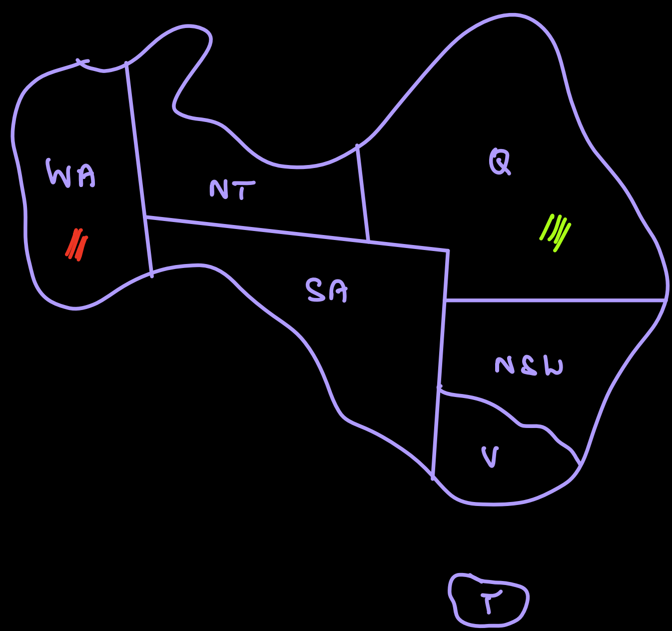
ARC CONSISTENCY:



V → NSW → SA

{R, G, B} {R, B} {B}

consistent not-consistent

For R : B R : B

G : R B : Nothing

B : R So drop B

V → NSW → SA

{R, G, B} {R, ~~B~~} {B}

inconsistent

For R : Nothing

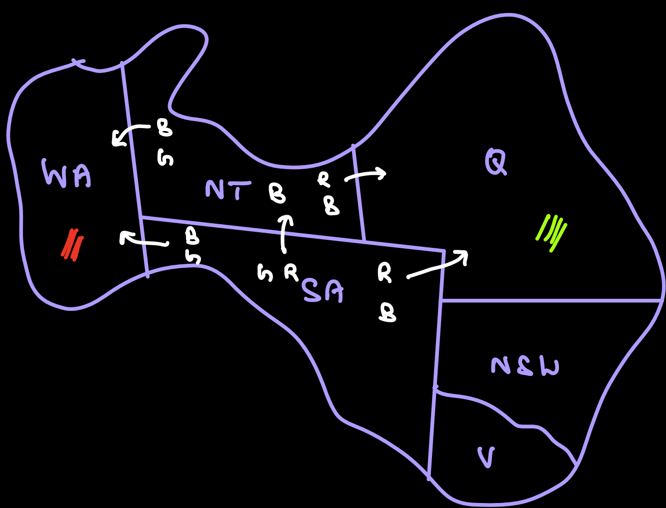
Drop R

$V \rightarrow NSW \rightarrow SA$

$\{\cancel{R}, G, B\}$ $\{R, \cancel{B}\}$ $\{B\}$

Consistent now!

Constraint Propagation!



SA has to be

(B or G) and (G or R) and (R or B)

$\Rightarrow$ no possible value.

COMPLEXITY:

* Complexity of enforcing a single-arc consistency (each variable has atmost d values): Check all x, y, such that $x \in X, y \in Y$: $O(d^2)$

* Binary CSPS (n variable)

  So ${}^nC_2$ possible constraints

  $O(n^2)$ arcs / constraints.

* How many times an arc or edge has to be checked?

  $X \to Y$ is checked everytime

  Y loses a value. As it can have d values

  $O(d)$

* Complexity of Arc Consistency

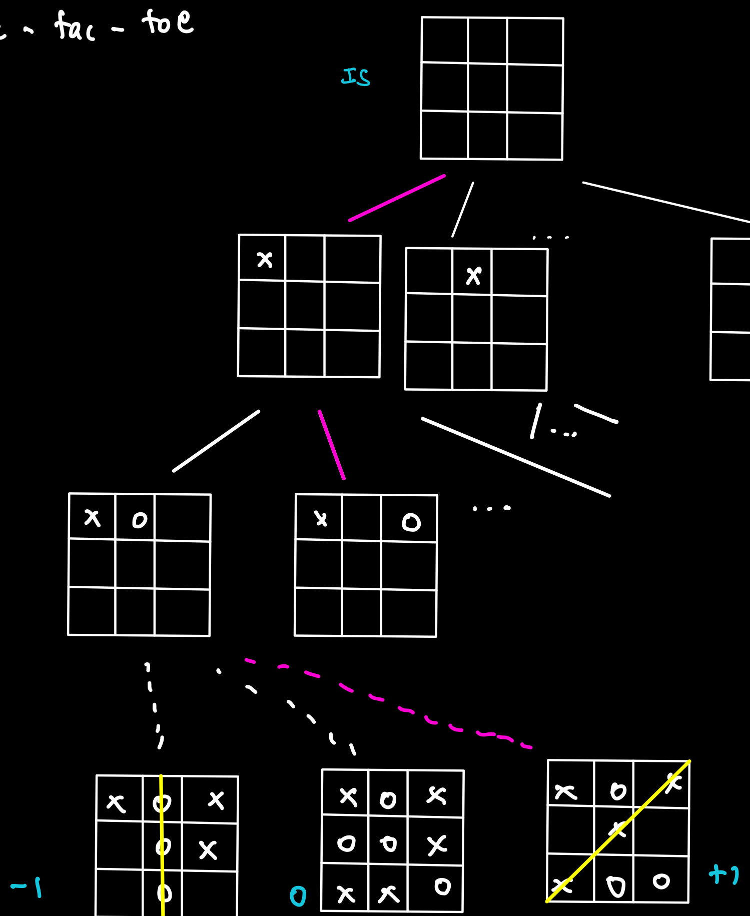
  $O(n^2 d^2 \cdot d) = O(n^2 d^3)$ //

Two - Player Games:

	Deterministic	Chance
Perfect Information (can observe state)	Chess, Go, Othello	Backgammon, Monopoly, Snake and Ladder (Dice throw)
Imperfect Information (no full visibility of current state)	...	Poker, Card Games, Bridge (Decided by what is drawn)

Game Tree:

Tic - tac - toe



+1 : x won

-1 : O won

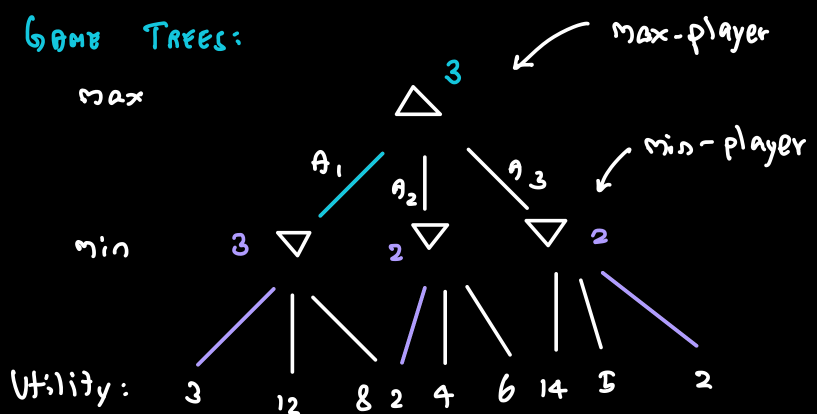
0 : Draw

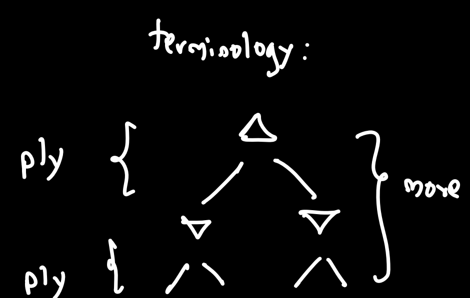
Terminal States

Note: Player: Assume X.

* Finite tree
* But terminal state can be at diffent depth.
* The path shows a path to win state. But unlike other search, we cannot just follow this as half the steps are taken by another player.
* It is timed. Need to make a move fast!
* So now goal is not full path.
    → Given current state, suggest best next move.

Game Trees:





What is the best move at start $A_1/A_2/A_3$?

→ Assume we play against the best player.

So, if A_1, opponent chooses 3

$A_2 \rightarrow 2$

$A_3 \rightarrow 2$

So A_1 is best!

"MINIMAX ALGORITHM"

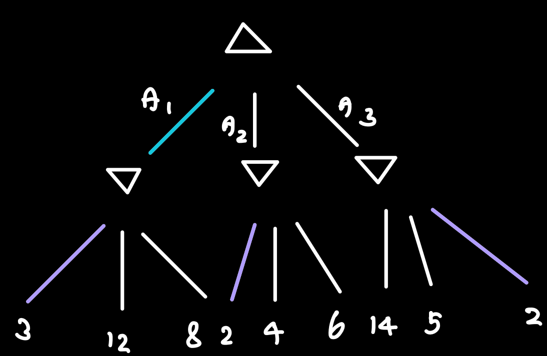
Can we use DFS to get the values?

TC : $O(b^d)$ → very bad!

SC : $O(bd)$

* It is timed. So we can only see, say next 4 rounds.
* We need a way to estimate the partial state after 4 rounds, to decide the move.

$fn(n)$ = estimate the utility.



Check A_1 3, 12, 8 and we know we get 3.

⇒ Check A_2's 2, now no need to check 4 or 6.

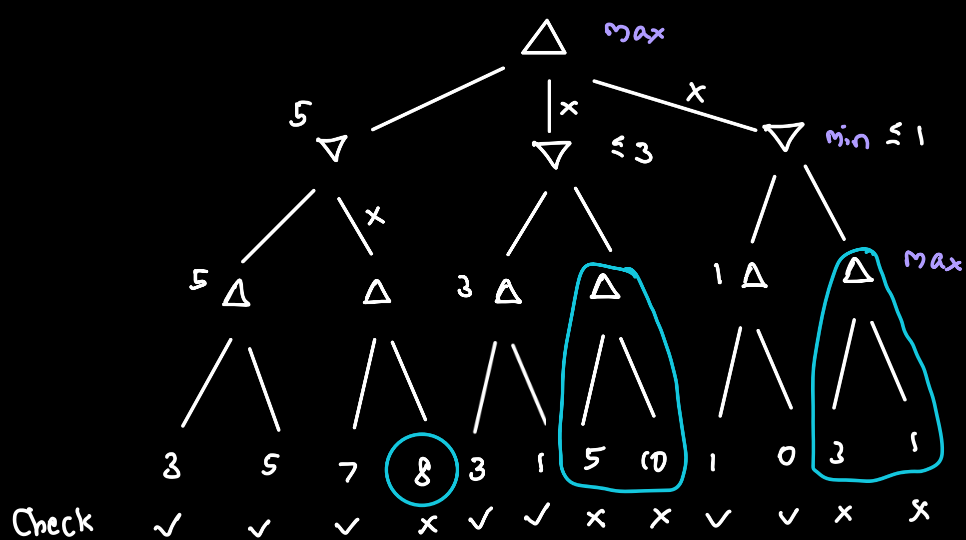
So best case is, we check min from lowest to highest. max from highest to lowest.

α-β PRUNING:

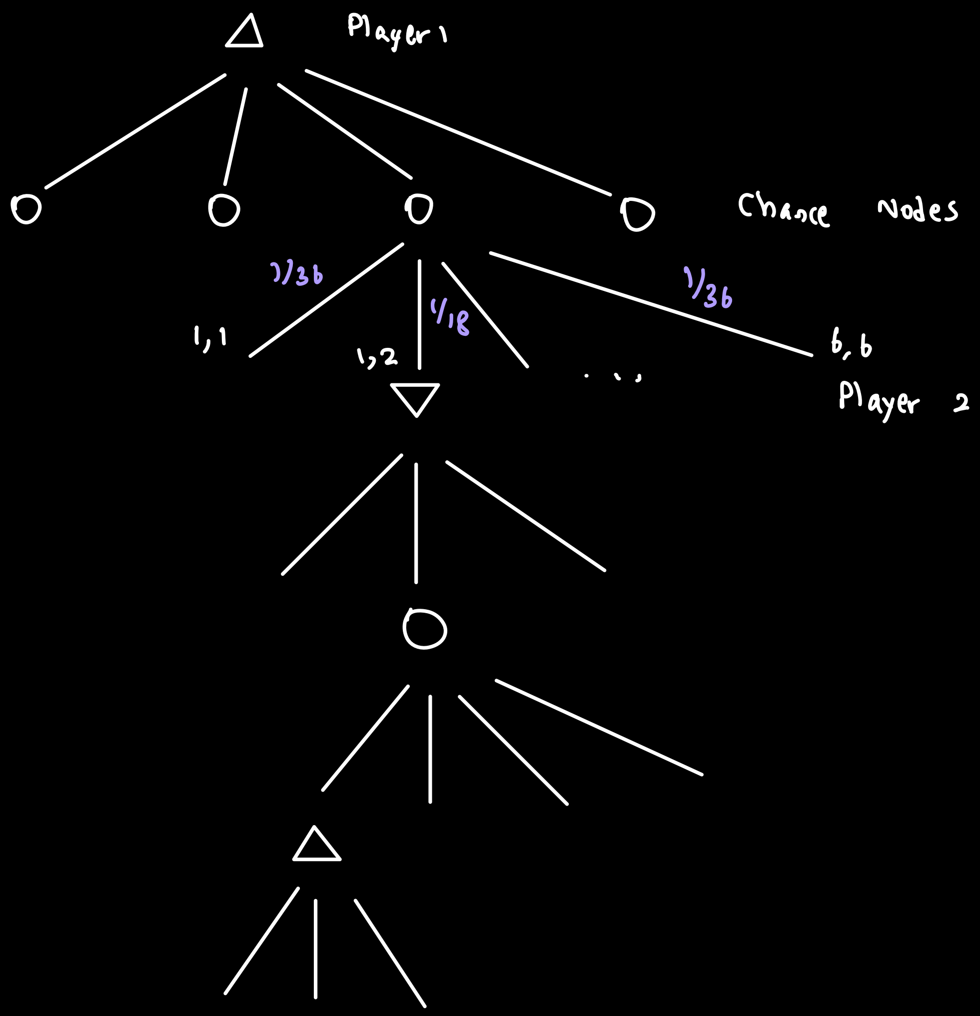
- depth d
- branching factor b

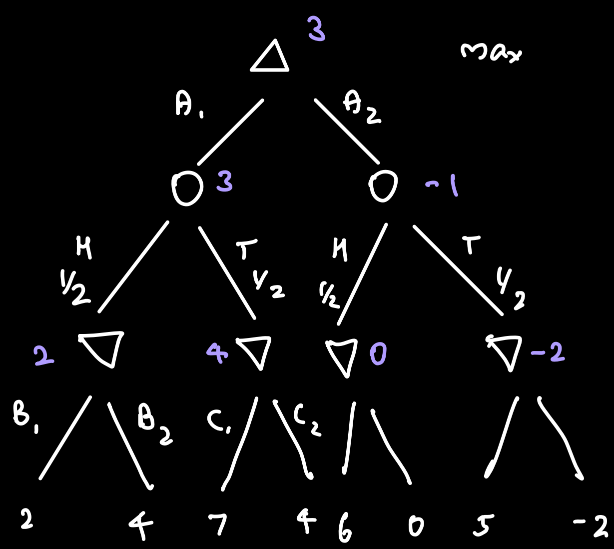
* Standard minimax $O(b^d)$
* Best case scenario for α-β pruning

$$O(b^{d/2})$$



Games with Chance:





expected minimax.

"Shallow depth with a serious evaluation functions".

Minimax:

If evaluation function for nodes

A	B	C	D
2	3	4	5
1000	1002	1050	200

Both are same as long as order is same.