Search Techniques

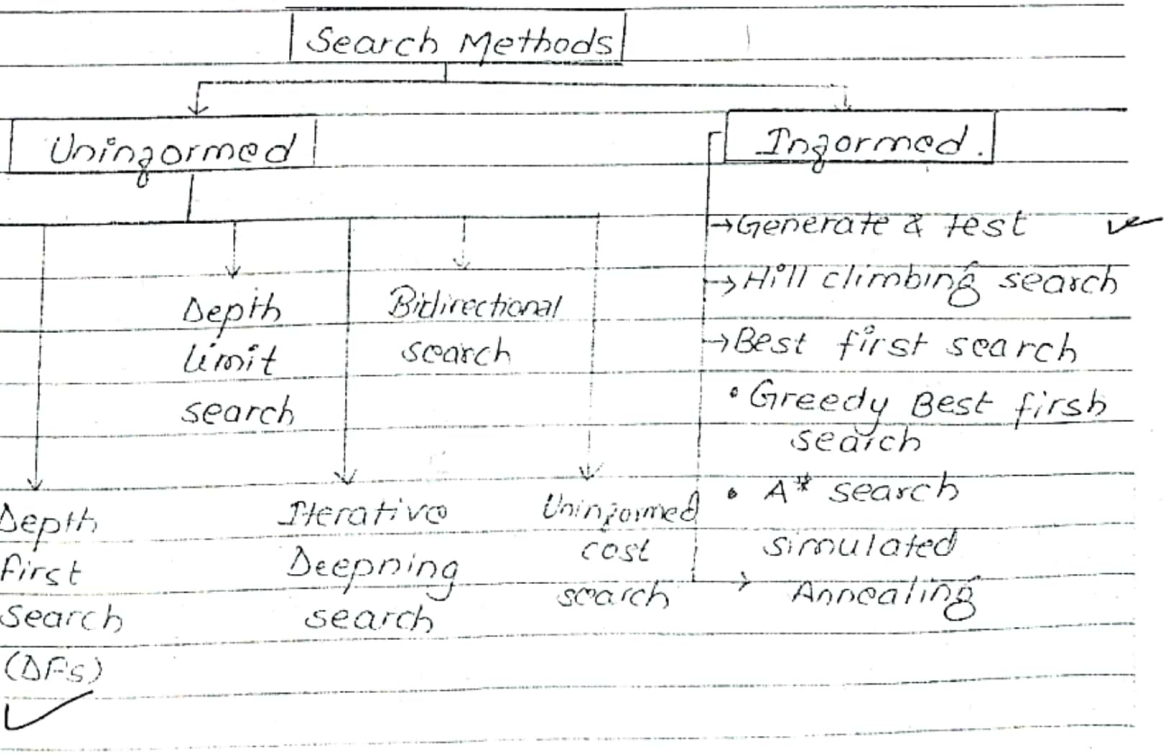
Chapter 3

⇒ Search techniques are used in many real life problem solving approaches like theorem proving, game playing, expert system and natural language processing.

⇒ They include the task like detection, ingerencing, planning, reasoning logic, etc.

Types of searching

1. Uninformed search / Blind search / Unguided search / Brute force search.
2. Informed search / Guided search / Heuristic search.



Search Terminology

1. Space Complexity

The maximum no. of nodes that are stored in memory

2. Time Complexity

The maximum no. of nodes that are created.

3. Admissibility

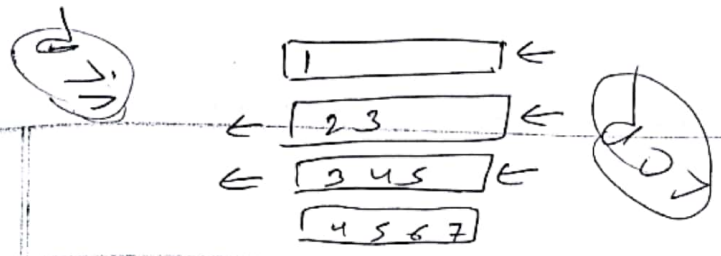
A property of an algorithm to always find an optimal solution.

1. Uninformed search

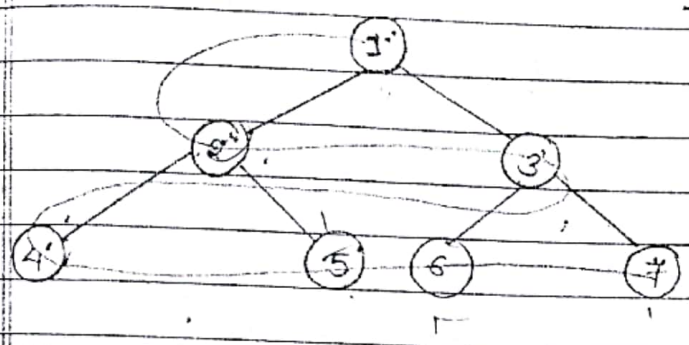
⇒ The search algorithm that do not use any extra information regarding the problem.

⇒ There is no order in which solution paths are consider that is it uses no domain specific information to search a solution in the search space

⇒ It is also called Brute Force Search because it examines every node in the tree until a goal is found.



a) Breadth First Search



→ ~~1~~
→ ~~2~~ 3
→ ~~3~~ 4 5
→ 4 5 ~~6~~ 7

⇒ Proceed level by level down the search tree.
⇒ Starting from the root node (initial state) explore all children of the root node from left to right.
⇒ If no solution is found expand the first (left most) child of the root node, then expand the second node & so on.
⇒ BFS can be done with the help of queue i.e. FIFO.
⇒ It is slower than DFS & requires more memory than DFS.
⇒ It is useful in finding shortest path in spanning tree.

Imp Algorithm of BFS

Step 1: Put the start node on a list called OPEN of unexplored nodes.
If the start node is a goal node, a solution has been found.

Step 2: If (OPEN ← EMPTY) or (OPEN ← Goal) search is terminated.

Step 3: Remove the first node 'A' from OPEN and place it in a list called CLOSED of the expanded node.

Step 4: Expand node 'A' if it has a sucessor go to step 2.

Step 5: Place all successor of node 'A' is a goal state then solution is found.

Note:-
b = branching factor
d = depth of shallowest solution.
m = maximum depth of the search tree.
L = l is the depth limit.

1 Completeness

Complete if the goal is at finite depth.

2. Optimality

It is guarantee to find the shortest path.

3. Time complexity: $O(b^{d+1})$ → branch factor d = depth

4. Space complexity: $O(b^{d+1})$.

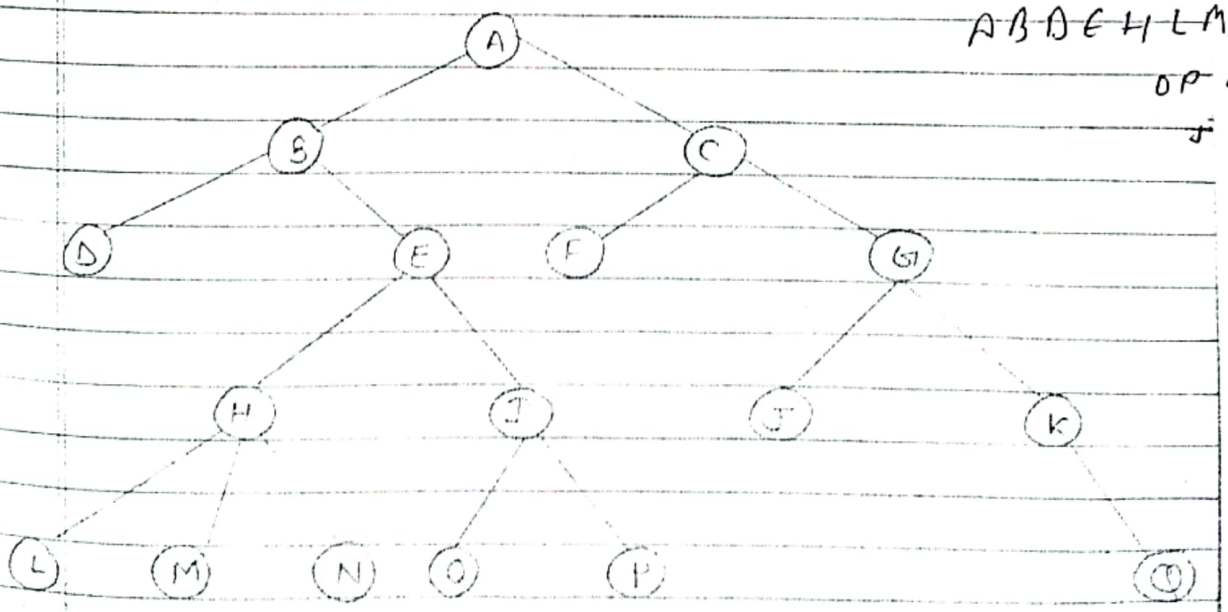
Advantages

1. BFS will never get trapped exploring the useless path forever.
2. If there is a solution BFS will definetly find it out.

Disadvantages

1. Memory requirement is quite high so can exhaust the computer memory within minutes.
2. If the solution is garther away from the root BFS will consume a lot of time.

(b) Depth First search



ABDEHLMNIOPCFGJKQ

fig - DFS

ABDEHLMNIOPCFGJKQ

⇒ Proceed down a single branch of a tree at a time.

⇒ Expand the root node than the left most child of the root node.

⇒ Only when the search hits the dead end the search backtrack & expand nodes at higher level.

⇒ DFS can be done with the stack i.e. LIFO.

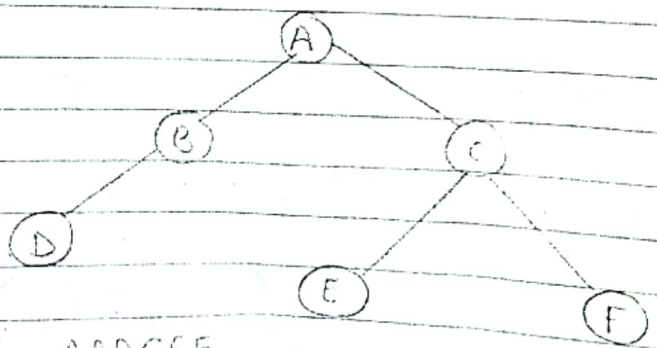
⇒ DFS is faster than BFS & require less memory as compared to BFS.

⇒ It is not useful in finding the shortest path.

⇒ This algorithm works in two stages; at first the visited nodes are pushed onto the stack and later on when there is no vertex further to visit they are popped off.

Application of DFS

⇒ Cycle detection
⇒ Connectivity testing
⇒ Finding some spanning tree.



ABDCEF

Algorithm

Step 1: Put the initial node on a list called START_LIST.

Step 2: If START_LIST is empty or START_LIST = Goal and terminate the search.

Step 3: Remove the first node from START_LIST called this node A.

Step 4: If A = Goal then terminate the search with success; else if node A has successors generate all of these and add them at the beginning of START_LIST.

Step 5: Go to step 2.

Advantages:

1. Memory requirement in DFS are less as only node on the current path are stored.
2. DFS may find the solution without examine much of the search space at all.

Disadvantage

Completeness: Incomplete as it may get stuck going down on a infinite branch that doesnot lead to solution.

Optimality: The first solution found by the DFS may not be the shortest.

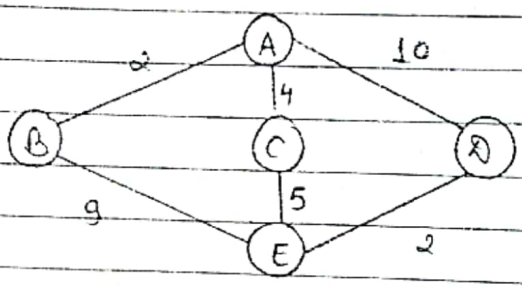
Time Complexity: $O(b^m)$

Space Complexity: $O(bm)$

c) Uniformed Cost Search (UCS)

⇒ Uniformed cost search is guided by path cost rather than path length like BFS.

⇒ The algorithm starts by expanding the root, than expanding the node with the lowest cost from the root, the search continues in this manner for all nodes.



Root A

AB = 2

AC = 4

AD = 10

Completeness:- Complete if the cost of each step exceeds some small positive integer.

Optimality:- Optimal in the sense that the node that it always expands is the node with least path cost.

Here, C be the cost of the optimal solution & assume that every action cost atleast e.

Time complexity: $O(b^{C/e})$

Space complexity: $O(b^{C/e})$

(d) Depth limit Search (DLS)

⇒ The main disadvantages of DFS is that it will not find a goal if it searches down a path that has infinite length. So, DFS isn't guarantee to find a solution so it is not complete.

⇒ This problem can be eliminated by limiting the depth of the search to some value. However this introduces another way of preventing BFS from finding the goal; if the goal is deeper than L than it will not be found.

Completeness: Incomplete as solution may be beyond specified depth level.

Optimality: It is not Optimal.

Space Complexity: b as branching factor & L as free depth length i.e. $O(b^L)$

Time Complexity: $O(b^L)$ ✓

✓ Q. Compare DLS and BFS in terms of time & space complexity, completeness & optimality.

✓ Q. BFS is an implementation of queue where as DFS is an implementation of stack. Verify this statement with suitable example.

✓ Q. BFS Vs DFS.

Bidirectional Search:

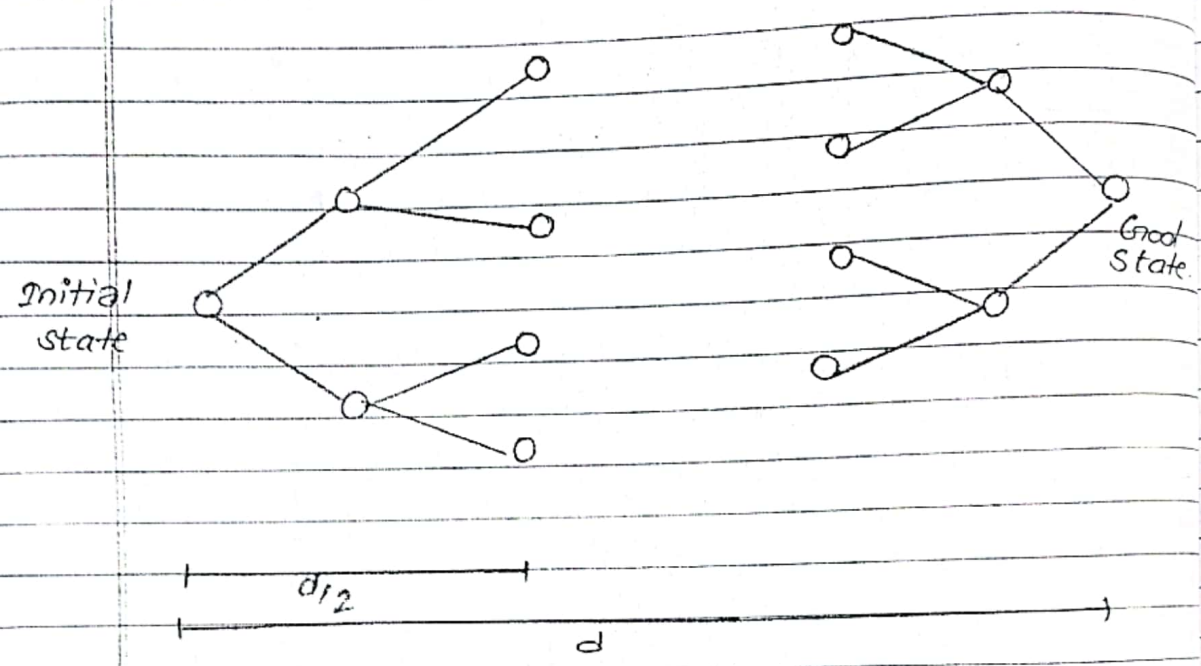


Fig:- Bidirectional search

⇒ As the name suggest, this search run two simultaneously search. i.e. one forward from initial state and other backward from Goal state.

⇒ The search stops when the two search's meet in the middle at some point.

- Completeness:
  It is complete when we use BF's on both search.

- Optimal:
  It is optimal when BF's is used.

IMP

Iterative Deepening DFS.

⇒ It is a general strategy often used in combination with DFS that finds the best depth unit.

⇒ It apply this strategy by gradually increasing the limit -
first 0 then 1, then 2 and so on until a goal is formed.

⇒ In effect, iterative deepening search combines the benefits of DFS and BFS.

⇒ Like DFS, its memory requirement are very modest.

⇒ Like BFS, it is complete when the branching factor is finite and optimal.

Completeness: It is complete.

Time complexity : $O(b^d)$

Space complexity : $O(bd)$

Optimality: It is optimal

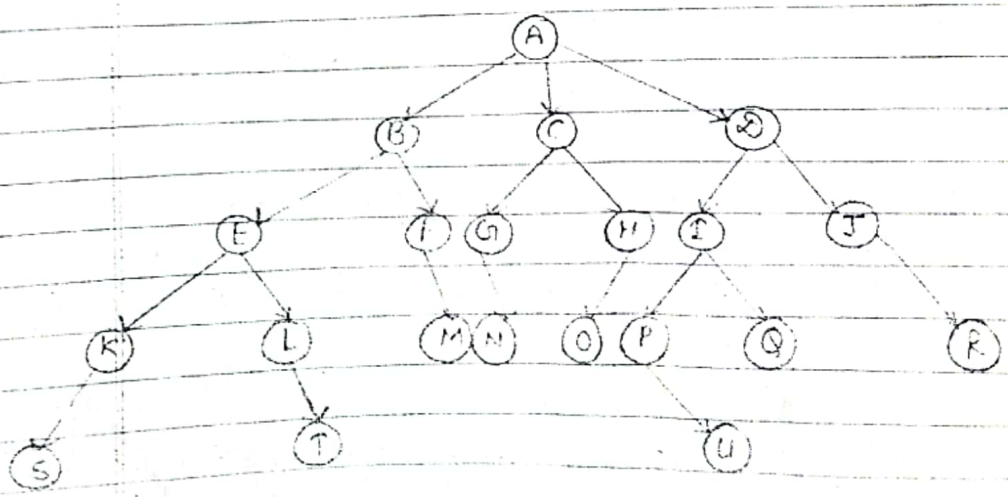
Illustration
Book pg no. 107 Fig 3.15
AI : A modern approach 2nd edition

Q. Write down the searching step differentiate betn uninformed search and informed search.

⇒ The searching steps are!

1) Start
2) Check whether the current state is the goal state or not.
3) Expand the current state to generate new sets of states.
4) Choose one of the new states generated for search which entirely depend on the selected search strategy.
5) Repeat the above steps until the goal state is reached for there are no more states to be expanded.
6) Stop.

Q. Diff. betn Iterative Deepening DFS and Depth Limit Search



Node at : A

successors : B, C, D

Fringe :- A

visited :- --

Node at :- B

successors : E, F

fringe :- B, C, D (FIFO)

visited :- A

Node at :- C

successors :- ~~C + B~~ G, H

Fringe :- C, D, E, F

visited :- A, B

Node at :- D

successor :- I, J

fringe - D, C, F, G, H

visited :- A, B, C

Node at :- E

successor :- K, L

fringe :- E, F, G, H, I, J

visited :- A B C D

Node at :- F

Successor :- M

Fringe :- F, G, H, I, J, K, L

visited :- A, B, C, D, E

Node at :- G

Successor - N

Fringe :- G, H, I, J, K, L, M

Node at:- H
sucessor:- O
Fringe:- H, I, J, K, L, M, N
visited:- A, B, C, D, E, F, G

Node at : I
Sucessor:- P, Q
Fringe:- I, J, K, L, M, N, O
visited:- A, B, C, D, E, F, G, H

Node at : J
sucessor : R
Fringe:- J, K, L, M, N, O, P, Q
visited:- A, B, C, D, E, F, G, H, I

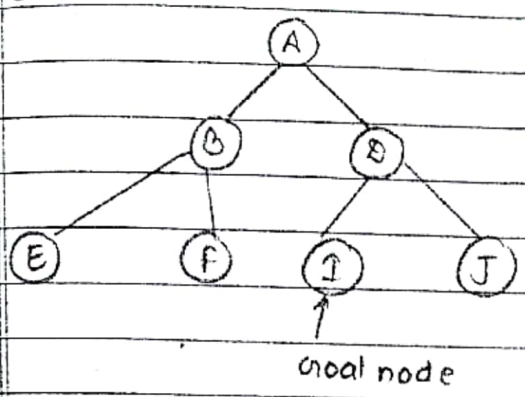
Node at : K
sucessor : S
Fringe:- K, L, M, N, O, P, Q, R
visited:- A, B, C, D, E, F, G, H, I, J

Node at : L
sucessor : T
Fringe:- L, M, N, O, P, Q, R, S
visited:- A, B, C, D, E, F, G, H, I, J, K

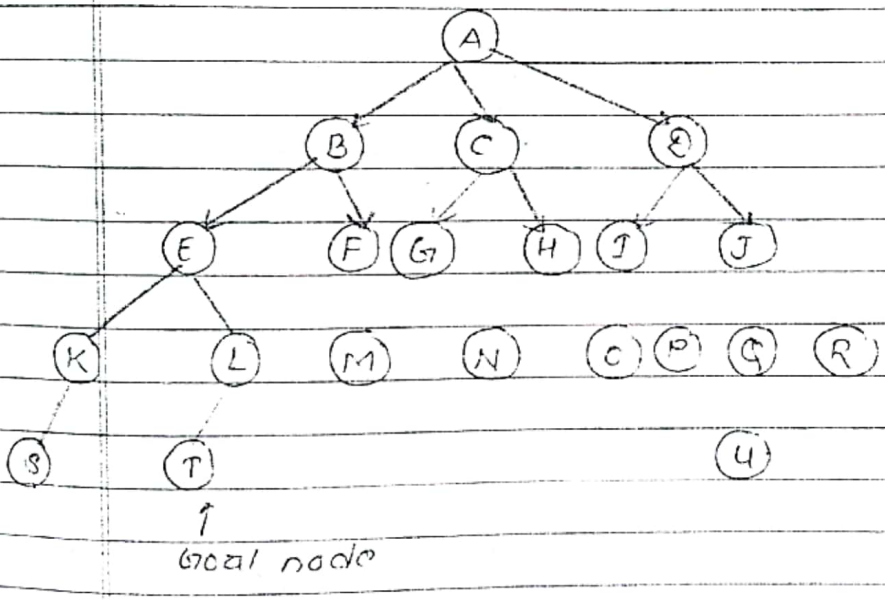
Node at : M
sucessor : —
Fringe:- M, N, O, P, Q, R, S, T
visited:- A, B, C, D, E, F, G, H, I, J, K, L

Node at : N
sucessor :—
Fringe:- N, O, P, Q, R, S, T

Ex:-



* Depth First Search



Soln

i) Node at A

Successors : B, C, D

Fringe : A

visited : –

ii) Node at B

Successor E, F

Fringe : B, C, D

visited : A

(iii) Node at : E

Successor : K, L

Fringe : E, F, C, D

visited : A, B

(iv) Node at : S

Successor :—

Fringe : S, L, F, C, D

visited : A, B, E, K

(v) Node at : L

Successors : T

Fringe : L, F, C, D

visited : A, B, E, K, S, L

Informed Search:

⇒ It is a strategy of problem solving web problem specific knowledge is known along with problem defn.

⇒ It finds soln more efficiently by the use of heuristic

⇒ It is also known as guided search or heuristic search.

⇒ Heuristic is a search technique that improves the efficiency of search process.

⇒ Heuristic are fallible i.e. they make mistake as well.

⇒ It is an approach following on informed guess of next step to be taken.

⇒ It is often based on experience.

⇒ Heuristic has limited information & hence can lead to sub-optimal soln or even fail to find any soln at all.

Heuristic Function:

Admissible Heuristics ✓

⇒ A heuristic function is said to be admissible if it never over estimates the cost of reaching goal that is the cost it estimates to reach goal is not higher than the lowest possible cost from the current ~~goal~~ point in path.

Eg:- 8 puzzle

7	2	4	1	2	3
5		6	4	5	6
8	3	1	7	8	

Start State Goal State

$h_1(n)$ = no. of misplaced files.

$h_2(n)$ = total manhattan distance (i.e. no. of squares from desired location of each tile)

$h_1(S) = 6$

$h_2(S) = 4+0+3+3+1+0+2+1$

$= 14$

Note:- सबै number लाई आफ्नो position मा minimal step लाउँदा 90° मा त्यो सबैको sum लिइन्छ

$h_1 + h_2 = 6 + 14 = 20$ (one of solution moves.

Q. What is heuristic search? Explain how heuristic func. can be admissible. 6E th ed.

Informed Search Technique

Generate & Test Algorithm:

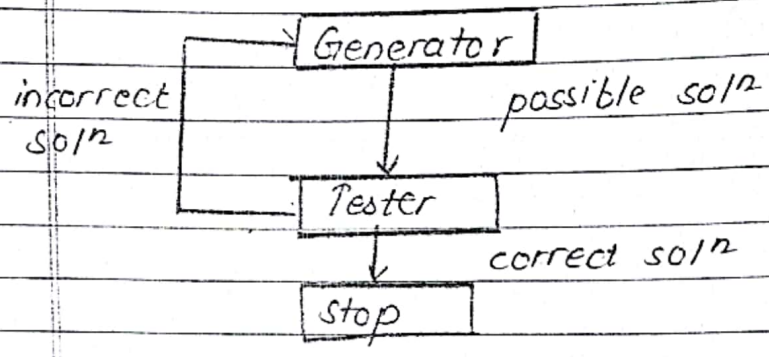


Fig:- Generate & test

Algorithm:

Generate & test Algorithm

- Acceptable for simple problems.
- Inefficient for problem with large space.

⊙ Algorithm.

1) Generate a possible solution (a state or path).
2) Test to see if this is actually a soln.
3) Quit if a soln has been found. Otherwise return to step 1.

It works on 2 module:

i) Generate module: It creates possible soln
ii) Tester module: If evaluates as test each of the proper soln either accepting or rejecting it.

Implementation:

1. Exhaustive generate-and-test

2. Heuristic generate - and - test
3. Plan generate - test :

Q. Write down the algorithm for generate & test approach & also write down its implementation.

* Best first Search (BFS):

Algorithm:

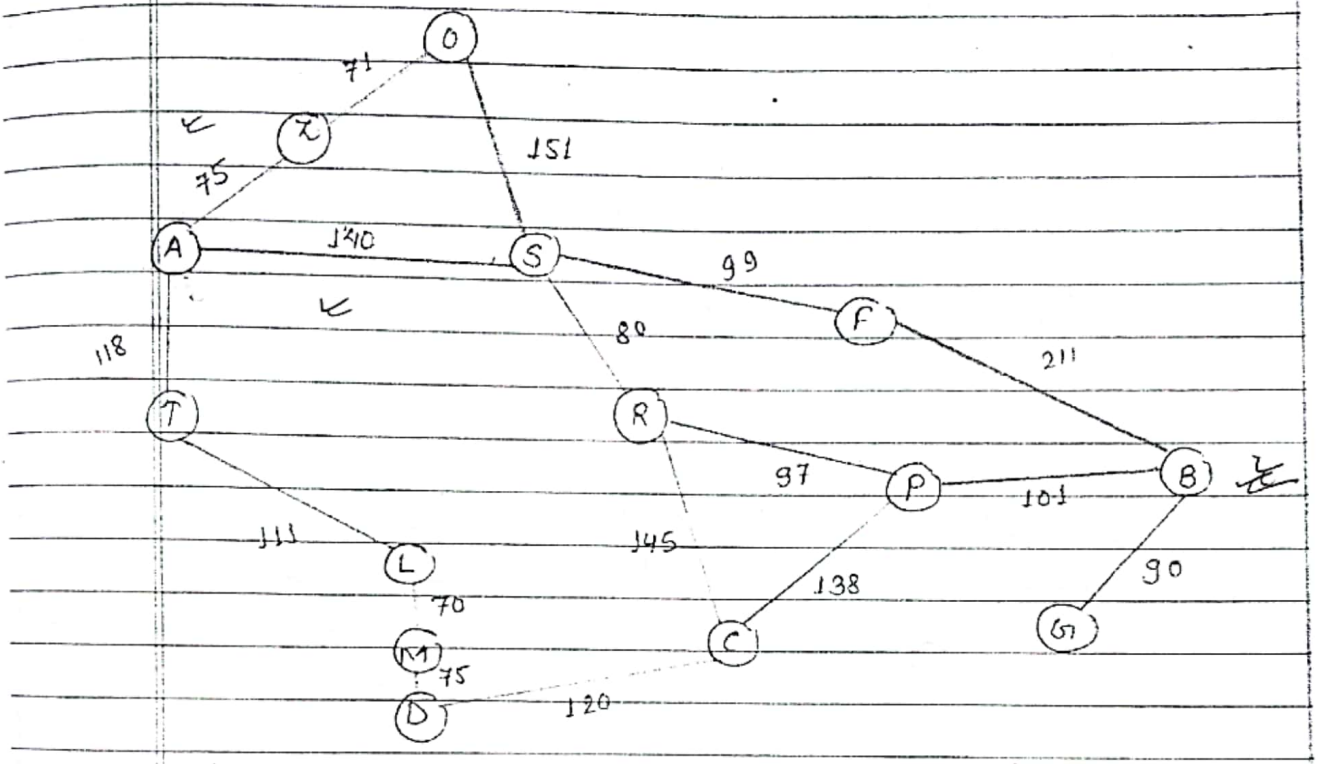
1. Put the initial node on a list START
2. If START = GOAL or START = EMPTY, then terminate search.
3. Assign the next node at START and call this node - A.
4. If A = GOAL, terminate the search with success
5. Else:- if, node has successor & generate all of them, find out how far they are from the goal node.
6. Sort all the children generated so far by remaining distance from the goal. Name the list as START - 1. Replace START = START-1
7. Go to Step - 2.

सुगम स्टेसनरी सप्लायर्स एण्ड फोटोकपी सर्भिस
बालकुमारी, ललितपुर ९८४१५९९५९२
NCIT College & Cosmos College

Types of BFS

1) Greedy Best First Search
2) A* Search

Given following graph of cities, starting at city "A" problem is to reach to the "B".



Shortest Line distance to B

	A	366
	B	0
	C	160
	D	242
←	F	176
	G	77
	L	244
	M	241
	O	380
	P	100
←	R	193
→	S	253
	T	329
→	Z	374

close	open
[A]	[ ]
[S, Z]	[A]
[Z]	[A, S]
[F, R, Z]	[A, S]
[R, Z]	[A, S, F]
[B, R, Z]	["]
[R, Z]	[A, S, F, B]

∴ A → S → F → B =
140 + 99 + 211 = 350 Ans.,

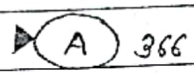
140
99
211
350

140

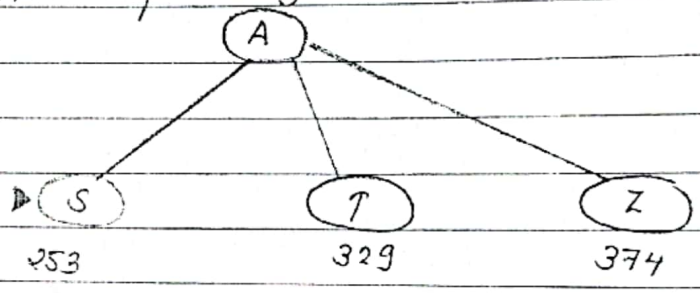
Solution,

using greedy best first can be as below:

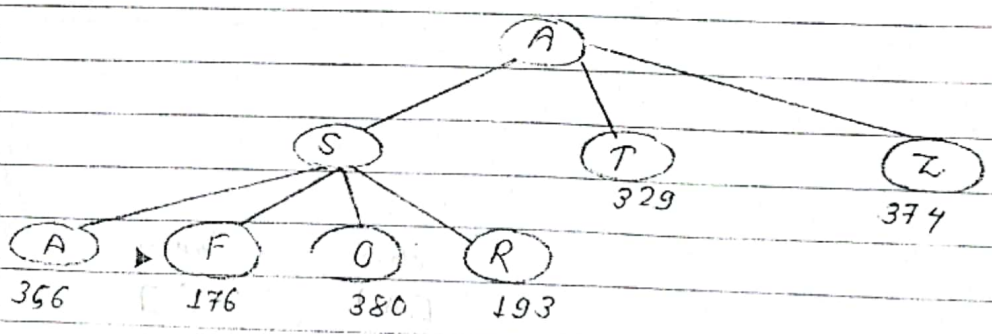
Step 1: Initial State

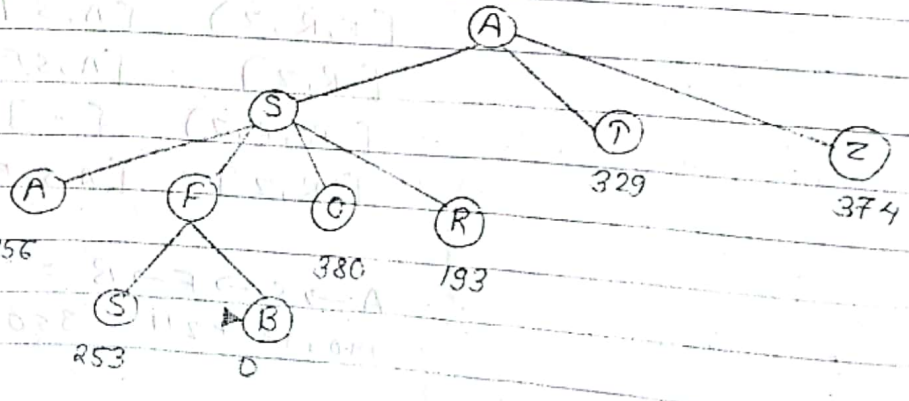


Step 2: After expanding A.



Step 3: After expanding S.

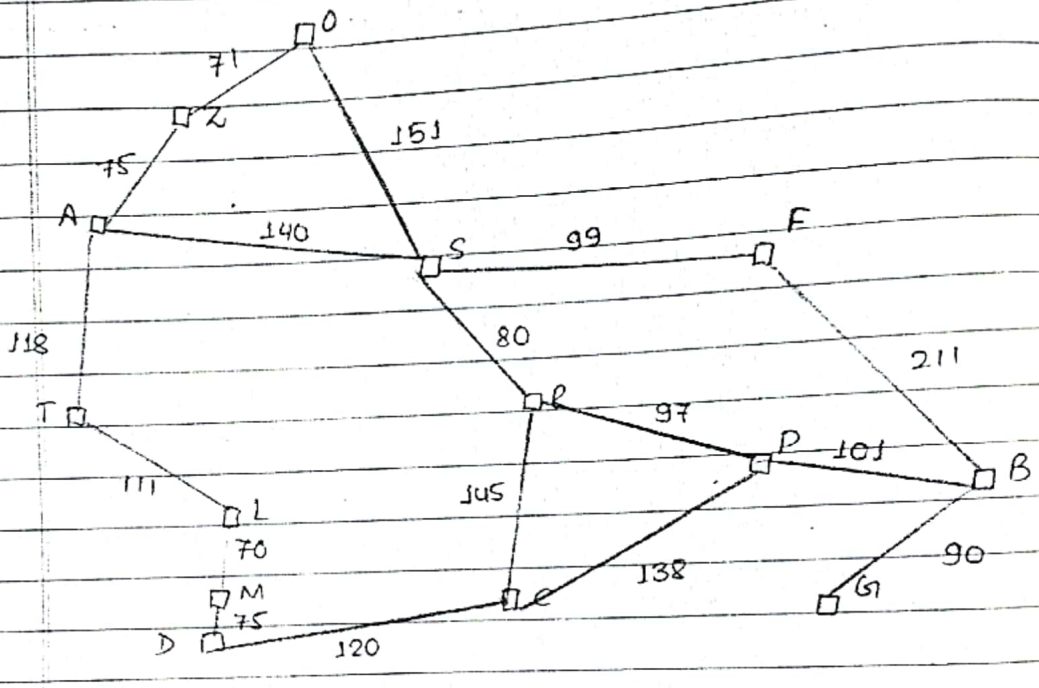




A* Search

⇒ It finds a minimal cost-path joining the start node

A* Search example (find path from A to B)



Shortest line distance.

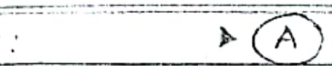
A	366	S	253
B	0	T	329
C	160	Z	374
D	242		
F	176		
G	77		
L	244		
M	241		
O	380		
P	100		
R	193		

Here, evaluate node connected to source. Evaluate $f(n) = g(n) + h(n)$ for each node. Select node with lowest $f(n)$ value.

g(n) = initial node to current node

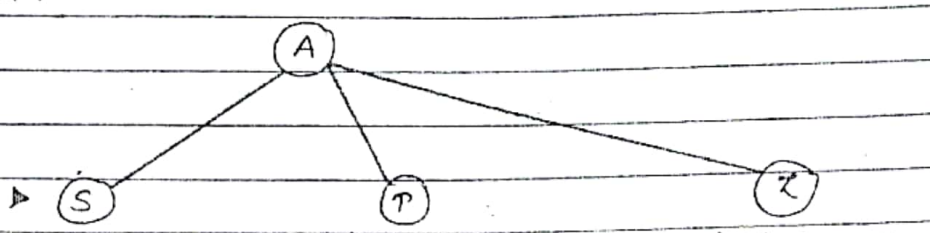
h(n) = current node to goal

Step 1:



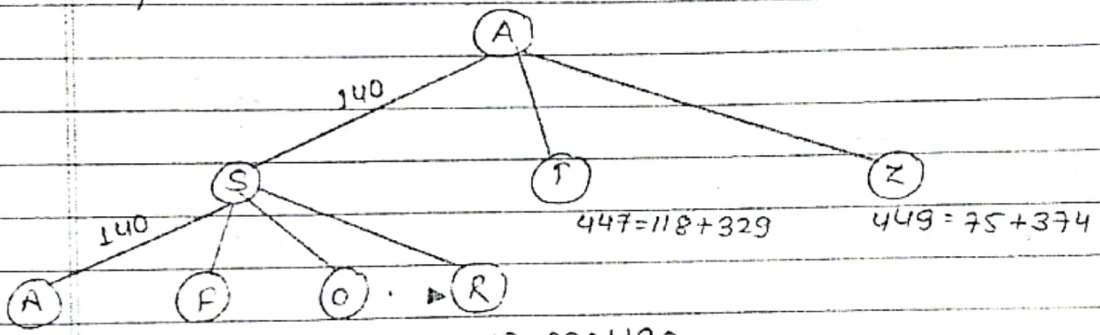
366 = 0 + 366

Step 2:



393 = 140 + 253 447 = 118 + 329 449 = 75 + 374

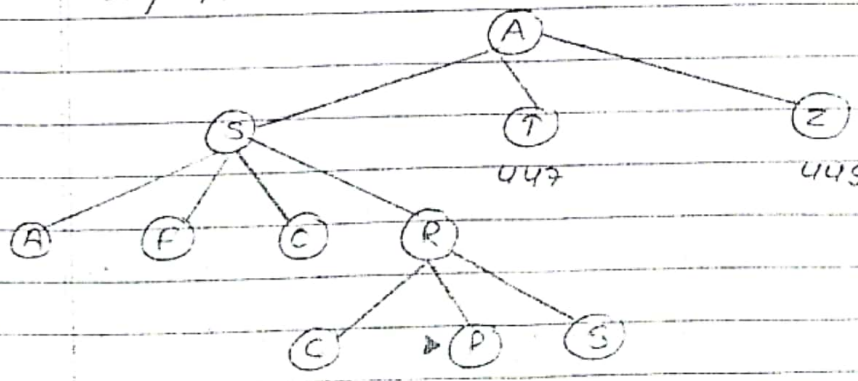
Step 3:



447 = 118 + 329 449 = 75 + 374

646 = 280 + 366 415 = 239 + 176 671 = 291 + 380 413 = 220 + 193

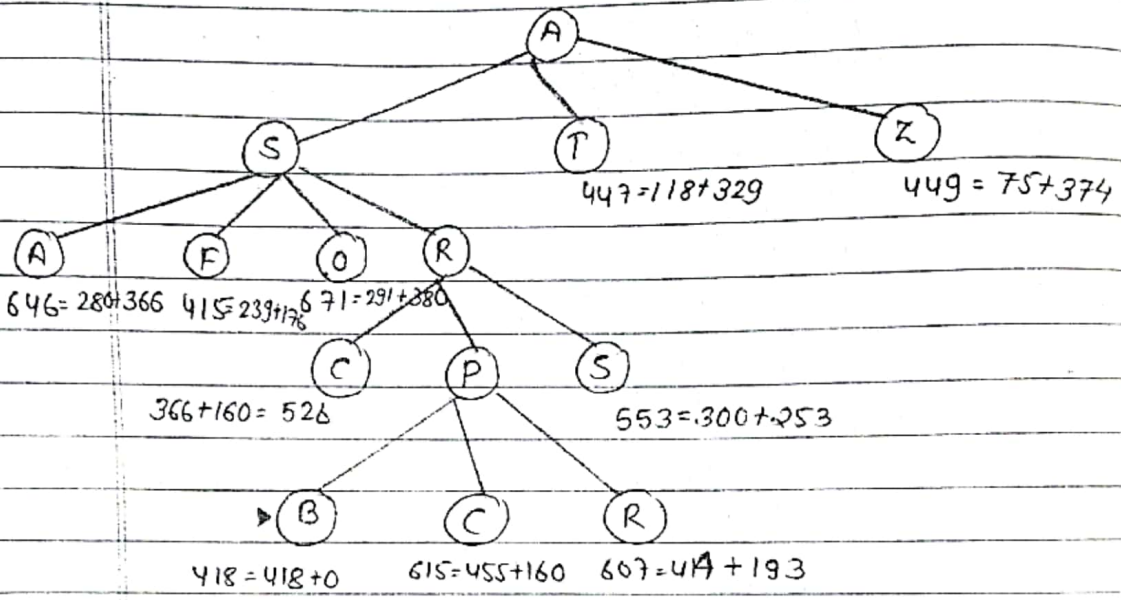
Step 4:



447 449

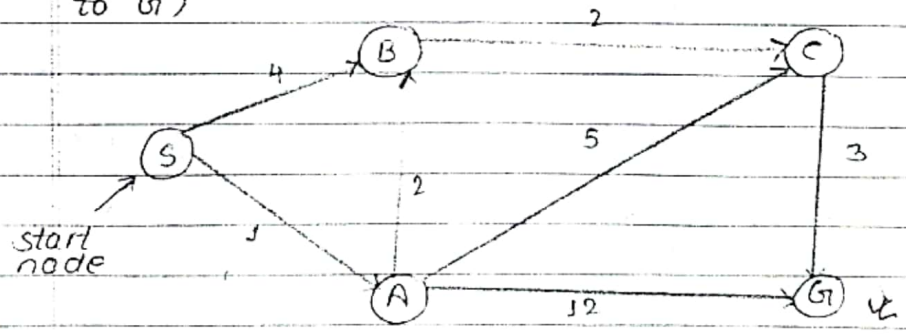
526 = 366 + 160 417 = 317 + 100 553 = 300 + 253

Steps:



Stepwise:

Q. Perform A* search on the figure below (from S to G)



	heuristic value (h(n))
S	7
A	6
B	2
C	1
G	0

S → B [4+2] = 6

S → A [1+6] = 7

S → B → C → G = (4+2+3) = 9

S → A → B = [1+2]+2 = 5

S → A → C = [1+5]+1 = 7

S → A → G = [1+12]+0 = 13

→ S → A → B → C → G = (1+2+2+3)+0 = 8

S → A → C → G = (1+5+3)+0 = 9

Step 1: Paths

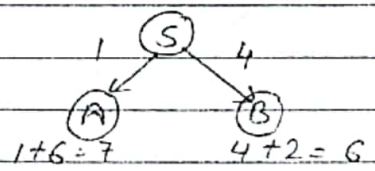
~~S → A , S → A → B , S → A → B → C , S → A → B → C → G~~

S → A , S → B , S → A → B , S → B → C , S → A → C ,
S → A → B → C → G , S → A → G , S → A → C → G ,
S → B → C → G

Step 2 :

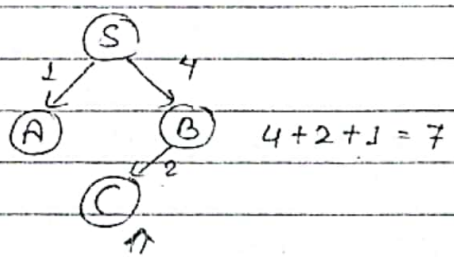
(i) S → A

f(n) = g(n) + h(n)
= 1 + 6
= 7

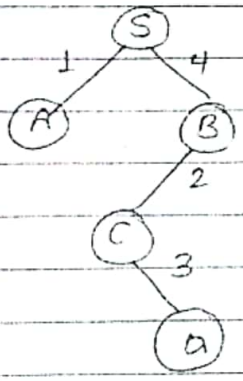


(ii) S → B

f(n) = g(n) + h(n)
= 4 + 2
= 6



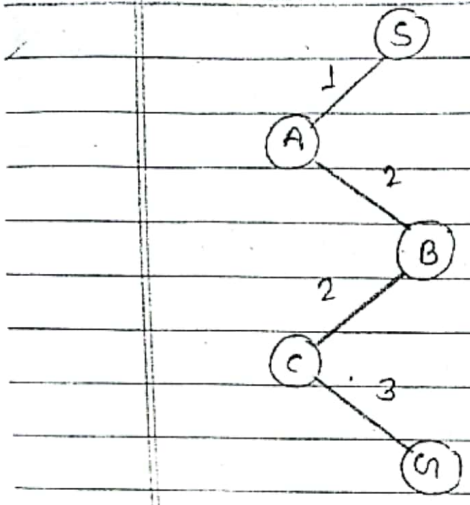
(iii) S → B → C → G



⇒ 4 + 2 + 3 + 0 = 9

(iv) S → B → C

(v) S → A → B → C → G

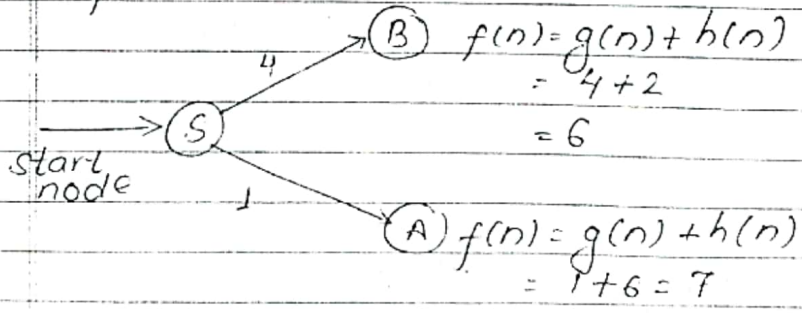


$\Rightarrow 1+2+2+3+0 = 8$ ✗

continue

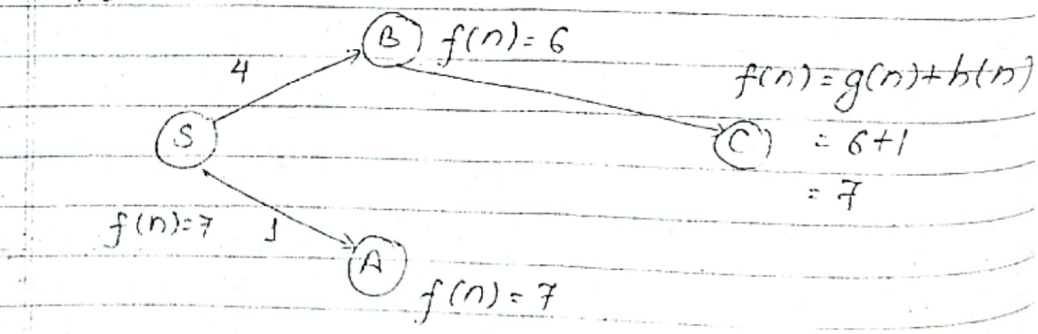
S.N // * Write down in brief about AO* search.

continue. stepwise sol^n,

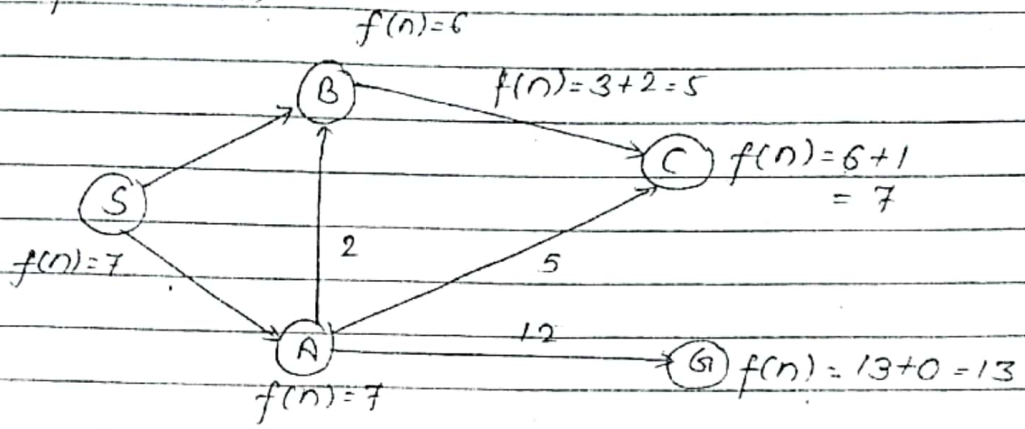


Step 2:

For B,

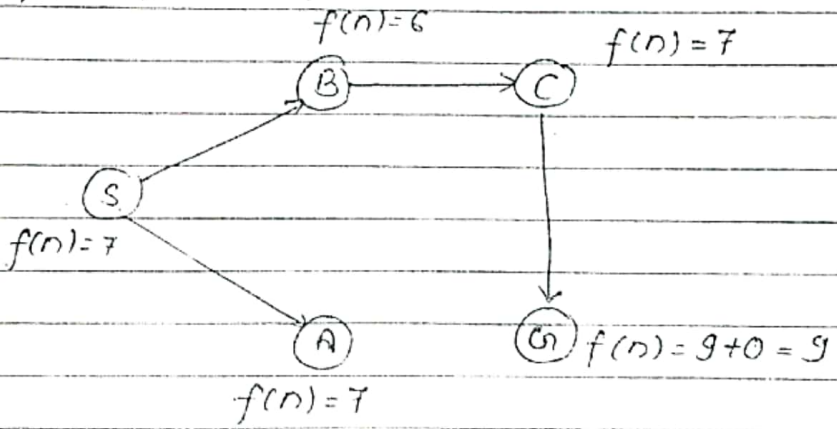


Step 3: For A,



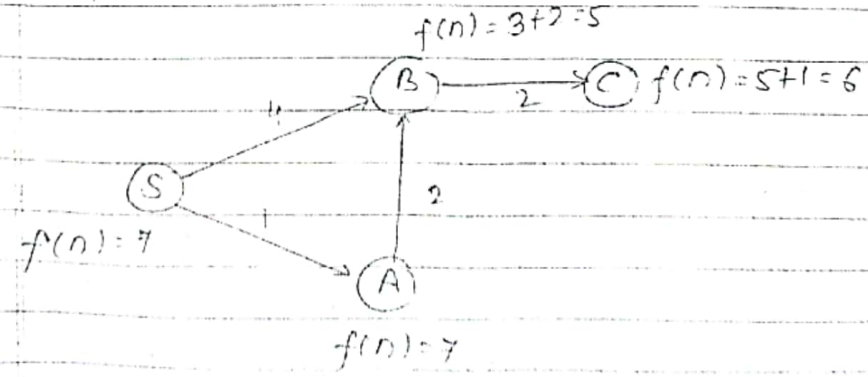
or,

for C,

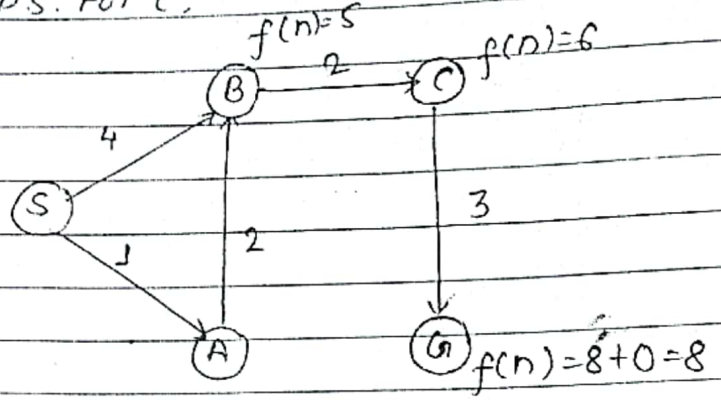


Step 4:

for B,



Step 5: For C,



$S \rightarrow A \rightarrow B \rightarrow C \rightarrow G$

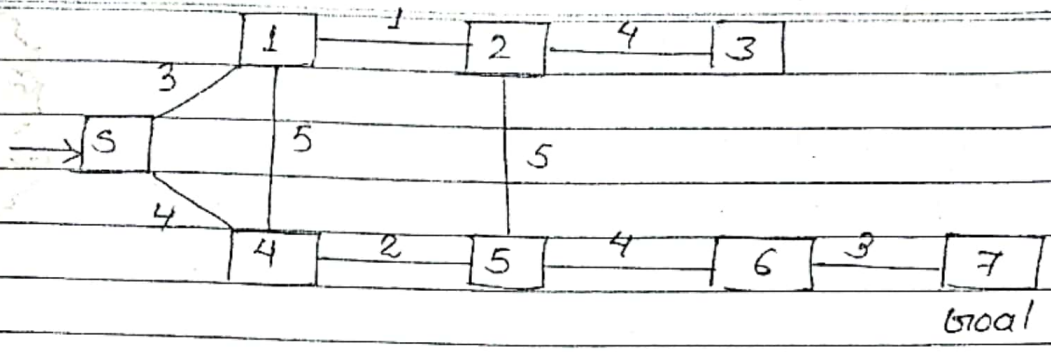
optimal //

10 marks

Q. Consider the graph given below assume that the initial state is S and goal state is 7 Find a path from the initial state to goal state. Using A* search. Also report the solution cost? The straight line distance heuristic estimates for the node as follow

h(1) = 14
h(2) = 10
h(3) = 8
h(4) = 12
h(5) = 10
h(S) = 15
h(6) = 10
h(7) = 0

[Note goal state always 0]



S

7

FORTY
TEN
TEN
SIXTY

F=2
29786
850
850
31486

N=0
FORTY
TEO
TEO
SIXTY

E=5
FORTY
T5O
T5O
SIXTY

T=8
R=
FOR8Y
850
850
SIX8Y

R=7

F078Y
850
850
SI48Y

Y=6, O=9
F9786
850
850
1486