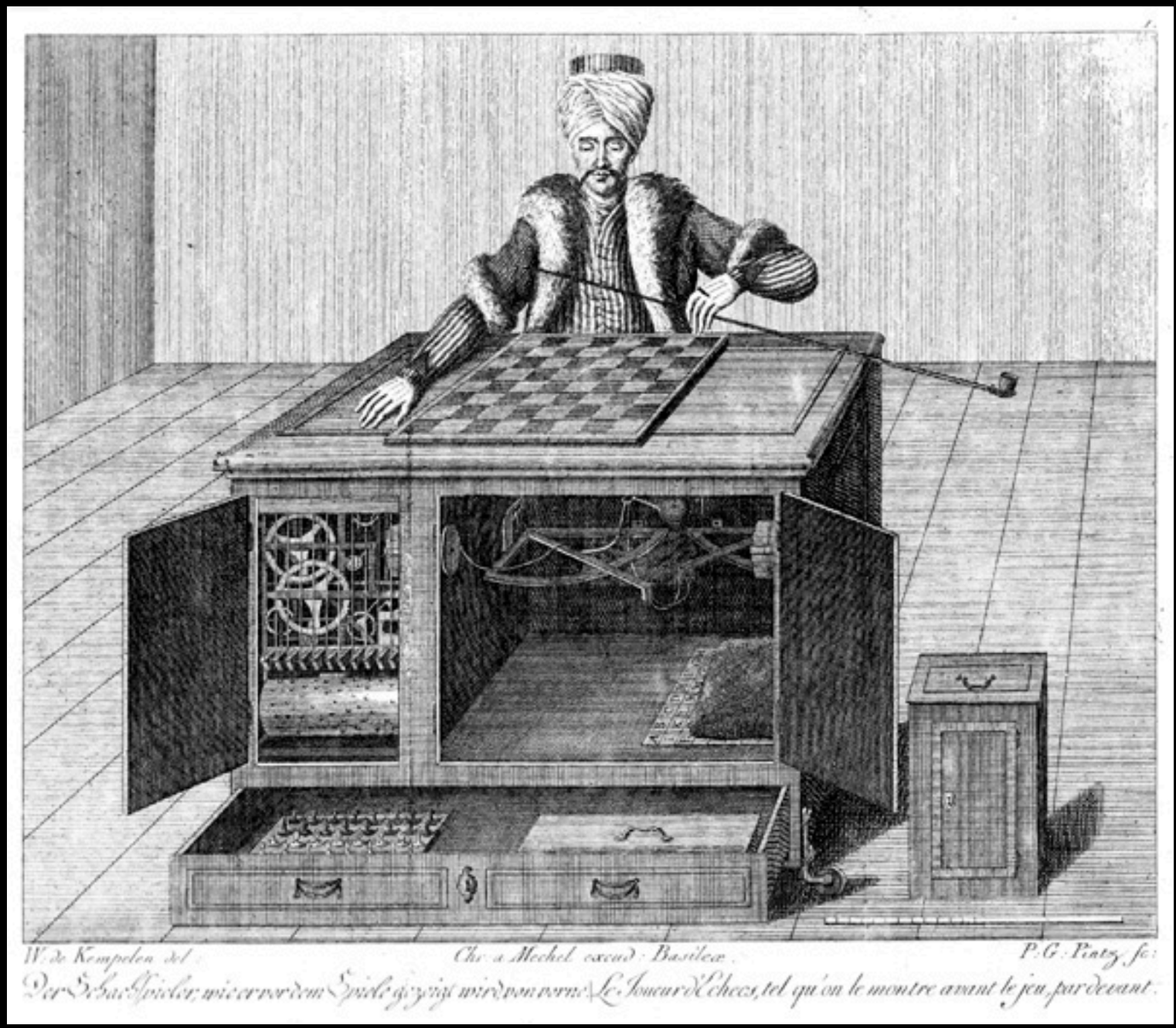
Computers Playing Games

Jordan Hoffmann

1769

- Wolfgang von Kempelen
  - Notable opponents:
    - Napoleon
    - Benjamin Franklin



1868

- Charles Hooper *Ajeeb*
  - Notable opponents:
    - Harry Houdini
    - Theodore Roosevelt

1878

- Mephisto
  - Controlled remotely

1948

- Turochamp — Alan Turing and David Champernowne
- Written before computers could run it.
- Played Kasparov in 2012— 2 ply.

1949

- Claude Shannon “Programming a computer for playing chess”

These and similar principles are only generalizations from empirical evidence of numerous games, and only have a kind of statistical validity. Probably any chess principle can be contradicted by particular counter examples. However, form these principles one can construct a crude evaluation function. The following is an example: -

$$f(P) = 200(K-K') + 9(Q-Q') + 5(R-R') + 3(B-B'+N-N') + (P-P') - 0.5(D-D'+S-S'+I-I') + 0.1(M-M') + \ldots$$

in which: -

(1)K,Q,R,B,B,P are the number of White kings, queens, rooks, bishops, knights and pawns on the board.

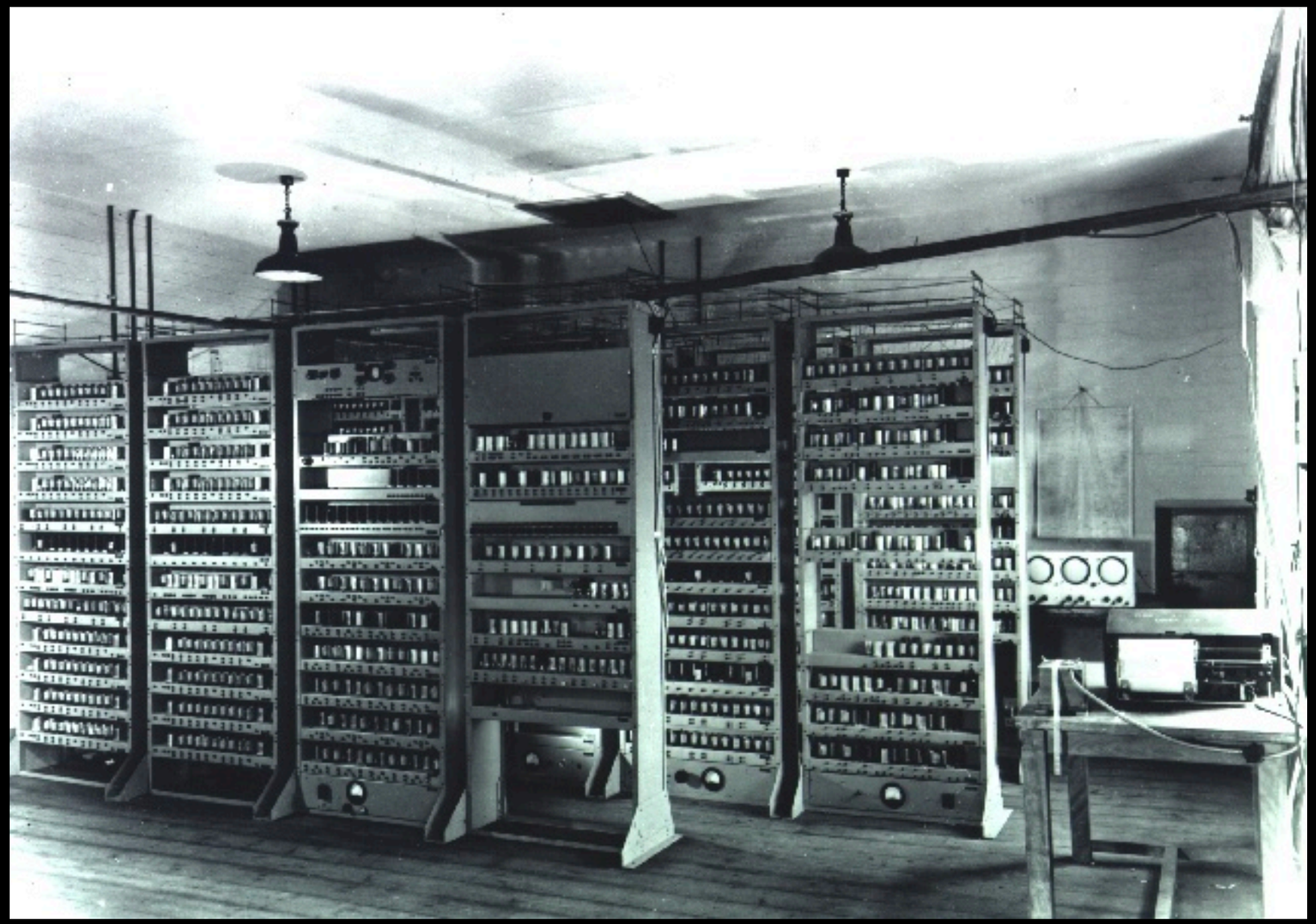
(2)D,S,I are doubled, backward and isolated White pawns.

(3)M= White mobility (measured, say, as the number of legal moves available to White).

Primed letters are the similar quantities for Black.

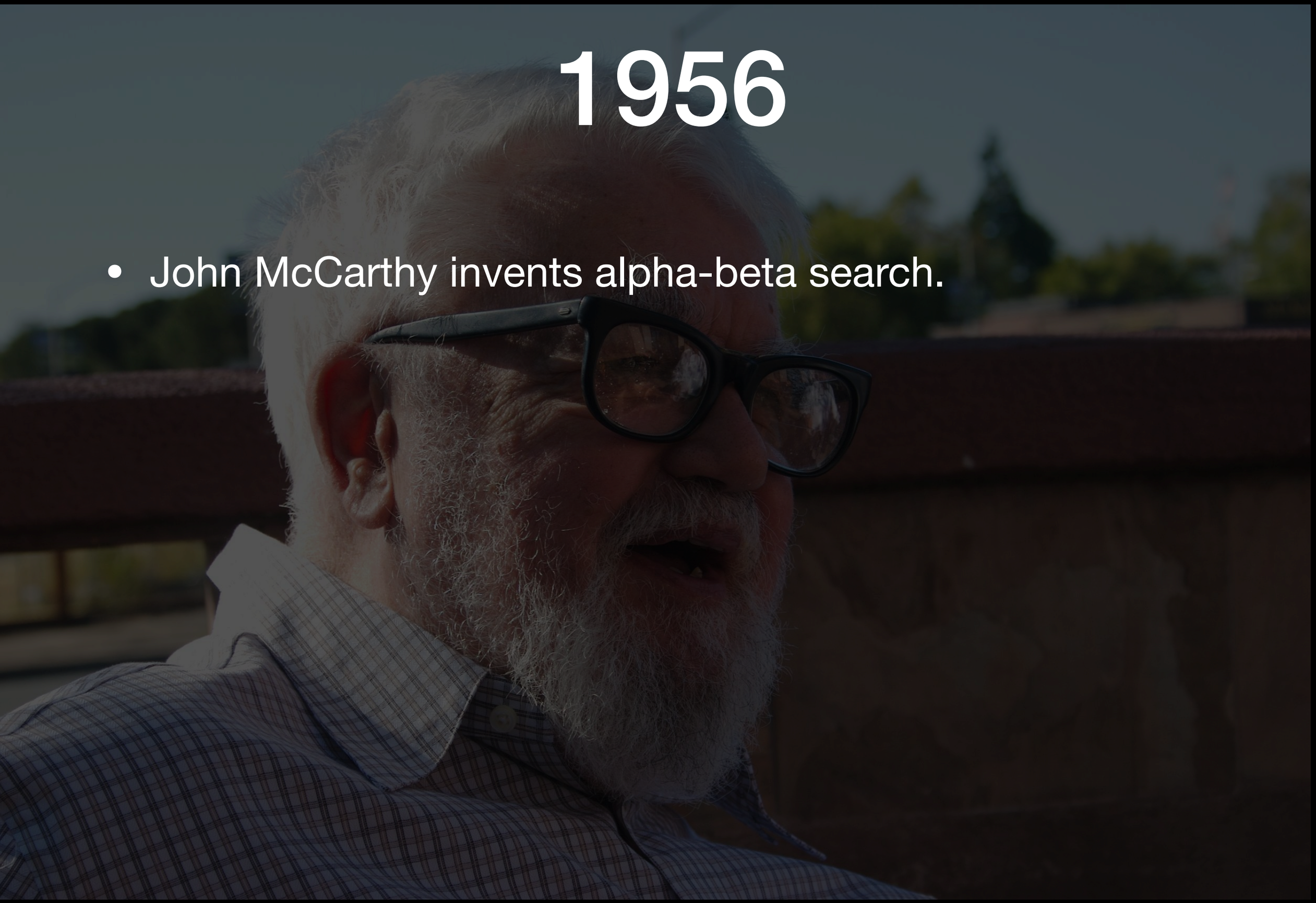
1952

- OXO PhD thesis at Cambridge, A S Douglas. First computer game.



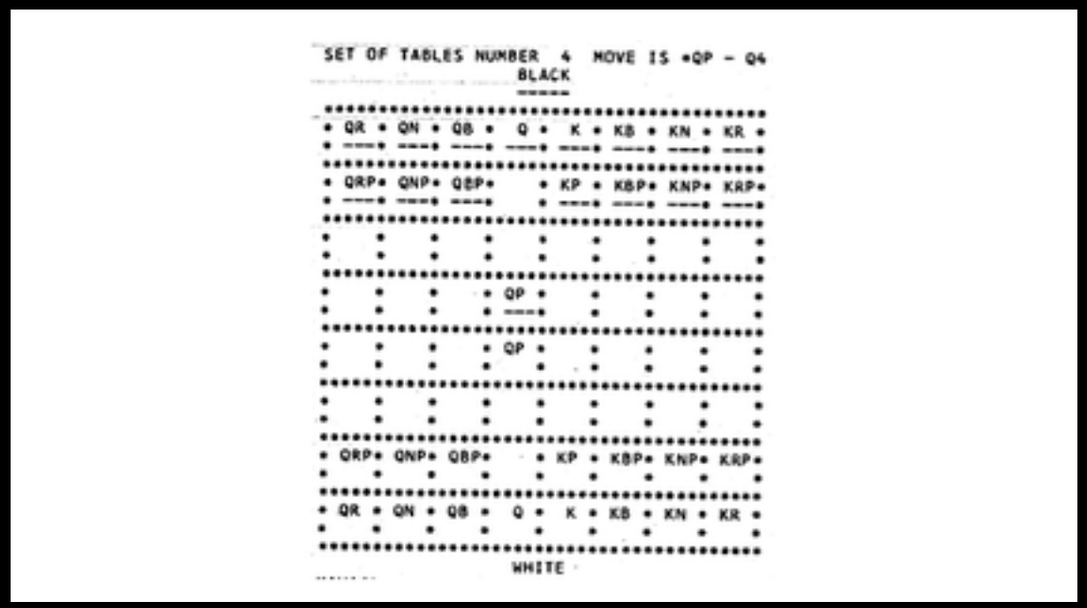
1956

- John McCarthy invents alpha-beta search.



1962

- Kotok-McCarthy is first program to play a recognizable game of chess. *Kotok did this for his MIT undergraduate thesis.*



SET OF TABLES NUMBER 4 MOVE IS •QP - Q4

BLACK

QR	QN	QB	Q	K	KB	KN	KR
QRP	QNP	QBP		KP	KBP	KNP	KRP
			QP				
			QP				
QRP	QNP	QBP		KP	KBP	KNP	KRP
QR	QN	QB	Q	K	KB	KN	KR

WHITE

1986

- Maven— Scrabble, 1998 beat world champion

```
Find all possible plays
Rank all
Randomly draw tiles for opponent and play two turns
Rack evaluation
```

1986

- Maven— Scrabble, 1998 beat world champion

Find all possible plays
Rank all
Randomly draw tiles for opponent and play two turns
Rack evaluation

When within one draw of 7 tiles, try to yield good “endgame”

1986

- Maven— Scrabble, 1998 beat world champion

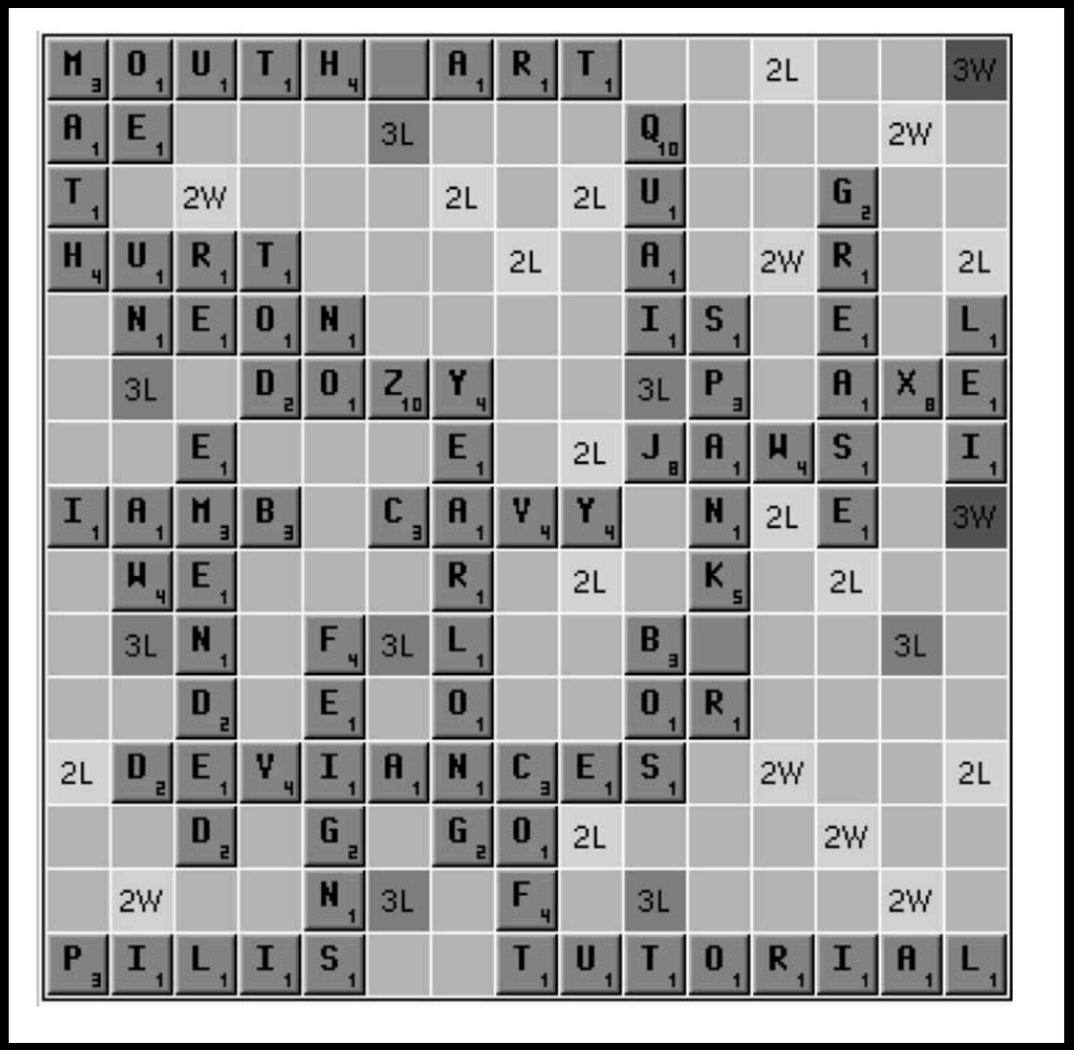
Find all possible plays
Rank all
Randomly draw tiles for opponent and play two turns
Rack evaluation

When within one draw of 7 tiles, try to yield good “endgame”

In the endgame, can use optimal play.
Note: does not use Alpha-Beta (will discuss in a moment)

1986

- Maven— Scrabble, 1998 beat world champion



1988

- Connect 4 solved, improved in 2015

1992

- TD-Gammon

Gradient of NN w.r.t weights

$$w_{t+1} - w_t = \alpha(Y_{t+1} - Y_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w Y_k$$

Change weight from last turn

Change in board evaluation

How important are older positions?

1992


1992

TD Gammon

$$L = \alpha||V_t(s_{t+1}) - V_t(s_t)||$$

We want predictions about present to match future.

TD Gammon

$$L = \alpha||V_t(s_{t+1}) - V_t(s_t)||$$

We want predictions about present to match future.

$$L = \alpha||z - V_t(s_t)||$$

Z is the true outcome.

TD Gammon

$$L = \alpha||V_t(s_{t+1}) - V_t(s_t)||$$

Leads to parameter updates: \delta_t is difference between two state predictions.

$$\theta_{t+1} = \theta_t + \alpha\delta_d \frac{\partial V_t}{\partial \theta_t}$$

But. Need to assign credit.

TD Gammon

$$\theta_{t+1} = \theta_t + \alpha\delta_d \frac{\partial V_t}{\partial \theta_t}$$

But. Need to assign credit.
Store ALL gradients. Weight:

$$\theta_{t+1} = \theta_t + \alpha\delta_t \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial V_k}{\partial \theta_t}$$

1996-7

- Deep Blue loses to, then beats Kasparov

2007

- Draughts solved, search space: 5 x 10^20

RL in Games: State of the Art

Program	Level of Play	RL Program to Achieve Level
Checkers	Perfect	*Chinook*
Chess	International Master	*KnightCap* / *Meep*
Othello	Superhuman	*Logistello*
Backgammon	Superhuman	*TD-Gammon*
Scrabble	Superhuman	*Maven*
Go	Grandmaster	*MoGo*[1], *Crazy Stone*[2], *Zen*[3]
Poker[4]	Superhuman	*SmooCT*

[1] 9×9
[2] 9×9 and 19×19
[3] 19×19
[4] Heads-up Limit Texas Hold'em

2015

- AlphaGo

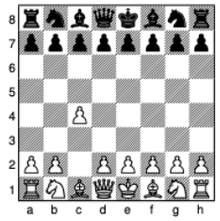
2017

- AlphaGo Zero beats AlphaGo 100-0

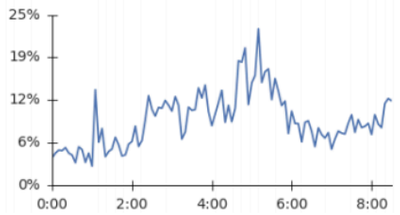
2017

- AlphaZero beats a version of Stockfish 28-0-78

A10: English Opening

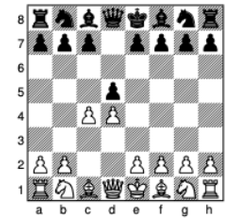


w 20/30/0, b 8/40/2

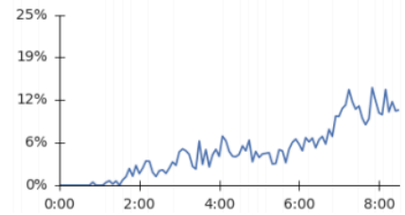


1...e5 g3 d5 cxd5 ♘f6 ♗g2 ♘xd5 ♘f3

D06: Queens Gambit

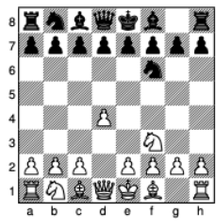


w 16/34/0, b 1/47/2

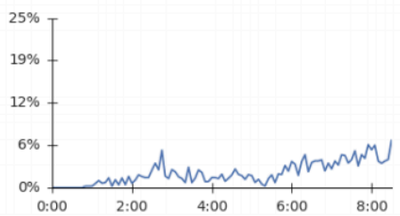


2...c6 ♘c3 ♘f6 ♘f3 a6 g3 c4 a4

A46: Queens Pawn Game

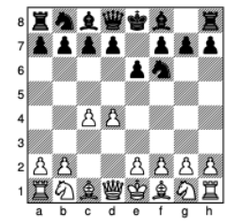


w 24/26/0, b 3/47/0

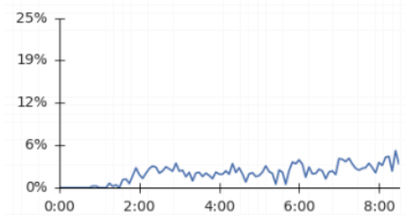


2...d5 c4 e6 ♘c3 ♗e7 ♗f4 O-O e3

E00: Queens Pawn Game

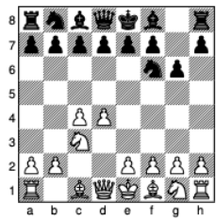


w 17/33/0, b 5/44/1

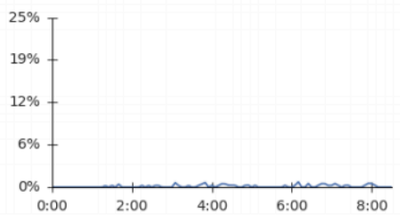


3.♘f3 d5 ♘c3 ♗b4 ♗g5 h6 ♕a4 ♘c6

E61: Kings Indian Defence

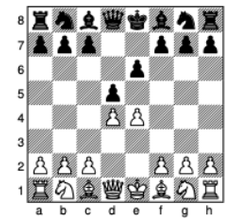


w 16/34/0, b 0/48/2

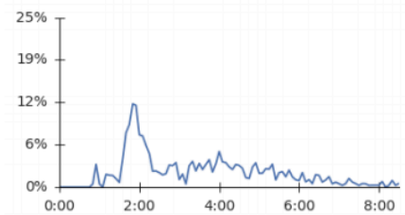


3...d5 cxd5 ♘xd5 e4 ♘xc3 bxc3 ♗g7 ♗e3

C00: French Defence

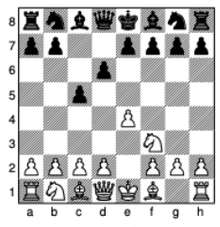


w 39/11/0, b 4/46/0

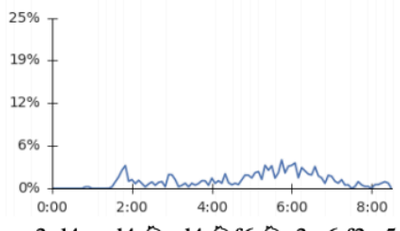


3.♘c3 ♘f6 e5 ♘d7 f4 c5 ♘f3 ♗e7

B50: Sicilian Defence

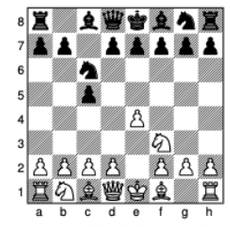


w 17/32/1, b 4/43/3

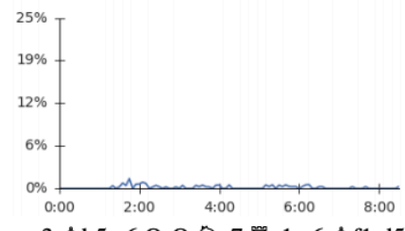


3.d4 cxd4 ♘xd4 ♘f6 ♘c3 a6 f3 e5

B30: Sicilian Defence

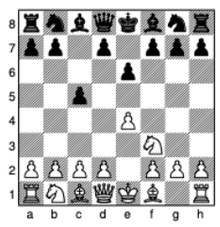


w 11/39/0, b 3/46/1

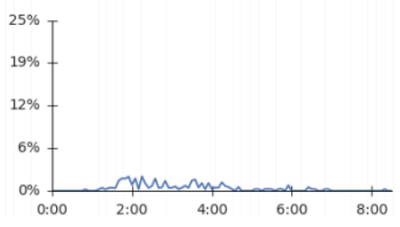


3.♗b5 e6 O-O ♘e7 ♖e1 a6 ♗f1 d5

B40: Sicilian Defence

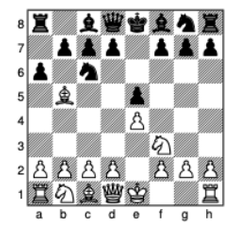


w 17/31/2, b 3/40/7

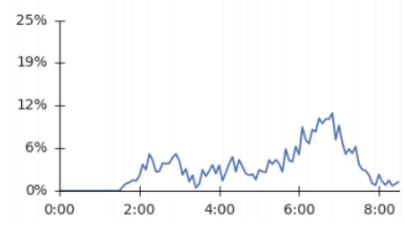


3.d4 cxd4 ♘xd4 ♘c6 ♘c3 ♕c7 ♗e3 a6

C60: Ruy Lopez (Spanish Opening)

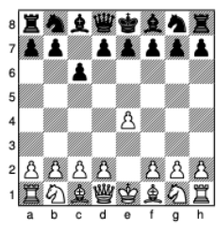


w 27/22/1, b 6/44/0

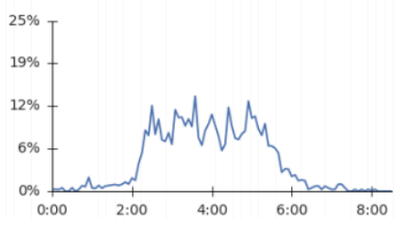


4.♗a4 ♗e7 O-O ♘f6 ♖e1 b5 ♗b3 O-O

B10: Caro-Kann Defence

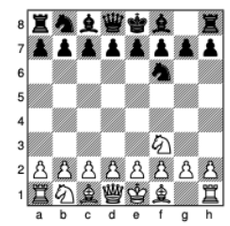


w 25/25/0, b 4/45/1

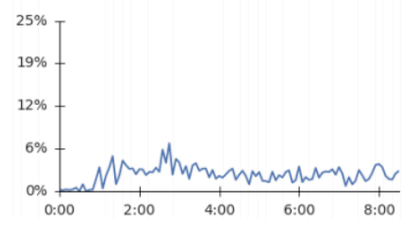


2.d4 d5 e5 ♗f5 ♘f3 e6 ♗e2 a6

A05: Reti Opening



w 13/36/1, b 7/43/0

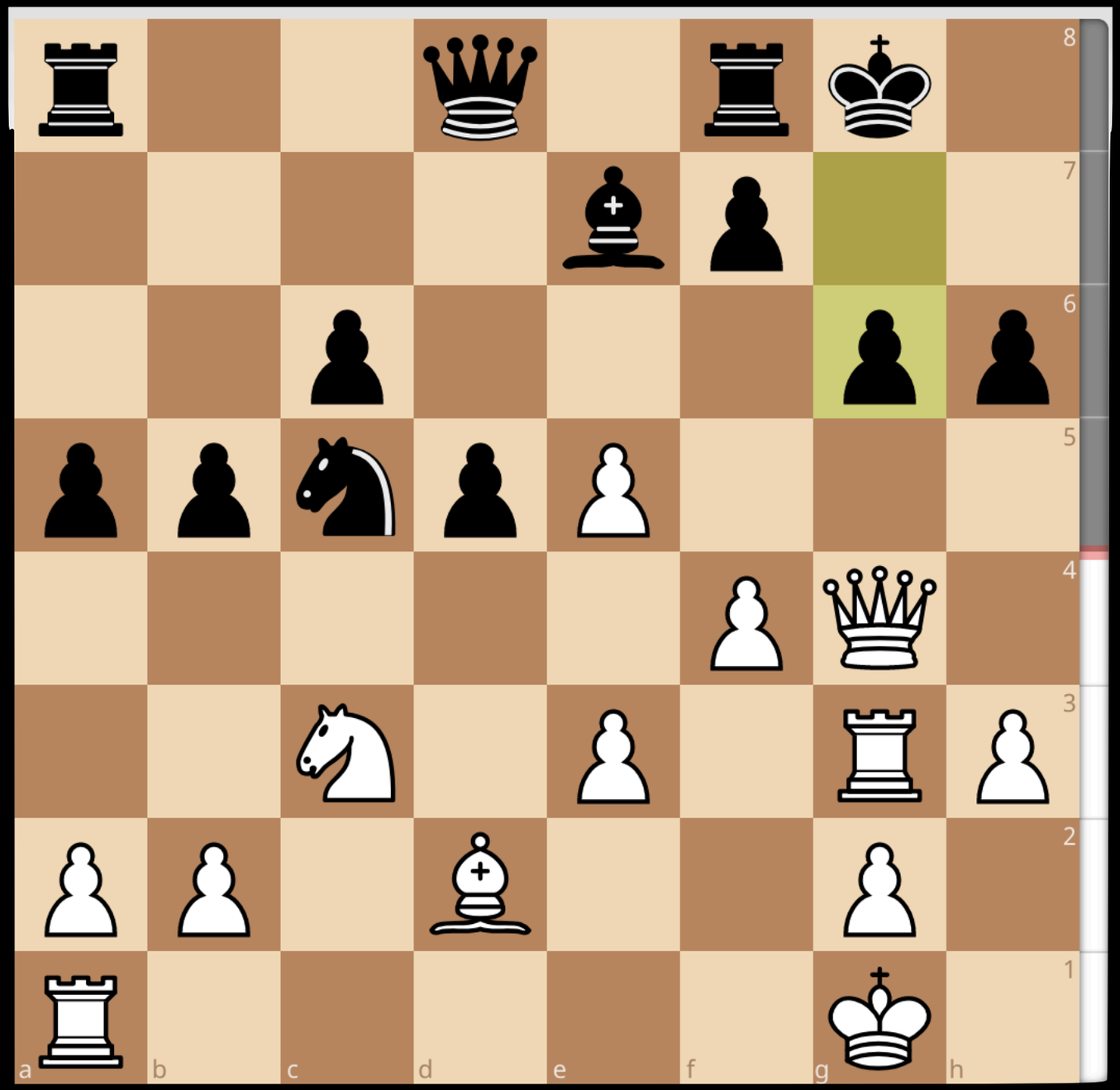


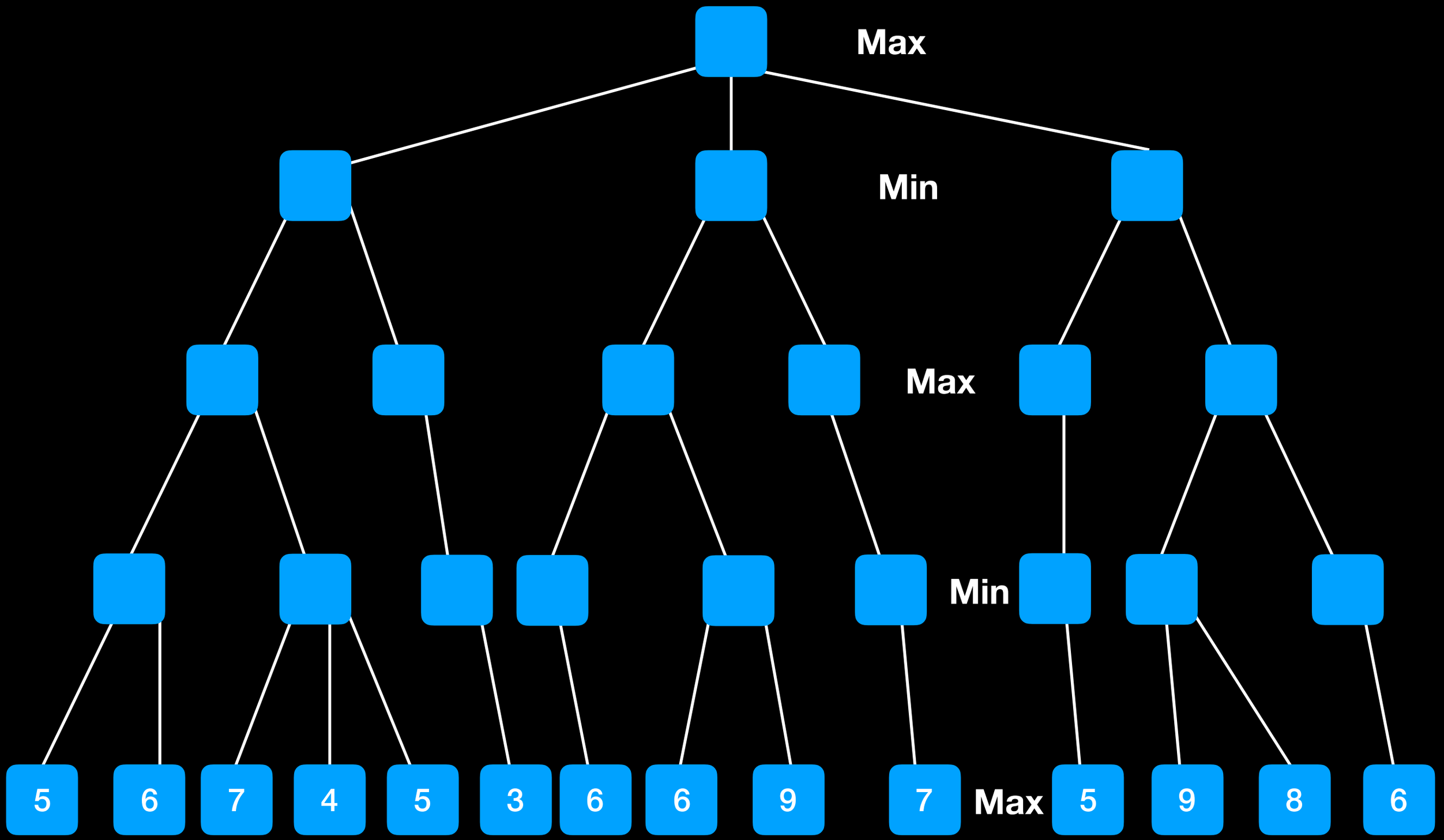
2.c4 e6 d4 d5 ♘c3 ♗e7 ♗f4 O-O

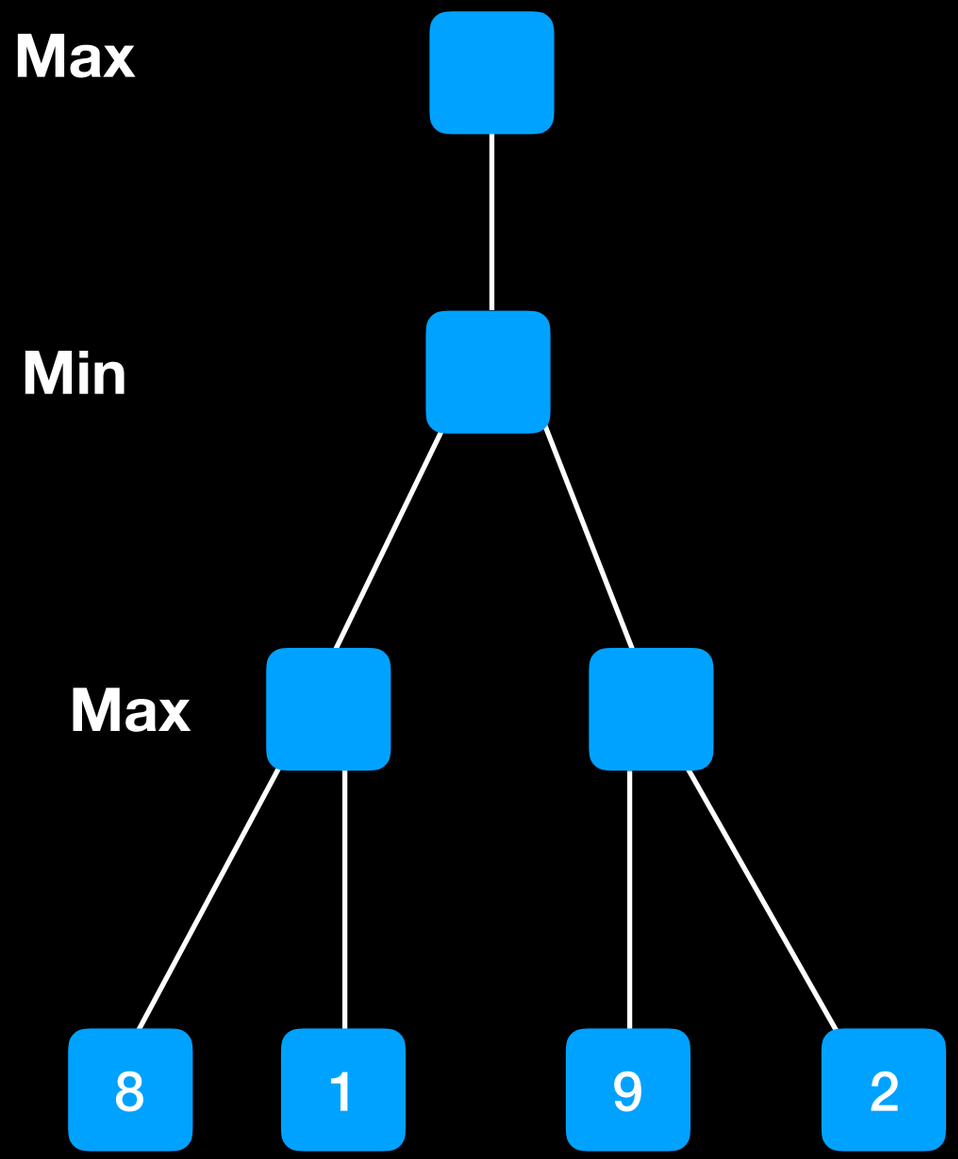
Total games: w 242/353/5, b 48/533/19

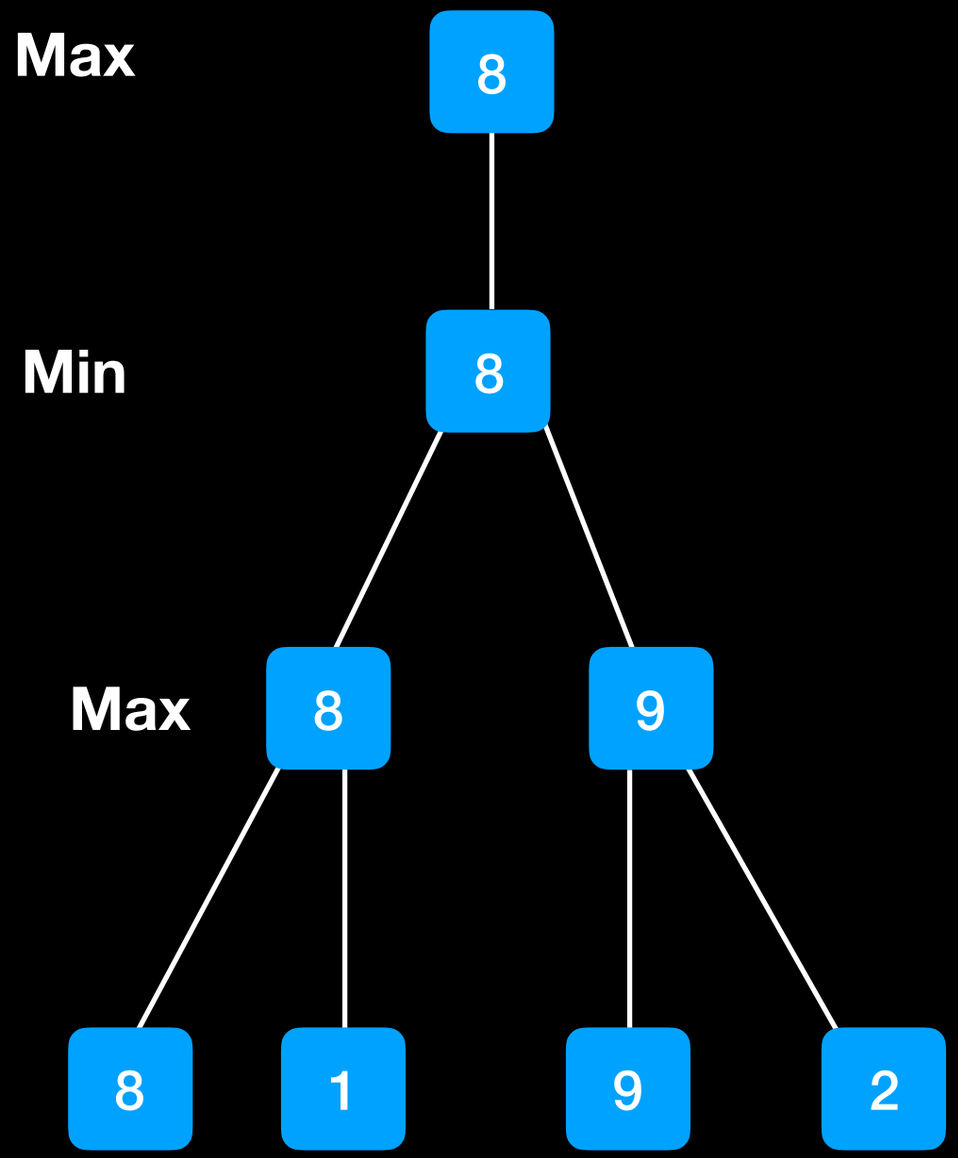
Overall percentage: w 40.3/58.8/0.8, b 8.0/88.8/3.2

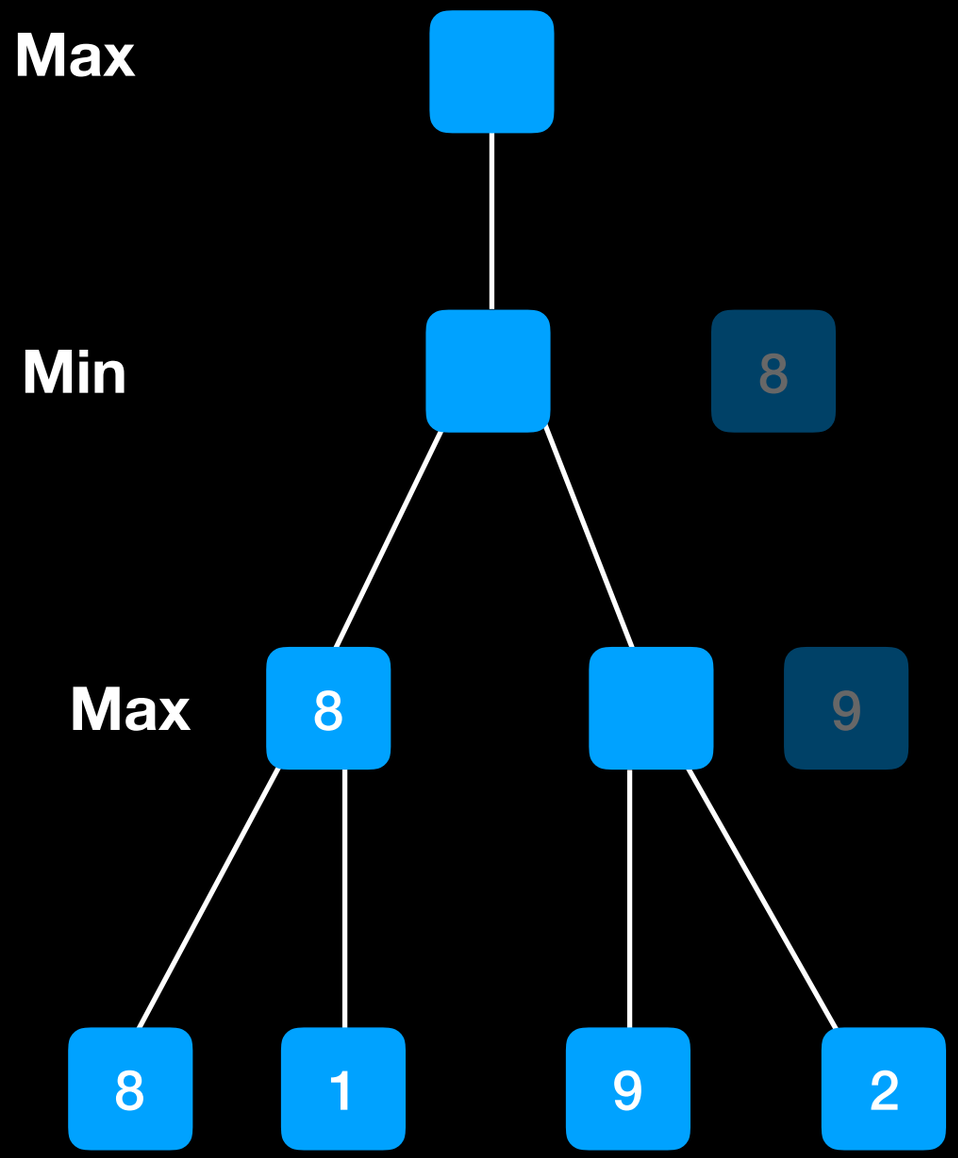
2018

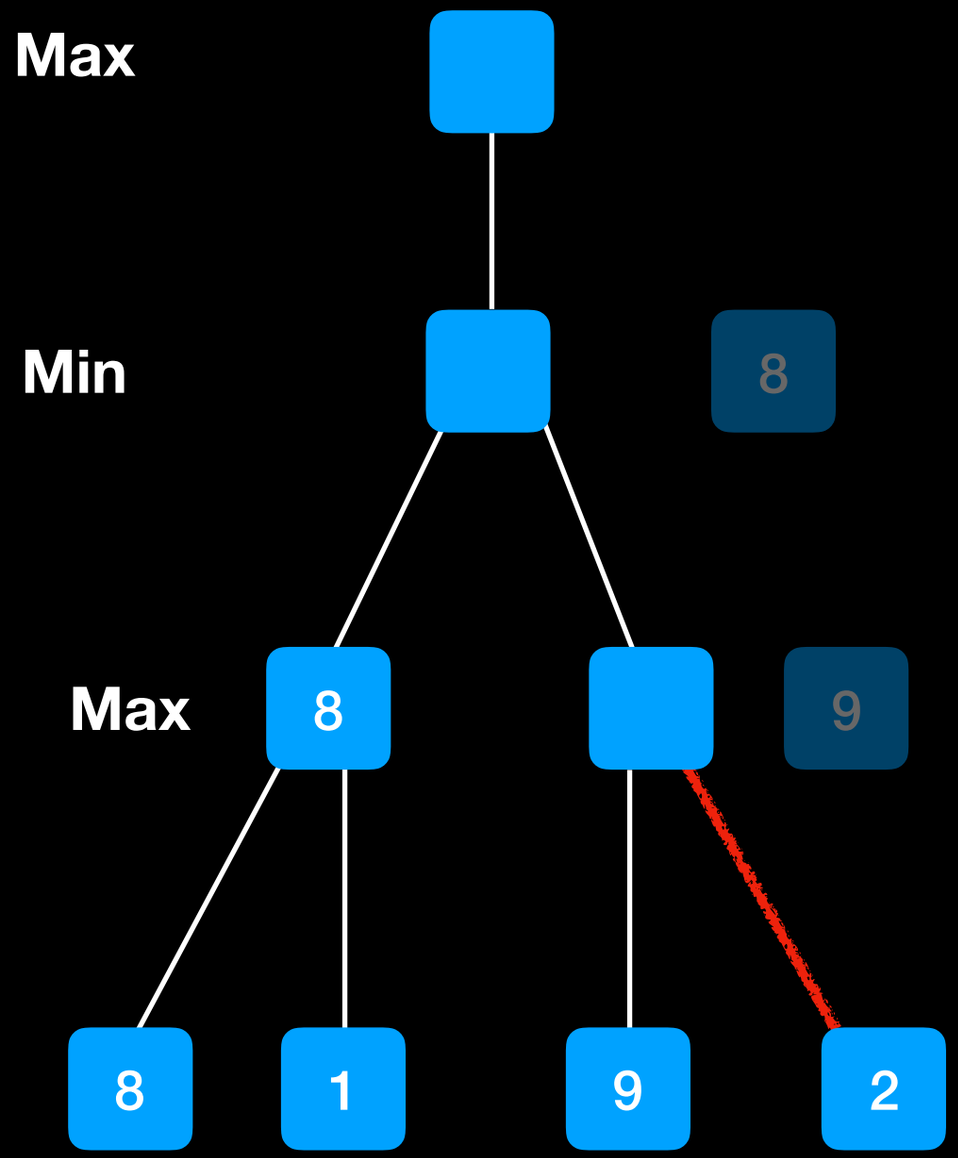

8
7
6
5
4
3
2
1
a
b
c
d
e
f
g
h


Max
Min
Max
Min
Max
5
6
7
4
5
3
6
6
9
7
5
9
8
6

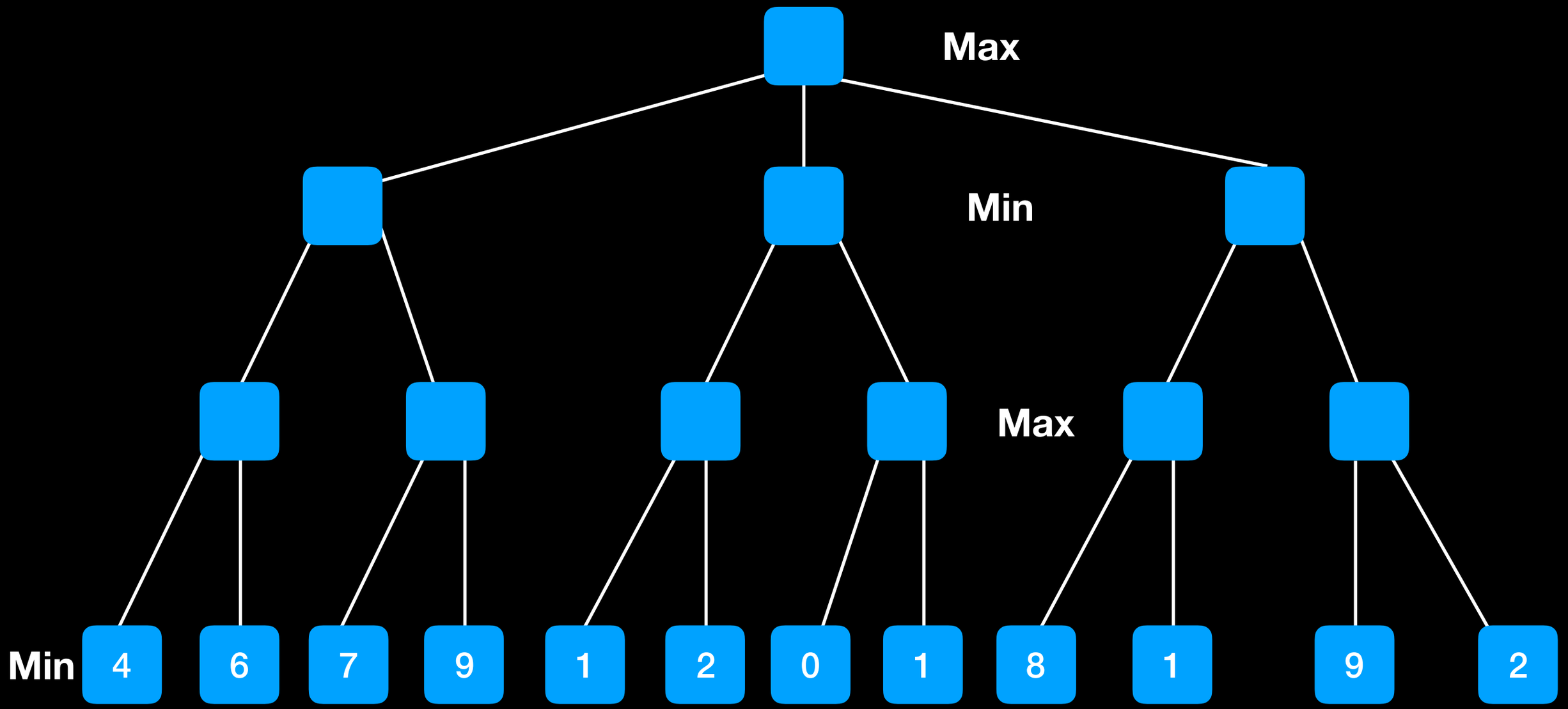

Max
Min
Max
8
1
9
2

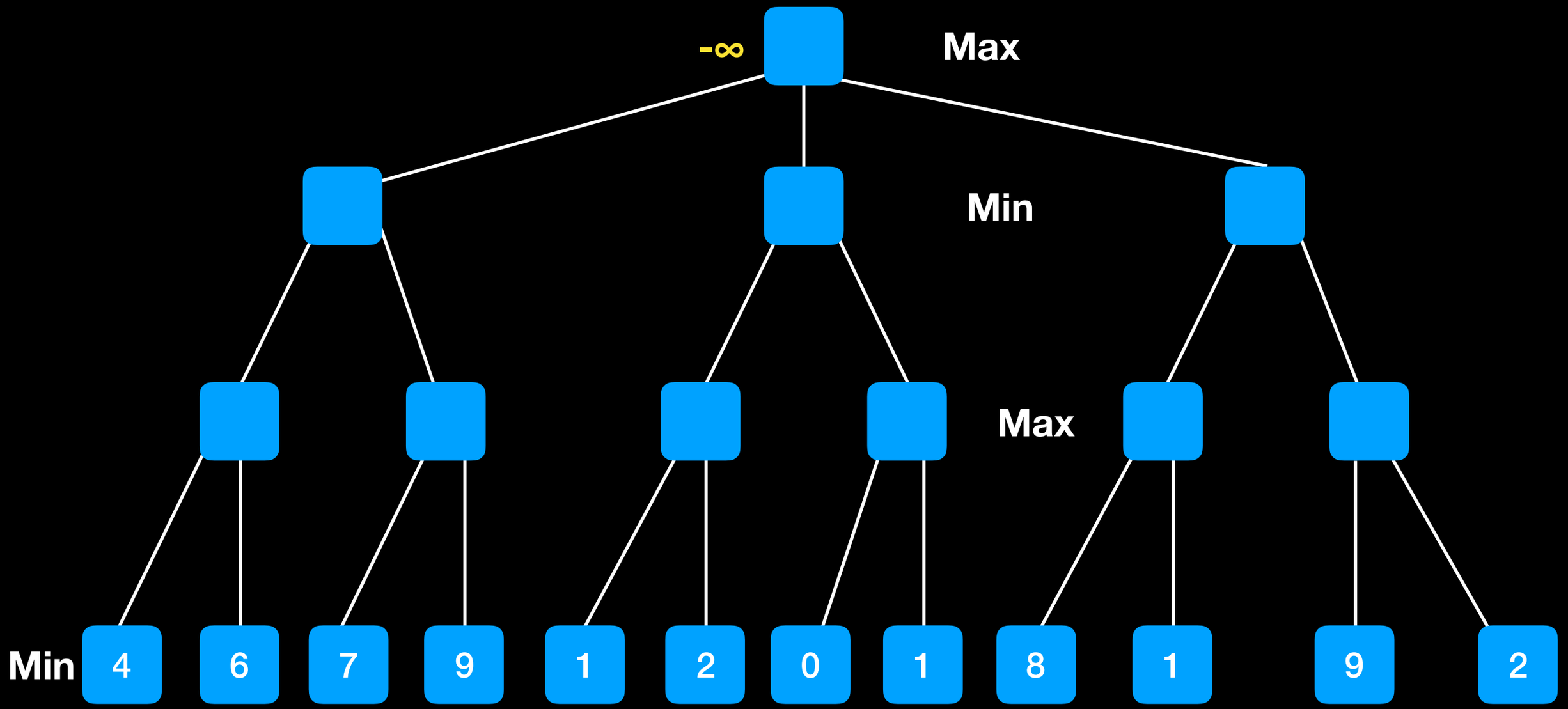

Max
8
Min
8
Max
8
9
8
1
9
2

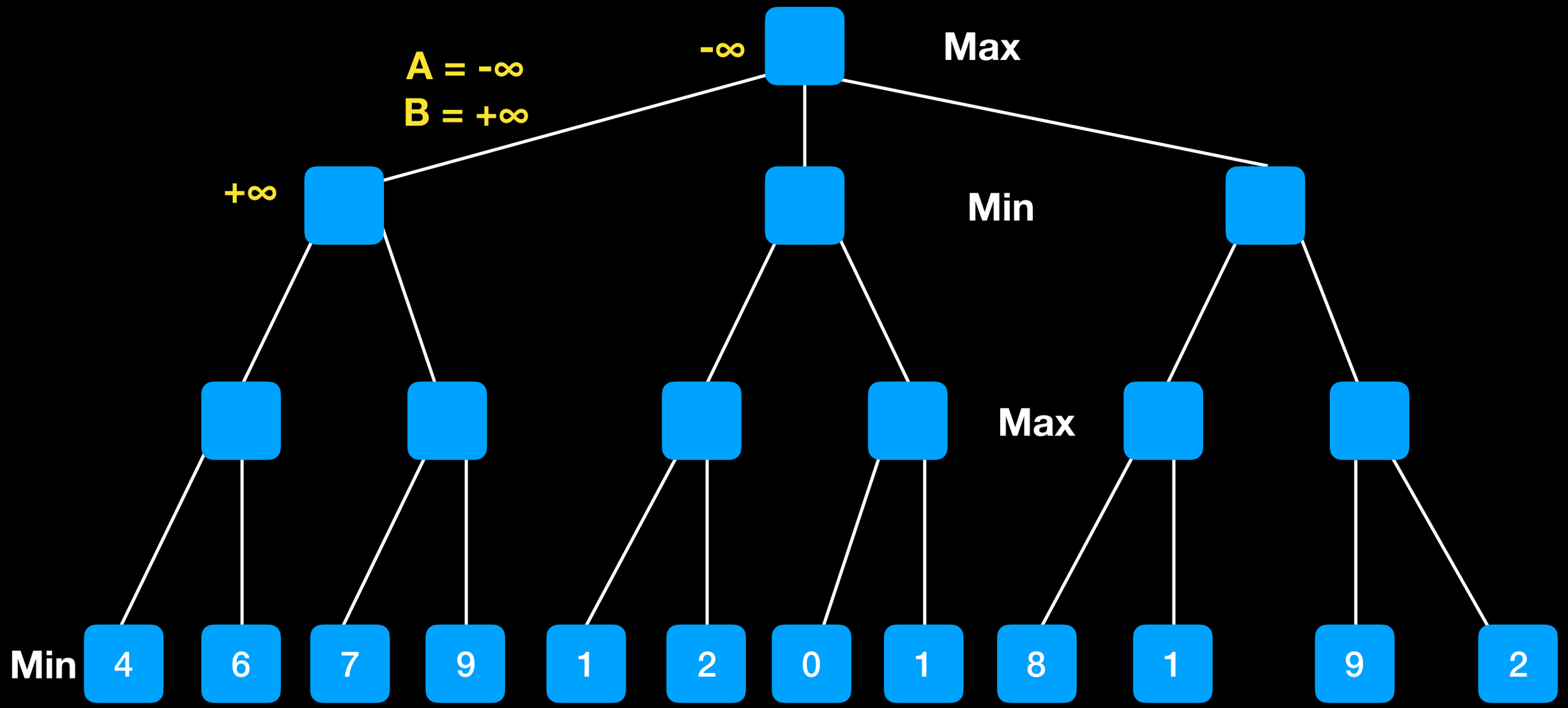

Max
Min
8
Max
8
9
8
1
9
2

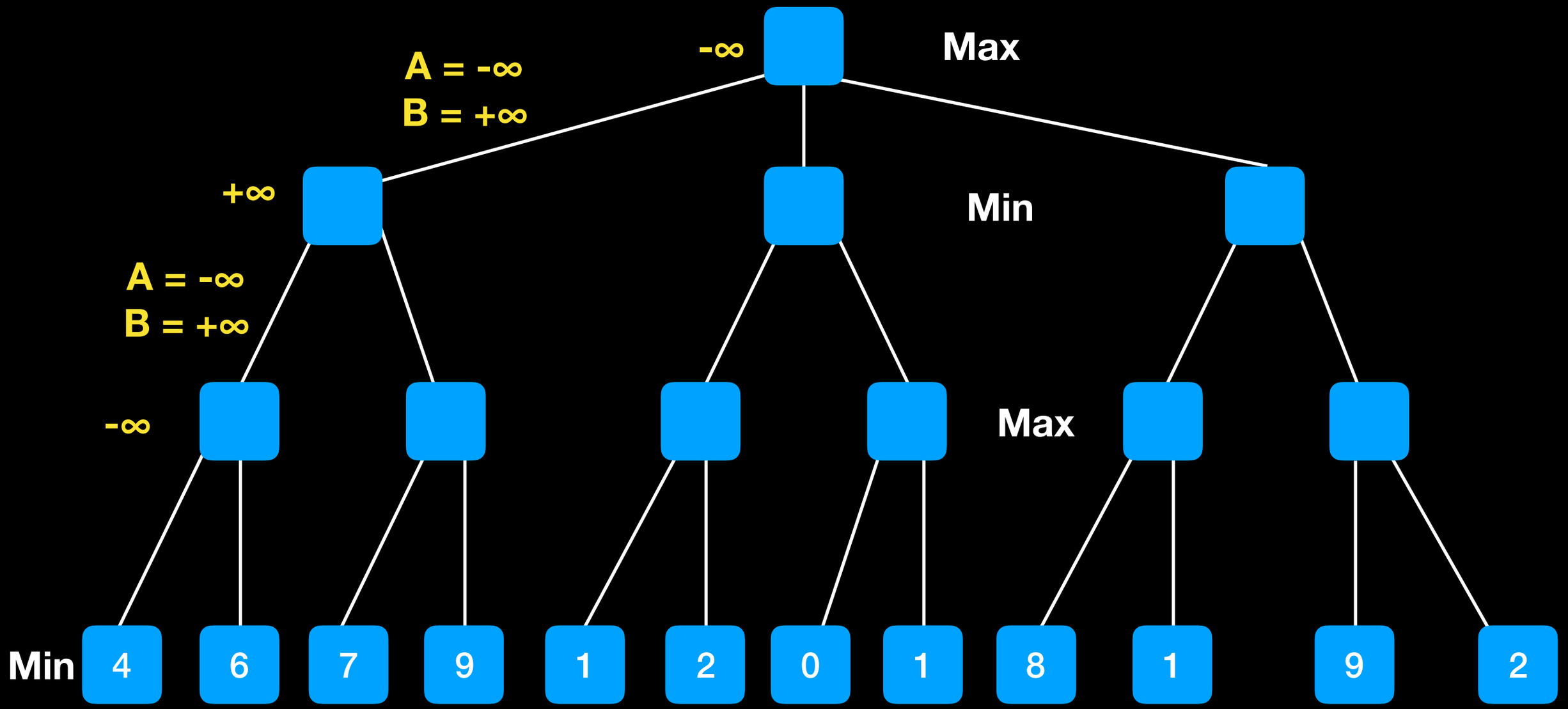

Max
Min
8
Max
8
9
8
1
9
2

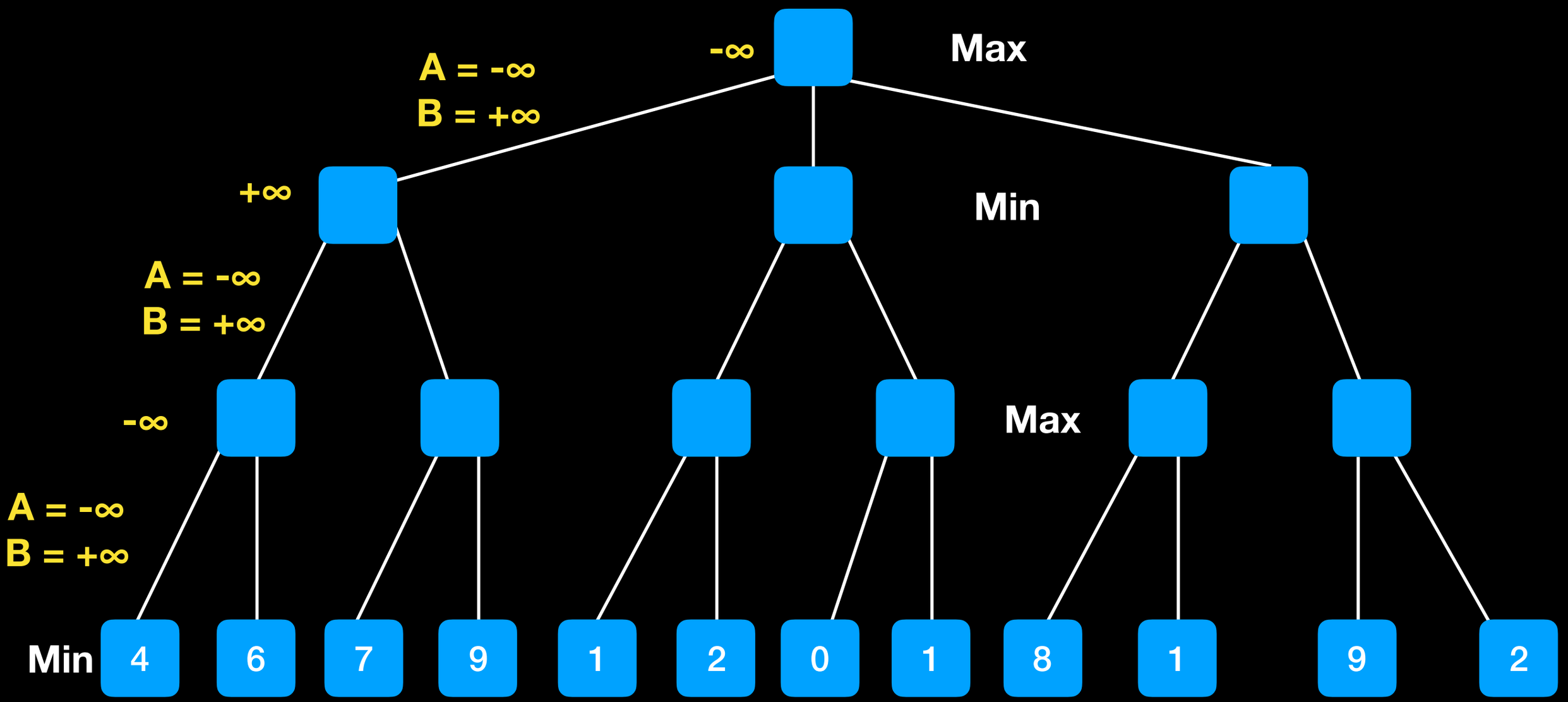
```
function alphabeta(node, depth, α, β, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, alphabeta(child, depth − 1, α, β, FALSE))
            α := max(α, value)
            if α ≥ β then
                break (* β cut-off *)
        return value
    else
        value := +∞
        for each child of node do
            value := min(value, alphabeta(child, depth − 1, α, β, TRUE))
            β := min(β, value)
            if α ≥ β then
                break (* α cut-off *)
        return value
```

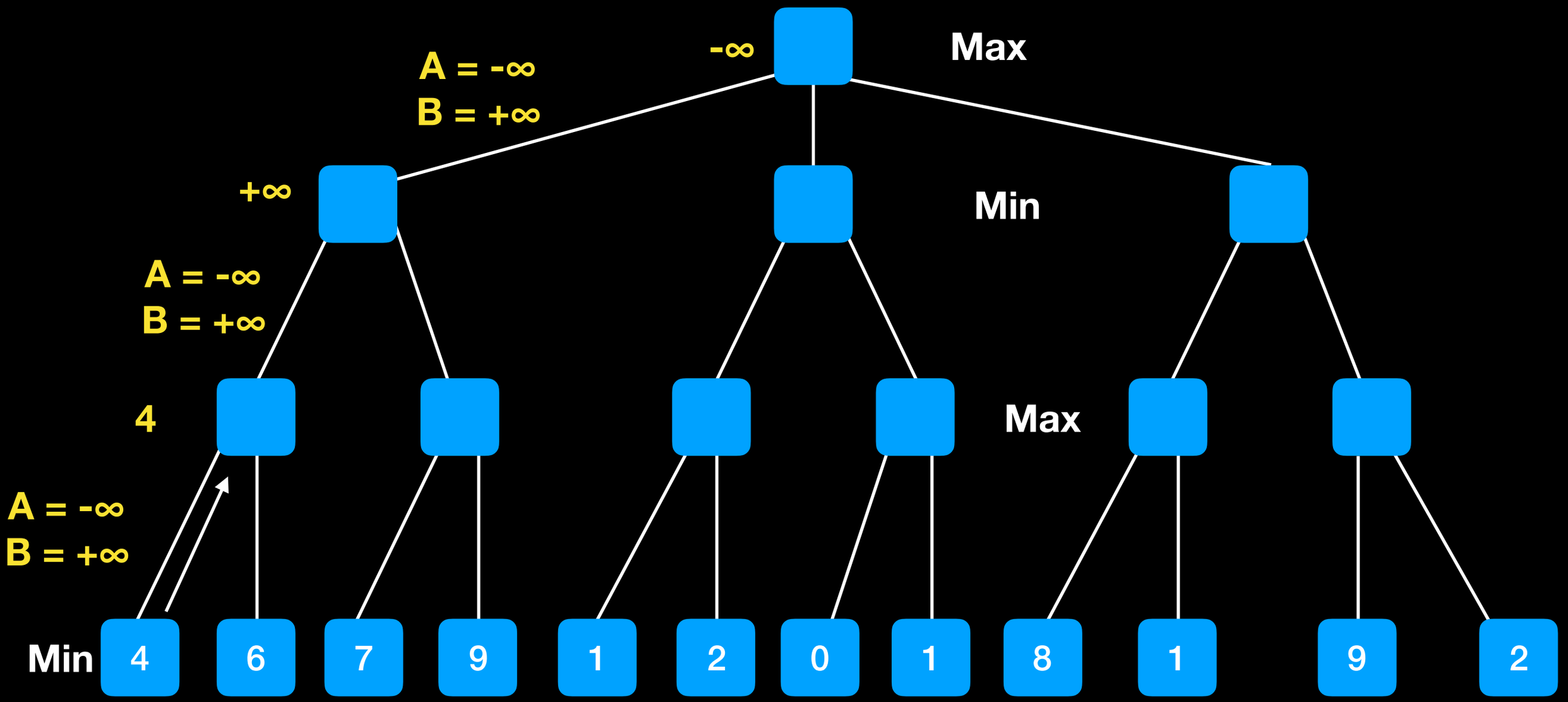

Max
Min
Max
Min
4
6
7
9
1
2
0
1
8
1
9
2


-∞
Max
Min
Max
Min
4
6
7
9
1
2
0
1
8
1
9
2

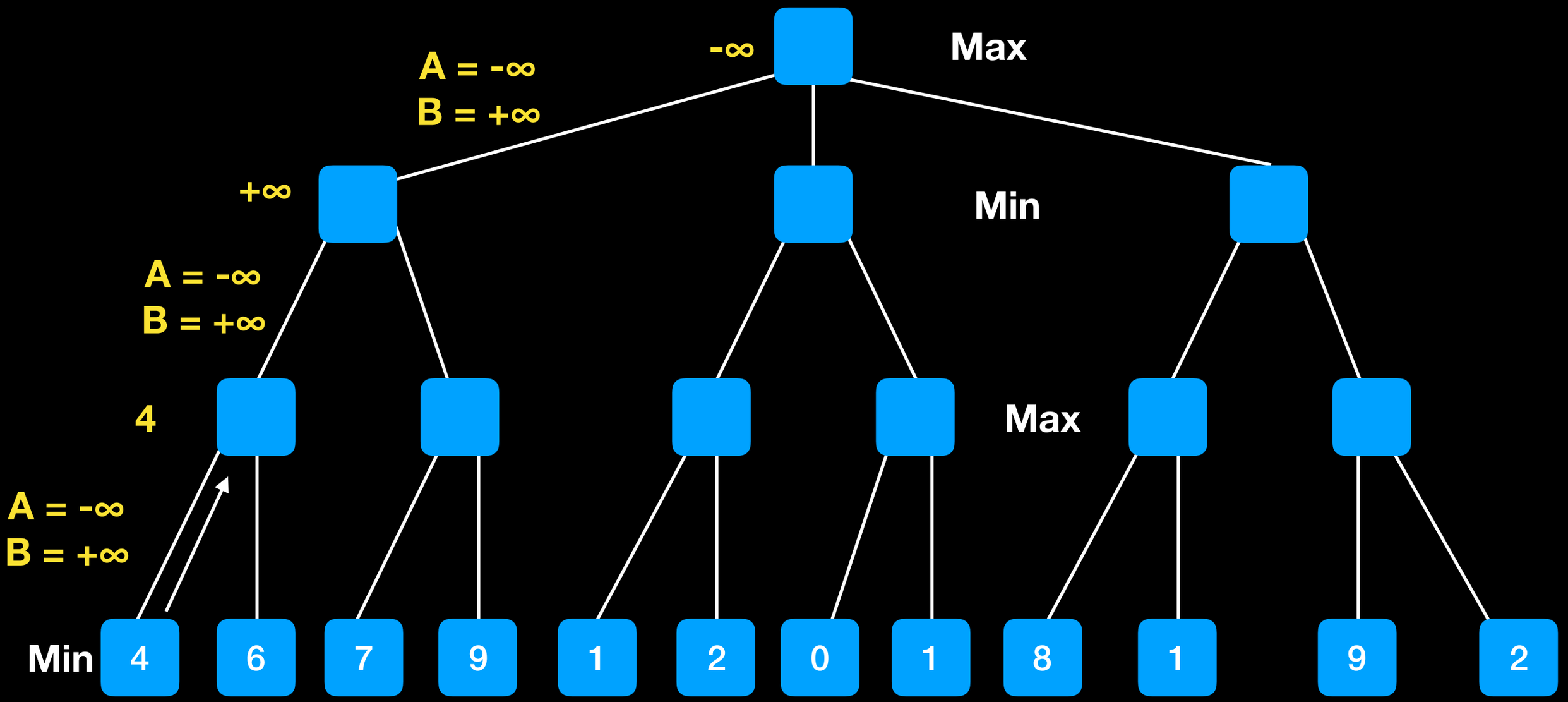

A = -∞
B = +∞
-∞
Max
+∞
Min
Max
Min
4
6
7
9
1
2
0
1
8
1
9
2


A = -∞
B = +∞
-∞
Max
+∞
Min
A = -∞
B = +∞
-∞
Max
Min
4
6
7
9
1
2
0
1
8
1
9
2

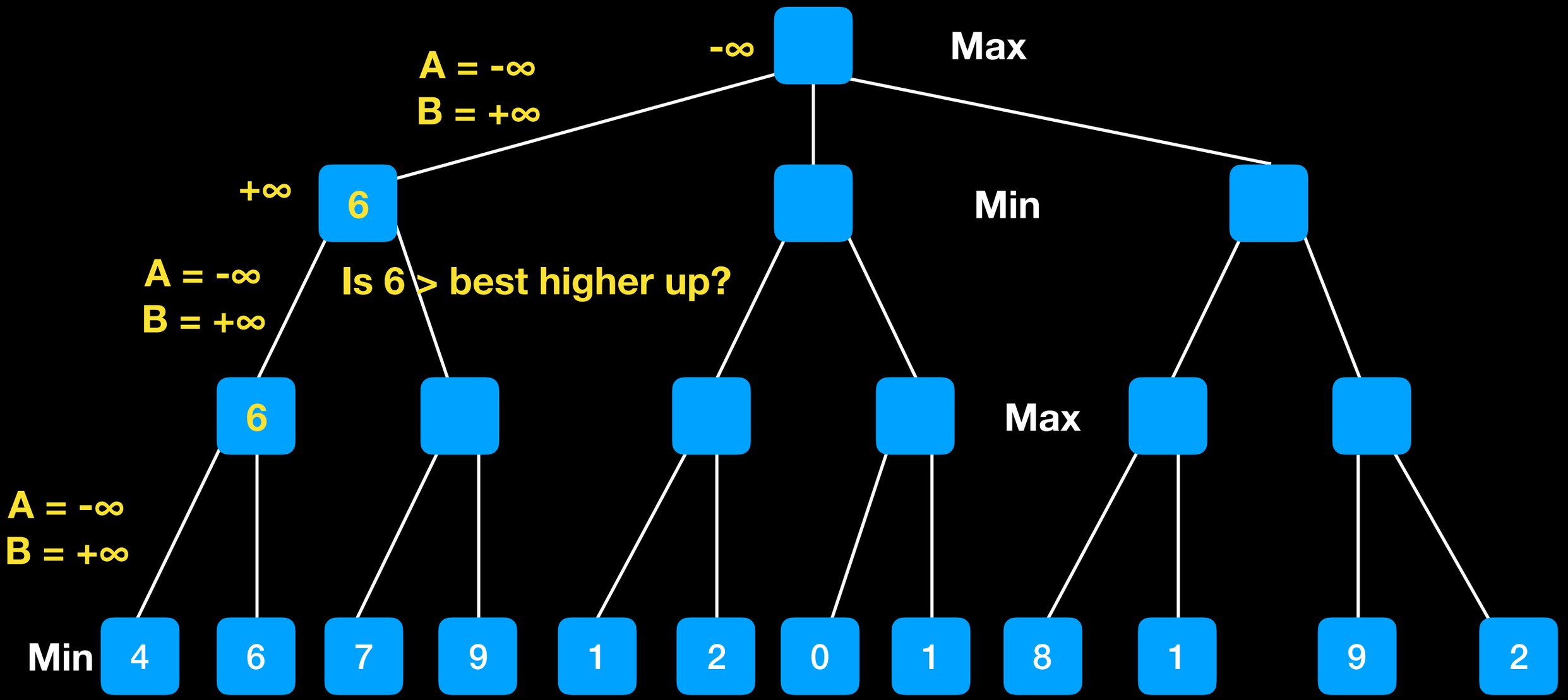

A = -∞
B = +∞
-∞
Max
+∞
Min
A = -∞
B = +∞
-∞
Max
A = -∞
B = +∞
Min
4
6
7
9
1
2
0
1
8
1
9
2

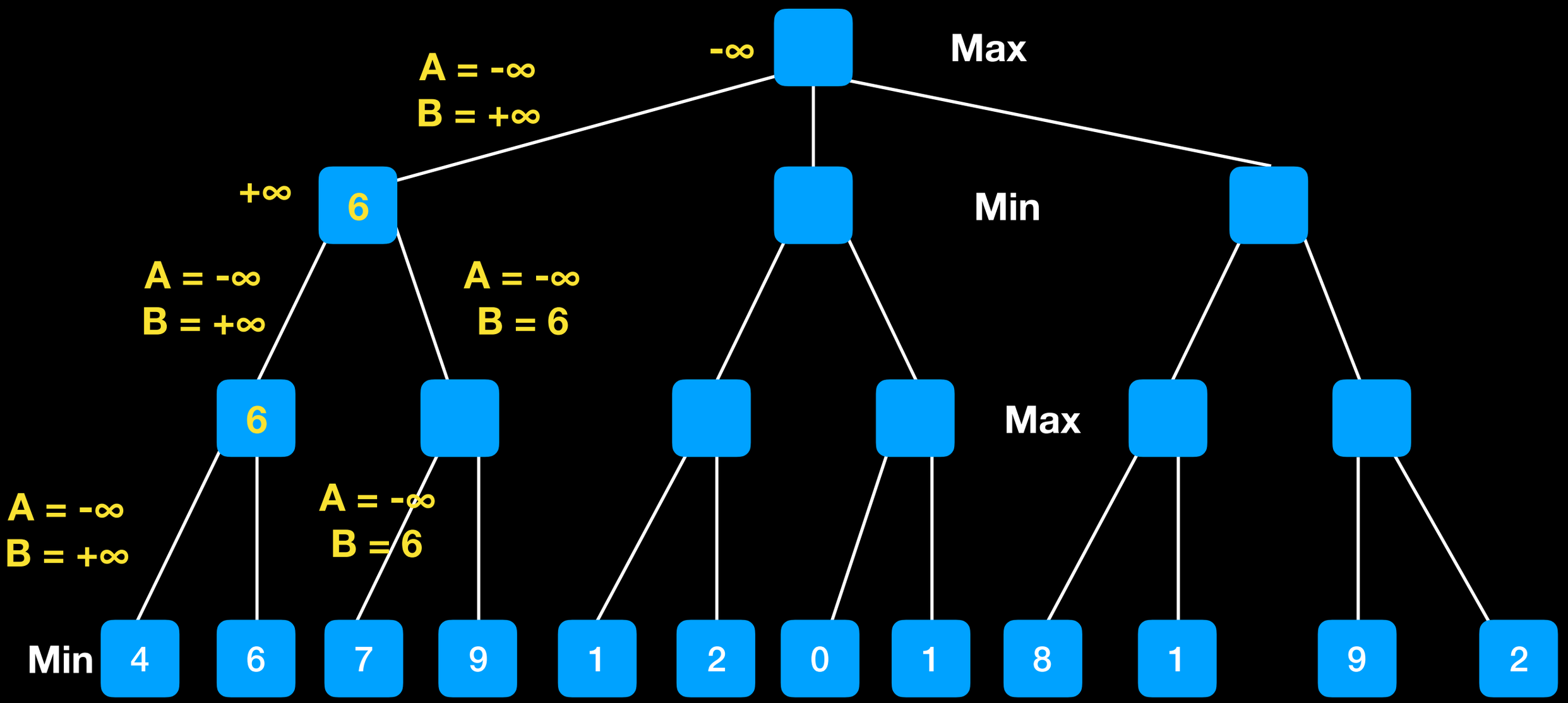


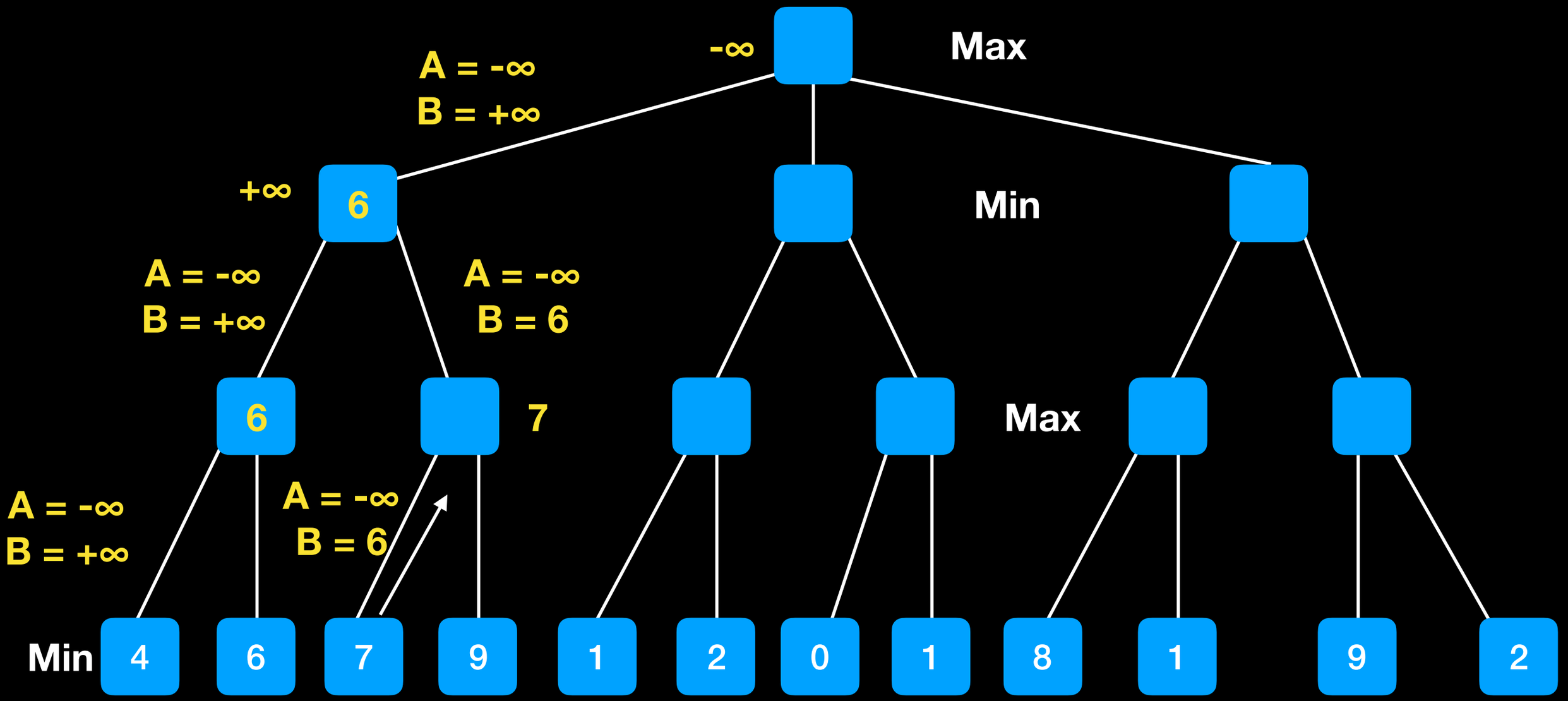
Is 4 > B?



Is 4 > B?


-∞
Max
A = -∞
B = +∞
+∞
6
Min
Is 6 > best higher up?
A = -∞
B = +∞
6
Max
A = -∞
B = +∞
Min
4
6
7
9
1
2
0
1
8
1
9
2


-∞
Max
A = -∞
B = +∞
+∞
6
Min
A = -∞
B = +∞
A = -∞
B = 6
6
Max
A = -∞
B = +∞
A = -∞
B = 6
Min
4
6
7
9
1
2
0
1
8
1
9
2


-∞
Max
A = -∞
B = +∞
+∞
6
Min
A = -∞
B = +∞
A = -∞
B = 6
6
7
Max
A = -∞
B = +∞
A = -∞
B = 6
Min
4
6
7
9
1
2
0
1
8
1
9
2

-∞

Max

A = -∞
B = +∞

+∞

6

Min

A = -∞
B = +∞

A = -∞
B = 6

6

7 > 6?

Max

A = -∞
B = +∞

A = -∞
B = 6

Min

4

6

7

9

1

2

0

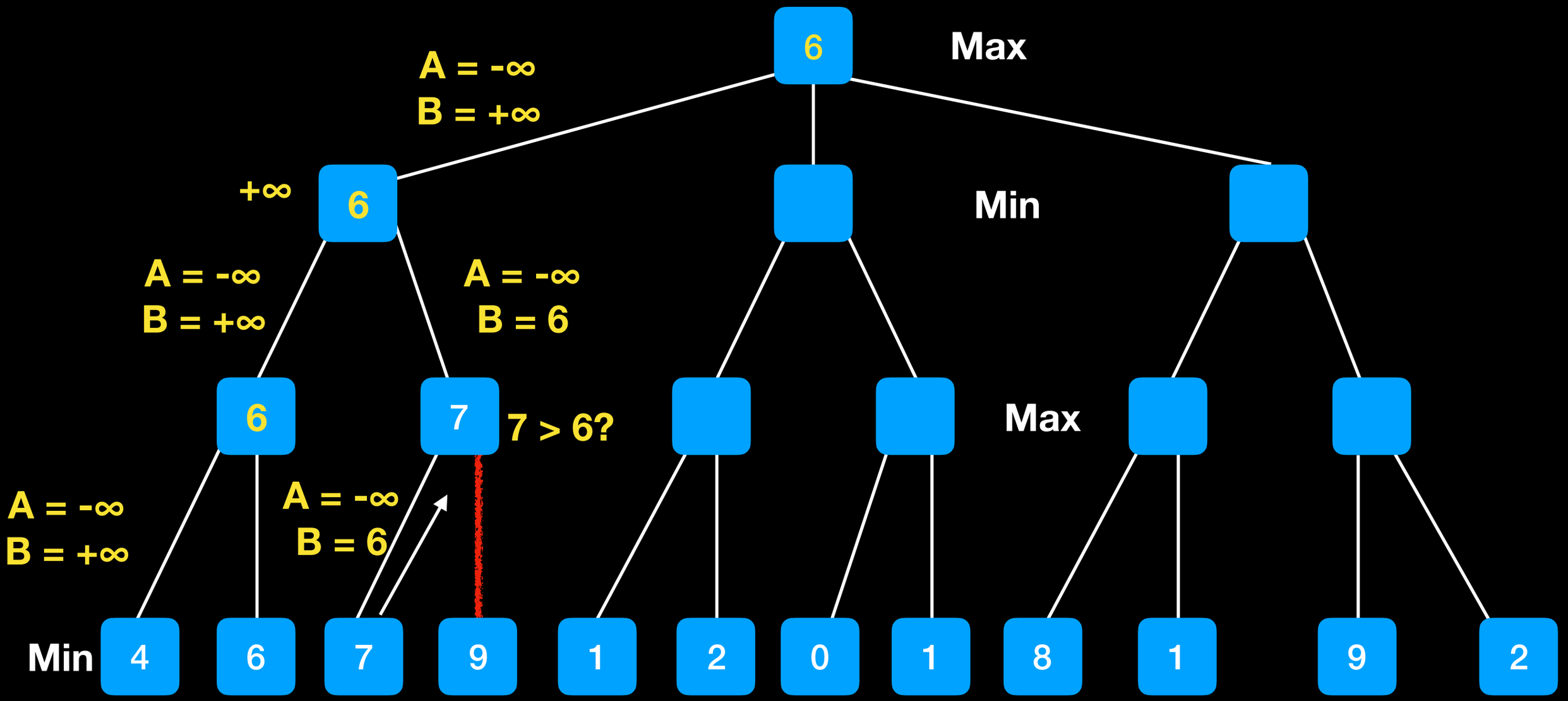
1

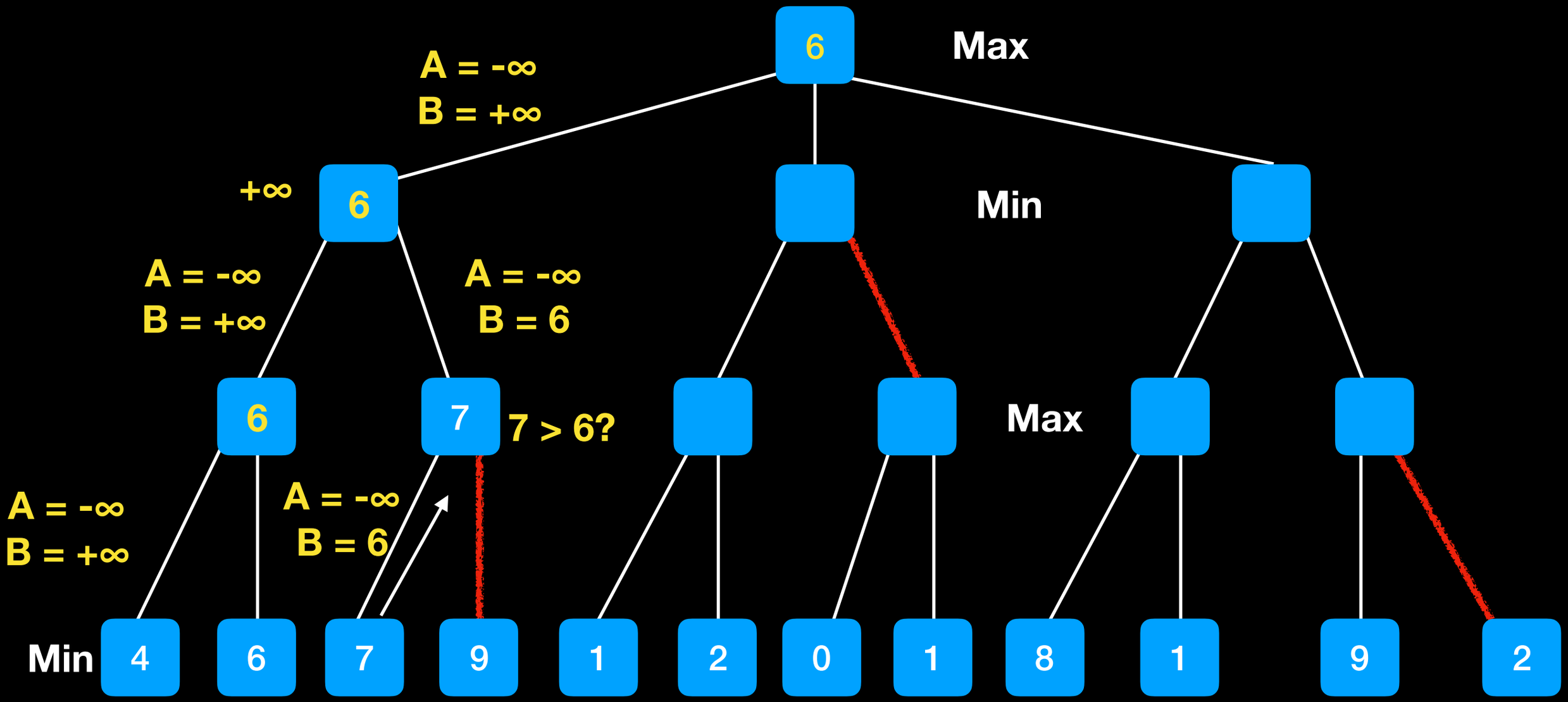
8

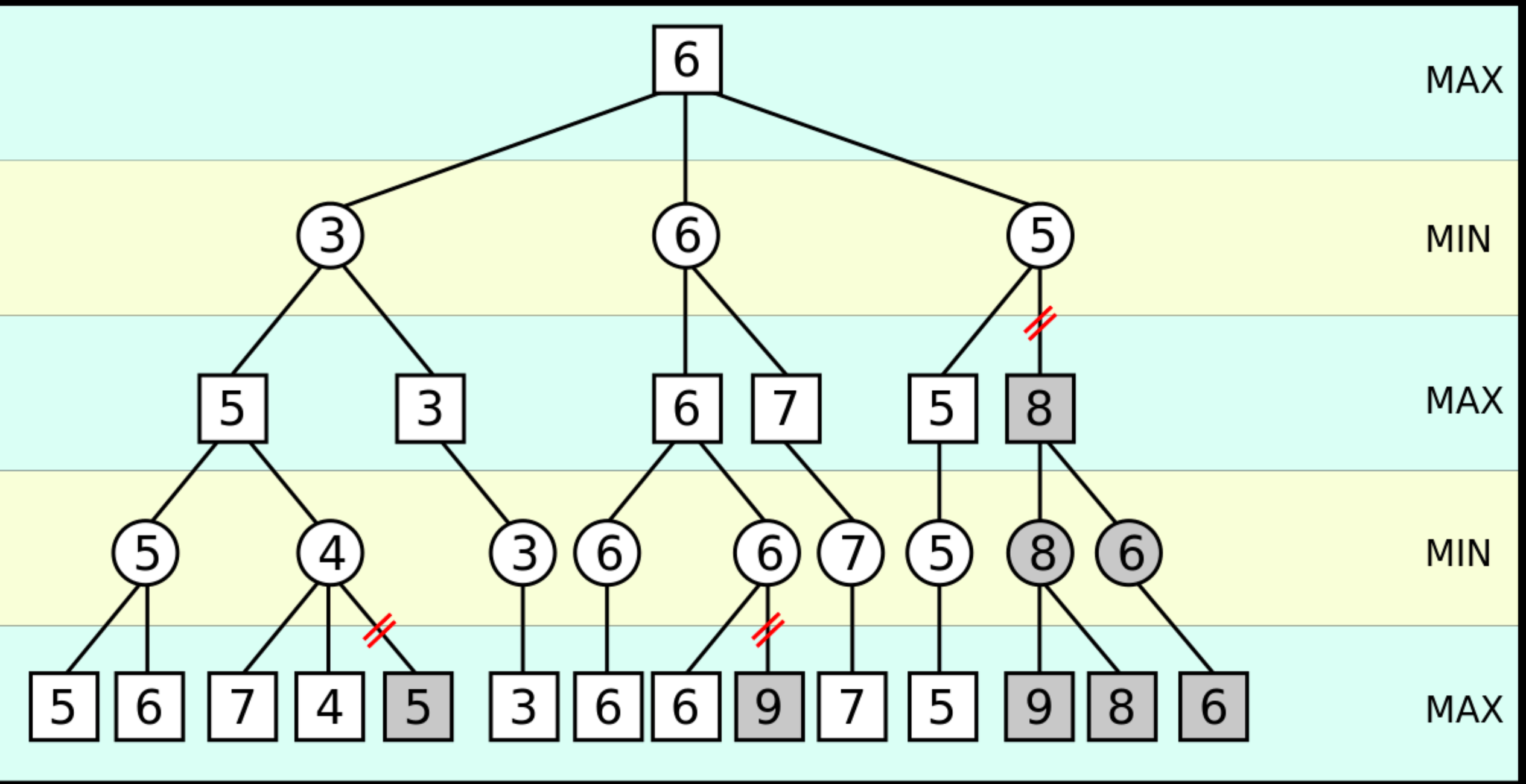
1

9

2


6
Max
A = -∞
B = +∞
+∞
6
Min
A = -∞
B = +∞
A = -∞
B = 6
6
7
7 > 6?
Max
A = -∞
B = +∞
A = -∞
B = 6
Min
4
6
7
9
1
2
0
1
8
1
9
2


6
Max
A = -∞
B = +∞
+∞
6
Min
A = -∞
B = +∞
A = -∞
B = 6
6
7
7 > 6?
Max
A = -∞
B = +∞
A = -∞
B = 6
Min
4
6
7
9
1
2
0
1
8
1
9
2


6
3
6
5
5
3
6
7
5
8
5
4
3
6
6
7
5
8
6
5
6
7
4
5
3
6
6
9
7
5
9
8
6
MAX
MIN
MAX
MIN
MAX

$$\text{No. of moves} = b^d$$

$$\text{No. of moves} = b^d$$

chess: $b \sim 35, d \sim 80$

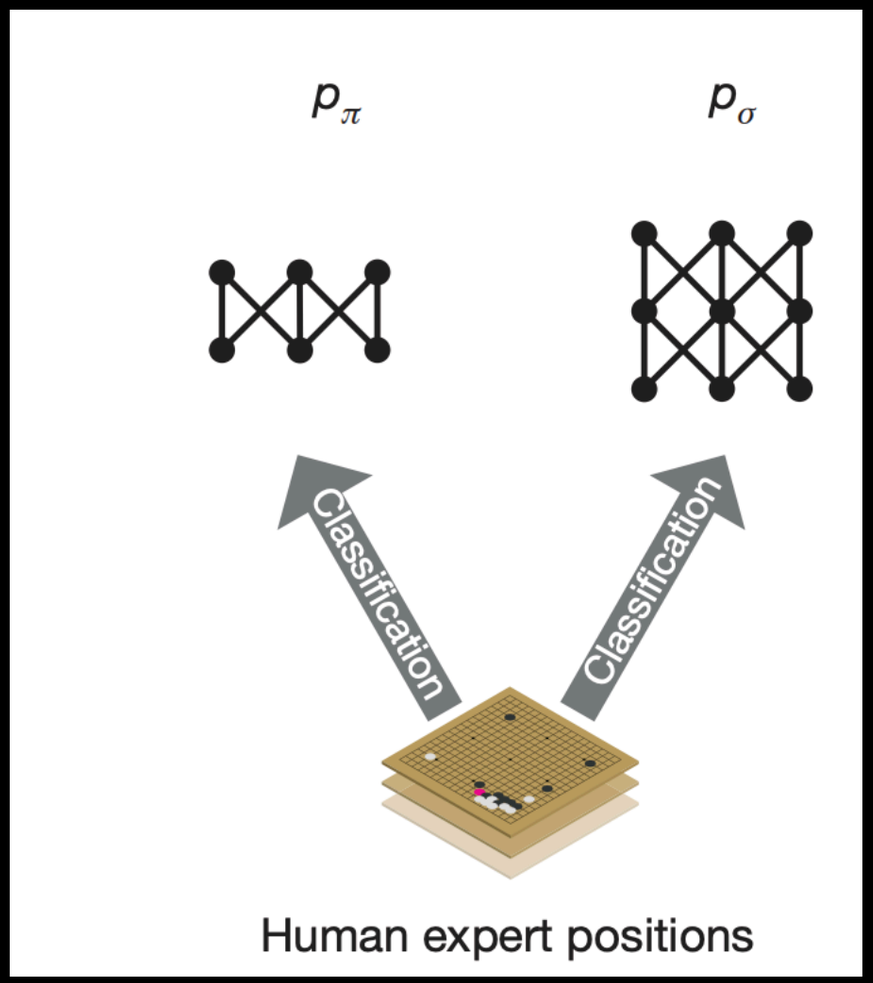
No. of moves $= b^d$

chess: $b \sim 35, d \sim 80$

Go: $b \sim 250, d \sim 150$

Rollout Policy
Fast

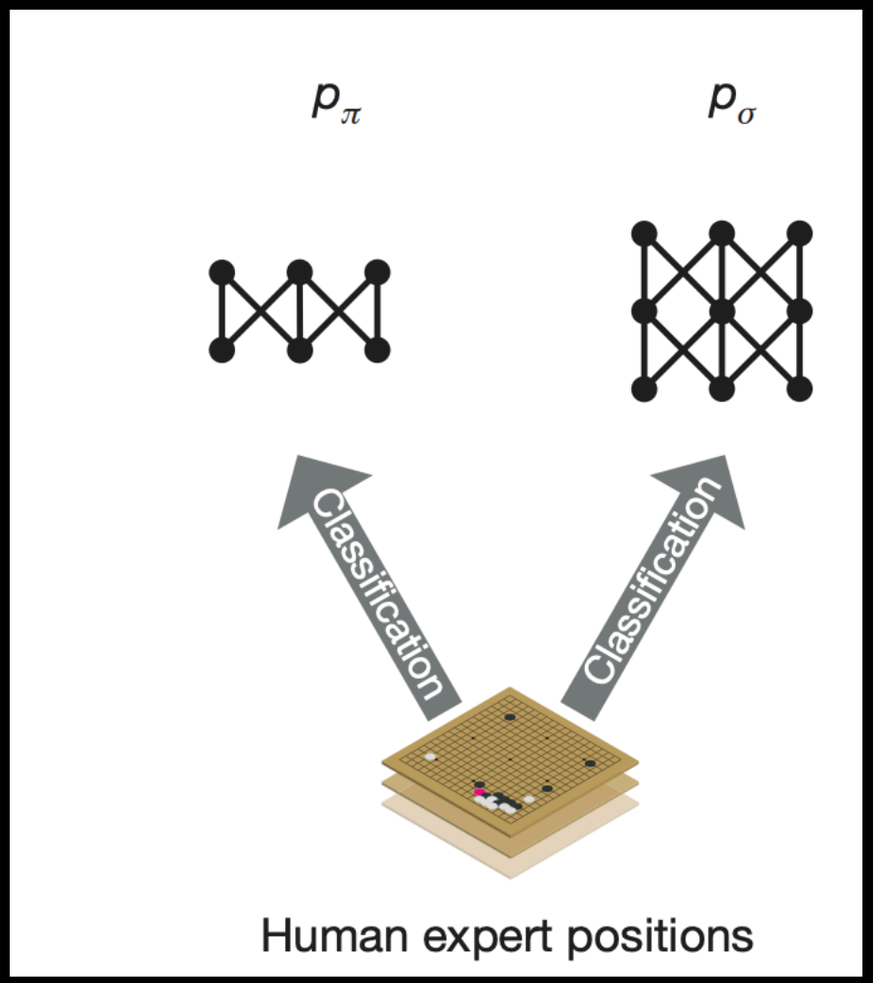
SL Policy



Used online game database of top players
30 million positions.

Rollout Policy
Fast

SL Policy



Used online game database of top players
30 million positions.

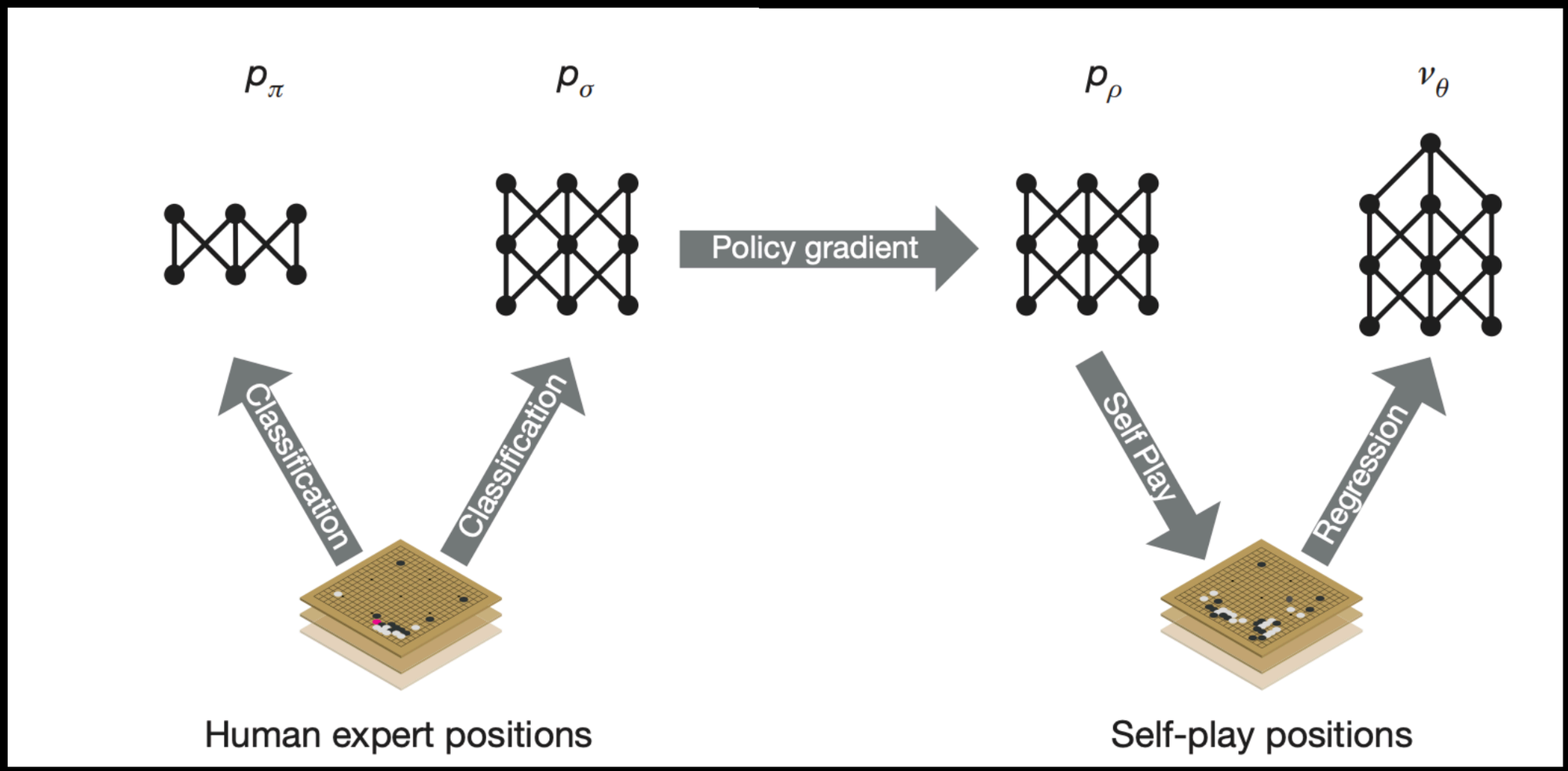
GOAL: Find human master move.

Rollout Policy
Fast

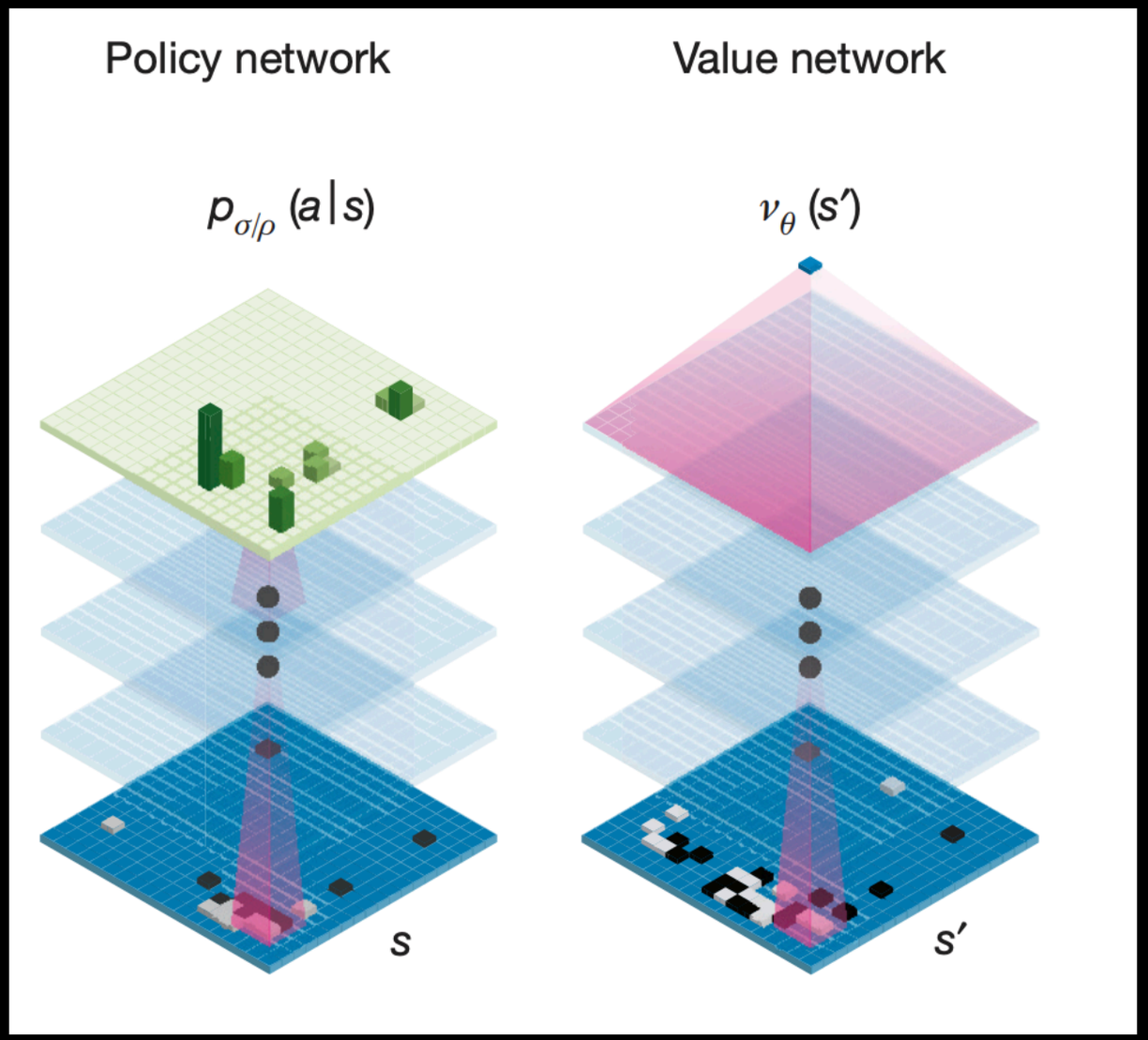
SL Policy

RL Policy Network

Value Network
Predict winner



GOAL: Win


Policy network
Value network
$p_{\sigma/\rho}(a|s)$
$v_\theta(s')$
s
s′

Monte Carlo Tree Search

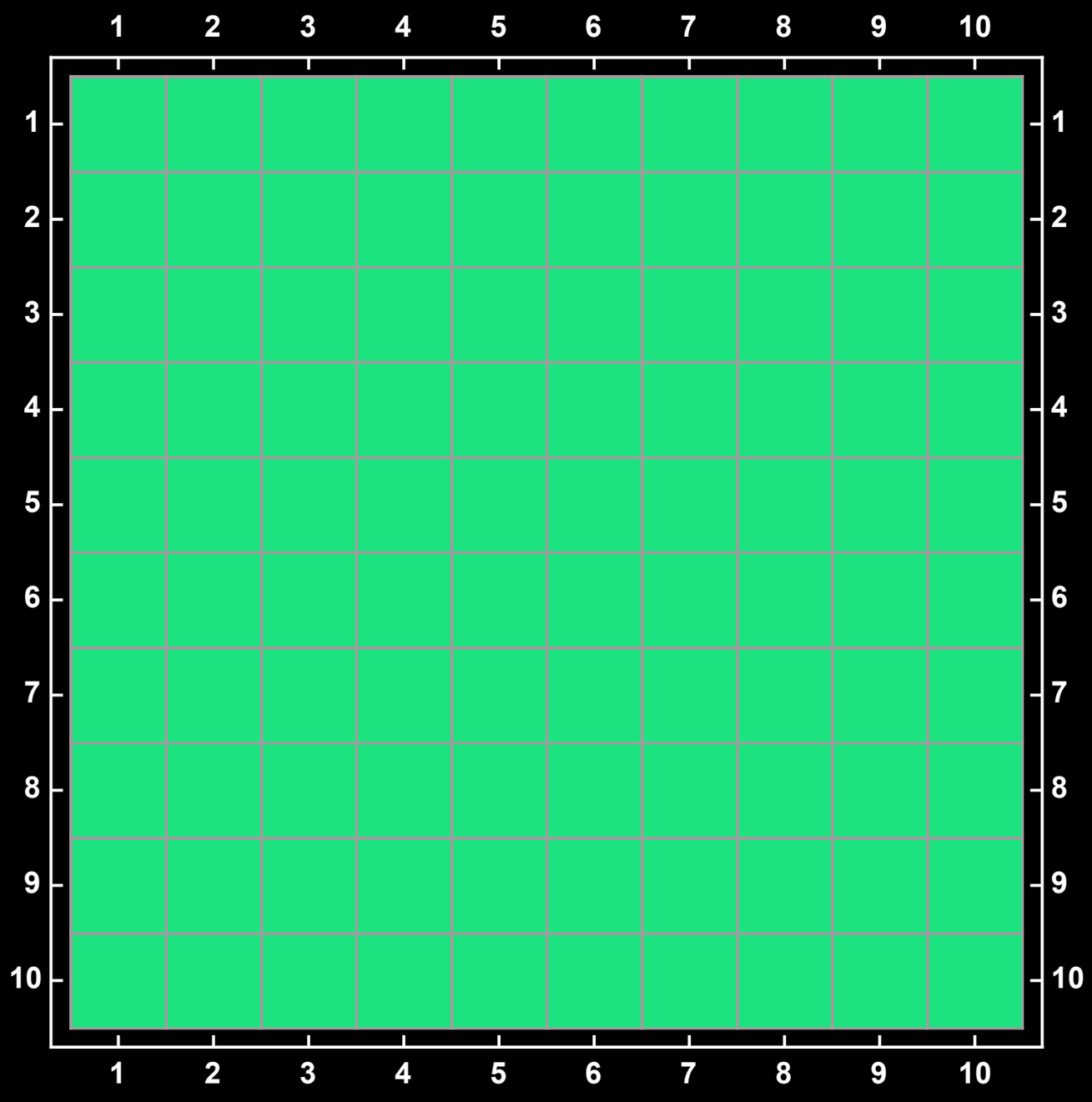
Monte Carlo Tree Search

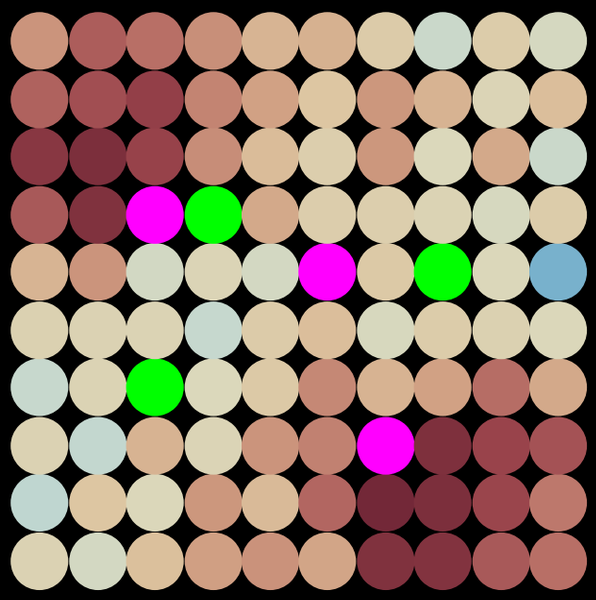
Random Rollout

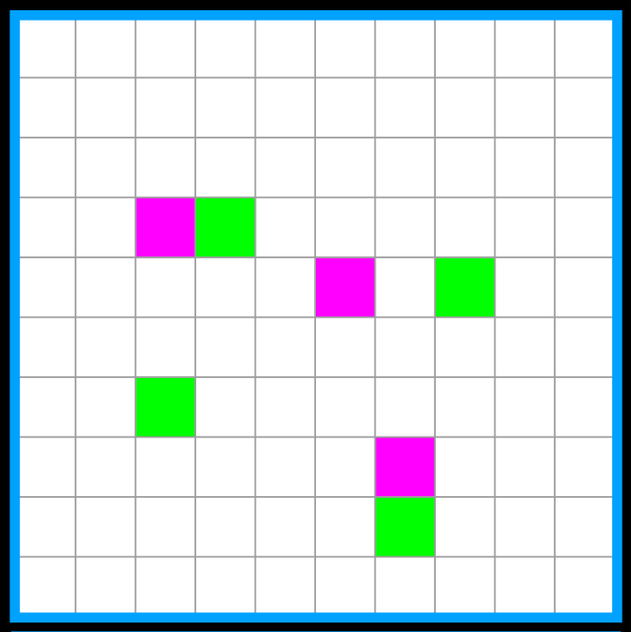
$$V(s_L) = (1-\lambda)v_\theta(s_L) + \lambda z_L$$

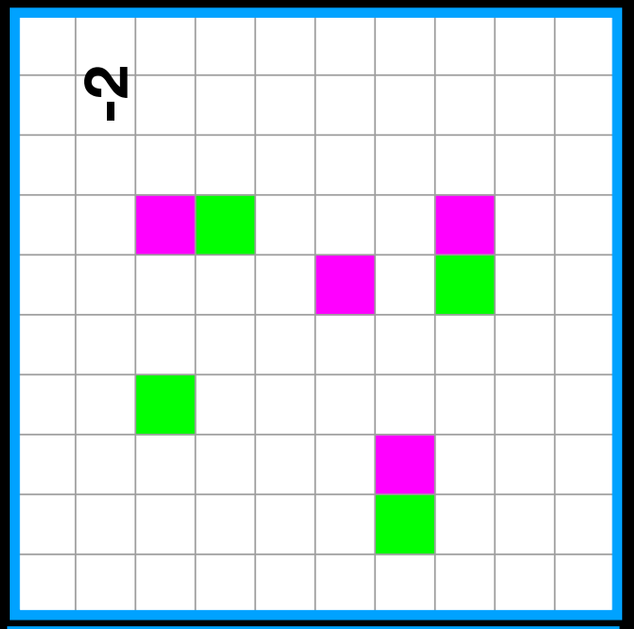
Value Network

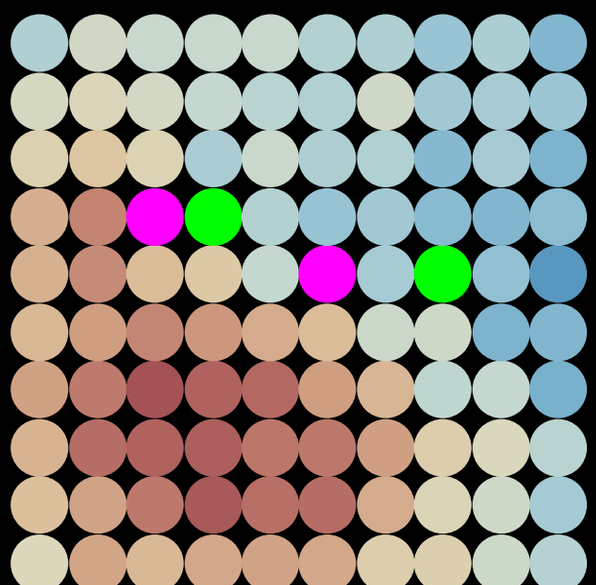
DEMO

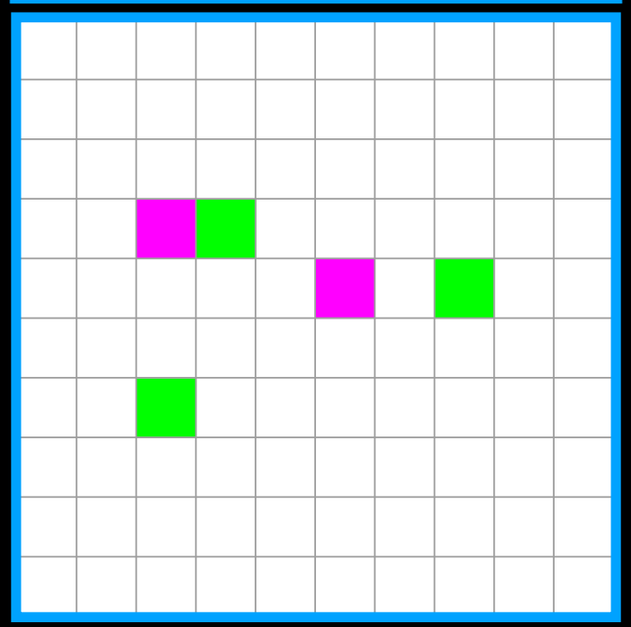


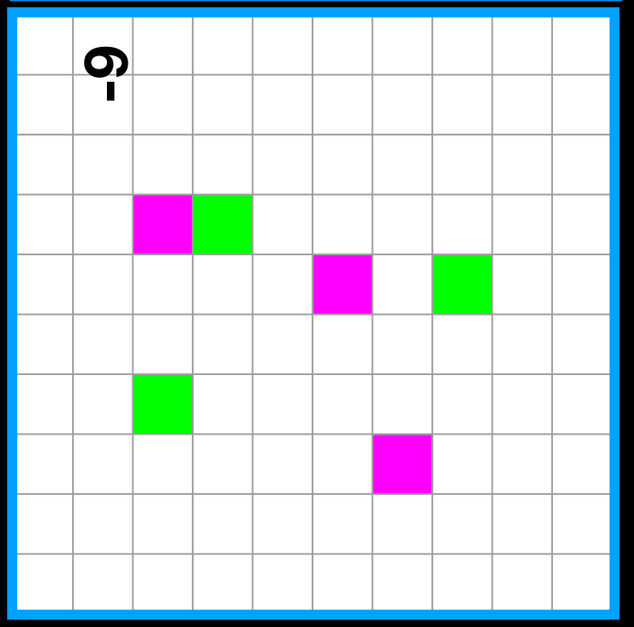


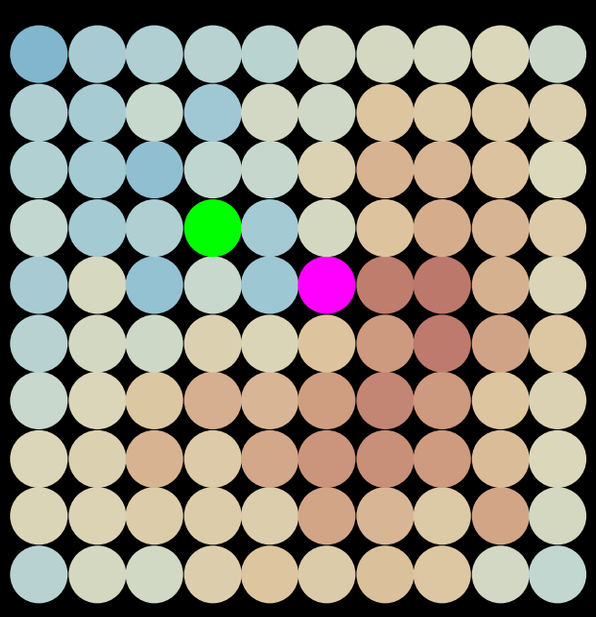


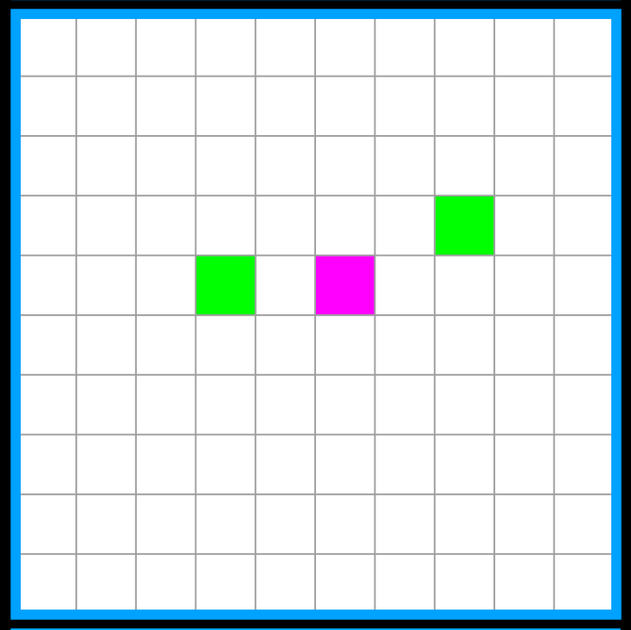


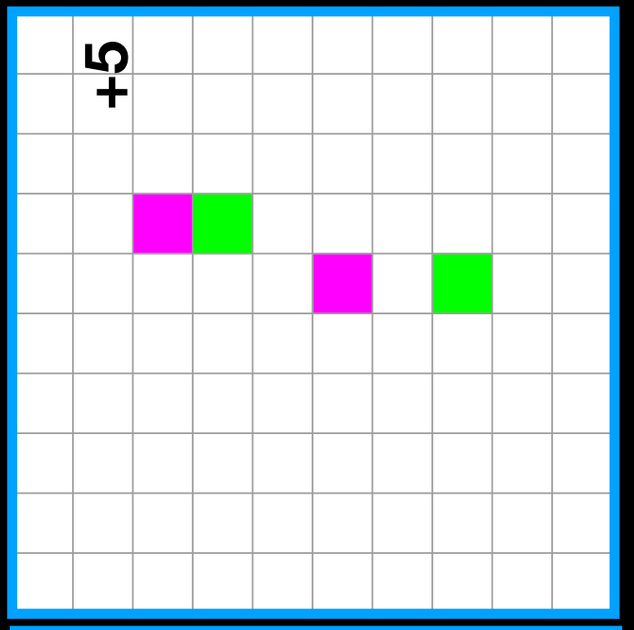


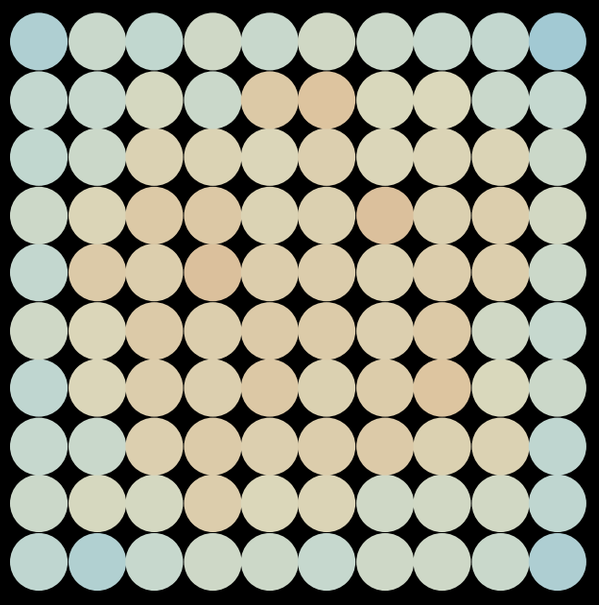


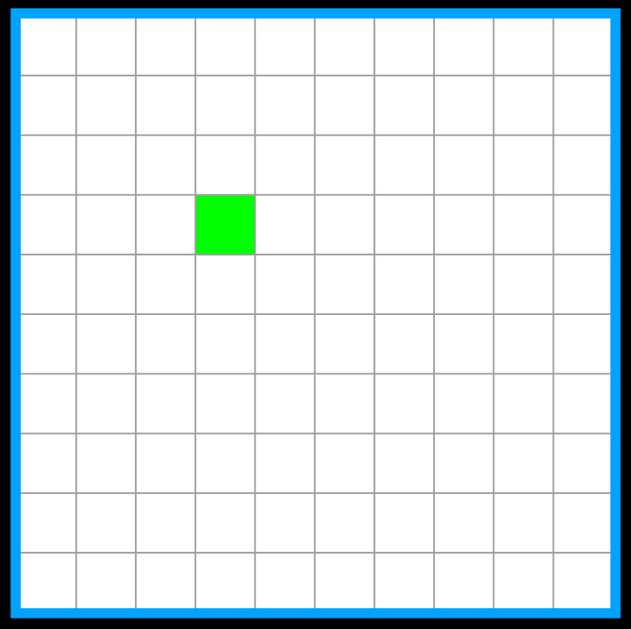


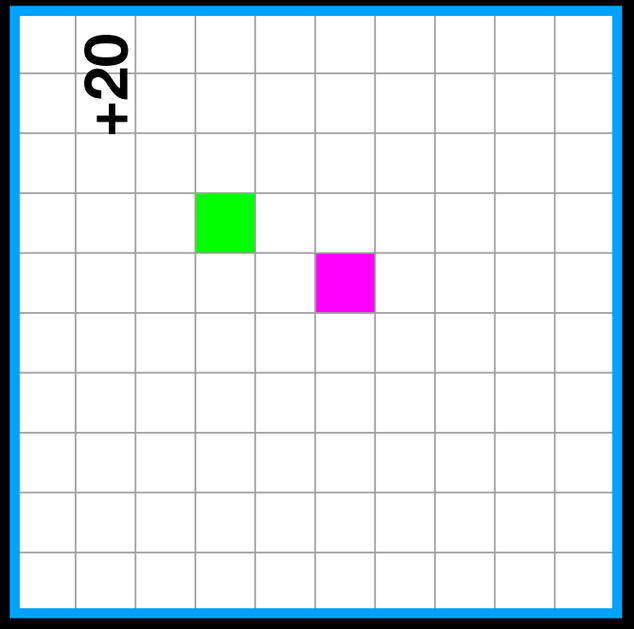










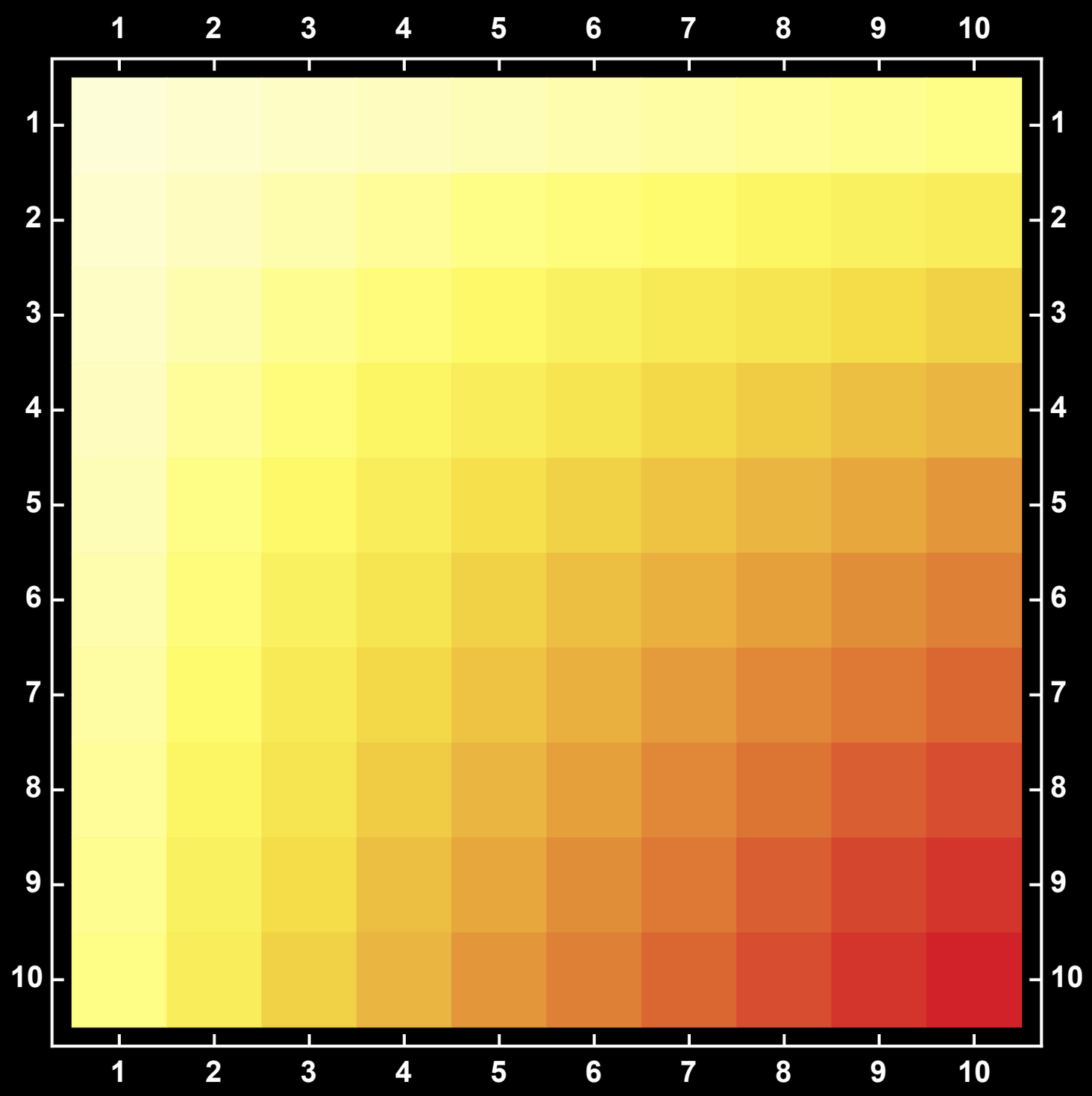


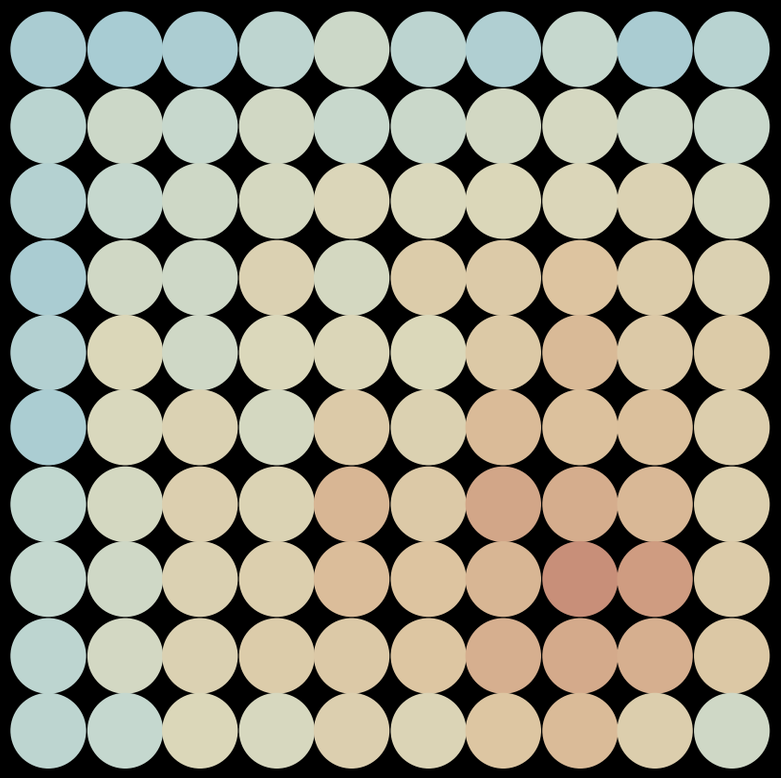
Probability

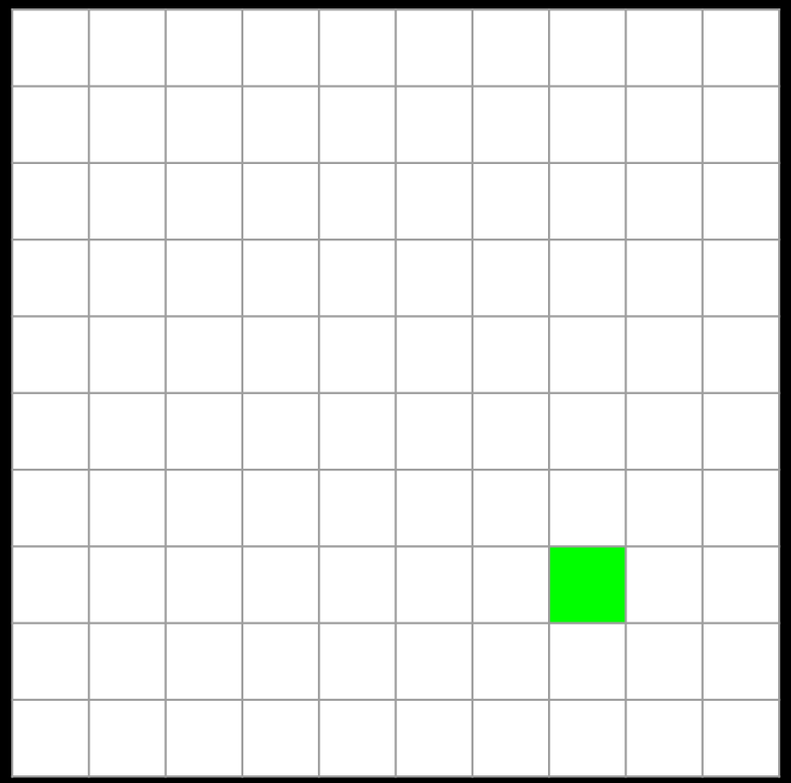
Computer Move

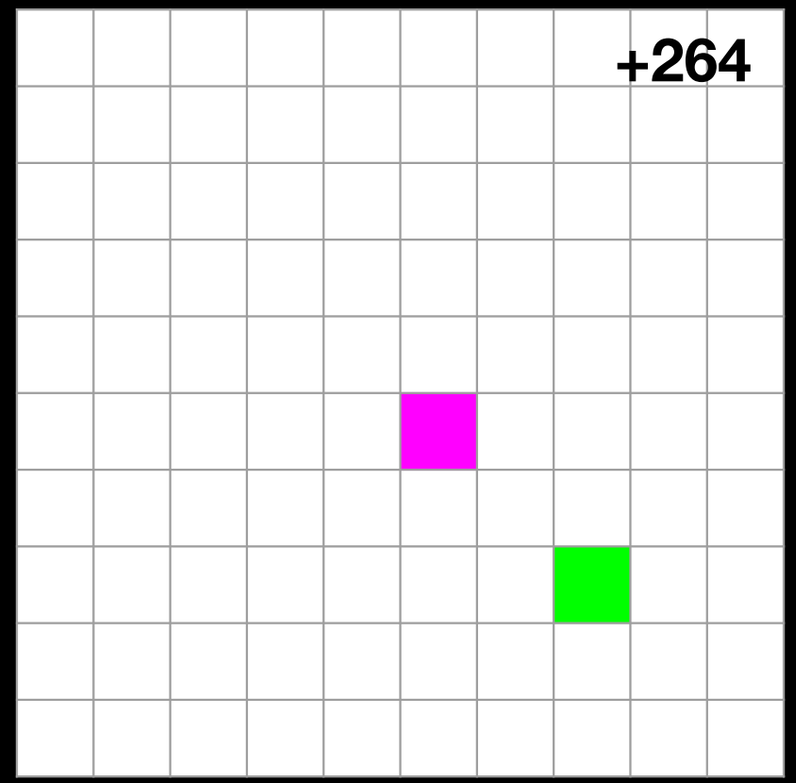
My Move

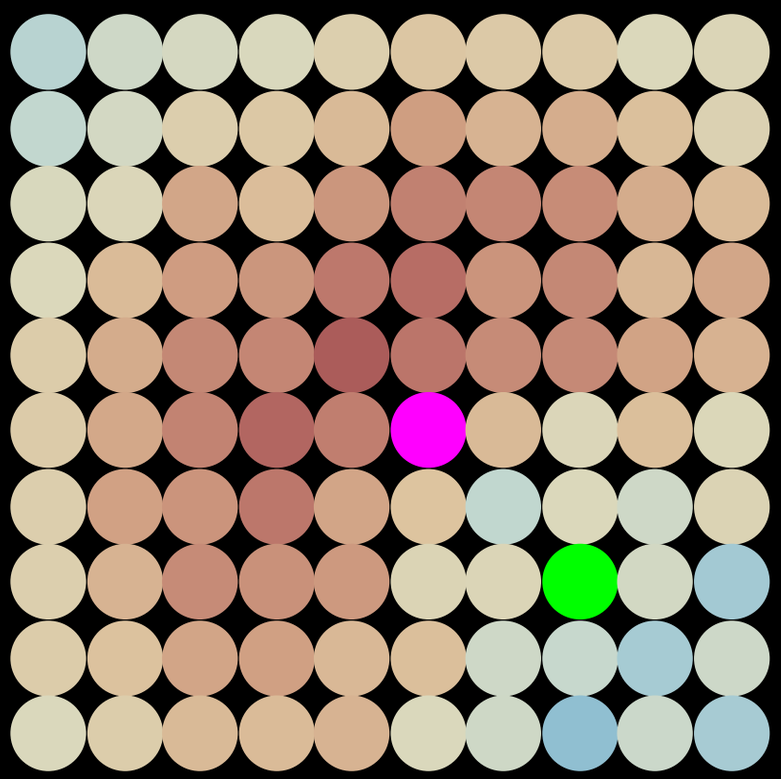
DEMO

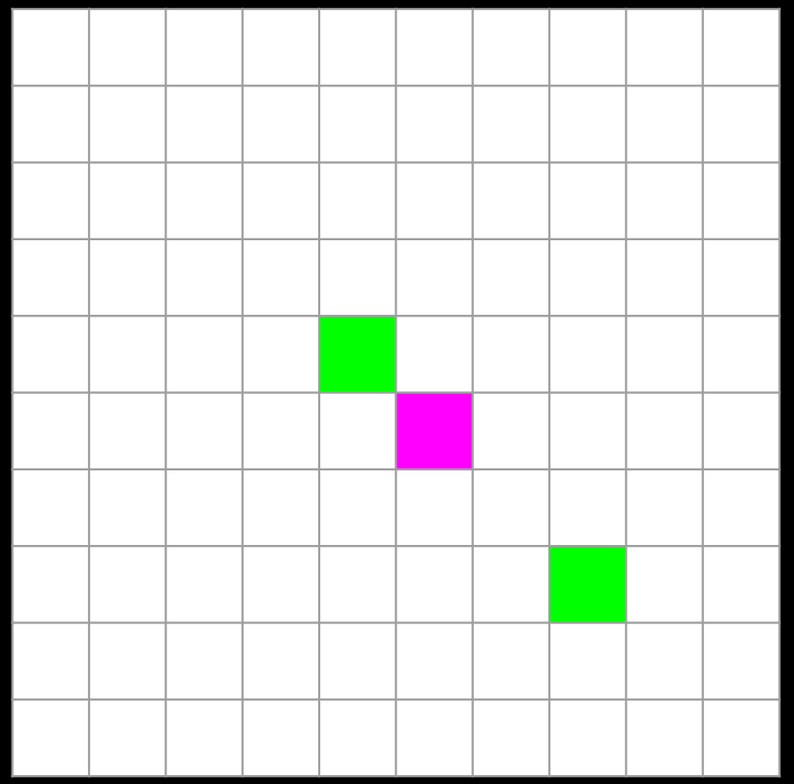






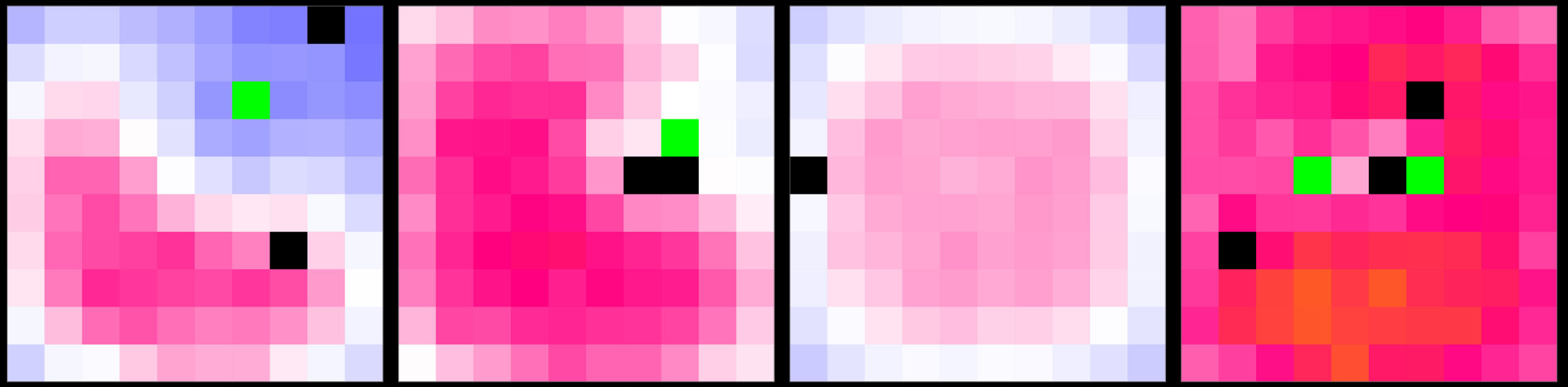

+264





Training Round

Predicted



Target