i) The exciting new efforts to make computer think. Machines with minds in the full and literal sense.

ii) The automations of activities that we associate with human thinking.

iii) The art of creating machines that perform functions that require intelligence when performed by people.

iv) The study of how to make computers do things at which, at a moment, people are better. (mimic human behavior)

v) The study of computations that make it possible to perceive, reason and act.

vi) The branch of computer science that is concerned with the automation of intelligent behavior.

⇒ The pre-requisite for K.S is to know about the problem or have knowledge about it. Then:

I] Representation of knowledge / rules various techniques like predicate logic, Symantic etc are available.
[Prolog works on predicate logic]

II] Reasoning, problem solving, decision making

III] Action → Valid.

When a tester is unable to distinguish whether the answer is coming from a human (expert) or a machine, the machine is said to have passed/ succeeded the Turing Test.

3/7/19 AI problem
which requires some sense to be solved

AI techniques
There is no specific technique available. Any technique which gives a solution to the AI problem can be classified as AI technique.

Steps:
- Define the problem
  Input state → goal state
  Representation

- Analyze
  No. of input & goal states
  Identify the pattern (how can it be solved based on the problems solved previously)

- Techniques

⇒

1	2	3
4	6	8
7	5	

Initial

1	2	3
4	5	6
7	8	

i) Calculate the no. of tiles which are out of place.

Initial	Goal
3	0
Move 8 down 3	0
Move 5 right 3	0

] Here, the no. is same but, if the no. increases, that move can be dropped.

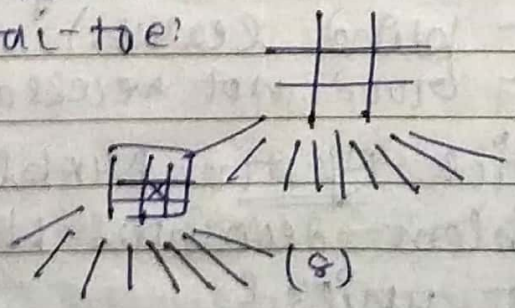
ii) Calculate the no. of steps by which each tile is away from its goal.

Initial	Goal
0+0+0+0+1+1+0+2	0
=4	
move 8 down 0+0+0+0+1+1+0+1	0
=3	
move 5 right 0+0+0+0+2+1+0+2	0
=5	

Hence, moving 5 right is not feasible. As it does not lead to the goal state faster.

Problem state space.
(Space in which all the states are available)

Tic-tac-toe:



This problem has multiple goal states

9 layers.

State space search - Searching for the exact space which can lead to the goal state(s).

Production System (Expert System)
Involves producing rules and solution.

- Define the problem (I & G states)
- Represent
  - Rules → set of rules/action. valid step.
  - Solution → Rule Applier (which rule to be applied which can take us nearer to goal state)

[Control Strategies]

Control Strategy
It should be such that:

i) gives you motion. - the task does not stop and give a new state with every move.

ii) Systematic - The move should be able to answer the question of why the particular move was taken. It is also called informed search. (Opposite - blind search)

iii) Efficient - Good, not necessarily best, solution.

5/7/19

Characteristics of the problem

- Is the problem decomposable into smaller independent units?
- Can solution steps be ignored or undone?

ignore - at any point of time, we can ignore all the steps so far and start from the beginning.

undo - go back to the (immediate) previous state. Problems in which this can be done are called recoverable problems.

[Irrecoverable problems are the ones in which steps cannot be undone]

- Is universe (of the problem) predictable? Can the result be predicted based on the steps taken.

- Is a good solution absolute or relative? Whether the goal depends on no. of states or does the answer is what matters in the end.

  Playing chess - relative - depends on opponent.

  8-tile puzzle - absolute

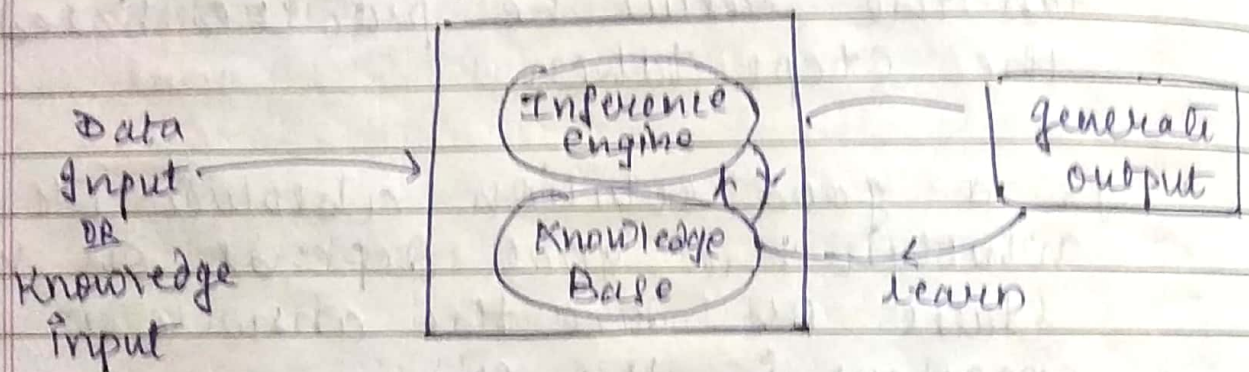
  Best path problem: Travelling salesman problem.

- Is the solution a state or path?

- What is the role of knowledge?
  Knowledge is important only to constrain the search for solution. It is required even to be able to identify or recognize a solution.

- Does the task require interaction with person?
  ex- where a computer is given a problem and it generates the result without any further involvement from the user, is called problem-in solution-out.
  Ex (where interaction req.) medical diagnosis.

Production System



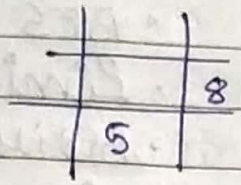
Production System Rules
can be categorized as deductive or abductive inference rules.

Recognize-Act cycle for production system
1. Match
2. Conflict resolution (Strategy to choose which rule to apply)
3. Apply
4. Check

Classes
i) Monotonic PS - Application of one rule should not affect the application of any other rule at any

given time.

Ex:



2 Rules: 8 ↓ or 5 →

Application of 8 ↓ will result into 5 → ~~never~~ not being applied in the next step. Thus, it is not monotonic.

ii) Partially commutative PS – Application of particular seq. leads from state x to y, then any permutation of that seq. will also lead from state x to y.

⇒

	Monotonic	Non-monotonic
Partially commutative		
Not partially commu.		

⇒
- Missionaries & canibals problem
- Tower of Hanoi
- Blok world
- Monkey Banana
- Cryptarithmatic → SEND + MORE = MONEY (no. assigned to each letter & the addition should stand true).
- 8-tile puzzle.
- Tic-tac-toe.

Problem solving by Search

↓

Informed Search | Uninformed Search / Blind Search

Page No.
Date

IS
- Heuristic search
- Hill climbing → simple
  ↳ steepest ascent HC
- Best first search
- A* (star)
- A* optimal
- Branch & bound
- Randomized

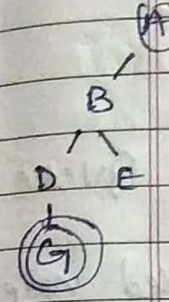
UIS/BS
- DFS
- BFS
- Limited DFS
- Bidirectional search
- Iterating deepening DFS

Evaluation of algorithm
- ~~Correctness~~ completeness
- Optimality
- Time - complexity
- Space - complexity

If the algorithm can guarantee to arrive at the correct result (solution) if there exists any, it is said to be complete.

The algorithm not only gives the solution but, gives the best one, it is optimal.

DFS
1. In the initial state, is goal state quit and return success.
2. Otherwise, do the following until success or failure is signaled.
   a. Generate successor, E, (or the new initial state)
      If there are no more successors, signal failure



infinite

8/7/19

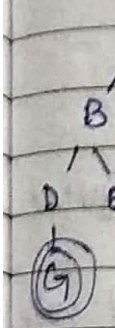
b. Call D-F-S with E as initial state.

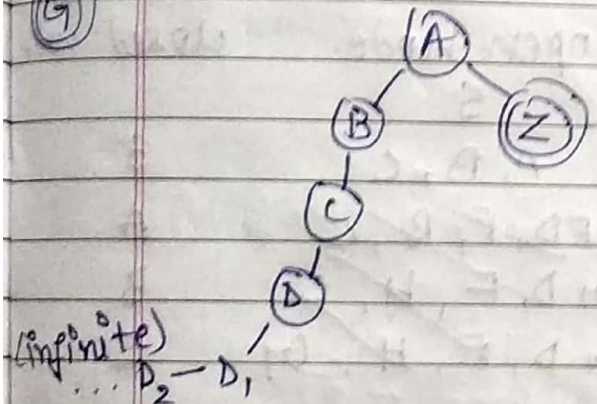
c. If success is returned, signal success, otherwise continue in this loop.

→ Data structure - Stack

Also,

open node	closed (visited) node
A	A
B, C	B
D, E, C	D
G, E, C	G.





DFS does not guarantee to find the solution (can enter into loops which can lead away from goal). Hence, it is not complete.

It is not optimal either as it does not always guarantee to find the correct solution.

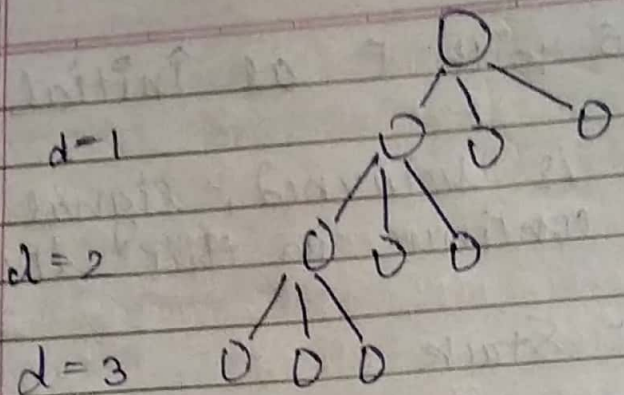
Branching factor - No. of branches generated per node.

8/7/19 Let b be the branching factor and d be the depth of the tree.

∴ Time complexity : $O(b^d)$

Space complexity : Let $b = 3$

then, Tree :



b^d

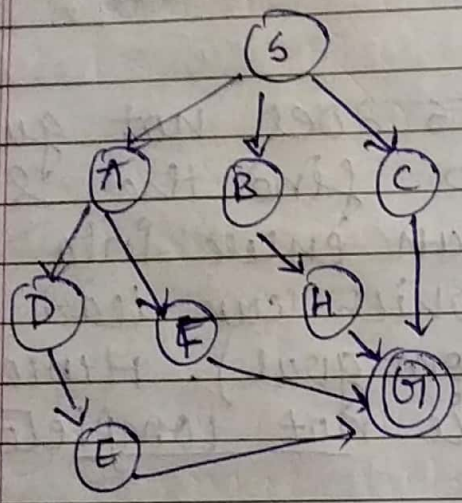
b^{d-1}

b^{d-2}

∴ Space complexity = $O(b^d)$

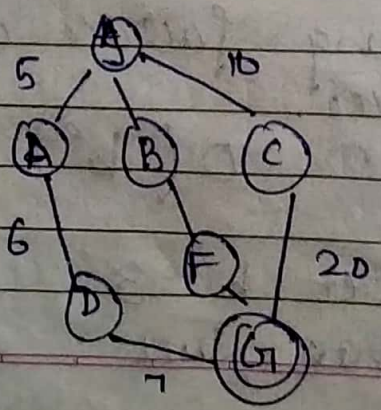
Breadth first search

① Data structure : Queue



open node	closed node
S	
A, B, C	S
B, C, D, F ~~D, F, B, C~~	A
C, D, F, H ~~D, F, H, C~~	B
D, F, H, G1 ~~D, F, H, G1~~	C

BFS can be complete if the branching factor is limited or finite. If the depth is infinite, there exists no solution. Hence, completeness does not come into picture at all.



DFS: S - A - D - G1

BFS: S - C - G1

In this case, DFS is better.

But, if we have uniform cost.
- BFS always finds the nearest solution
∴ BFS is optimal only when we have uniform cost graph

Time complexity : $O(b^d)$

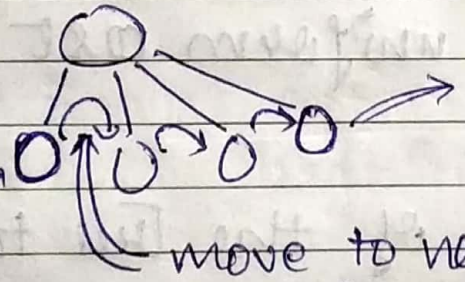
Space complexity : $O(b^d)$

⇒ DFS is better than BFS in terms of space complexity (time complexity is same)

Search progess is better in BFS than DFS. (exhausts every level before moving to next)

Iterative Deepening DFS:
- work progress like BFS
- Space complexity DFS.

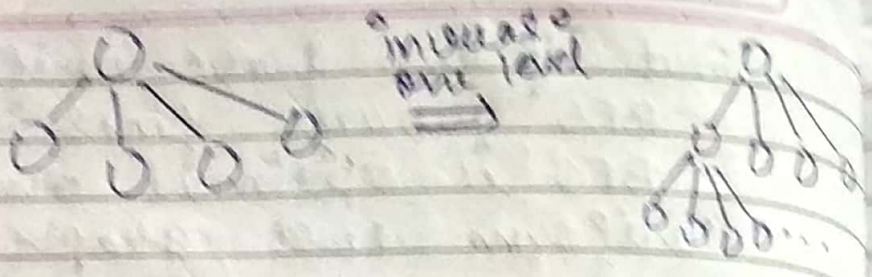
11/7/19



Thus, we are iteratively moving in depth of the tree.

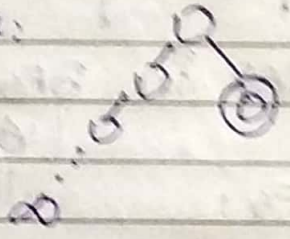
Time complexity :
Assume $b=4$.

Remove the nodes which aren't goal nodes from the memory.

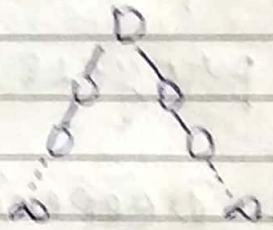


We make sure that no node in a particular level is left unvisited helps to solve the infinite loop in terms of depth. (cannot work in entire tree if infinite)

Ex:



works.



cannot work

- It is complete.
- Optimal when uniform cost

Limited DFS

Restrict the depth of the tree to a fixed value.

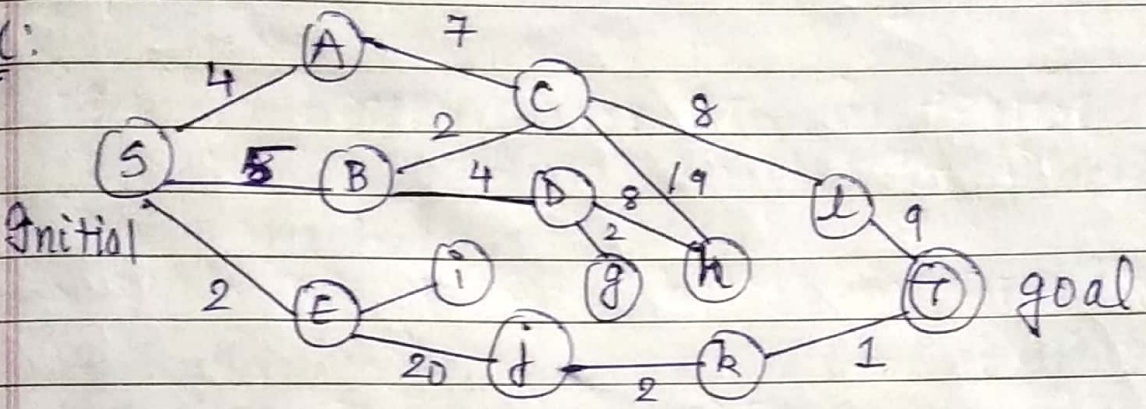
Tree may have depth beyond this no. but, the algorithm won't search beyond the fixed level.

Not complete as the solution may be at the very next level. complete only if the solution lies inside the limited depth.

Not optimal either as if the solution does not exist, there is no point in finding the best one.

Time complexity = $O(b^l)$
Space complexity = $O(bl)$
where, l = limited depth.

Ex:



DFS:

open nodes	closed nodes	
S		
A, B, E	S	
C, B, E	A	4
h, l, B, E	C	7
l, B, E	h	19 ×
T, B, E	l	8
B, E	T	9
		28

BFS:

open nodes	closed nodes	
S		
A, B, E	S	
B, E, C_A	A	4
E, C_A, D	B	
C_A, D, i, j	E	
D, i, j, h, l	C	

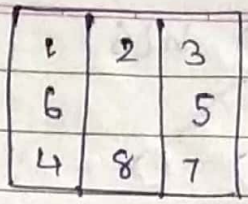
(we need to consider node c from B as well to get the optimal result because we don't have uniform cost here))

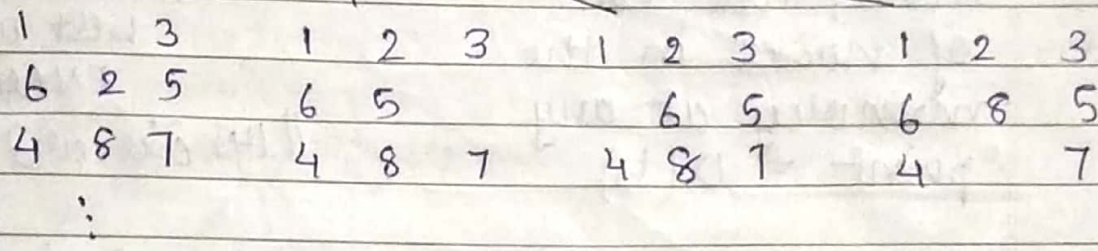
i, j, h, l, g
j, h, l, g
h, l, g, k
l, g, k
g, k, T
k, T
T, F

Dijkhlgk T

15/7/19

- Heuristic / Informed Search
Every node has some information regarding the goal state, i.e. how good or bad a node is which on visiting might lead us nearer or far from the goal state.

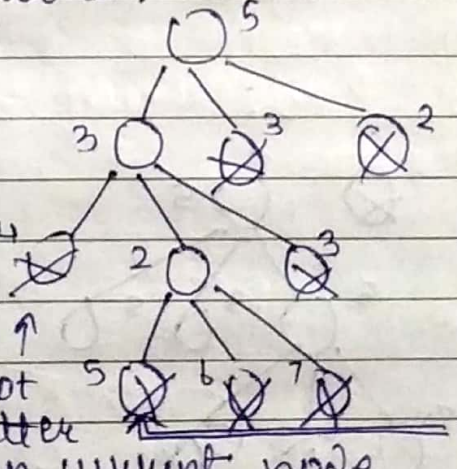




heuristic func.
- f_1 = no. of tiles at incorrect places
- f_2 = no. of tiles at correct places
- f_3 = manhattan distance

A heuristic func. is said to be optimal if it gives the exact value for an answer.

Simple hill climbing
Choose the best way available and discard all other nodes. (strictly DFS)



h(n)

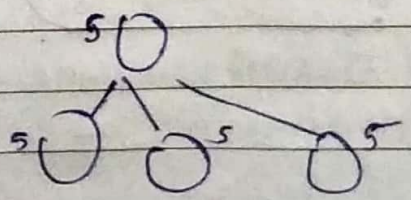
No back-up.
If no sol^n is found, backtracking is not possible.

not better than current node

Steepest ascent hill climbing

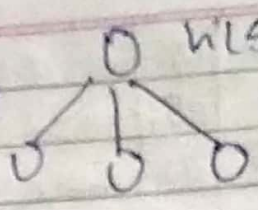
If this node is better than its parent, other two nodes (siblings) are simply discarded

cannot go any further.



18/1/19

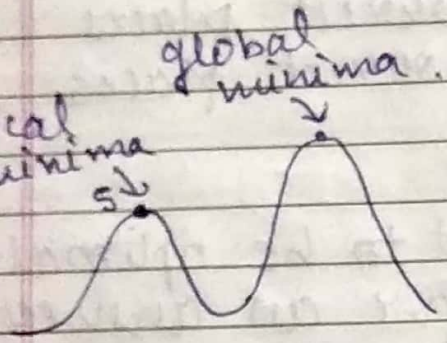
Variance of Hill climbing

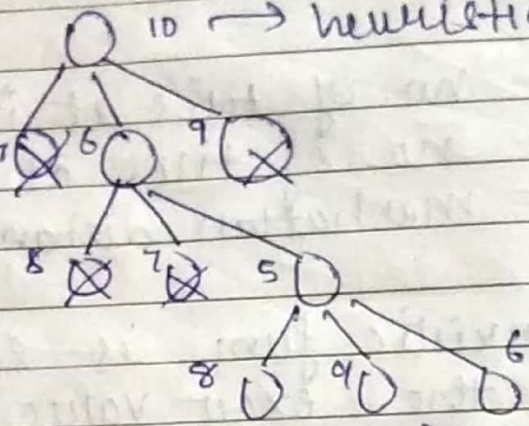


// (1) assign heuristic value
// (2) generate successors
// (3) search for the best option at every level
// (4) discard other nodes

Hence, the no. of nodes in the memory at any point = O(b)

fx: i)

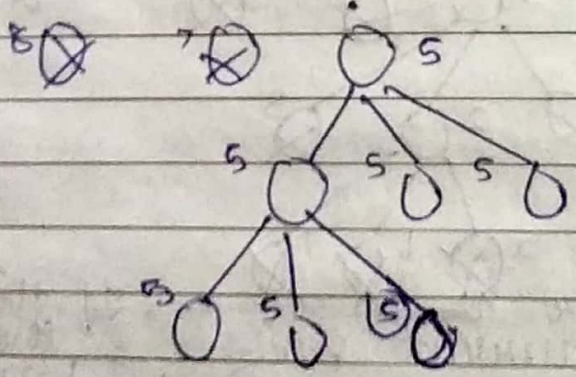




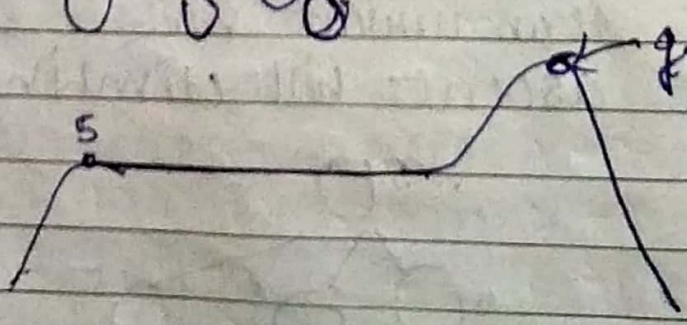
Problem of local minima

Since, no node has a heuristic value better than (5), the algorithm / procedure terminates.

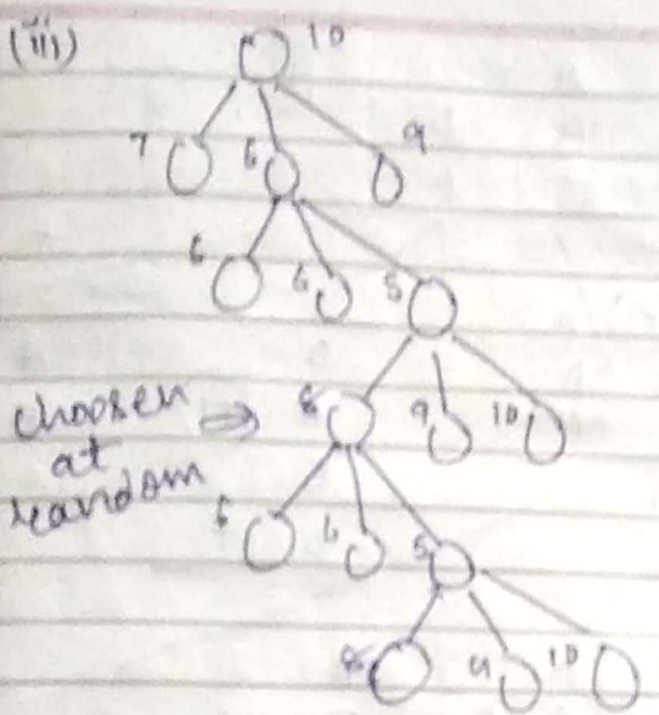
(ii)



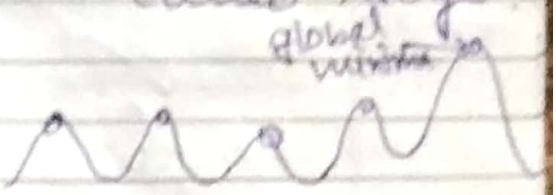
we cannot decide where to go next. This is called plateau (flat area).



(iii)



When we cannot find a better solution, a node is chosen at random and others discarded this problem is called ridge

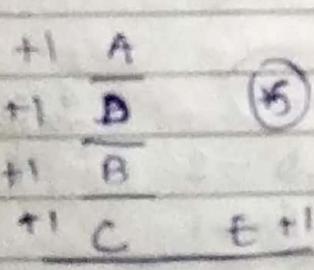


Solutions:

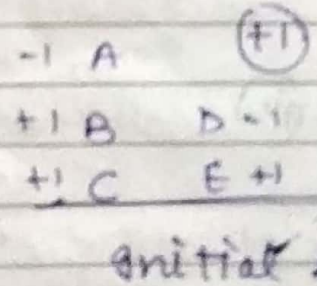
i) Backtracking - A mechanism to store more nodes rather than discarding them. This increases the space complexity but backtracking can be executed.

ii) Random jump - Apply a random func to explore any node in the memory because we didn't discard them.

Block World Problem

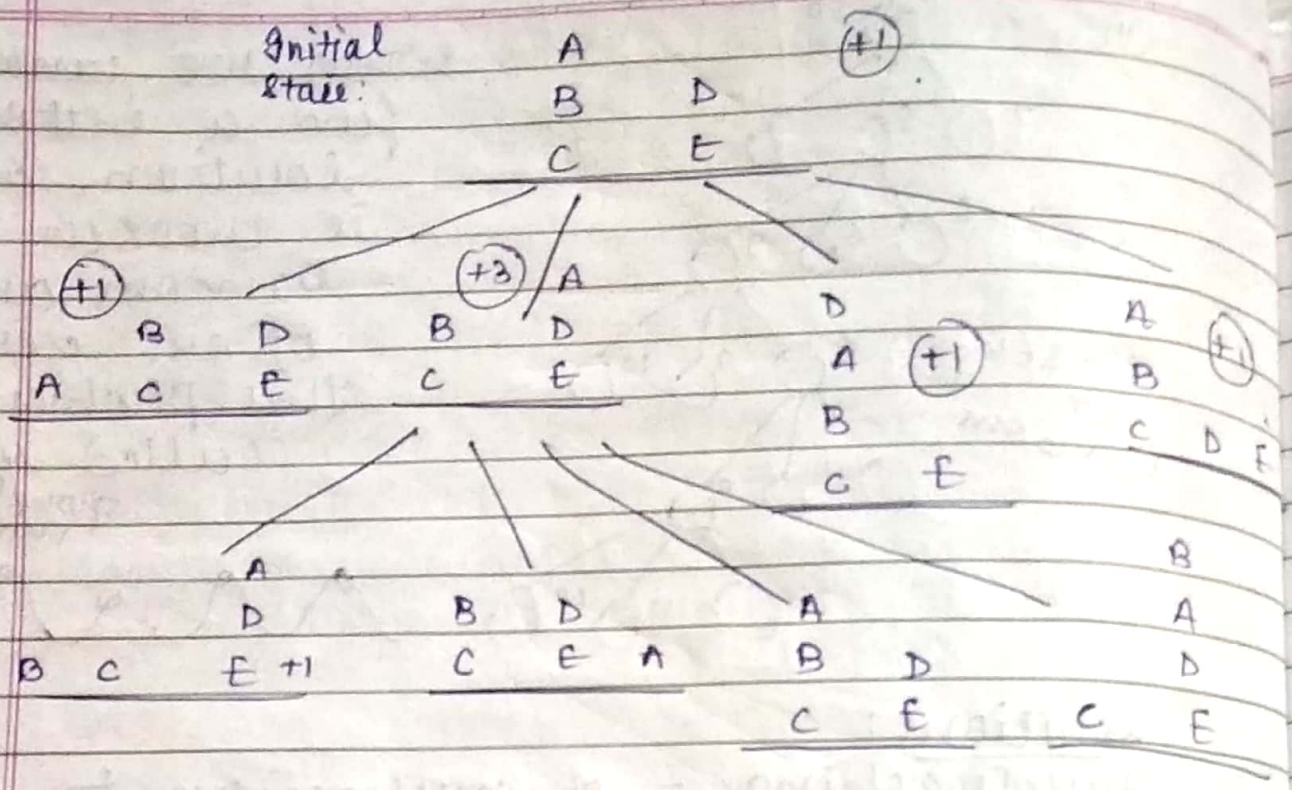


(5)



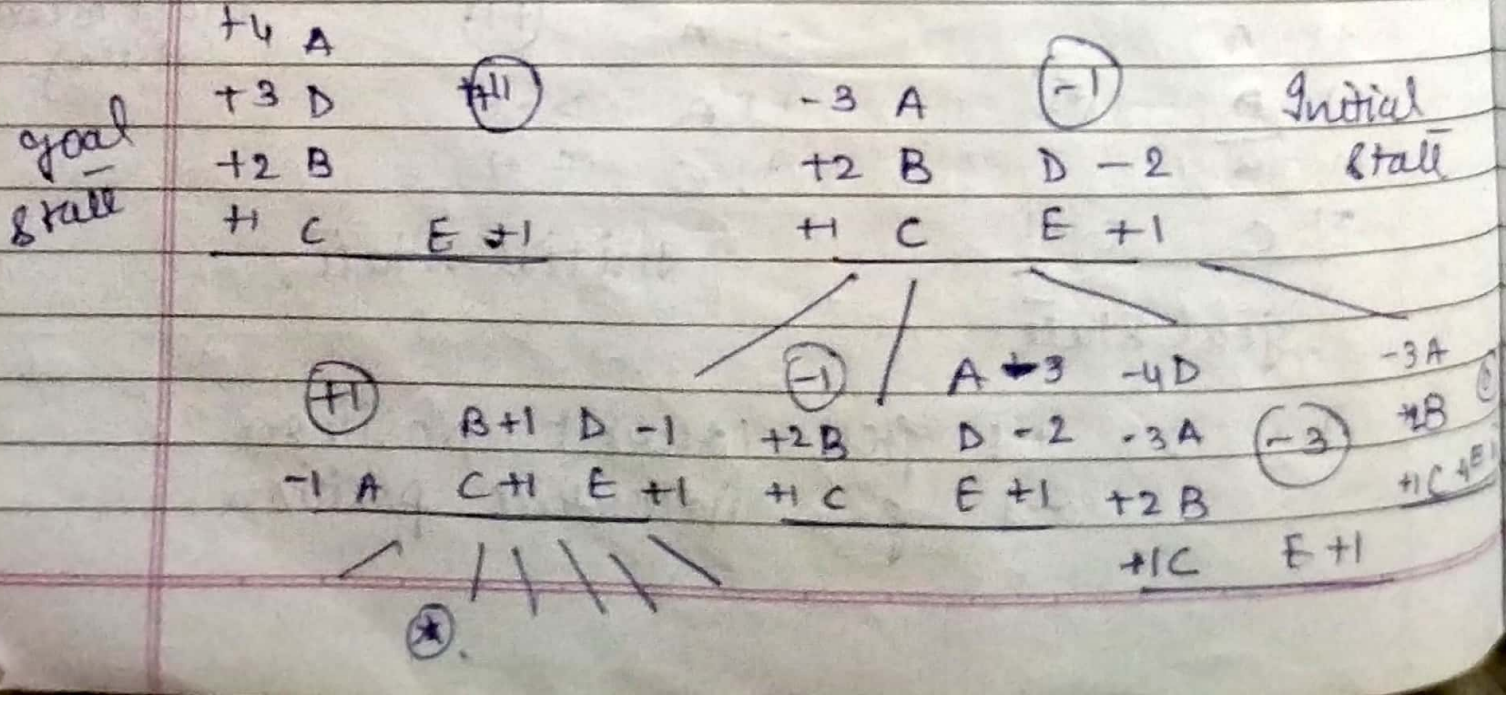
(+1)

h_1 : +1 for correct position
-1 for incorrect position

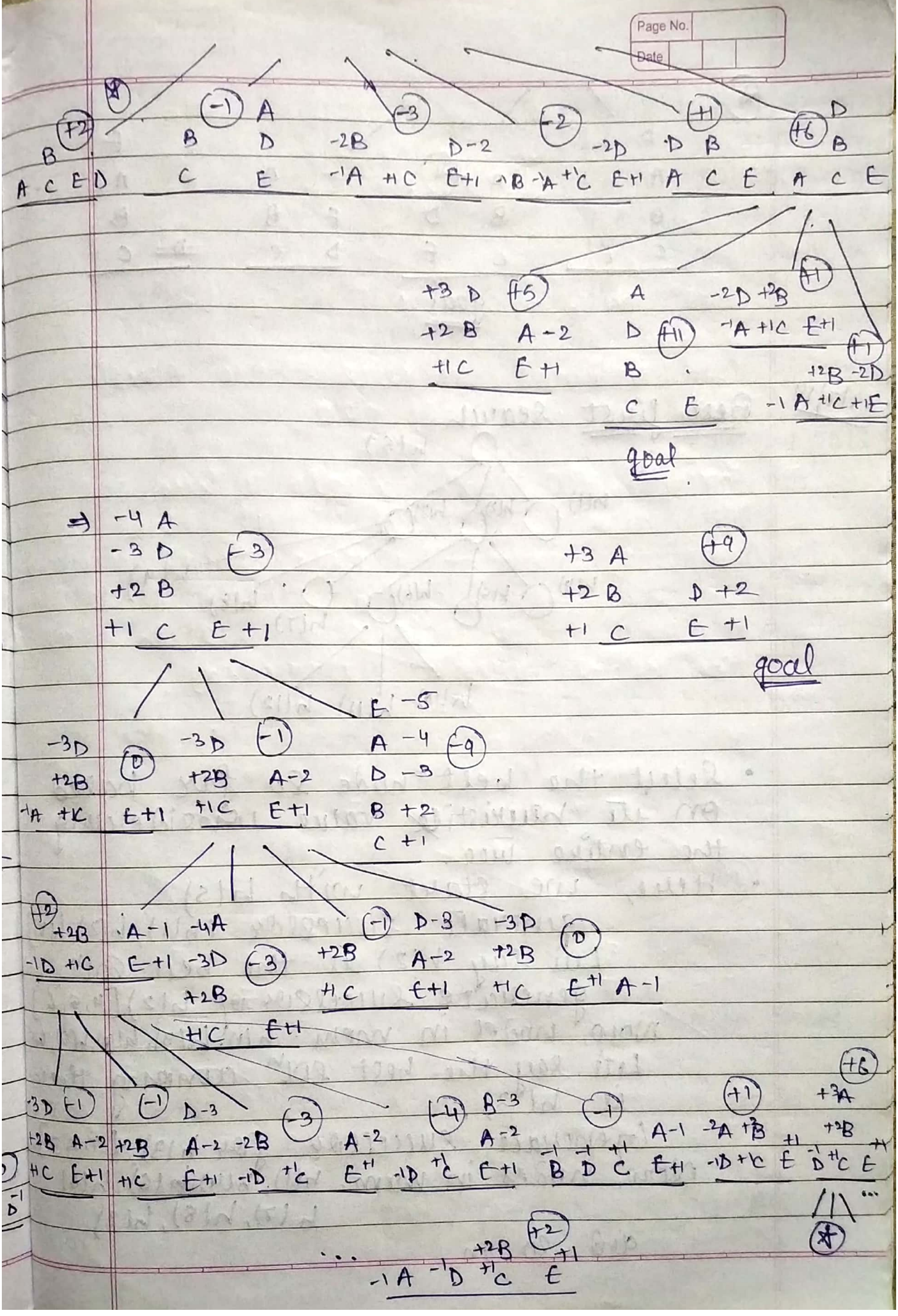


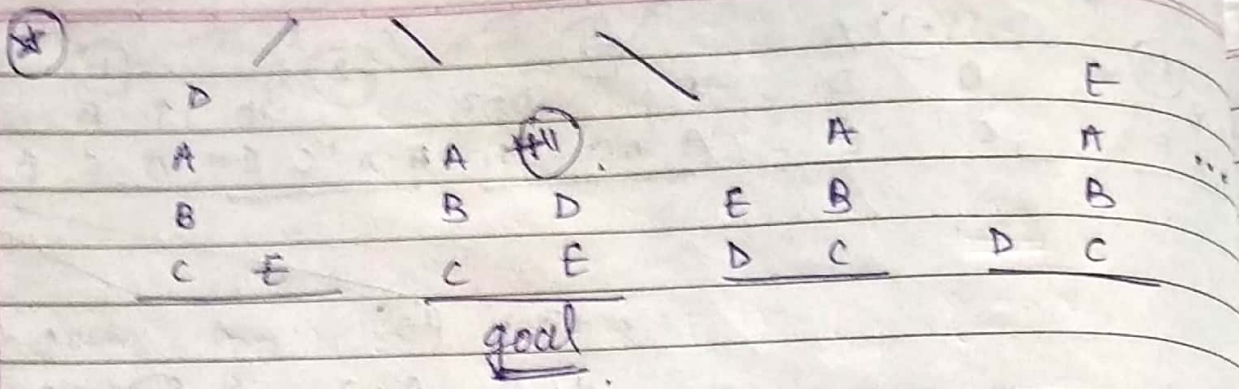
Since, we don't get a better solution than +3, the procedure terminates. Hence, the heuristic func. must be incorrect or the algorithm is incorrect.

h_2: Add +1 for every block in correct structure
: -1 for every block for incorrect structure

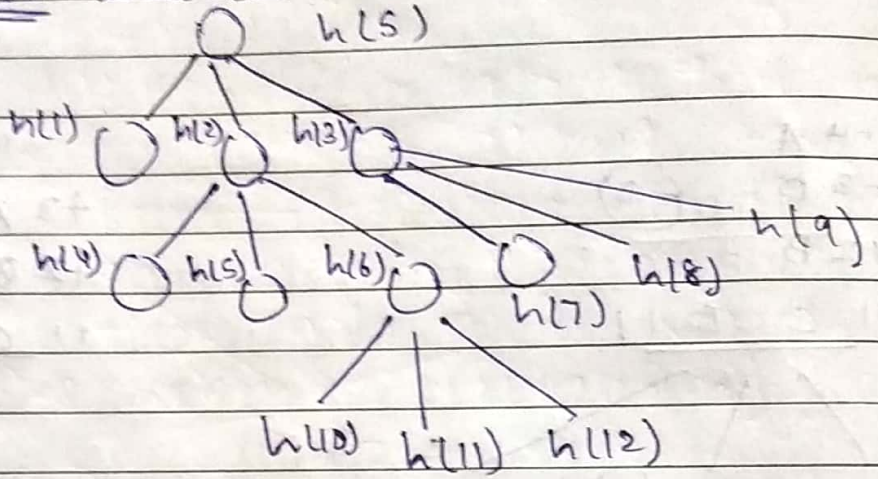


Scanned by CamScanner


Page No.
Date
goal
goal



19/2/19 **Best first search**

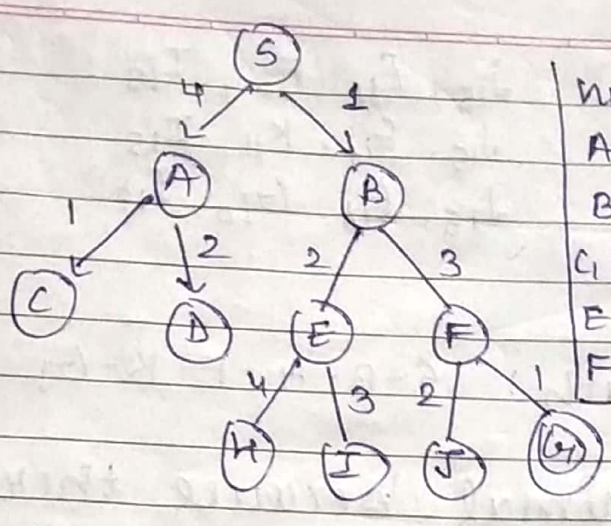


- Select the best node so far based on its heuristic value considering the entire tree.
- Here, we start with h(5)
  generate successor (h(1), h(2), h(3)
  Let's say h(2) is the best so I'll generate successor of h(2) [4,5,
  Now, nodes in mem: h(1), h(3), h(4), h(5)
  Let's say the best so I'll among the is h(3).
  Generate successor for h(3) [4,8,
  Now, nodes in mem: h(1), h(4), h(5), h(6), h(7), h(8), h(9)

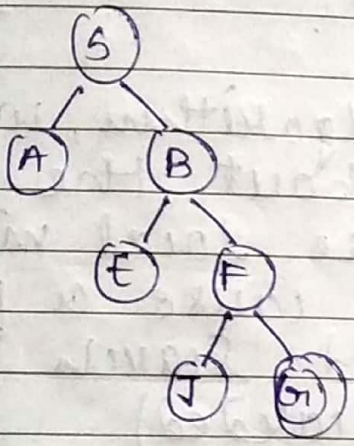
and so on.

Page No.
Date

Ex:

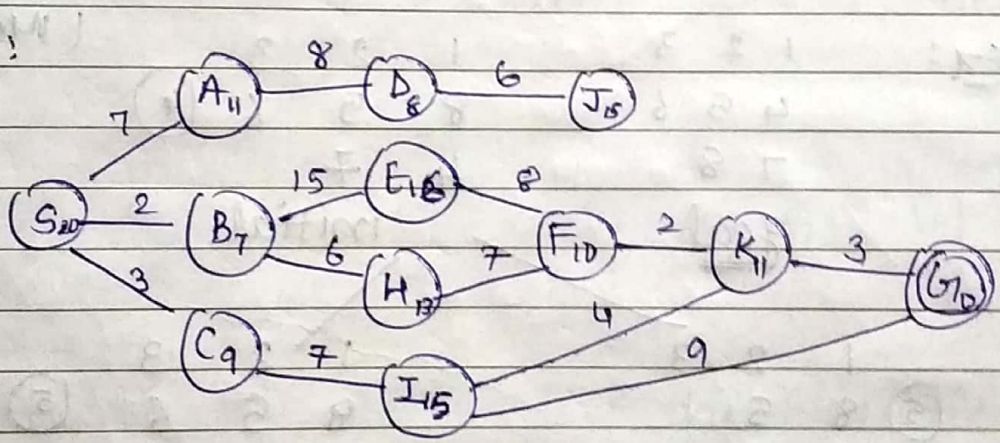


node	h(n)	node	h(n)
A	11	H	7
B	5	I, J	3
C, D	9	S	15
E	4	G_1	0
F	2		



OPEN NODES	CLOSED NODES
S_{15}	
A_{11}, B_5	S_{15}
A_{11}, E_4, F_2	B_5
A_{11}, E_4, J_3, G_{10}	F_2
	G_{10}

Ex:



OPEN NODE	CLOSED NODES
S_{20}	
A_{11}, B_7, C_9	S_{20}
$A_{11}, E_{16}, H_{13}, C_9$	B_7
$A_{11}, E_{16}, H_{13}, I_{15}$	C_9
$D_8, E_{16}, H_{13}, I_{15}$	A_{11}
$J_{15}, E_{16}, H_{13}, I_{15}$	D_8

$J_{15}, E_{16}, F_{10}, I_{15}$ H_{13}
$J_{15}, E_{16}, K_{11}, I_{15}$ F_{10}
$J_{15}, E_{16}, G_{10}, I_{15}$ K_{11}
G_{10}

Sol^n path: S-B-H-F-K-G (20)

Not optimal because there exist path where cost < 20.

⇒ Except Hill climbing algorithm, we do not need to worry about the heuristic func. because local minima or maxima does not cause a problem in any other informed search (as the nodes aren't discarded).

Ex:

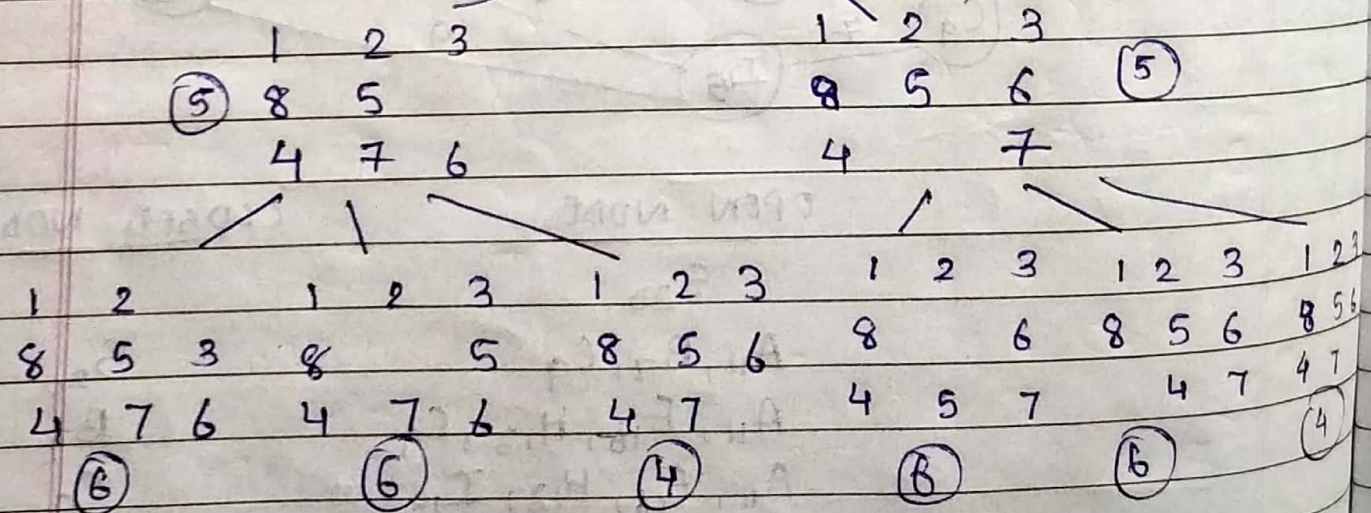
1	2	3
4	5	6
7	8	

goal

1	2	3
8	5	6
4	7	

initial (4)

(Manhattan distance)

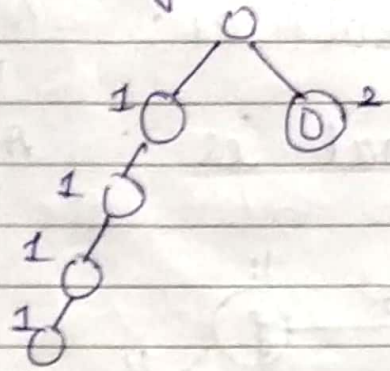


Page No.
Date

Time complexity for Best First Search:
$O(b^d)$

Space complexity = $O(b^d)$

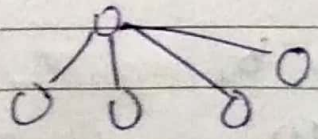
Depth can be infinite but, branching factor is fixed



Thus, it is not complete nor optimal.
[Depends on the heuristic function. Better the heuristic func., better the algorithm]

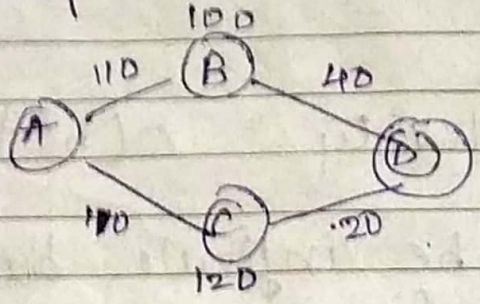
22/7/19 A* algorithm
(*-algo. can be made optimal by applying a no. of constraints)

Instead of considering the huristic value only (for every node), we take into consideration the actual cost too.



$f(n) = h(n) + g(n)$
huristic value, actual value.

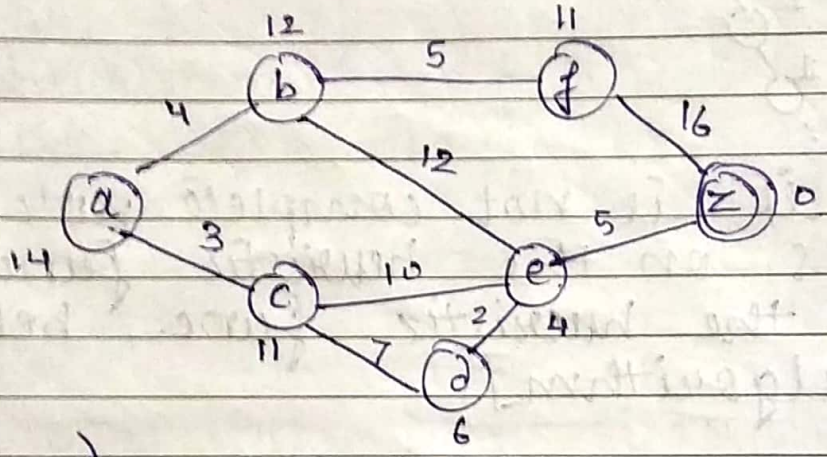
Considering heuristic value only:



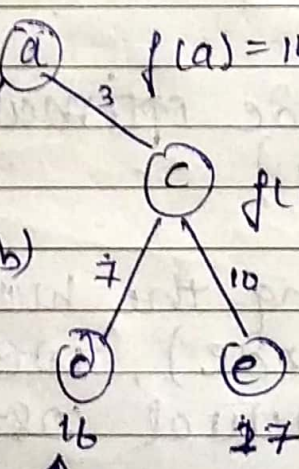
We obtain sol^n as: A - B - D
110 40 = 150.

but, optimal is: A - C - D
110 20 = 130

Ex:



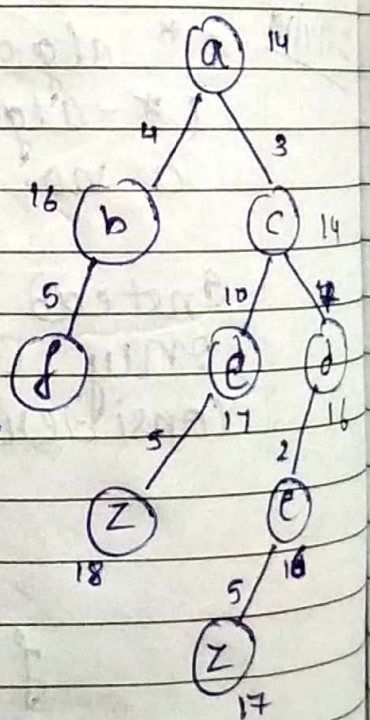
not removed from min. ⇒



f(b) = g(b) + h(b)
= 4 + 12
= 16

f(c) = g(c) + h(c)
= 3 + 11
= 14

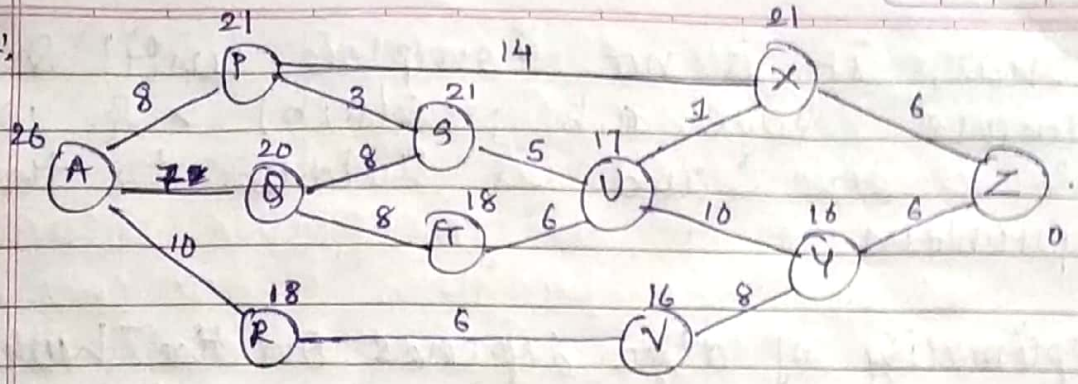
⇒



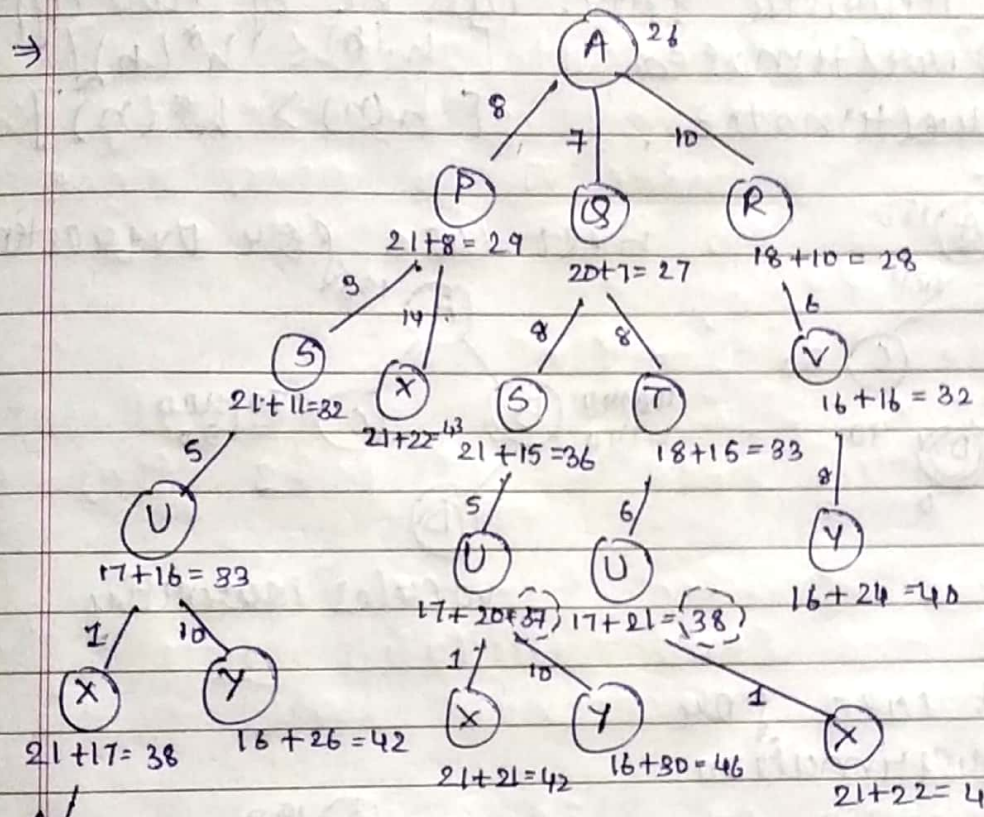
(same as b, so follow alphabetical order).

Page No.
Date

Ex:

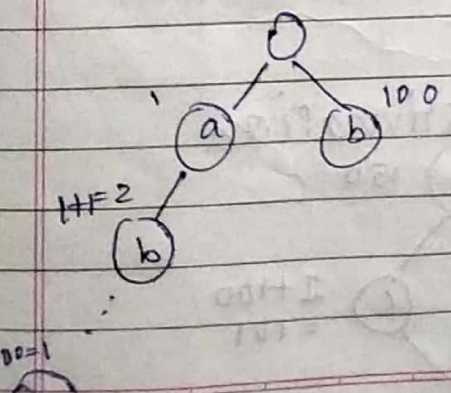


⇒

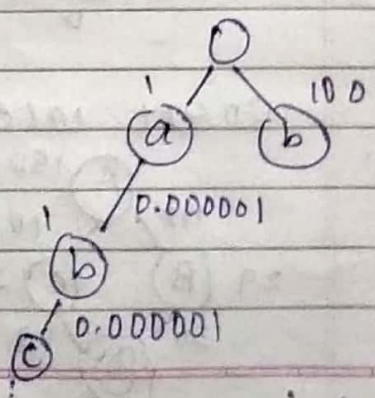


⬚ → not required to explore because we already have a lower cost path till U than 37 or 38

25/7/19



Then, we get a solution

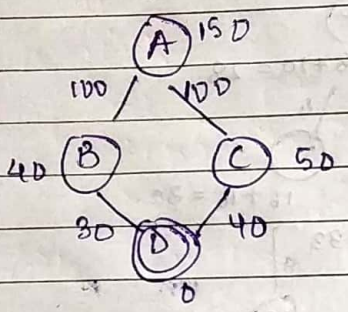


Here, it goes into an infinite loop

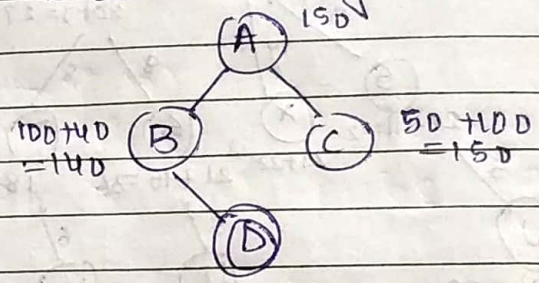
Page No.
Date

Thus, it is not complete until & unless edge a, b ; e(a, b) > ε where ε is +ve and is related to the problem.

Optimality of algo depends on the heuristic func. Heuristic func. can be of two types.
- Underestimated $[h(n) \leq h^*(n)]$
- Overestimated. $[h(n) > h^*(n)]$

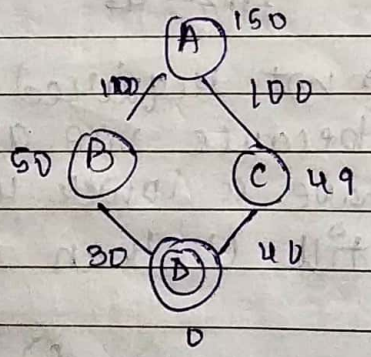


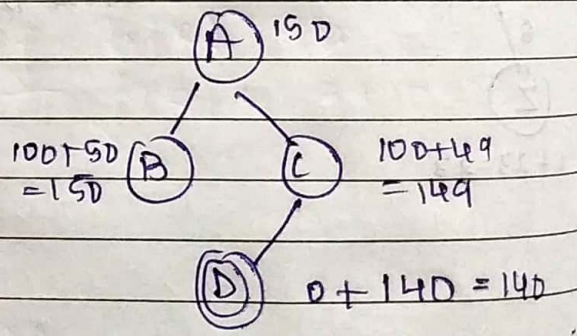
Best case for overestimation



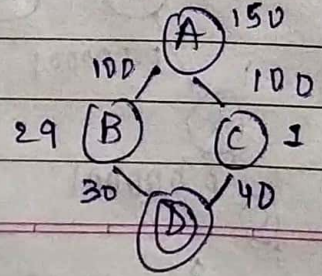
works correctly.

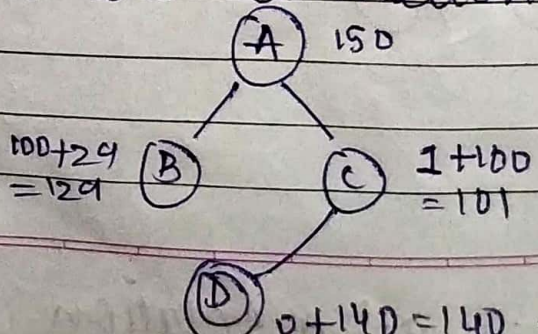
Worst case for overestimation:





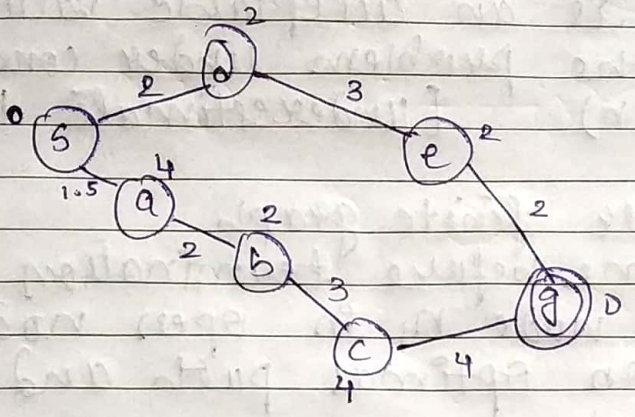
Worst case for underestimation

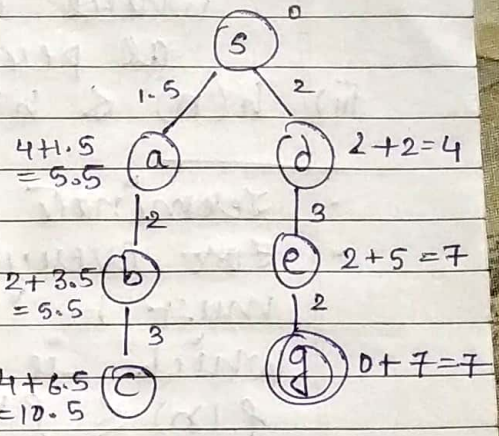




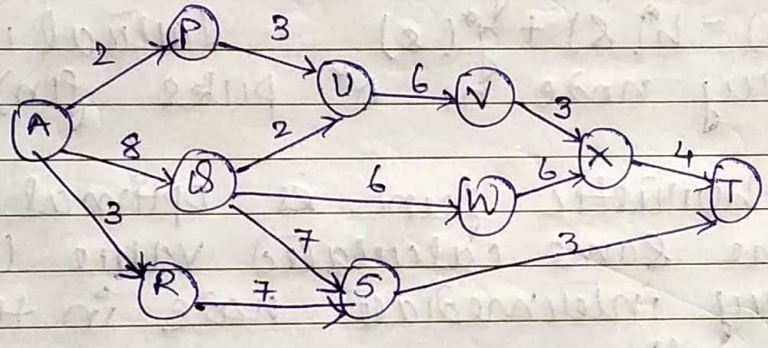
Since, $140 \geq 129$, the algo. goes on to explore B and thus, gets the optimal path. Hence, we should go for underestimation

Ex:





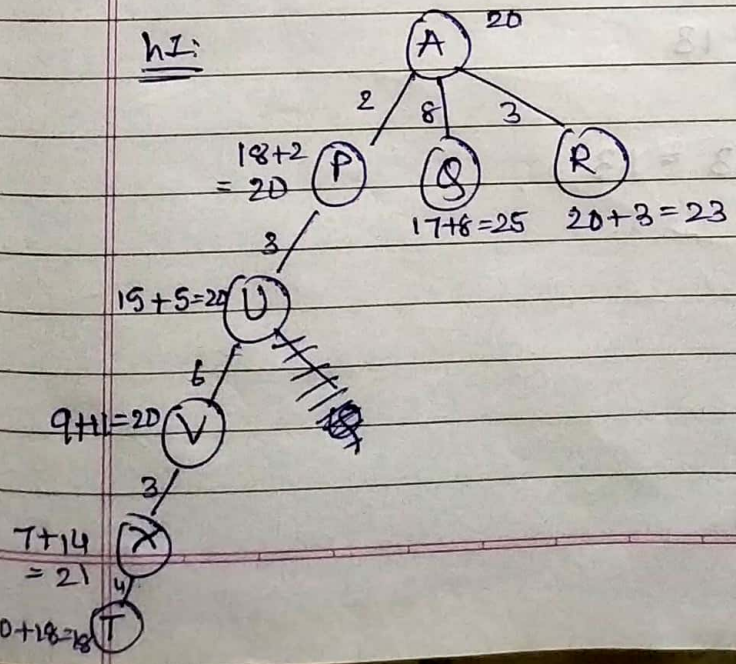
Ex:



	h_1	h_2
A	20	20
P	18	8
Q	17	14
R	20	12
S	10	3
U	15	12
V	9	3
W	20	2
X	7	3
T	0	0

[h_1 → Overestimated heuristic func]
[h_2 → Underestimated H.F].

h_1:



Conditions for optimality:

i) finite bounding factors.

ii) for each edge, cost $e(a,b) > \epsilon$; (depth)
where ϵ is an acceptable value as per the problem under consideration

iii) $h(n) \leq h^*(n)$ [underestimate HF]

- Terminate for finite graph
- ~~For~~ everytime before termination, there must be a node n_i in open node which is on optimal path and
  $f(n) \leq f^*(s)$
  $f(n) = h(n) + g(n)$
  $f^*(s) = h^*(s) + g^*(s)$ [actual cost]
- for every node n, A* picks $f(n) \leq f^*(s)$

If the heuristic func. is optimal, it gives the same calculated value ($f(n)$) for every intermediate node in the optimal path.

In previous ex: If $h(R) = 10$
$f(R) = 8 + 10 = 13$

$h(S) = 3$
$f(S) = 3 + 10 = 13$
$h(T) = 0$
$f(T) = 3 + 7 + 3 = 13$