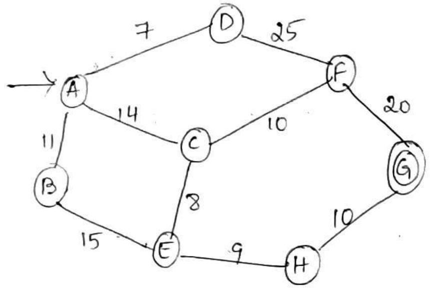
8/12/23 Heuristic Search Techniques (uninformed Search)

* Best First
* A* Algorithm.

→ Best First Search ~~Alg~~



States	h(n)
A	40
B	32
C	25
D	35
E	19
F	17
G	10
H	0

g(n) → cost from start state to any node

h(n) → heuristic value
↳ cost from any node to goal state

(*) {Heuristic value of goal node is always 0}

1) push start state to open list ⇒ (2) assigned to NODE

(3) check goal or not

(4) CLOSED & expand (5)

B C D

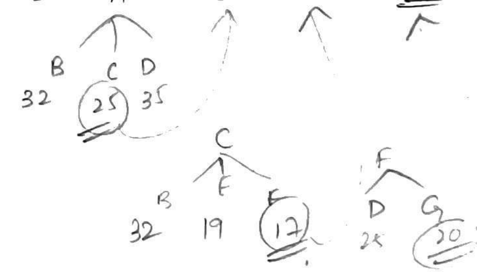
(6) } sorted acc. to heuristic value, not cost (*)

(only diff in Best first Search as compared to Uniform Cost Search ⇒ is sorted acc. to heuristic value & not cost

OPEN	CLOSED
~~A~~	1 A → 2 B C D
3 B C D } sort	

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

Path ⇒ A $\xrightarrow{40}$ C → F → G



OPEN	CLOSED	
~~A~~	A	It 1: A == G — False; Succ. of A: B = 32, C = 25, D = 35; Sort of ": C, B, D
B = 32, C = 25, D = 35 ⇒ ~~C~~, B, D		
~~C~~	C	It 2: C == G — False; Succ of C: B = 32, E = 19, F = 17
B, E, F ⇒ ~~F~~, E, B		
	F	

A* Algorithm

(8 puzzle prob using " algo)

Type 1

3	7	6
5	1	2
4	□	8

Start

5	3	6
7	□	2
4	1	8

Goal

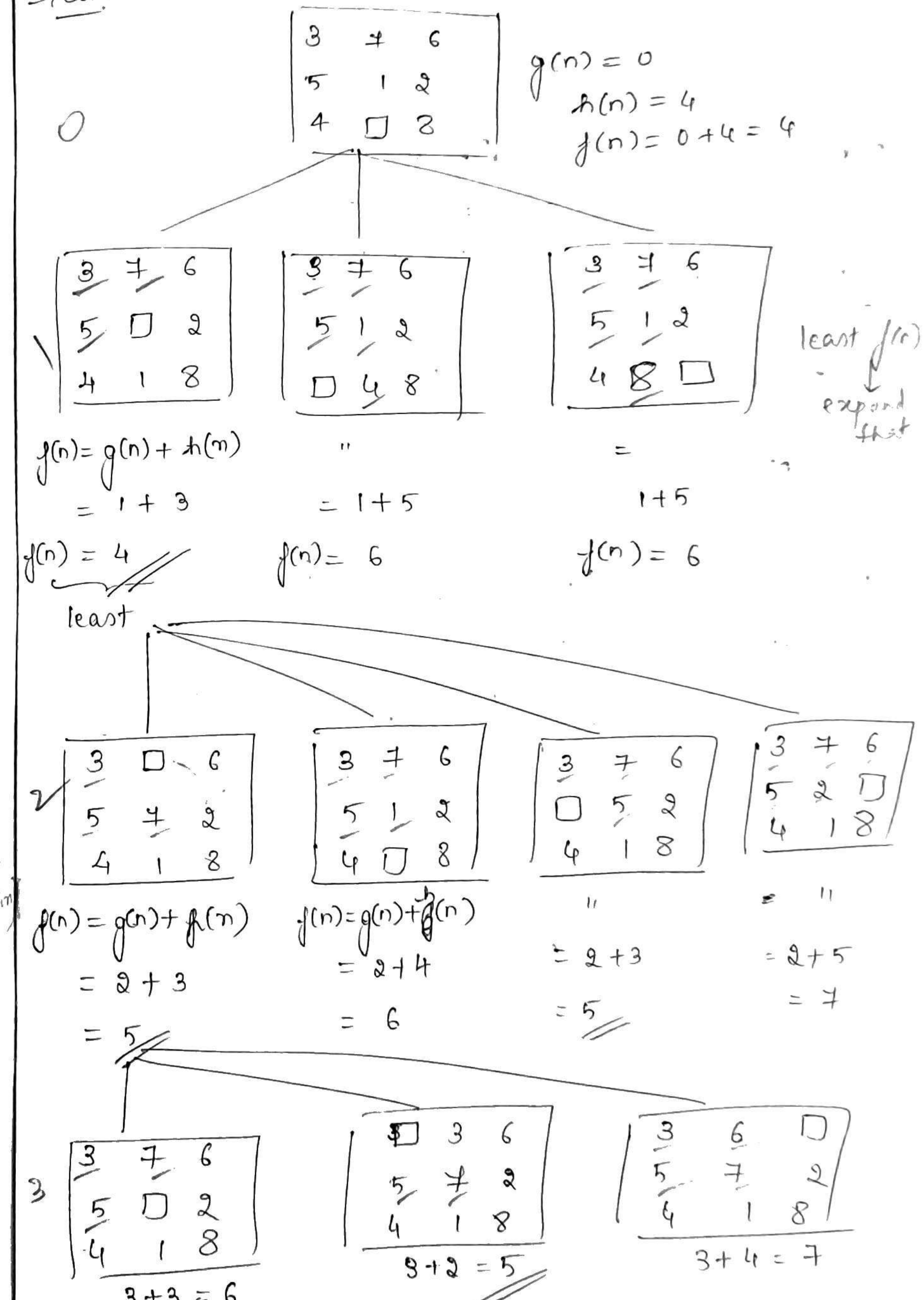
$h(n) = 4$ (in 1st iteration)

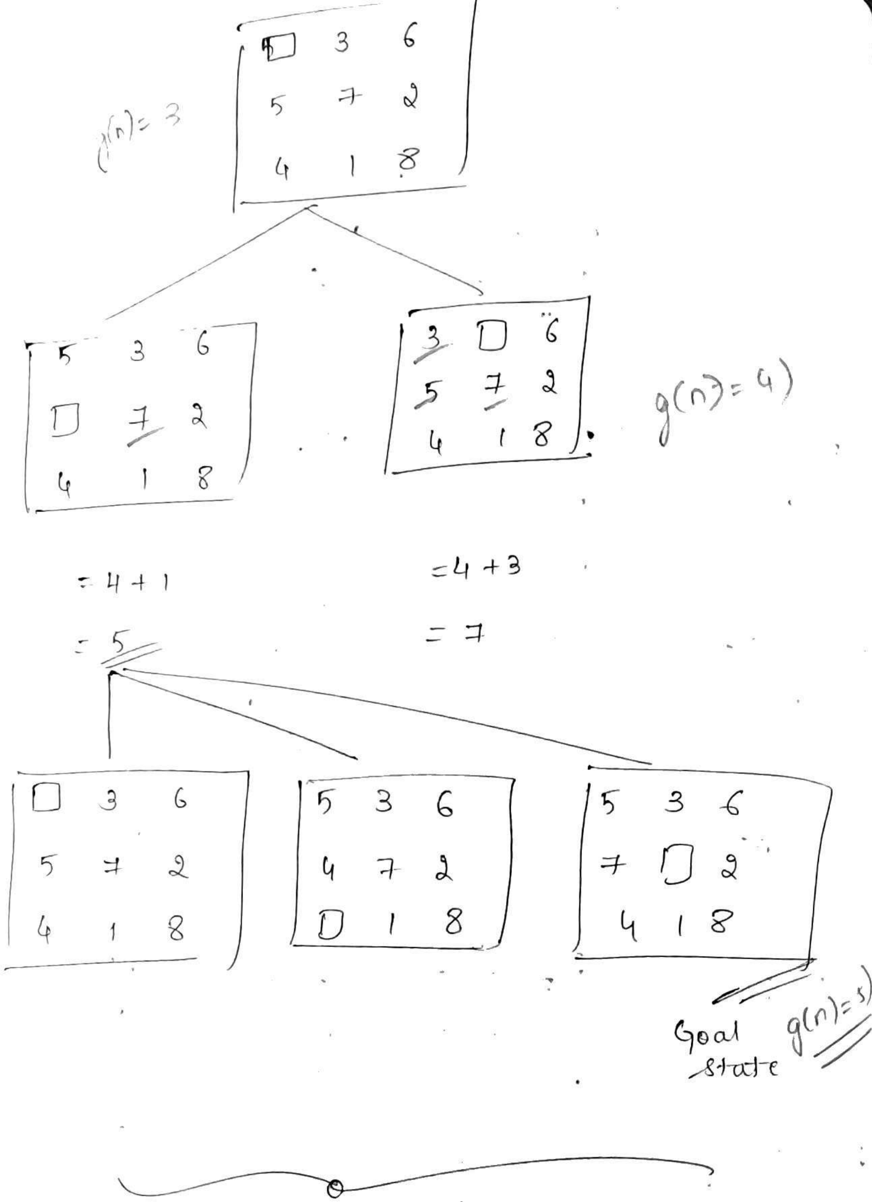
$g(n) \rightarrow$ level no. [for start state ⇒ $g(n) = 0$

$h(n) \rightarrow$ no. of places/positions are misplaced } for ex: in 1st iteration ⇒ (by comparing it with goal)

$f(n) = \underbrace{0+4}_{g(n)+h(n)}$

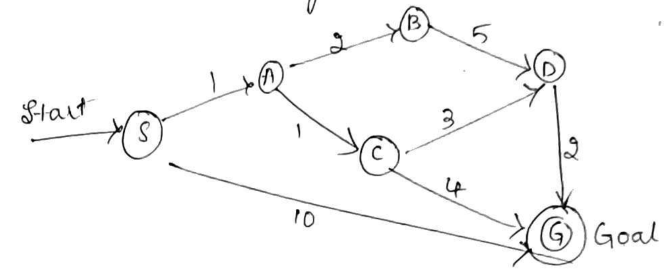
⇒ Soln.





Type ②

here $g(n) \to$ edge cost
$h(n) \to$ given.



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

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

$S \to A \Rightarrow f(n) = g(n) + h(n)$
$= 1 + 3$
$= 4 \longrightarrow$ least $\quad S \to A <^{B}_{C}$

$S \to G \Rightarrow f(n) = 10 + 0$
$= 10$

$S \to A \to B \Rightarrow f(n) = 2 + 4 = 6$

$S \to A \to C \Rightarrow f(n) = 1 + 2 = 3 \longrightarrow$ least $\quad S \to A \to C <^{D}_{G}$

$S \to A \to C \to D$ } $\Rightarrow f(n) = 3 + 6 = ?$

$S \to A \to C \to G$ } $\Rightarrow f(n) = 4 + 0 = \underline{\underline{4}}$ (least)

$\Downarrow$

Goal State (so stop)

(if heuristic not used $\Rightarrow S \to A \to C \to D \to G$ long)

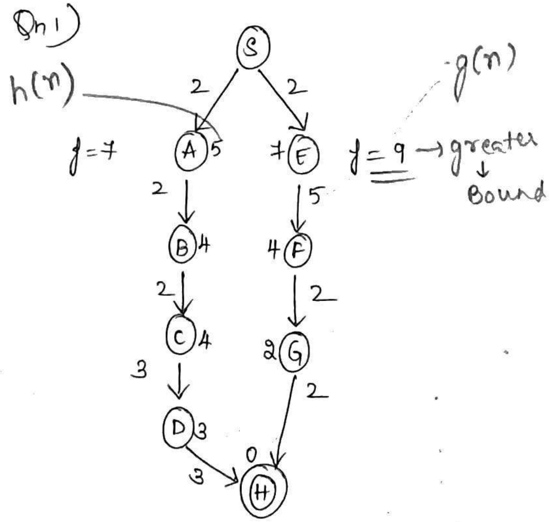
11/12/23

* Recursive Best first Search (RBFS)

Qn 1)

$f(n) \Rightarrow$ F - score

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



A $f = 7$

B $f = 2+2+4 = 8$

C $f = 10$

D $f = 2+2+2+3+3 = 12$

$E = f = 9$

$F = 5+2+4 = 11$

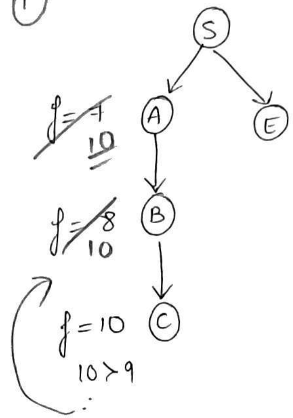
$G = 2+5+2+2 = 11$

everytime expand
↓
compared with Bound value

$\boxed{\text{if} < \text{Bound} \to \text{expand}}$

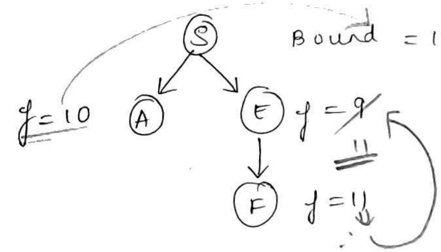
(1)

Bound = 9



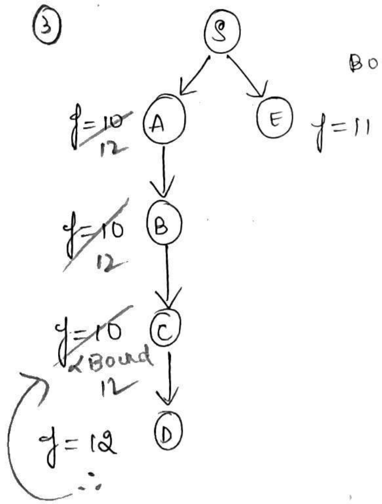
(2)

Bound = 10



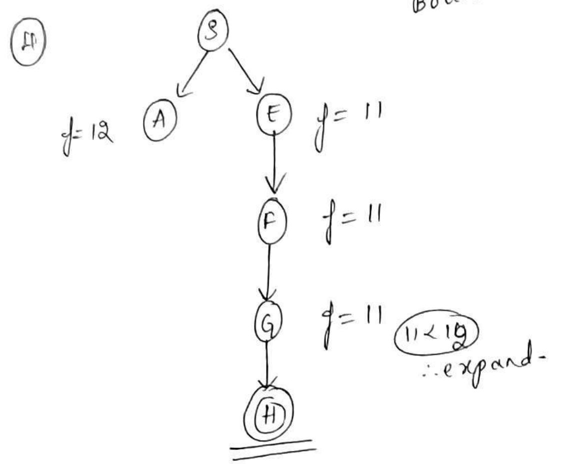
(3)

Bound = 11

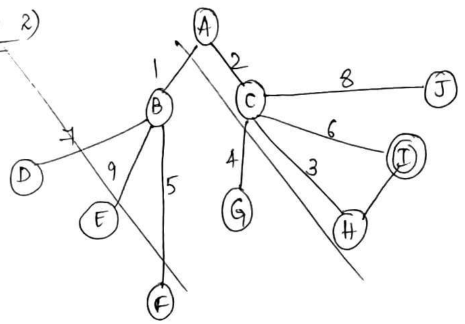


Bound = 12

(4)

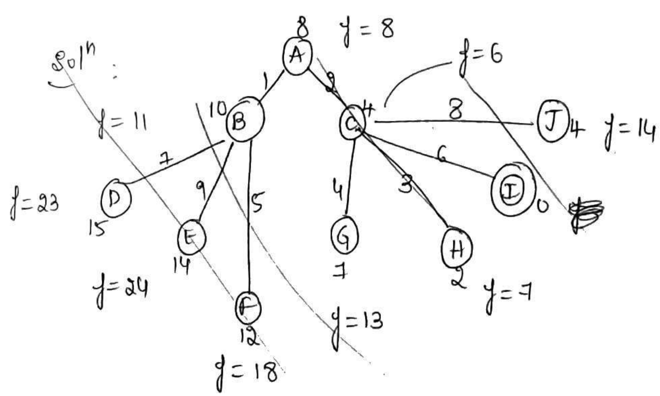


Qn 2)

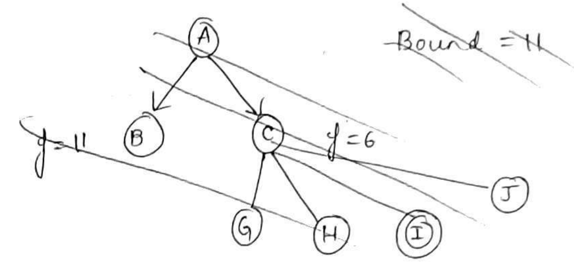


h(n)	
A	8
B	10
C	4
D	15
E	14
F	12
G	7
H	2
I	0
J	4

Soln :



~~Bound = 11~~



Qn. on A* Algo (8 puzzle prob.)

1)

7	2	4
5	□	6
8	3	1

□	1	2
3	4	5
6	7	8

2)

1	2	3
8	□	4
7	6	5

2	8	1
□	4	3
7	6	5

1) Soln.

Start State:

7	2	4
5	□	6
8	3	1

Goal State

□	1	2
3	4	5
6	7	8

(X)

Qn vvvv X:

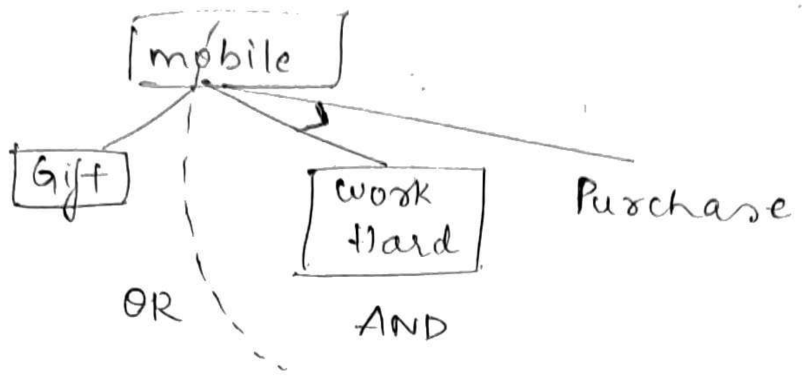
Consider the foll. prob:

A farmer wants to get a lion, a fox & a goose & some corn across the river, there is a boat but he can only take one in addition to himself on each trip. or else both the goose & the corn ~~&~~ or both the fox & the corn. The corn cannot be left with goose, ~~similarly~~ as it will eat the corn. Similarly the fox can eat goose & also lion cannot be left with the fox. How does everything get across the river. ~~Anima~~ Assume animals do not wander off when left alone

1) Give start & goal states along with constraints
2) Draw state space search trees using DFS & BFS to find first solution.

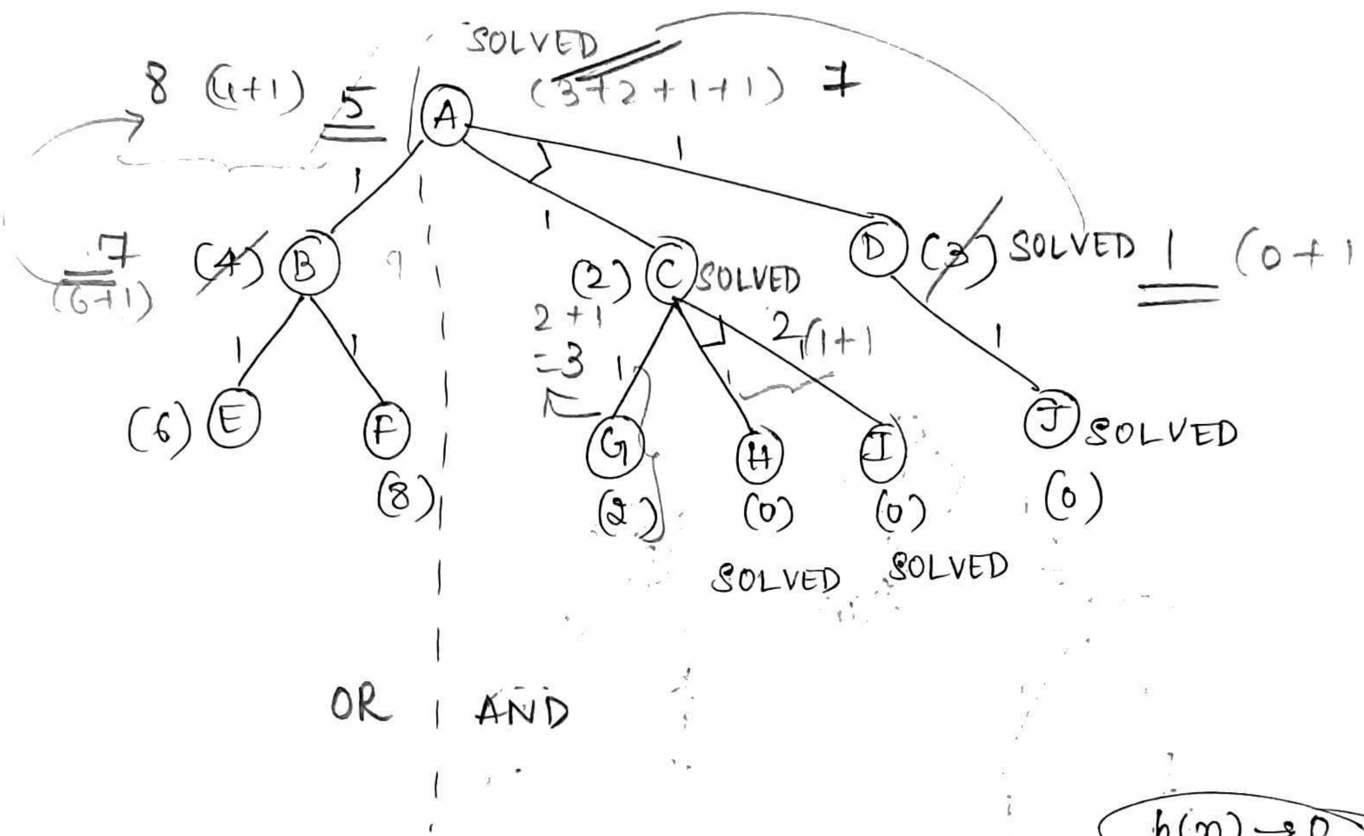
15/12/23

AO^* algorithm → explained using AND - OR graph

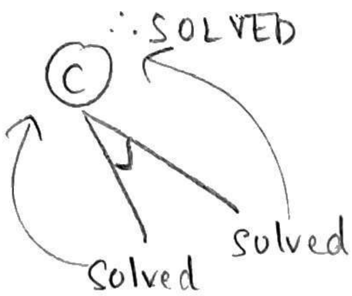


edge values → $g(n)$ → default taken as 1

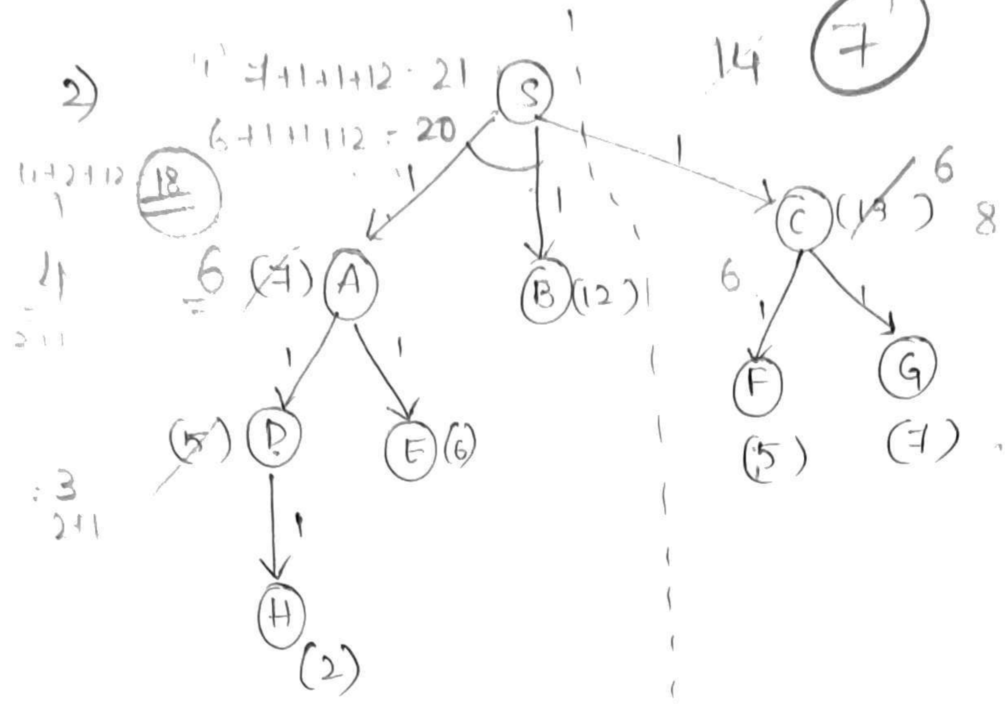
1)



$h(n) \rightarrow 0$
↓
Goal node
↳ label it as SOLVED



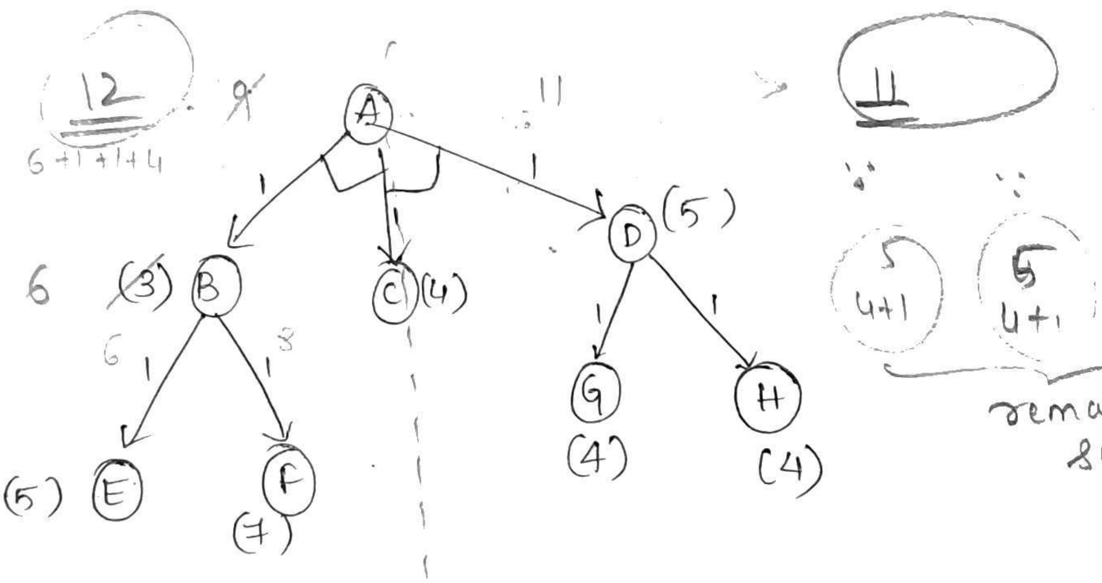
2)



Path 1:

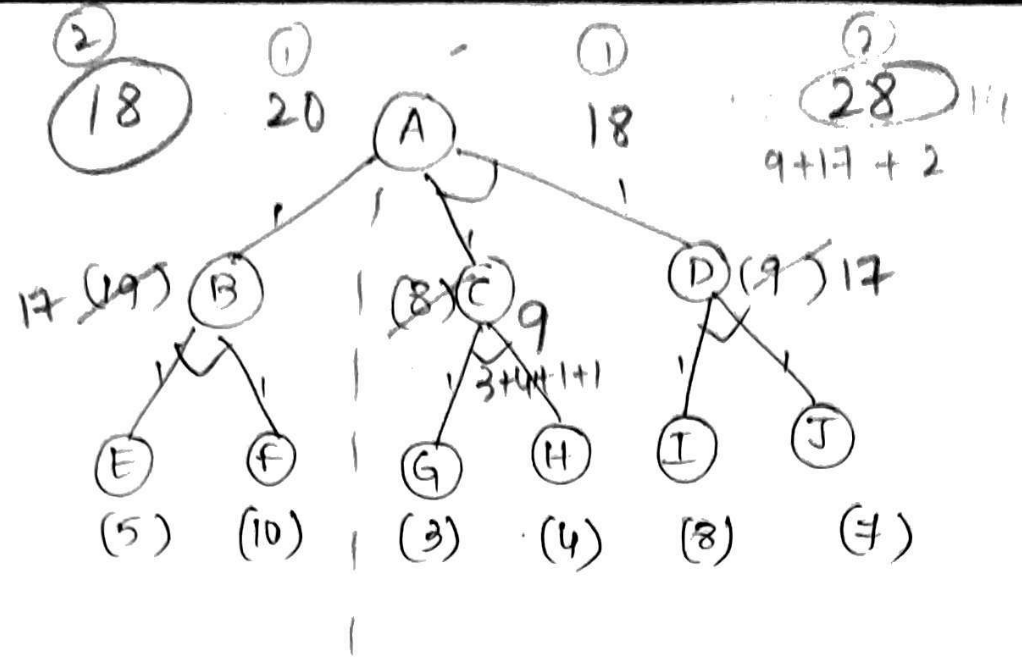
(explore all nodes)
E was large → 10

3)



remains same

4)



Constraint Satisfaction - Crypt - Arithmetic Puzzle

- 0 - 9 } digits to be assigned
  - ↳ 8+9 max → carry vl be always 1

1)

```
   T O          □ □
 + G O          □ □
 -----        -------
 O U T        □ □ □
```

left most → assign 1 (∵ carry)

```
    2  1
 +  8  1
 --------
 1  0  2
```

if it is 9 → 2+9

```
 1 1
  ↓
  0 X
```

+1 ⇒ carry ∴ 10 (9+1)

2)

```
   S E N D            9  5  6  7
 + M O R E    ⇒   +   1  0  8  5
 ---------        --------------
 M O N E Y        1   0  6  5  2
```

2, 3, 4

3)

```
   B A S E           □ □ □ □
 + B A L L    ⇒   +  □ □ □ □
 ---------        ----------
 G A M E S        □ □ □ □ □
```

```
    7  4  □  □
 +  7  4  □  □
 -------------
 1  4  8  □  □
```

4+4 ×

5
6
7
8
9

```
 1  ~~1~~ |  1 1    |  1
 9   8    |  8  6   |  7  4   A
 9   8    |  8  6   |  7  4
 8   6    |  6  2   |  4  8 ✓
```

0 1 × 2 × 3 4 × 5 6 7 × 8 × 9

5/1/24.

Game Playing - Nim Game Problem)

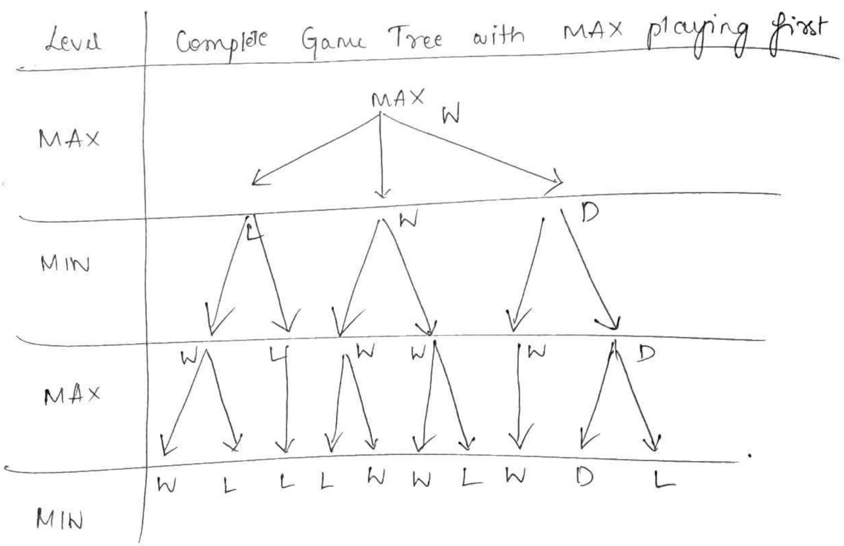
if j is a non-terminal MAX node then, WIN - when

Status(j) = { WIN - if any are WIN
LOSS - if all is LOSS
DRAW - if any is DRAW & NONE is WIN }

if j is a non-terminal MIN node then

Status(j) = { WIN - if all are WIN
LOSS - if any is LOSS
DRAW - if any is DRAW & None is LOSS }

Status Labeling Procedure



MAX player (computer) MIN player

✓ pile of match sticks

can take half of pile (maximum)

✓ Goal: MAX should win

if MIN is first player → MIN (L) ⇒ it means Max Lost

NOTE Irrespective who plays first ⇒ the root node label is MAX's status

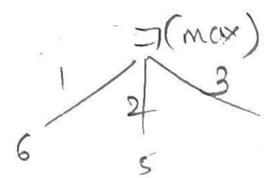
✓ players take alternate turns
✓ labeling → done in bottom up approach
✓ whole game → MAX's point of view ⇒ MAX shud win

~~Level Complete~~

1 stick left → loss.

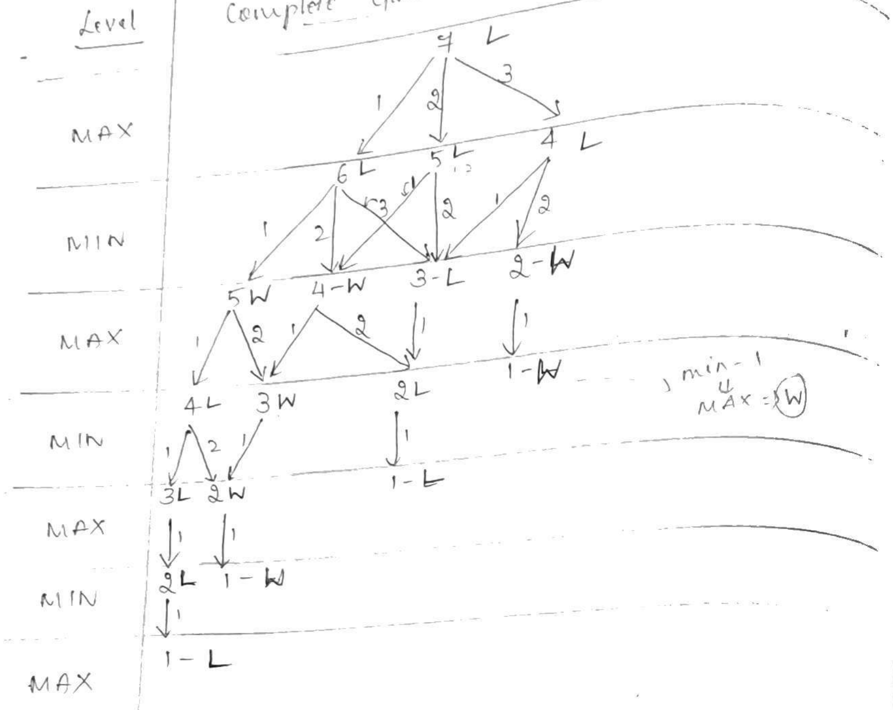
if MIN has ⇒ (W)

if MAX has ⇒ (L)



} half $\frac{7}{2} = 3$

Complete Game Tree with MAX playing first.



N = 29

2 strategies (do it for MAX)

MIN → half & distribute

① Binary Rep.

N = 13

8 4 2 1

1 1 0 1

0 1 0 1

0 0 1 1

0 1 1 0

8 4 2 1 → 6

Remove MSB & add.

② ⇒ N = { 3, 7, 15, 31, 63, ... }

if N = 13 ⇒ subtract nearest

⇓

13 − 7 = 6

Case 1: MAX is the first player & N ∉ { 3, 7, 15, 31, 63, ... } matchsticks

Case 2: MAX is the second player & N ∈ { 3, 7, 15, 31, 63, ... } matchsticks

1

N = { 3, 7, 15, 31, 63, ... }

2 strategies

① Binary repr

2

13 − 7

6

1) Case-1

N = 29

N = { }

MAX - first player

2 Cases.

16 8 4 2 1

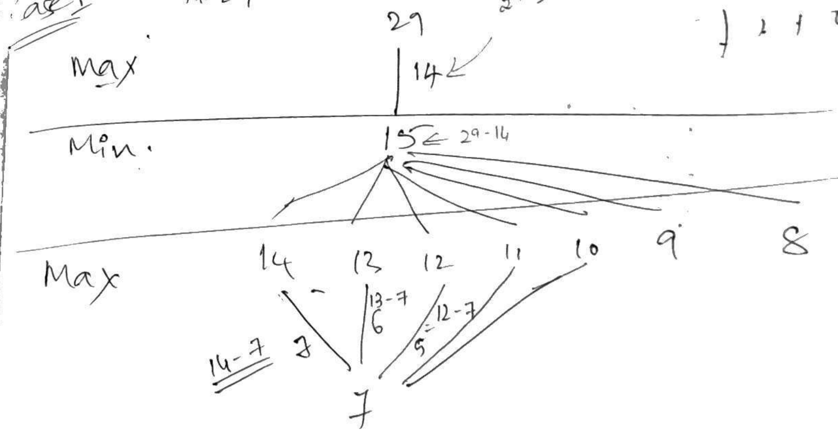
1 1 1 0 1

0 1 1 0 1

1 1 1 0

29 − 15 nearest in N of ?

Case I N = 29

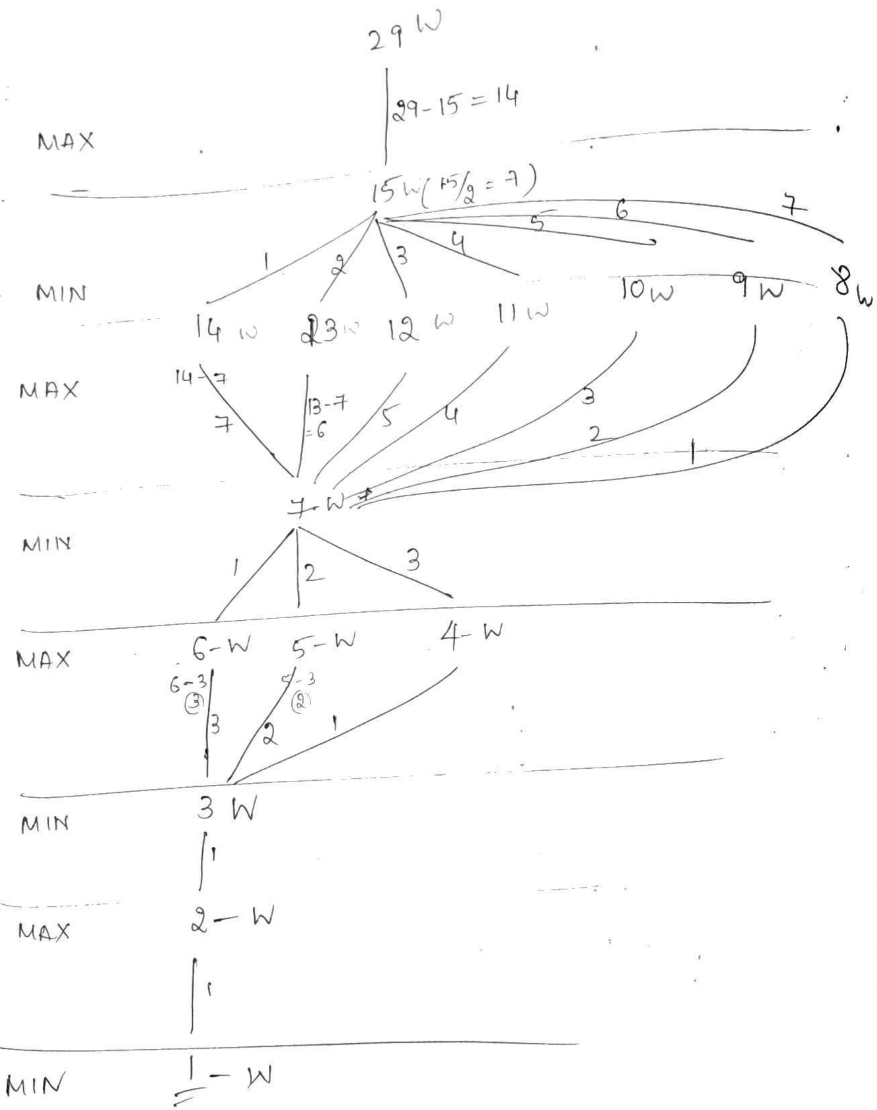


Sol

Given : N = 29

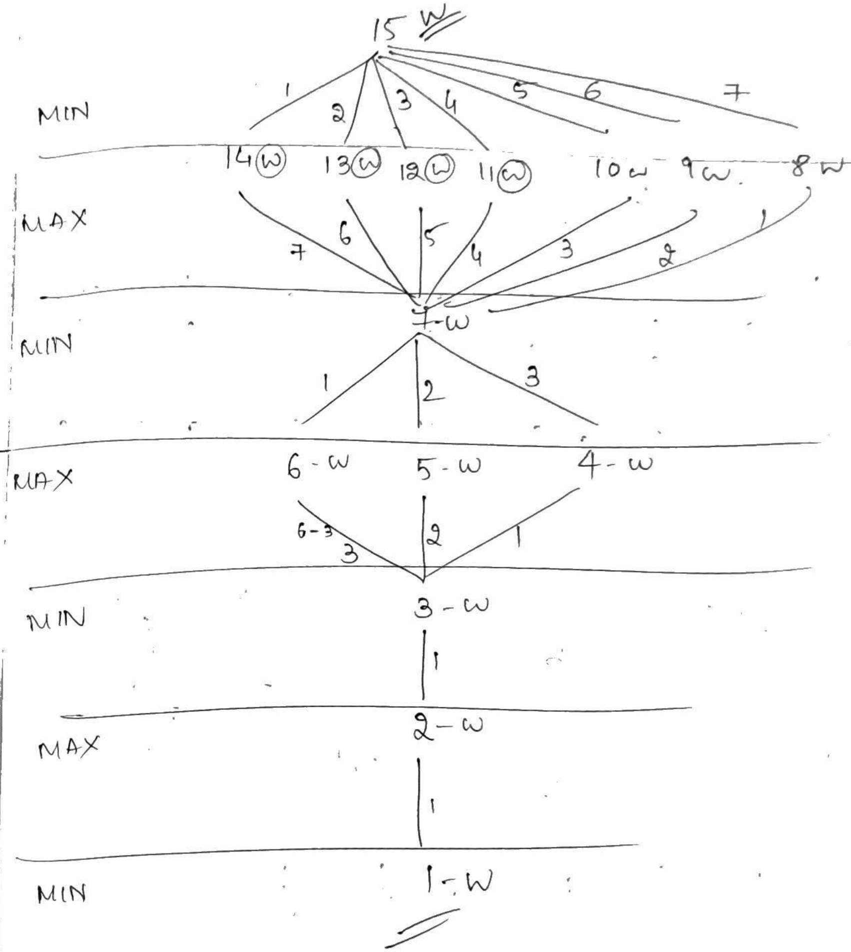
~~N ∈ {3~~ N = { }

N ∉ 29 ∉ { } → ∴ MAX is the first player.



(2) N = 15

Case 2.

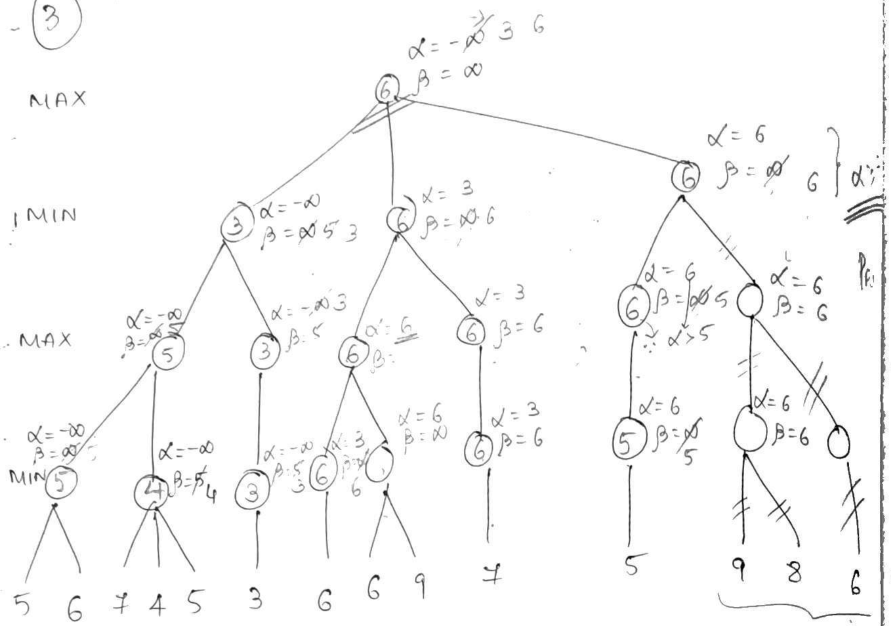


$\alpha \rightarrow$ MAX

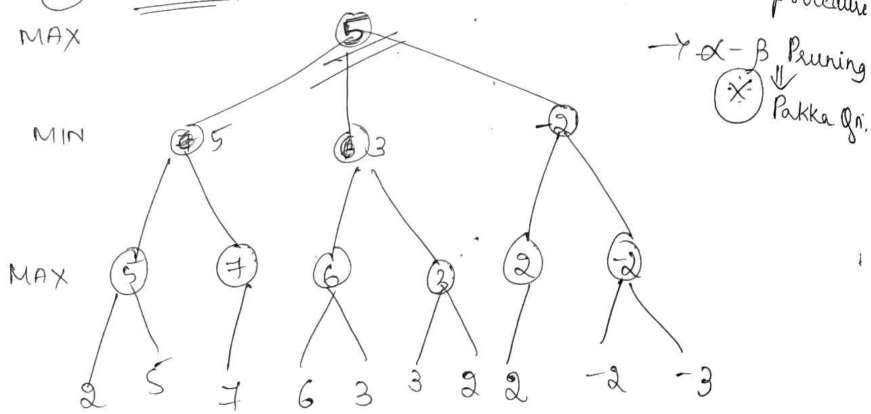
$\beta \rightarrow$ MIN

α - β Pruning

(3)



(a) MINIMAX



→ MINIMAX procedure

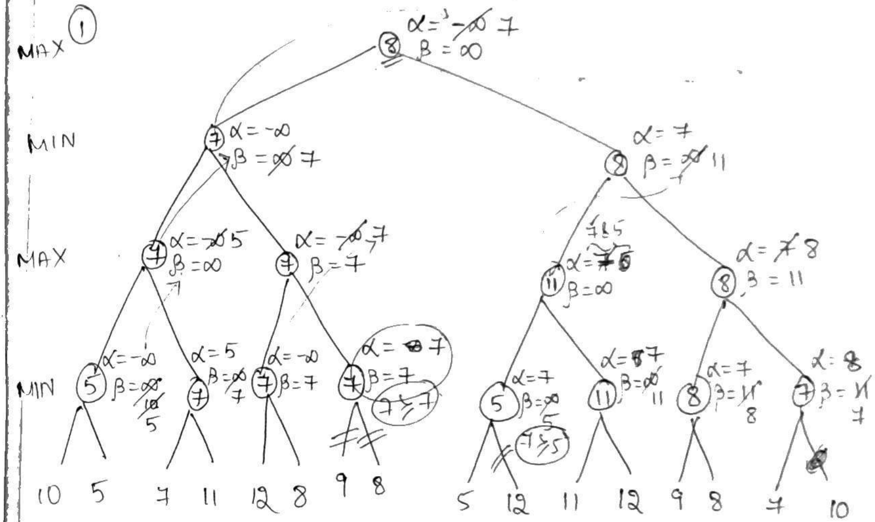
→ α - β Pruning ⇓ Pakka Qn.

- if $\alpha \geq \beta \rightarrow$ prune
  ↳ delete that branch

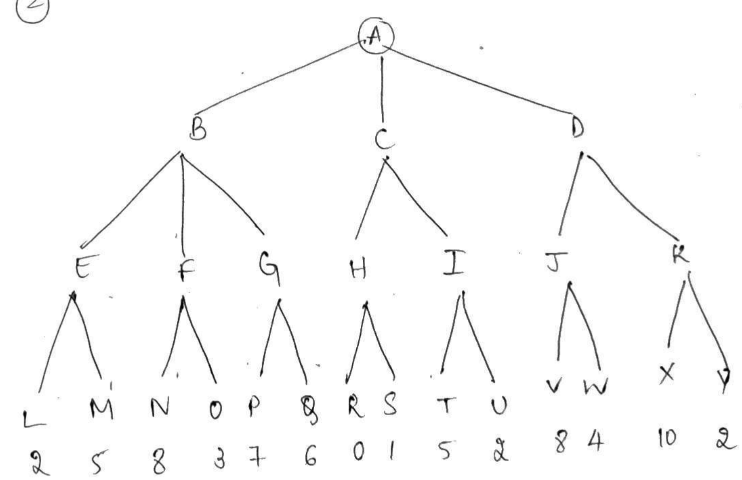
- MAX $\rightarrow \alpha$
- MIN $\rightarrow \beta$

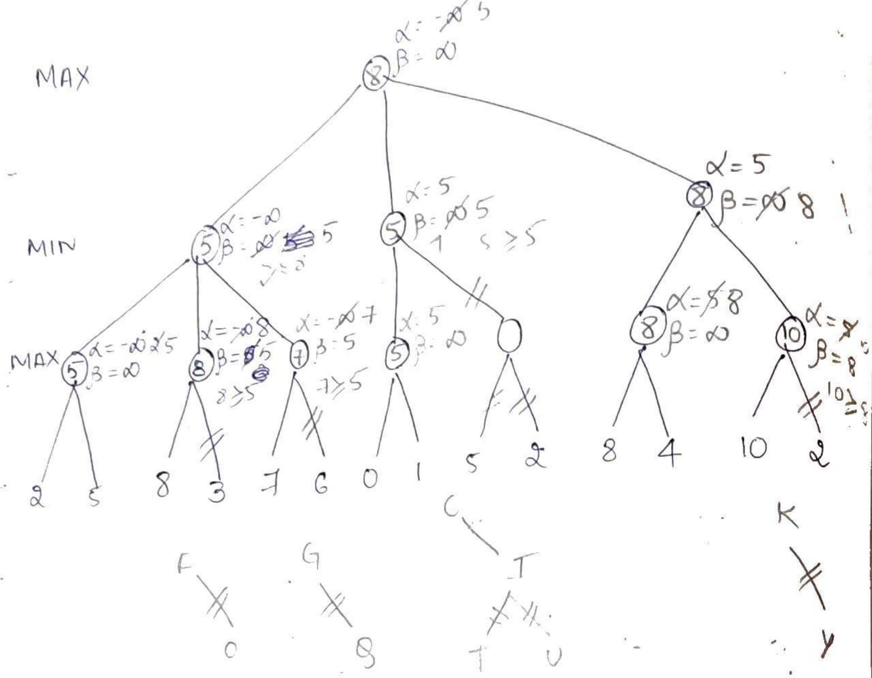
- while deciding parent node ⇒ check no. in child nodes, don't take α, β

(1)



(2)





Constraint Satisfaction

1)
```
  R O A D S
+ C R O S S
-----------
D A N G E R
```

2)
```
  D O N A L D
+ G E R A L D
-------------
  R O B E R T
```

1) Sol^n:
```
  R O A D S          6 2 5 1 3
+ C R O S S   =>   + 9 6 2 3 3
-----------        -----------
D A N G E R        1 5 8 7 4 6
```

16/1/24.

Block World Problem

⇒

```
  b                 a b
a c d               c d
-------             -----
Start state         Goal state
```

/ arm can manipulate ~~the~~ one block at a time

/ block 'a' ... block 'd' } are all of same size

pick b ⟶ put it on top of d

pick a ⟶ put it on top of c

⇒ Operations:

UNSTACK ⟶ US (X, Y) } / ~~place~~ pick X which is on top of Y / arm → free

STACK ⟶ S (X, Y) } place X on Y / arm → shud hold X

PICKUP ⟶ PU (X) } X → on table, arm shud pick up.

PUTDOWN ⟶ PD (X)

⇒ Predicates (status)

ON (X, Y) ⟶ O(X, Y) — X is on Y

ONTABLE (X) ⟶ T (X) } (a, c, d) (c, d)

CLEAR (X) ⟶ C (X) } b, c, d } clear ⇒ i.e nthg on its top

HOLDING (X) ⟶ H (X) } robot arm is holding

ARMEMPTY ⟶ AE } robot arm is empty

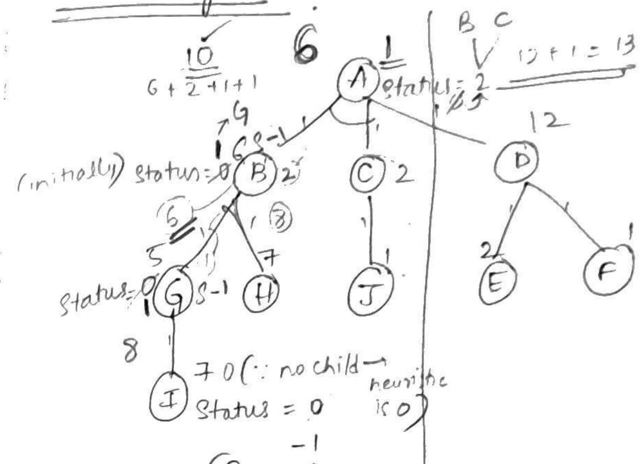
→ Operators with Pre, Del & Add

Operator	Pre List	Del List	Add List
ST(X,Y)	C(Y) ∧ H(X)	C(Y) ∧ H(X)	AE ∧ O(X,Y)
US(X,Y)	O(X,Y) ∧ C(X) ∧ AE	O(X,Y) ∧ AE	H(X) ∧ C(Y)
PU(X)	T(X) ∧ C(X) ∧ AE	T(X) ∧ AE	H(X)
PD(X)	H(X)	H(X)	T(X) ∧ AE

Y should be clear; Arm should be holding X

should be empty to pick it up from stack | X should be clear, for arm to pickup X

AO* Algo



(Solved)

status = 0 (default) {of all nodes}
⇓
no. of child nodes
↓
(Solve in iterations)
↓
write graph

goal ⇒ make status = -1 if (no child node)