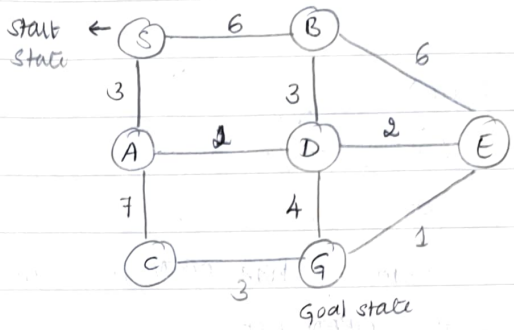
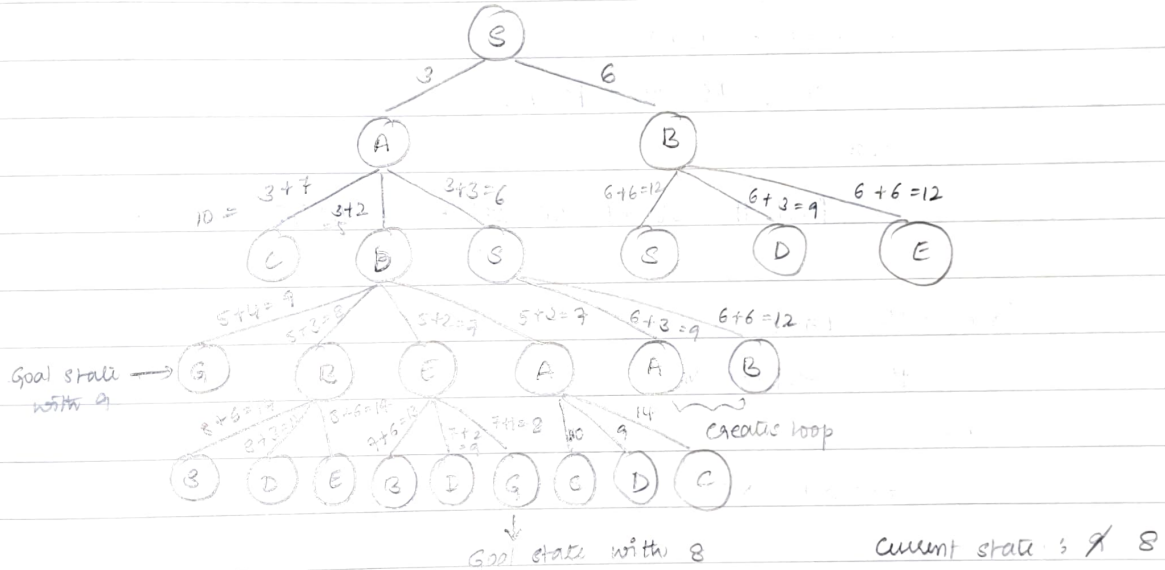
Heuristic search technique : (Informed Search)

Branch and Bound :





optimal path : S → A → D → E → G

Beam search (BEAM search) :

width - $\beta = 2$ which indicates number of nodes to be explored at each level.

Level - B

Algorithm :

Input - start and goal states, ~~local variables~~

Local variables - OPEN, NODE, SUCCESSOR, W_OPEN, FOUND

Step 1 :

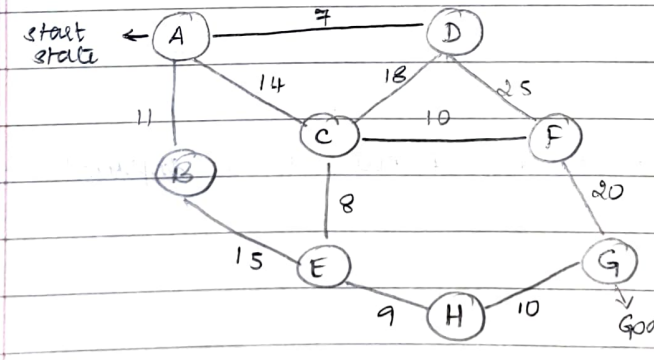
NODE = ROOT_NODE, FOUND = false

If NODE is the goal node then FOUND = true ~~else~~

else find SUCCESSOR of NODE and store it in OPEN list.

Step 2:

```
while(FOUND = false)
do{
    sort OPEN list
    select top 'w' elements from OPEN list and put it in w-OPEN list and
        empty the OPEN list
    for each NODE from w-OPEN list{
        if(NODE == goal)
            then FOUND = true.
        else
            find the SUCCESSOR of those NODE and store it in OPEN list
    }
    if(FOUND = true)
        then return YES
    else
        return NO
}
```



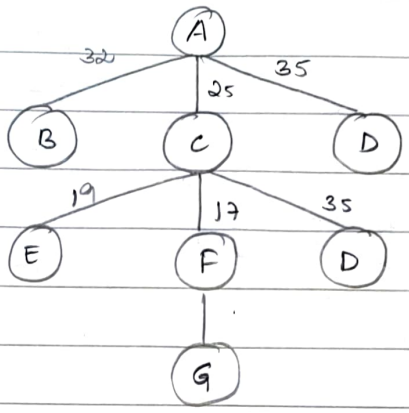
Straight line distance

A → G = 40	E → G = 19
B → G = 32	F → G = 17
C → G = 25	H → G = 10
D → G = 35	G → G = 0

$\beta = 1$

Start traversing from start state A with the help of priority queue.

Beam search gives best solution but not optimal solution.



Path : A → C → F → G.

A	B	C	D
A	C	B	D

→ Prioritised

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

→ Prioritised

A	C	F	E	B	D	G
A	C	F	G	E	B	D

→ Prioritised

3/12/24

Best first search : (Backtracking is not allowed)

→ It is advancement of beam search wherein it considers only one node at each level.

→ Time and space complexity is reduced.

→ It does not give optimal solution.

Algorithm :

Input : Start and goal node

Local variable : OPEN, CLOSE, NODE, FOUND

Step 1 : Initialize OPEN-LIST by ROOT node, CLOSE = ~~ROOT~~ NULL, FOUND = false

While (OPEN != NULL && FOUND = false) {

do {

if the first element = Goal node, ~~FOUND = false~~

then FOUND = true

else

remove it from OPEN_LIST and put it in CLOSED_LIST

add its successor if any in the OPEN_LIST,

sort the entire list by the value of some heuristic function that is assigned to each node to reach the goal node.

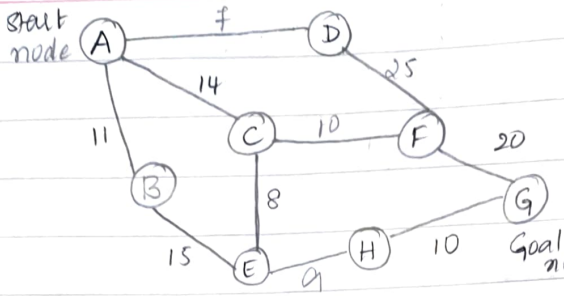
}

if FOUND == true, then

return yes

otherwise

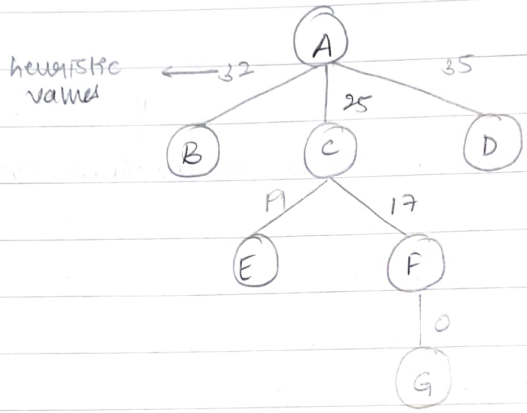
return no



Heuristic value (straight-line dist.)

G → G = 0	D → G = 35
A → G = 40	E → G = 19
B → G = 32	F → G = 17
C → G = 25	H → G = 20

Note: The node which has heuristic value as 0 is considered to be the goal node.



Priority

open list: ~~A~~ | B | C | D → C | B | D

close list: A | C

close list: A | C | F ← prioritised

open list: B | D | E | F → F | E | B | D

close list: A | C | F | G ← prioritised

open list: E | B | D | G → G | E | B | D

Path of solution: A → C → F → G

→ The total cost of path ACFG is 44 which is not optimal but the best.

→ The optimal path will be ACEHG where total cost is 41.

A* Algorithm (Astar):

admissible (acceptable)

→ Will definitely get optimal and best solution.

→ It has both node to node distance as well as heuristic value.

→ Backtracking is also allowed.

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

f(N) is the estimated cost of the optimal solution from start to goal node.

g(n) is the cost of to reach node 'n' from start state. i.e., actual cost from starting node to node 'n'.

h(n) is the cost to reach node 'n' to goal node i.e., the estimated cost to move from 'n' to goal node which is refereed as heuristic value.

h(n) = number of value that is different from that of goal state values.

Date 3/12/24
Page 29

3	7	6
5	1	2
4		8

start state

5	3	6
7		2
4	1	8

Goal state

level 0

3	7	6
5	1	2
4		8

$f(n) = g(n) + h(n)$ ← value of the level

$f(A) = 1 + 3 = 4$

$f(B) = 1 + 5 = 6$

$f(C) = 1 + 5 = 6$

h(n) = 3 → A; h(n) = 5 → B; h(n) = 5 → C

level 1

A

3	7	6
5		2
4	1	8

B

3	7	6
5	1	2
	4	8

C

3	7	6
5	1	2
4	8	

level 2

D (h(n) = 3)

3	7	6
	5	2
4	1	8

E (h(n) = 4)

3	7	6
5	2	
4	1	8

F (h(n) = 4)

3	7	6
5	1	2
4		8

G (h(n) = 3)

3		6
5	7	2
4	1	8

$f(D) = 2 + 3 = 5$

$f(E) = 2 + 4 = 6$

$f(F) = 2 + 4 = 6$

$f(G) = 2 + 3 = 5$

level 3

h(n) = 3

	7	6
3	5	2
4	1	8

h(n) = 4

3	7	6
4	5	2
	1	8

h(n) = 3

3	7	6
5		2
4	1	8

h(n) = 2

	3	6
5	7	2
4	1	8

h(n) = 4

3	6	
5	7	2
4	1	8

h(n) = 3

3	7	6
5		2
4	1	8

level 4

h(n) = 1

5	3	6
	7	2
4	1	8

h(n) = 3

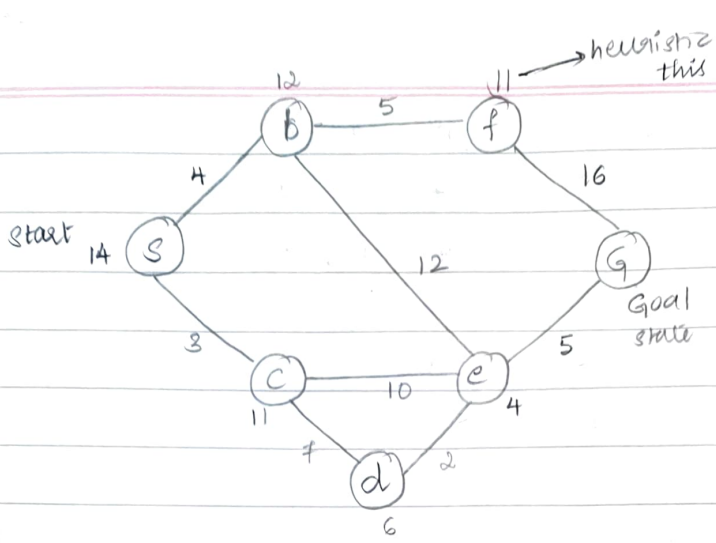
3		6
5	7	2
4	1	8

5	3	6
4	7	2
	1	8

5	3	6
7		2
4	1	8

Goal state

	3	6
5	7	2
4	1	8



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

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

2. $f(S \to B) = 4 + 12 = 16$

   $f(S \to C) = 3 + 11 = 14$

for any node other than source node we need to calculate $f(S \to \text{node})$ + heuristic value.

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

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

$f(S \to B) = 4 + 12 = 16$

$f(S \to C) = 3 + 11 = 14$

$f(SB \to f) = (4+5) + 11 = 20$

$f(SB \to e) = (4+12) + 4 = 20$

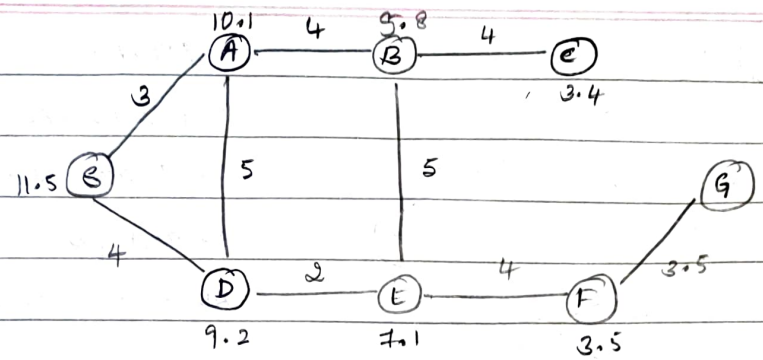
$f(SC \to e) = (3+10) + 4 = 17$

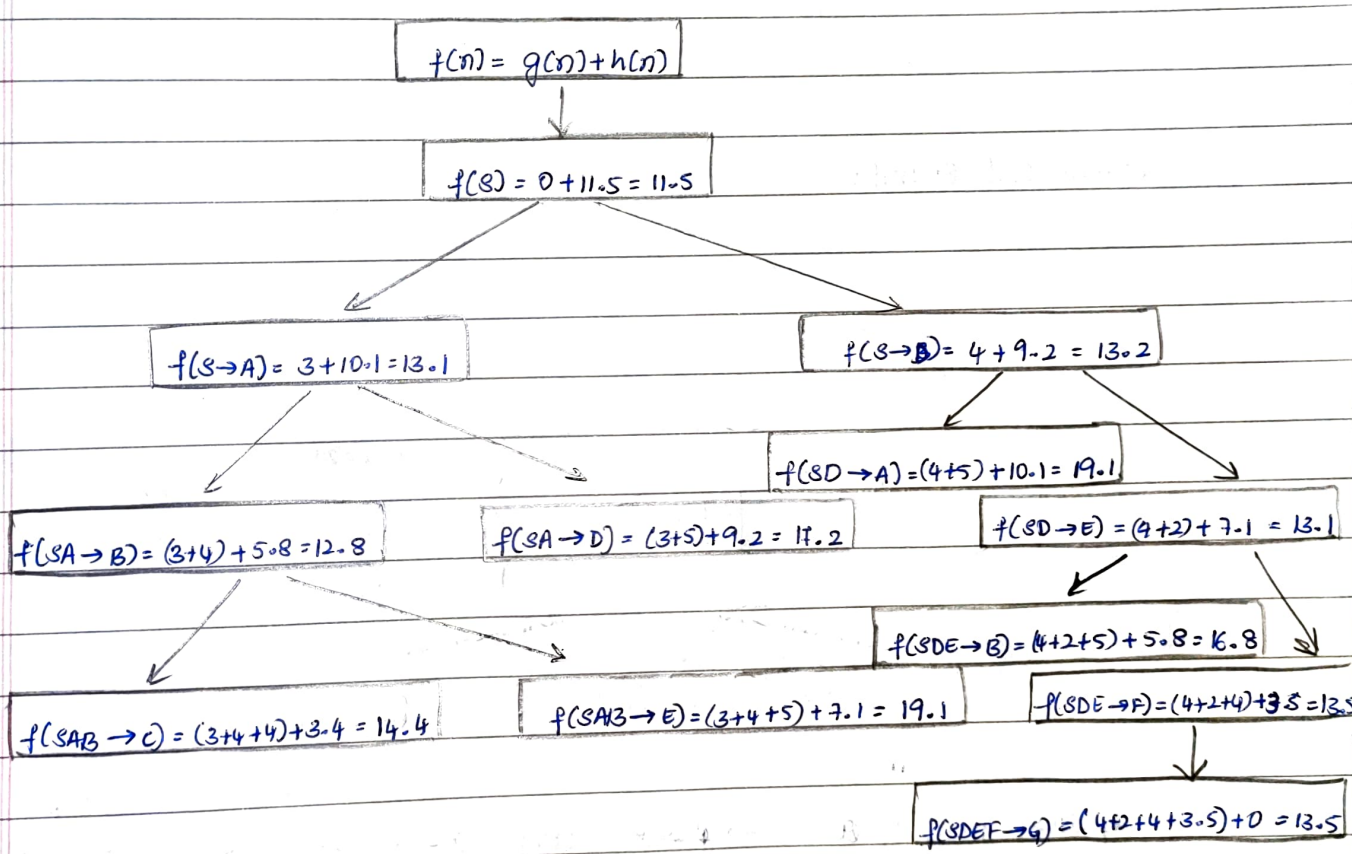
$f(SC \to d) = (3+7) + 6 = 16$

$f(SCd \to e) = (3+7+2) + 4 = 16$

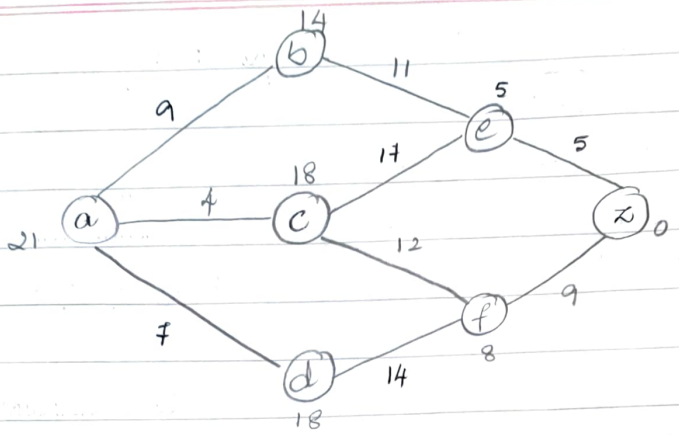
$f(SCde \to G) = (3+7+2+5) + 0 = 17$

The optimal path is $S \to C \to d \to e \to G$ with the heuristic cost of 17.





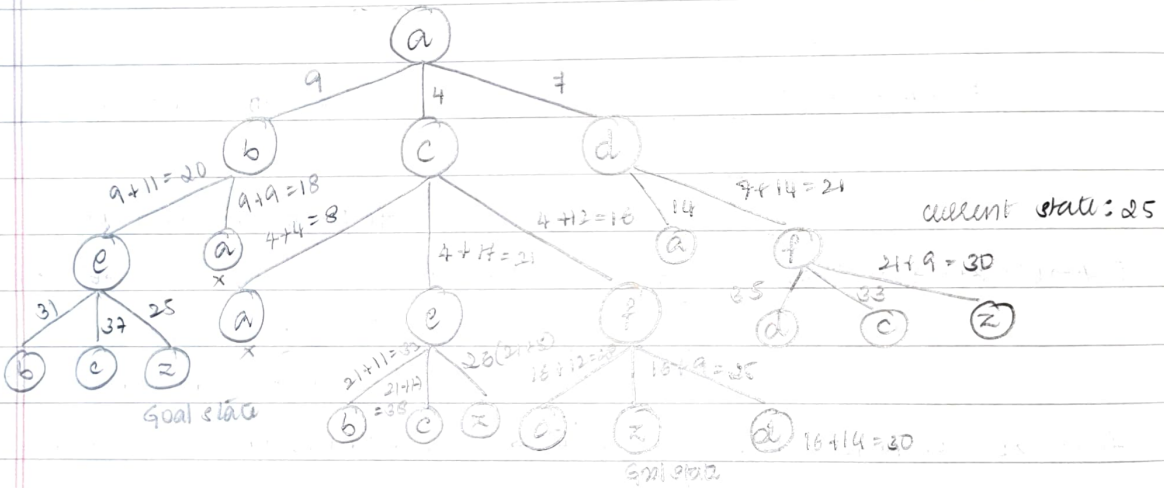
The optimal path is S → D → E → F → G with the heuristic cost of 13.5



Apply :

1) Branch and Bound
2) Beam search with $\beta = 2$
3) Best first search
4) Astar algorithm.

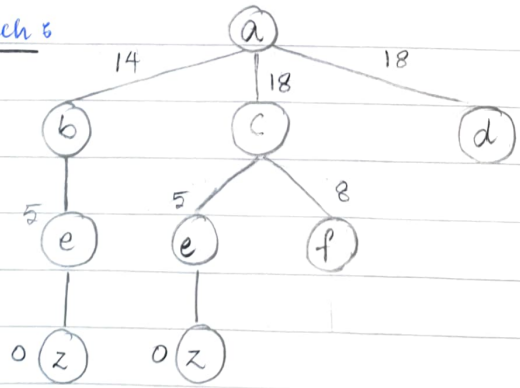
Branch and Bound :



Optimal path : a → c → f → z (or) a → b → e → z with the cost of 25.

Beam Search :

$\beta = 2$



a	b	c	d
a	b	c	d

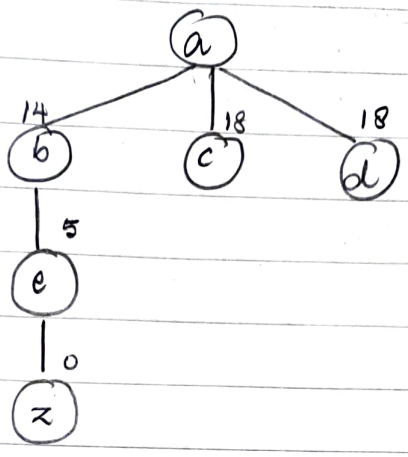
pruned

a	b	e	d	e	f
a	b	e			
a	b	e	f		

a	b	c	d	e	f	z
a	b	e	z			
a	c	e	z			

Optimal path : a → b → e → z
a → c → e → z

Best first search :



open list : | A | B | C | D | prioritized

close list : | A | B |

| a | b | e | c | d | prioritized

open list : | A | B | C | D | e |

close list : | a | b | e |

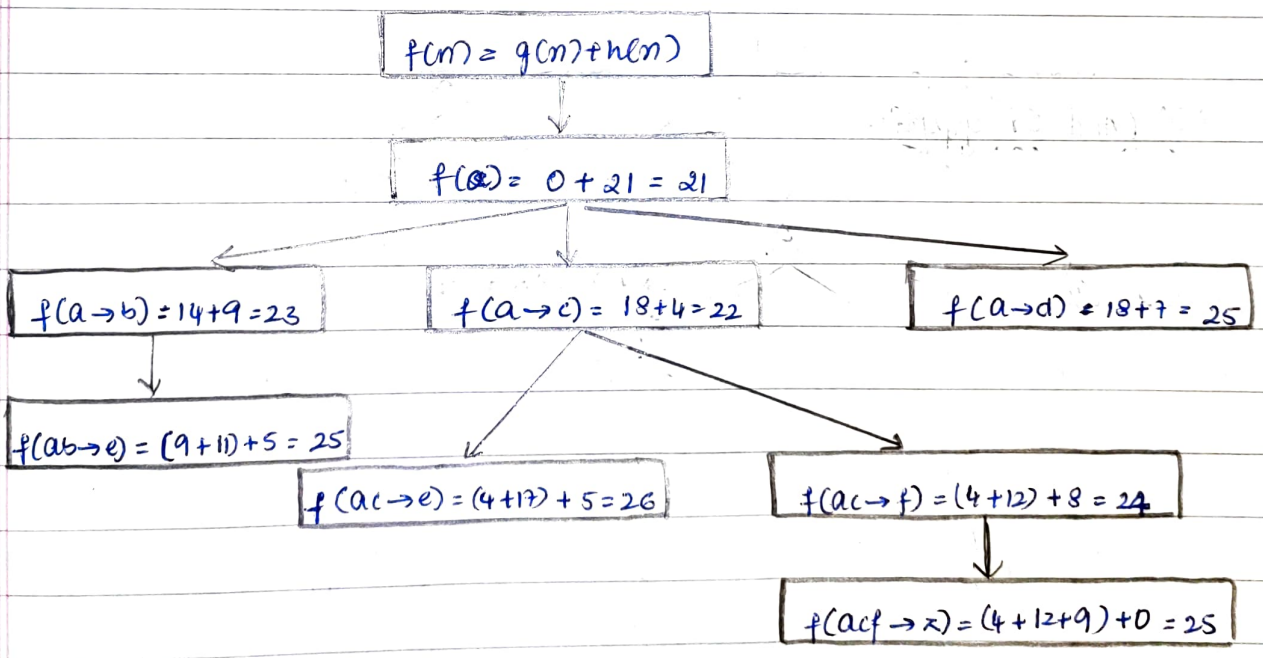
| a | b | e | z | ~~d~~ | d |

open list | a | b | e | c | d | z | prioritised

close list | a | b | e | z |

Path : a → b → e → z

Astar algorithm :

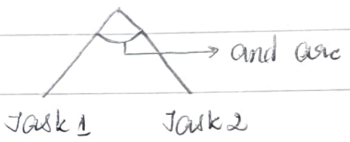


Unit - III : Knowledge Representation & Predicate Logic

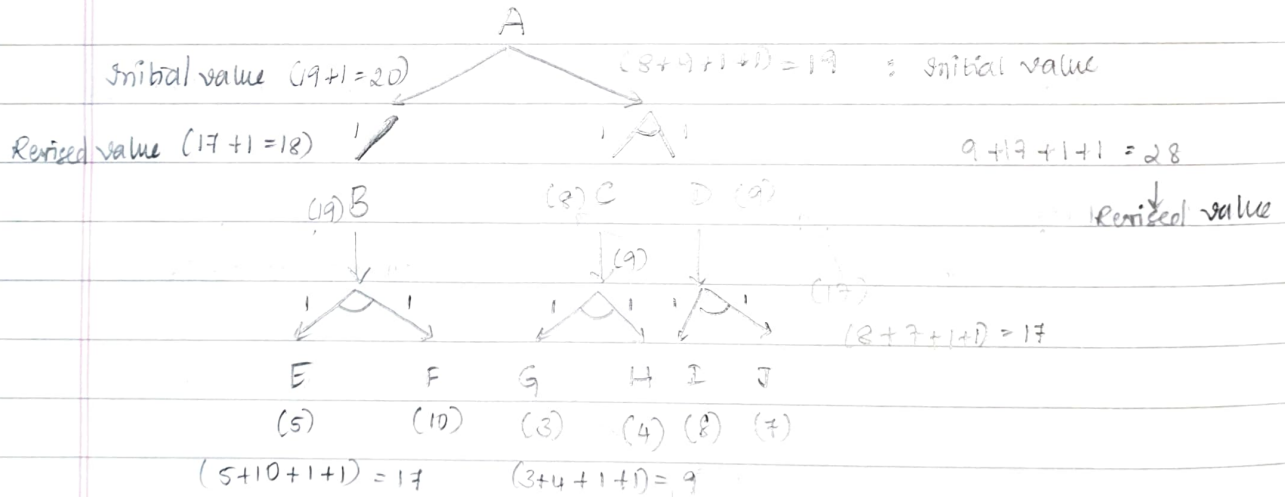
* Game playing :
  i) Min max algorithm
  ii) α-β pruning algorithm.

* Problem reduction :
  - AO* (And or graph)

* Dependent tasks are represented using 'and arc'.



(6-8m) **AO* (And Or graph)**

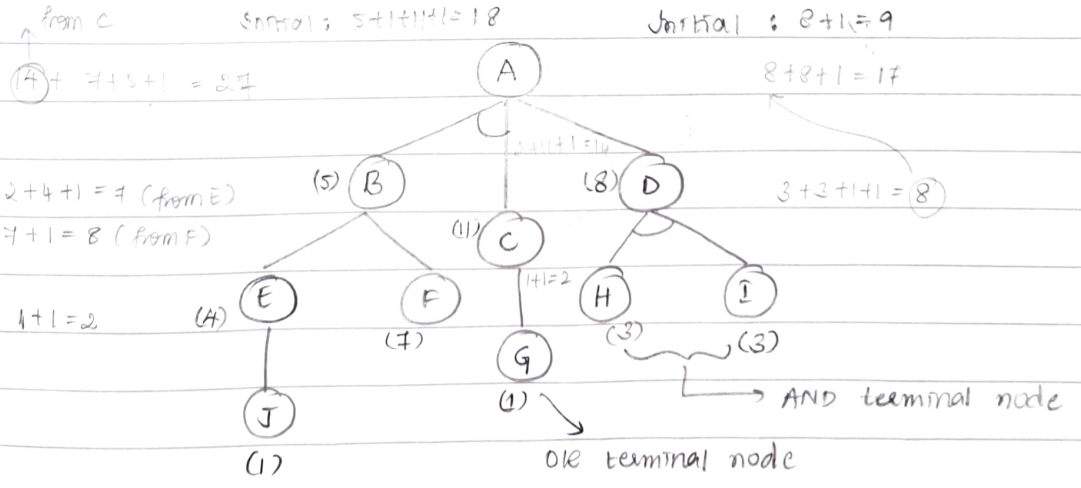


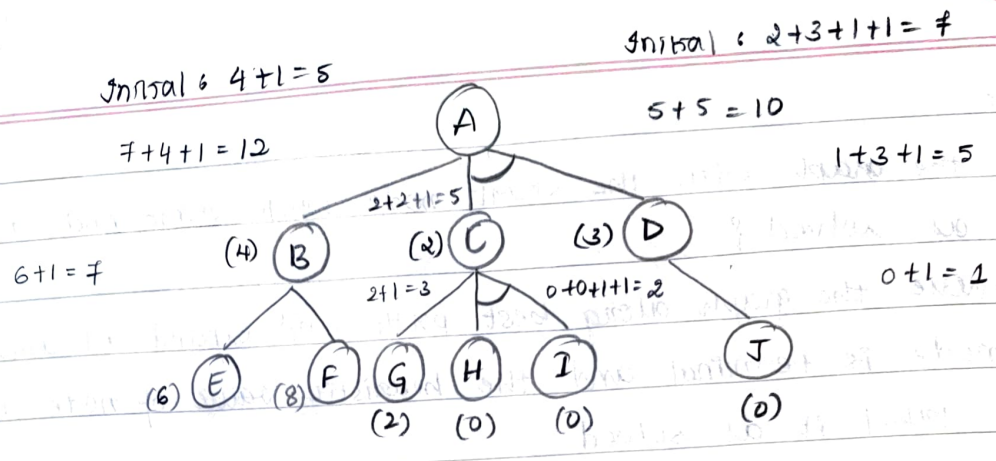
If we choose the path 'B' the cost will be 18.
But if we choose the path 'C' and 'D', then the cost will be 28.
So we choose the path A → B → E → F

Algorithm :

1. Initialize the graph with the start node while start node is not labelled as solved {
   traverse the graph along best path and expand all unexpanded nodes on it.
   if node is terminal and the heuristic value of node is 0,
      label it as solved
   else
      label it as unsolved and propagate the status up to the start node (root node).
   if the node is non terminal and its successor with heuristic value in the graph
      revise the cost of expanded node and propagate this change along the path till start node.
   choose the current best path.
}
if start = solved
   the leaf node of best path from root are the solution nodes.
else
   no solution exists.

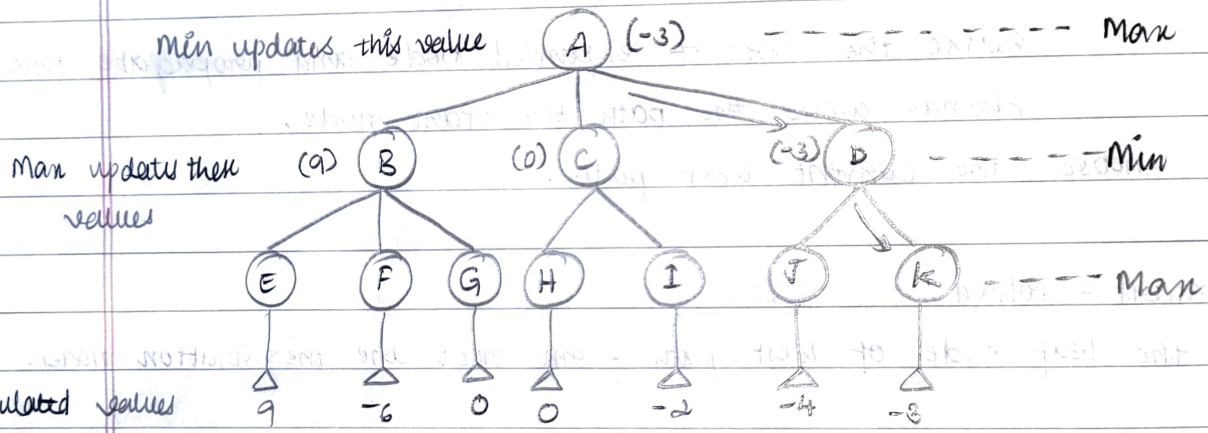
10/12/24



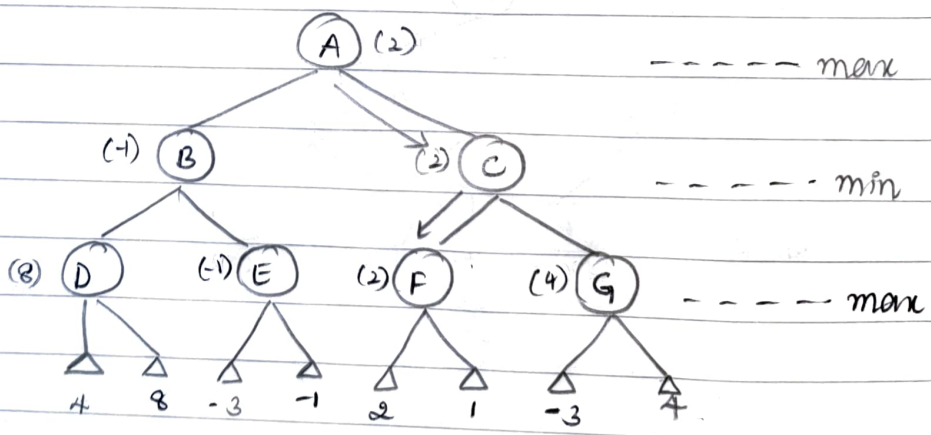


Game Playing :

Min-Max (Miniman) :



The path A → D → K which is the path through which max wins the game



The path is A → C → F.