Heuristic in AI [What, Why, How] ⑤

What is Heuristic in AI :- Kind of guess / Kind of quicksol

→ It is the problem (technique) to solve a problem quickly.

→ Rule of thumb (Greedy method)

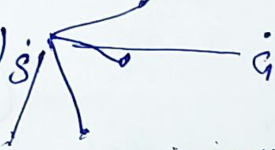
→ Heuristic Value low

– Heuristic function
  ↳ Gaurantee – Good
  | optimal (may be or not)

NP → P

How / Ecludian distance

(x_1, y_1) s



g (x_2, y_2)

minimum cost

Straight line distance

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Manhattan –

H/V

1	3	2
6	5	4
	8	7

1	2	3
4	5	6
7	8	

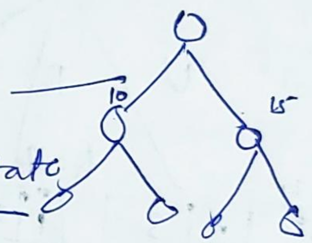
✓ $0 + 1 + 1 + 2 + 0 + 2 + 2 + 0 = 8$

✓ No. of misplaced tile – 5

Why

Cost × 2 = Selling price
$x = 100$
cost price = this

All State
↓
goal state



exponatially grow

8 Puzzle
$O(b^d)$
3^{20}
$3 \cdot 3 \cdot 4^8$

15 Puzzle
Brute force
10^{13}

24 Puzzle
10^{24}

Time & space
↳ exponantially increase

NP

✓ optimal

8-Puzzle with Heuristic - informed Search Tech

1	2	3
	4	6
7	5	8

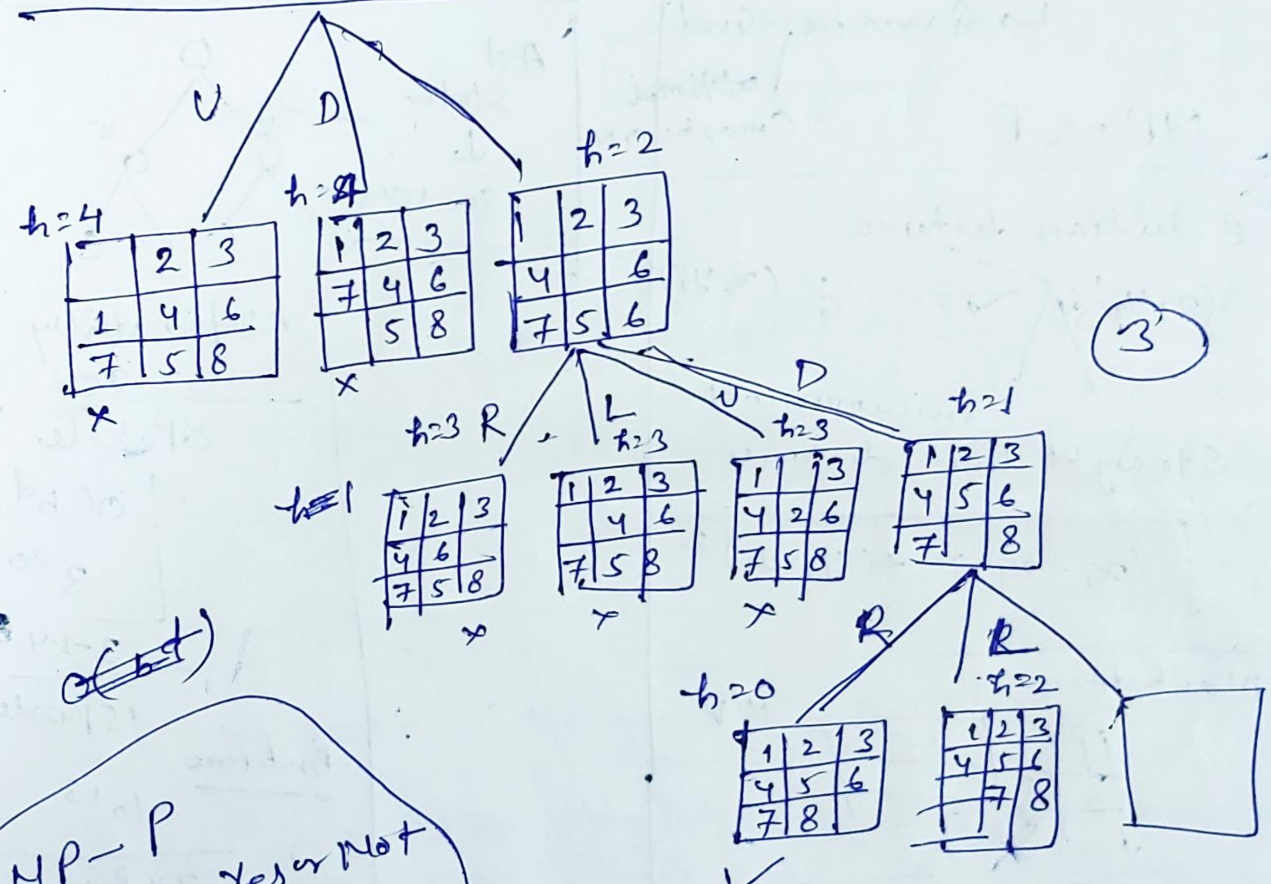
S

1	2	3
4	5	6
7	8	

G

h = heuristic Value.

h = 3

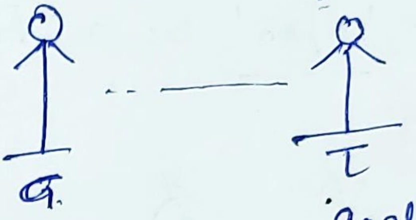


NP - P
optimal - Yes or Not
good solⁿ - Yes

Generate & Test (Heuristic Techniques) / DFS with backtracking

(7)

Generate — Possible solⁿ.
Test — to see if this is Actual solⁿ.



goal State Achi
solⁿ Accept
Drop

Properties of good generator

- Complete
  - → Atleast one solⁿ
- Non Redundent → No replica.
  - → exponential increase
- Informed
  - → domain Know.

---

100 100 100

0 0   0 0   0 0

0 0   0 0   0 1

0 2

0 3

3 No. Pin - 2 digit each

$(100)^3 = 1M$

exponential time
Brute force

---

1 M = 5

1 h = 5×60 = 300 (linear = $\frac{N}{2}$) = 5 lakh soln   24×10 w

---

heuristic → Problem Knowledge / → Domain Knowledge
↓
for exp → All Num Prime

0-99   0-99   0-99

0 0   0 0   0 0

25   25   25

$(25)^3 = 15000$

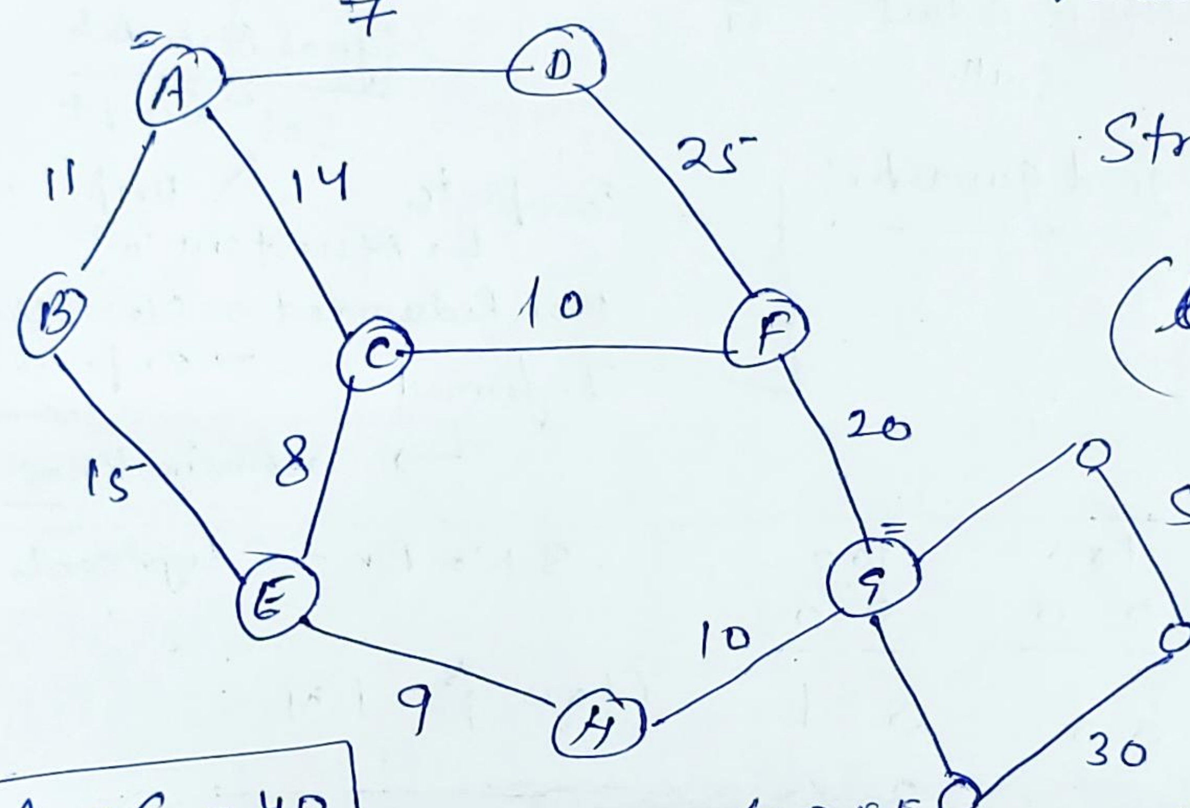
1 M - 5×60 = 30

$\boxed{2 \text{ days.}}$

Best first search — (Informed, Heuristic)

→ Select 'open' be a priority queue

(Node Heuristic Value → priority



Straight line distance (Eucledian distance)

(Heuristic Value)

Less - best

A → G = 40

B → G = 32

C − G = 25

D − G = 35

E − G = 19

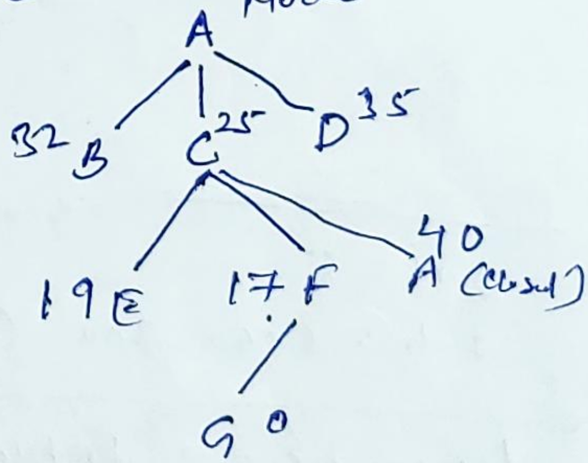
F − G = 17

H → G = 10

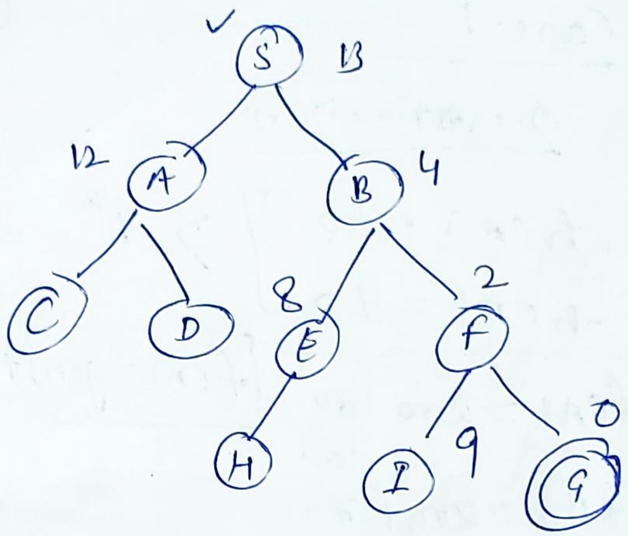
G − G = 0

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

Node



A → C − f − G



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

open close

initialization

Open = {A, B}   close [S]

1. open {A},   close [S, B]
2. open {E F A},   close [S, B, ~~F~~]

   open (E, A},   close [S, B, F]

3. open (E A I G},   close (S, B, F)

   open (E, A, I },   close (S, B, F, G)

$$\boxed{S \to B - F \to G}$$

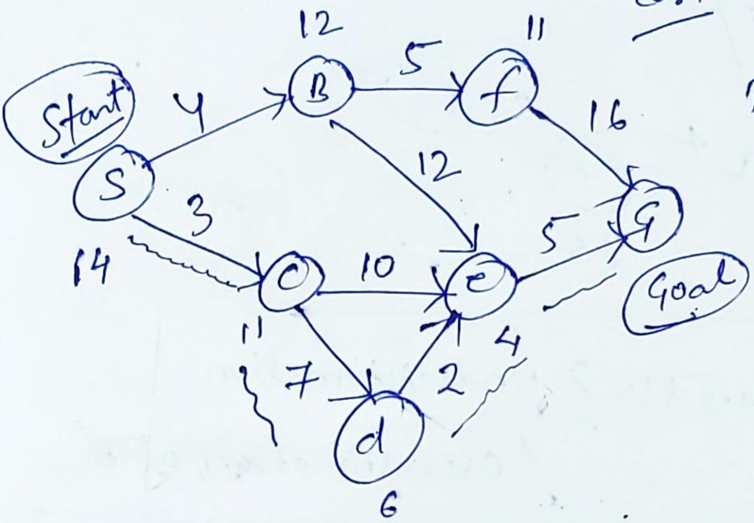
A^* Heuristic Algorithm — Informed search

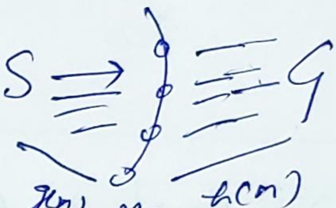
$f(n) = g(n) + h(n)$

cost — $f(n)$

Actual cost from start node to n — $g(n)$

estimation cost from n to Goal Node — $h(n)$





heuristic value

↓

optimal

$A^* \to * \to$ Admissible

$f(S) = 0 + 14 = 14$

(I)

$S \to B$: $4 + 12 = 16$

$S \to C$: $3 + 11 = 14$

(II)

$SC \to E$: $(3+10) + 4 \Rightarrow 17$

$SC \to d$: $(3+7) + 6 = 16$

(III)

$SB - f$: $5 + 4 + 11 = 20$

$SB \to e$: $4 + 12 + 4 = 20$

(IV)

$SCd - e$: $3+7+2+4 = 16$

(V)

$SCde \to G$: $3+7+2+5+0 = 17$

optimal cost = 17

$S \to C - d - e - G$

$TC = A^* = O(V+E)$ $(O(b^d))$

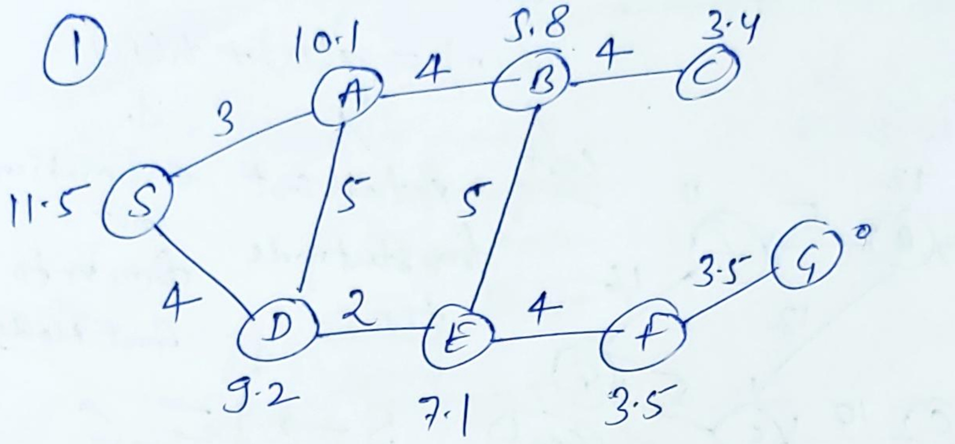
$d \to$ depth

$b \to$ branch

$SC = O(bd)$

[Underestimate
overestimate

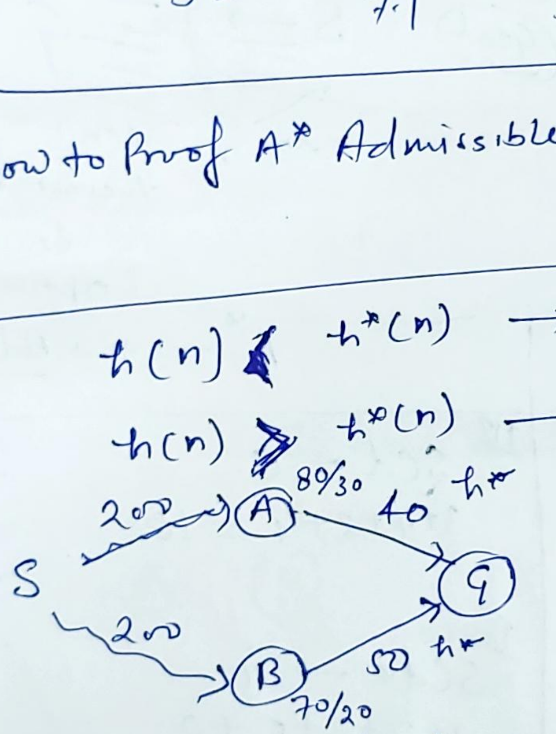
1



How to Proof A* Admissible ? Underestimation / Overestimation of A*

$h(n) < h^*(n)$ → underestimation

$h(n) > h^*(n)$ — overestimation.



S → n − G
g(n) h(n)

$h^*(n)$ → optimal
$h(n)$ − estimated

Labtop :- optimal - 30000
/actual

estimation → ?

Shop1 Shop2

Overestimation

Underestimation

40 > 30
↓
Next Shop

~~30~~ ~~20~~ 20 < 30
↓
Next Shop

$g(A) = 200$

$g(B) = 200$

Case-1

Overestimation

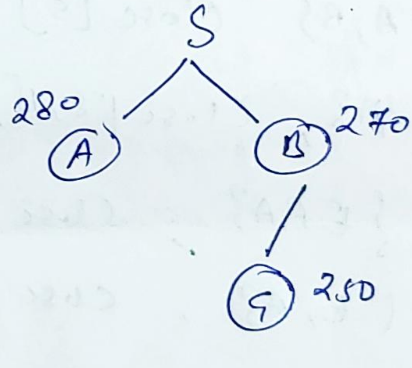
$\left.\begin{array}{l} h(A) = 80 \\ h(B) = 70 \end{array}\right] > h^*$

$\boxed{f(n) = g(n) + h(n)}$

$f(A) = 200 + 80$
$= 280$

$f(B) = 200 + 70$
$= 270$

$f(G) = g(G) + h(G)$
$= 250 + 0 = 250$



Case-2

Underestimation

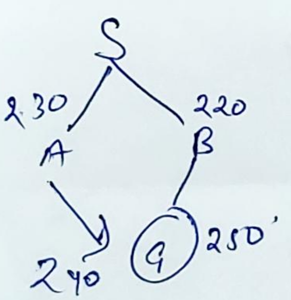
Underestimation

$h(A) = 30$

$h(B) = 20$

$f(A) = 200 + 30 = 230$

$f(B) = 200 + 20 = 220$



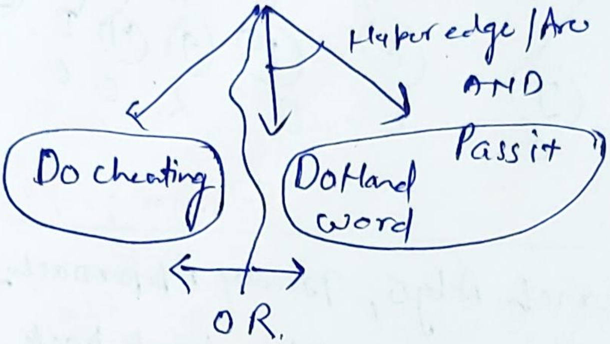
$f(G) = g(G) + h(G)$
$= 250 + 0$
$= 250$

$f(A/G) = g(G) + h(G)$
$= 240$ 0

AO* → Problem Decomposition (AND/OR) (13)

(Breakdown complex Problem to Smaller pieces.)
↓
solution

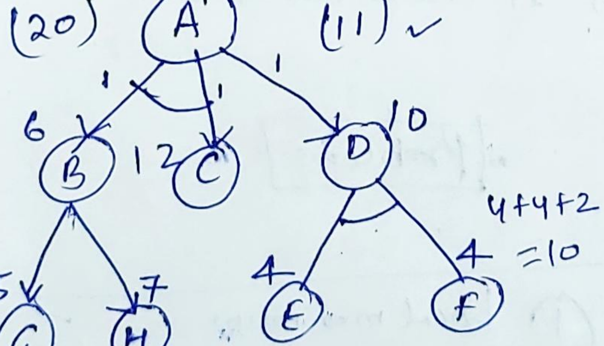
Want to Pass in exam

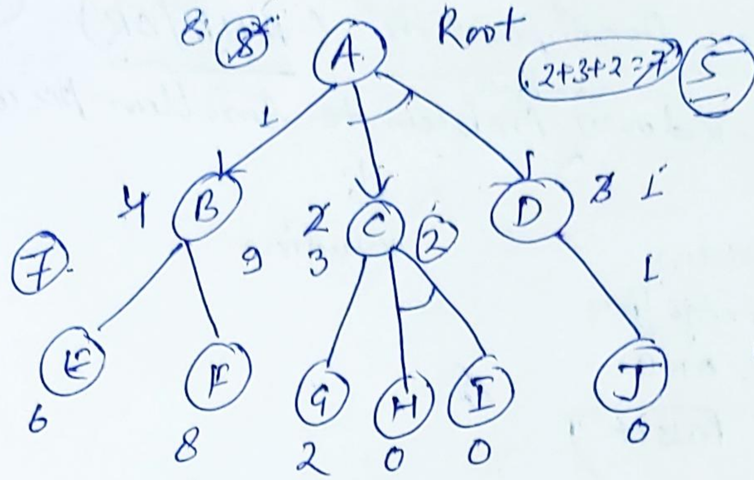


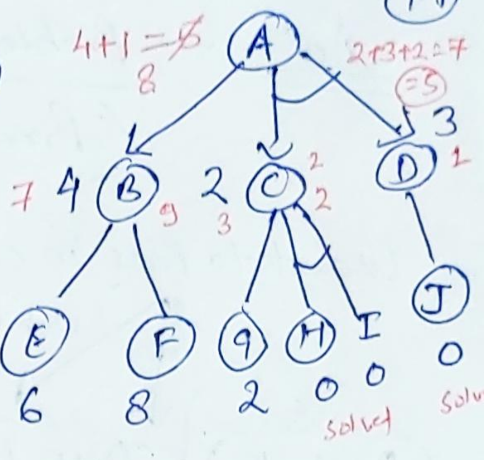
AO* A* → informed Search

A* → Underestimation
↓
optimal

AO* does not explore all the Solution paths once it got a Solution



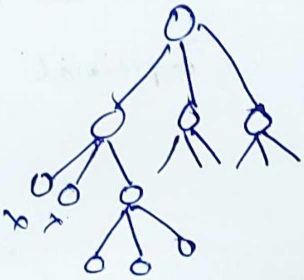




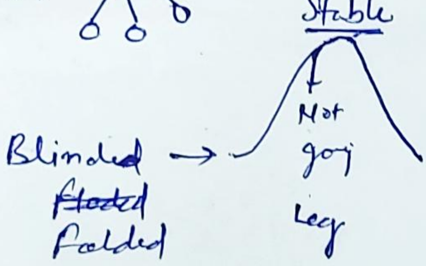
Solution graph

Hill climbing:- Local Search Algo, greedy Approach, No. back track

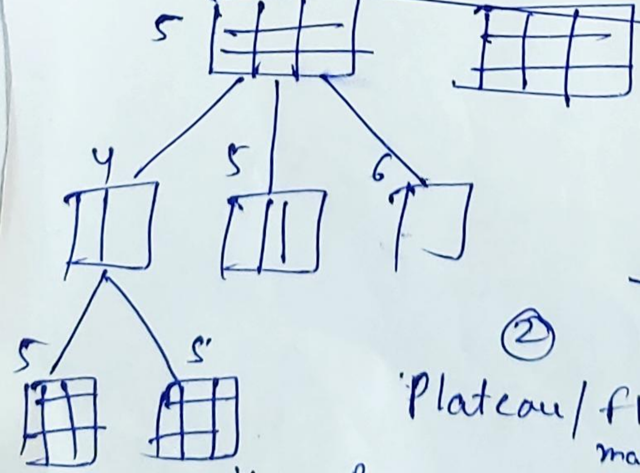
till Best then stop



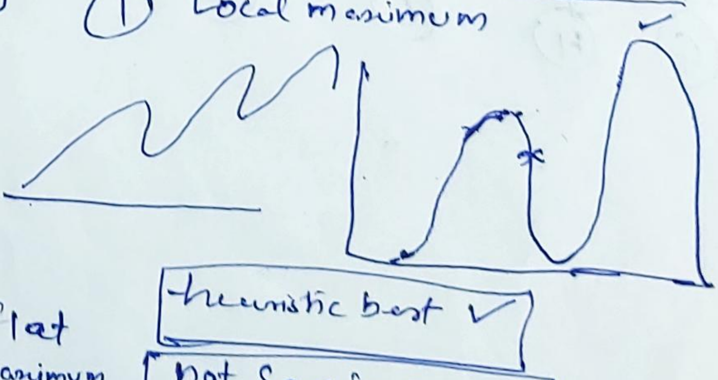
better than Current state then it new Current state.



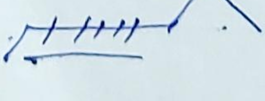
Problems



① Local maximum



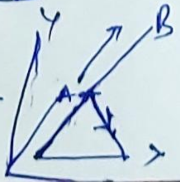
② Plateau/flat



heuristic best ✓

Not Save in memory.

3-Ridge



Exp: 8 Puzzle. (Already discussed)

Tic-Tac-Toe

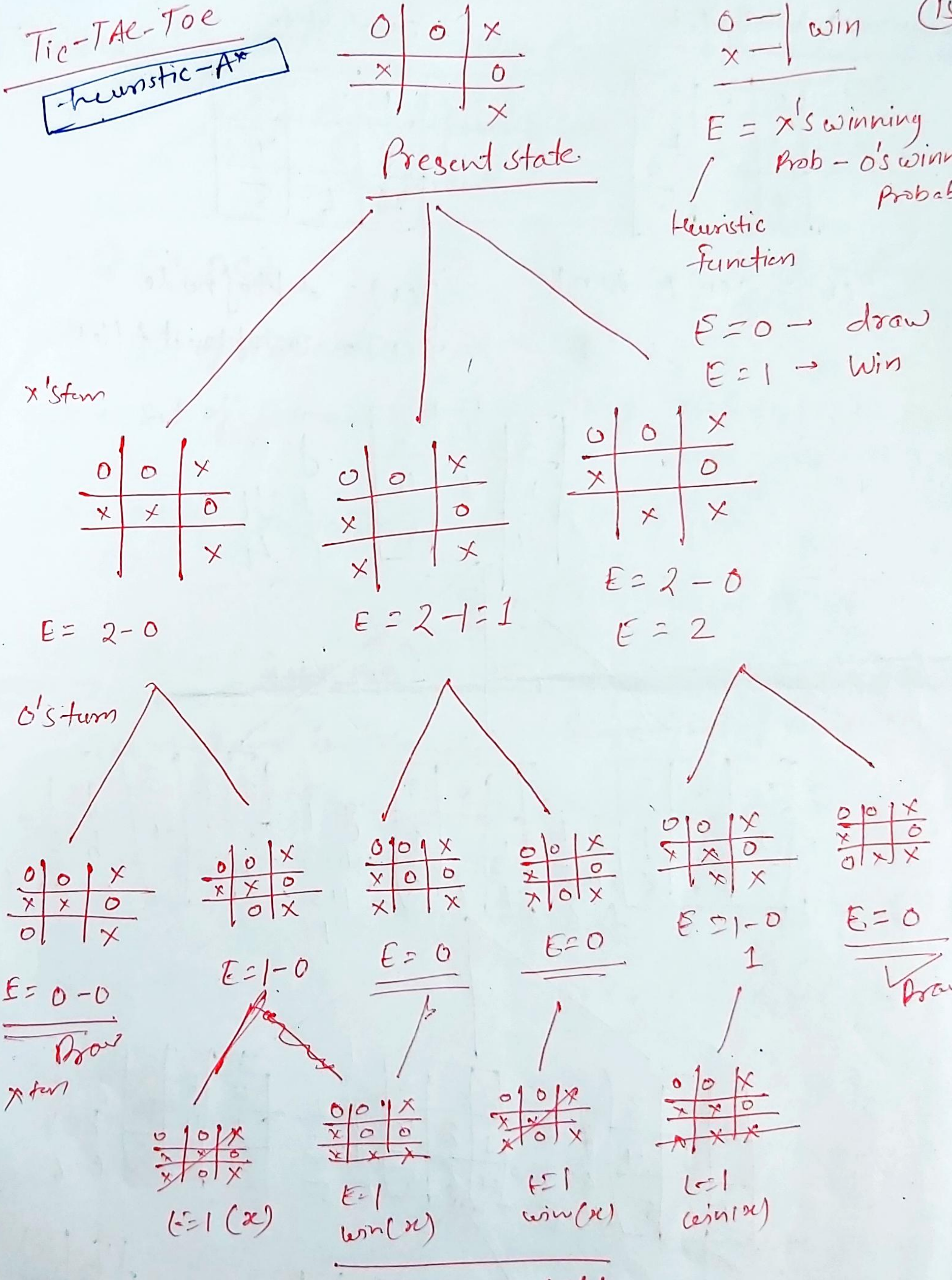
Heuristic-A*

O —| win
X —|

E = X's winning Prob − O's winning Probability

Heuristic function

E = 0 → draw
E = 1 → Win



This is best step.

initial state

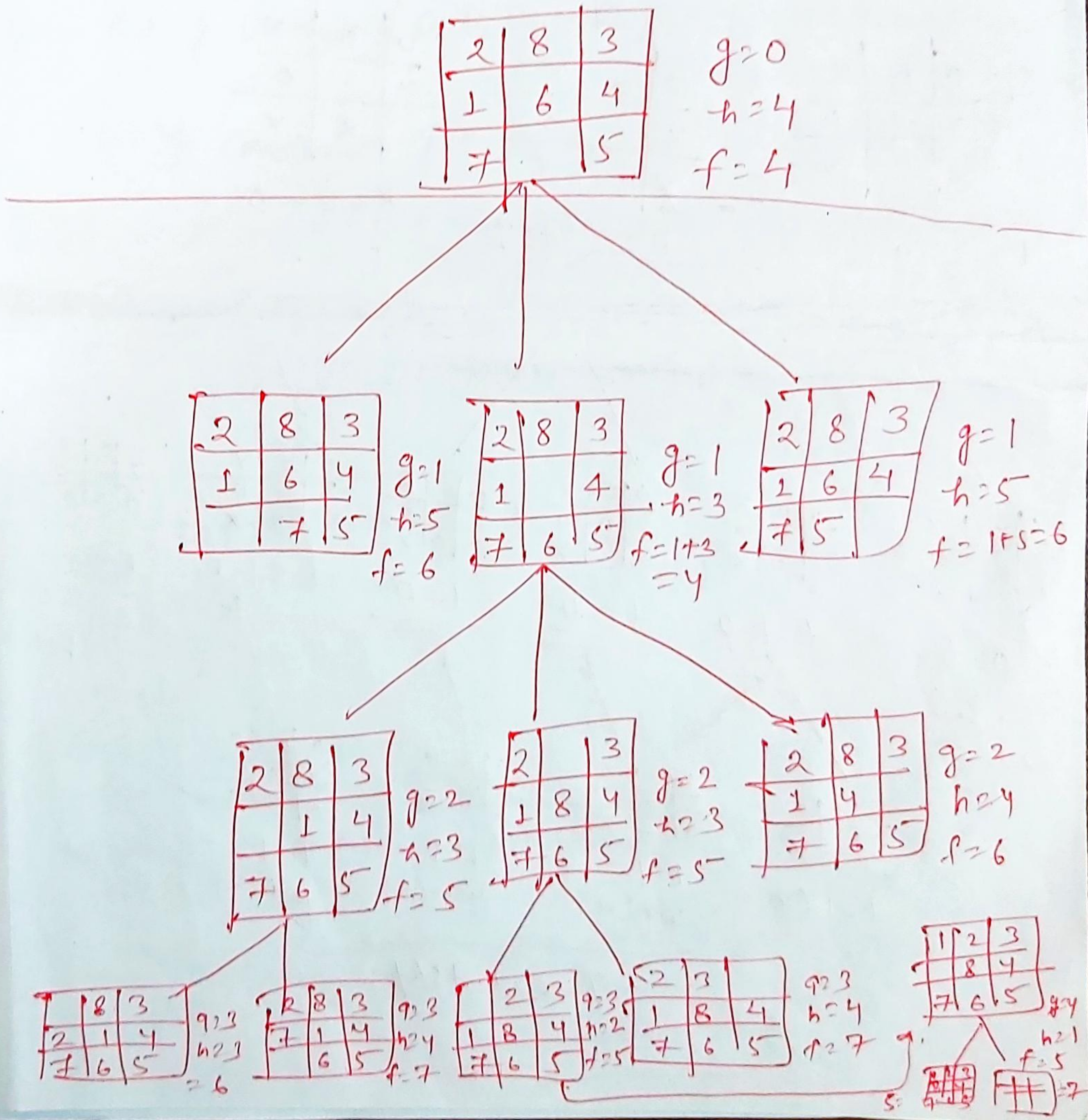
2	8	3
1	6	4
7		5

final state

1	2	3
8		4
7	6	5

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

$g(n)$ = depth of node
$h(n)$ = misplaced tiles.



Constraint Satisfaction Problem (CSP)

↳ way of representation is different

Informed / Uninformed

→ Tree – graph

↳ State space

CSP (VDC)

Sudoko
map Coloring
graph Colouring

V :- Set of Variable ($V_1 V_2 \cdots V_n$)

D :- Set of Domain ($D_1 D_2 D_3 — D_n$) one for each Variable

real, whole, Alphabet, natural

$V_1 — D_1$

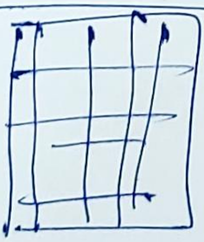
C :- Set of Constraints { $C_1 . C_2 C_3$)

↳ Kind of rule → Combination of Values.

C_i = (Scope, rel)

rel :- Value that variable Take.

Scope – Set of Variable



9×9 = 81 Square

(3×3) – 9 box

| 1 – 9 |

$$\begin{matrix} (V_1, & V_2) \\ | & | \\ A & B \\ (1,2) & (2,4) \end{matrix}$$

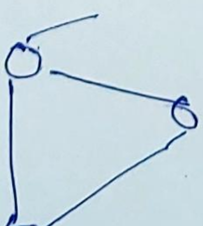
$C_1 = ((V_1\ V_2)$
$V_1 \neq V_2)$

$C_1 = (V_1, V_2), (A, B)$

$C_1 = (V_1\ V_2)\ \{(1,2)$
$(1,4)$
$(2,4)\}$

Space :- Var — 81 ←
domain → 1 – 9

Constraints ! – Number not Same $\boxed{3\times3}$



Constraints – No Same Color in Neighbour

Achieve rule.