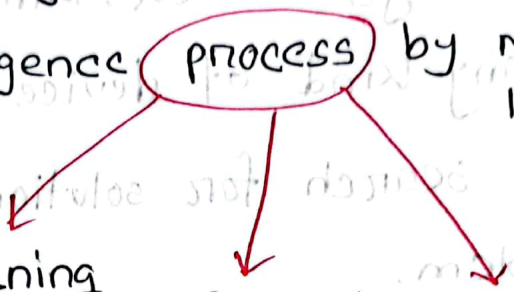
Chapter-1

what is AI ?

→ Artificial Intelligence is the simulation of human Intelligence process by machine like computer system



Four Goals of AI

Thinking Humanly → cognitive Science approach	Thinking Rationaly – Think well
Acting Humanly → Turning test approach	Acting Rationaly → ACT well → heunistic

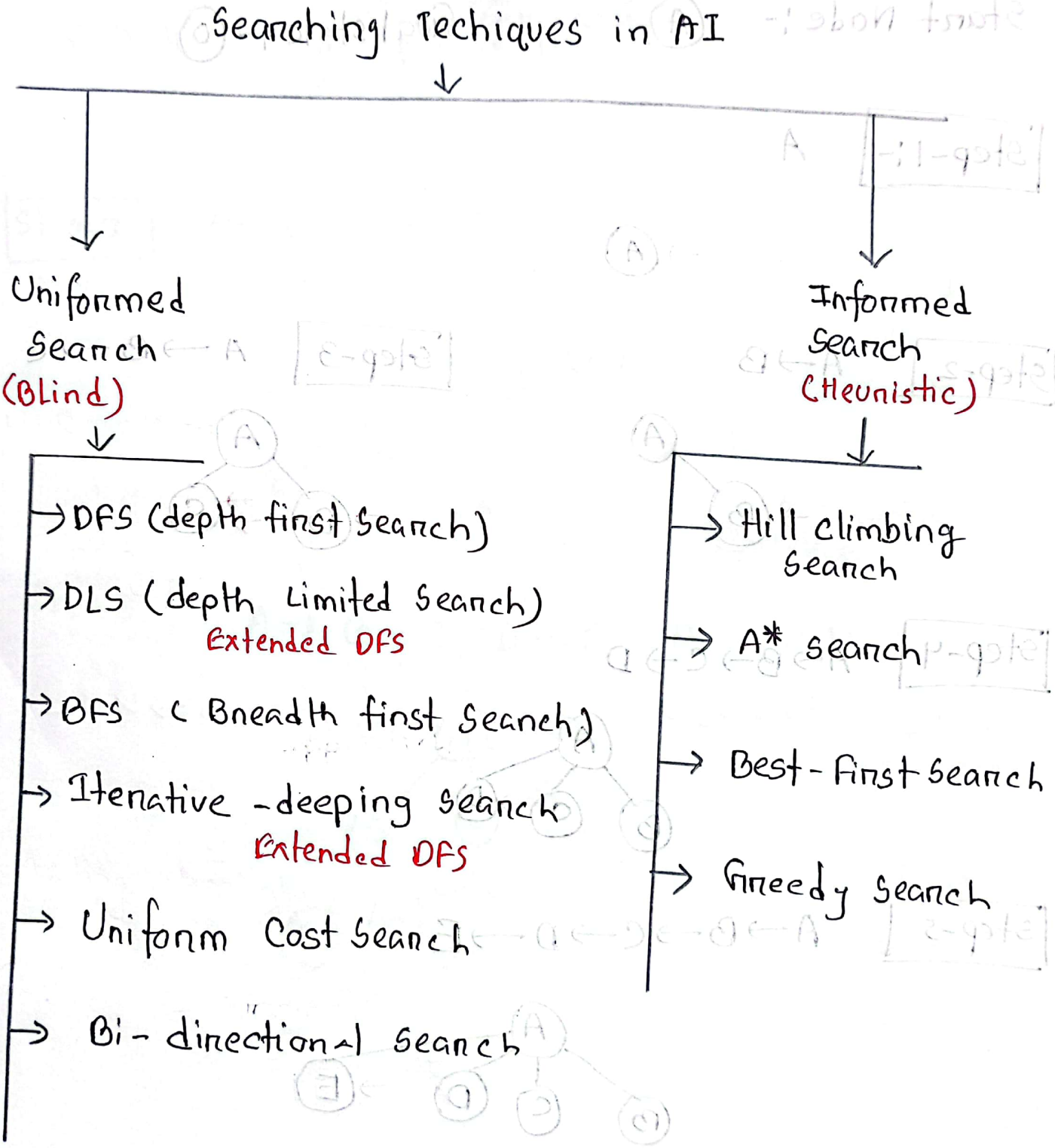
what is Heuristic? method /Rule

→ Heuristic is a rule of thumb, strategy, trick, simplification on any kind of device which drastically limits search for solutions in large problem.

→ Heuristic method enable AI to produce approximate solutions rather than exact one.

→ Instead of looking for perfect solution, heuristic method looks for quick solution that falls with in acceptable range of accuracy

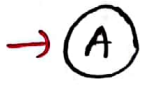
chapter-3


Searching Techiques in AI
Uniformed Search (Blind)
Informed Search (Heuristic)
DFS (depth first Search)
DLS (depth Limited Search)
Extended DFS
BFS (Bneadth first Search)
Iterative-deeping Search
Extended DFS
Uniform Cost Search
Bi-directional Search
Hill climbing Search
A* search
Best-First Search
Greedy Search

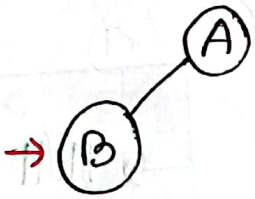
BFS

Start Node :- (A) Final state :- (O)

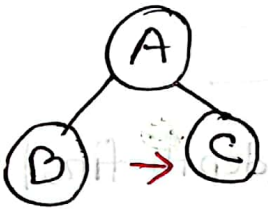
Step-1 :- A



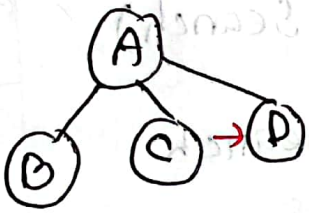
Step-2 A → B



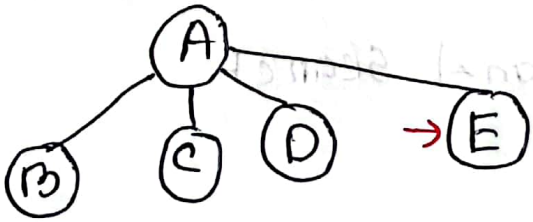
Step-3 A → B → C



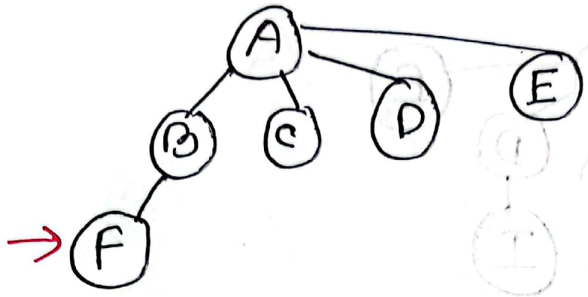
Step-4 A → B → C → D



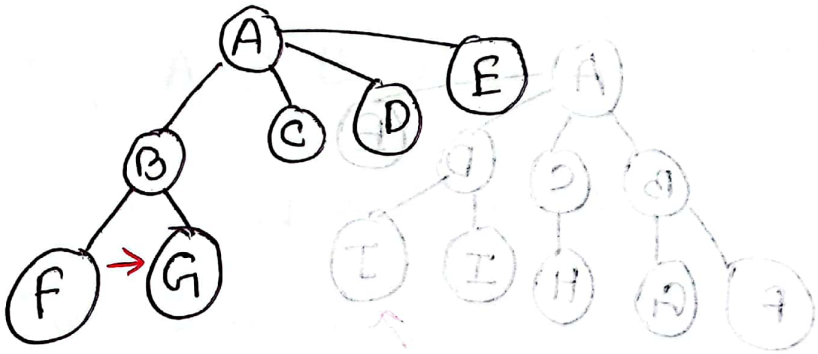
Step-5 A → B → C → D → E



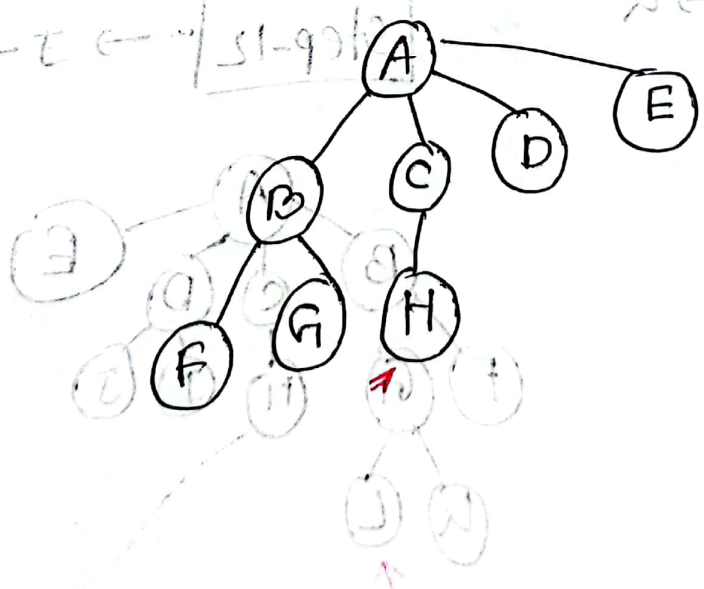
Step-6 A → B → C → D → E → F



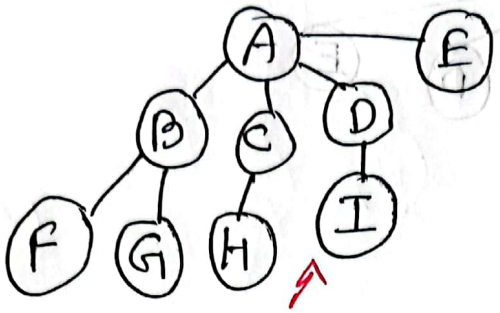
Step-7 A → B → C → D → E → F → G



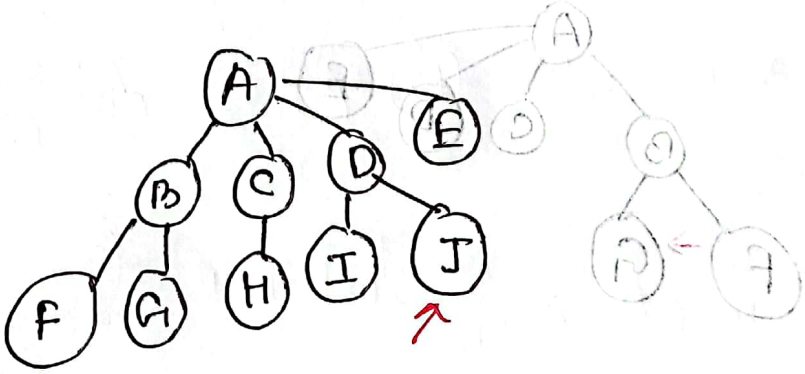
Step-8 A → B → C → D → E → f → G → H



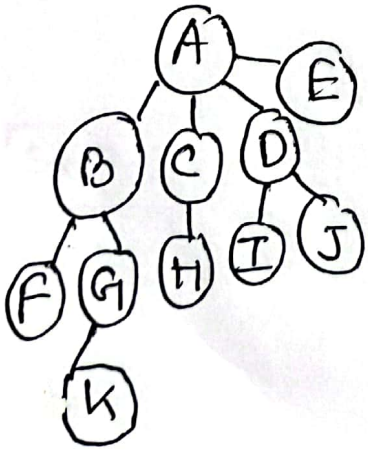
Step-9 $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G \rightarrow H \rightarrow I$



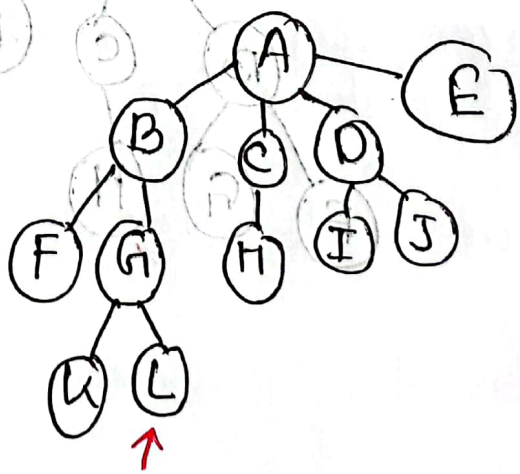
Step-10 $\cdots \rightarrow I \rightarrow J$



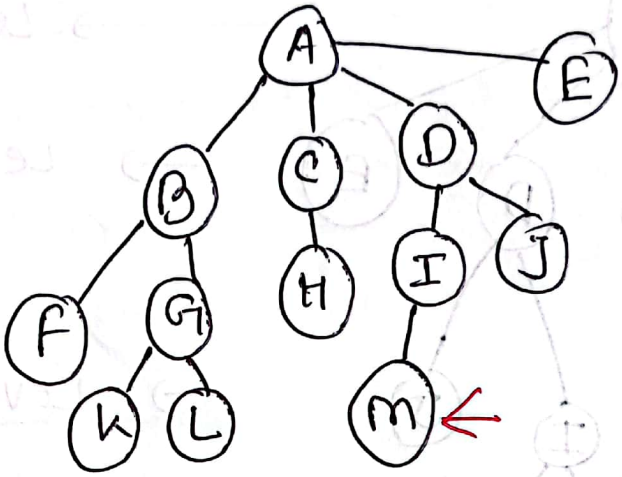
Step-11 $\cdots \rightarrow I \rightarrow J \rightarrow K$



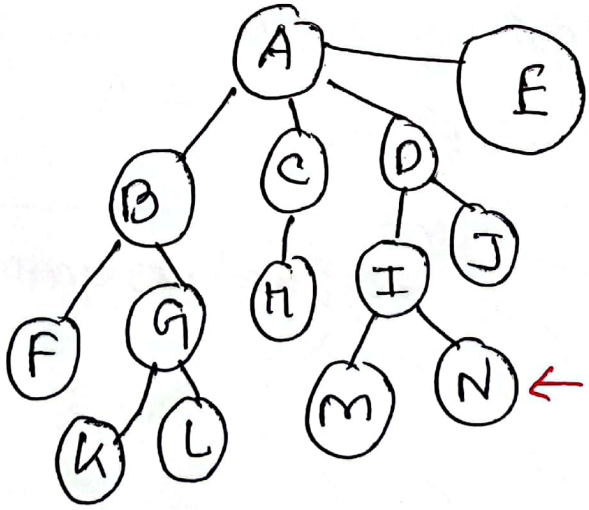
Step-12 $\cdots \rightarrow J \rightarrow K \rightarrow L$



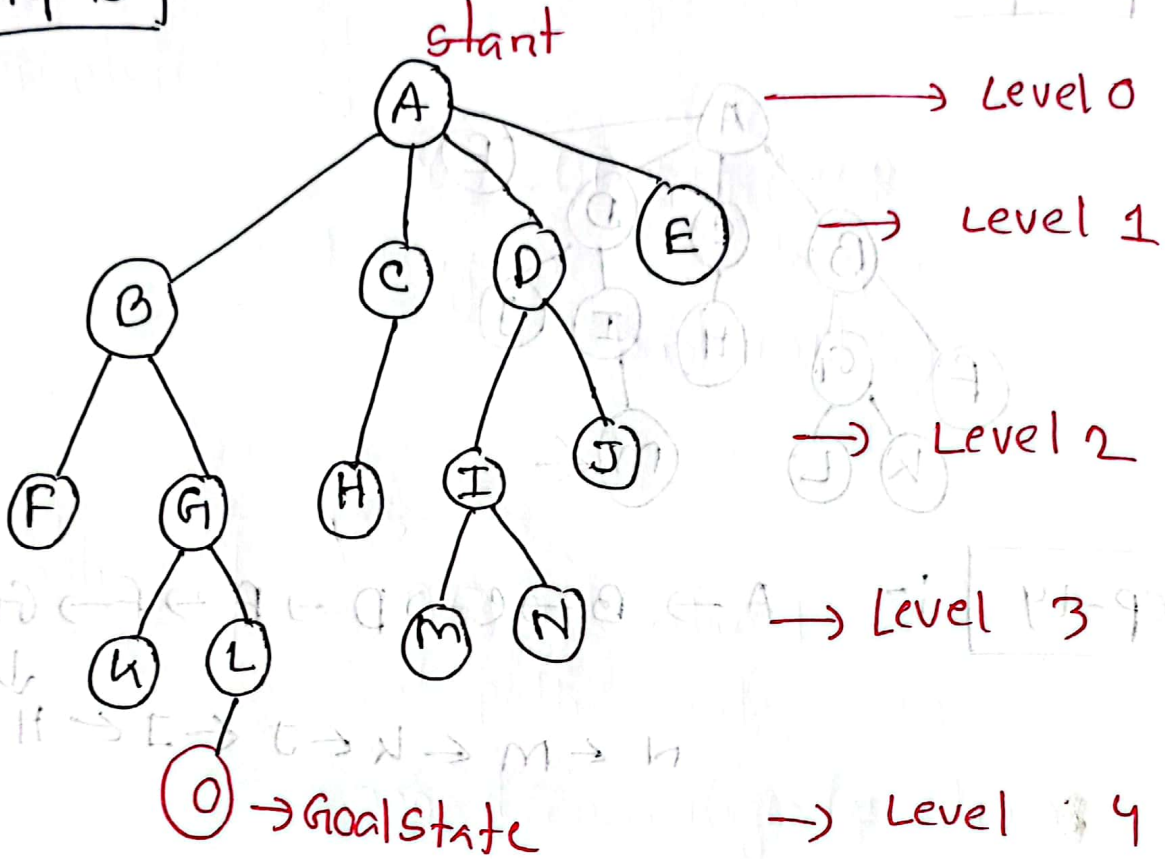
Step-13 $A \rightarrow \cdots\cdots \rightarrow J \rightarrow K \rightarrow M$



Step-14 is- $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G \rightarrow H \rightarrow I \leftarrow J \leftarrow K \leftarrow M \leftarrow N$



Step-15


start
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
→ Goal state
→ Level 0
→ Level 1
→ Level 2
→ Level 3
→ Level 4

BFS

dis
Advantage:- → requires lots of memory (store each level)

→ needs lots of time if goal Node is far from root node

Space problem

Time complexity → $O(b^{d+1})$

b → maximum branching factor in a tree

d → depth of shallowest solution

space complexity → $O(b^d)$

keeps every node in memory

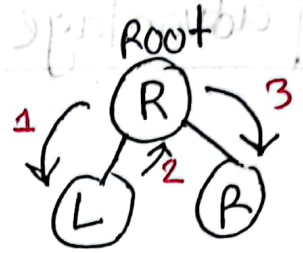
Completeness?

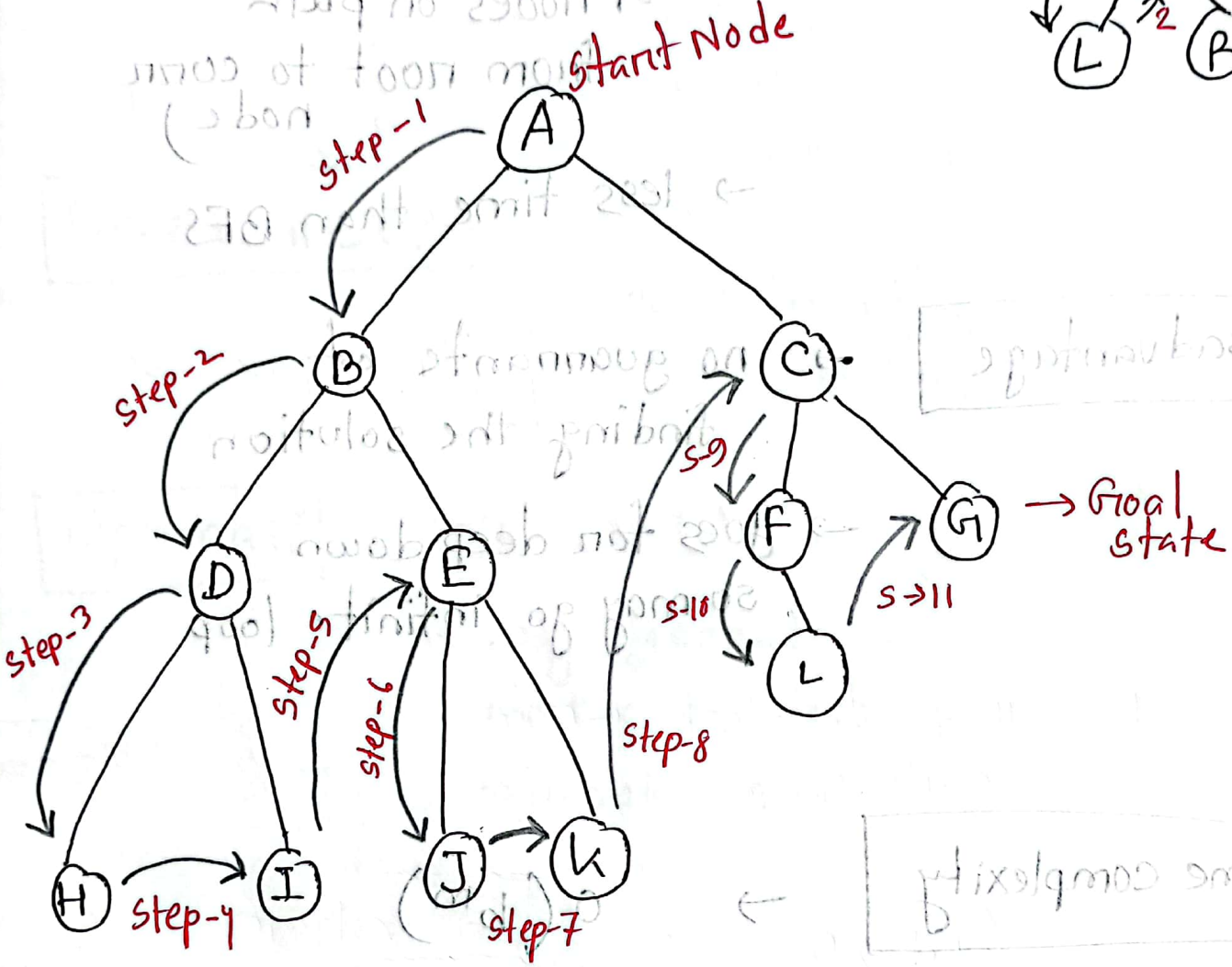
Yes. BFS is complete
it always reaches to goal if
(b is finite)

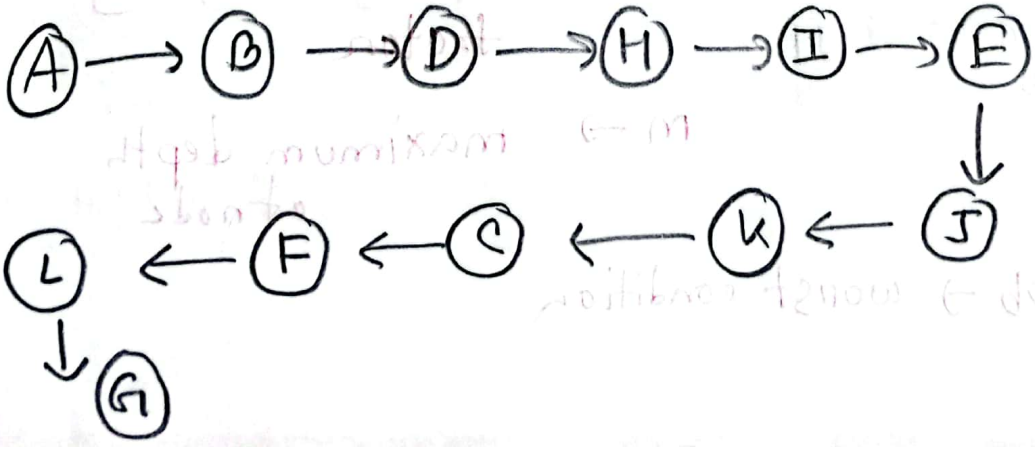
Optimality?

yes
BFS is optimal
if path cost is in
non-decreasing function
of depth of the node
Ex:- (Step cost = 1)

DFS Depth first search







advantage :-

→ use less memory
(only store a stack of nodes on path from root to curr node)

→ less time than BFS

disadvantage

→ no guarantee of finding the solution

→ goes for deep down so may go infinite loop

Time complexity → $O(b^m)$

b → maximum branching factor

m → maximum depth of node

m > b → worst condition

space complexity $O(bm)$

Linear space

completeness? NO

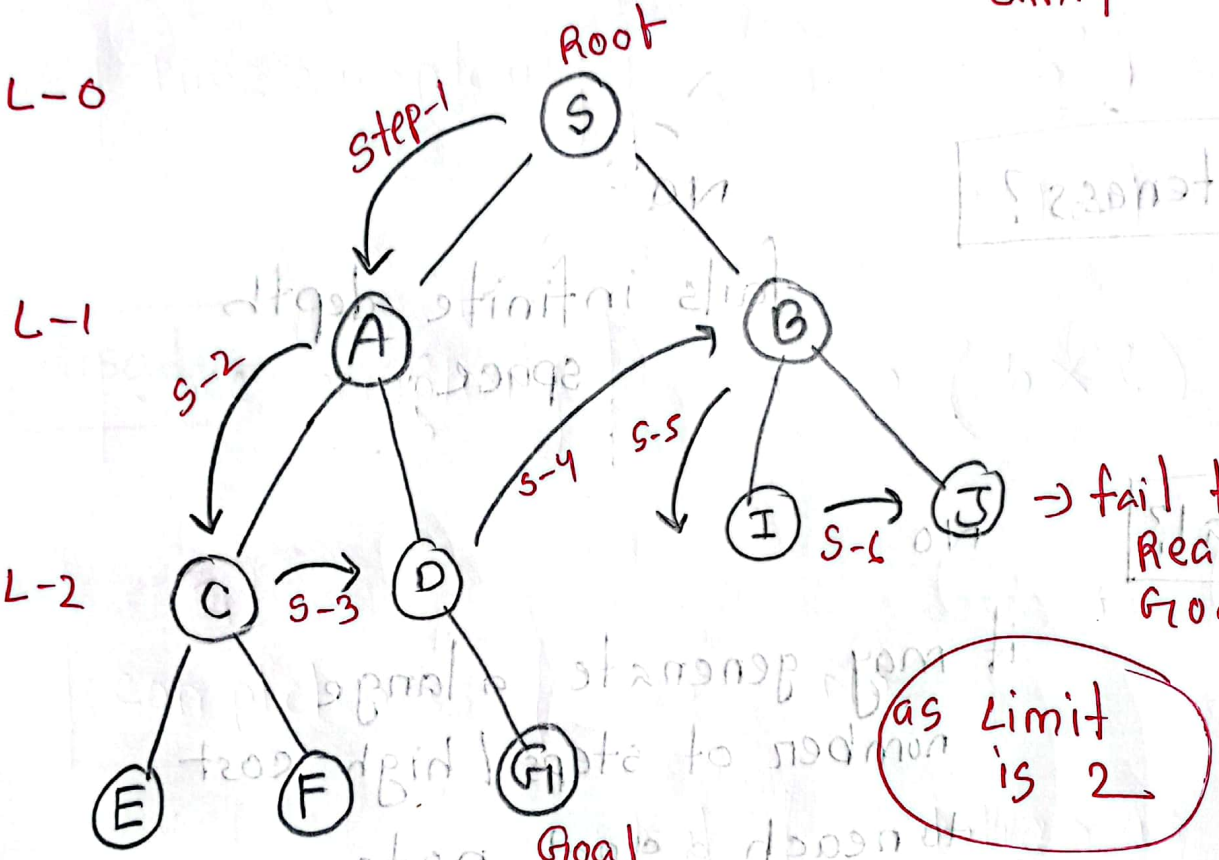
fails infinite depth space

Optimal? NO

it may generate a large number of steps / high cost to reach a goal node

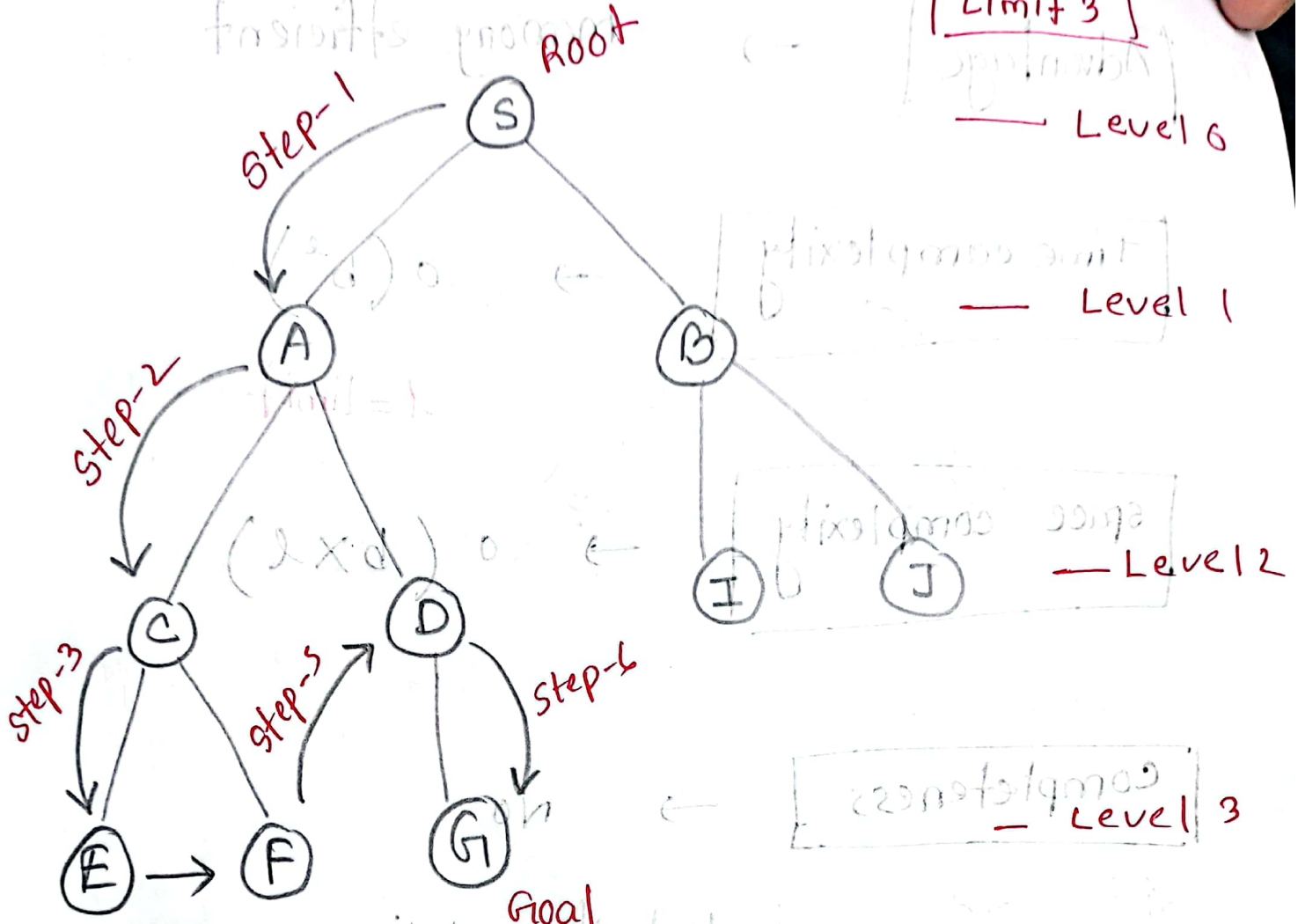
DLS Depth Limited search

Limit = 2



S → A → C → D → B → I → J → X

fail
no solution



S → A → C → E → F → D → G

Success

Reached goal

Advantage → memory efficient

time complexity → $o(b^l)$

l = limit

space complexity → $o(b \times l)$

completeness → NO

but if solution is above depth limit then yes

optimal → NO

even if $l > d$

IDS — Iterative Deepening Search

Step-1 Level 0

DLS bound with = 0

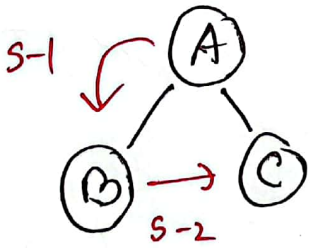
— Level 0

Iteration^{-1} :- A

∴ Goal not found

∴ increase Level = 0+1

Step-2 Level 1 ; DLS bound = 1



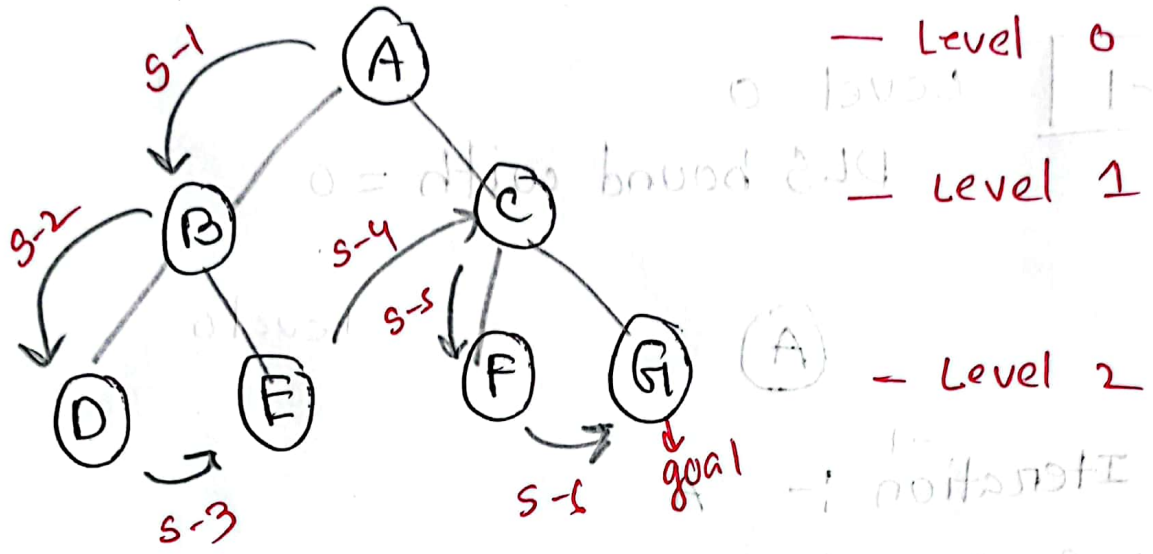
— Level 0

— Level 1

Iteration^{-2} :- A → B → C

∴ Goal not found increasing level

Step-2 Level : 2 DLS bound : 2



Iteration :- 3 A → B → D → E → C → F → G

Goal node found in 3rd iteration

advantage — BFS (FAST beanch)
— DFS (memory efficiency)

Time completexity - $o(b^d)$

Spacc complexity — $o(bd)$

Completeness? — Yes

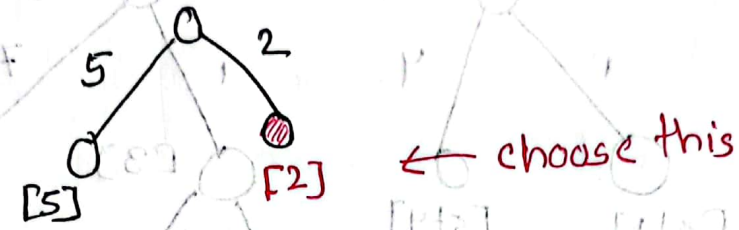
complete when
b is finite

Optimal ? — Yes
if step cost = 1
or
increasing function
of depth

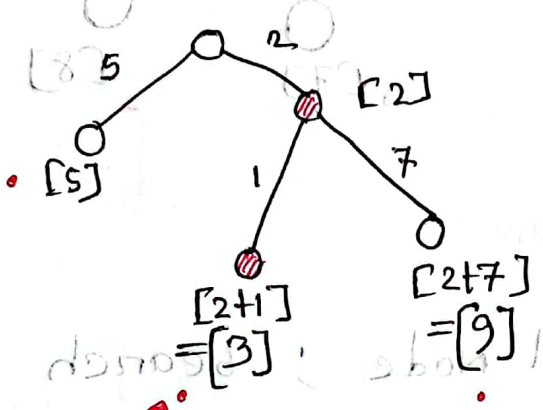
Uniform Cost

Expand node with samallest cost g(n)

Step-1

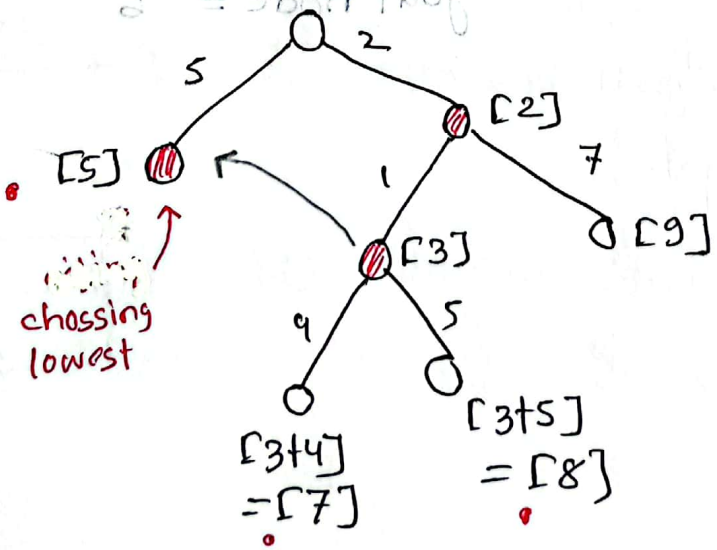


Step-2



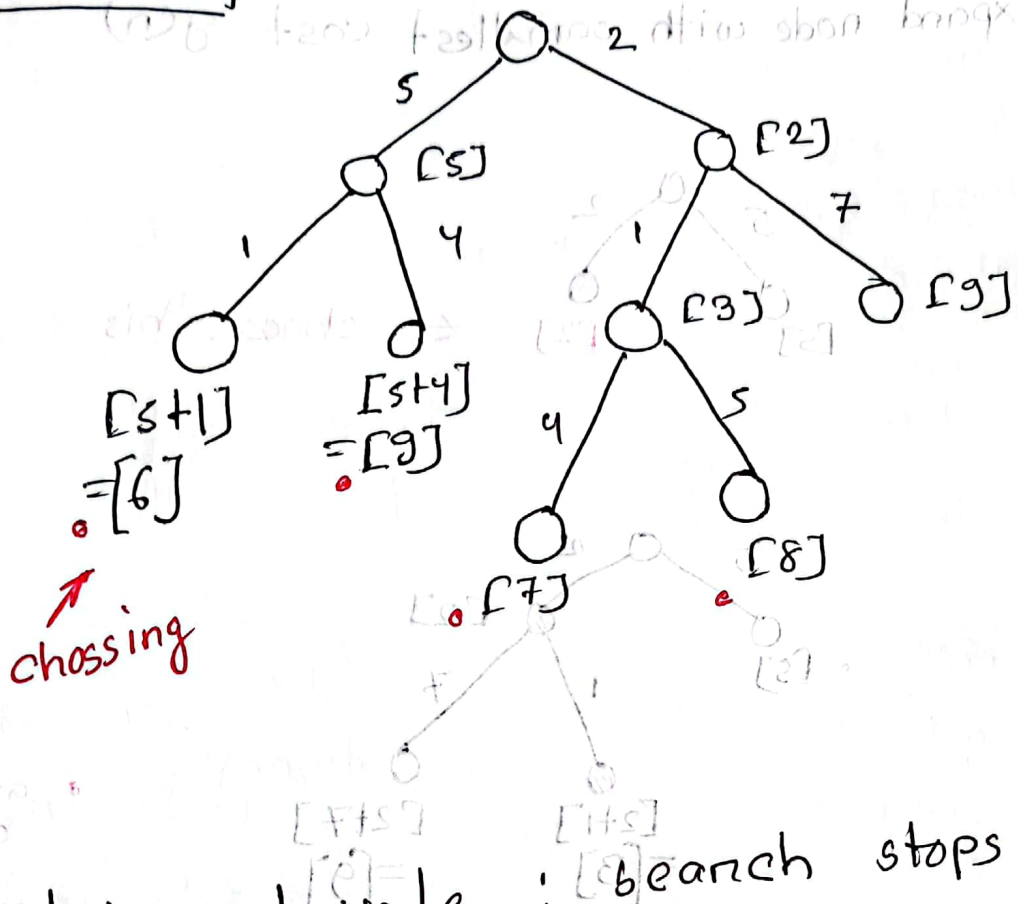
• available option to g.
[5]
[3] ✓
[9]

Step-3



[5] ✓
[7]
[8]

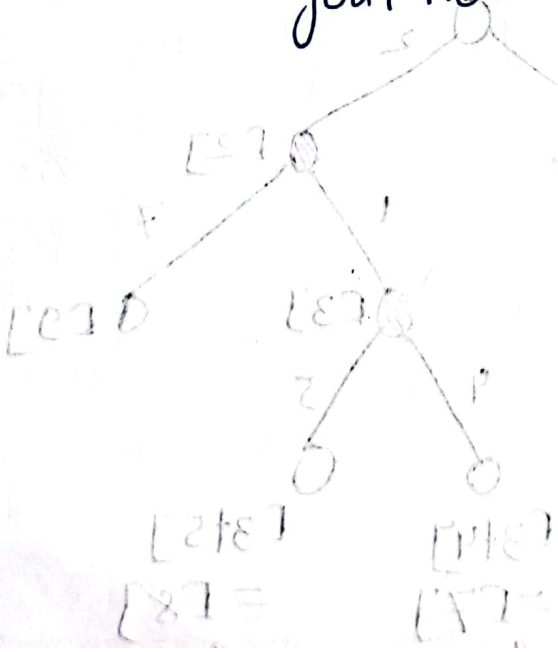
step-4



[6] ✓
[9]
[7]
[8]

as it is goal node ; search stops

∴ cost to reach goal node = 6



Disadvantage — doesn't cane about number of steps only concened about path cost
may stuck infinity loop

Time complexity $O(b^{\lceil C^*/\epsilon \rceil})$

b = branching factor

C^* = cost of optimal solution

ϵ = each step to get closen to goal node

Space complexity $O(b^{\lceil C^*/\epsilon \rceil})$

Completeness ? Yes if step cost $\geq \epsilon$
otherwise stuck in loop

optimality ? Yes for any step cost

Uninformed search	informed search
① is a searching technique that has no additional information about the distant from the current state to the goal	① is a searching technique that has additional information about the distant from the current state to goal.
② No use of knowledge	② Use knowledge to find steps to the solution
③ medium efficiency	③ Highly efficient
④ cost companatively High	④ Cost Low
⑤ Speed Slowen than informed	⑤ Find solution more quickly

Informed	Uninformed
⑥ Example Algorithm Best First Search Greedy search A* search	⑥ BFS, DFS, IDS Uniform Cost
⑦ Known as Heuristic Search	⑦ Known as Blind Search
⑧ less length while implementation	⑧ more length
use heuristic /domain knowledge to choose the best move	use

what is heuristic function ?

Heuristic function is a function, $f(n)$ that gives an estimation on the "cost" of from node n to goal state - so the node with least cost among all possible choice can be selected for expansion first.

Approaches of defining f

(1) f measure the value (goodness) of current state

(2) f measure the estimated cost of getting to the goal from the current state

$$f(n) = h(n)$$

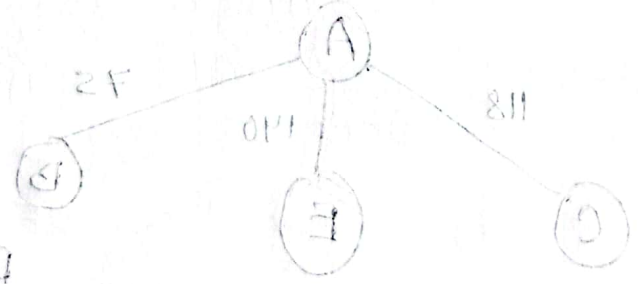
$h(n)$ = is the estimation of the cost to get from n to goal

(3) f measures the estimated cost of getting to goal state from current state and the cost of existing path to it.

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

$g(n)$ = the cost to get n from intial state

$h(n)$ = an estimate of cost to get from n to goal



$f(n) = h(n)$

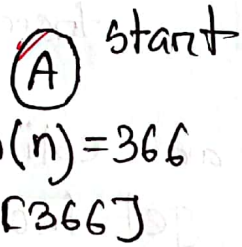
Greedy Best First Search Algorithm

→ follow the lowest h(n)

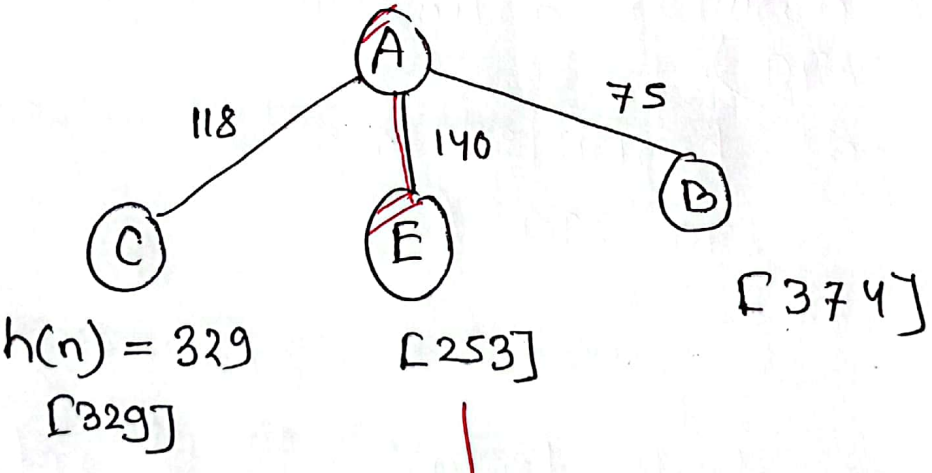
→ dont care about cost

→ just focus on h(n)

Step-1

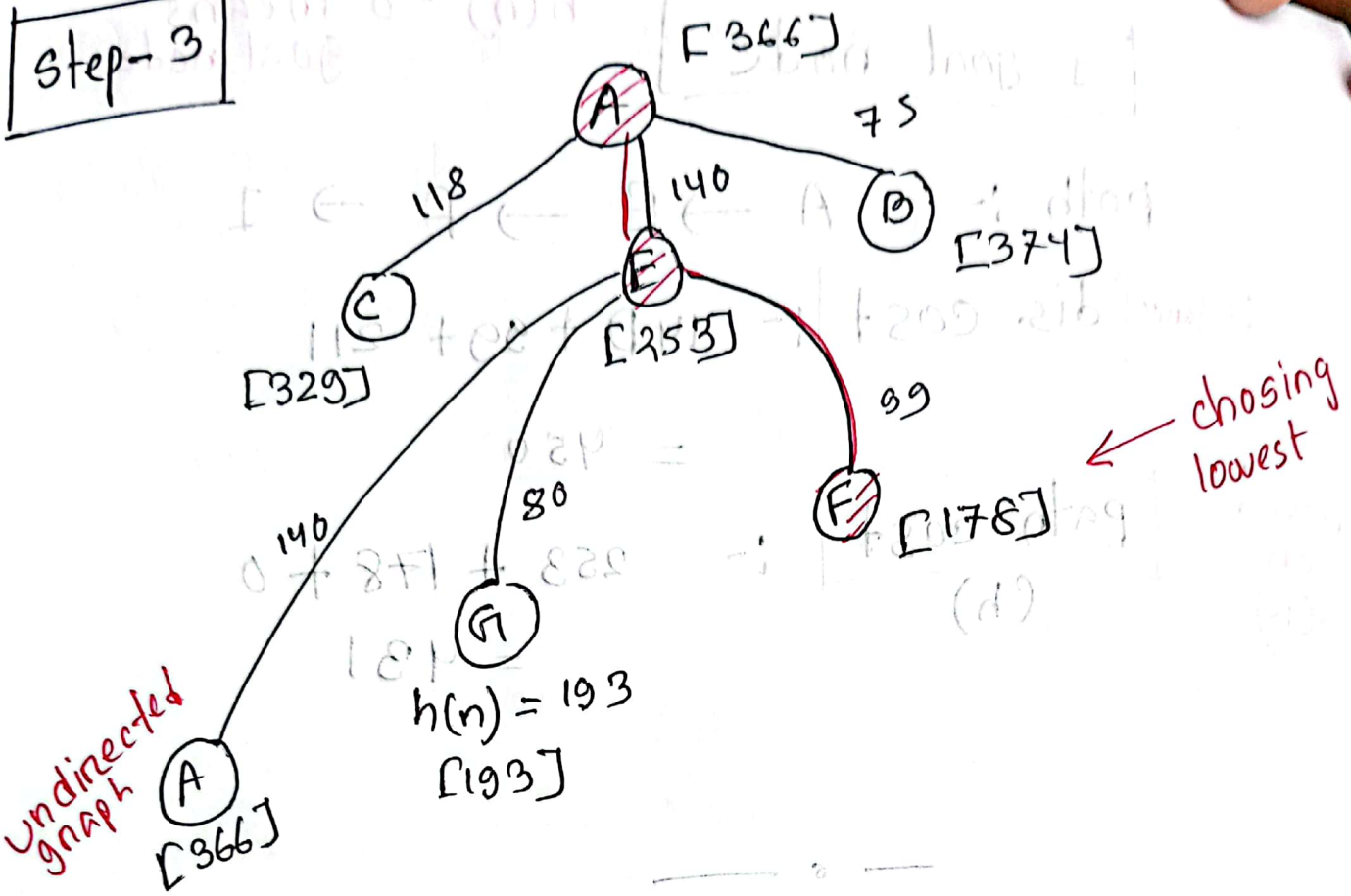


Step-2

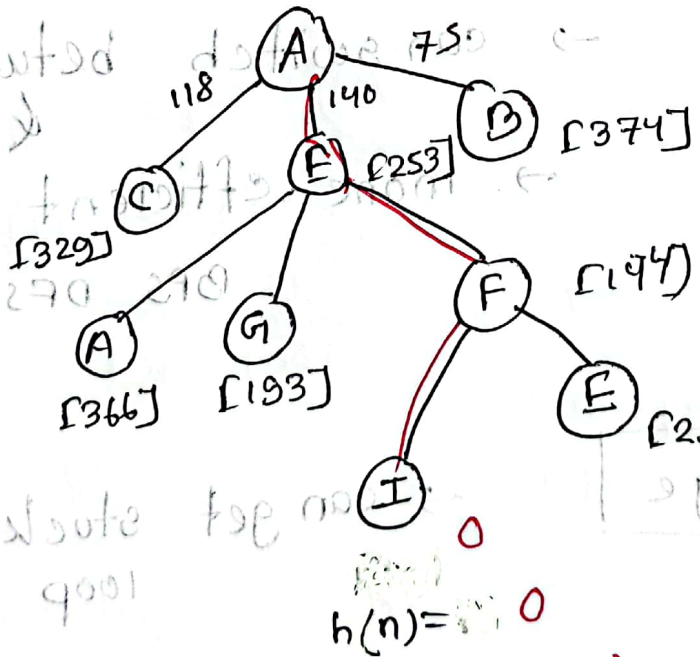


chosing cause lowest h(n)

Step-3



Step-4



lowest h(n) select

1 goal node

$h(n) = 0$ means goal node

path :- A → E → F → I

distance — dis. cost :- 140 + 99 + 211

= 450

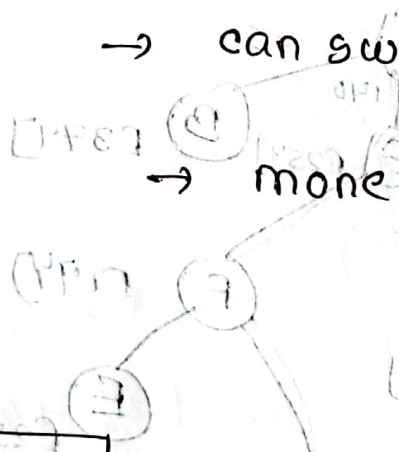
based on h(n) — path cost (h) :- 253 + 178 + 0

= 431

— ∘ —

Advantge → can switch between BFS & DFS

→ more efficient than BFS DFS



disadvantge → can get stuck in a loop like DFS

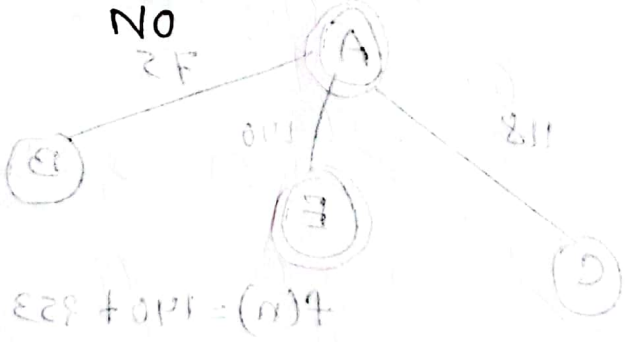
Time complexity $O(b^m)$

b = branching factor
m = maximum path length

Space complexity $O(b^m)$

Completeness No (even if state space finite) incomplete cause donot checking of repeated states

Optimal No



Combination of uniform cost and greedy

A* search

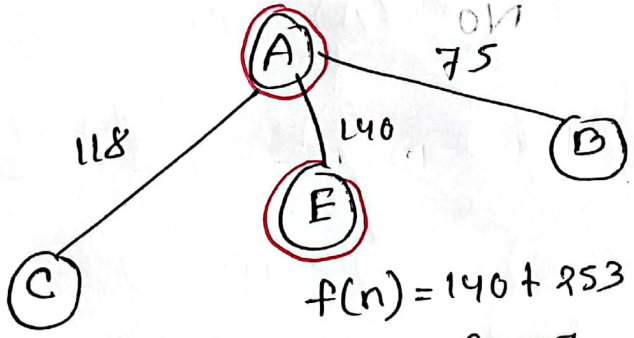
$$f(n) = g(n) + h(n)$$

- $g(n)$: exact cost to reach n node
- $h(n)$: estimation to reach goal node

Step-1

$f(n) = 0 + 366 = [366]$

Step-2



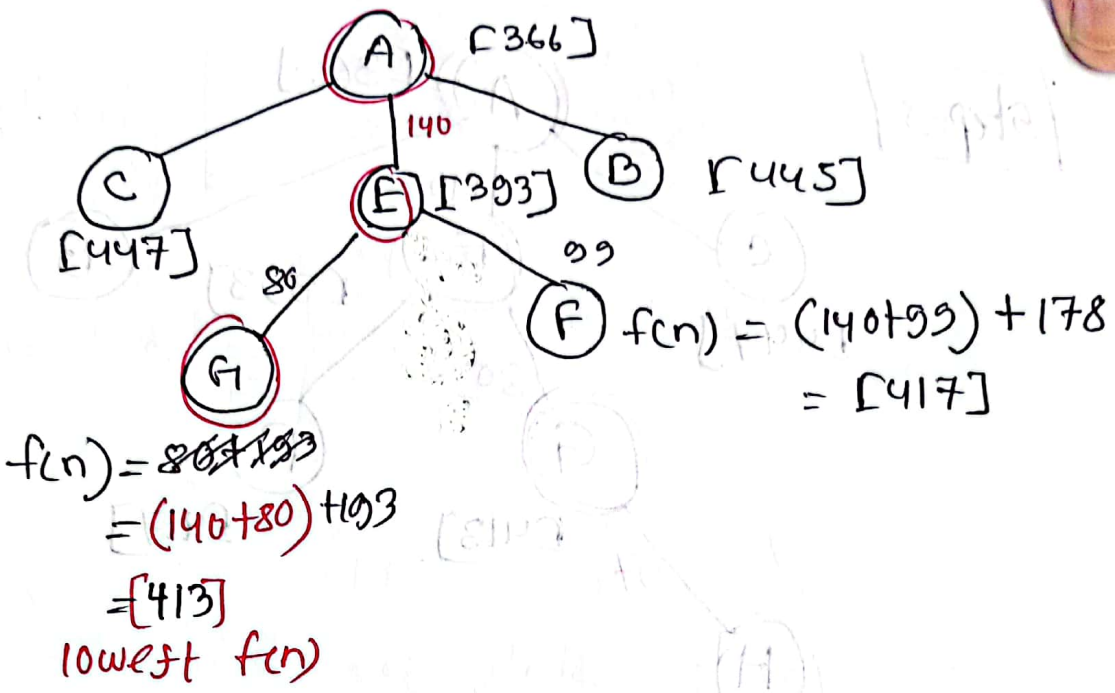
$f(n) = 175 + 378 = [445]$

$f(n) = 140 + 253 = [393]$

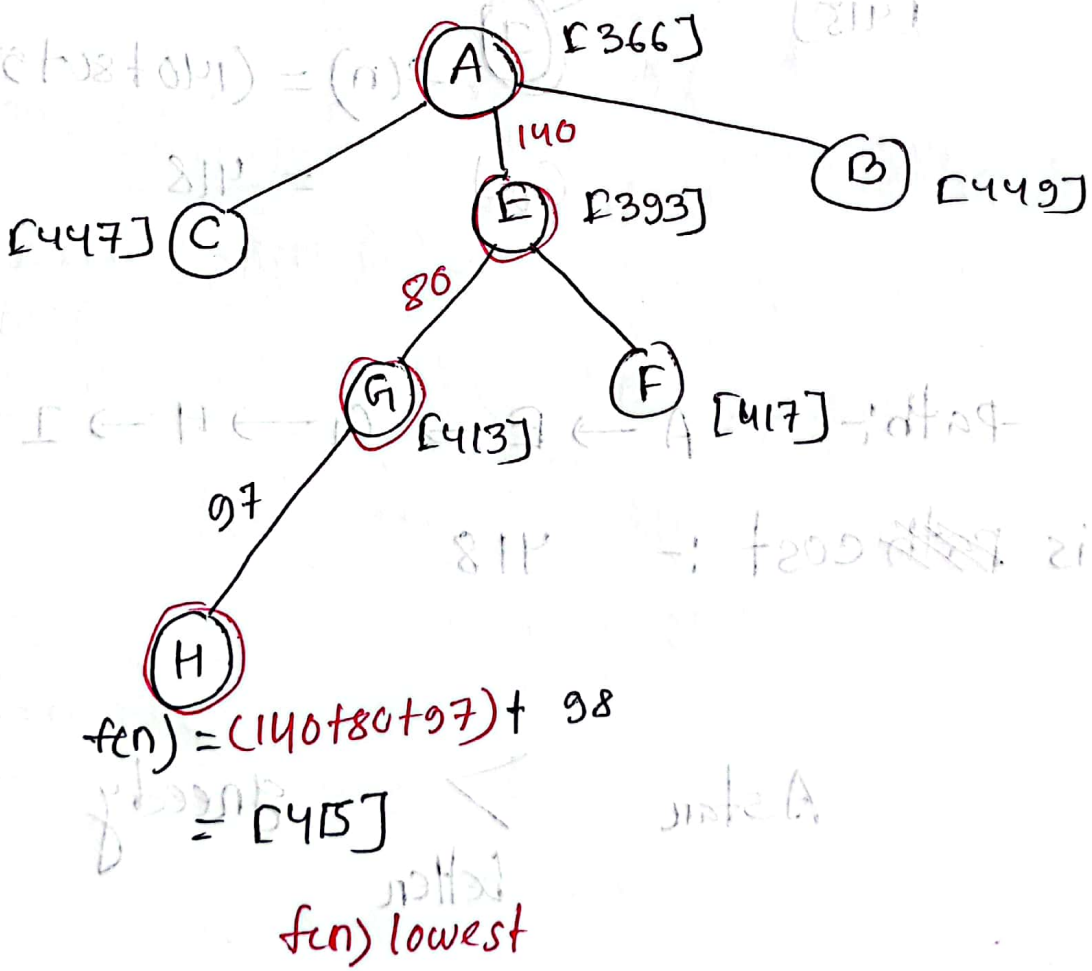
$f(n) = 118 + 329 = [447]$

lowest f(n)

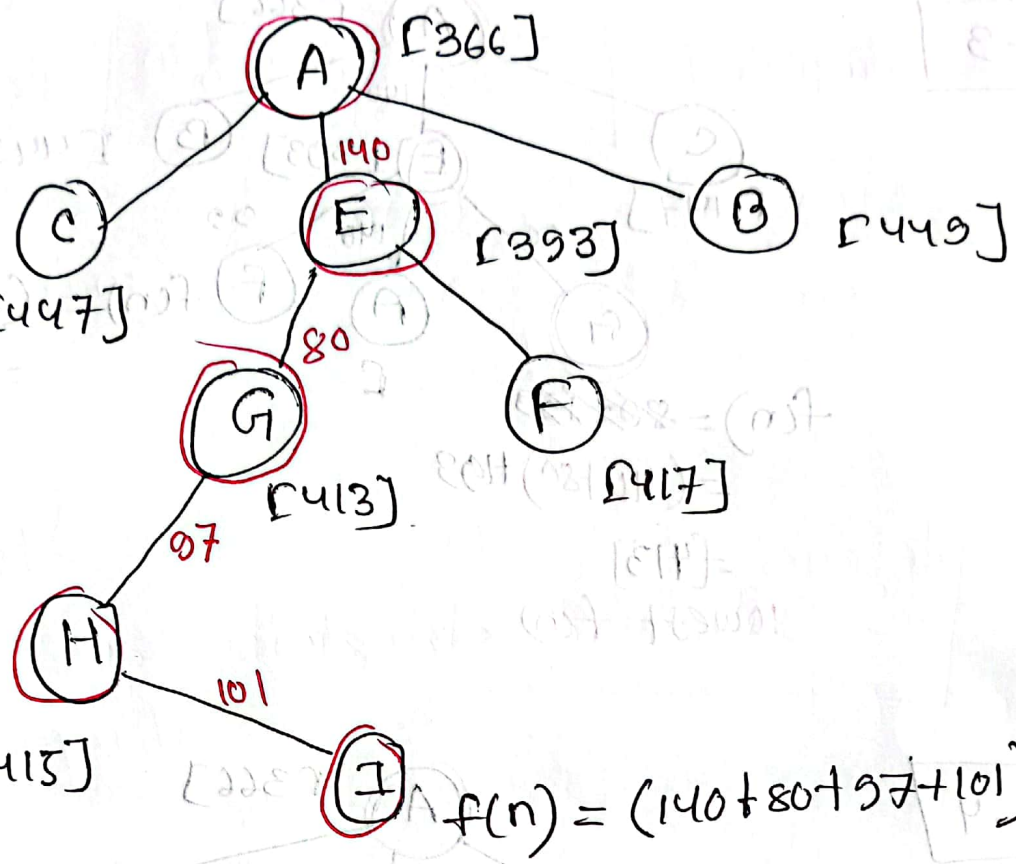
Step-3


A
[366]
140
C
[447]
E
[393]
B
[445]
80
99
G
F
f(n) = (140+99)+178
= [417]
f(n) = (140+80)+193
= [413]
lowest f(n)

Step-4


A
[366]
140
[447]
C
E
[393]
B
[449]
80
G
[413]
F
[417]
97
H
f(n) = (140+80+97)+98
= [415]
f(n) lowest

Step-5



$f(n) = (140+80+97+101) + 0$

$= 418$

path :- A → E → G → H → I

dis ~~path~~ cost :- 418

Astar > greedy

better

advantage

- best of all
- can also solve complex problems

disadvantage

- Higher memory requirment
- not always produce shortest path

Time complexity → $O(b^d)$

depends on $h(n)$

Space complexity → $O(b^d)$

completeness

Yes if

(b = finite)

(cost is fixed)

optimal Yes

if (h(n) is admissible heuristic)

What is admissible heuristic ?

a heuristic h(n) is admissible

if for every node n, $h(n) \leq h^*(n)$

where $h^*(n)$ is the true cost to reach the goal state from n

What are the concepts in a search?

1. Initial state
2. Action / Transition Model
3. state space graph
4. Goal te-st
5. solution / optimal solution

State space search

Def-1

A state space search is a method used in AI to find solution to a problem searching through the set of possible states.

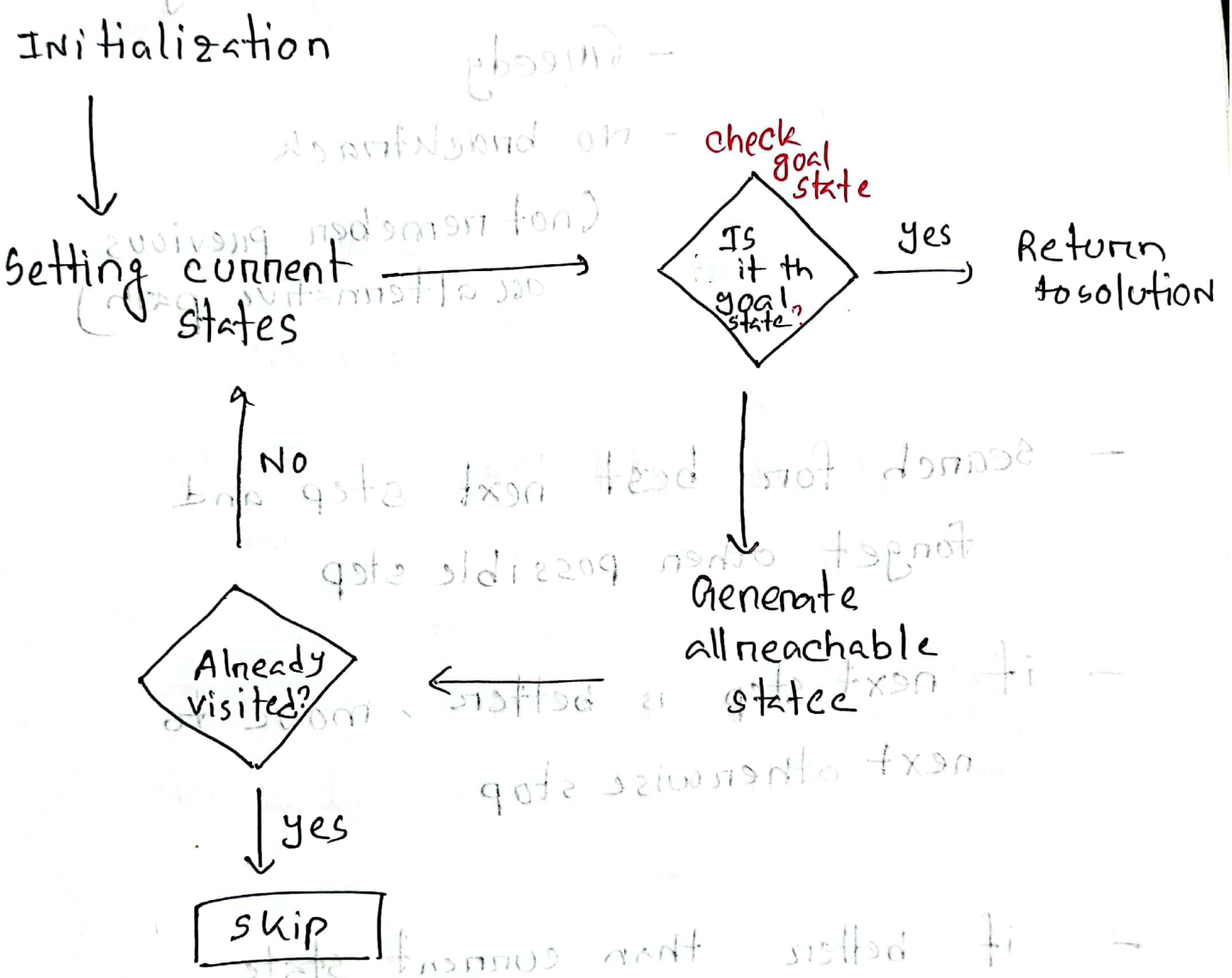
Def-2 Formal def

State space is represented 4 tupes [N, A, S, GD]

1. N → set of nodes or states of graph
2. A → set of arch or links between nodes
   (arch: an instance of an action)
3. S → subset contain start state of a problem
4. GD → subset containg goal state of problem

Algorithm

Ex :- A*
BFS, DFS
Hill climbing



node type

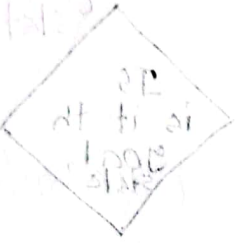
① Not generated yet

② Open : genenated but not expanded

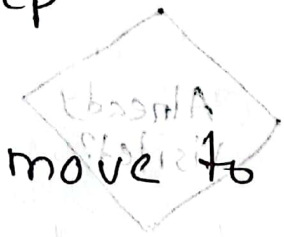
③ closed
genenated and expanded

HILL climbing

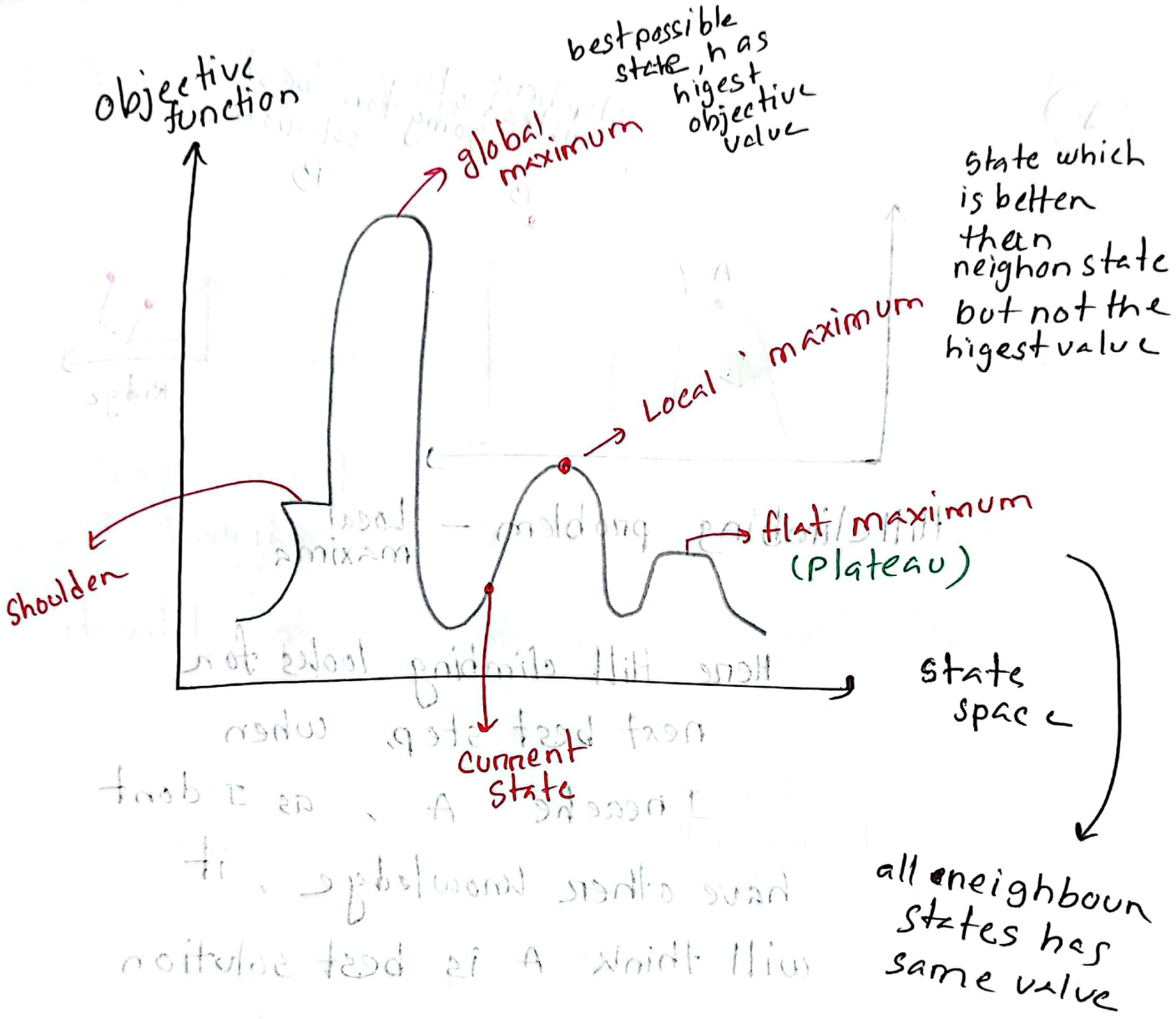
- Local search Algo
- Greedy
- no backtrack
  (not remeber previous or alternative path)



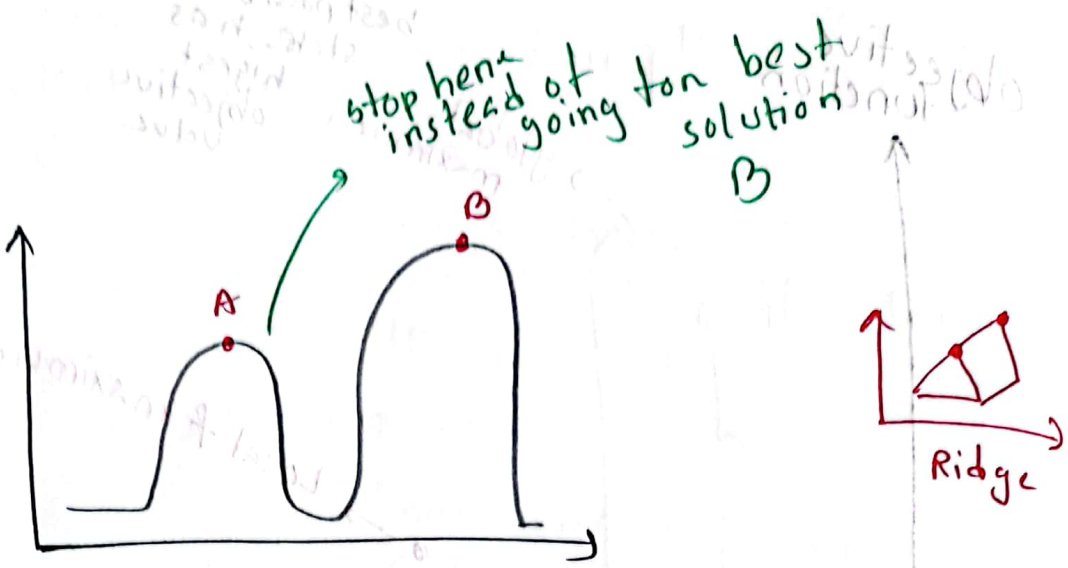
- search for best next step and forget other possible step
- if next step is better, move to next otherwise stop



- if better than current state then # it is a new current state


objective function
best possible state, has higest objective value
global maximum
state which is better than neighon state but not the higest value
Local maximum
Shoulder
flat maximum (plateau)
state space
current state
all neighboun states has same value

1



Hillclimbing problem – Local maxima

Here Hill climbing looks for next best step, when I reache A, as I dont have other knowledge, it will think A is best solution and stop.

willnot move forward

$f(2+2+2) = 6$

Stant

1	2	5
8	7	4
	6	3

goal

1	2	3
8		4
7	6	5

$f=(0)$

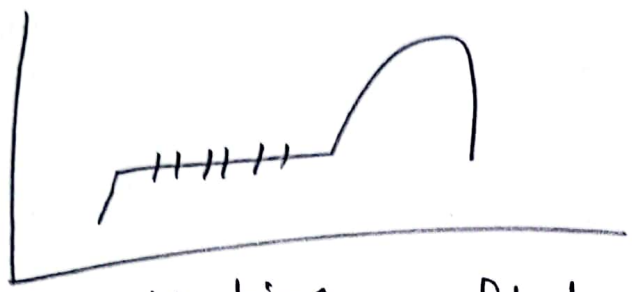
f $=(1+2+2+2)$ $= -7$

1	2	5
	7	4
8	6	3

1	3	5
8	7	4
6		3

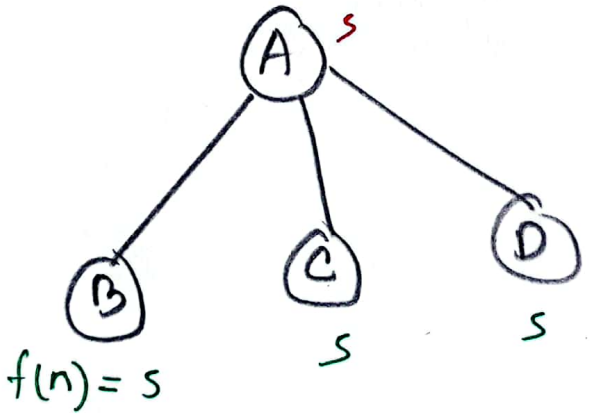
$f(-7)$

(2)



Hill climbing :- flat maximum / plateau problem

∴. if all neighbouring states has same value it will not move for next step and stops.



Algorithm stop to go next step cause all has same value it only search for next best option

IDS

starts from level 0. search for goal node though iteration.

if not found increase Level of 1. now search for goal node

if not found increase level by 1 and search through iteration until yo found the goal node