<u>Best first search</u>

↳ selects the path which is best
↳ DFS & BFS combination
↳ uses heuristic $H(n)$ & search
↳ at each step, most promising node can be used.
↳ Greedy algorithm by priority queue
$\boxed{f(n) = g(n)}$

↳ informed

<u>Example</u>

Suppose, we have the graph as follows:

→ Advantage

- Can switch DFS & BFS
- More efficient

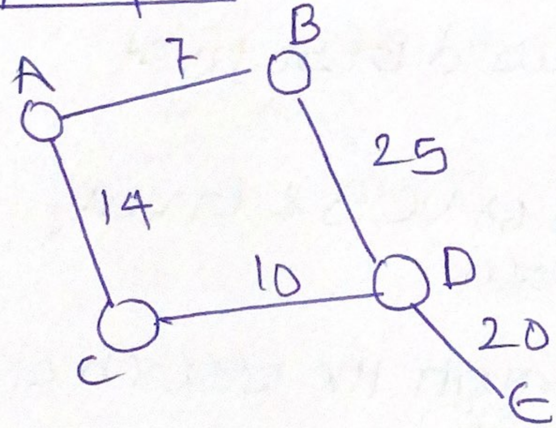
→ Disadvantage

→ behave as unguided

→ Not optimal

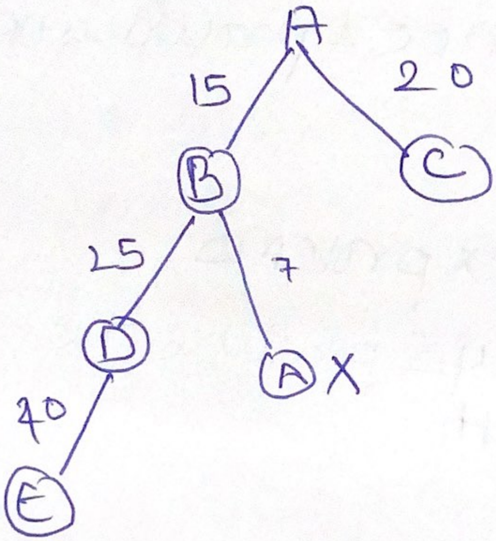
$O(b^m)$ → m depth

## Example




### Heuristic Values

| | |
|---|---|
| A → E | 10 |
| B → E | 15 |
| C → E | 20 |
| D → E | 40. |
| E → E | 0 |




Path followed
A → B → D → E

# A* Algorithm

→ Most commonly used Best first search

→ combined features of UCS & Greedy Best First search

→ shortest path through the search space using Heuristic

→ expands less tree & provides optimal solution

→ complete

→ solve complex problems

→ does not always produce the shortest path

→ more memory.

→ $f(N) = g(N) + h(N)$

$g(N)$ → Actual cost

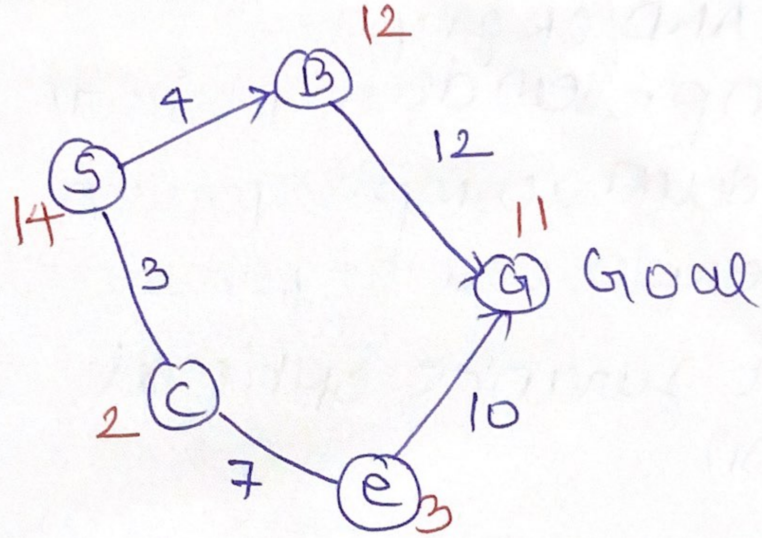
$h(N)$ → estimation cost

→ $TC = O(V + E)$

$= O(b^d)$

$SC = O(b^d)$

# Example




$f(S) = g(S) + h(S)$

$= 0 + 14 = 14$

$S \to B$

$4 + 12 = 16$

$S \to C$

$3 + 5 = 8$

$SC \to e$

$10 + 3 = 13$

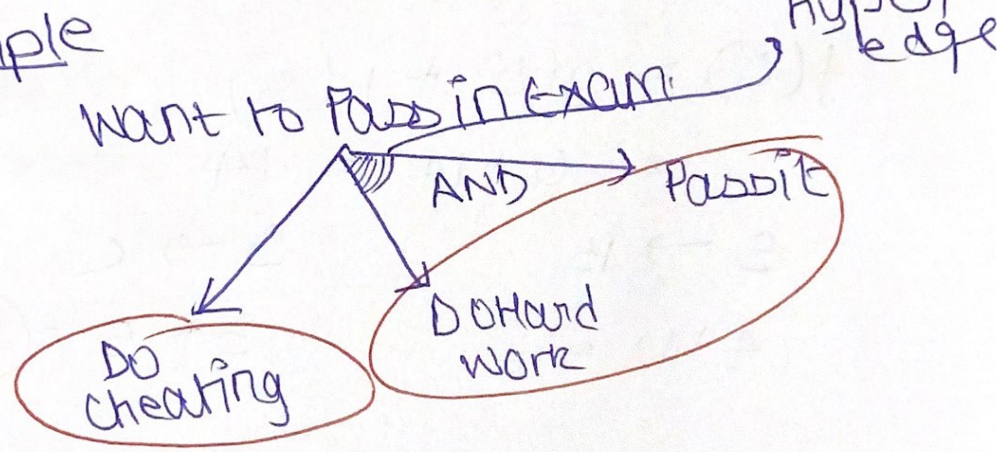
$SCe \to G$

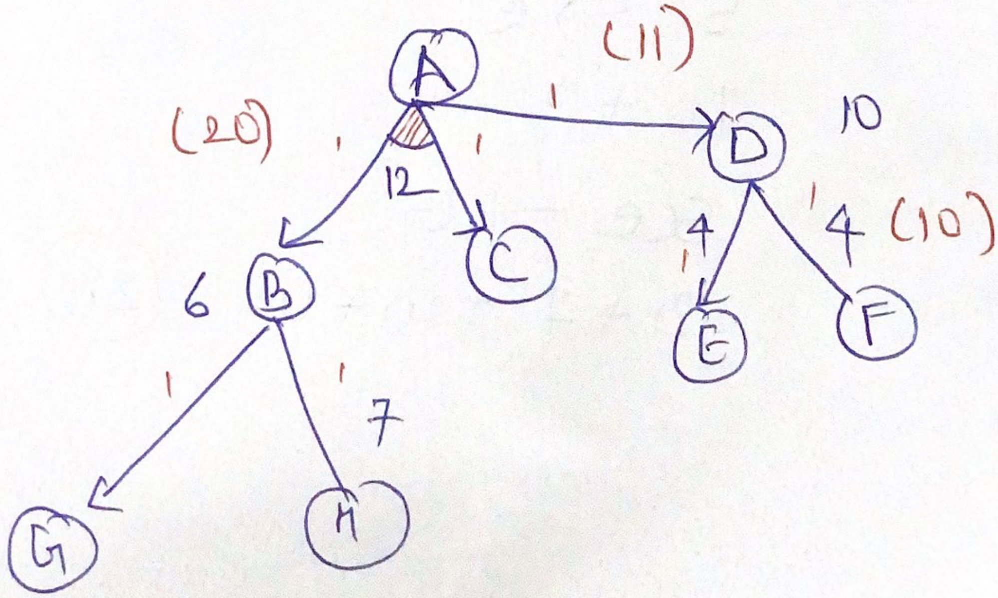
$3 + 7 + 10 + 0 = 20$

# AO* Algorithm

→ Based on AND/OR graph
→ works on problem decomposition
→ breaks down a complex problem
→ does not explore all the paths
→ does not guarantee optimal solution

Example



# Local Search Algorithm

a) does not focus on path, only focuses on solution state

b) works on Greedy

c) best move

d) Has knowledge of only local domain

e) No Backtracking

f) constant Amount of memory

g) Better space complexity

h) incomplete Algorithm

Types Simple

## Hill Climbing Search

a) Evaluate the initial state

b) Loop untill a solution is found or there are no operators left

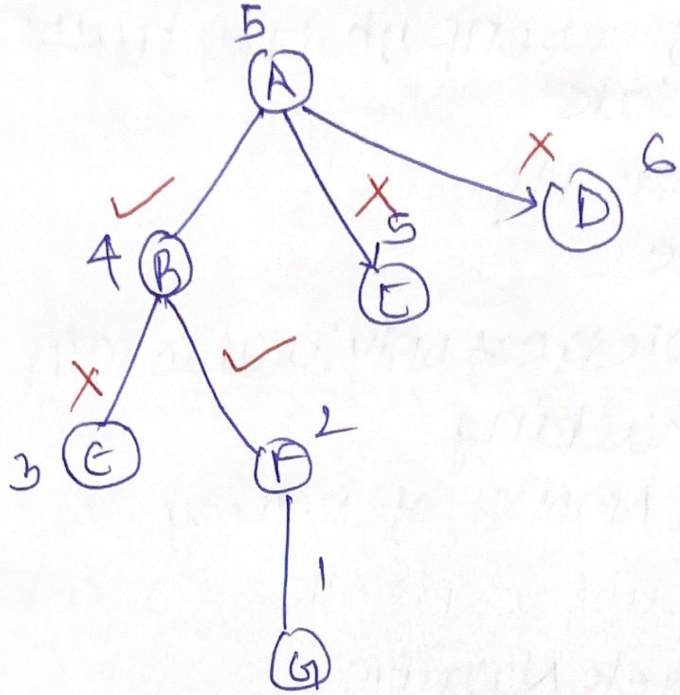
c) select & Apply a new operator

d) Evaluate new state

→ if goal then quit

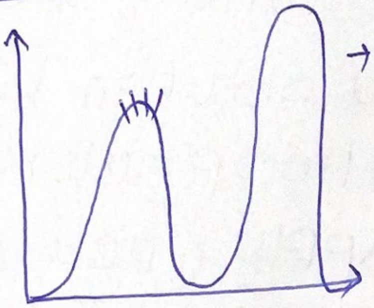
→ if better than current state then it is new state.

Example



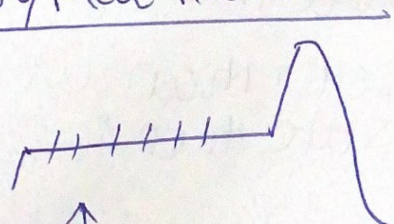

Problems in Hill climbing

a) Local Maximum



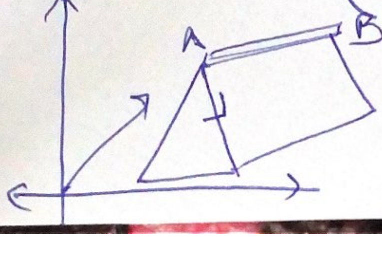
→ Local or non better value

b) Plateau/Flat Maximum



→ similar value

c) Ridge




→ one direction

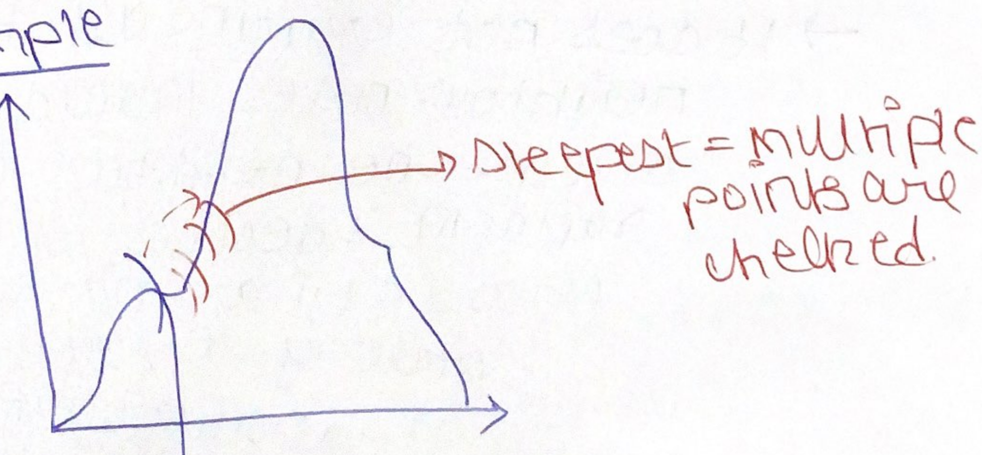
# Steepest Ascent Hill climbing

→ optimal solution
→ Time consuming.
→ In steepest Ascent Hill climbing multiple points are checked
→ Selects the best among the children states that are better than the current state.

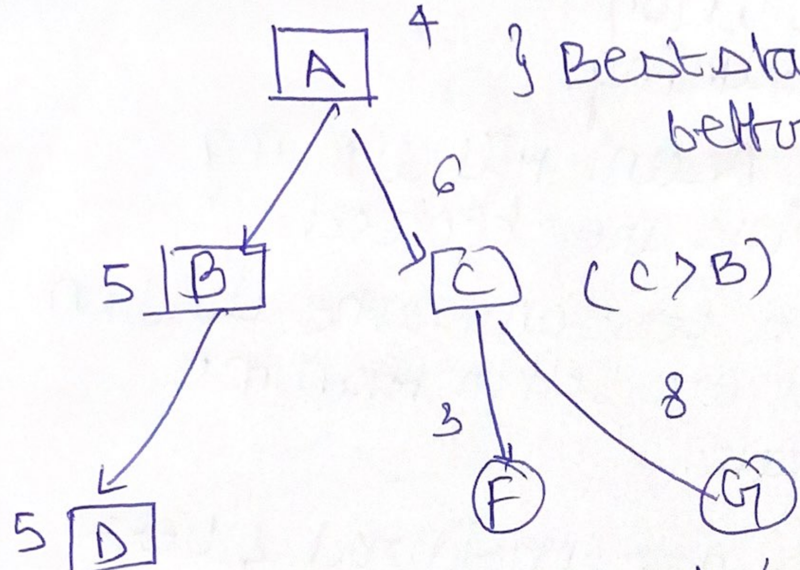
↳ All moves are considered & best one is selected as next state

↳ examines all neighbouring nodes and selects nodes closest as the next node.

Example



## Example




} Best state is node with better heuristic value (>=)

(C > B)

(G > F)

Simple Hill climbing

→ Steepest Ascent

→ slower than Simple Hill

## Stochastic Hill climbing

→ It does not examine all the neighbour nodes. Instead it selects one neighbour at random & decides whether to choose it as the current state or examine another state.

## Simullated Annealling

a) Simullated Annealling allowed downward.

b) whereas, downwards are not allowed in the simple hill climbing.

c) Easy to code for complex problems also

d) Gives good solution to a problem

e) statiscally guarantees finding

f) Slow Process

g) can't tell whether an optimal solution is found (some other method is required.

h) Annealling schedule is maintained. whereas, it is not done in the Hill climbing.

i) Moves to worst state's may be expected

j) Best state, found so far is maintained.

## Local Beam Search

→ Take care of space complexity (constant)

→ Beam width is given (β)

→ Heuristic search Algorithm

→ optimized version of best first search

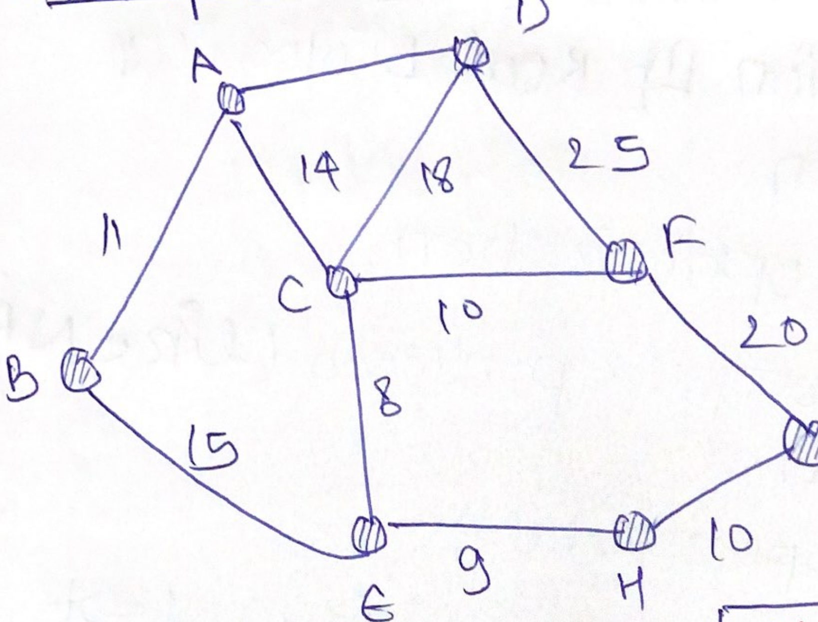
→ Greedy Algorithm use.

→ explores a graph by expanding the most promising node in a limited set.

→ β is predetermined no of best partial solution are kept as candidates

→ class of problems includes in Machine translation, job scheduling, vehicle Routing

## Example

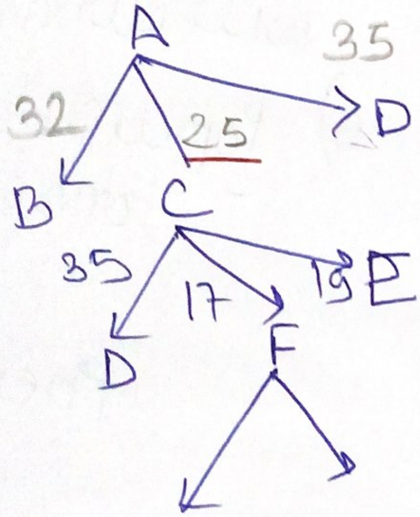



Sorting

~~A~~ ~~C~~ B D F E

## Heuristic values

| | |
|---|---|
| A → G | 40 |
| B → G | 32 |
| C → G | 25 |
| D → G | 35 |
| E → G | 19 |
| F → G | 17 |
| H → G | 10 |
| G → G | 0 |




$\beta = 2$ ( keep any best two & remove others).

$O(b^d)$.

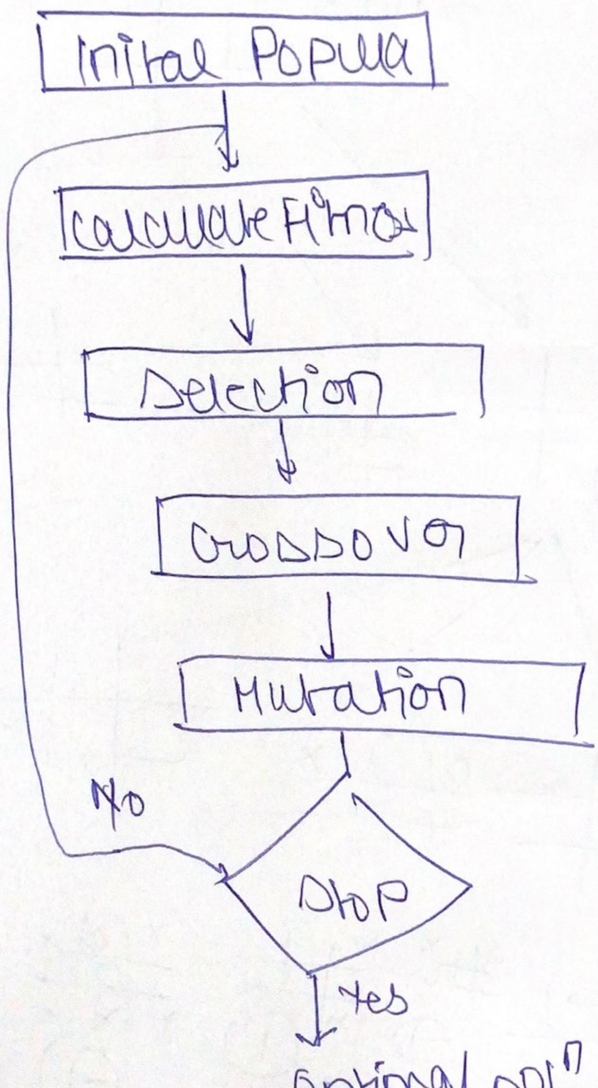
# Genetic Algorithm

1) Abstraction of Real Biological Evolution
2) Focus on optimization
3) Solves complex problems (like NP hard)
4) Search space is large
5) From a group of individuals best will survive.
6) Population of possible solution for a given problem.

$$\text{Phenotype} \underset{\text{decode}}{\overset{\text{encode}}{\rightleftarrows}} \text{Genotype}$$

## Genetic Operators

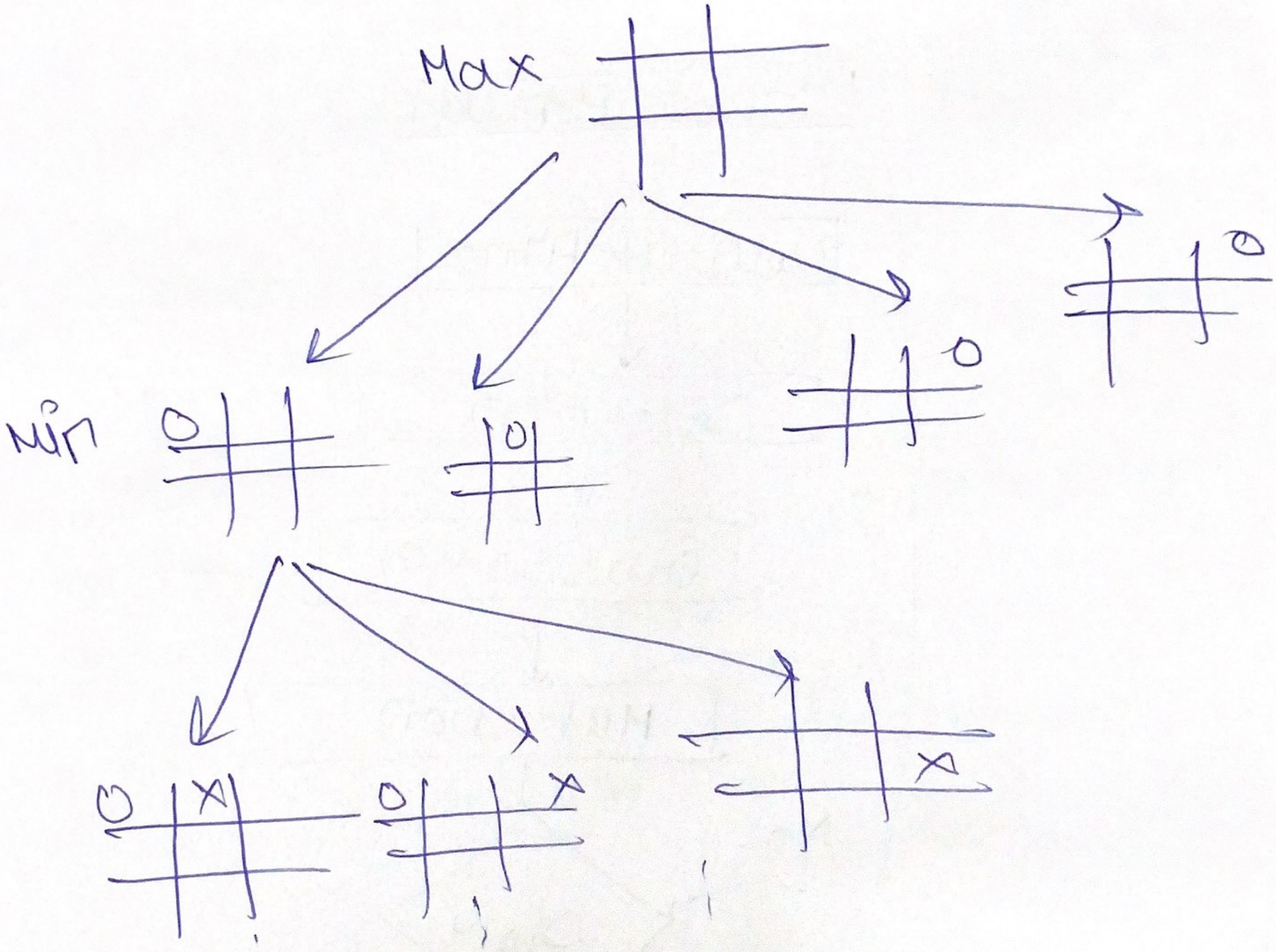
1) Selection
2) Mutation
3) Crossover.



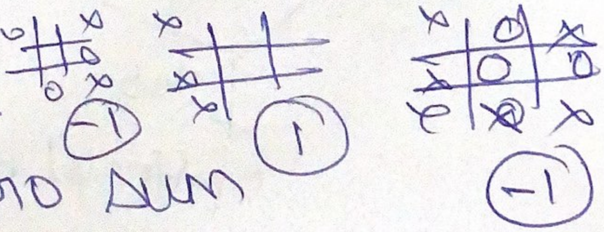
# Game Playing Algorithm

a) Minimamx Algorithm

b) Alphabeta (α-β) Pruning.




Utility

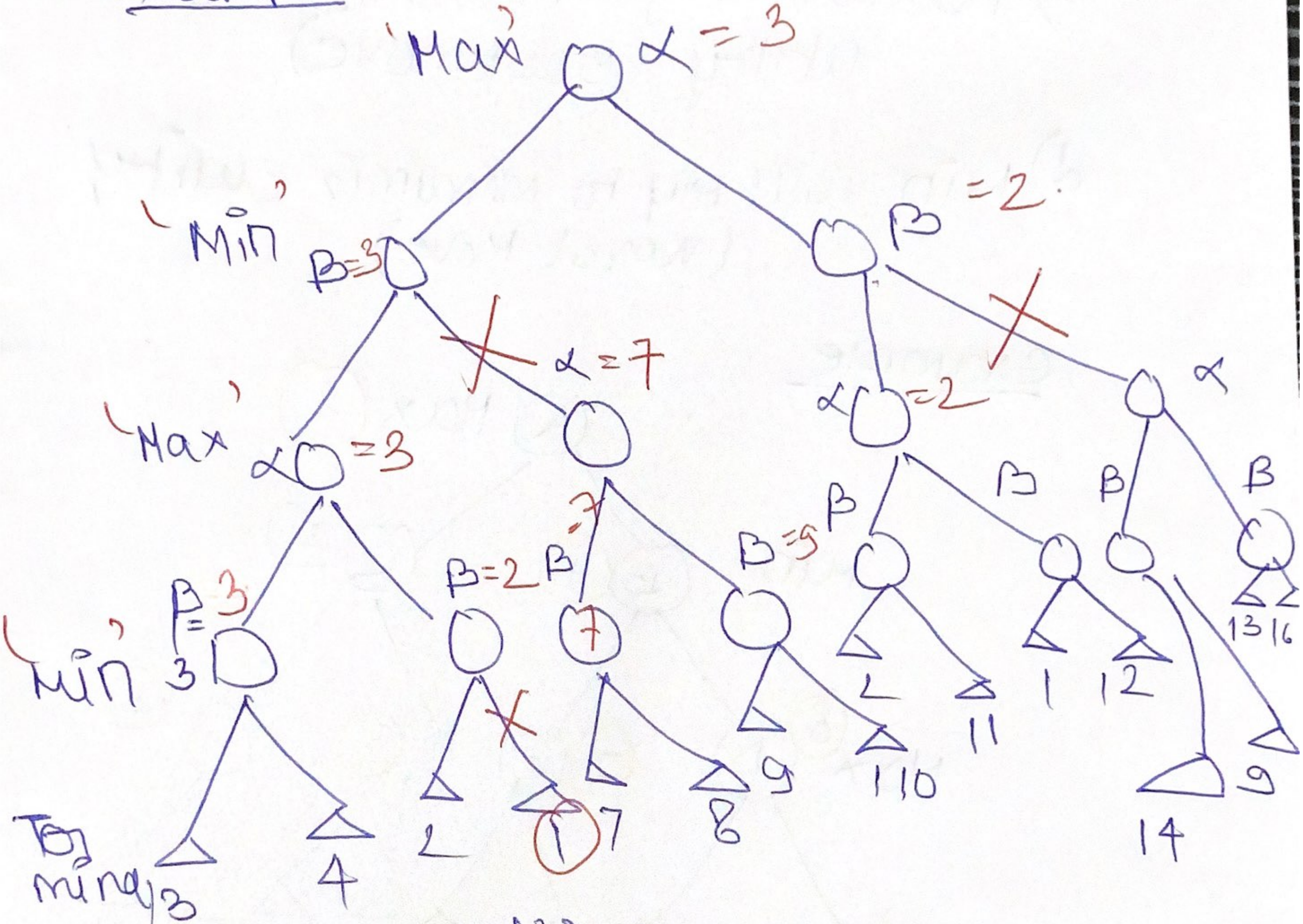



→ Zero sum

# Alpha Beta Pruning

→ cut off search by exploring less no of nodes.

## Example



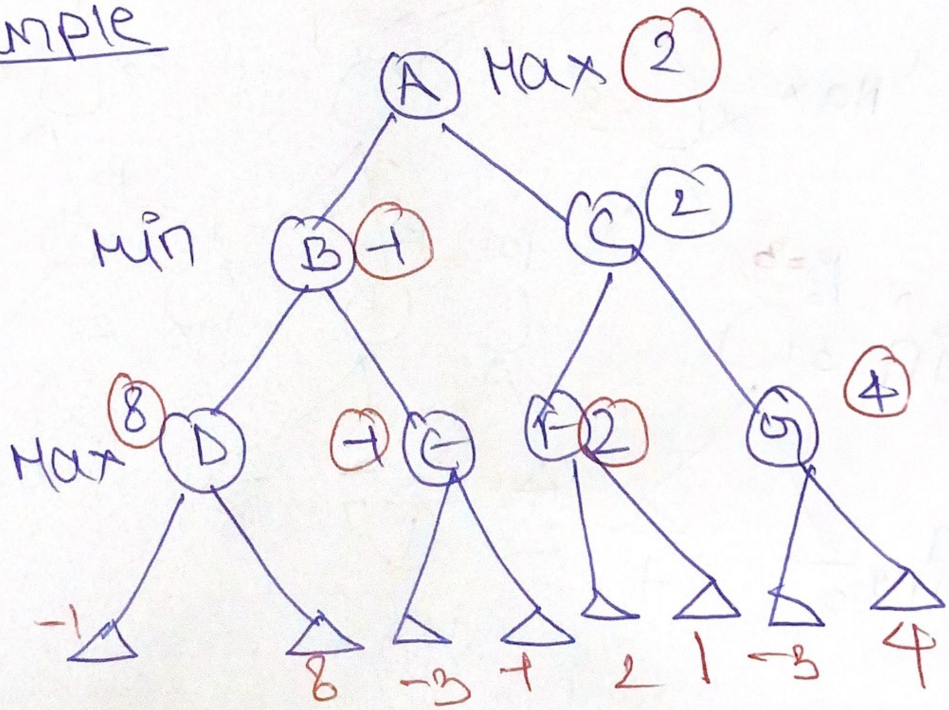

$O(b^{d/2})$

# Min-Max Algorithm

a) Back tracking Algorithm
b) Best move strategy used
c) Max will try to maximize its utility (Best move)

d) Min will try to minimize utility (worst move)

Example




$O(b^{d})$ → ply

↓

branching factor