SEARCH METHODOLOGIES

12/4/2022

Search

Search is a method that can be used by computers to examine a problem space in order to find a goal.

Data driven & Goal driven.

- Forward chaining — Backward chaining
- Top down — Bottom up
- Goal state unknown — Goal state is known.

Ex: Maze problem, Theorem proving

Goal driven search is particularly used in situations in which the goal can be clearly specified.

Ex: The theorem that is to be proved.

Data driven search is most useful when the initial data are provided & there is not clear what the goal is.

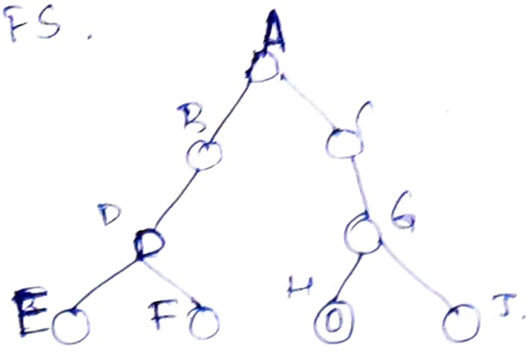
Ex: Analyzing astronomical data.

Generate and Test — (Blind Search) ⇒ Brute Force Search.

Generator: 3 Properties

1. It should be complete. — all possible sol^ns
2. It should be non redundant — not produce any sol^n twice
3. It must be well informed. — not propose a possible state. sol^n which does not match the problem

Apply BFS.



A B D E F C G H

~~A B C D G~~ B E ~~F H~~.

DFS = E F D B H G C A. — Space complexity.

BFS = A B C D G E F H. — Time complexity.

Properties of Search Methods

1) Complexity.
2) Completeness
3) Optimality.
4) Admissibility
5) Irrevocability.

BFS is complete search method whereas DFS is incomplete search method in case of infinite depth.

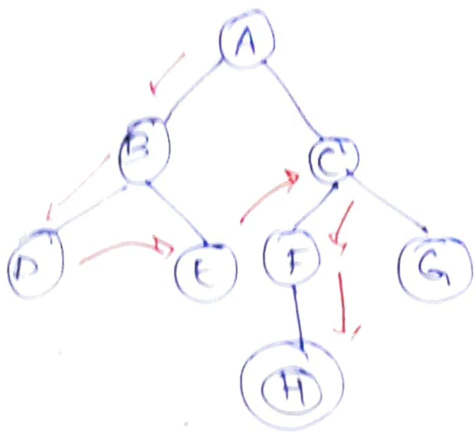
* Branch factor – No. of children for ~~each~~ a node.
(in BFS).

* Admissible – The search method that finds an optimal soln. in the quickest possible time is called admissible search method.

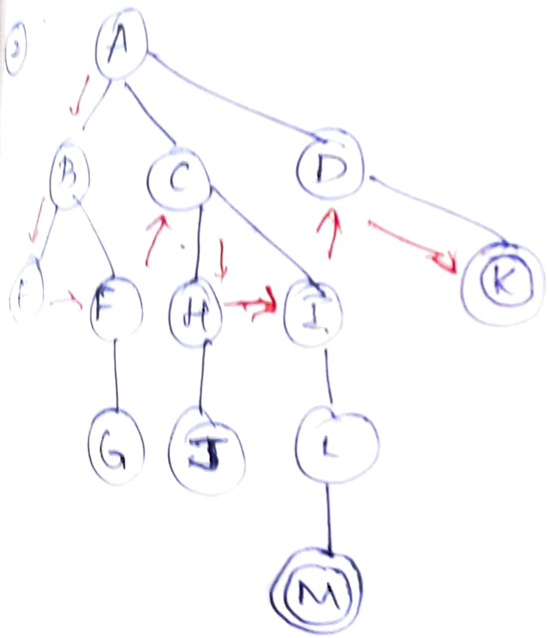
* Irrevocable – If backtracking is not allowed such a search method is called irrevocable.

Search method for which backtracking is allowed is called Tentative Method.

DFID - Depth First Iterative Deepening



Depth 0 → A.
Depth = 1 → A B C.
Depth = 2 → A B D E C F G.
Depth = 3 → A B D E ~~G~~ C F H.



D 0 → A
D 1 → A B C D
D 2 → A B E F C H I D K

D 0 → A
D 1 → A B C D
D 2 → A B E F C H I D K.
D 3 → A B E F G C H J I L D K.
D 4 → A B E F G C H J I L M.

For <u>BFS & DFS</u>

$\boxed{1 + b + b^2 + b^3 + \dots + b^d} \rightarrow \boxed{\frac{1-b^{d+1}}{1-b}}$

d = 2, b = 2

$\rightarrow \frac{1-2^3}{1-2} = \frac{1-8}{-1} = \underline{\underline{7}}$

* In DFS & BFS each & every node is explored only once but not in DFID.

* For DFID,

$\therefore \boxed{(d+1)1 + b(d) + b^2(d-1) + b^3(d-2) + \dots + b^d}$

① $d = 2$. $b = 2$.

$\Rightarrow 3 + 2(2) + 4(1) + \cancel{8(0)} + \dots$

$\Rightarrow 11$.

$d = 5$, $b = 10$.

Nodes $= \dfrac{1 - b^{d+1}}{1 - b} = \dfrac{1 - 10^6}{1 - 10} = \dfrac{1 - 1000000}{-9} = \dfrac{999999}{9} = 111111$

For DFS & BFS

DFID $d = 5$, $b = 10$.

Nodes.

$= (5+1)1 + 10(5) + 10^2(4) + 10^3(3) + 10^4(2) + 10^5$

$= 6 + 50 + 400 + 3000 + 20000 + 100000$

$= 123456$.

Imp.

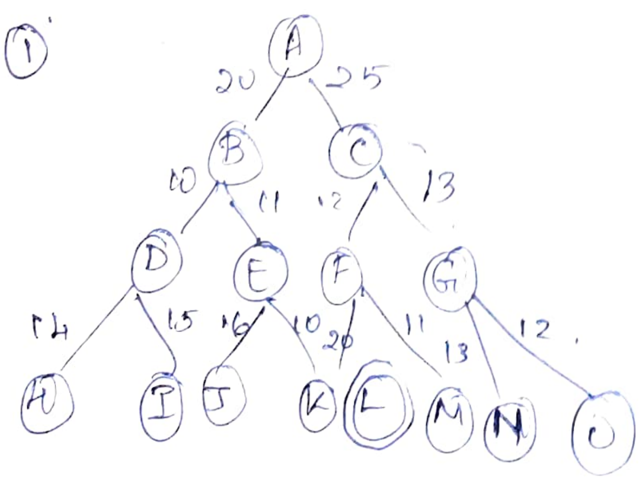
* Time Complexity & Space Complexity of DFID

TC → $O(b^d)$ & for BFS

SC → $O(bd)$ & for DFS.

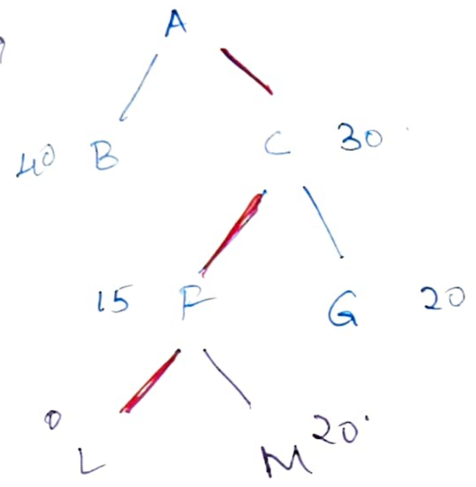
* Best First Search

①



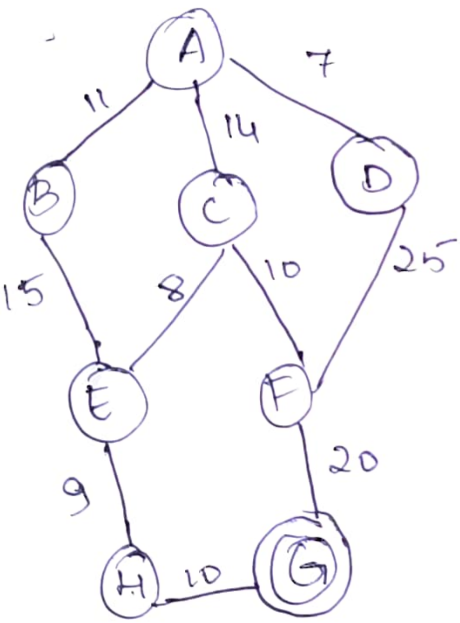
Heuristic value

A → L 50
B → L 40
C → L 30
D → L
E → L
F → L 20 14
G → L 15 20
M → L 20
L → L 0

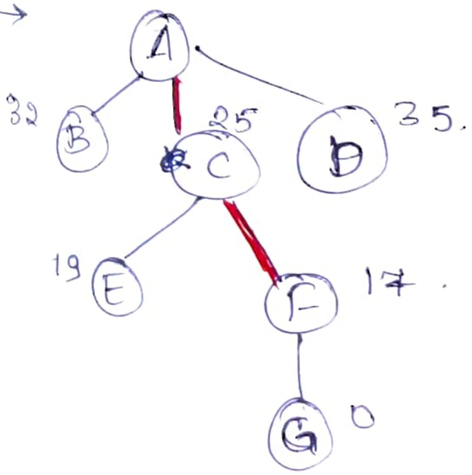


Open queue	Closed queue
~~A~~ B C	A
C B	
~~C~~ B	A C
F G B	A C
~~F~~ G B	A C F
L M G B	
~~L~~ M G B	A C F L
M G B.	A C F L.

(2)



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



Open	Closed.
A	
~~A~~ B C D	A
~~C~~ B D.	A C.
~~F~~ E B D	A C F
~~G~~ E B D	A C F B.

Open Queue Implementation

Function best().

```
{
    queue = [ ];
    State = root_node;
    while( true )
    {   if is_goal (state)
            then return SUCCESS;

        else
        {
            add_to_front_queue(successor (state));
            sort(queue);
        }
        if queue == [ ]
            then return FAILURE;

        state = queue[0];
        remove_from_front(queue);

    }
}.
```

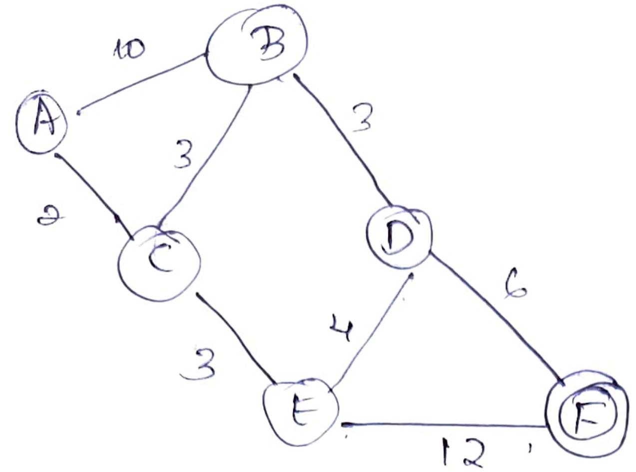
⇒ queue

State - A.
remove A.
queue - BCD
Sort CBD.

State = C
queue - EFBD
remove C

Sort - FEBD

(3)



$A \rightarrow F = 50$

$B \rightarrow F = 45$

$C \rightarrow F = 38$

$D \rightarrow F = 32$

$E \rightarrow F = 30$

$F \rightarrow F = 0$

queue.

State = A.

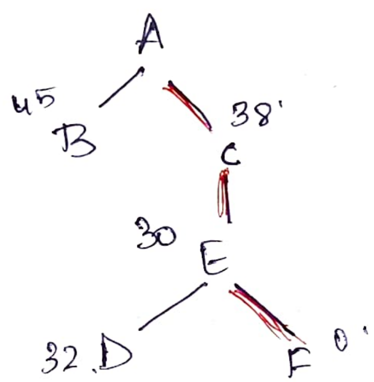
add = ~~A~~ B C.

Sort – C B ~~A~~

state ⇒ C

remove from front.

~~queue~~. open   close.



add = E

Sort = E

State = E

remove

add D F

Sort F D

State F

remove

→ success F is goal

* Best First search gives quicker soln not the best. — disadvantage.

TC is lesser than DFS, BFS, DFID.

2 conditions

(1) * In order to decide which is best heuristic value, choose the one which is more informed i.e which is less.

(2) * Reduce the no of nodes to be examined → 2nd condition

Start state: 8 puzzle game

7	6	
4	3	1
2	5	8

20 - Depth.

BF (middle) = 4.

BF (corner) = 2.

BF (edges) = 3.

Average BF = 3.

goal state

1	2	3
8		4
7	6	5

$h_1(\text{node}) = 8$

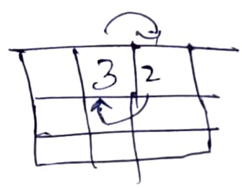
↑

heuristic values is the no. of nodes wrongly places

↑ Start State

$h_2(\text{node}) = 2 + 2 + 2 + 2 + 3 + 3 + 1 + 3 = 18$.

$h_3(\text{node}) = h_2(\text{node}) + (2 \times K(\text{node}))$



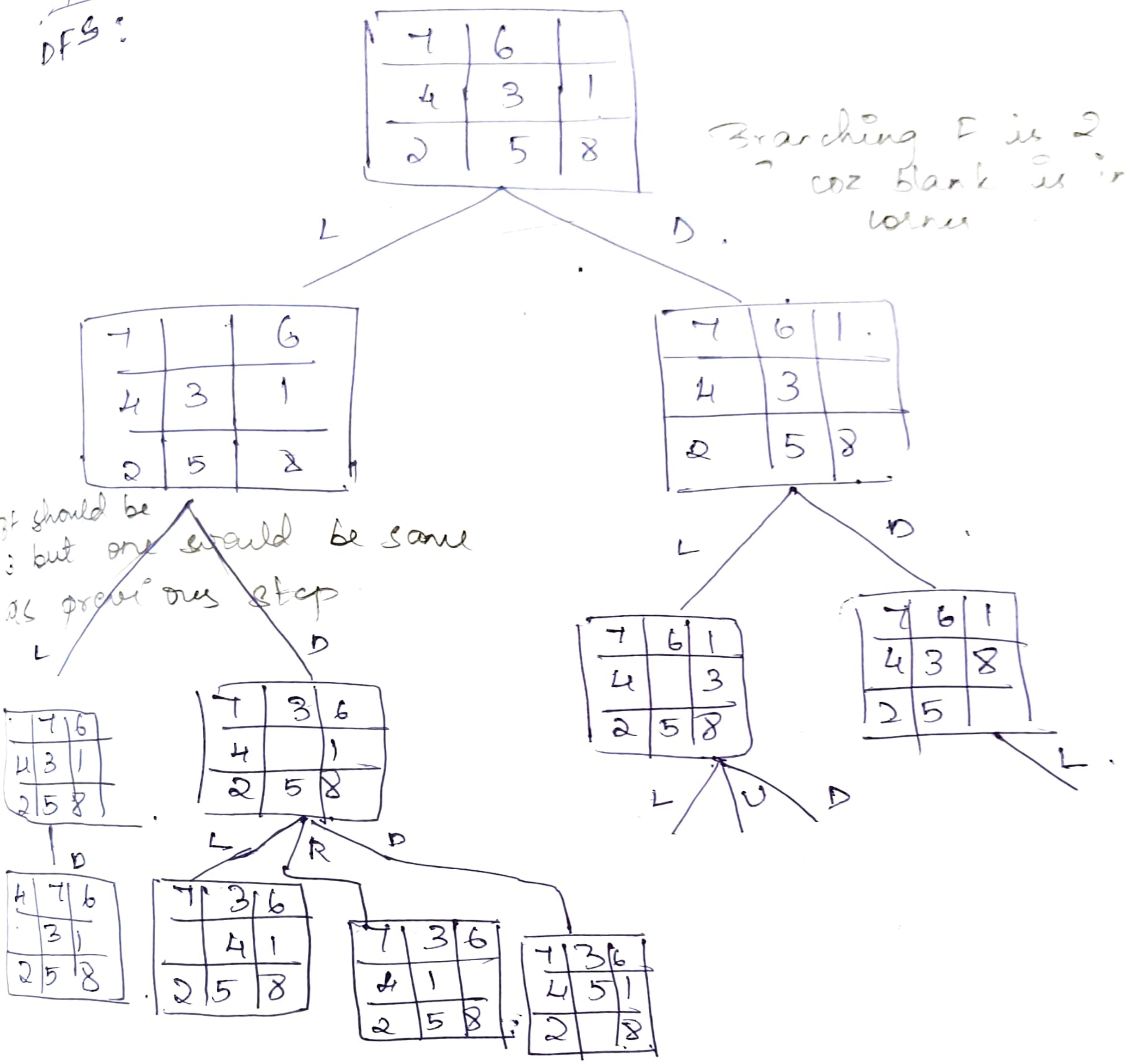
$$\boxed{h_1(\text{node}) < h_2(\text{node}) \leq h_3(\text{node})}$$

* **Admissible heuristic**

A heuristic which never overestimates the cost of changing from a given state to the goal state is defined as Admissible heuristic

Depth - 4 — Means there are 5 levels.

DFS:

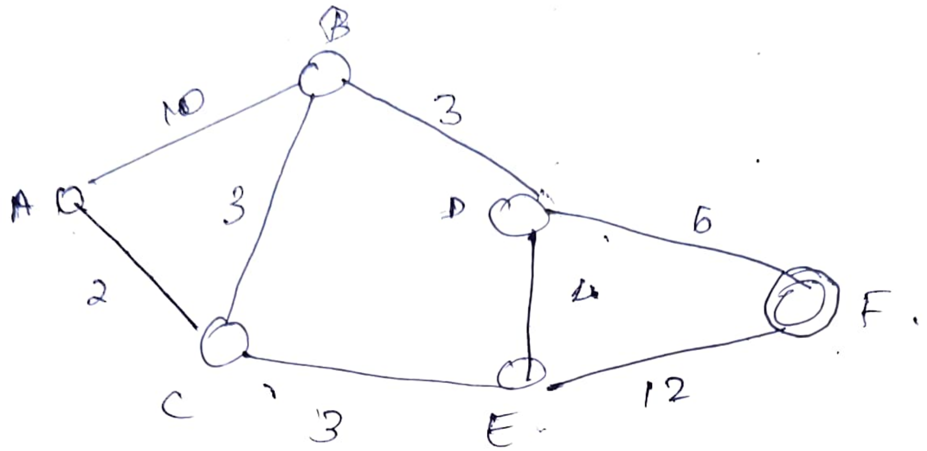


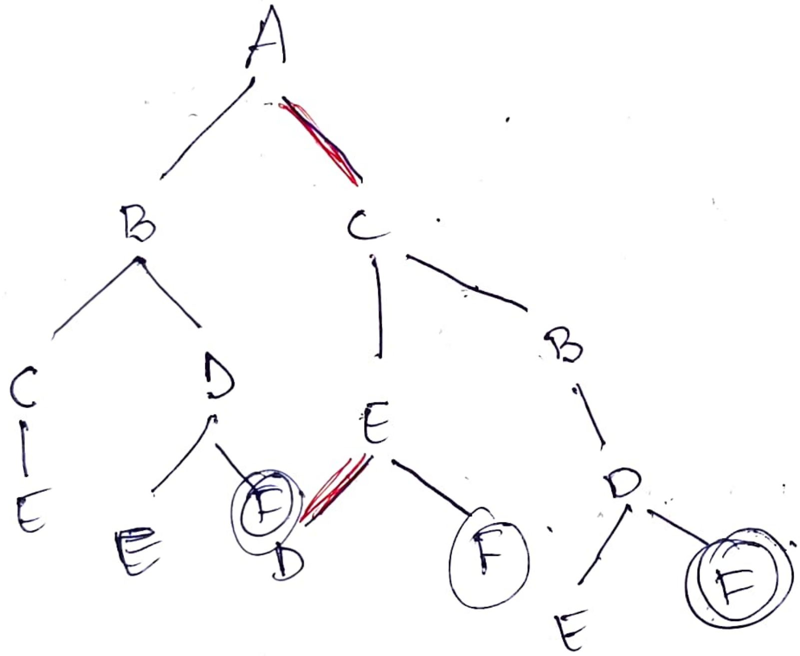
* Monotonicity

* A search method is described as Monotone if it always reaches a given node by the shortest possible path.

* A search method that reaches a given node at different depth in the search tree is not Monotone.

(1) * A monotonic heuristic is a heuristic that
(2) has the stated property.





*