# ASSIGNMENT - 2

## Q1. UNIFORM SEARCH :

a) In iteratively lengthening, as we further explore only the paths that lead lowest cost (path cost) at each iteration. Hence, the solution found will be optimal as only the shortest paths as per path costs are explored.

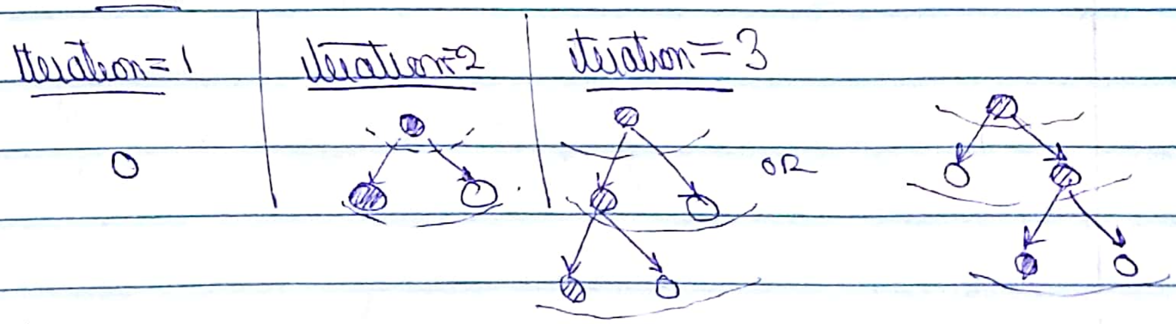
b) b → branching factor
d → solution depth
unit step costs

As per iterative lengthening algo, it compares all frontiers and explores the lowest path cost and as the first solⁿ found is optimal

$\Rightarrow$ no. of iterations req. = $d+1$

eg

d=2




NOTE

As unit step cost, it will operate like BFS

c) Step cost $\longrightarrow [\varepsilon, 1] \quad \forall \; 0 < \varepsilon < 1$

goal node at depth 'd'

worst cost $= d$.

and cost $= \varepsilon x =$ cost $\times$ no. of iteration without root.

$$\Rightarrow x = \frac{d}{\varepsilon}$$

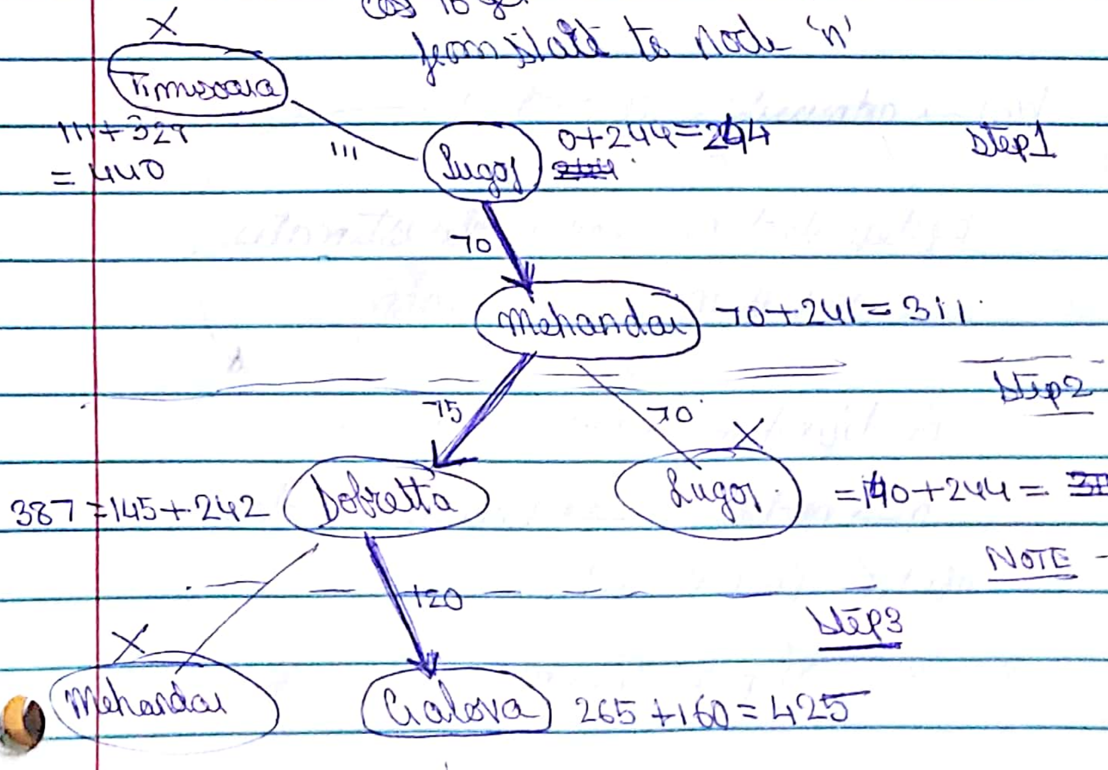
$\Rightarrow$ Total no. of iteration $= \frac{d}{\varepsilon} + 1$ checking root node.

## Q2 INFORMED SEARCH

$f(n) = g(n) + h(n)$ used this terminology

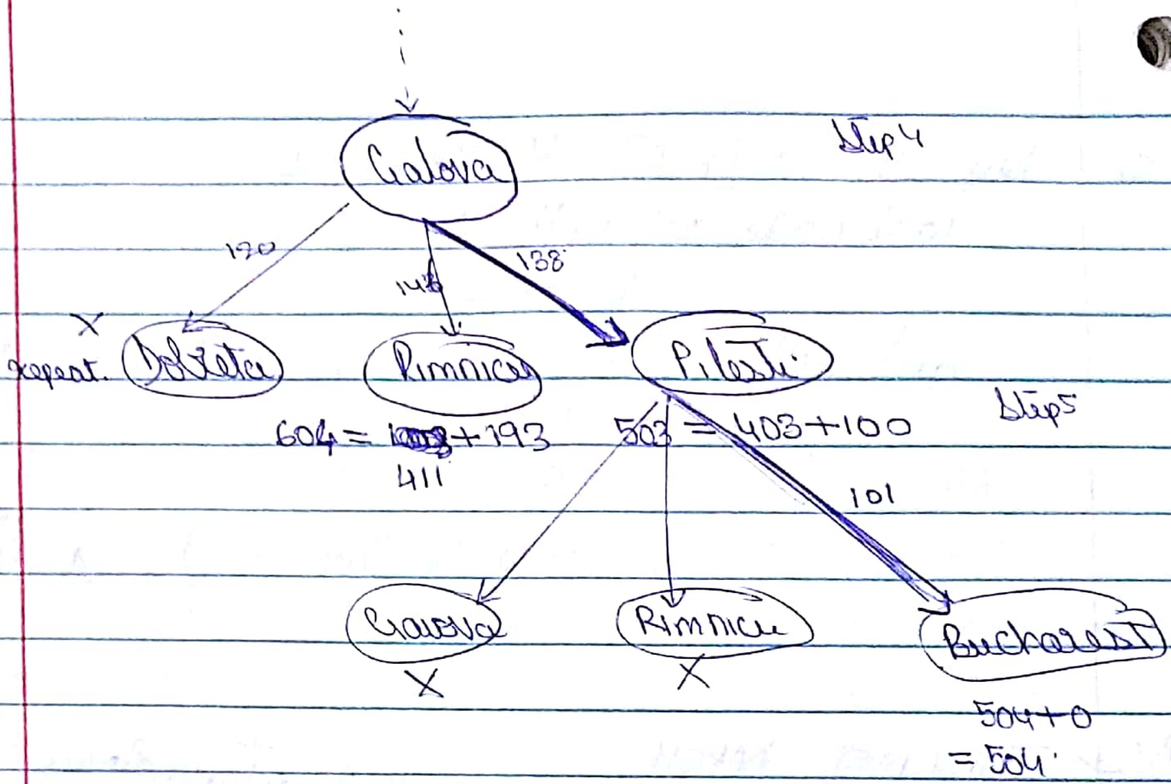
$g(n)$: cost to get from start to node 'n'

$h(n)$: straight line heuristic




CONTD NEXT PAGE

NOTE - all loop repetitions are directly avoided in A* algorithm & we won't calculate here




Lugoj → Mehandai → Dobreta → Calova

Bucharest ← Pitesti

Q3 $h(n) \rightarrow$ admissible, consistent

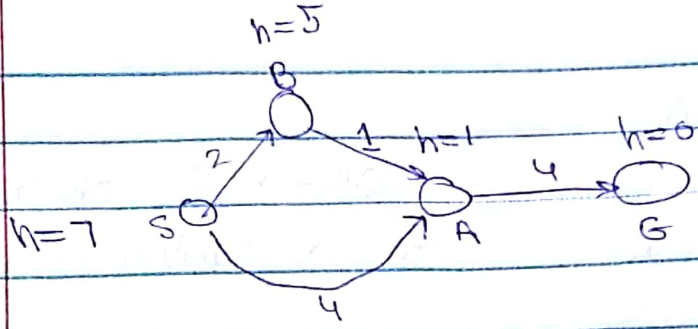
by definition $h(n)$ never overestimates and is an underestimator

By definition, consistency means:

$n \rightarrow$ node, $n' \rightarrow$ successor node

$h(n) \le c(n,a,n') + h(n')$

stepcost $(n \rightarrow n')$




looking at it, $h(n)$ is never more than the actual remaining path cost

⇒ It is admissible.

## CONSISTENCY

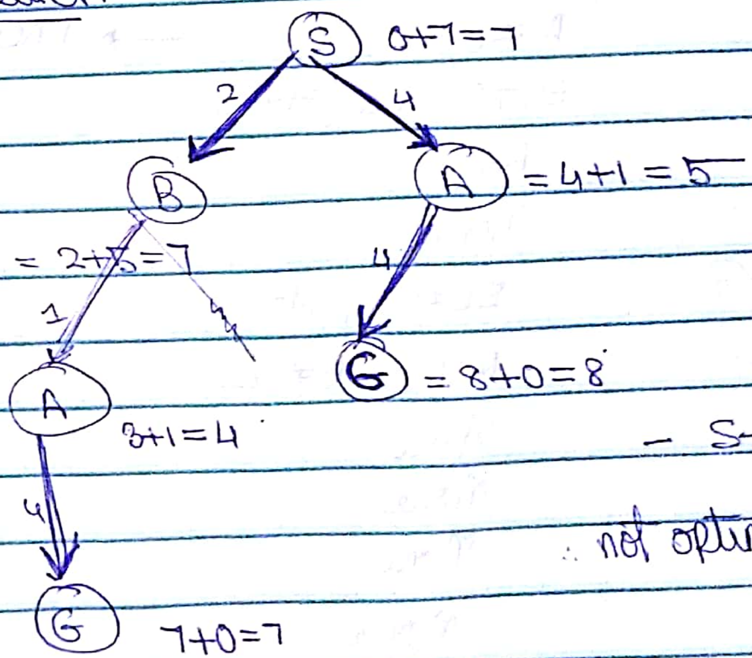
BAG $\quad c(BA) + h(A)$

$h(B) = 5 > 1 + 1 = 2$

doesn't match with definition

∴ not CONSISTENT

## A* search




A* algo

I $S \to A$

II $A \to G$

total cost = 8

OPTIMAL PATH

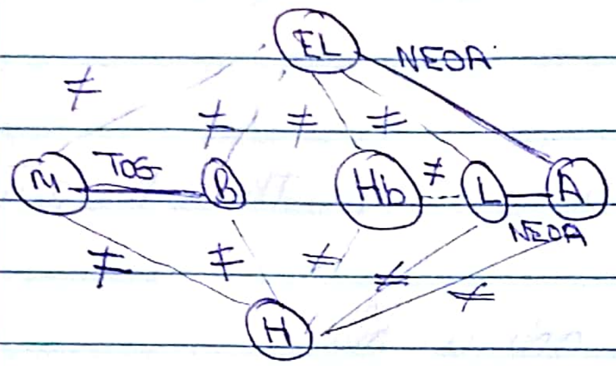
– $S \to B \to A \to G$

∴ not optimal in directed graphs

Q4 CSP PROBLEM

NEOA → not = OR adjacent

TOG → Together




L → 1

Hb → 2 3 4

A → 3 4

EL → 2 3 4

H → 2 3 4

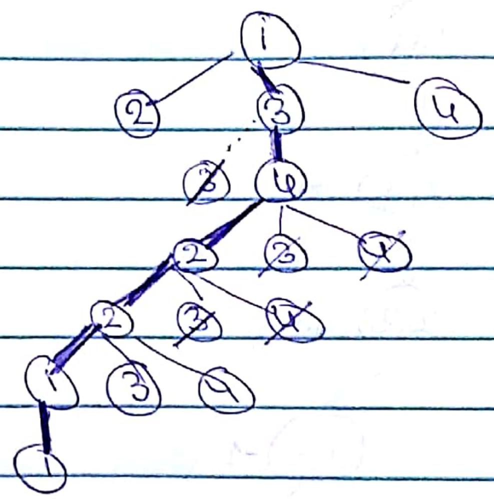
M → 1 2 3 4

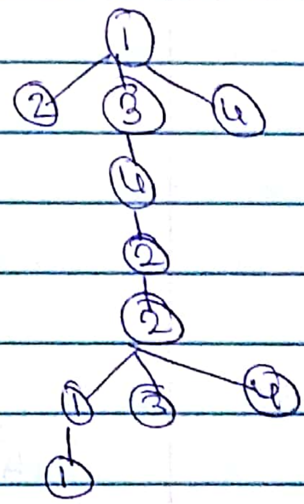
B → 1 2 3 4.

| | Selection | Eliminated conditions |
|---|---|---|
| 1 | L=1 | None |
| 2 | Hb=2 | EL≠2 and H≠2 |
| 3 | EL | A≠3,4. → BACKTRACKING HERE |
| 4 | Hb=3 | EL≠3 and H≠3 |
| 5 | EL | A≠3 |
| 6 | H | None |
| 7 | A | EL≠4, H≠4 |
| 8 | EL | M≠2, B≠2 |
| 9 | H | None |
| 10 | B | None |
| 11 | M | None |
| 12 | A=4 | None |
| 13 | EL=2 | None |

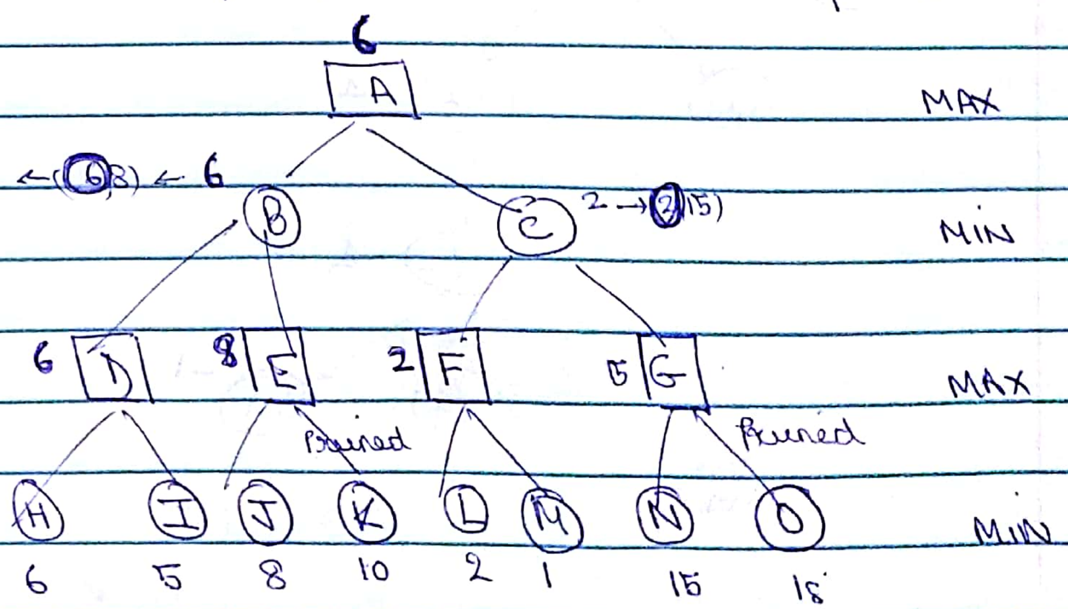
| | | |
|---|---|---|
| 14 | H=2 | None |
| 15 | M=1 | $B \neq 3$, $B \neq 4$. |
| 16 | B | None |
| 17 | B=1 | None |

LION
HORNBILL
ANTELOPE
EVIL LION
HYENA
MEERKAT
BOAR

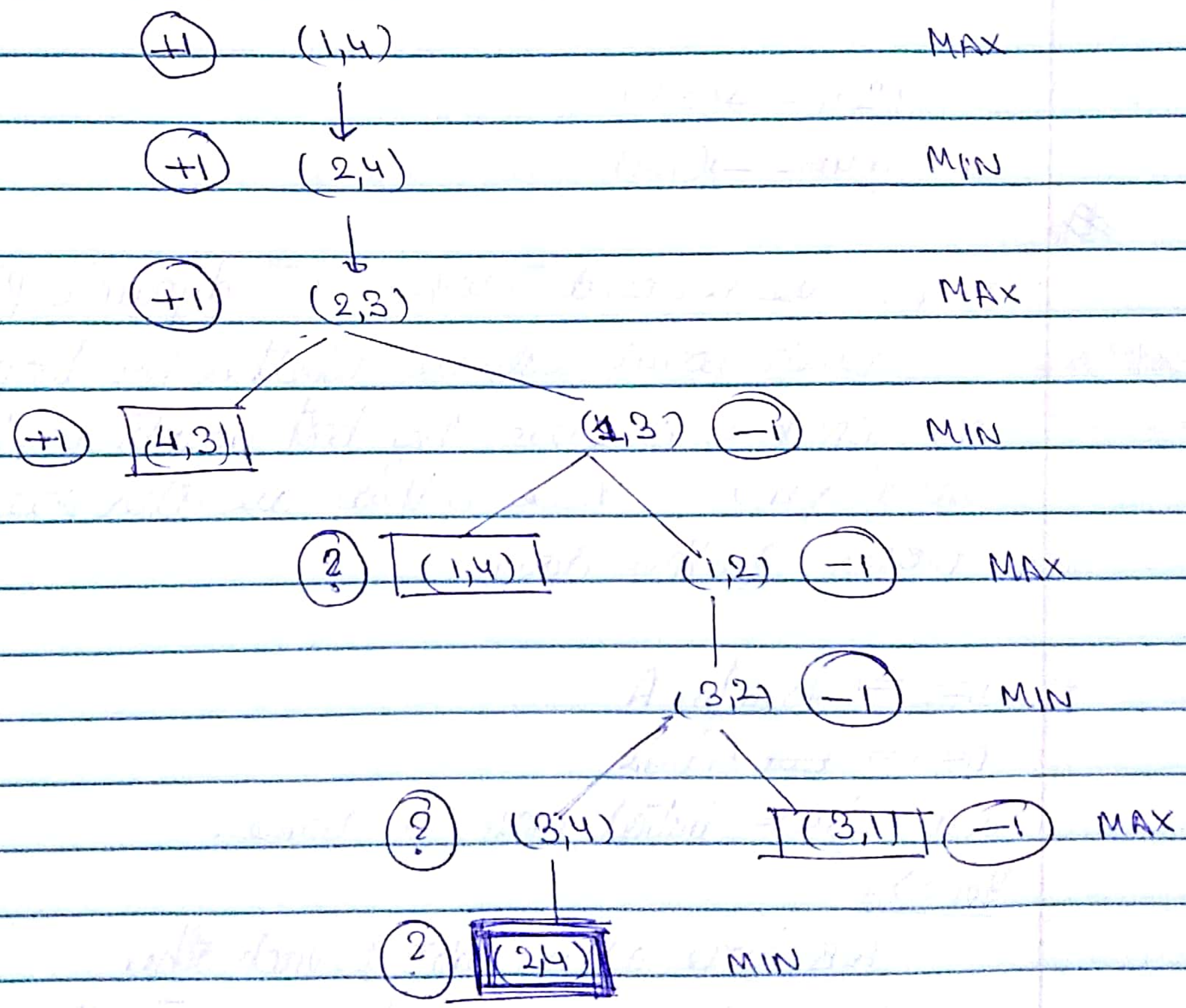




Q5



Q6 a) ADVERSARIAL SEARCH

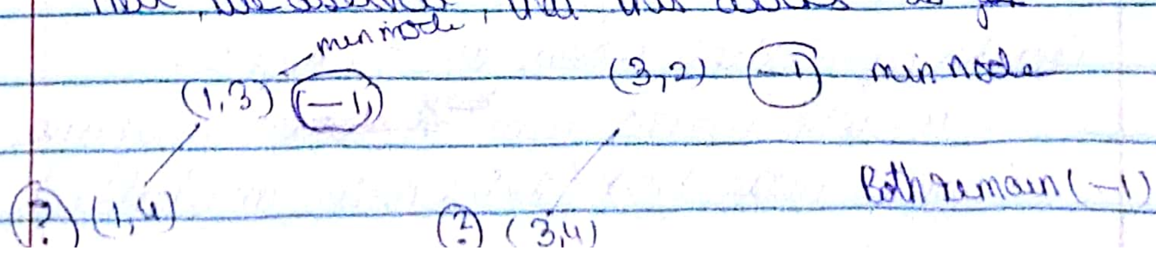



b) In maxmin algorithm

max(?,?) = 1, for max nodes; max(?, +1) = 1

min(?,?) = −1, for min nodes min(?, −1) = −1

Here, we observed that this works as ~~for~~



c) Although min-max resolves ② issue, MIN-MAX doesn't work for loops and in this case we have loops in search tree.

(2,4) ⟶ (2,4)

(1,4) ⟶ (1,4)

In this case, we need to implement dynamic programming which would evaluate whether the loop is one which is preferred by both agents implying end of game or check if there are other possible moves further down.

d) $n=3 \Rightarrow$ loss for A

$n=4 \Rightarrow$ ~~###~~ A wins

and $n>4 \Rightarrow$ initial moves are same.

For $n>4$

A,B move a step towards each other

$\Rightarrow$ $n-2$ subgame on [2 to $n-1$] with multiple moves at 2 and $n-1$

If we ignore multiple moves,

A wins game for "n" if A wins "n-2" games

~ B wins game for "n" if B wins "n-2"

Considering multiple moves at $n-1$ and 2,

A,B who would win ~~would~~ subgame won't move back (extra move available).

Exploring if loser takes the extra move, the loser will still lose as other player continues the original path. making it a subgame of $n-2a$ which loser will lose.

Observing from above, we note that

if $n \rightarrow$ even $\longrightarrow$ A wins
$n \rightarrow$ odd $\longrightarrow$ A loses

extrapolated for
$n=3 \longrightarrow$ A loses
$n=4 \longrightarrow$ A wins