10.10.2022

Knowledge [past exp used to make a decision]

sensor → decision maker → action

Approaches to AI [types of AI systems]

thought

human-like ←	→ ideal
Systems that think like humans Eg: Turing test	Systems that think rationally Eg: thought process
Systems that behave like humans Eg: cognitive science	Systems that behave rationally Eg: rational agents

behaviour

★ Turing test: tests whether a machine thinks like a human.

Portions

chap 1 Pred logic
→ prop logic
2, 3 proof by laws
truth table
proof FC
proof resolution

11.10.2022

Propositional Logic

Proposition: it's a statement or assertion with a truth value, i.e. it is always either true or false

Eg: $2+3=5$ is always true.

$x > 5$ isn't a proposition

Represented as P, Q, R, etc..

Connectors: we use this to write well-defined formulas.

And $\wedge$

Or $\vee$

Not $\neg$

Implication $\rightarrow$

Equivalence $\Leftrightarrow$

Eg: $P \wedge Q \rightarrow R$

Truth tables:

And:

P	Q	$P \wedge Q$
F	F	F
F	T	F
T	F	F
T	T	T

Or:

P	Q	P ∨ Q
F	F	F
F	T	T
T	F	T
T	T	T

implication

P	Q	P → Q
F	F	T
F	T	T
T	F	F
T	T	T

if P is T then Q has to be T, we don't care if P is F

iff

P	Q	P ⟺ Q
F	F	T
F	T	F
T	F	F
T	T	T

if P is T, then Q and vice versa

~~To write~~ a

Eg:

P: Alice is a PG student

Q: Alice registered for elective course 'x'

R: Alice is a Mtech ARI student.

$P \wedge Q \rightarrow R$

Eg. If Alice is a student who scored CGPA < 8, then she will <u>not</u> get distinction.

P: Alice is a student

Q: Alice scored CGPA < 8

R: Alice will get distinction.

$P \wedge Q \rightarrow \neg R$

Q If Alice is a CS student who borrowed book from central library then she won't get access to dept. library and digital library.

P: Alice is a CS student
Q: Alice borrowed a book from central library
R: Alice ~~has~~ will get access to dept. library
S: Alice will get access to digital library

$P \wedge Q \rightarrow \cancel{\neg(R \wedge S)}\ \neg R \wedge \neg S$

$\neg(R \wedge S)$
$\rightarrow \neg R \vee \neg S$
so don't write this as meaning changes

Q. $P \wedge Q \rightarrow \neg R$

P	Q	R	$P \wedge Q$	$\neg R$	$P \wedge Q \rightarrow \neg R$
F	F	F	F	T	T
F	F	T	F	F	T
F	T	F	F	T	T
F	T	T	F	F	T
T	F	F	F	T	T
T	F	T	F	F	T
T	T	F	T	T	T
T	T	T	T	F	F

* Tautology: True for all possible values of the prepositions
  a well defined formula that's
Eg: $P \vee \neg P$ ↳ aka valid

* Contradiction False for all
Eg: ↳ aka unsatisfiable

$A \wedge F = F$
$A \vee F = A$
$\sim(A \vee B) = \sim A \wedge \sim B$

Laws: used to reduce statements

1. $A \wedge T = A$
2. $A \wedge B = B \wedge A$
3. $A \vee 1 = 1$
4. $A \vee AB = A(1 \vee B) = A \cdot 1 = A$
5. $\neg(A \wedge B) = \neg A \vee \neg B$ } de Morgan's Laws
6. $\neg(A \vee B) = \neg A \wedge \neg B$

$A \wedge T = A$
$A \wedge B = B \wedge A$
$A \vee T = T$

$A \vee AB$
$= A(T \vee B)$
$= A \wedge T$
$= A$

17.10.2022

Binary clause: has 2 variables → $p \vee q$
Horn clause: 0 or atmost 1 positive literal.
Eg: $\neg p \vee \neg q$, $\neg p \vee q$
$p \vee q$ isn't a horn clause

→ $p \to q$ isn't but we can write it as: $\neg p \vee q$ which is ~~Horn~~ Horn
↓
if there is a q whether $p \to q$ is Horn or not, say yes as $\neg p \vee q$ is Horn.

* $\boxed{p \to q \equiv \neg p \vee q}$

p	q	$p \to q$	$\neg p$	$\neg p \vee q$
F	F	T	T	T
F	T	T ~~F~~	T	T
T	F	F	F	F
T	T	T	F	T

CNF
DNF

* ~~Satisfiable~~: true for atleast one value.

* S_1 entails S_2 : wherever S_1 is true, S_2 must also be true.

Eg. S_1 : $x \geq 10$ $\quad x \geq 10$

S_2 : $x \geq 9$ $\quad x \geq 9$

$R \to \sim R$

$\sim R \vee \sim R$

$\sim R$

Q $R \to \neg R$

This is satisfiable.

R	$\neg R$	$R \to \neg R$
T	F	F
F	T	T

$(S \wedge W) \wedge (S \wedge \sim S)$

$(S \wedge W) \wedge F$

$= F$

Q. $S \wedge (W \wedge \neg S) \equiv (S \wedge W) \wedge (S \wedge \sim S) \equiv (S \wedge W) \wedge F \equiv F$

S	W	$\neg S$	$W \wedge \neg S$	$S \wedge (W \wedge \neg S)$
F	F			
F				

$\therefore$ unsatisfiable / contradiction

Proof: to prove a fact (or infer sentences) from knowledge base

everything that is provable is true

→ Forward Chain: [bottom to top]

Sound → if all sentences generated are correct

complete → can generate all facts from knowledge base

- based on the rule Modes ponens

if ~~we know that~~ $A \to B$ and A is always True, then B is true. → $\boxed{\text{if } \frac{A \to B \ \& \ A}{B}}$

if everything that is true has a proof

* complete.

$A \to B \quad A = T$

$B = T$

* Sound:

can only generate some, not all facts

- FC isn't complete, but is sound

if written in Horn clause, and ~~proof~~ sentences consists of single letter then FC will be complete.

- difficult if KB is huge as we have to prove more facts

Q. Apply FC.

Knowledge base:

$P \rightarrow Q$

$L \wedge M \rightarrow P$

$B \wedge L \rightarrow M$

$A \wedge P \rightarrow L$

$A \wedge B \rightarrow L$

~~A → B~~

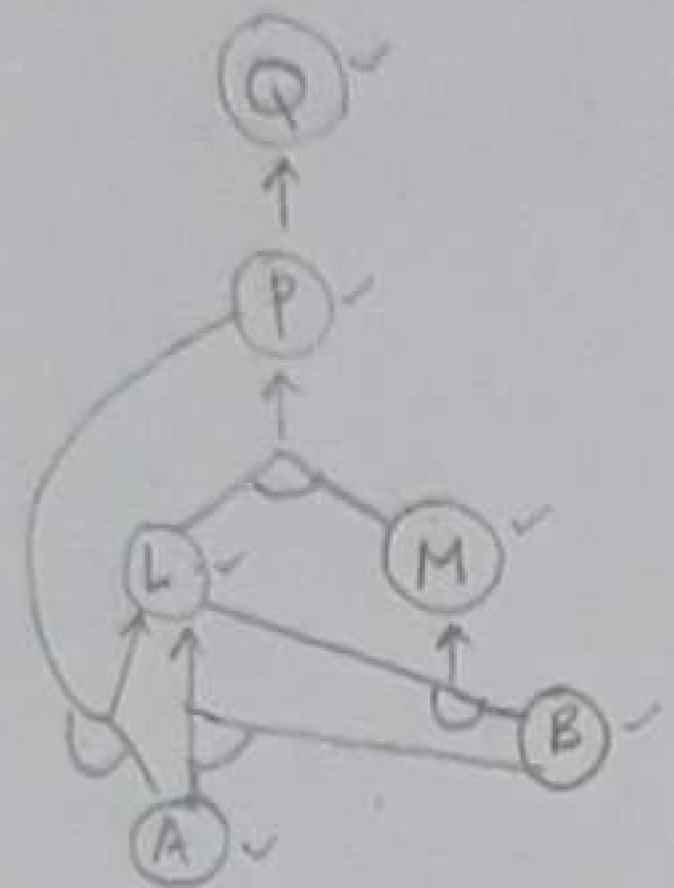
Database:

$\left.\begin{matrix} A \\ B \end{matrix}\right\} T$

Prove: Q is True.

Yes

✓ mean T



Basically we backtrack → we start from A & B, as we know they are T, then we work our way up to Q and prove that is T

Q. Knowledge base:

$A \wedge C \rightarrow F$

$A \wedge E \rightarrow G$

$B \rightarrow E$

$G \rightarrow D$

DB:

A

B

E

G

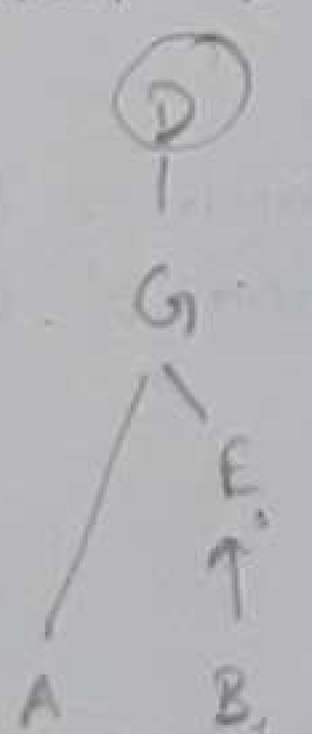
From modus ponens:

$B \rightarrow E$ & $B \Rightarrow E$ is T

$A \wedge E \rightarrow G$ & A & E $\Rightarrow$ G is T

$G \rightarrow D$ & G $\Rightarrow$ D is T

Goal: D is true?



~~A ∧ C → F~~

$B \rightarrow E \quad B$

$A \wedge E \rightarrow \underline{G}$

$G \rightarrow \underline{\underline{D}}$

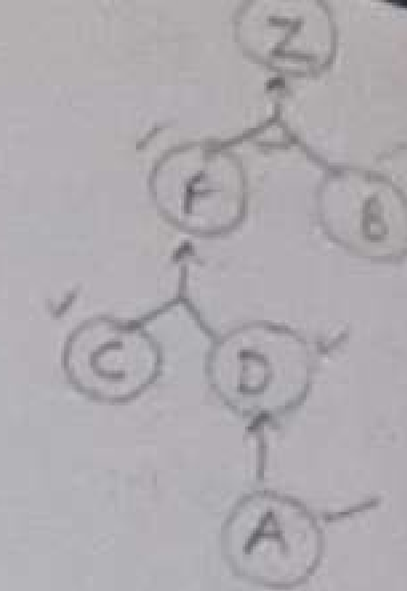
propositional calculus

predicate logic

Q. $F \land B \to Z$ $\quad$ A, B, C are T

$C \land D \to F$ $\quad$ Goal : Z

$A \to D$



Using Modus ponens: $A \to D$ & $A \Rightarrow D$ is T

$C \land D \to F$ & $C, D \to F$ is T

$F \land B \to Z$ & $F, B \longrightarrow Z$ is T

→ Backward Chain: [top to bottom]

18.10.2022

→ Resolution:

- Requires KB to be CNF

Eg: we can write $B \Leftrightarrow P_1 \lor P_2$ as:

$$(B \to (P_1 \lor P_2)) \land ((P_1 \lor P_2) \to B)$$
$$\sim (\neg B \lor (P_1 \lor P_2)) \land (\neg (P_1 \lor P_2) \lor B)$$
$$\sim (\neg B \lor (P_1 \lor P_2)) \land ((\neg P_1 \land \neg P_2) \lor B)$$
$$\sim (\neg B \lor (P_1 \lor P_2)) \land ((\neg P_1 \lor B) \land (\neg P_2 \lor B))$$

which is in the form of CNF

Eg: $(A \lor B) \Leftrightarrow (C \lor D)$

$$\sim ((A \lor B) \to (C \lor D)) \land ((C \lor D) \to (A \lor B))$$
$$\sim (\neg (A \lor B) \lor (C \lor D)) \land (\neg (C \lor D) \lor (A \lor B))$$
$$\sim ((\neg A \land \neg B) \lor (C \lor D)) \land ((\neg C \land \neg D) \lor (A \lor B))$$
$$\sim \{(\neg A \lor (C \lor D)) \land (\neg B \lor (C \lor D))\} \land \{(\neg C \lor (A \lor B)) \land (\neg D \lor (A \lor B))\}$$

which is in CNF form

- uses the rule: $\boxed{\dfrac{(P \vee \alpha) \quad (\neg P \vee \beta)}{(\alpha \vee \beta)}}$ } both are true implies either α or β is T
- sound, but not complete.
- uses proof by contradiction

$$\frac{P \vee \alpha \qquad \sim P \vee \beta}{\alpha \vee \beta}$$

Q. if W goes J will go
if W doesn't go, J will still go.
Will J go?

W: W will go
J: J will go

$W \to J$
$\neg W \to J$
J?

goal is to reduce to empty str { }

Assume $\neg J$, then: $(\neg W \vee J)$
$(W \vee J)$
$\neg J$

Applying the rule:

$$\frac{(W \vee J) \quad \neg J}{W} \qquad \frac{(\neg W \vee J) \quad \neg J}{\neg W} \qquad \frac{\neg W \quad W}{\{\ \}}$$

$\therefore$ assumption is wrong.
$\neg(\neg J)$ is true
J is true

$$\frac{\dfrac{\dfrac{\sim J \qquad W \vee J}{W} \qquad \sim W \vee J}{J} \qquad \sim J}{\{\ \}}$$

Q. If the unicorn is mythical, then it's immortal.
If it isn't mythical, it is a mammal.
~~If it's either mythical, or ma then it i~~
~~immortal or~~
If it's either immortal or ~~mythical~~ mammal, then it is horned

M : mythical
I : immortal
A : mammal
H : horned

$M \to I$
$\neg M \to A$
$(I \vee A) \to H$

Prove H is true

$M \to I$	$\neg M \to A$	$(I \vee A) \to H$
$\cong \neg M \vee I$	$\cong M \vee A$	$\cong \neg(I \vee A) \vee H$
		$\cong (\neg I \wedge \neg A) \vee H \cong (\neg I \vee H) \wedge (\neg A \vee H)$

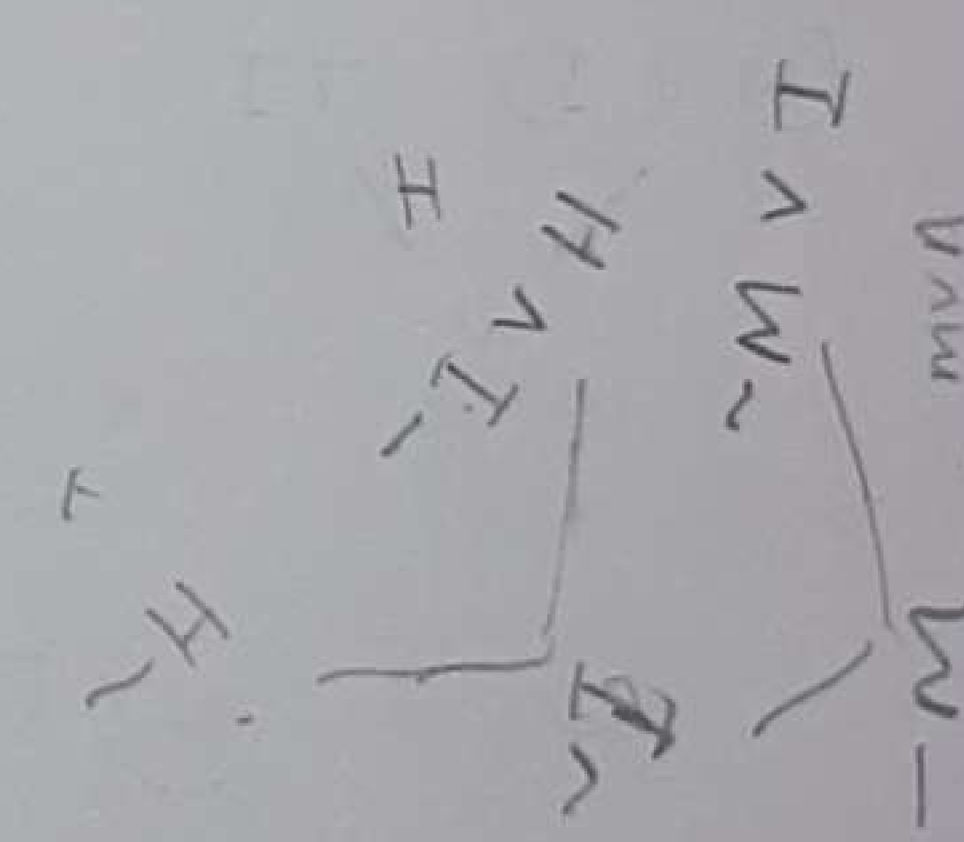
Assume H isn't true.

Facts: $\neg H$ $(\neg I \vee H)$ $(\neg A \vee H)$ $(\neg M \vee I)$ $(M \vee A)$

$\neg I$ $\neg A$ $(I \vee A)$

I

$\{\}$



20.10.2022

$\forall_x$ $\exists_x$

Predicate Logic (FOL) 1st order logic

who, its any action

constants, variables, predicates, functions, connectors, quantifiers

$\forall_x$ - for all x
$\exists_x$ - there exists a x
$\vee$ - 'or' can be used as 'some' as well

Eg: Parrot is a bird. → bird (parrot)

Parrot → Subject (like a const)
bird → predicate
bird (parrot) → how we represent in First Order Logic (FOL)

Q. Hari is the father of Aby.

Father (Hari, Aby)

* main connective for $\forall$ is →
* main connective for $\exists$ is $\wedge$

Q. All students write the exam.

$\forall_x$ Student (x) → Write (x, Exam)

Q. All birds can fly.

$\forall_x$ Bird (x) → Fly (x)

Q. Some boys are intelligent. this is eq to Not all boys are intelligent

$\exists_x$ Boy (x) $\wedge$ Intelligent (x).

Q. Not all cars have ~~car break~~ carberators.

$\exists_x$ Car (x) $\wedge$ ~Carberators (x)

or we can write

~($\forall_x$ Car (x) → Carberator (x))
~($\forall_x$ ~car (x) $\vee$ carberator (x))
$\exists_x$ ~(~ car (x) $\vee$ carberator (x))
$\exists_x$ car (x) $\wedge$ ~carberator (x)

Q. Every connected and rooted graph is a tree.

$\forall_x$ [connected (x) $\wedge$ rooted (x) $\wedge$ graph (x)] → tree (x)

Q. Every number is either negative or has a square root.

$\forall_x$ number (x) → (negative (x) $\wedge$ ~ squareroot (x))
$\vee$ (~negative (x) $\wedge$ sqrt (x)) } specify both options

25.10.2022

★ To remove existantial quantifier $\exists x$, we use <u>Skolemization</u>.

Eg: $\forall_x \exists_y (P(x) \vee Q(y))$

$\cancel{\forall_x}$ $\frac{g(x)}{y}$, $\forall_x\ P(x) \vee Q(g(x))$

if there is universal $\forall$ by $\exists$, replace y with a fn, else use a constant

Eg: $\exists_y (P(x) \vee Q(y))$

$\frac{a}{y}$ $P(x) \vee Q(a)$

Q. ~~If with~~ John likes all kinds of food
Apples and vegetables are foods.
Anything anyone eats and not killed is food
Anil eats peanuts and still alive.
All persons are alive iff they are not killed

P.T. John likes peanuts.

CNF
→ remove implication and double implication
→ ~~convert~~
→ remove $\exists$ if there
→ $\neg$ comes inside

$\forall_x$ food $(x) \rightarrow$ likes (John, x)

food (Apple) $\wedge$ food(vegetable)

$\forall_y \forall_z$ eats $(\underset{\text{person}}{y}, \underset{\text{food}}{z})$ $\wedge$ ~killed$(y) \rightarrow$ food (z)

anyone anything

eats (Anil, peanut) $\wedge$ alive (Anil)

$\forall_w$ alive $(w) \leftrightarrow$ ~ killed (w)

To prove: likes (John, peanuts)

$\leftrightarrow$

$\forall_x [\neg food(x) \vee likes(John, x)]$ — ①

$food(apple)$ — ②

$food(vegetable)$ — ③

$\forall_y \forall_z \neg [eats(y, z) \wedge \neg killed(y)] \vee food(z)$ — ④

$eats(Anil, peanuts)$ — ⑤

$alive(Anil)$ — ⑥

$\forall_w [alive(w) \rightarrow \neg killed(w)] \wedge [\neg killed(w) \rightarrow alive(w)]$

$\cong \forall_w (\neg alive(w) \vee \neg killed(w)) \wedge \cancel{\neg}(killed(w) \overset{\vee}{\cancel{\wedge}} \cancel{\neg} alive(w))$ — ⑦

$\cancel{\forall_x} (\neg food(x) \vee likes(John, x))$

$food(apple)$

$food(vegetable)$

$\cancel{\forall_y \forall_z} (\neg eats(y,z) \vee killed(y) \vee food(z))$

$eats(Anil, peanuts)$

$alive(Anil)$

$\cancel{\forall_w} (\neg alive(w) \vee \neg killed(w)) \wedge (killed(w) \vee alive(w))$

Assume that John doesn't like peanuts, i.e. $\neg$ likes(John, peanuts)

$\neg likes(John, peanuts)$ $\quad$ $\neg food(x) \vee likes(John, x)$

$\neg food(peanuts)$ $\quad$ $\neg eats(y, z) \vee killed(y) \vee food(z)$

$eats(Anil, peanuts)$ $\quad$ $\neg eats(y, peanuts) \vee killed(y)$

$killed(Anil)$ $\quad$ $\neg alive(w) \vee \neg killed(w)$

$\neg alive(Anil)$ $\quad$ $alive(Anil)$

$\{\}$

As we got $\{\}$
$\therefore$ John likes peanuts is true.

Q. Some patients like all doctors'
No patients like any Quack.

we can't use $\rightarrow$ with $\exists_x$

Prove by resolution: No doctor is a quack.

$\exists_x$ patient$(x) \wedge [\forall_y$ doctor$(y) \rightarrow$ likes$(x, y)]$

$\forall_x$ patient$(x) \rightarrow$ ~~likes(x, quack)~~ $[\forall_z$ quack$(z) \rightarrow \neg$ likes$(x, z)]$

P.T: $\forall_w$ doctor$(w) \rightarrow \neg$ quack(w).

$\exists_x$ patient$(x) \wedge [\forall_y \neg$ doctor$(y) \vee$ likes$(x, y)]$

$\forall_x \neg$patient$(x) \vee [\forall_z$ quack$(z) \rightarrow \neg$ likes$(x, z)]$

$\sim \forall_x \neg$ patient$(x) \vee [\forall_z \neg$ quack$(z) \vee \neg$ likes$(x, z)]$

$\forall_w \neg$ doctor$(w) \vee \neg$ quack(w).

$\exists_x \forall_y$ patient(x)

$\exists_x$ patient$(x) \wedge [\forall_y \neg$ doctor$(y) \vee$ likes$(x, y)]$

$\exists_x \forall_y$ patient$(x) \wedge [\neg$ doctor$(y) \vee$ likes$(x, y)]$

$\forall_y$ patient$(a) \wedge [\neg$ doctor$(y) \vee$ likes$(a, y)]$

patient$(a) \wedge (\neg$ doctor$(y) \vee$ likes$(a, y))$ —— ①

$\forall_x \neg$ patient$(x) \vee [\forall_z$ quack$(z) \rightarrow \neg$ likes$(x, z)]$

$\forall_x \neg$ patient$(x) \vee [\forall_z \neg$ quack$(z) \vee \neg$ likes$(x, z)]$

$\forall_x \forall_z, \neg$ patient$(x) \vee \neg$quack$(z) \vee \neg$likes(x, z)

$\neg$ patient$(x) \vee \neg$ quack$(z) \vee \neg$ likes(x, z) —— ②

$\cancel{\neg[\forall_w \neg doctor(w) \vee \neg quack(w)]}$

$\cancel{\exists_w\ doctor(w) \wedge quack(w)}$

$\forall_w\ \neg doctor(w) \vee \neg quack(w)$

$\neg[\forall_w\ \neg doctor(w) \vee \neg quack(w)]$

$\exists_w\ doctor(w) \wedge quack(w)$

$doctor(b) \wedge quack(b)$ — (3)

(4) (5)

$doctor(b) \qquad \neg doctor(y) \vee likes(a,y)$

$likes(a,b) \qquad \neg patient(x) \vee \neg quack(z) \vee \neg likes(x,z)$

$\neg patient(a) \vee \neg quack(b) \qquad quack(b)$

$\neg patient(a) \qquad patient(a)$

$\{\}$

$\therefore$ No doctor is a quack is true

31.10.22

Intelligent Agents:

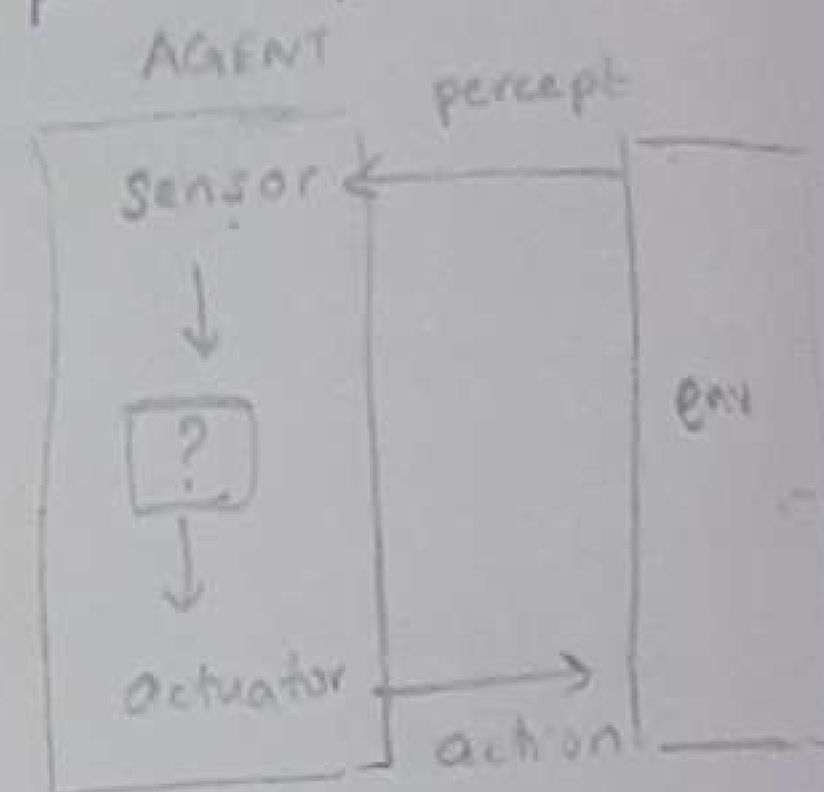
An agent is anything that can be viewed as ~~persuing~~ perceiving its environment through sensors and acting upon the environment through actuators.

Agents has sensors, it will sense the env and will take necessory i/p and it will fund the action to perform and inform the actuator

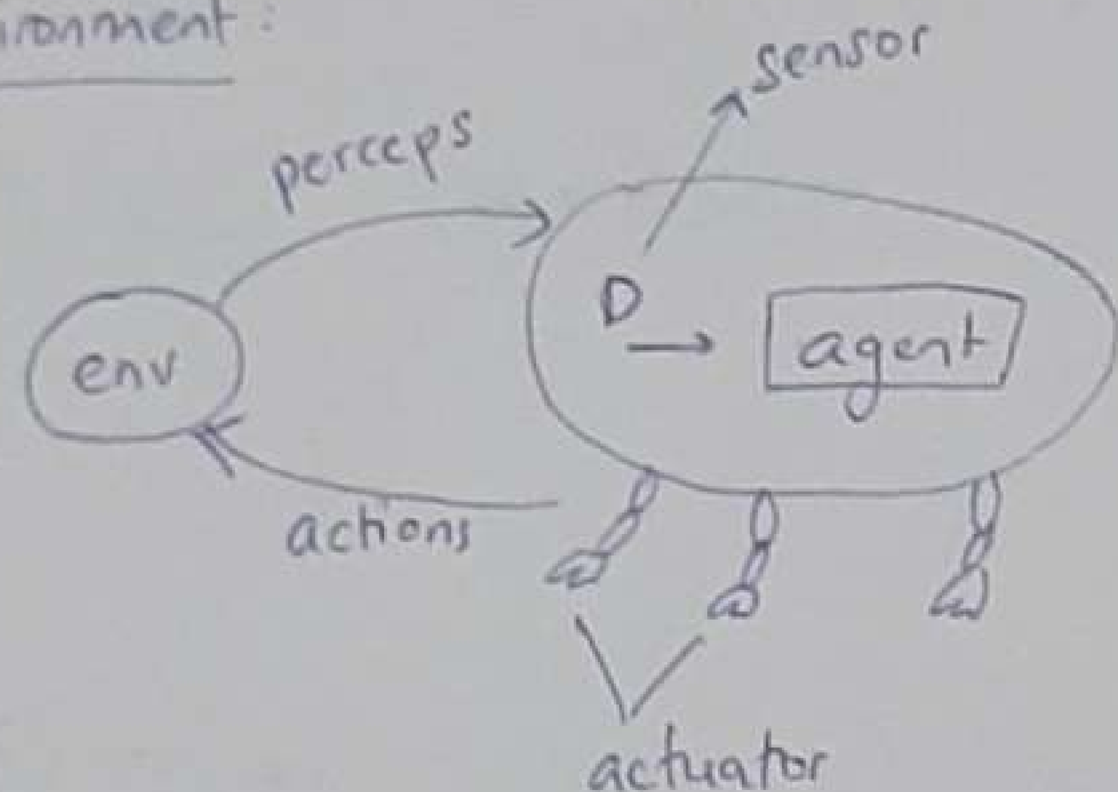
→ in humans: eye, ears ⇒ sensors
hand, leg, mouth ⇒ actuator

→ robotic agents: camera & infrared range finders for sensors
various motor for actuators

→ s/w agents: key strokes, file contents, n/w packets for sensors



Agent & Environment:



★ The agent function map from percept history to actions
$[f : P^* \rightarrow A]$

★ Agent program: ~~sen~~ run on physical architecture to produce fn
agent = ~~archetet~~ architecture + program

Vacuum Cleaner world

2 tiles that might be clear or dirty
We have 1 vacuum cleaner to clean them

Percepts: location and contents Eg [A, duty]
Action: left, right, suck, NoOp
Agent function: look up table

State space diagram: draw all possible states & transitions

Rational Agents

* Rationality of agents is based on:
  - performance measuring success
  - agents prior knowledge of env
  - actions that agent can perform
  - agents percept sequence to data

* Rational agent: For each possible percept seq. a rational agent should select an action that's expected to max. its performance measure.

Autonomy in Agents:

It is the extent to which its behavior is determined by its own expense rather than knowledge of the designer.

* Extremes:
  - No autonomy: ignore env/data
  - Complete autonomy: must act randomly

PEAS

Every agent requires:

- Performance measure: safe, fast, legal, ...
- Env: road, pedastrian, customer
- Actuator: accelerator, break, signal, ...
- sensors: camera, speedometer, GPS

Eg: Agent - part picking robot
  Performance measure - % of parts in correct bin
  Env - conveyor belt
  Actuator - jointed aim and hand
  Sensor - camera, joint angle sensor

Env Type:

- Full observable vs partially observable
- Single agent vs multi agent
- Determanistic vs ~~stoch~~ stochiastic
- ~~Epo~~ Episodic vs sequential
- Static vs dynamic
- Discrete vs Continuous

→ Fully Observable vs Partially

Is everything an agent require to choose its action available to it via its sensor?

If so, It is fully observable

Cross ~~word~~ word	Poker	Part Picking robot	Image Analysis	Taxi Driver
Fully	Partially	Fully	Fully	Partially

→ Single Agent vs Multi Agent:

Single Agent ↓ an agent operating by itself in an env

Multi Agent ↳ there are many agents working together
- competitive
- cooperative

Crossword	Pooker	Taxi driver	Part Picking robot	Image Analysis
single	multi	multi	Single	Single

→ Deterministic vs Stochasic

If env, after a particular action, is based on previous state and action taken by the agent then it's deterministic.

Cross ~~word~~ word	Poker	Taxi driver	Part Picking Driver	Image Analysis
D	S	S	S	D

→ Episodic vs Sequential

↓

if the next episode doesn't depend on the action that's taken in prev episode, then it's episodic.

Cross word	Poker	Taxi driver	Part Picking Robot	Image Analysis
S	S	S	E	E

→ Static vs Dynamic:

↓

doesn't change

* Semi dynamic: env won't change but agent's performance score does change

crossword	poker	Taxi driver	PPR	IA
S	S	D	D	Semi

→ Discrete vs Continuous

↓

a limited no. of distinct, clearly defined percepts and actions

a range of values.

crossword - D

poker - D

Agent Types

4 basic types in order of ↑ generality

- simple reflex agent
- model based reflex agent
- goal based agent
- utility based agent

} learning agent / not

→ Simple Reflex Agents

Simple but limited intelligence

Action doesn't depend on percept history, only on current percept

1.11.2022

Problem Searching

→ Uninformed search:

We don't know anything about the env (domain)

It's like brute force

→ Informed search.

Some domain knowledge is available. Based on it, we are trying to reach the goal.

↓ we have some knowledge

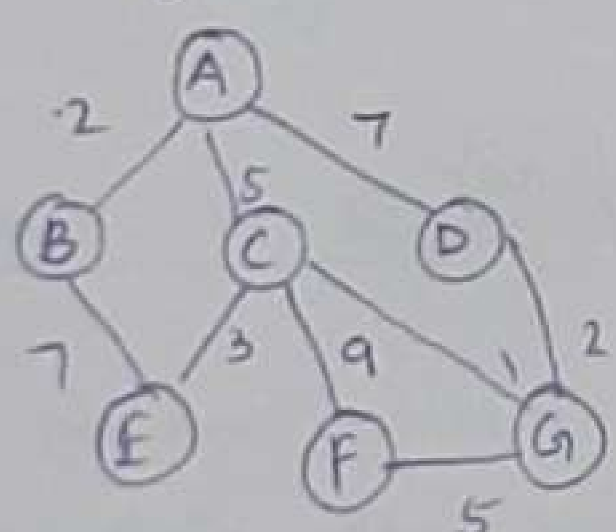
Properties of Searching Algorithms:

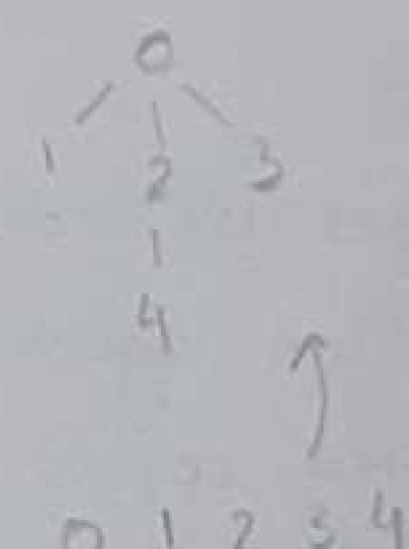
- completeness: a searching algo is said to be complete if it can find the sol if it exists.
- optimality: an algo is said to be optimal if it will find the best sol

Uninformed Searching Techniques: BFS, DFS, DLS, Bidirectional search, uniform cost search

1. BFS

Expanding breadth first



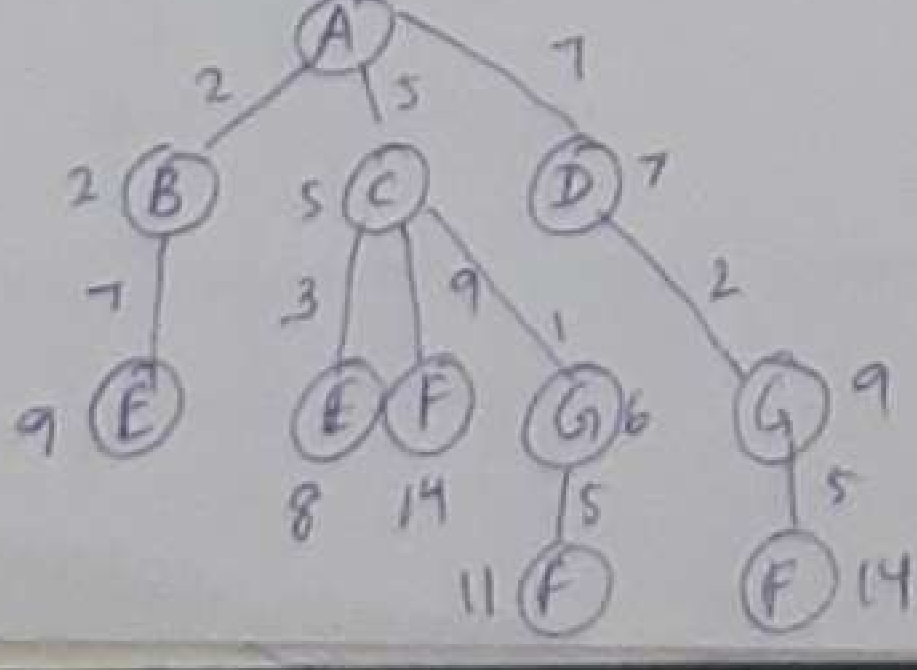


path from A to F

A B C D E F

Try to find path by expanding breadth first.

* After expanding a node in level 0, it will go to level 1 and visit all nodes and go down.

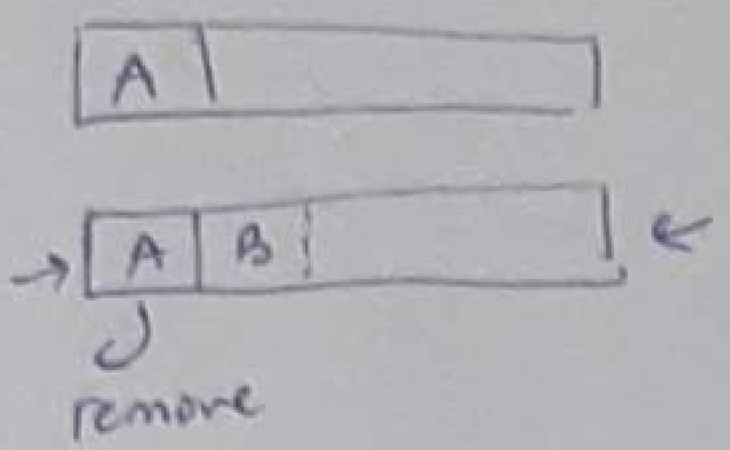


⟷ not optimal

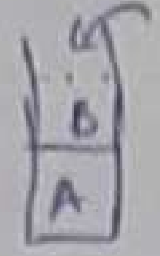
* BFS is optimal if it doesn't stop after reaching the goal.

it's complete

Queue - FIFO



Stack - LIFO



Add A: A

Remove A: B C D

Remove B: C D E

Remove C: D E F G

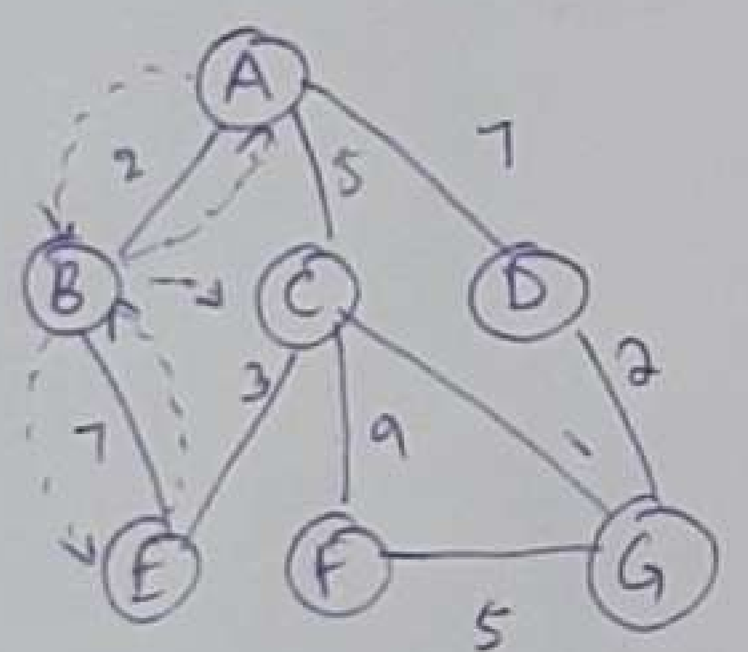
Remove D: E F G

Remove E: F G

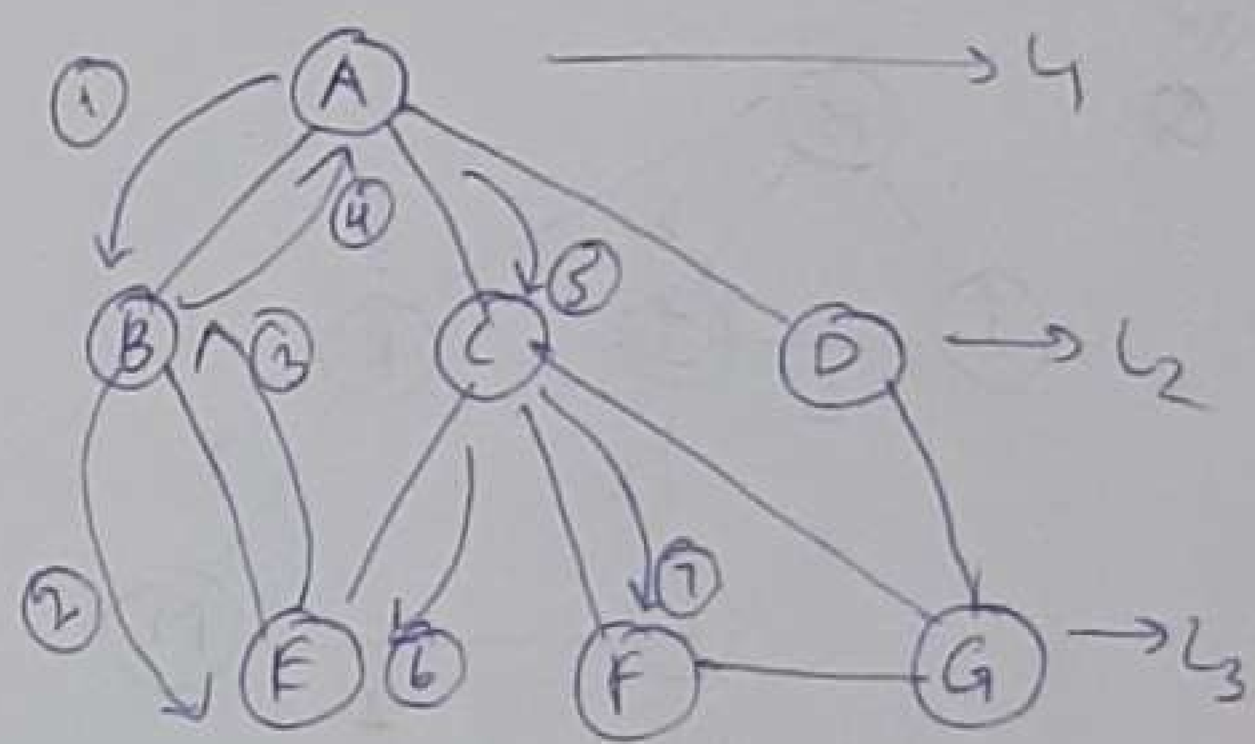
Remove F: G

2. <u>DFS</u> complete, not optimal

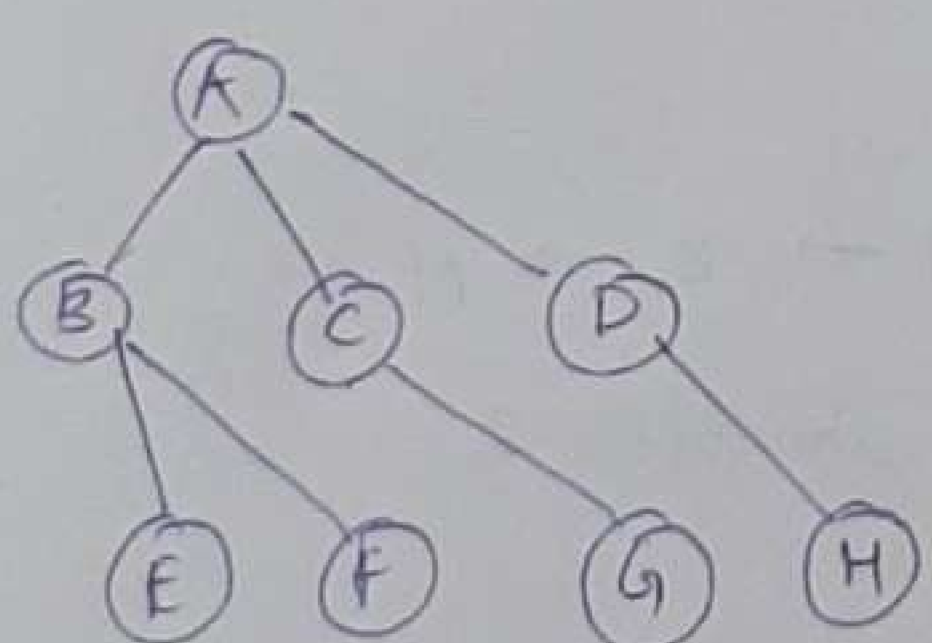
Expanding Depth 1st



Goal F:



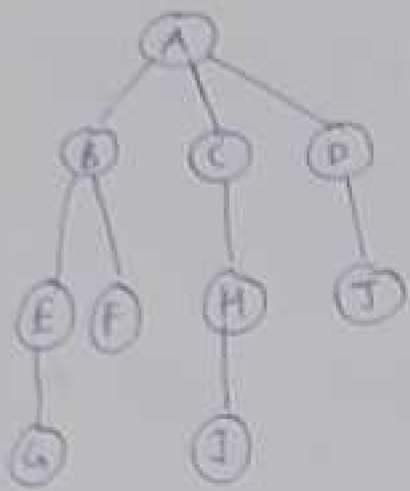
Q. Goal G:



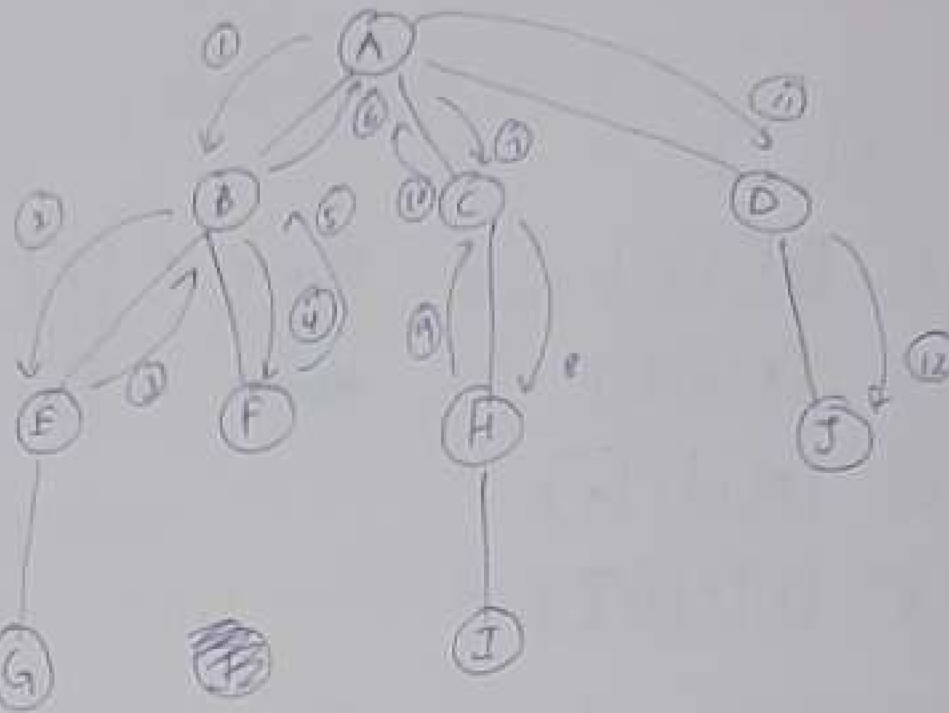
A B E F C G

* DFS can be implemented using stack.

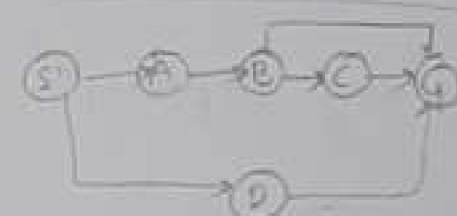
3. Deapth Limited Search



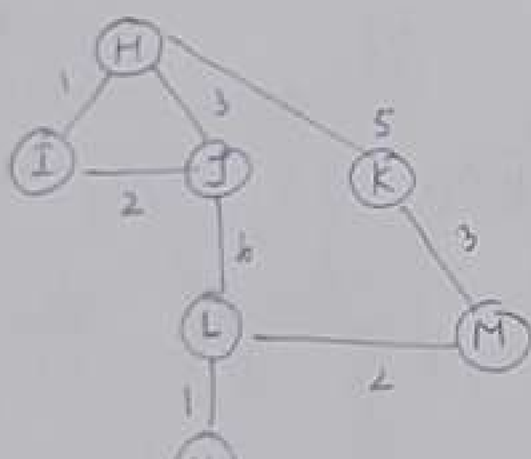
depth limit = 2



- it's not a complete searching technique
- not optimal

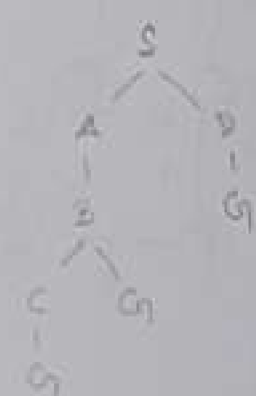


HW
Q

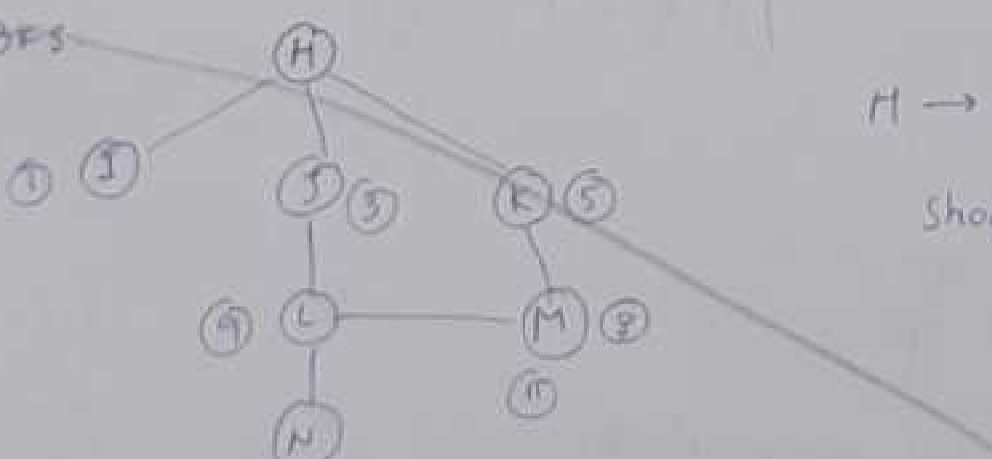


Goal : M

traversal path = S→D→G



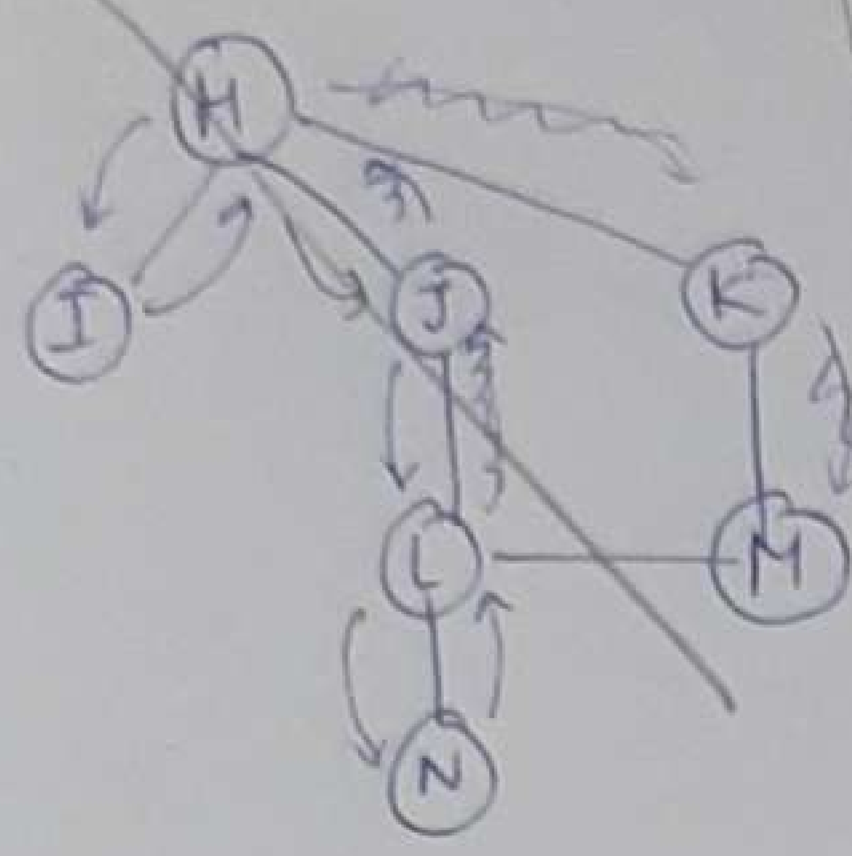
BFS

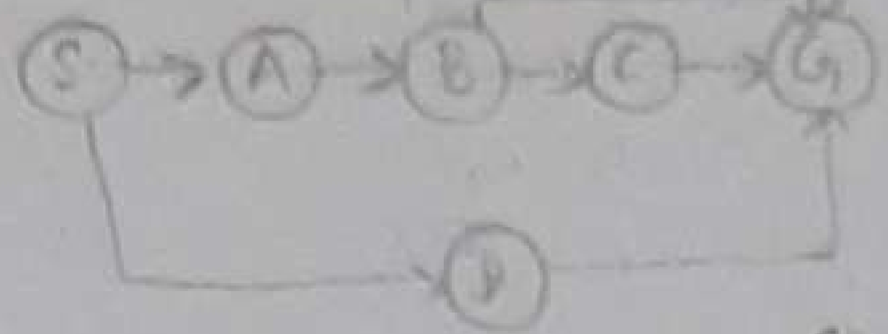


H → k → M

Shortest : 8

DFS:

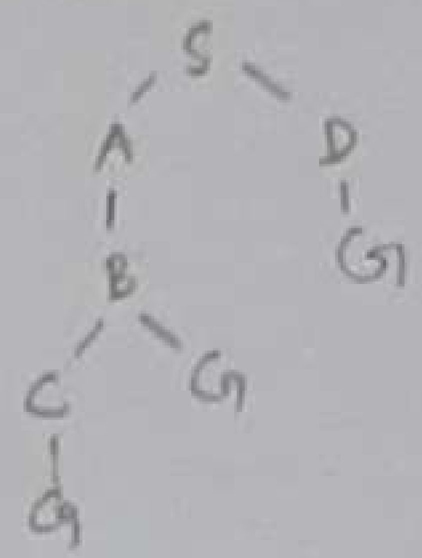




Another Q

~~G~~
~~C~~
B
A
S

search tree

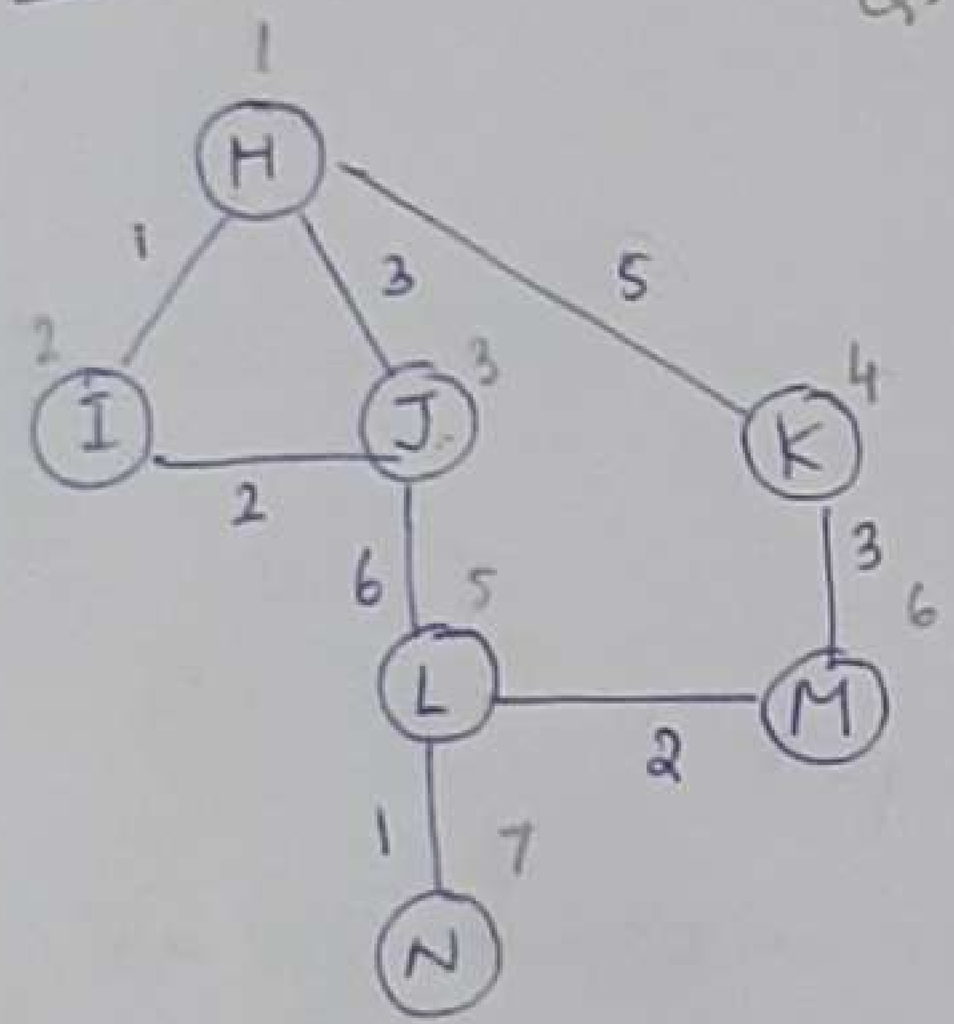


traversal

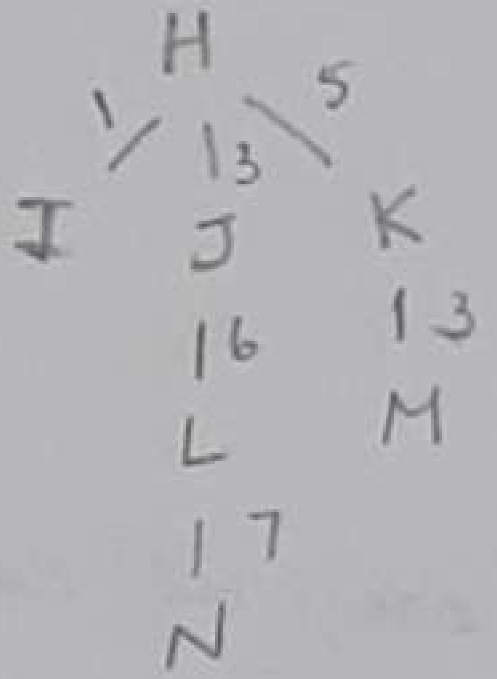
S
|
A
|
B
|
C
|
G

path = S → A → B → C → G

BFS:



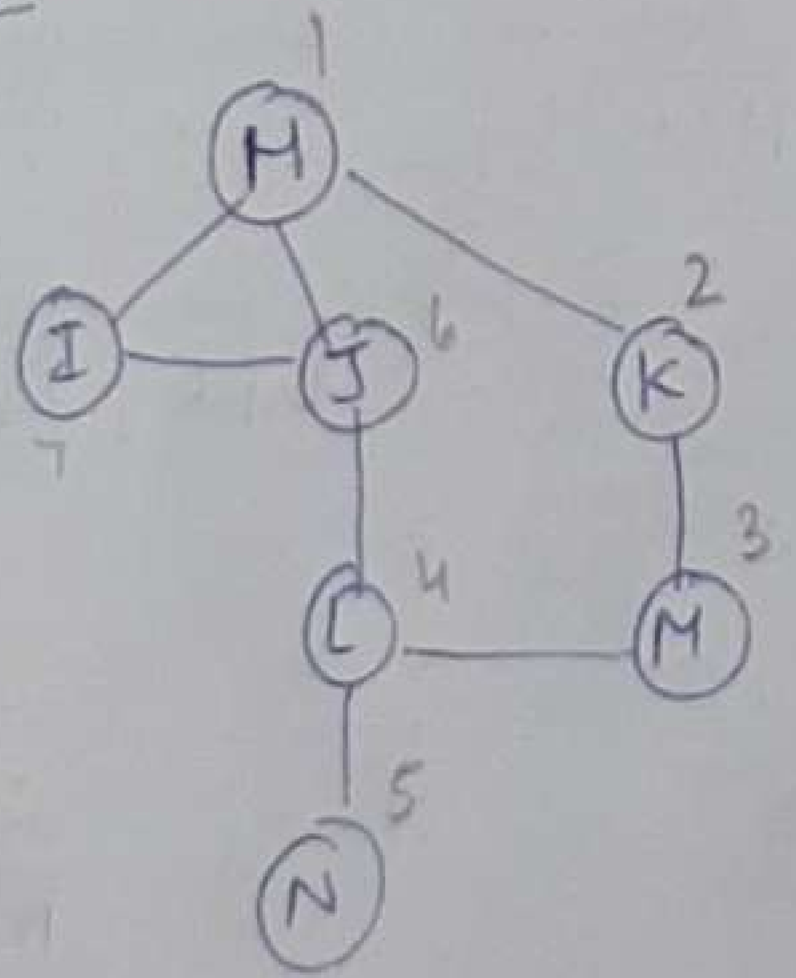
Q. | ~~H~~ | ~~I~~ | ~~J~~ | ~~K~~ | ~~L~~ | ~~M~~ | ~~N~~ |



∴ Shortest path to M:

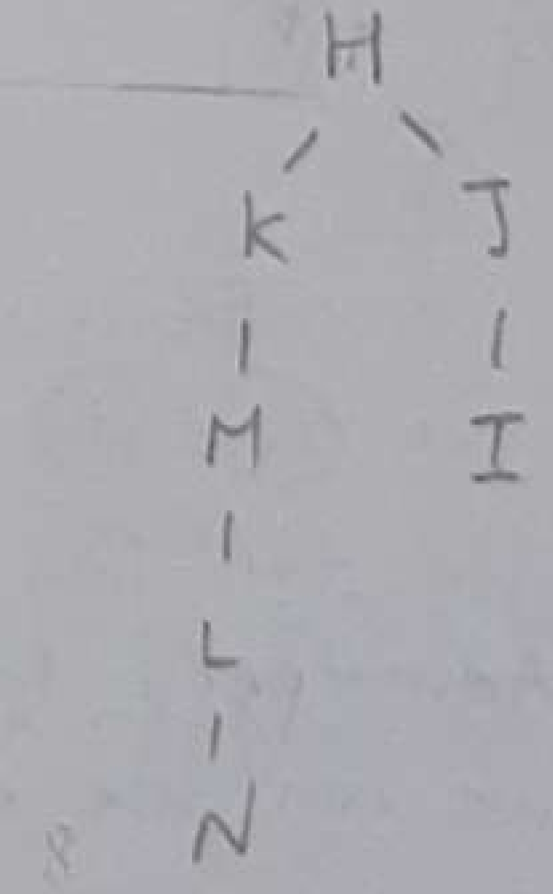
H → K → M = 8

DFS:



~~N~~
~~L~~
~~M~~
~~K~~
~~J~~
~~I~~
~~H~~

H K M L N J I



∴ H - K - M

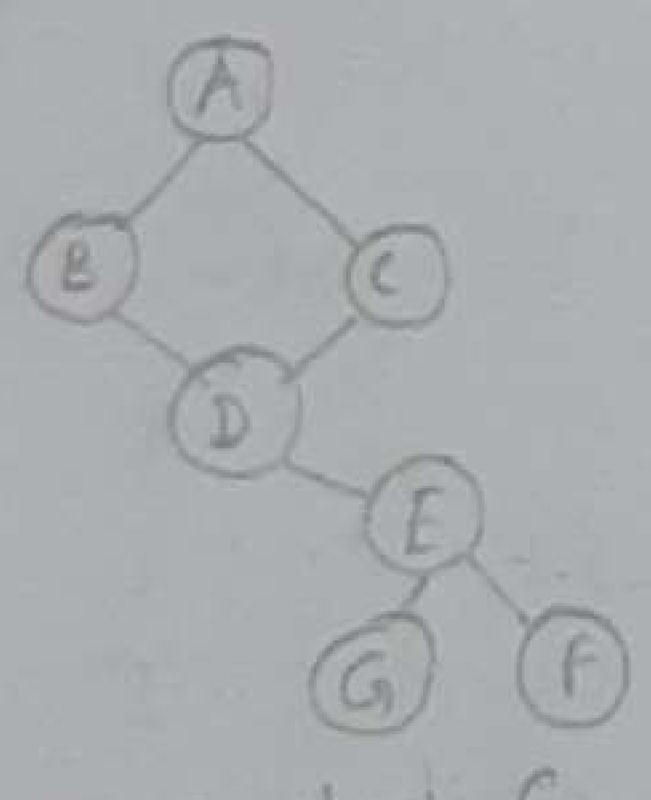
8

DLS

3.11.2022

Bidirectional Search:

- we know source and sink, but not the path



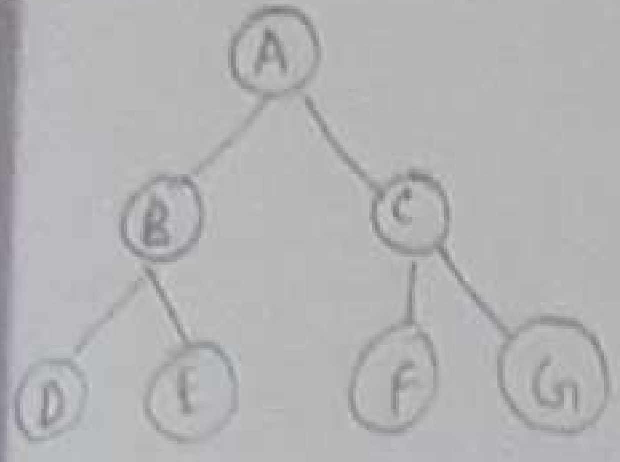
we want to find the path b/w A & G } prev we knew the src and had to find both sink and path ↓ now we just have to find path.

- we start from both src & sink, and work in parallel till we meet at a common node
- Advantage: time reduces
- we can use either BFS or DFS

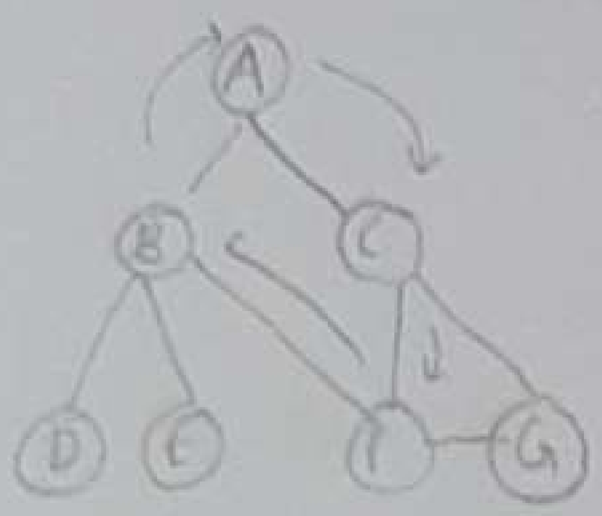
DFS: A B D
G E D } Eg. ⇒ path: $\boxed{\text{A B D E G}}$

- complete if BFS is used. Isn't complete if DFS is used.

Algo will continue to run until a common node is reached. [Doesn't matter if path is found, common node ↓ has to be found]



complete in both DFS & BFS

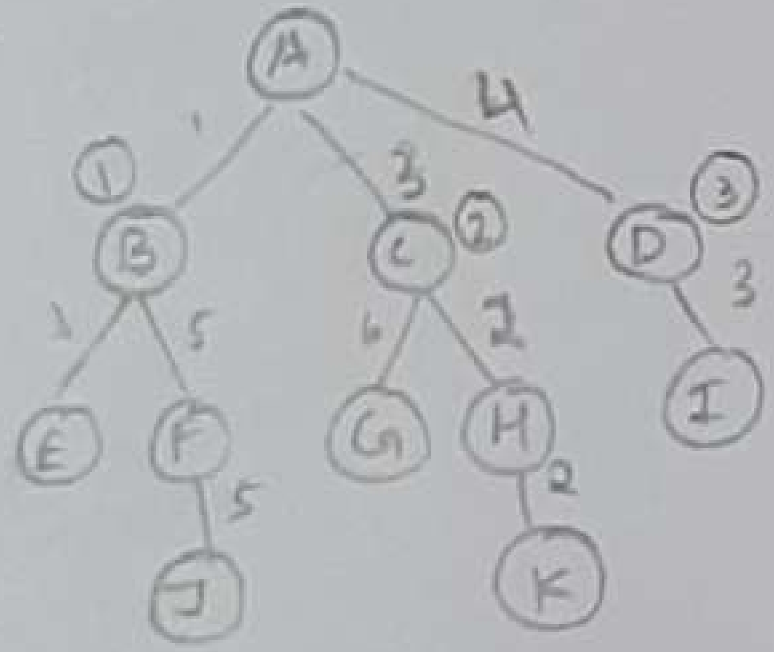


DFS: doesn't find common node

* in the case of 2 nodes, common node won't be there

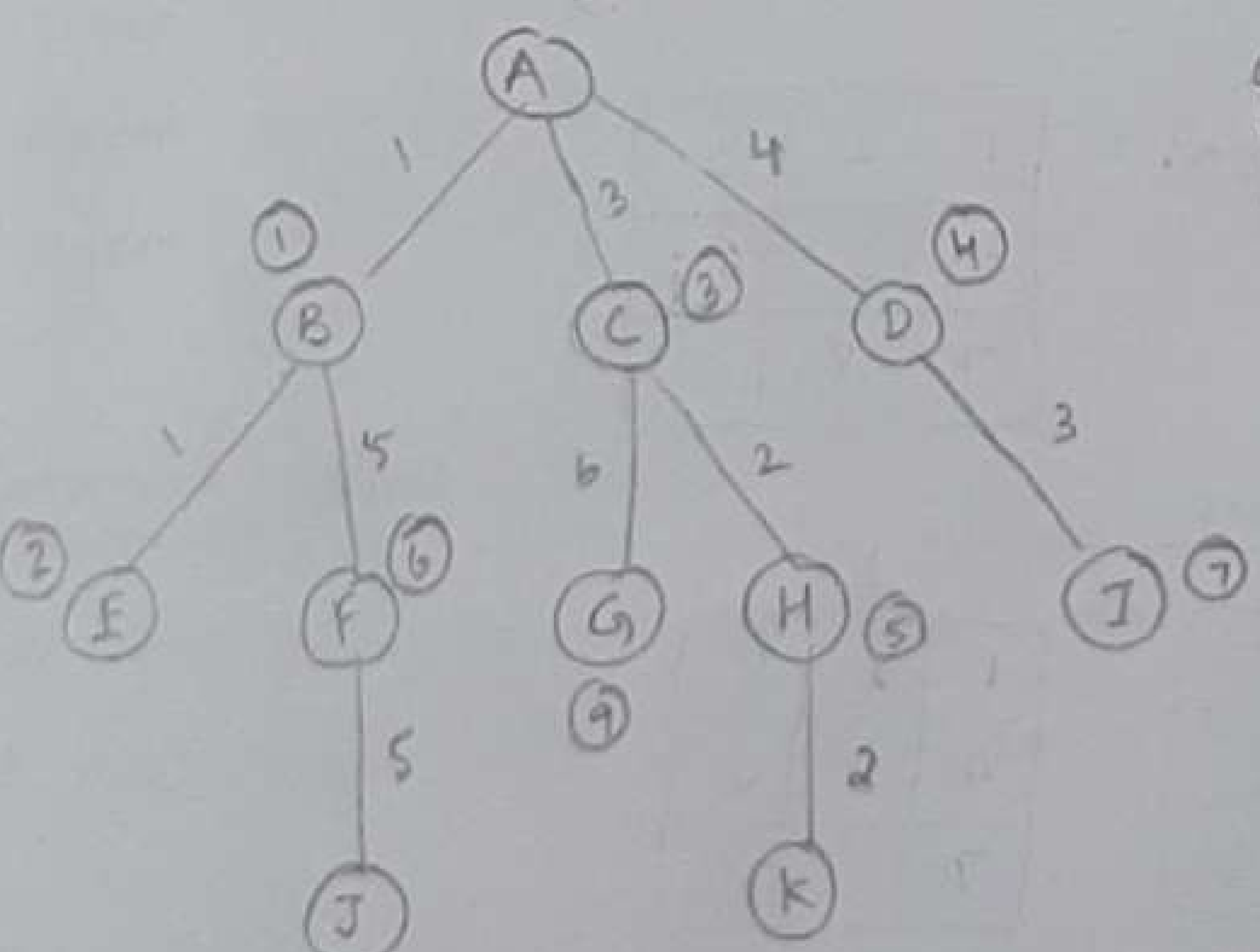
follows Greedy Approach

Uniform Cost Search



goal: G

We go through the nodes with least cost. (ascending order)



go in the order of least cost

- start with A
- check its adj nodes
  - ↳ go in the order of least cost, i.e B, C, D
- expand B 1st
  - ↳ least cost = E 2nd
  - ↳ E has no node
- expand C 3rd
- expand D 4th
- expand H 5th
- expand F 6th
- nothing to check ↵

when we check all nodes

* op!

8.11.2022

Uninformed Search:

We need to say: (to define a problem)

1. initial state
2. actions
3. transitions [what will be the next state]
4. Goal list
5. Cost

Puzzle Problem:

1	5	6
7	3	2
4	7	

→

1	2	3
4	5	6
7	8	

⇑
initial state

actions L, R, U, D (all have the same cost)

cannot use uniform cost search as ↑

we can use BFS, DFS

→ using BFS:

initial state =

1	2	5
4		6
7	8	3

goal =

1	2	3
4	5	6
7	8	

4 possible moves

moving L

1	2	5
	4	6
7	8	3

L

1	2	5
4	6	
7	8	3

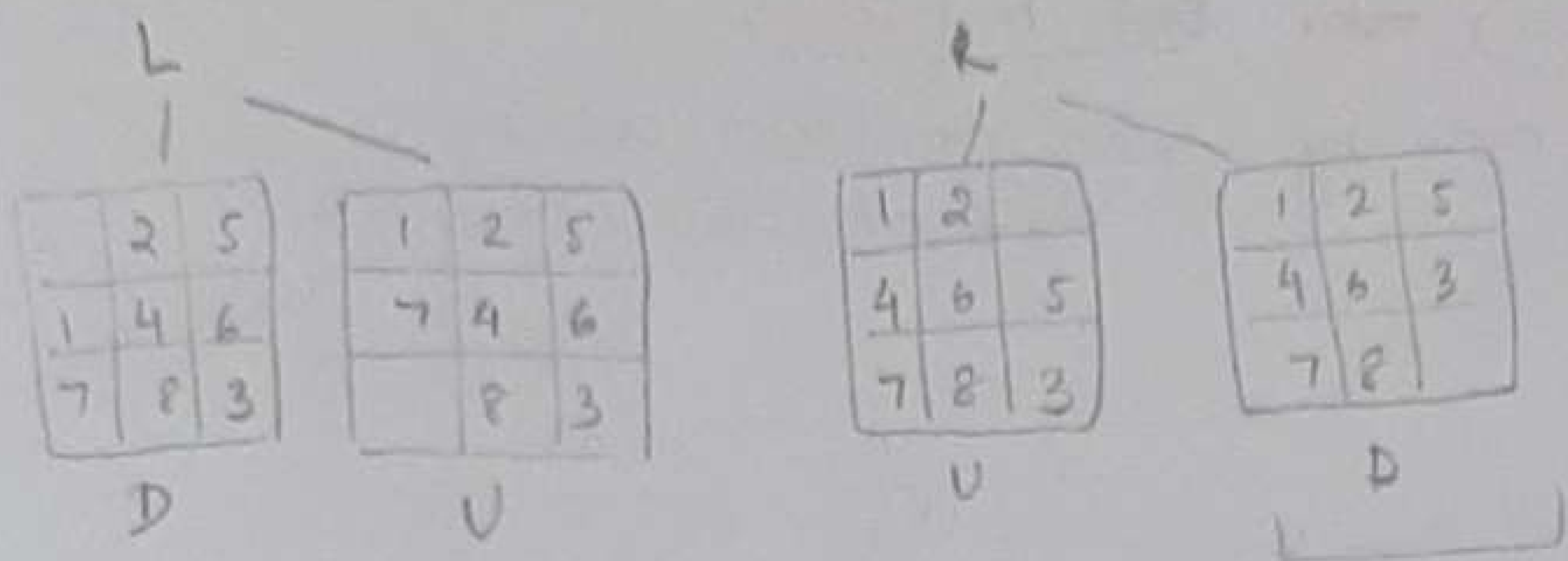
R

1		5
4	2	6
7	8	3

D

1	2	5
4	8	6
7		3

U



if this was the goal, then the actions (path) = R, D

* for such Q, do DLS, not DFS as there is a good chance that it won't stop as we might not reach the goal

8 Queens Problem:

initial state: empty chessboard

goal: place 8 q so that no 2 q attack each other

we have an agent which perform DFS and BFS

10.11.2022

Initial state:

- We have an empty chessboard
- place 8 queen so that no queen will attack each other
- agent will do BFS or DFS
- agent will give the actions

Informed Search BeFS, A*

We have some info about where the goal node is.

At each point, ISA will calculate a heuristic function in each search.

Take a cost, S.

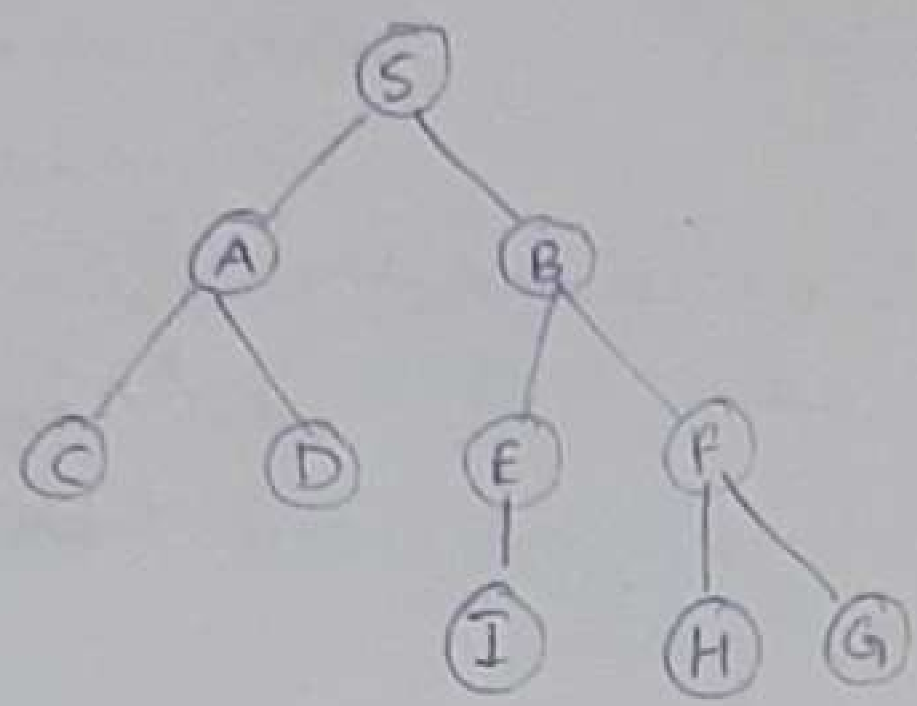
estimated cost > actual cost (hf value) $h(f) > a(f)$

All informed searching algo are based on the heuristic function.

→ BFS (informed search) aka Best First Search

we will find the next node by selecting the lowest cost node. (select path with least heuristic value)

Q.



n	h(n)
S	13
A	12
B	4
C	7
D	3
E	8
F	2
G	0
H	4
I	9

Find the best path from S → G

- It will maintain 2 lists

→ open list ⇒ close list

- place start node in open list
- Start 1st iteration - take a start node

Open	close	
S		
AB	S	A = 12, B = 4 ✓
AEF	SB	A = 12, E = 8, F = 2 ✓
AE IG	SBF	A = 12, E = 8, I = 9, G = 0
	SBFG	← goal = G

∴ path = S B F G

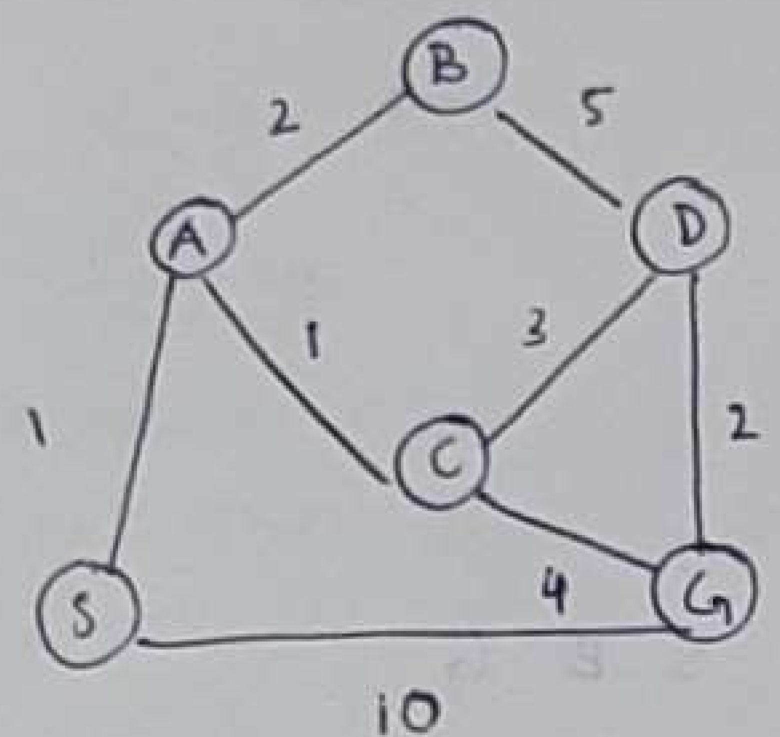
→ A^* (informed search)

- A^* is a variant of BFS
- Select a node with least $f(n)$
- $f(n) = g(n) + h(n)$

$g(n)$ ↓ actual cost to search from source node to n

$h(n)$ ↳ heuristic value of n

Q.

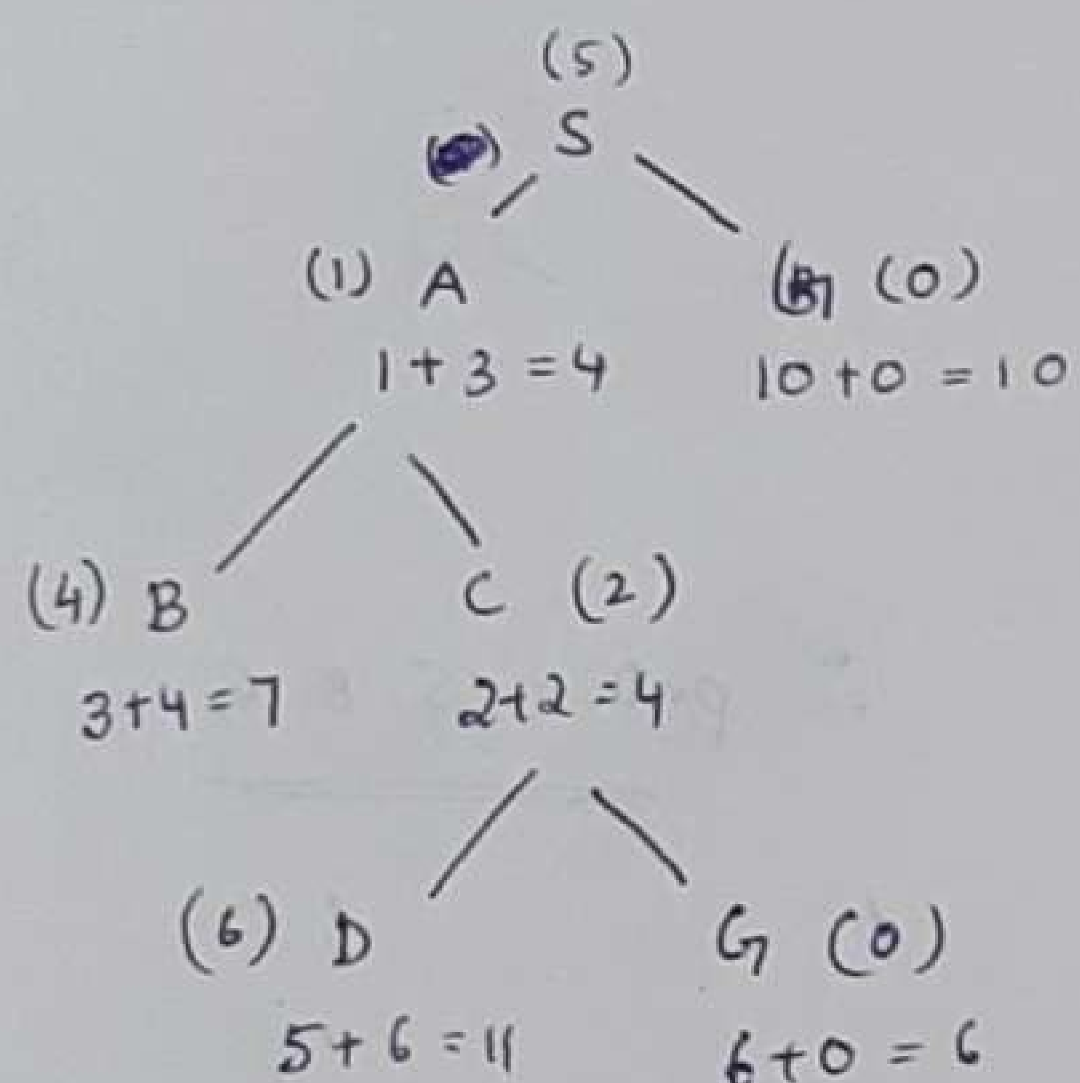


n	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

source = S

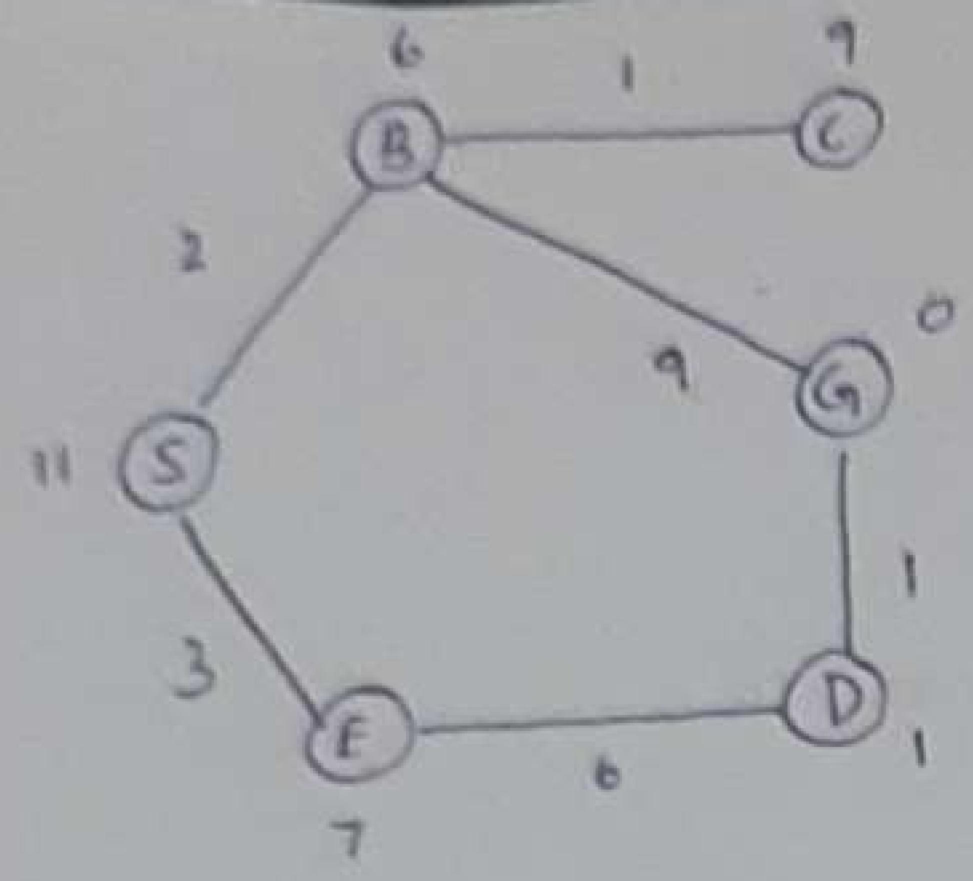
goal = G

At each point, A^* algo will select the node with min $f(n)$ value.



∴ path = S A C G

Q.



n	h(n)
S	11
B	6
C	9
G	0
E	7
D	1

complete, optimal (based on turistic fn)

Source = S, Goal = G

A*

(11) S

(6) B E (7)

$2+6=8$ $3+7=10$

(9) C G (0)

$3+9=12$ $11+0=11$

$2+9$

$\therefore$ path = S B G

Best FS

open	close
S	
BE	S
CGE	SB
CE	SBG

B = 6, E = 7

C = 9, G = 0, E = 7

$\therefore$ path = S B G

28 11.2022

Problem Searching:

→ Informed Search:

- heuristic function
- BFS : chooses the node with min $h(n)$

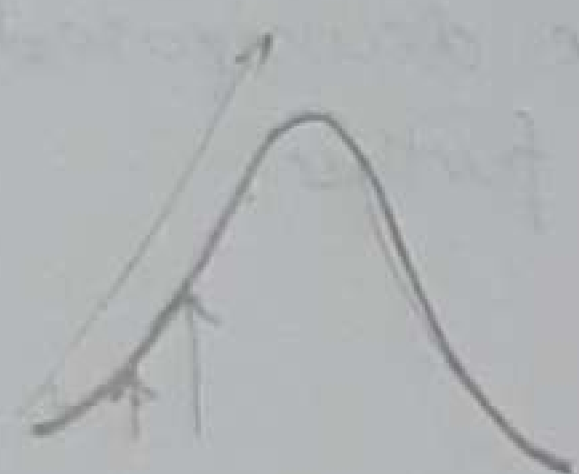
↳ problem: it doesn't consider the actual cost b/w 2 nodes.

A^* : $g(n) + h(n)$

Hill Climbing:

it chooses x as cur state with $h = a$

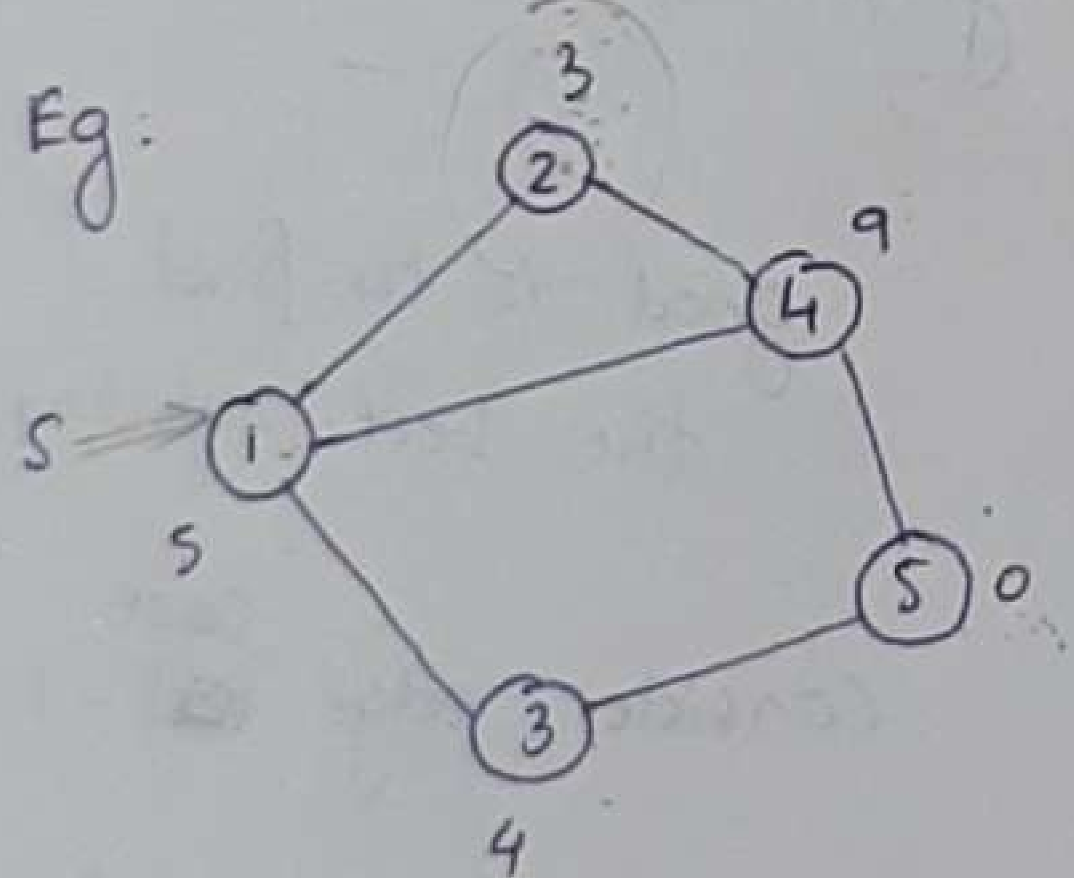
it will then choose a state whose h is better than a



problem: we can have local minima.

Eg

Eg:



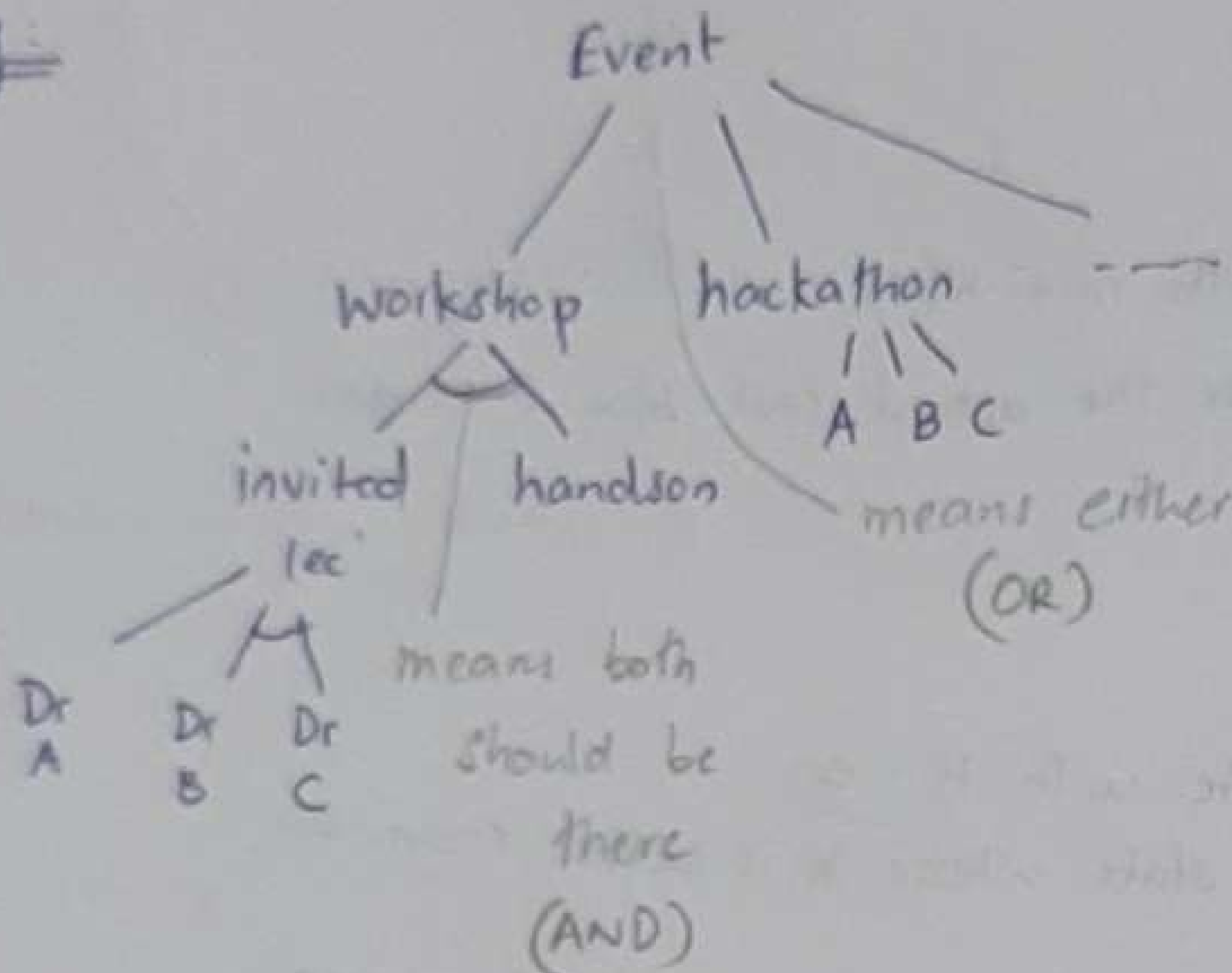
From ① which has 5, it chooses ② with 3. But there is nothing near ② which has a better 'a' even though there is a node ⑤ with cost 0

→ local minima

Problem Decomposition:

We can decompose a problem into subproblems.

Eg:



This graph is called **AND-OR Graph**

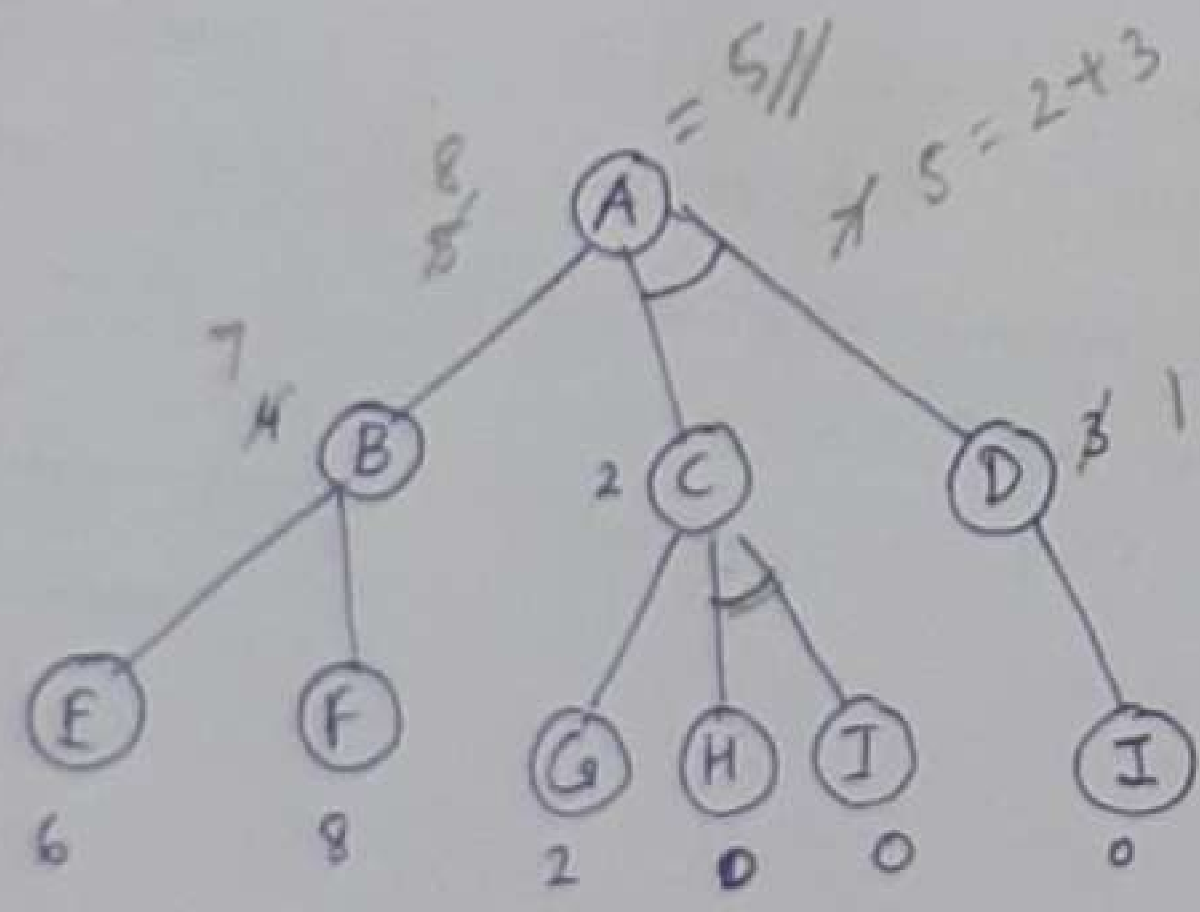
it represents prob decomposition.

* We decompose until the prob cannot be decomposed further.

Goal: to find the best ~~one~~ solution.

Algo to find:

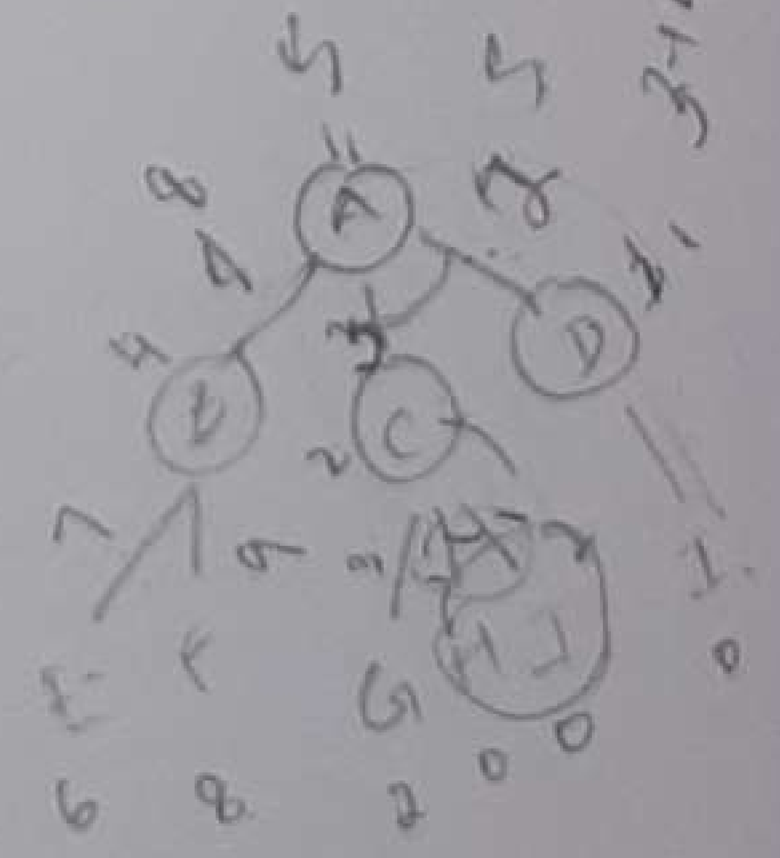
AO* [used for prob searching in AND-OR graphs]



goal is to find the best solution.

Consider edge cost = 1

$g(n) + h(n)$



Add A to open list.

$g(n) + h(n)$

open

~~A~~

close

B has better cost, choose it.

→ E has better cost

from A to B: $4+1=5$

A to CD: $3+4=7$

$2+1 \quad 3+1$

Update cost from A, each time

~~A — B — E~~ we update cost of its children

5

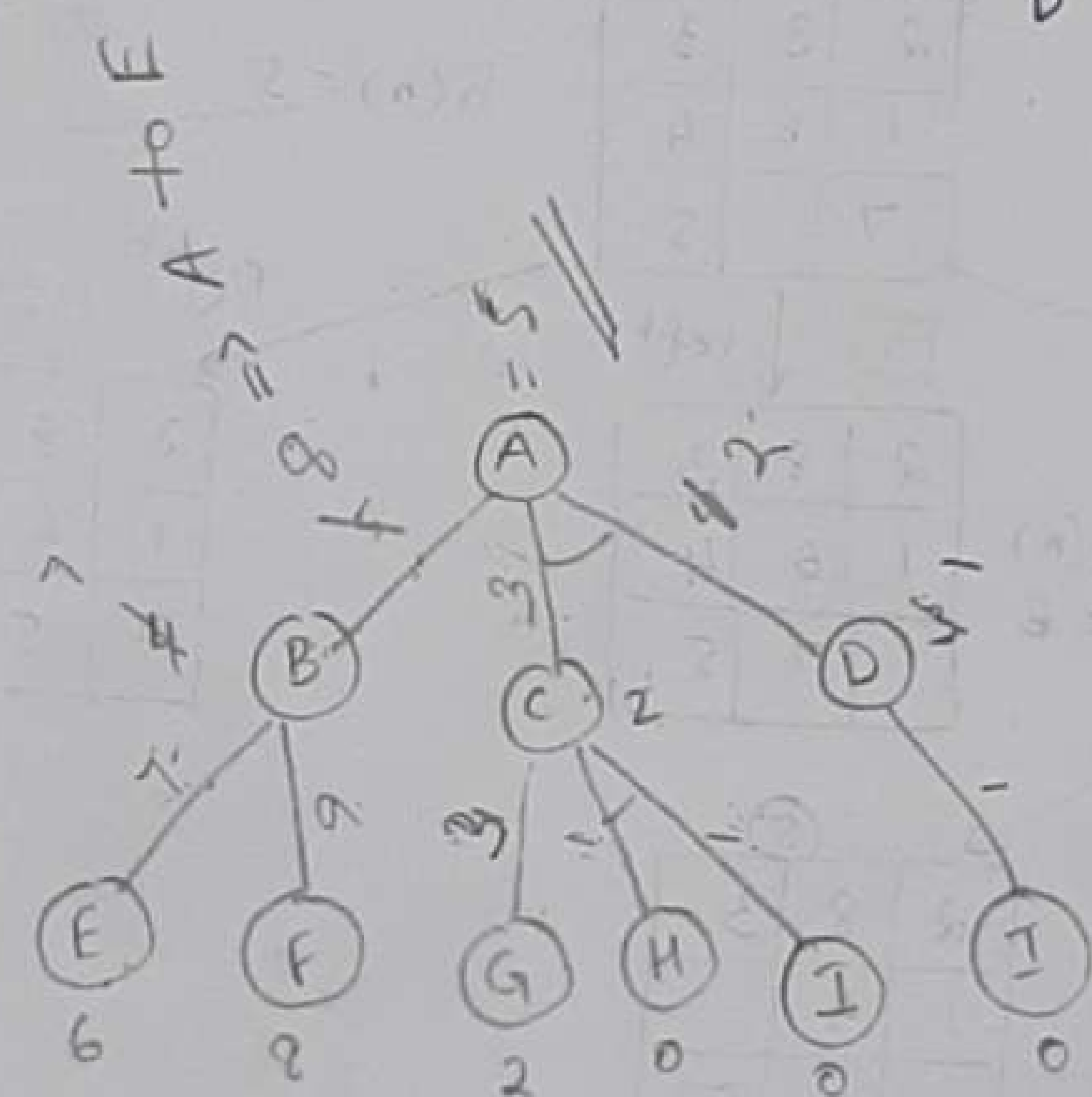
B to E : $6+1 = 7$

B to F : $8+1=9$

C to G = +3

C to HI = 1 + 1

D to I = 1



8 Puzzle Problem: [with heuristic]

initial state:

2	8	3
1	6	4
7		5

2, 2, 1

goal state:

1	2	3
8		4
7	6	5

without heuristics, can take a long time.

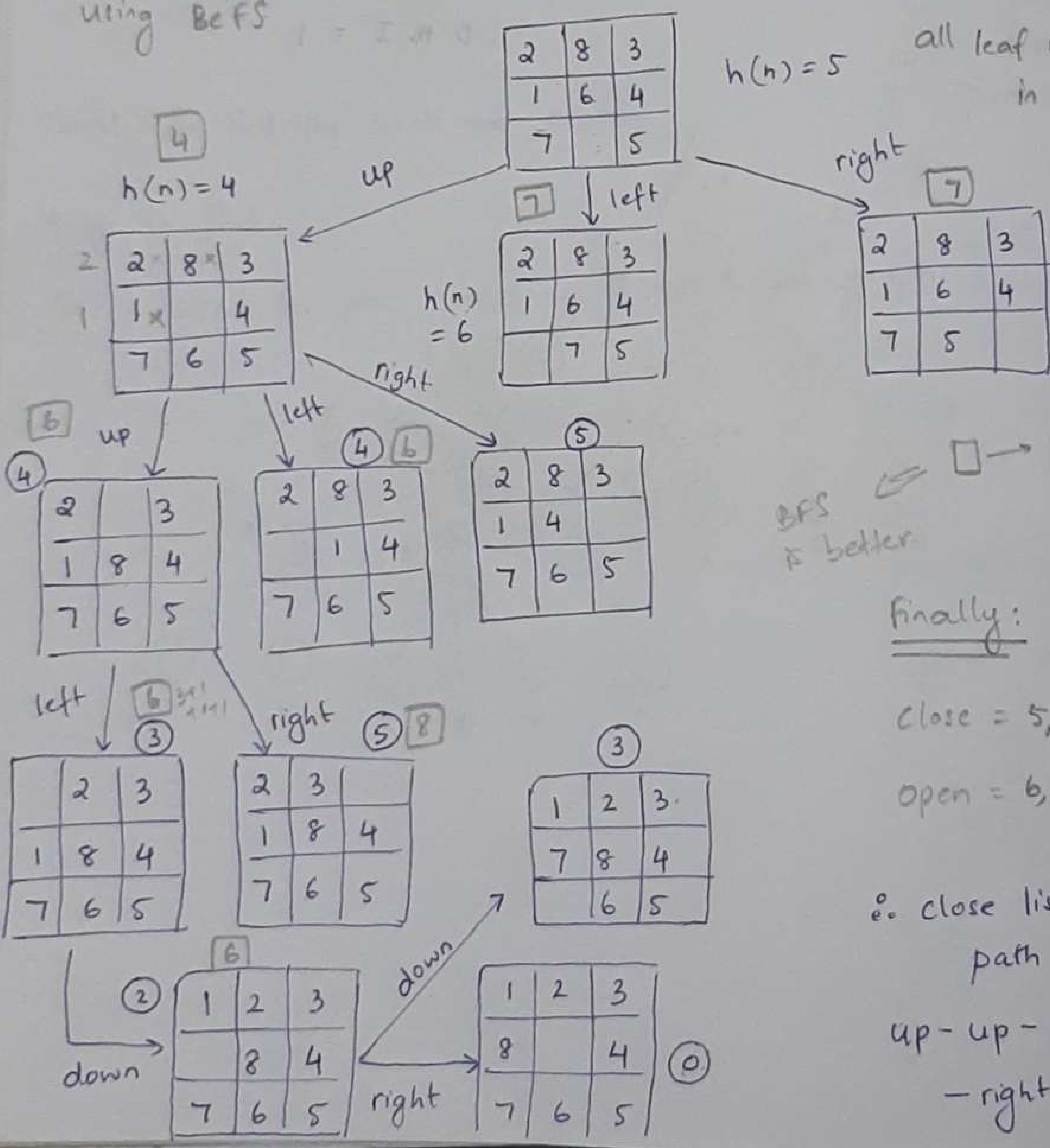
heuristic fn $h(n)$ = # of misplaced blocks.

→ for initial : $h(n) = 5$
goal : $h(n) = 0$

using BeFS

Whichever state we choose, move to closed list

all leaf nodes will be in open list



h(n) = 6.

□ → if A* was used instead of BeFS

BFS is better

Finally:

Close = 5, 4, 4, 3, 2, 0

Open = 6, 6, 4, 5, 5, 3

∴ close list is the path:

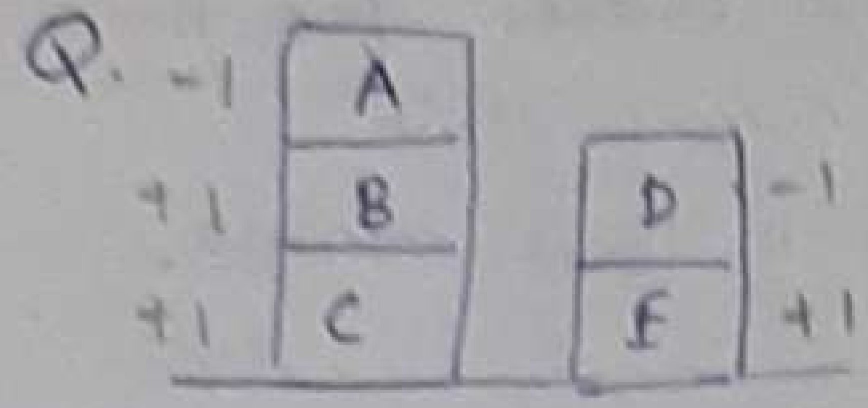
up - up - left - down - right

* Using the same approach for hill climbing, the algo chooses the next state whose h(n) is less than the h(n) of current state

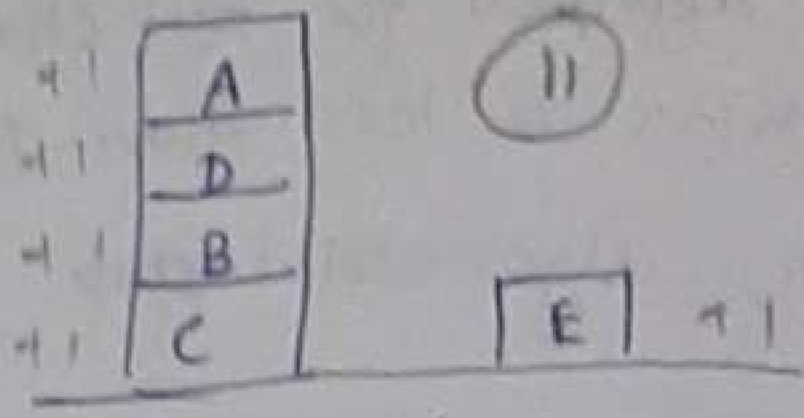
In this eg, algo will stop at 3, which is local minima.

using A^*

Q.



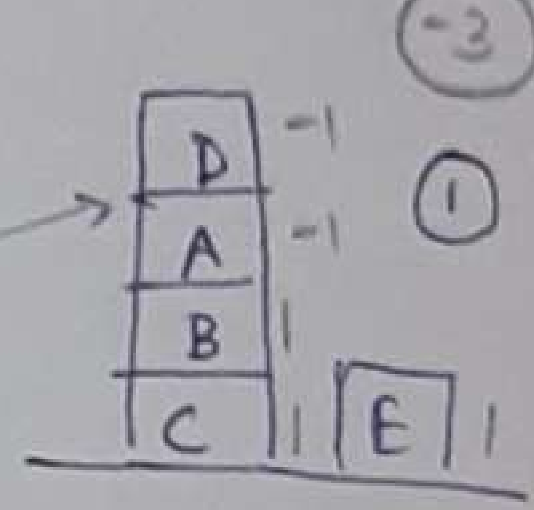
start ⓵
$h(n) = 1$

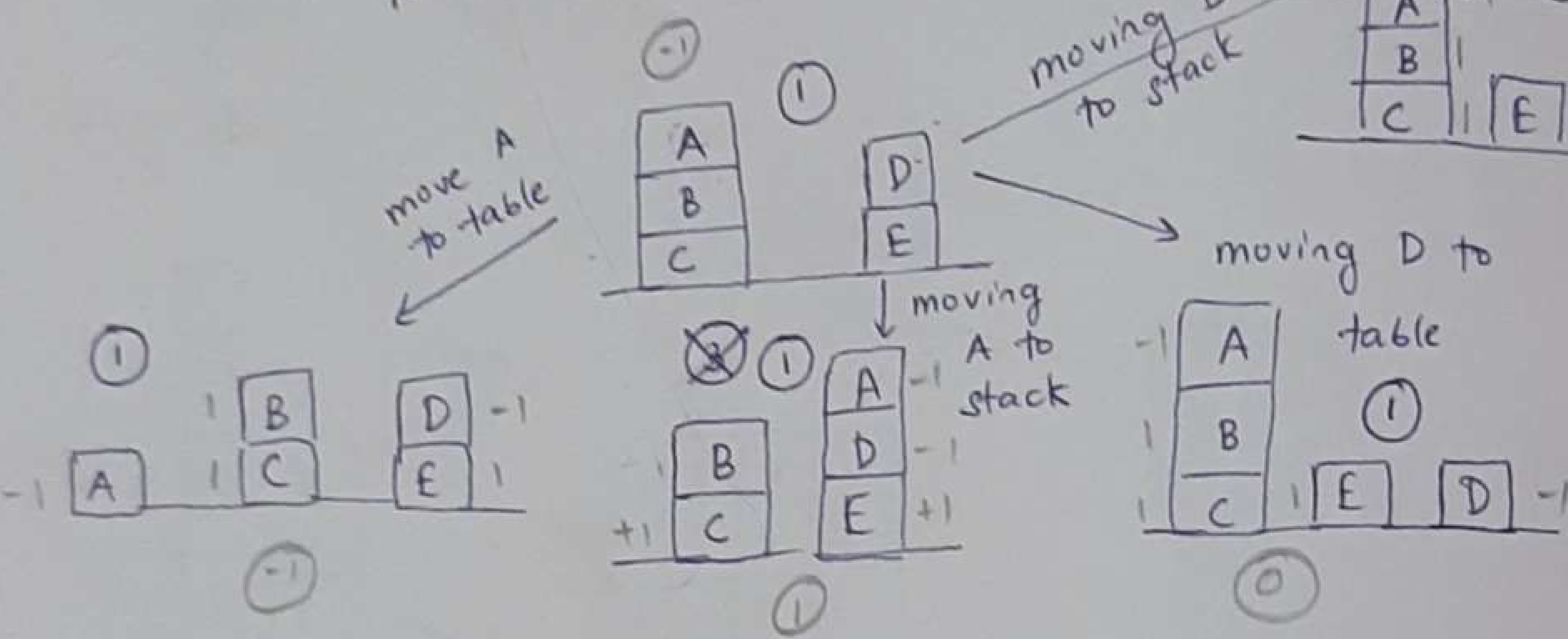


goal. $h(n) = 5$

action: we can move one block at time and place it either on top of stack or on table.

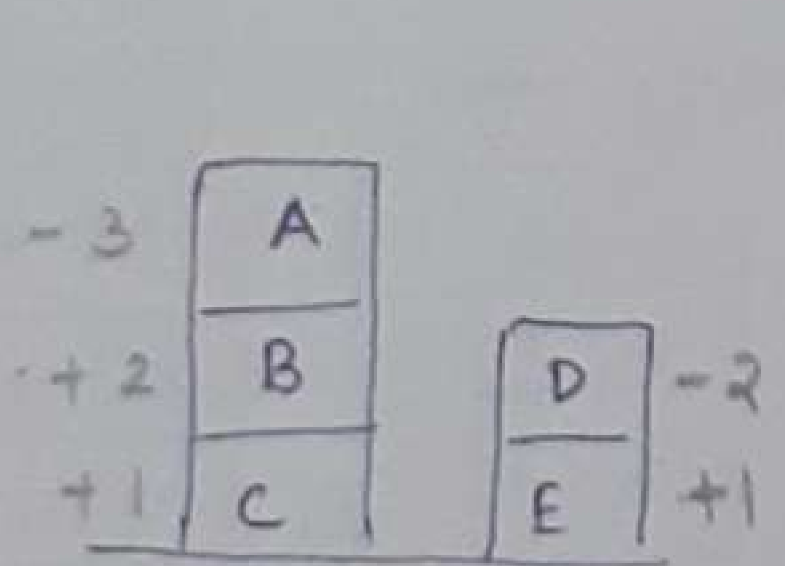
$h(n) = +1$ for all blocks placed correctly.
$\quad\quad\;\; -1$ for all blocks placed incorrectly



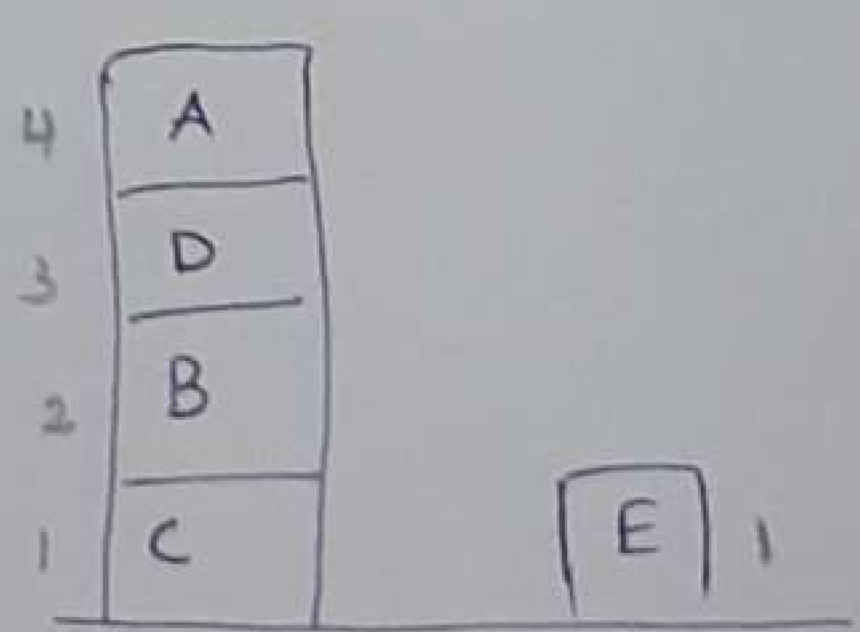


As all have same $h(n)$, we change the def of $h(n)$

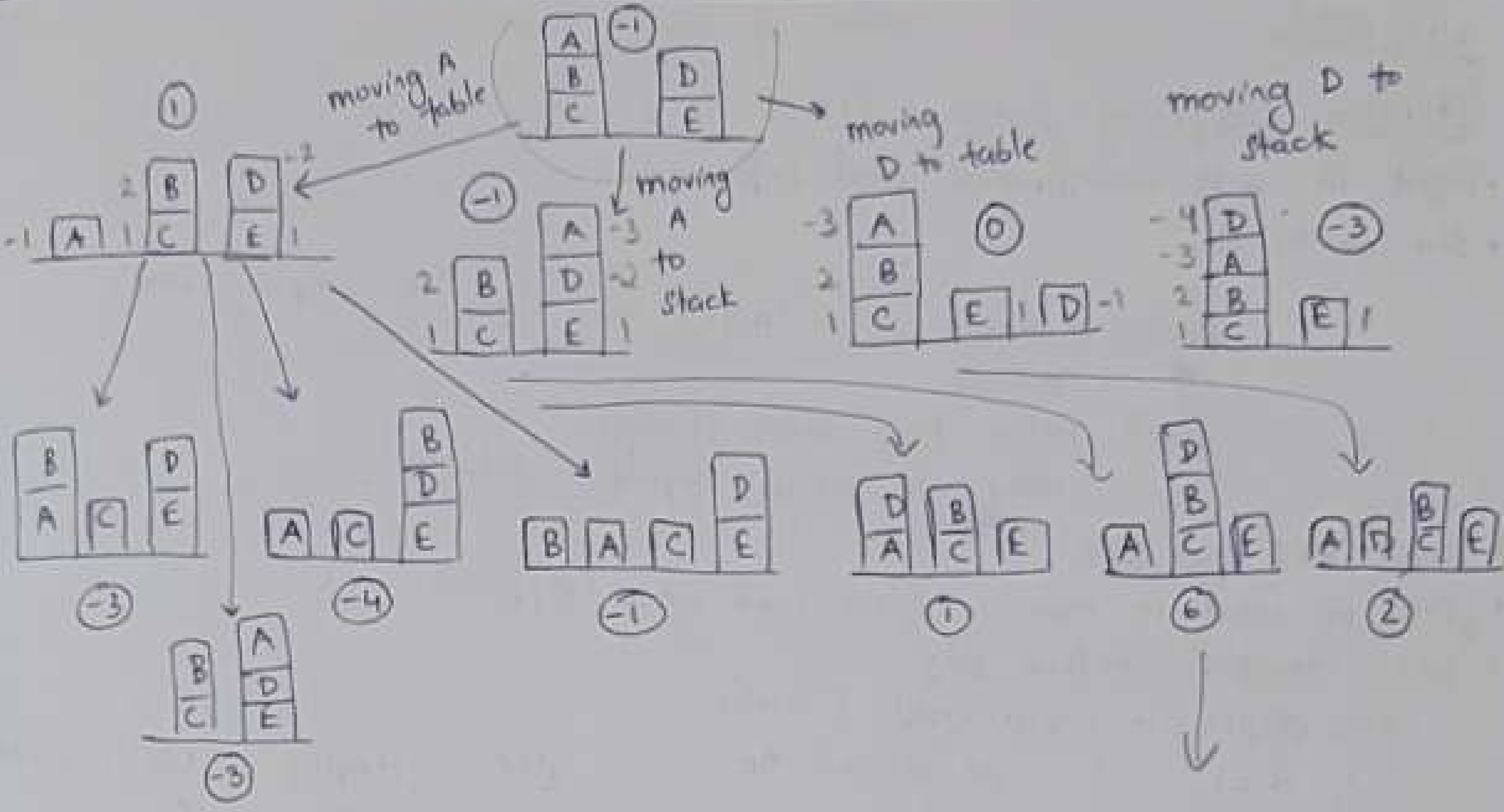
$h(n) = 1 + h(n)$ of all blocks below it if placed correctly
$\quad\quad\; -1 - h(n)$ of " " " " incorrectly.



start
$h(n) = -1$



goal: $h(n) = 11$


moving A to table
moving A to stack
moving D to table
moving D to stack

24.11.2022

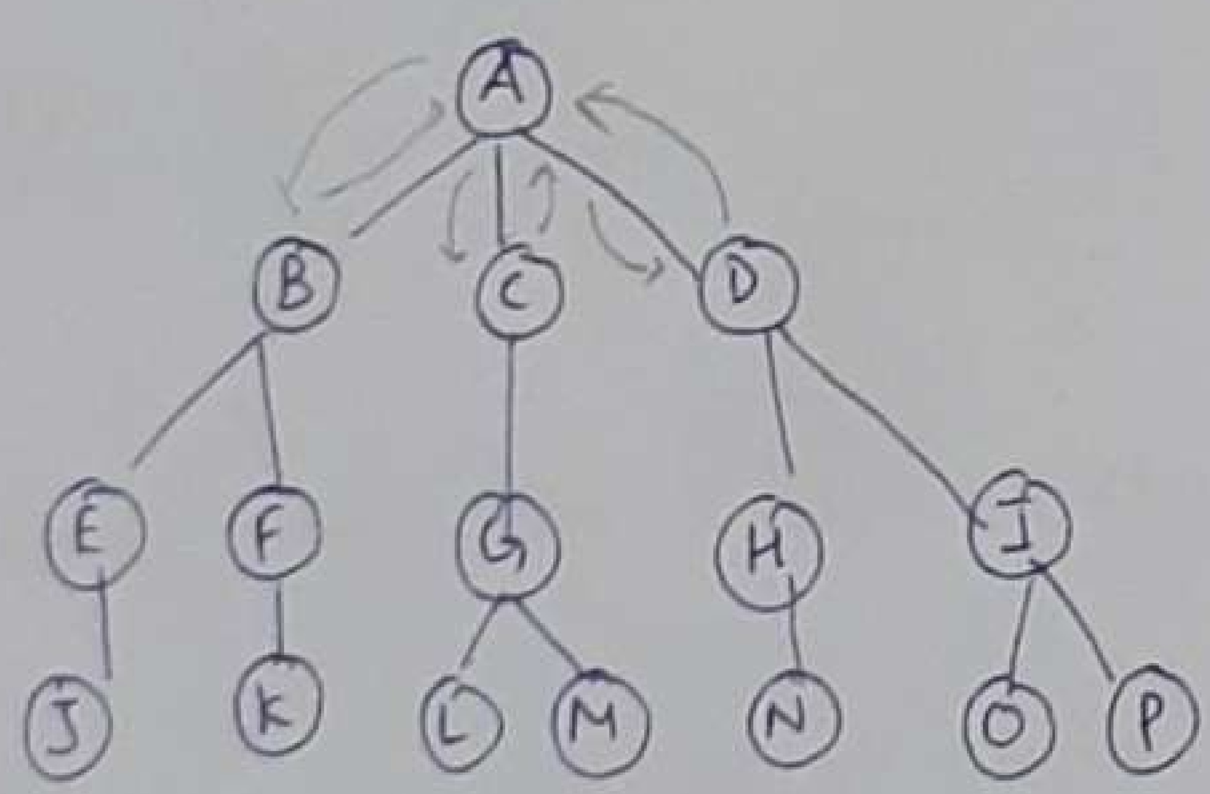
Iterative Deepening Search: (ID)

- used in both uninformed and informed
- Similar to DLS
  ↓
  prob of DFS? : if depth is too large, more time to explore whole branch
  more time, more storage
  may not get optimal solution.

- uses BFS and DFS
- can be used in memory constrained probs / applications
- each iteration, perform DFS
  $i=0$, depth $=0$
  $i=1$, $d=1$
  :
  $i=n$, $d=n$
  and check if nodes at d are the goal.

iter	depth	nodes visited
0	0	A
1	1	ABCD
2	2	ABEFCGDH
3	3	ABEJFKCGL MDHNIOP

goal has been reached.



goal = P

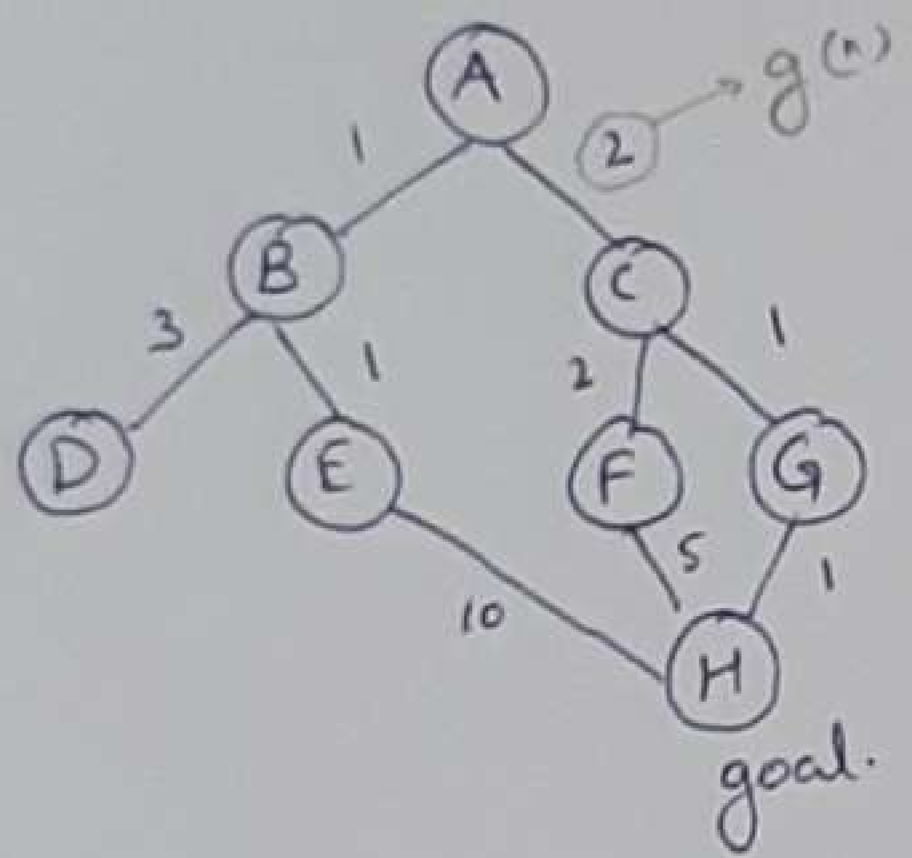
★ More iterations are required
We do the same thing again and again.

IDA*

A* is based on $f(n) = g(n) + h(n)$ (actual cost from src node over $g(n)$)

For IDA*, threshold = $\underbrace{\text{fscore}}_{\text{don't go beyond this}}$

↳ the threshold should increase with each iteration



n	h(n)	f(n)
A	5	5+0=5
B	3	3+1=4
C	4	4+2=6
D	5	9=
E	4	6
F	5	
G	2	
H	0	

itr = 0

Set threshold = 5 [as f(n) of A = 5]

~~open = A B E~~

choose B as it has min f(n)

~~closed =~~ ~~BE~~ A

~~open = BE~~

~~open~~ closed : A B

~~close~~ open : ~~A~~ ~~B~~ $\underbrace{\text{C D E}}_{\text{fn > threshold so we go to next iteration}}$

I = 1

next highest f(n) = threshold = 6

★ Disadv: in every itr, we have to perform A*

Adv: can be used in memory constrained problems.

open: A BC

closed: B A