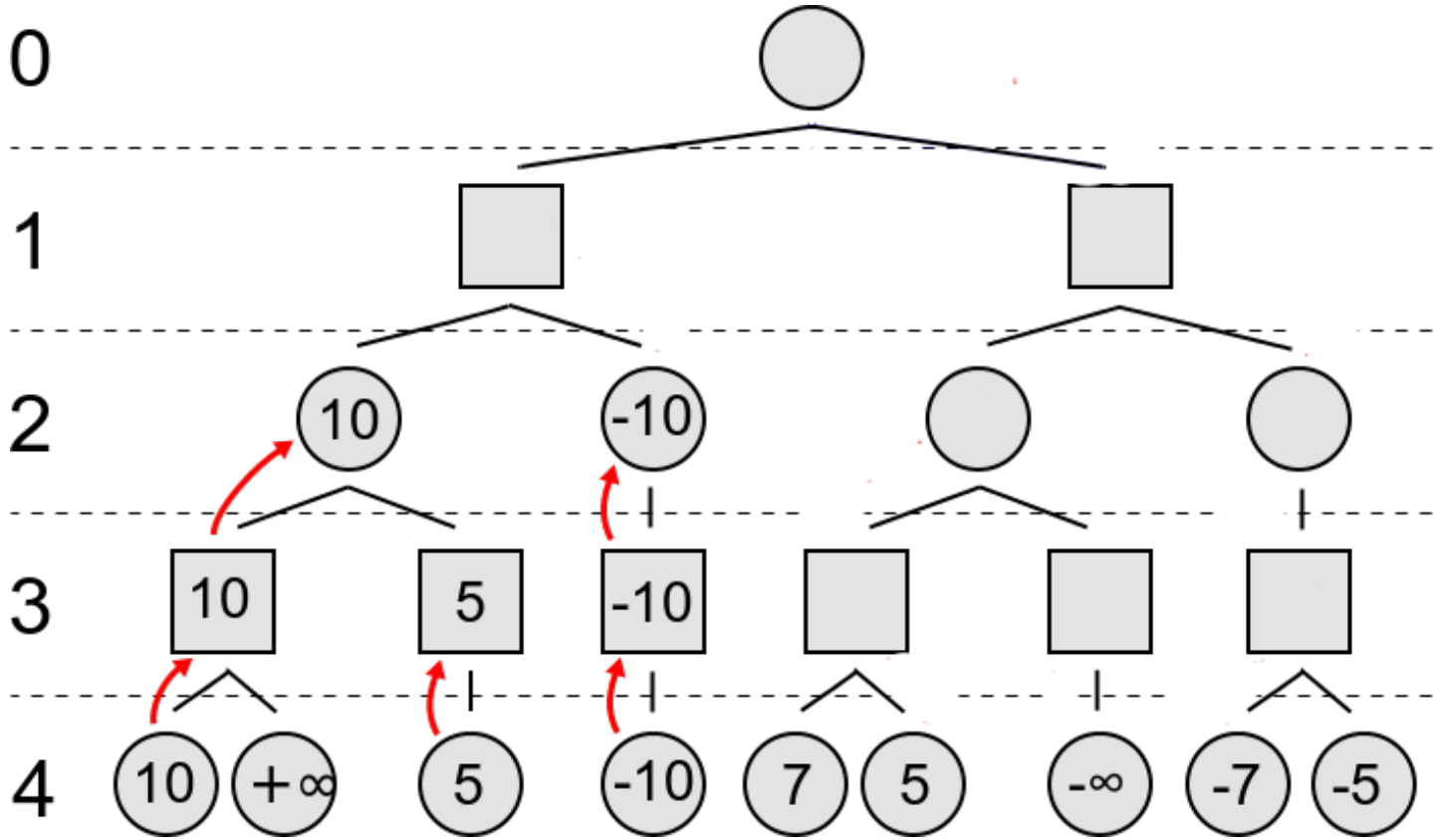

0
1
2
3
4
10
+∞
5
-10
7
5
-∞
-7
-5

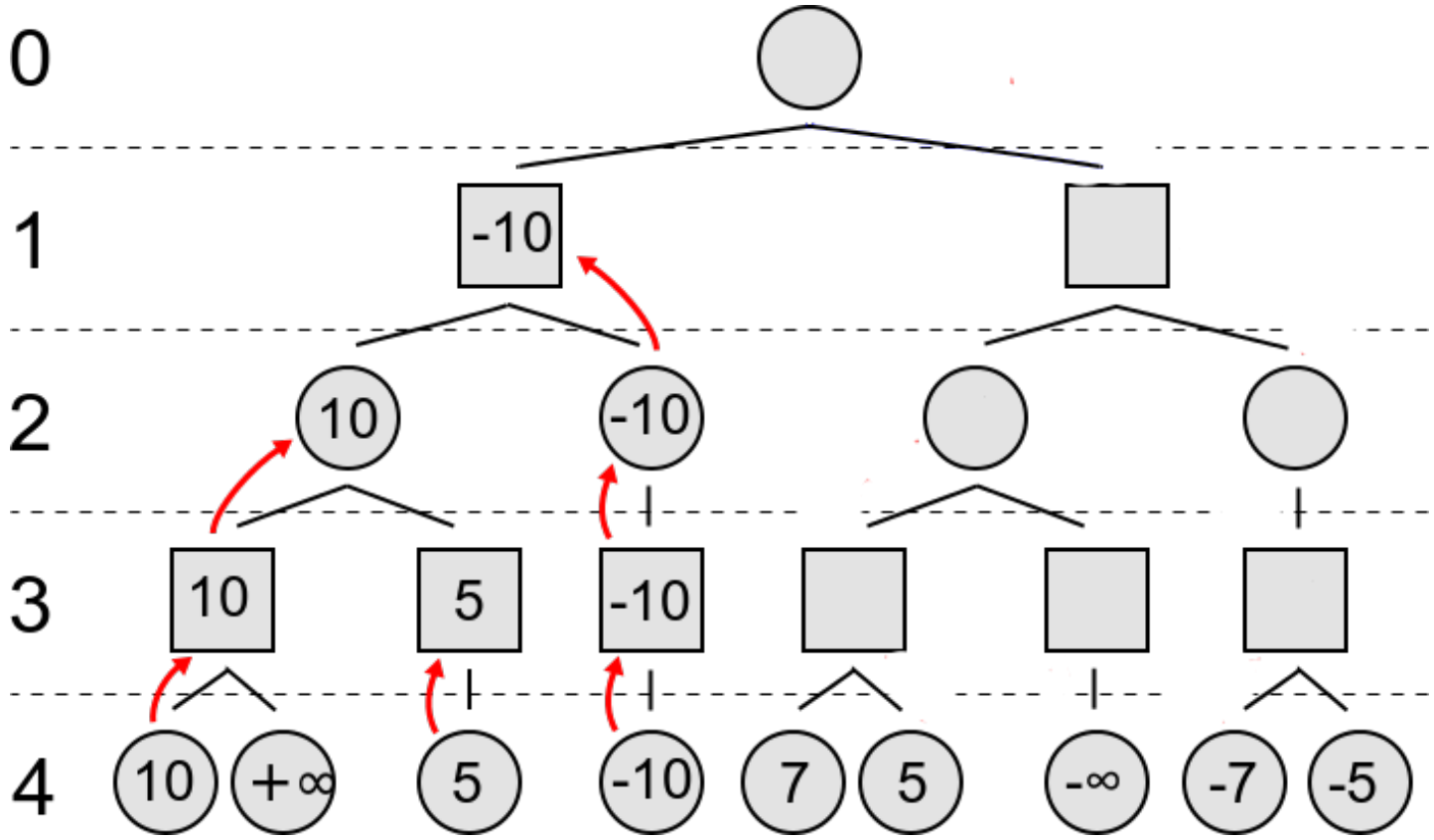

0
1
2
3
4
10
10
+∞
5
-10
7
5
-∞
-7
-5

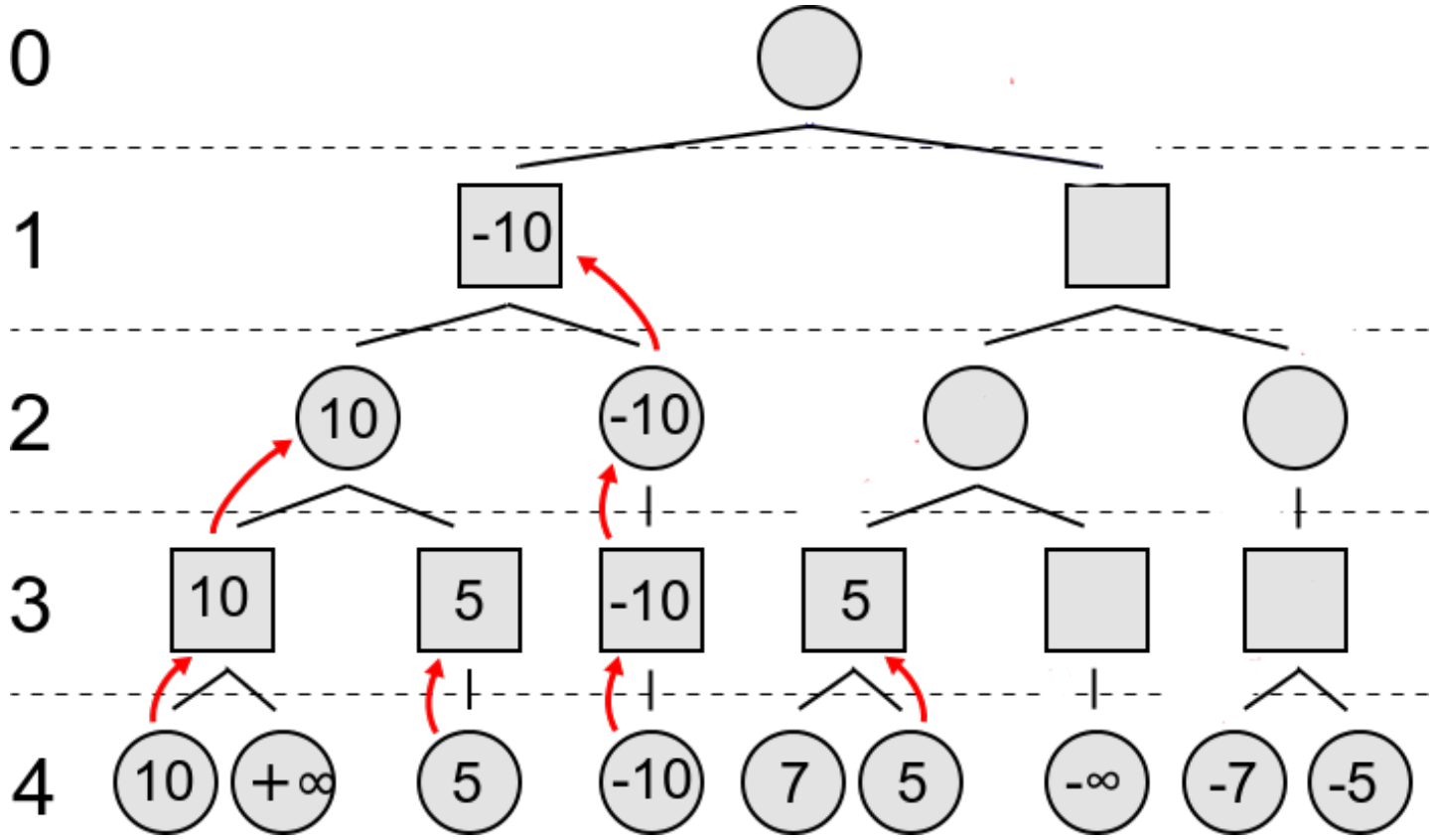

0
1
2
3
4
10
5
10
+∞
5
-10
7
5
-∞
-7
-5

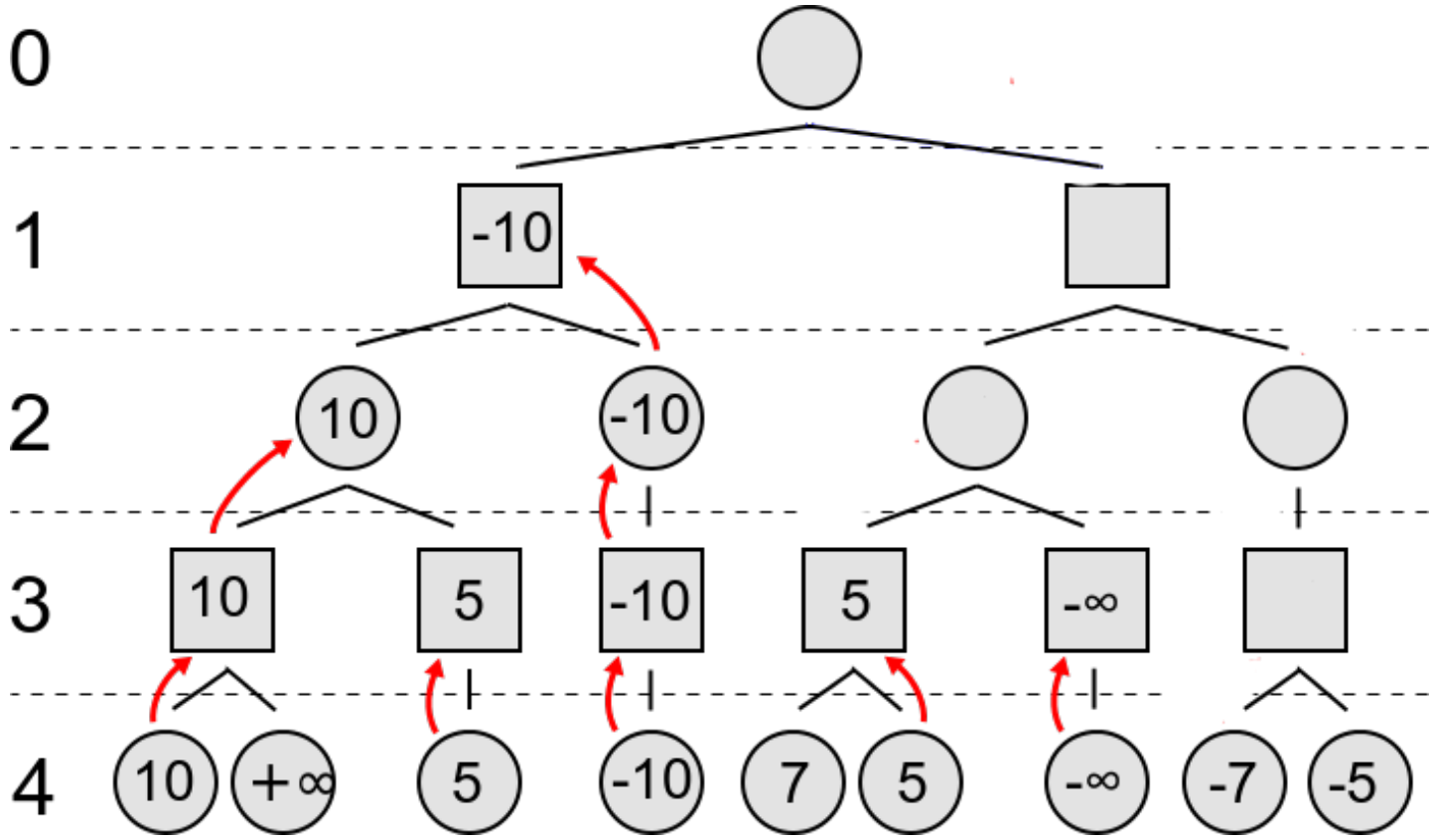

0
1
2
3
4
10
10
5
10
+∞
5
-10
7
5
-∞
-7
-5

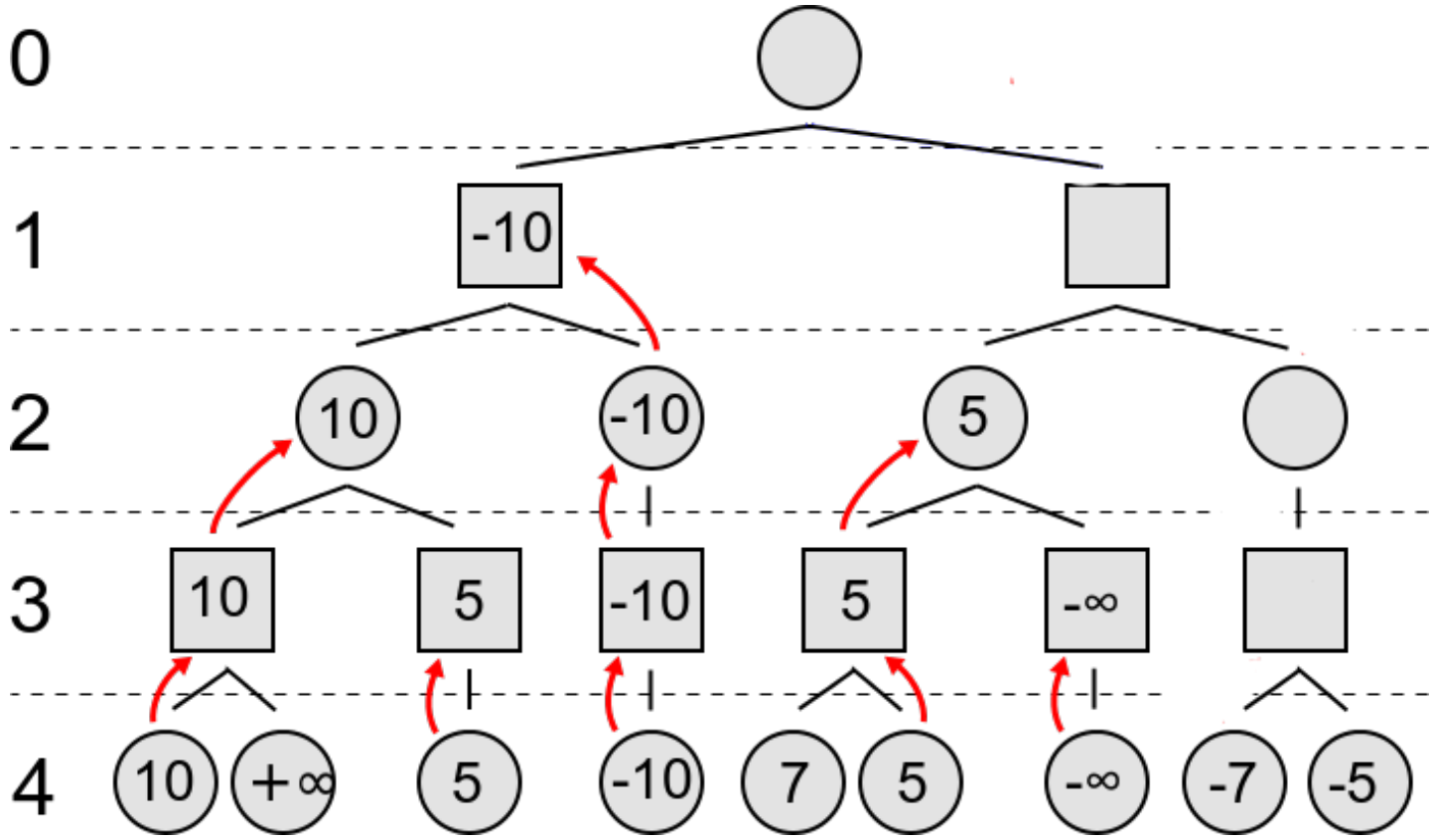

0
1
2
3
4
10
10
5
-10
10
+∞
5
-10
7
5
-∞
-7
-5

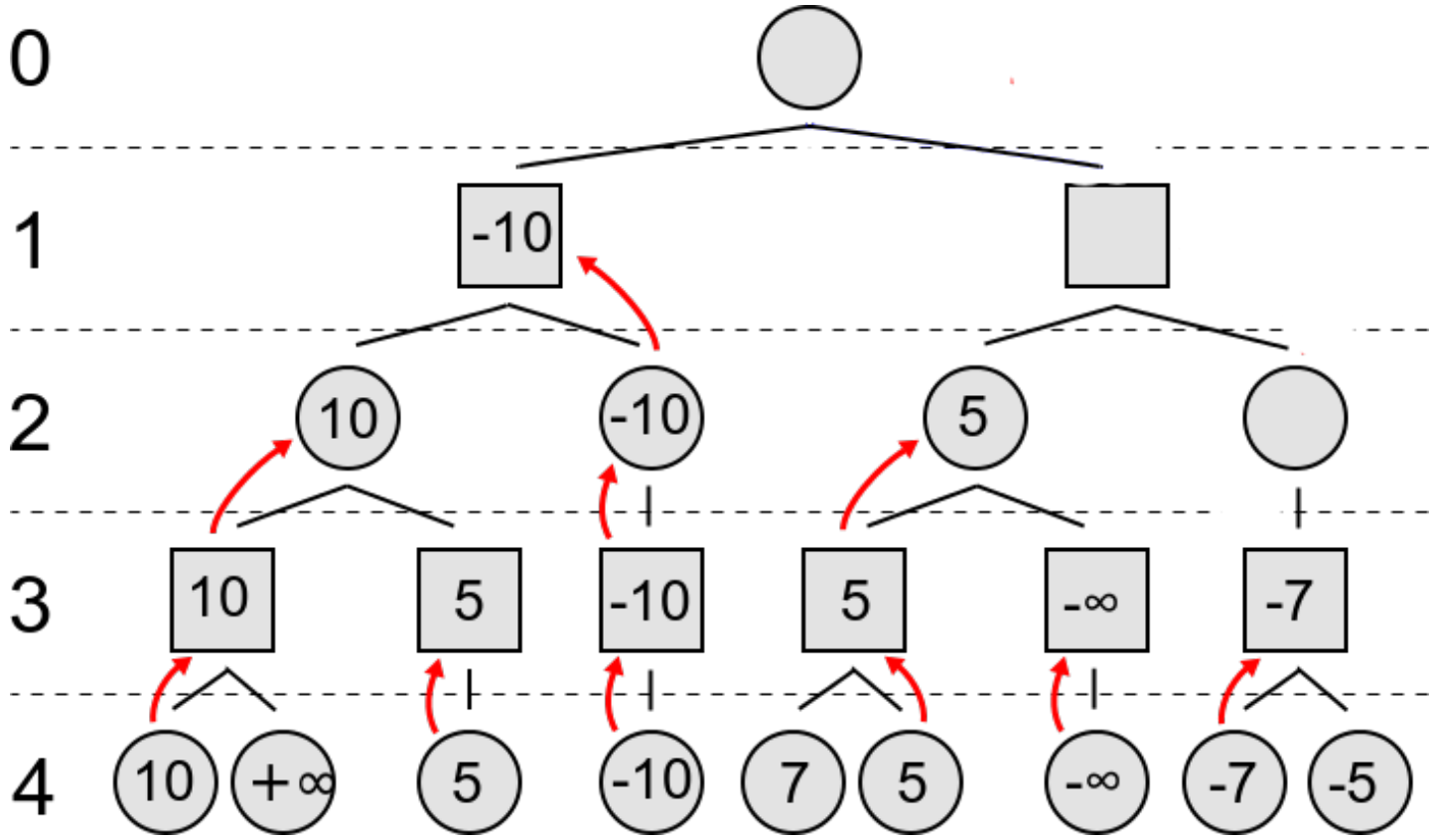

0
1
2
3
4
10
-10
10
5
-10
10
+∞
5
-10
7
5
-∞
-7
-5

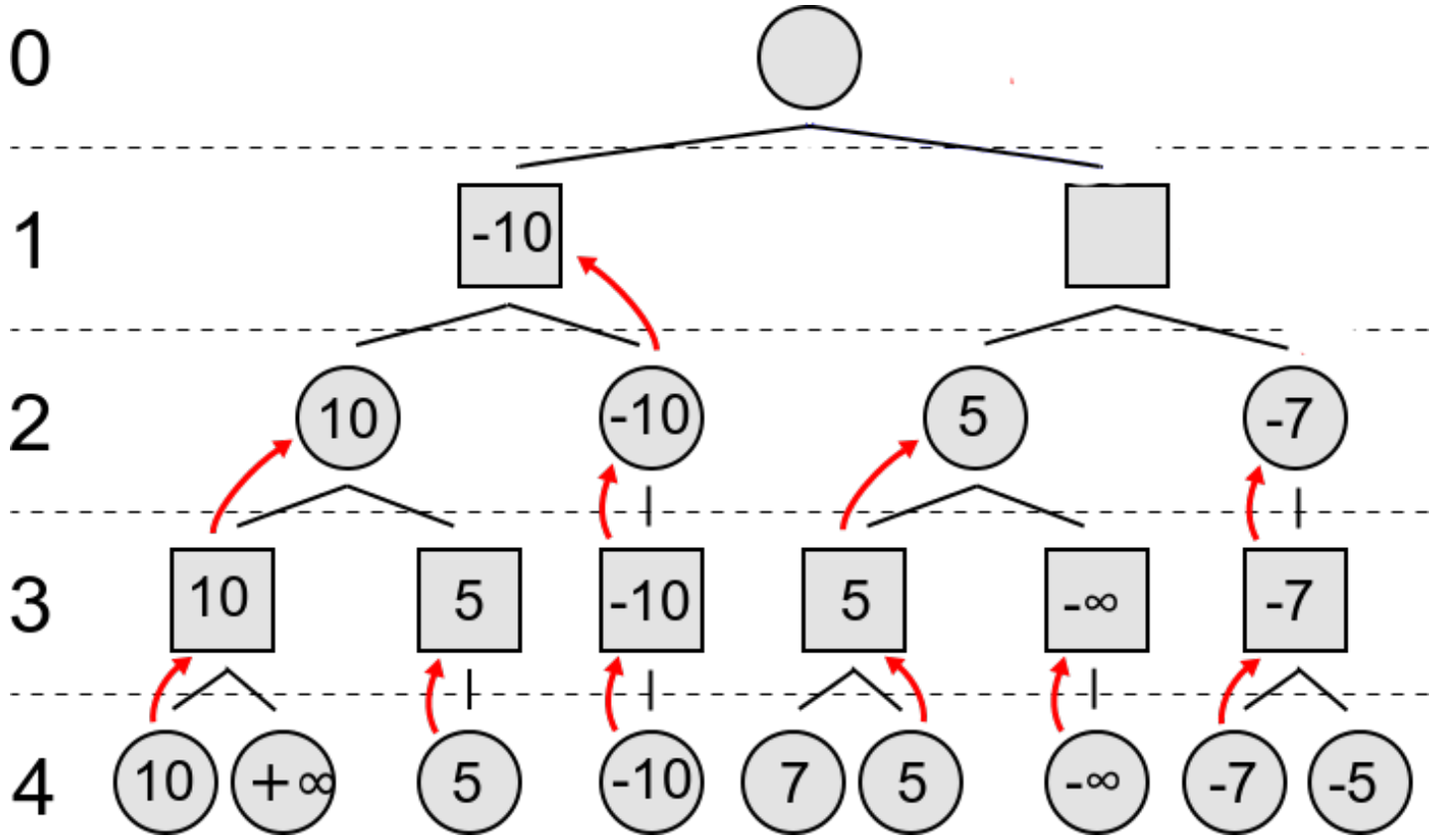

0
1
2
3
4
-10
10
-10
10
5
-10
10
+∞
5
-10
7
5
-∞
-7
-5

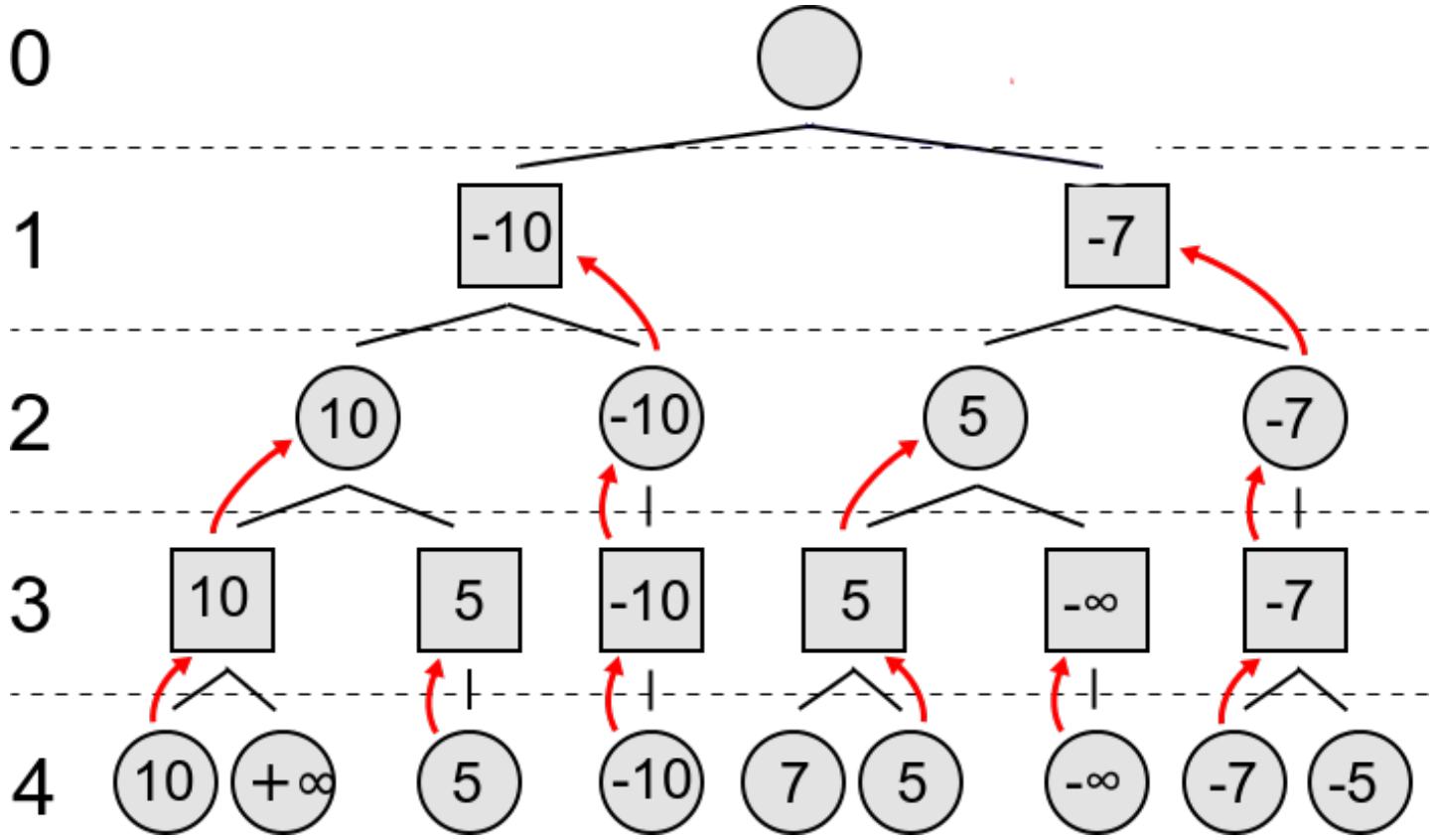

0
1
2
3
4
-10
10
-10
10
5
-10
5
10
+∞
5
-10
7
5
-∞
-7
-5

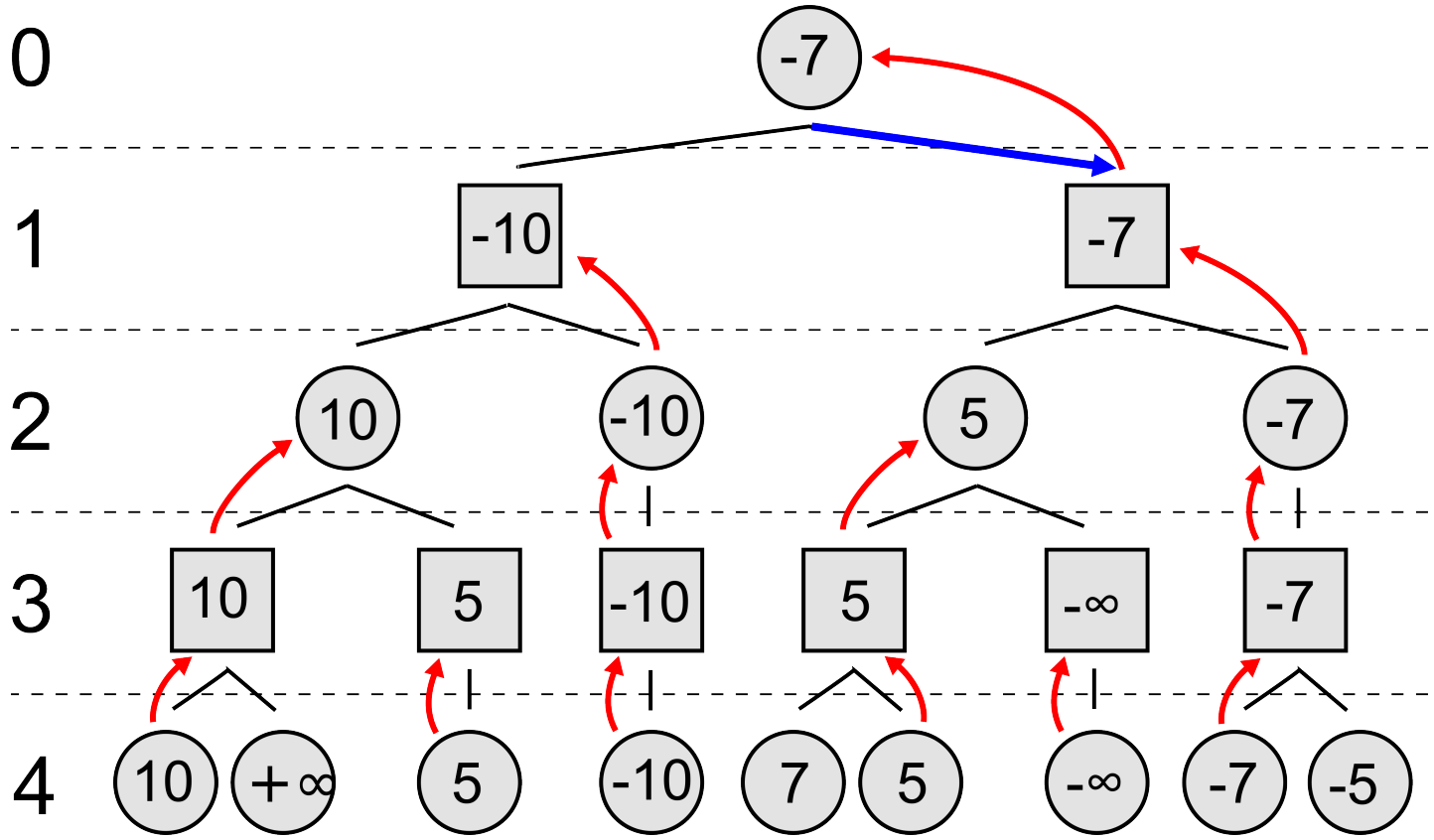

0
1
2
3
4
-10
10
-10
10
5
-10
5
-∞
10
+∞
5
-10
7
5
-∞
-7
-5


0
1
2
3
4
-10
10
-10
5
10
5
-10
5
-∞
10
+∞
5
-10
7
5
-∞
-7
-5


0
1
2
3
4
-10
10
-10
5
10
5
-10
5
-∞
-7
10
+∞
5
-10
7
5
-∞
-7
-5


0
1
2
3
4
-10
10
-10
5
-7
10
5
-10
5
-∞
-7
10
+∞
5
-10
7
5
-∞
-7
-5


0
1
2
3
4
-10
-7
10
-10
5
-7
10
5
-10
5
-∞
-7
10
+∞
5
-10
7
5
-∞
-7
-5


0
1
2
3
4
-7
-10
-7
10
-10
5
-7
10
5
-10
5
-∞
-7
10
+∞
5
-10
7
5
-∞
-7
-5

Vorteile von Minimax basiertem Schach

- Einfach umszusetzen
- Programmierer braucht wenig Schachkentnisse
- Je schneller die CPU desto besser das Programm
- Leicht erweiterbar
- Spielen gut Schach

Fazit

- Schachprogrammierung aus Sicht der Forschung voller Erfolg?

  Keine bahnbrechende Erkenntnisse
- Computer spielen nicht wie menschen
- Alternativen: Go, Jeopordy, Roboter

Referenzen

Ensmenger 2011: Is Chess the Drosophila of AI? The Social History of an Algorithm

https://de.wikipedia.org/wiki/Schachprogramm