1 (**) Complexity

If we apply Bidirectional Search with Iterative Deepening the complexity will be as follows

Time Complexity : $O(b^{d/2})$

Space Complexity : $O(b * d)$

where b = branching factor , d = depth of goal node .

2 (**) Search Algorithms

1 . DFS :

Expanded Node	Node List
a	{b}
b	{g,c}
g	{l,h,c}
l	{k,q,m,h,c}
k	{p,q,m,h,c}
p	{q,m,h,c}
q	{v,m,h,c}
v	{w,m,h,c}
w	{x,m,h,c}
x	{s,m,h,c}
s	{t,n,m,h,c}

No Of Nodes Expanded = 11 , **solution path = a -> b -> g ->l -> q ->v -> w -> x -> s**

2 . BFS :

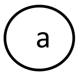
Expanded Node	Node List
a	{b}
b	{g,c}
g	{c,l,h}
c	{l,h,d}
l	{h,d,k,q}
h	{d,k,q}
d	{k,q,e}
k	{q,e,p}
q	{e,p,v}
e	{p,v,j}
p	{v,j,}

v	{j,w}
j	{w,o}
w	{o,x}
o	{x,n,t}
x	{n,t,s}
n	{t,s,s' }
t	{s, s' , s'' }
s	{ s' , s'' }

No of Nodes Expanded = 19 , **Solution Path = a -> b -> g ->l -> q ->v -> w -> x -> s**

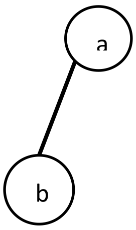
3. IDDFS :

Depth 0 :



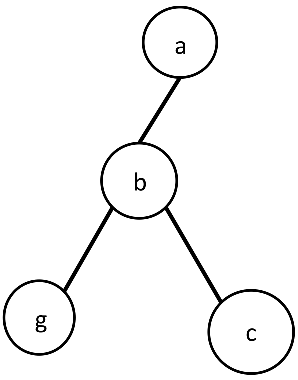
Node Expanded order= a

Depth 1 :



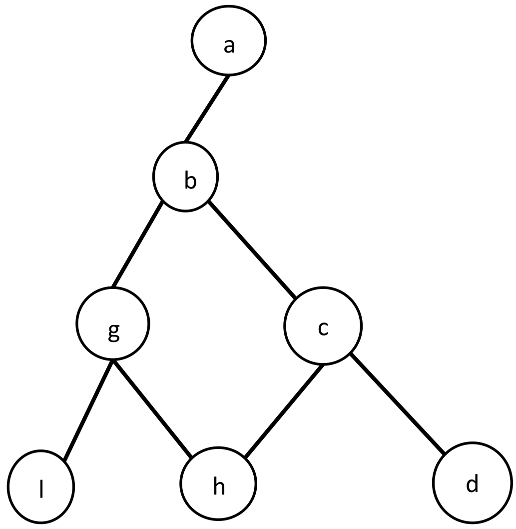
Node Expanded order= a,b

Depth 2 :



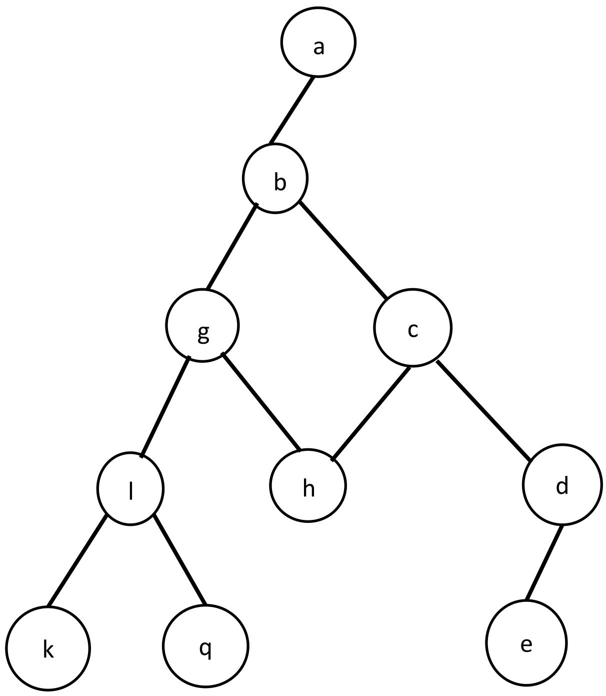
Node Expanded order= a,b,g,c

Depth 3 :



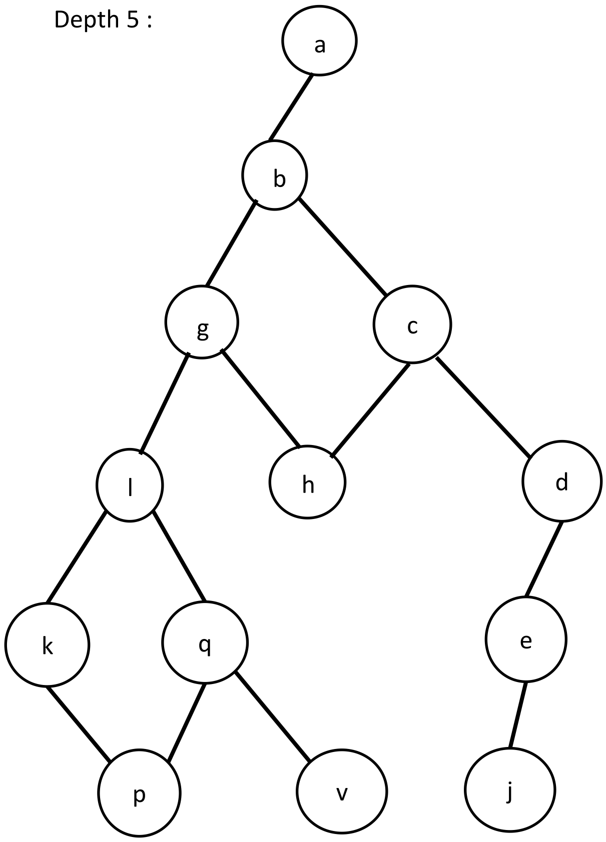
Node Expanded Order = a,b,g,l,h,c,d

Depth 4 :



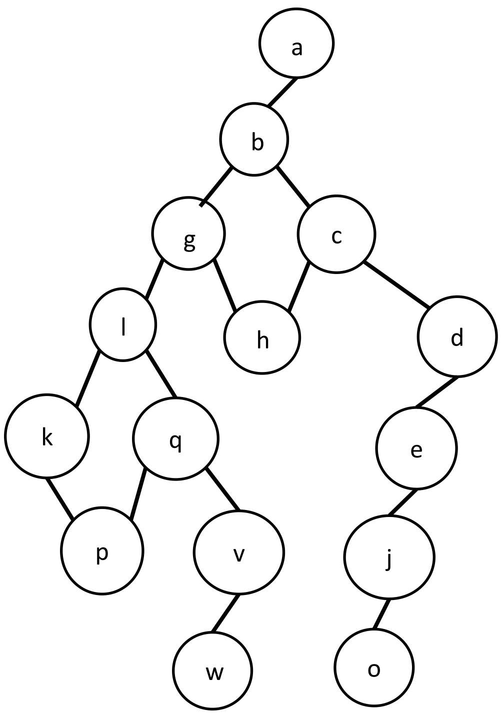
Node Expanded Order = a,b,g,l,k,q,h,c,d,e

Depth 5 :



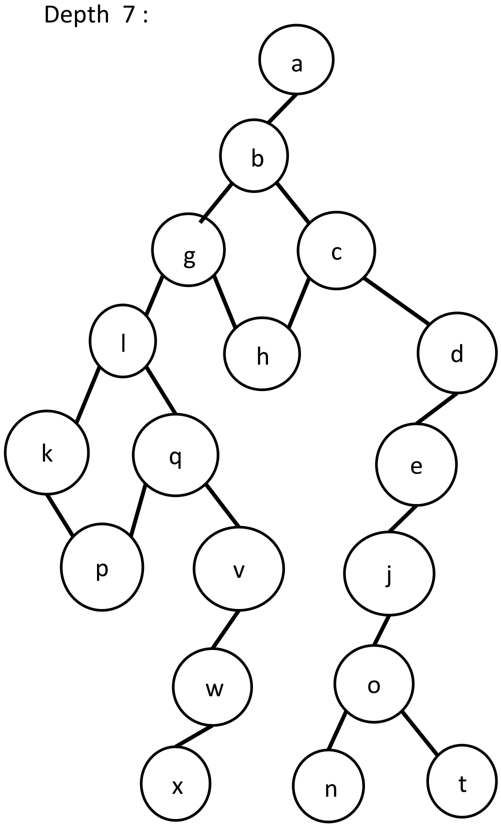
Node Expanded Order = a,b,g,l,k,p,q,v,h,c,d,e,j

Depth 6 :



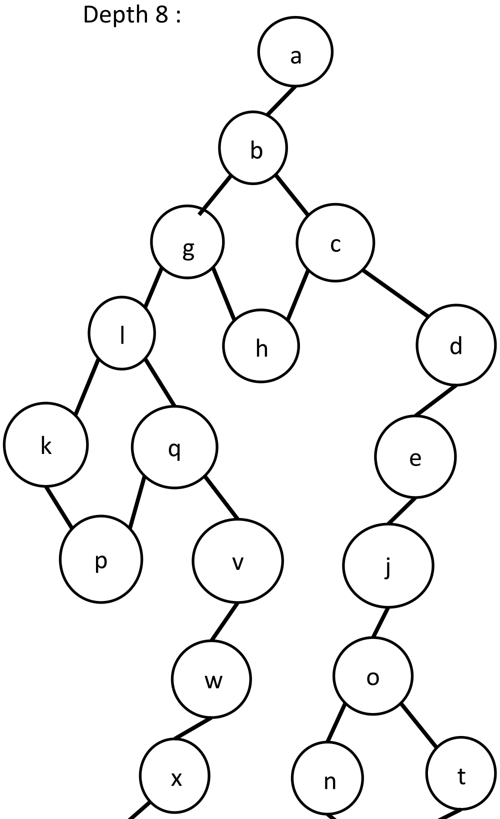
Node Expanded Order = a,b,g,l,k,p,q,v,w,h,c,d,e,j,o

Depth 7 :



Node Expanded Order = a,b,g,l,k,p,q,v,w,x,h,c,d,e,j,o,n,t

Depth 8 :



Node Expanded Order = a,b,g,l,k,p,q,v,w,x,s

Hence the solution path = a -> b -> g ->l -> q ->v -> w -> x -> s

3 (**) A* Algorithm:

1 . The no of steps used to achieve goal state = **16**

2 . considering initial state as state 1 , the 5th state which is 4 steps after the initial state is as follows .

5th state from start :

	2	3	4	5
1		7		8
6	10	11	12	15
9		14		20
13	16	17	18	19

Considering goal state as the last state , the 5th state from the last which is 4 steps before the goal state is as follows .

5th state from last :

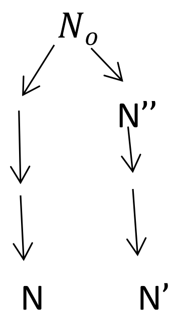
1	2	3	4	5
6		7		8
9	10	11	12	15
13		14		20
16	17	18	19	

3 . Considering initial state as a state being explored , The no of states explored before reaching the goal state are = **23**

P.T.O

4 (**) Optimality of A* Algorithm:

Let us assume that A* algorithm is not optimal and for a given space having nodes as follows.



For the situation above , N_o is the initial node and N is the goal node found through A* algorithm.

Let us assume that there is an actual optimal path leading to the optimal goal node N' . Since N' is the optimal goal node we should have $g(N') < g(N)$. Given that h(s) is admissible hence h () is a monotonic function i.e $h(s+1) < h(s)$.

There are two possibilities for A* algorithm to choose N as the goal node .

Case 1 : N' and N were on the open list and N was picked up by the A* algorithm . In that case cost function f() should hold the relation

$$f(N) \leq f(N') \text{ , where } f(n) = g(n) + h(n)$$

$$g(N) \leq g(N') \text{ , since } h(N) = h(N') = 0 \qquad \text{- (1)}$$

But (1) is contradicting our assumption that $g(N') < g(N)$, hence our assumption is wrong in this case .

Case 2 : N' was not in the open list when N was picked , but there is some ancestor of N' which is N'' was present in the open list and N was picked up by A* algorithm .
In that case cost function f() should hold the relation .

$$f(N) \leq f(N'') \text{ , where } f(n) = g(n) + h(n)$$

$$g(N) \leq g(N'') + h(N'') \qquad \text{- (2)}$$

Since N'' is an ancestor of N' and h (s) is an admissible function we have the equation .

$$g(N'') + h(N'') \leq g(N') \qquad \text{- (3)}$$

Substituting (3) in (2) we have $g(N) \leq g(N'') + h(N'') \leq g(N')$

Which will again land us to the equation $g(N) \leq g(N')$ -(4)

Again (4) is contradicting our assumption that $g(N') < g(N)$

Since both the cases are contradicting our assumption , our assumption that there exists an optimal path other than found through A* algorithm is False .

Hence we can conclude that A* algorithm is Optimal