Artificial Intelligence (AI)

(06/12)

How to make machine behave like a human.

OR

It is the study of how to make machine behave intelligently.

AI can be divided into 4 categories :

- thinking like human
- thinking rationaly
- acting humanly
- acting rationaly

① Thinking humanly

→ the exciting new effort to make computers think; machine with mind in full and literal sense.

given by 1985.

② Thinking Rationaly

→ The study of the computation that make it possible to perceive, reason and to act.

③ Acting Humanly

→ The art of creating machine that perform function that require intelligence that is performed by people.

④ Acting rationaly

→ Computational intelligence is a study of a design of a intelligent agent.

Acting Humanly (m Turing Test Approach)

- Proposed by Alan Turing (1950)

→ A computer passes the test if a human interogate after pacing some written question cannot detect whether the written responses come from a person or a computer.

→ To pass the test, the computer would head to possess the following capabilities → NLP to enable it to communicate successfully in English. knowledge representation to store what it knows or hears.

→ Automated Reasoning to use the stored info, to answer question and to draw new conclusions.

→ M/C learning to adapt to new circumstances and to detect the patterns.

* Total Turing Test: This includes a video signal so that the interrogator can test the subjects perpetual abilities and as well as the opportunity to pass physical objects through the hatch to pass the total turing test, the computer will need computer vision to percive objects and robotics to manipulate objects.

09/12/24

Thinking Humanly

It means trying to understand and model, how a human mind works.

- There are atleast two possible routes that human use to find the answer to a question.

① We reason about it to find the answer. This is called introspection.
② We conduct experiments to find the answer.

- The field of cognitive science focuses how people think.

Thinking Rationally

Modelling thinking as a logical process where conclusions are drawn, based on some kind of symbolic logic.

Acting Rationally

Performing actions that increase the value of the state of the agent, or the environment in which the agent is acting.

- It means acting to achieve one's goals given one's understanding about the world (environment).

Agent : An agent is a system that perceives an environment and acts within that environment.

Intelligent agent : Agent which provides correct output based on the environment.

Types of AI agents

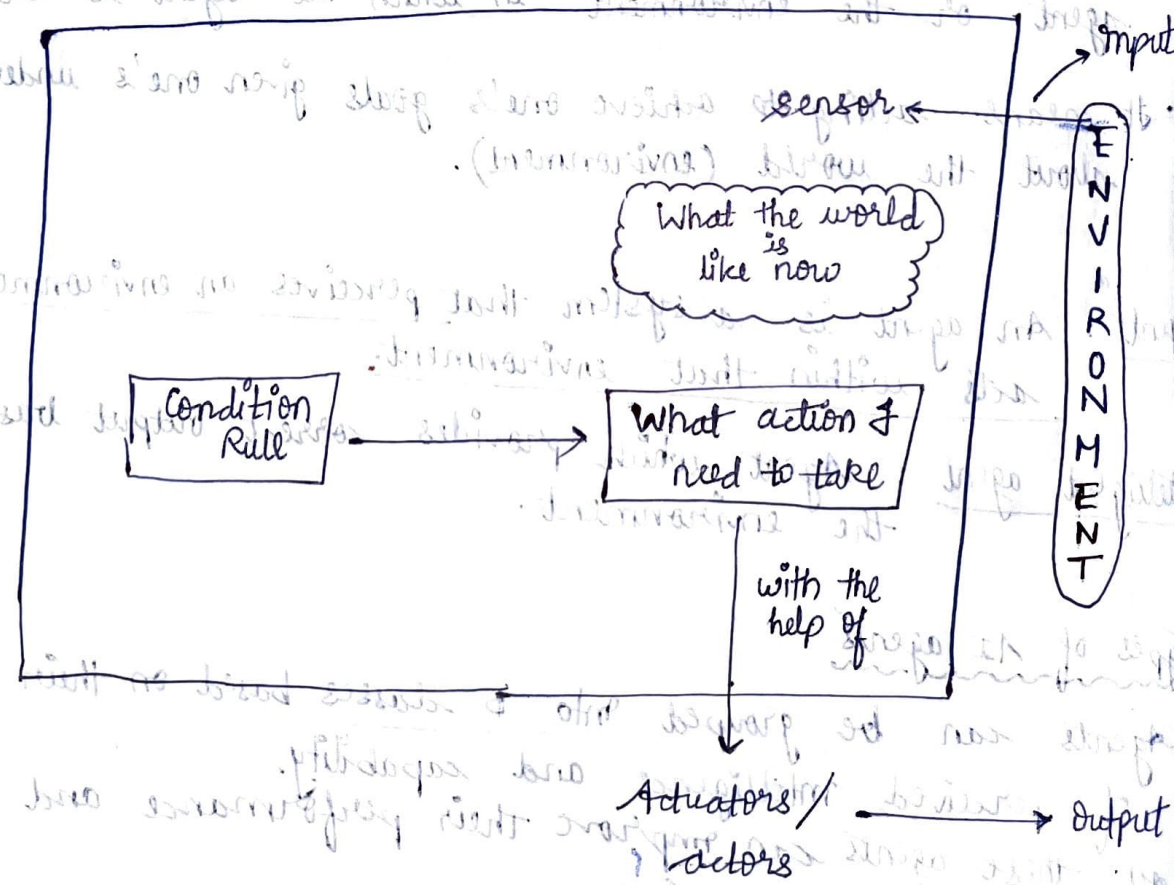
Agents can be grouped into 5 classes based on their degree of perceived intelligence and capability.
All these agents can improve their performance and generate better action over time.

① Simple reflex agent

These are the simplest agents.
These agents take decision on the basis of the current percepts (inputs), ignoring the history of the percepts.
These agents only succeed in a fully observable environment.
Work on condition action rules like "if condition, then action" which maps the current state to action.
Such as a room cleaner agent.
· It works only if there is dirt in the room

Limitations

(i) They have very limited intelligence.
(ii) They do not have knowledge of non-perceptual parts of the current state.
(iii) Not adaptive to the changes in the environment.



② Model Based Reflex Agent

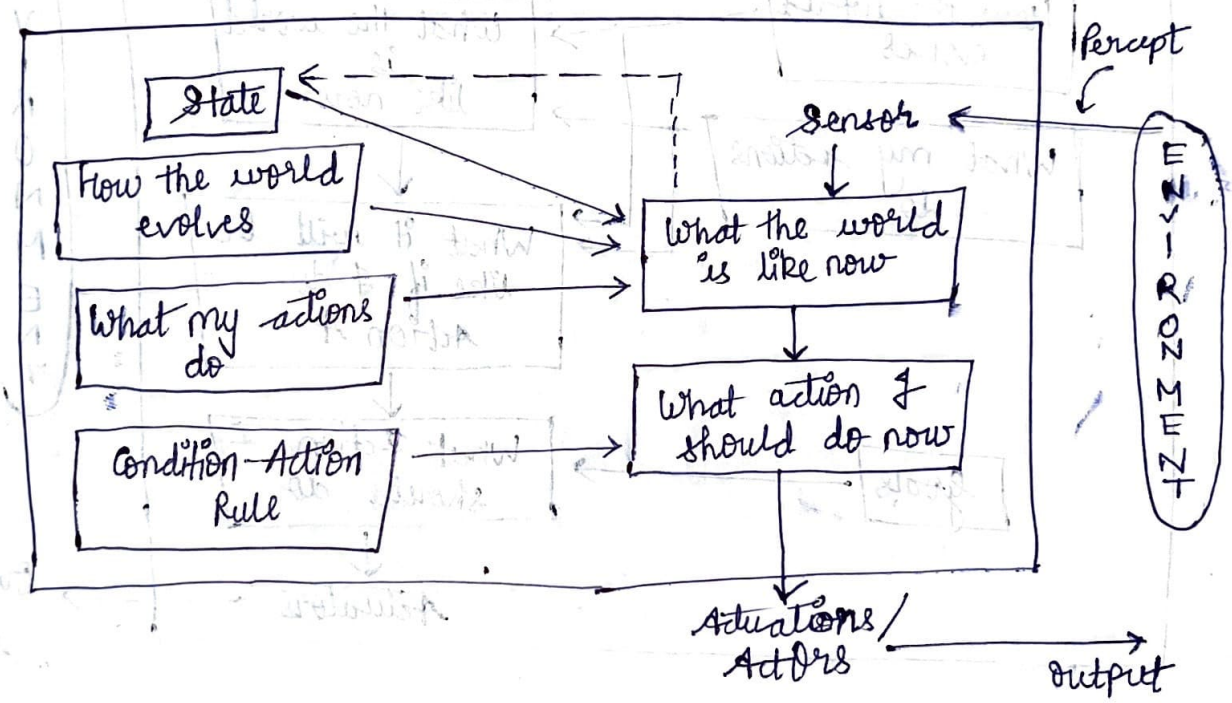
This model based reflex agent that can work in a partially observable environment and track the situation.

It has two important factors:

(i) Model
It is the knowledge about how things happen in the world

[ii] Internal state
↳ It is the representation of the current state based on the percept history.

These agents have the model which is knowledge of the world and based on the model they perform actions. Updating the agent's state requires information about how the world evolves and how the agents action affects the world



③ Goal Based Agents

The knowledge of the current state environment is not always sufficient to decide for an agent to what to do.

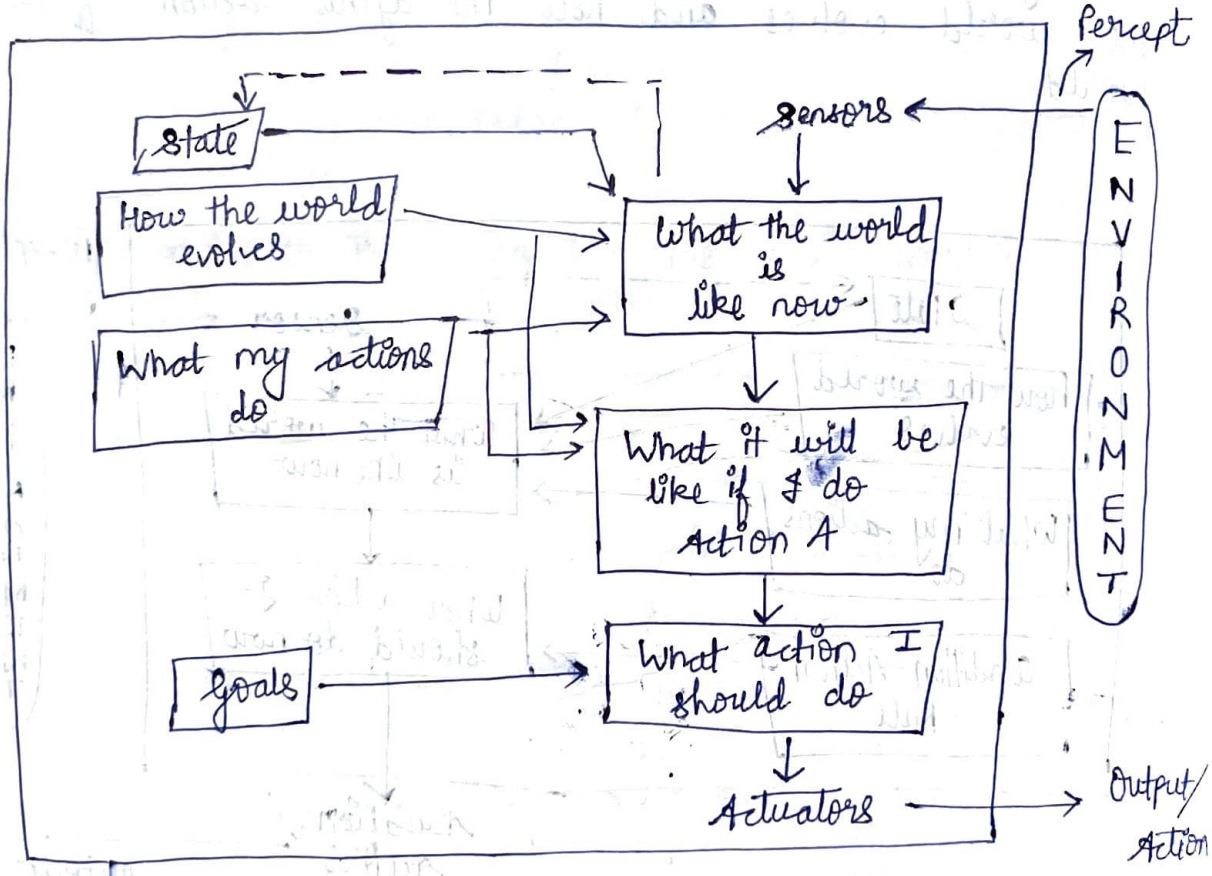
The agents needs to know its goal which describes desirable situations.

Hello!

Goal based agents expand the capabilities of the model based agents by having the goal information. They choose an action so that they can achieve that goal.

These agents may have to consider a long sequence of actions before deciding whether the goal is achieved or not.

Such considerations of different scenarios are called searching and planning which makes an agent proactive

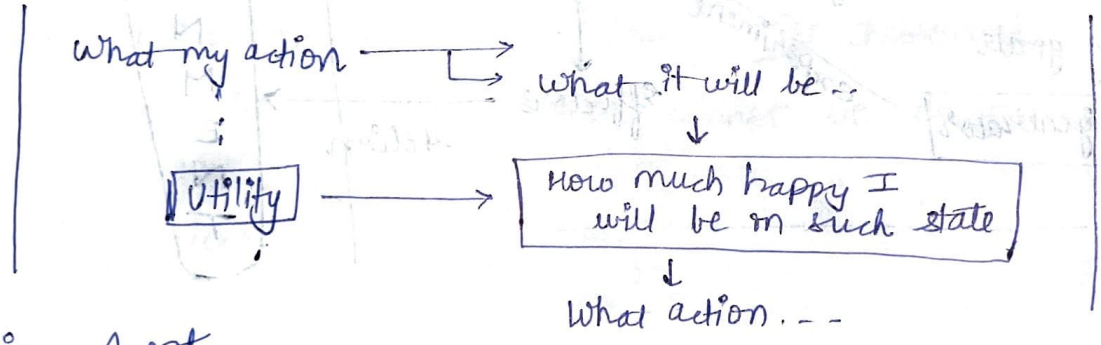


④ Utility Based Agent

→ These agents are similar to goal based agent but provide an extra component of utility measurement which makes them different by providing a measure of success at a given state.

→ At whichever path max^m utility (happiness) the agent gets, it chooses that path.

- Utility based agent act based not only on goals but also the best way to achieve the goal. It is useful when there ~~is~~ are multiple possible alternatives and the agent has to choose in order to perform the best action.
- The utility function maps each state to a real number to check how efficiently each action achieves the goal



• Learning Agent

A learning agent in AI is a type of agent that learns from past experiences. It has learning capabilities. These agents start to act with basic knowledge and then able to act and adapt automatically through learning.

→ A learning agent has mainly four conceptual components, which are:

① Learning element

It is responsible for making improvements by learning from environment.

② Critic

Learning element takes feedback from critic which describes how well the agent is doing wrt a fixed performance standard.

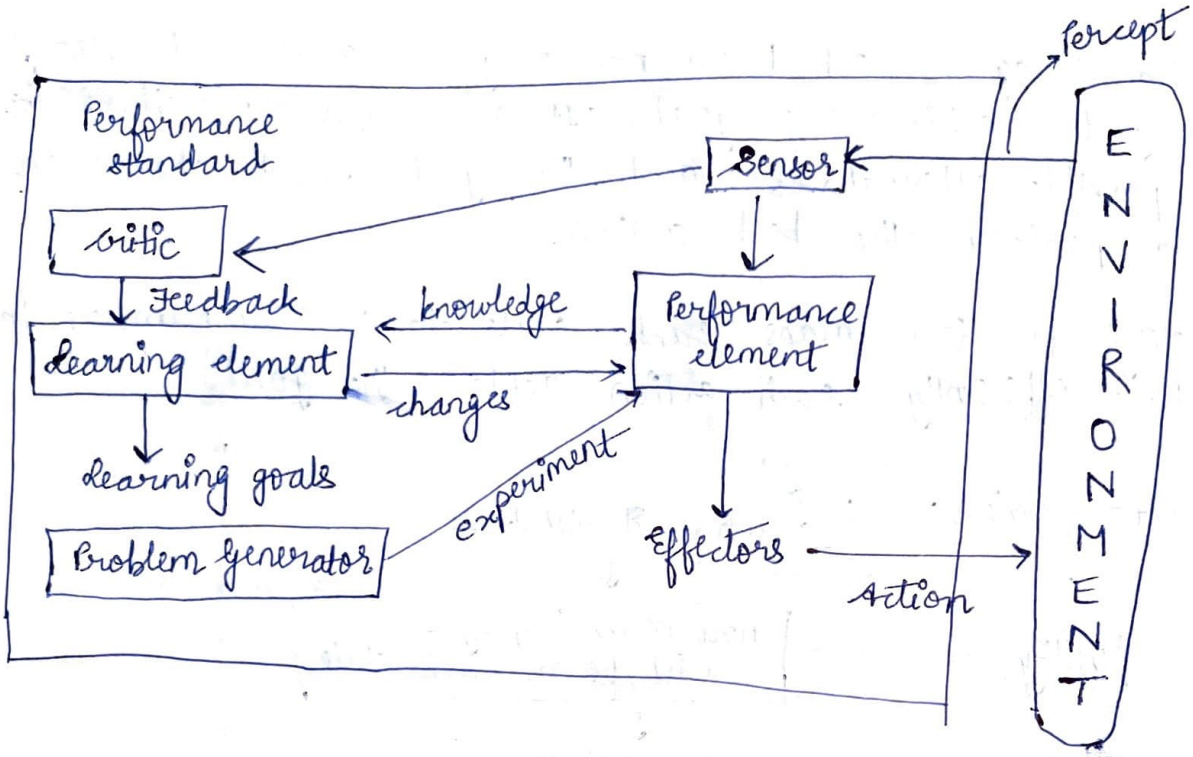
③ Performance element

→ It is responsible for selecting external actions.

④ Problem generator

→ This component is responsible for suggesting actions that will lead to new and informative experiences.

Hence, the learning agent are able to learn, analyze performan and look for new ways to improve the performance.



• Environment

→ An environment is everything which surrounds the agent but a part of agent itself.

→ An agent takes the input from the environment through se and perform the action in the environment through actuator

There are several types of environments:

① Fully observable env v/s partially observable envi.
② Deterministic v/s stochastic env.
③ Static v/s dynamic env.
④ Discrete env v/s continuous env.
⑤ Episodic v/s sequential env.
⑥ Single agent v/s multi-agent env.

① Fully observable v/s partially observable env

- When an agent's sensor is capable to ~~sensor~~ sense, or access the complete state of an environment at each point of time, it is said to be fully observable env, else partially observable

→ Maintaining a fully observable env. is easy as there is no need to keep track of the history of the surrounding

Eg: chessBoard (Fully obs.)
Board and opponents' moves are observable

- In case of self-driving cars, the environment is partially observable, because we don't know whats around the corner.

② Deterministic v/s stochastic env

- If an agent's current state and selected action can completely determine the next state of the environment, then such an env is called a deterministic env. Eg: chessboard
- stochastic env is random and cannot be determined completely by an agent.

Eg: Trading prices

③ Static v/s dynamic env

- An environment that keeps constantly changing itself when the ~~action~~ agent is deliberately some action is called dynamic

Eg: Self-driving cars

- An idle env with no change in its state, is static env.

Eg: ~~Sudoku~~ Sudoku

④ Discrete v/s continuous env

- If in an environment, there are finite number of percepts and actions that can be performed within it, then such env. is called discrete env.

Eg: chess ; there are fixed number of moves user can input.

- The env. in which the actions are performed can not be numbered, ~~it is~~ said to be continuous.

Eg: Self-driving car, how many times brakes are pressed, accelerator can not be counted / are not a finite number.

⑤ Episodic v/s sequential env.

- In an episodic env, each of agent action is divided into atomic incidents or episodes.

→ There is no dependencies between the current and previous incidents.

→ In each incidents, an agent recieves input from the env. and then performs the corresponding action.

Eg: Pick and place robot which is used to detect defective parts from the conveyor belts.

- Here, everytime the robot will make the decision of the current state of the env.
- In a sequential environment, the previous decisions can affect all the future decisions. The next ~~agent~~ action of the agent depends on what action he has taken previously & what action he is supposed to take in the future.

Eg: Chess is an example of sequential environment.
The players must consider the history of the game, the current position of the pieces, the previous moves and potential future moves.
All these influence the best course of action.

⑥ Single Agent v/s multi Agent env.

- An env consisting of only one agent is single-agent env.

→ In single-agent env., the agent doesn't need to consider the actions of decisions of other entities.

Eg: In solitaire, only one user can play/interact with the game

- An env. consisting more than one agent is known as multi-agent

Eg: Multiplayer games such as Ludo, soccer.

~~In mul~~

The outcomes of the game depends upon coordinated actions and strategies of all the ~~actio~~ agents playing the game

Q]: Find the type of env, the actuators, the sensors ~~and agents~~ in the following agents.

① self-driving cars

Env – Partially observable, ~~deterministic~~ stochastic, dynamic, continuous, ~~sequential~~ multi-agent

Actuators – brakes, signals, horn

Sensors – sonar, camera, gps, accelerometer, speedometer

② Part-picking robot / pick & place robot

Env – Episodic, deterministic, dynamic, fully observable,

Actuators – hands, legs, ~~car~~

Sensors – cameras, radar

→ ~~PEAS~~

P → Performance Measure
E → Environment
A → Actuators
S → sensors

- Peas is a framework used to specify the structure of an intelligent agent in AI
- It breaks down the agent's interaction with the env. into four key components:
  ① Performance Measure – defines the success of the agent's action.
  ② Environment – the ~~content~~ in which the agent operates
  ③ Actuators – the mechanism through which the agent interacts with the env.
  ④ Sensors – these are the tools the agent uses to perceive its environment.

Q]: Task Environment

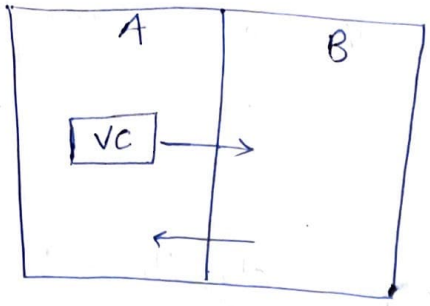
	Fully / Partially observable	single Agent / multiagent	Deterministic / stochastic	Episodic / Sequential	Static / Dynamic
① Crossword Puzzle					
② Poker					
③ Medical diagnosis					
④ Image Analysis					
⑤ Taxi driving					
⑥ Refinery Controller					

Q]: Agent Type

	Performance Measure	Environment	Actuators	Sensors
① Medical diagnosis system	Reports, Recovery (accurate, fast)			
② Satellite Image Analysis System				
③ Part Picking Robot				
④ Refinery Controller				
⑤ Interactive English Tutor	Grade	Platform or app	speaker, screen	microphone, camera

* Unobservable Environment : The agent has no sensors; so it cannot get any input from the environment; hence unobservable environment.

- Vaccum cleaner Problem



- 2 Moves
  → Right → Left
- Actions
  → Move to right
  → Move to left
  → Suck the dirt
  → Do Nothing

Input	Action
[A, clean]	Move right
[A, dirty]	Suck the dirt
[B, clean]	Move left
[B, dirty]	Suck the dirt
[A, clean] [A, dirty]	Suck
[B, dirt] [B, clean]	Move to left

→ Rational agents (or problem solving agents) mostly use the search strategies or algorithms to solve the specific problem to provide the best result.

Search

Searching is a step-by-step procedure to solve a search problem in a given search space.

- Search space represents a set of possible solutions which a system may have.

  Start State – this is the initial state from where agent begins it search

  Actions – It gives the description of all the available actions to the agent.

  Path Cost – It is a function which assigns a numeric cost to each path

Solution – It is an action sequence which leads from start node to goal node.

Optimal sol^n – It is the solution which has the lowest cost among all the solutions.

→ Properties of a searching algorithm

Following are the 4 essential properties of search algorithms to compare the efficiency of these algos.

1. Completeness
2. Optimality
3. Time Complexity
4. Space Complexity

uniform-cost-search (problem) returns a solution, or failure
    node and problem initialization
    path-cost
    frontier ← a priority queue ordered by path cost
    explored ← an empty set

loop do
    if empty? (frontier) then return failure
    node ← pop (frontier)

BFS Algorithm

```
private static boolean bfs (adj, src, dest, v, pred[], dist[]) {
    linked list <Int> queue = new LL <Int> ();
    boolean visited = new boolean [v];
    for (i=0; i<v; i++)
        visited [i] = false;
        dist[i] = Int. max;
        pred[i] = -1;
    }
    visited [i] = true;
    dist[src] = 0;
    queue add (src);
    while (!queue. isEmpty ()) {
        Int cur = queue. remove()
        for (Int i=0; i<adj.get (u) . size (); i++) {
            Int neighbour = adj. get (u). get (i);
            if visited (neighbour) == false)
        }
        visited I. height = = true;
        dist [neighbour] = dist [cur]+1;
        pred [neighbour] = cur;
        queue . add (neighbour);
        if (neighbour == list)
            return true;
        }
    }
}
```

Q):

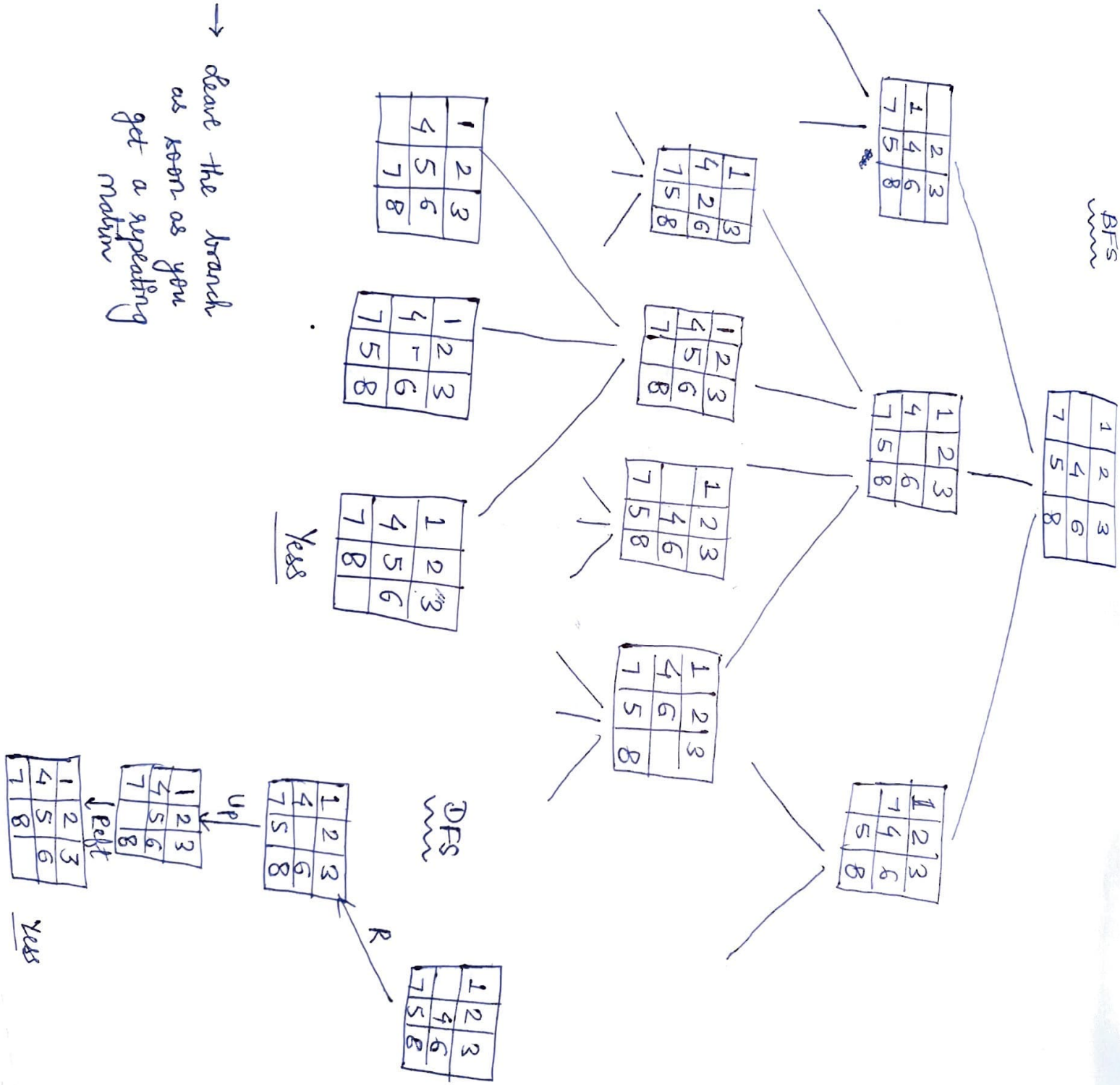
Initial

1	2	3
	4	6
7	5	8

→

Goal

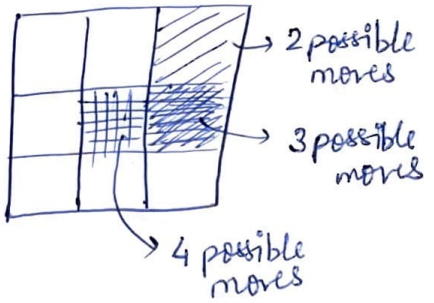
1	2	3
4	5	6
7	8	



* BFS guarantees optimality.

→ DFS will be first exploring the entirety of one branch, hence making the solution lengthier.

→ 8-PUZZLE PROBLEM



No. of total moves possible for an unknown empty block

$= (4 \times 2) + (4 \times 3) + (1 \times 4)$

$= 24$

Avg. $= \frac{24}{9} = \boxed{2.67 = b}$ (Branch factor)

Time complexity will be $(2.67)^d$ where d → No. of levels.

• Uniform Cost search

→ It is a searching technique used for traversing a weighted graph or tree.

→ This algorithm is used when a different cost is available for each edge.

→ The primary goal of the uniform cost search is to find a path to the goal node which has the lowest cumulative cost.

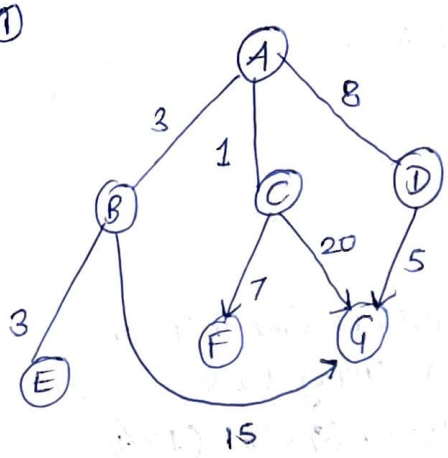
→ Uniform cost search expands nodes according to their path cost from root node

→ It is implemented using priority queue

→ Uniform cost search

→ It gives the maxm priority the lowest cumulative cost. Uniform cost search is equivalent to bfs if the path cost is same.

Eg: ①



Find the minimum cost path from A to G

min Priority Queue

Ⓐ

C	B	D
1	3	8

Ⓒ

B	D	F	G
3	8	8	~~21~~

Ⓑ

E	D	F	G
6	8	8	18

Ⓔ

D	F	G
8	8	18

Ⓓ

F	G
8	13

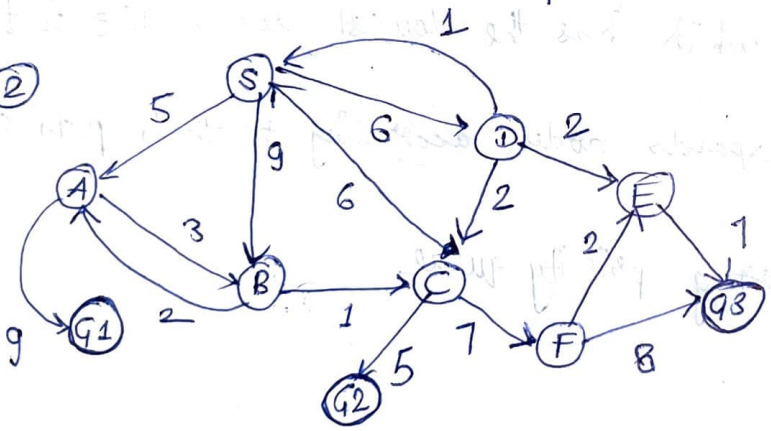
Ⓕ

G
13

Ⓖ

Ⓐ → Ⓓ → Ⓖ

Eg: ②



G2 → 13

Ⓢ

A	D	B
5	6	9

Ⓐ

D	B	G1
6	8	14

Ⓓ

B	E	C	G1
8	8	8	14

Algorithm

uniform_cost_search (problem) returns a solution or failure

node ← problem.Initial_state

path_cost = 0

frontier ← a priority queue ordered by path_cost

explored ← an empty set

loop do

if Empty? (frontier) then return failure

node ← pop (frontier)

if problem.Goal_Test (node.state) then return solution (node)

add node.state to explored

for each action in problem.Action (node, state) do

child ← child_Node (problem, node, action)

if child.state is not in explored or frontier then

frontier ← Insert (child, frontier)

elseif child.state is in frontier with higher path_cost

then replace that frontier node with child

- The time complexity here becomes $b^{d/2}$ because we are not going to the extreme ends (in best case).
- In worst case, the time complexity is $2b^d$ or b^d.

* Always gives the optimal result.

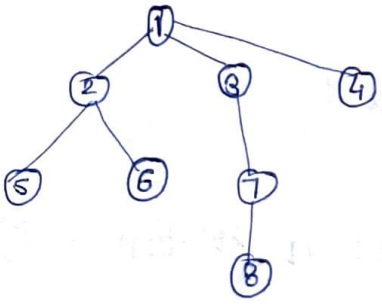
- If the graph is non-directed, the time complexity will always gonna be $b^{d/2}$.

But, if its a directed graph, there are chances that we might not find a common point in the middle or somewhere.
So, the time complexity can escalate to b^d.

- Depth-Limited Search (DLS)

* If a solution is optimal, the algorithm is a complete algorithm.
On the other hand, if a algo is complete algorithm, it will atleast give a solution (that need not be optimal)

Iterative



I_1 Iteration Level 0, 1
I_2 Iteration Level 1, 2, 3, 4
I_3 Iteration Level 2,

$= d * b + (d-1) * b^2 + (d-2) * b^3 + \ldots$
$= O(b^d)$

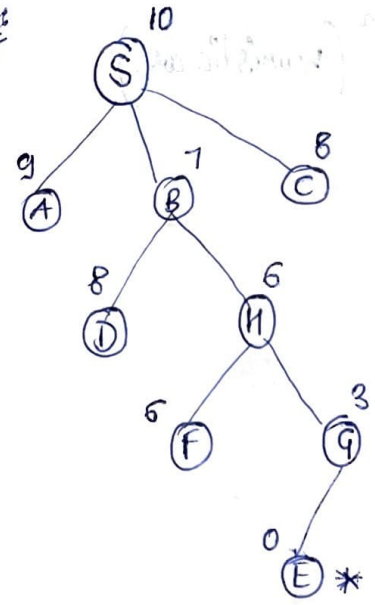
b — branches
d — level/depth

Informed Cost Search

1) Best First Search

* (Heuristic value) → estimates the cost to reach the goal node.

Eg:

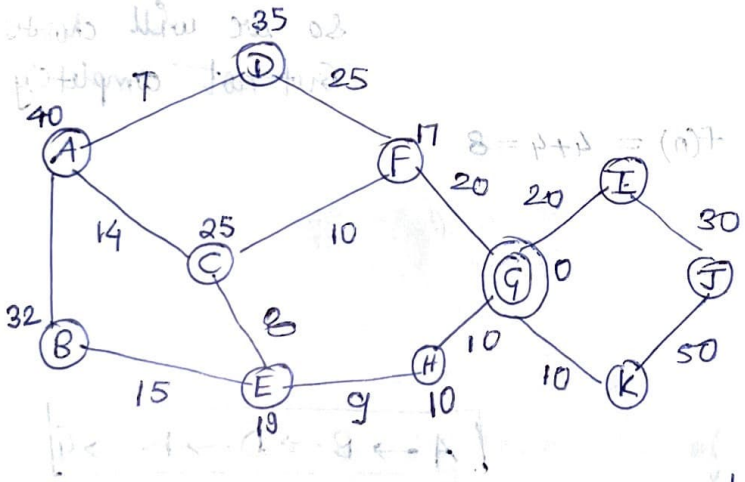


Open list	Heuristic value H(n)	Closed List	Parent
~~S~~	~~10~~	S	—
~~BCA~~	~~7~~, 8, 9	SB	S
~~HCDA~~	~~6~~, 8, 8, 9	SBH	B
~~GFCDA~~	~~3~~, 6, 8, 8, 9	SBHG	H
~~EFCDA~~	~~0~~, 6, 8, 8, 9	SBHGE	G

The path is

$S \rightarrow B \rightarrow H \rightarrow G \rightarrow E$

Eg:



Open List	Heuristic value	Closed List	Parent
~~A~~	~~40~~	A	—
~~C~~, B, D	~~25~~, 32, 35	AC	A
~~F~~, E, B, D	~~17~~, 19, 32, 35	ACF	C
~~G~~EBD	~~0~~, 19, 32, 35	ACFG	F

$A \rightarrow C \rightarrow F \rightarrow G$

$(14 + 10 + 20) = 44$

- But, $A \rightarrow C \rightarrow E \rightarrow F \rightarrow G$ gives the total cost to be $41 < 44$.

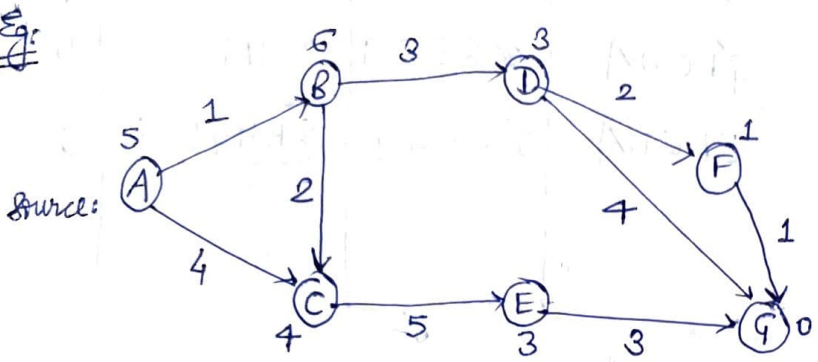
So, best first search search isn't giving us the optimal value

→ A* search

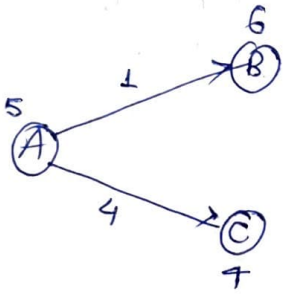
$$f(n) = g(n) + h(n)$$

- $f(n)$: cost function for a particular node
- $g(n)$: Actual cost from source to that node
- $h(n)$: estimated cost from that node to the goal node (heuristic cost)

Eg:



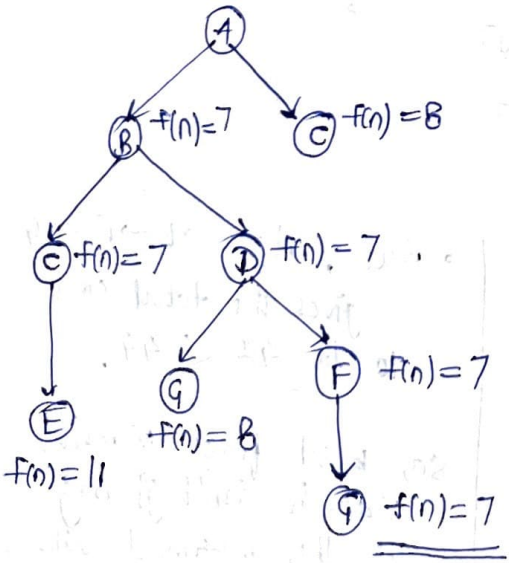
Here,



$f(n) = 1+6 = 7$

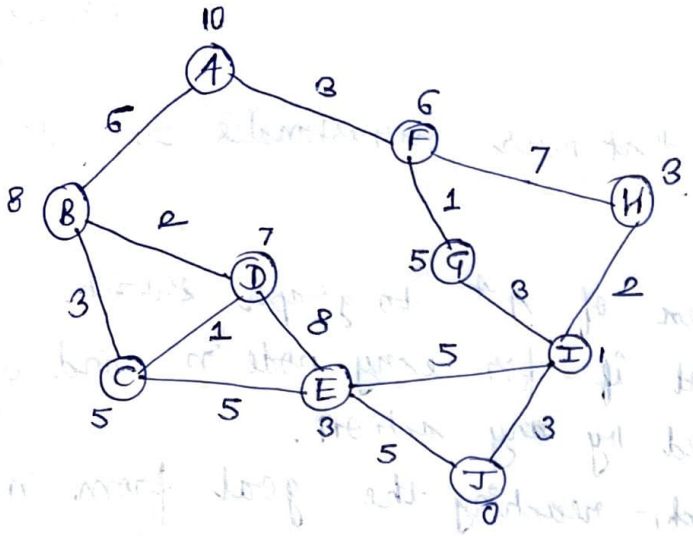
$f(n) = 4+4 = 8$

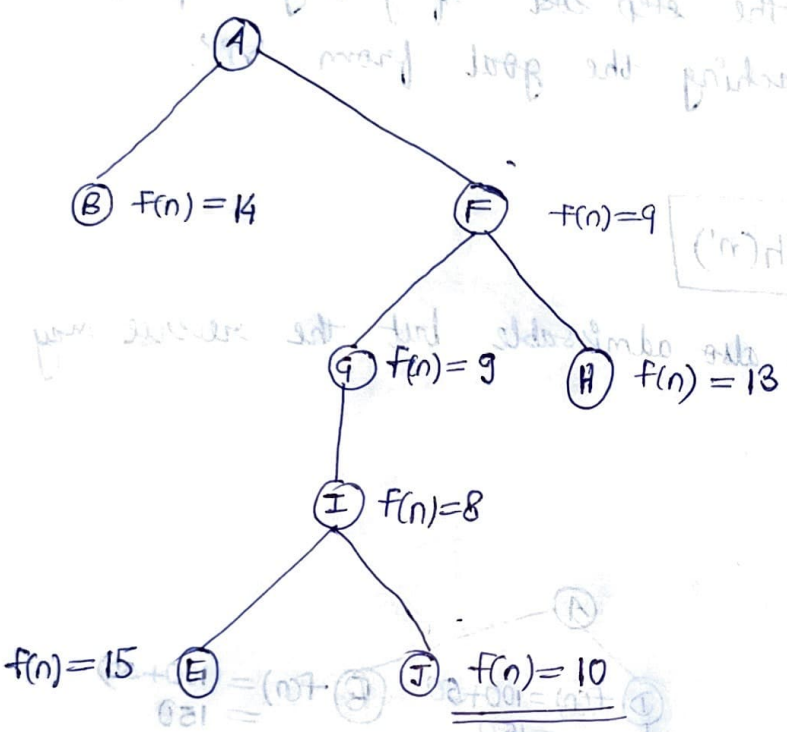
So we will choose B, But not completely



A → B → D → F → G

Eg:





A → F → G → I → J

* Still, we are using the estimation of heuristic value which affects the overall solution.

We use underestimation for our benefit.

overestimation

An admissible - heuristic is one that never overestimates the cost to reach the goal.

Consistency is required for appln of A* to graph search.

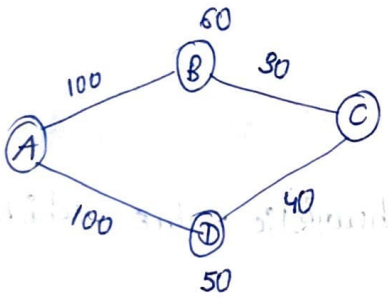
- A heuristic $h(n)$ is consistent if for every node n and every successor n' of n generated by any action.
- The estimated cost of reaching the goal from n is not greater than the step cost of getting n' plus the estimated cost of reaching the goal from n'.

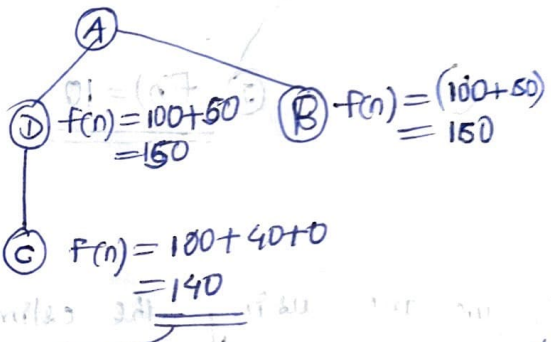
$$h(n) <= c(n, a, n') + h(n')$$

Every consistent heuristic is also admissable but the reverse may or maynot be true.

Optimality of A* search

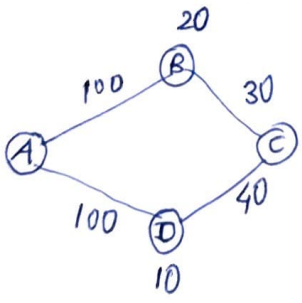
- Overestimation

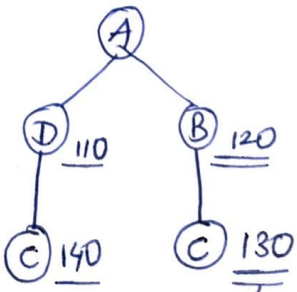




this is optimal according to A*
But A → B → C seems optimal

- Underestimation



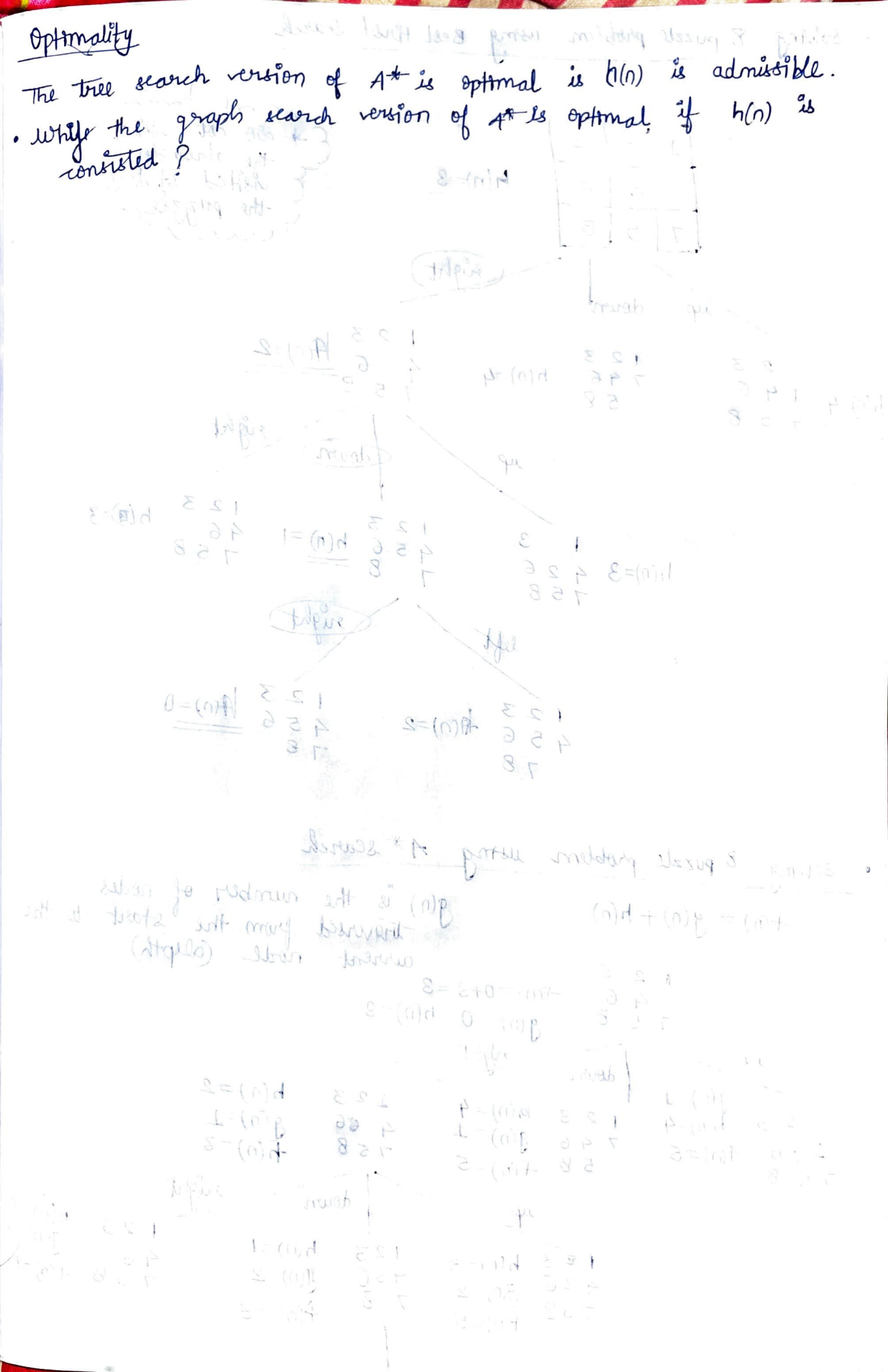


this is optimal according to A*
and its the actual optimal
A → B → C

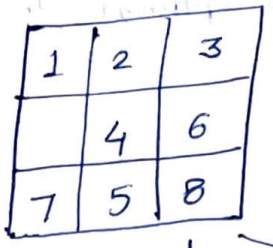
Optimality

The tree search version of A* is optimal is $h(n)$ is admissible.

- While the graph search version of A* is optimal, if $h(n)$ is consisted ?



- solving 8-puzzle problem using Best First search



h(n)=3

* do not visited the already visited state of the puzzle

up | down | right

up:
2 3
1 4 6
7 5 8
h(n)=4

down:
1 2 3
7 4 6
5 8
h(n)=4

right:
1 2 3
4 6
7 5 8
h(n)=2

up:
1 3
4 2 6
7 5 8
h(n)=3

down:
1 2 3
4 5 6
7 8
h(n)=1

right:
1 2 3
4 6
7 5 8
h(n)=3

left:
1 2 3
4 5 6
7 8
h(n)=2

right:
1 2 3
4 5 6
7 8
h(n)=0

- solving 8 puzzle problem using A* search

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

g(n) is the number of nodes traversed from the start to the current node (depth)

1 2 3
4 6
7 5 8
f(n)=0+3=3
g(n)=0
h(n)=3

up:
2 3
1 4 6
7 5 8
g(n)=1
h(n)=4
f(n)=5

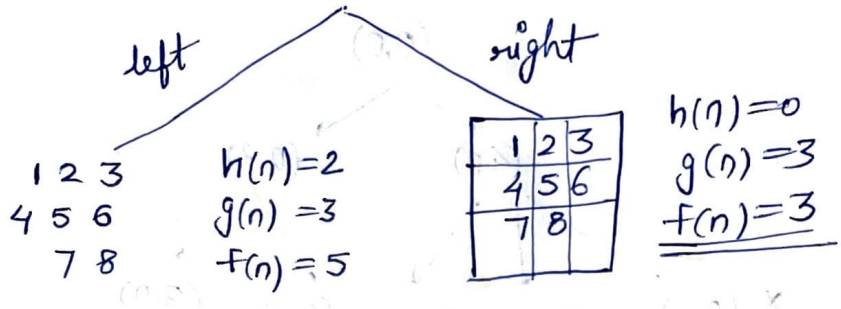
down:
1 2 3
7 4 6
5 8
h(n)=4
g(n)=1
f(n)=5

right:
1 2 3
4 6
7 5 8
h(n)=2
g(n)=1
f(n)=3

up:
1 3
4 2 6
7 5 8
h(n)=3
g(n)=2
f(n)=5

down:
1 2 3
4 5 6
7 8
h(n)=1
g(n)=2
f(n)=3

right:
1 2 3
4 6
7 5 8
h(n)=
g(n)=
f(n)=

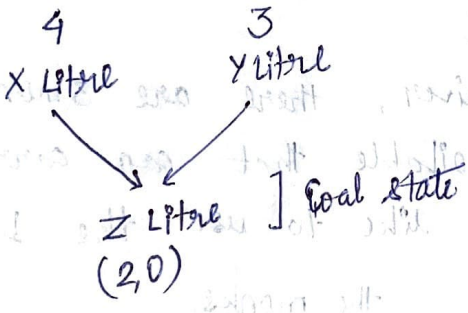


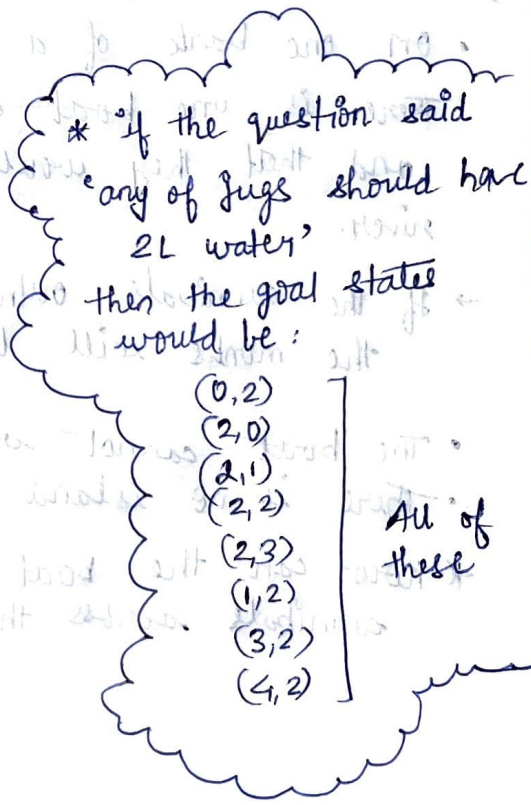
→ Water Jug Problem

- You are given 2 jugs, a 4L and a 3L jug.
- A pump which has unlimited water which you can use to fill the jug and the ground on which water can be poured.
- Neither jug has ~~no~~ any measuring marks.

→ How can you get exactly 2L of water in one jug and no water in other jug.

Let us assume, we need the 2L in the 4L jug.

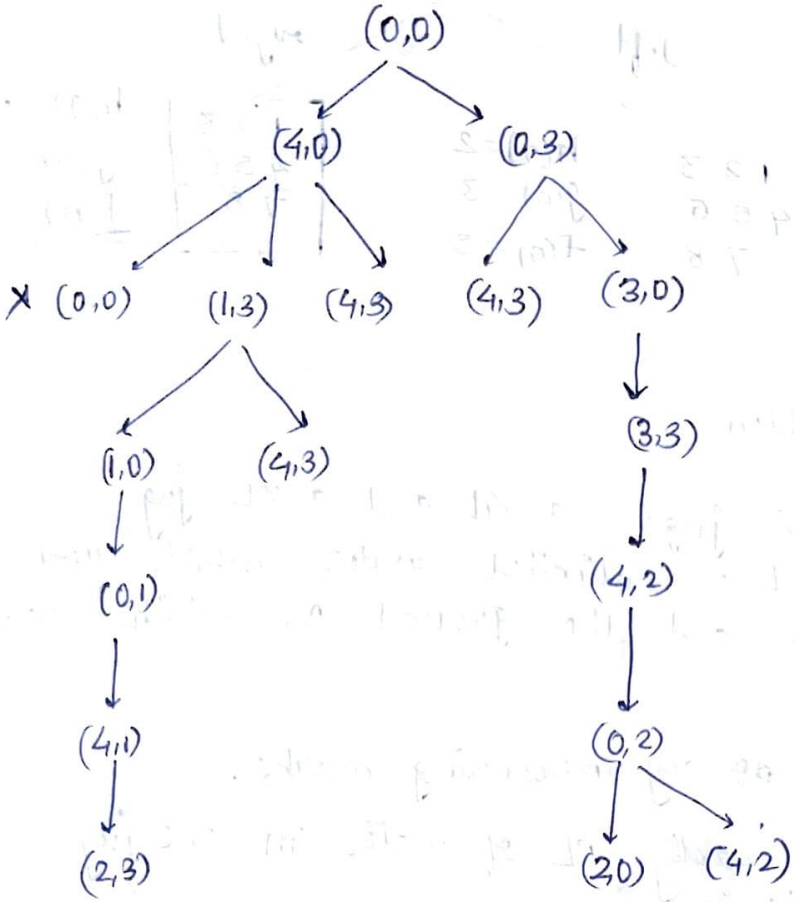




→ possible actions

- Fill 4L jug
- Fill 3L jug
- Empty 4L jug
- Empty 3L jug
- Pour from 4L to 3L jug
- Pour from 3L to 4L jug

Start state
(0,0)



→ ~~Monk~~ - Cannibal Problem

- On one bank of a river, there are 3 monks and 3 cannibals. There is one boat available that can carry almost 2 people and that they would like to use the boat to cross the river.

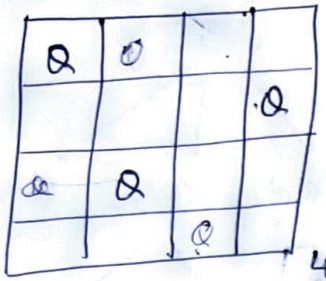
→ If the cannibals outnumber the monks on either of the river banks, the monks will be devoured.

- The boat cannot cross the river by itself.
- There is no island in the middle of the river.

* How can the boat be used to carry all the monks and cannibals across the river safely?

state tree
(0,0)

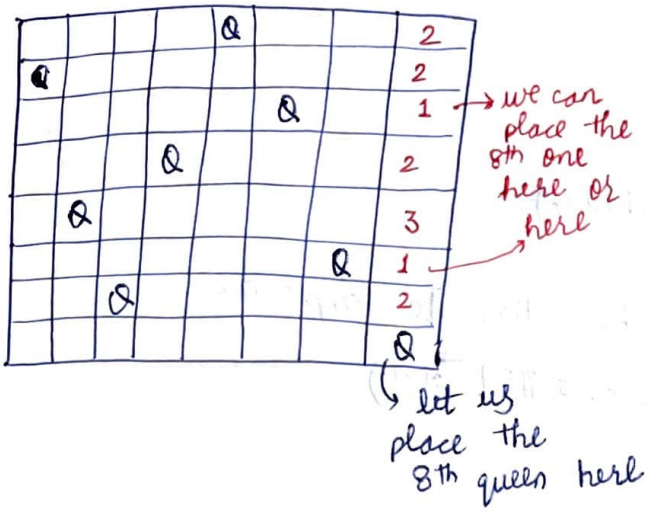
- 4-Queen Problem

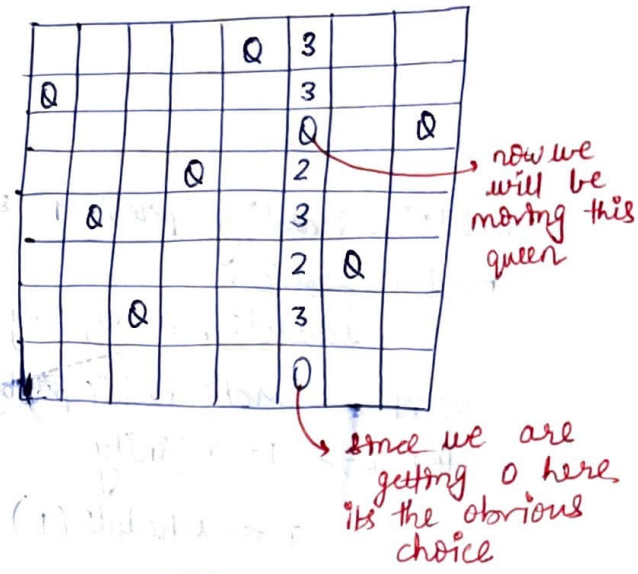


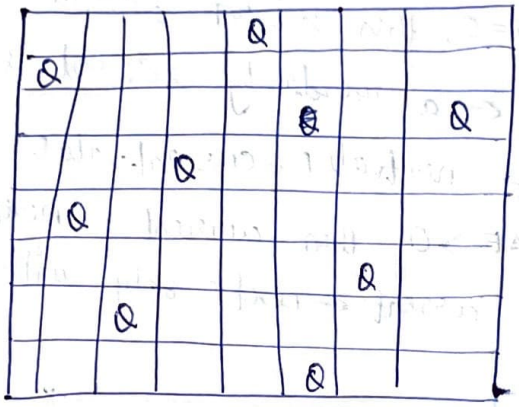
4×4 Board

(27/01)

8-Queen Problem (using Hill climb)







This is the optimized one

Q: Find the challenges of simple hill climbing algorithm and also find its solution

① Local Maxima

The algorithm may get stuck at a local maximum where it cannot find higher value even though there may be a better solution elsewhere.

Solution - Use random restarts to reinitiate the search from different points or use simulated annealing to allow occassional steps in the wrong direction to escape local maxima.

② Plateaus

The algorithm might encounter a plateau, where many neighbouring states have the same value, making it difficult to find a better state.

Simulated-Annealing (problem, schedule)

input: problem
schedule, a mapping from time to temperature

current ← Make-node (problem. Initial-state)
for t=1 to infinity
    T ← schedule (t)
    if T=0, then return current.
    next ← a randomly selected successor of current
    ΔE ← nextvalue − currentvalue
    if ΔE > 0 then current ← next
    else current ← next only with probability $e^{\Delta E/T}$

Solution: Implement stochastic hill climbing or random walks to move across plateaus by allowing random moves.

③ No backtracking: Once a move is made to a neighbouring state, the algorithm doesn't reconsider previous states which could have been better in the long run.

Solution: Introduce backtracking or explore beam search to consider multiples stakes ~~which could have been better in the long run~~. at each step.

Genetic Algorithm

① Selection — Process of selecting two chromosomes for crossover to produce new ~~crossover~~ chromosome

② Crossover

③ Mutation

Selection

- Tournament selection

  $k = 3$ means that we will pick 3 chromosomes from the population randomly and take the one with the most fitness value as the parent chromosome
  → we will do this until we get enough parents.

- Roulette wheel selection
  (Fitness Proportionate selection)

  → Each individual is assigned a probability of being selection based on its fitness.

  → The "roulette wheel" is spun to select individuals, where higher-fitness individuals have a large slice of the wheel.

Crossover

- Single-Point Crossover

  → A single crossover point is randomly selected along the length of the parent chromosomes.

  → The parts of the chromosomes beyond this point are swapped between two parents to create two offsprings.

Eg: P1 : 10 1 | 1 00 110
P2 : 0 11 | 0 11 111
C1 : 10 1 | 0 11 111
C2 : 0 11 | 1 00 110

- Two Point crossover
  - → Two crossover points are randomly selected along the length of the parent chromsomes.
  - → The segments between these points are swapped between parents to create offspring.

Eg:
P1: 10 | 11 | 00
P2: 01 | 00 | 11

C1: 10 00 00
C2: 01 11 11

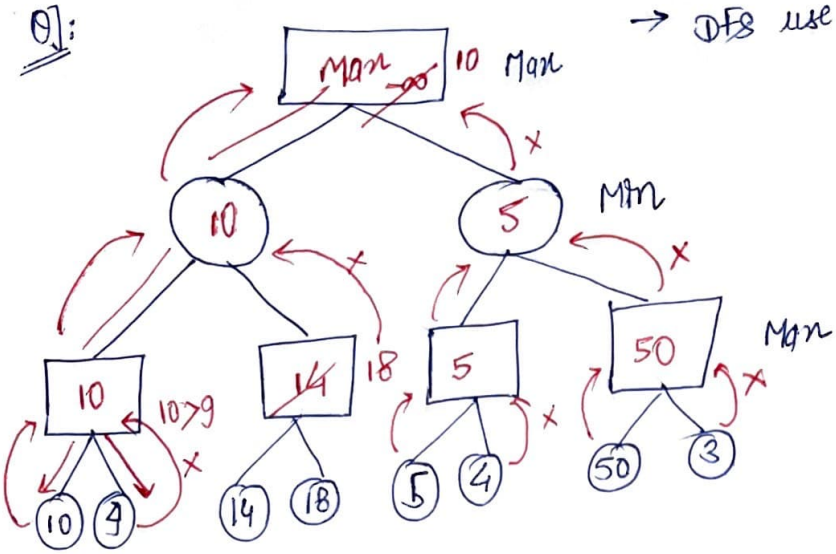
Mutation

07/02

Max-Min Algorithm

→ DFS use hoga

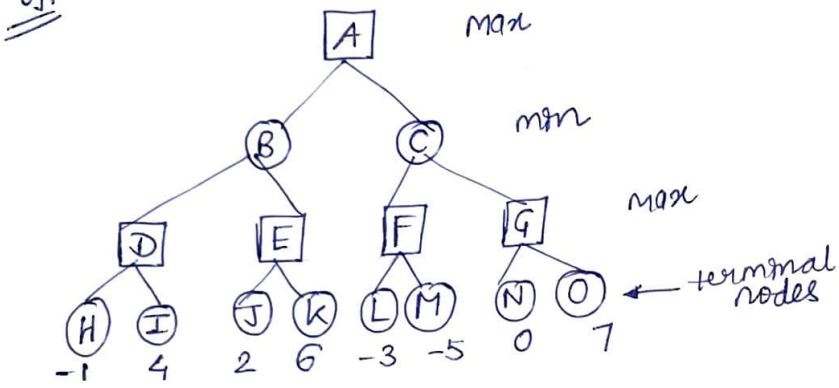
Q]:

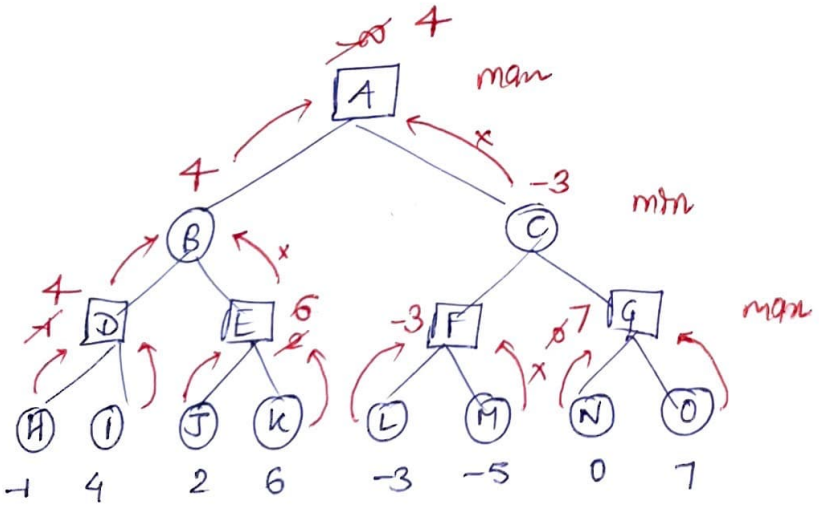


- Max & Min are playing
We will start with max
& min = +∞

Optimal Result = 10

Q]:





Optimal Result = 4

A → B → D → I

α-β Pruning

- It is an optimization technique for the min-max algorithm.
- It reduces the number of nodes evaluated in the game tree by eliminating branches that cannot influence the final decision.

- This is achieved by maintaining two values α and β, which represent the minimum score that maximizing player is assured of and the maximum score that the minimizing player is assured of, respectively.

Steps for α-β pruning

① Initialization

→ Start with α set to $-\infty$ and β set to $+\infty$.

α ↓ max player

β ↓ min player

② Max node evaluation

- For each child of max node,
  Evaluate the child node using min-max algorithm with α-β pruning.

- Update max (α, child value) ⇒ α

→ If $\alpha \geq \beta$; prune the remaining children

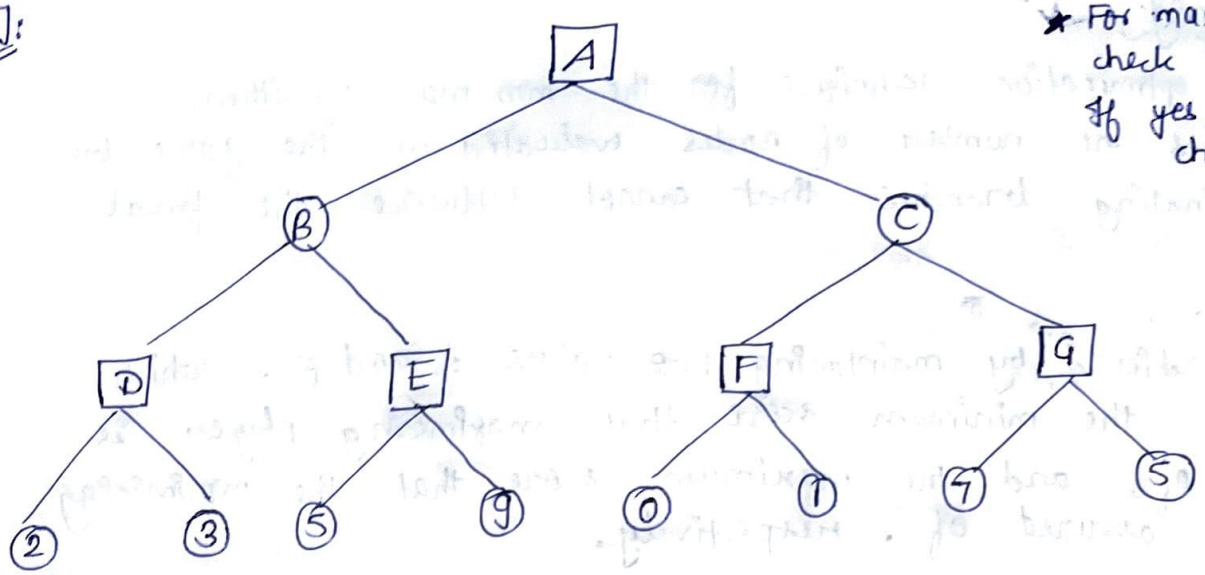
③ Min Node Evaluation

- For each child of a min node, evaluate the child node with
- min-max algorithm with α-β pruning.

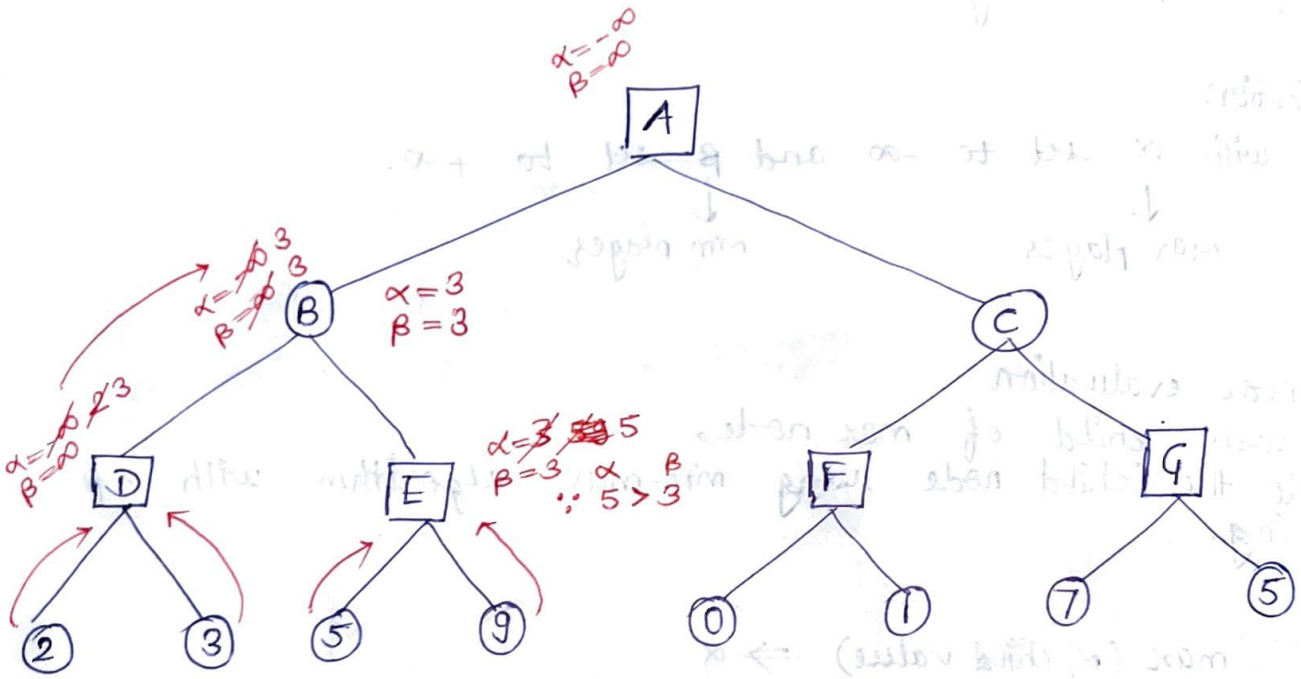
- Update β as min (β, child value)

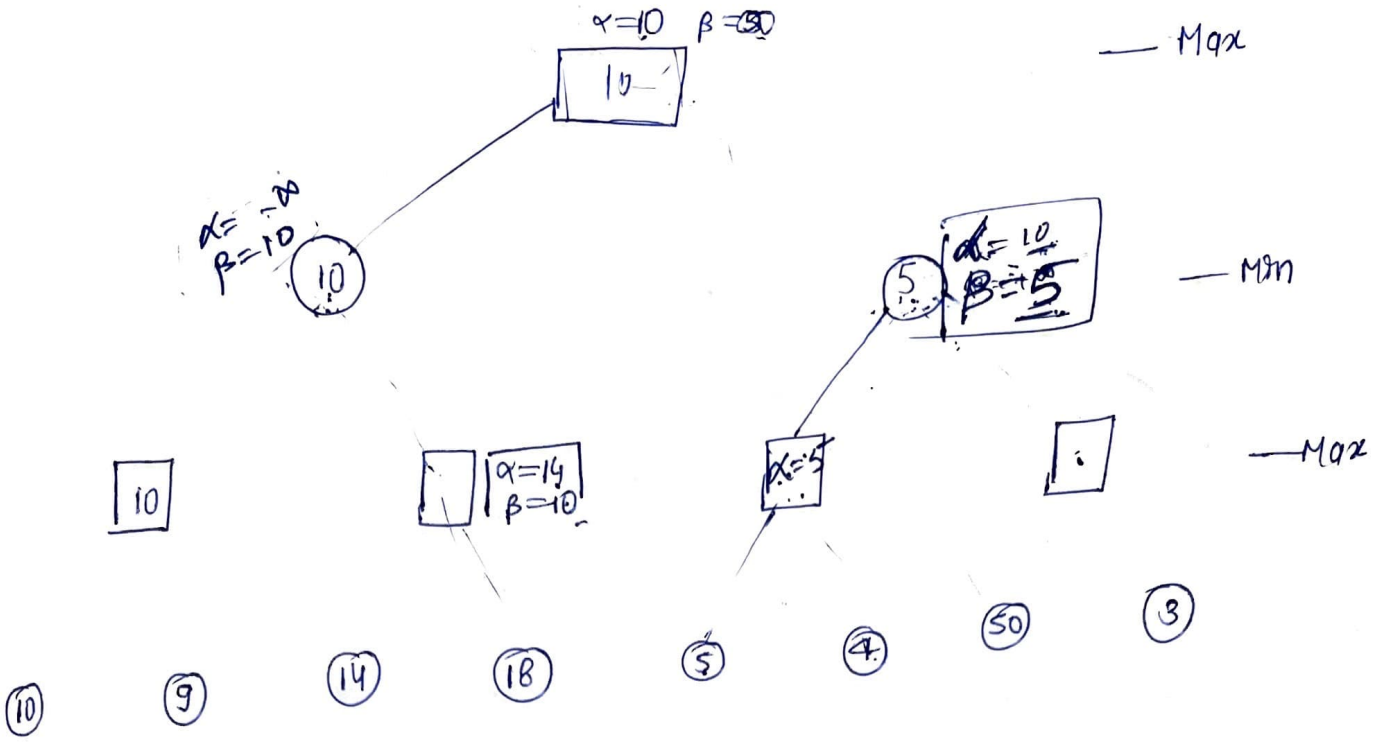
→ If $\beta \leq \alpha$; prune the remaining children

Q]:

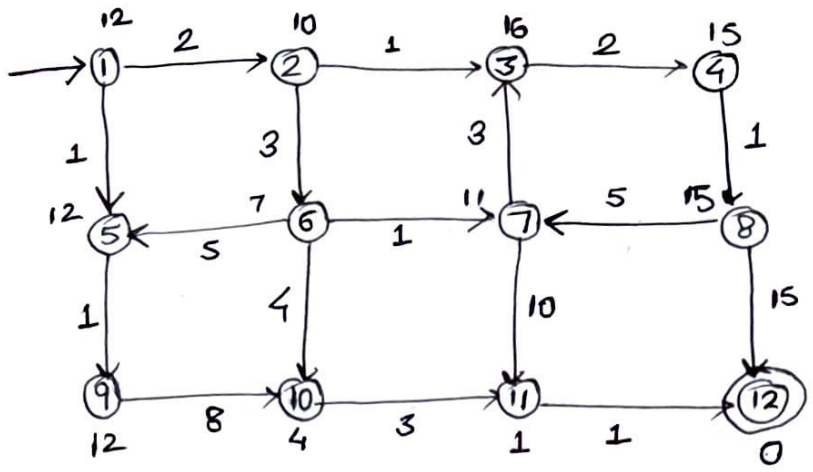


★ For max, check $\alpha \geq \beta$. If yes, no need check other children




α=10 β=∞
10
Max
α=-∞
β=10
10
5
α=10
β=5
Min
10
α=14
β=10
α=5
Max
10
9
14
18
5
4
50
3

Alok Gupta



Solve by:
① UCS
② A*

* ①

Node	5	2
Weight	1	2

⑤

2	9
2	2

* ②

9	3	6
2	3	5

⑨

3	6	10
3	5	8

③

6	4	10
5	5	8

* ⑥

4	7	10
5	6	8

④

8	7	10
6	6	8

⑧

7	10	12
6	8	23

* ⑦

10	11	12
8	16	23

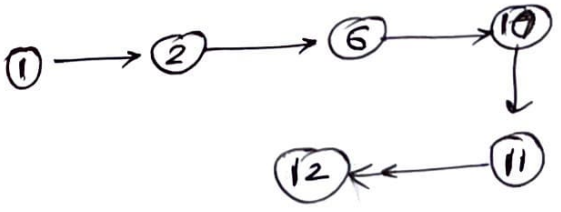
* ⑩

11	~~12~~	12
12	~~16~~	23

* ⑪

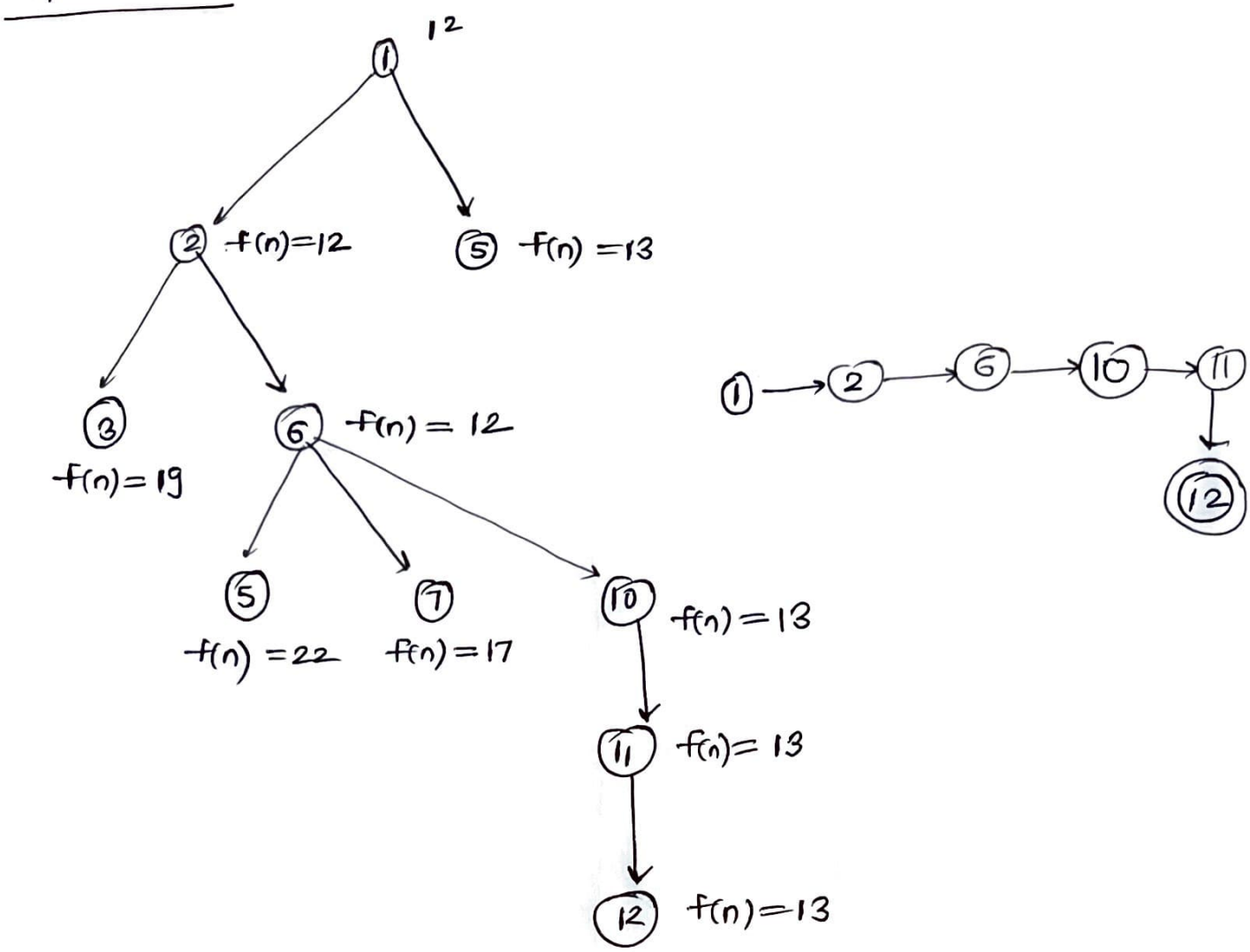
12
12

⑫



Cost — 12 Ans

A* search


12
1
2
f(n)=12
5
f(n) =13
3
f(n)=19
6
f(n) = 12
5
f(n) =22
7
f(n)=17
10
f(n)=13
11
f(n)= 13
12
f(n)=13
1
2
6
10
11
12