State space search

Here the state space where to be placed is search for achieving goal by using left, right, up, down operations

The initial & goal positions of 8 puzzle board is given below

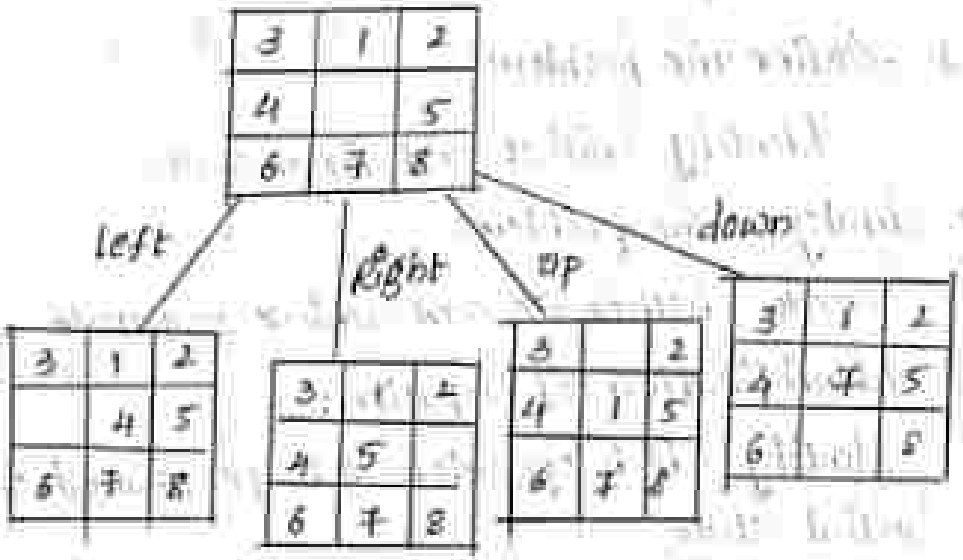
3	1	2
4		5
6	7	8

Initial

	1	2
3	4	5
6	7	8

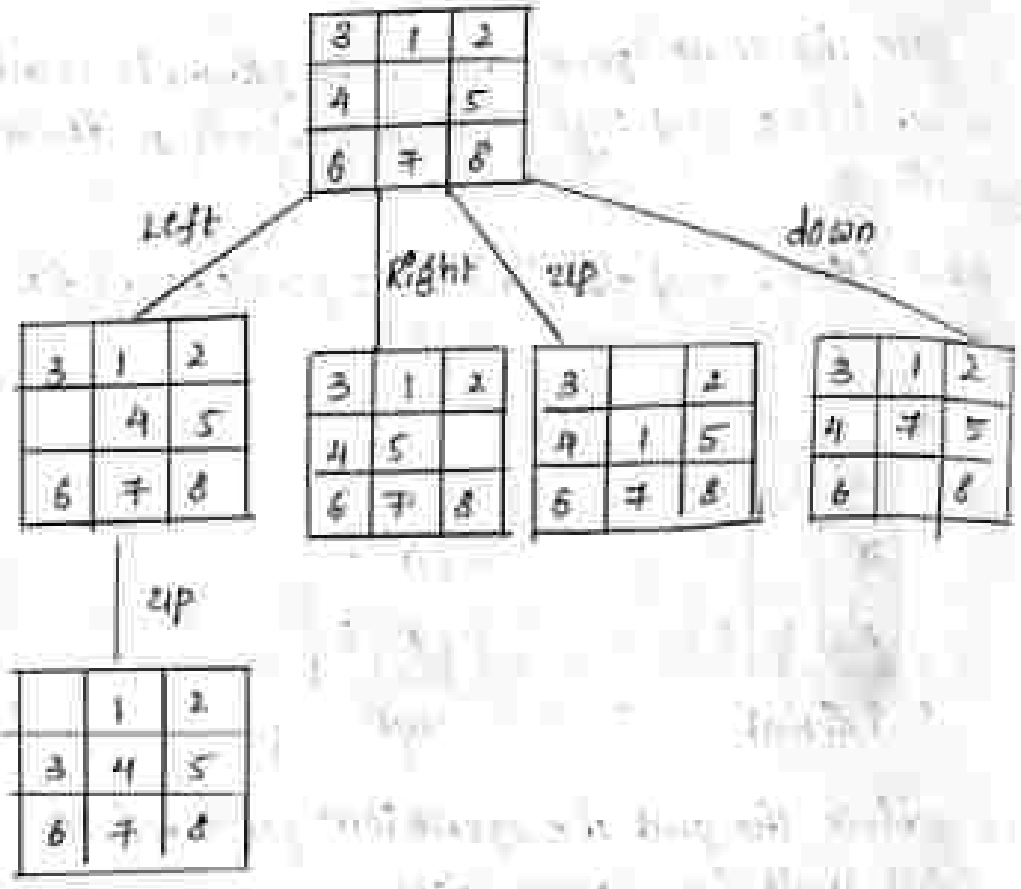
Goal

To achieve the goal the operations performed on initial state is shown below



In the above figure we didn't find the solution. So look at the solutions we got & any possibility of achieving goal

thus by performing up operation on sub solutions of left operation we achieve our goal as shown below



To solve the problem

1. Define the problem:

   Identify initial and final states

2. Analyze the problem:

   check whether we can achieve goal state

3. Identification of solution:

   Identify which operation to be performed on the initial state

4. check the best solution:

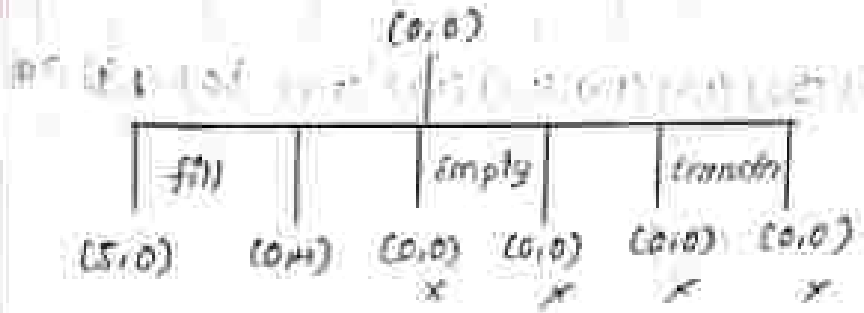
   see the obtained solution whether we can achieve goal or choose a sub solution from which we can achieve goal.

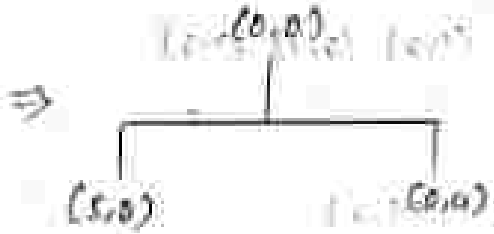
Water Jug problem

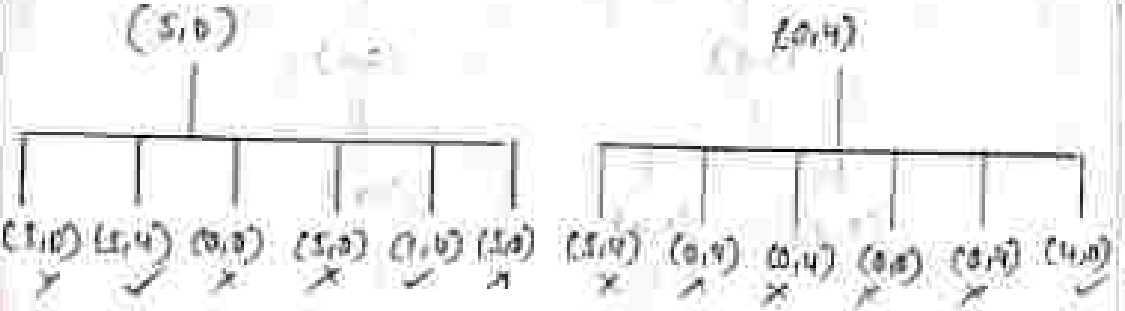
Initially we have two jugs of capacity 5L & 4L. Source are initial state or starting state (0,0), destination point to find is (2,0)

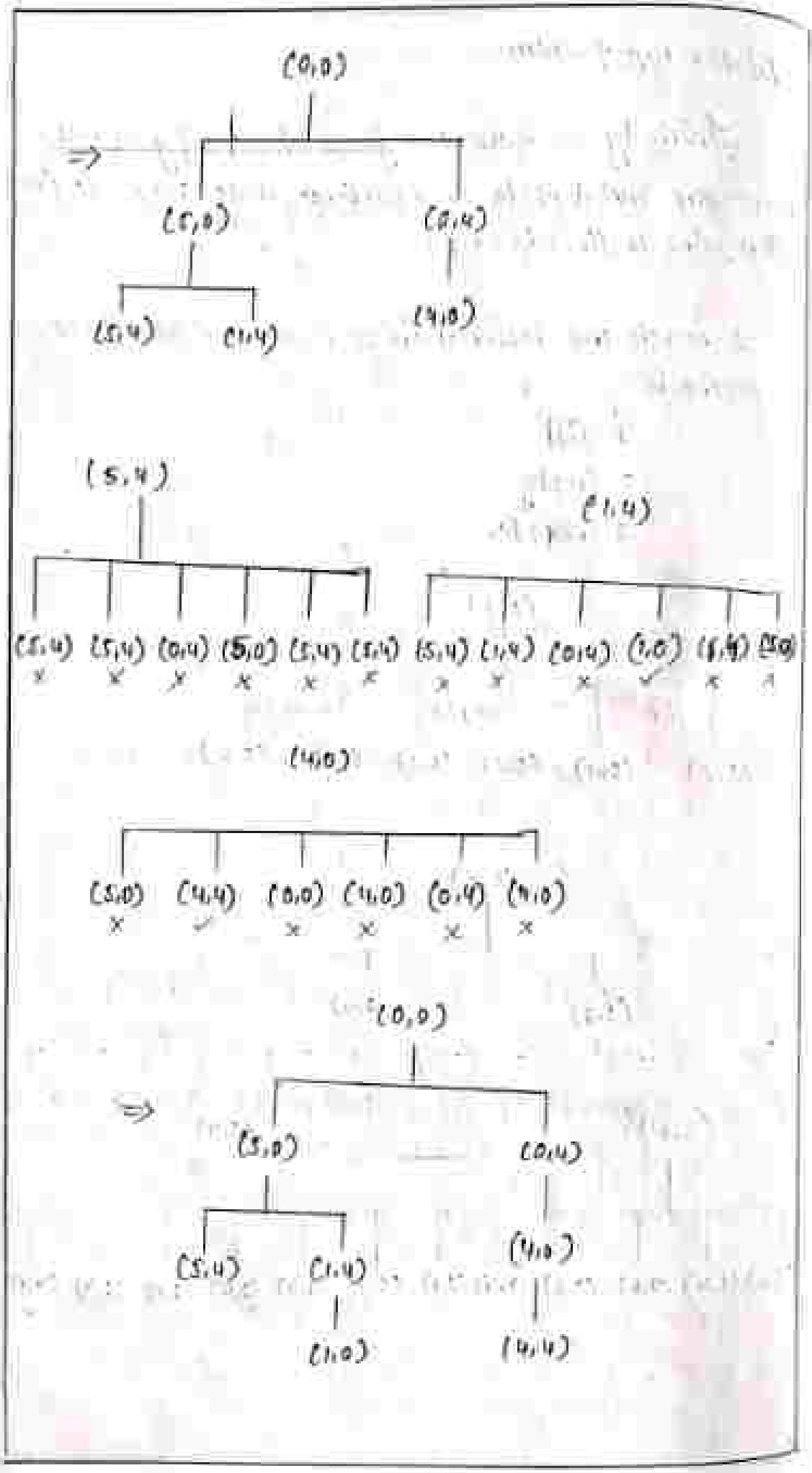
To reach the destination to reach we use three methods:

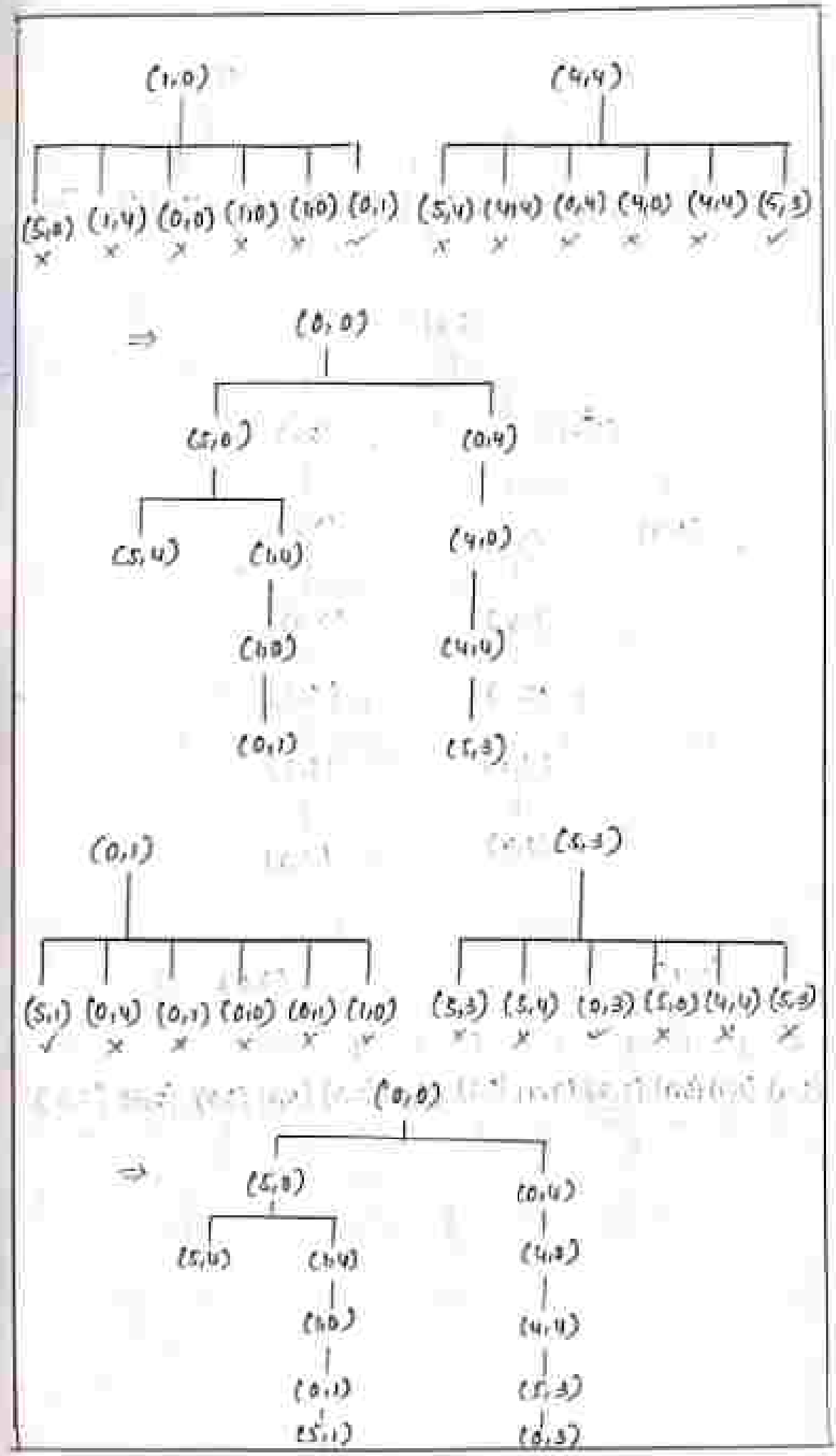
1. fill
2. Empty
3. transfer

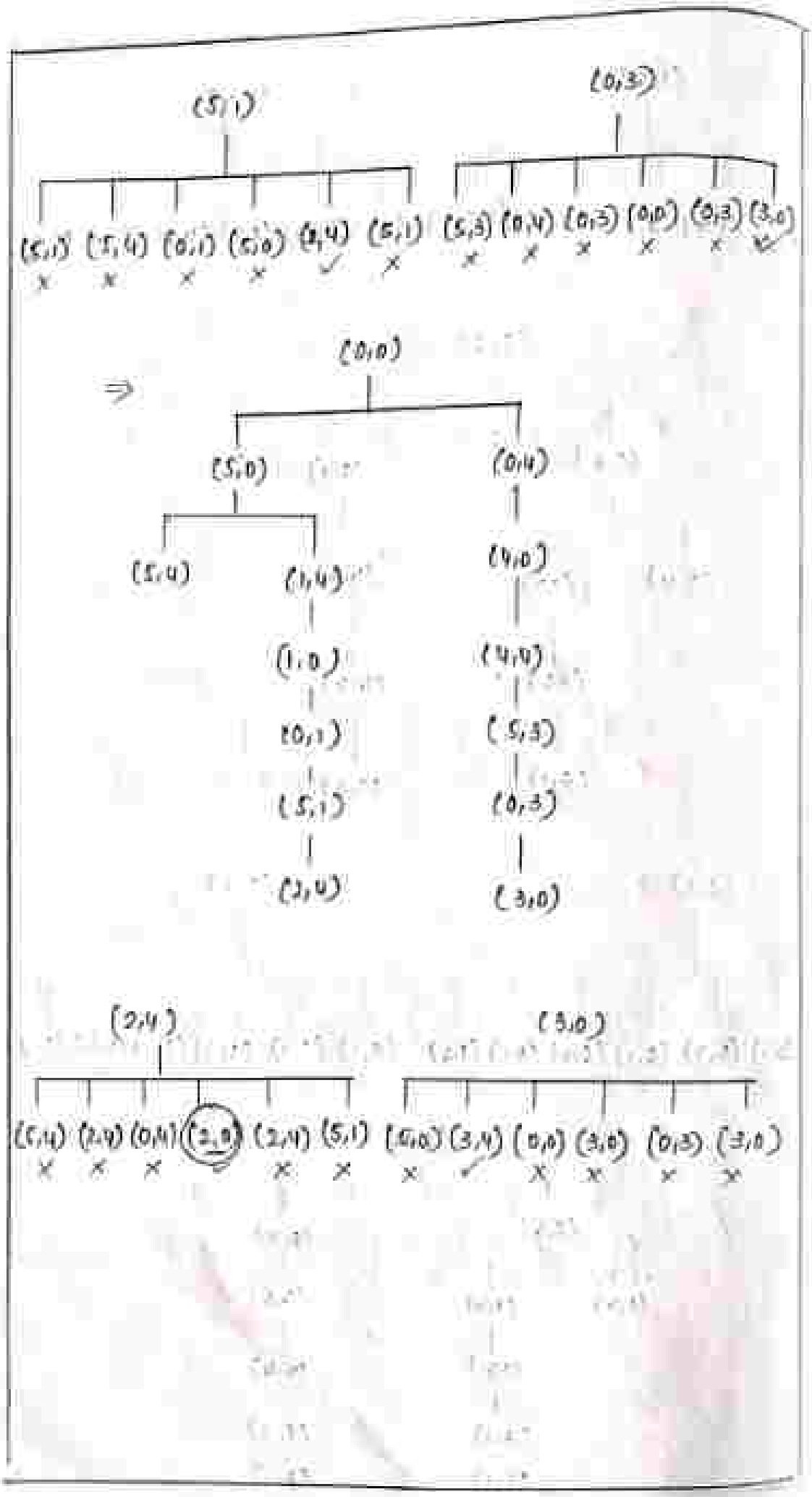


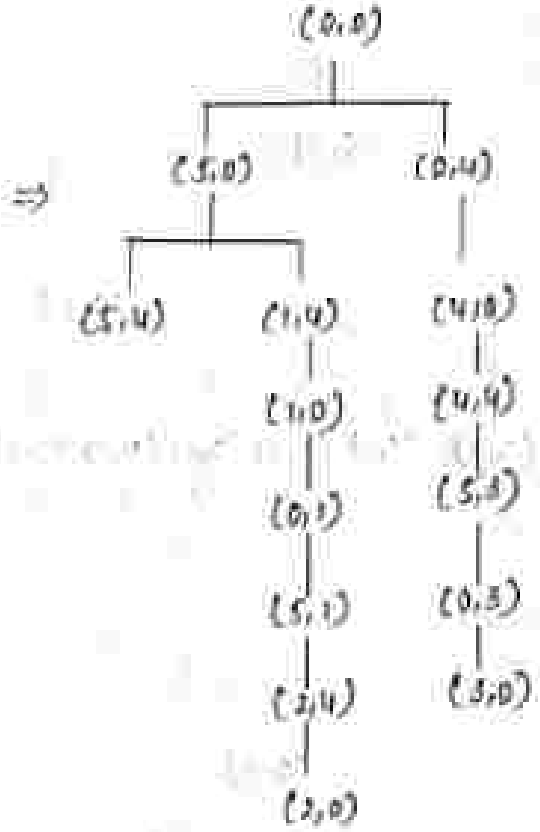





(0,0)
(5,0)
(0,4)
(5,4)
(1,4)
(4,0)
(5,4)
(5,4)
(0,4)
(5,0)
(5,4)
(5,4)
(5,4)
(1,4)
(0,4)
(1,0)
(4,0)
(5,0)
(4,4)
(0,0)
(4,0)
(0,4)
(4,0)
(0,0)
(5,0)
(0,4)
(5,4)
(1,4)
(4,0)
(1,0)
(4,4)


(1,0)
(4,4)
(5,0) (1,4) (0,0) (1,0) (4,0) (0,1)
(5,4) (4,4) (0,4) (4,0) (4,4) (5,3)
⇒
(0,0)
(5,0)
(0,4)
(5,4)
(1,4)
(4,0)
(1,0)
(4,4)
(0,1)
(5,3)
(0,1)
(5,3)
(5,1) (0,4) (0,1) (0,0) (0,1) (1,0)
(5,3) (5,4) (0,3) (5,0) (4,4) (5,3)
⇒
(0,0)
(5,0)
(0,4)
(5,4)
(1,4)
(4,3)
(1,0)
(4,4)
(0,1)
(5,3)
(5,1)
(0,3)


(5,1)
(0,3)
(5,0) (0,4) (4,4) (5,1)
(5,3) (0,4) (0,3) (0,0) (0,3) (3,0)
⇒
(0,0)
(5,0)
(0,4)
(5,4)
(1,4)
(4,0)
(1,0)
(4,4)
(0,1)
(5,3)
(5,1)
(0,3)
(2,4)
(3,0)
(2,4)
(3,0)
(5,4) (2,4) (0,4) (2,0) (2,4) (5,1)
(5,0) (3,4) (0,0) (3,0) (0,3) (3,0)

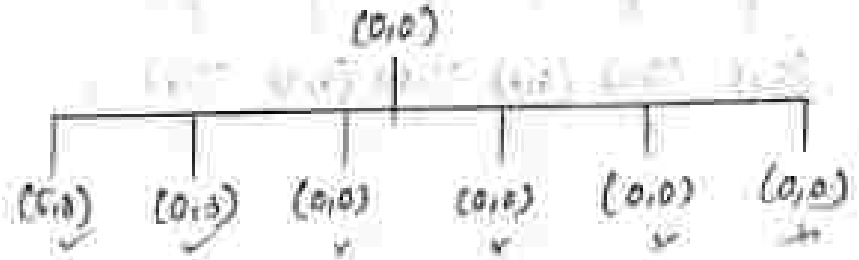


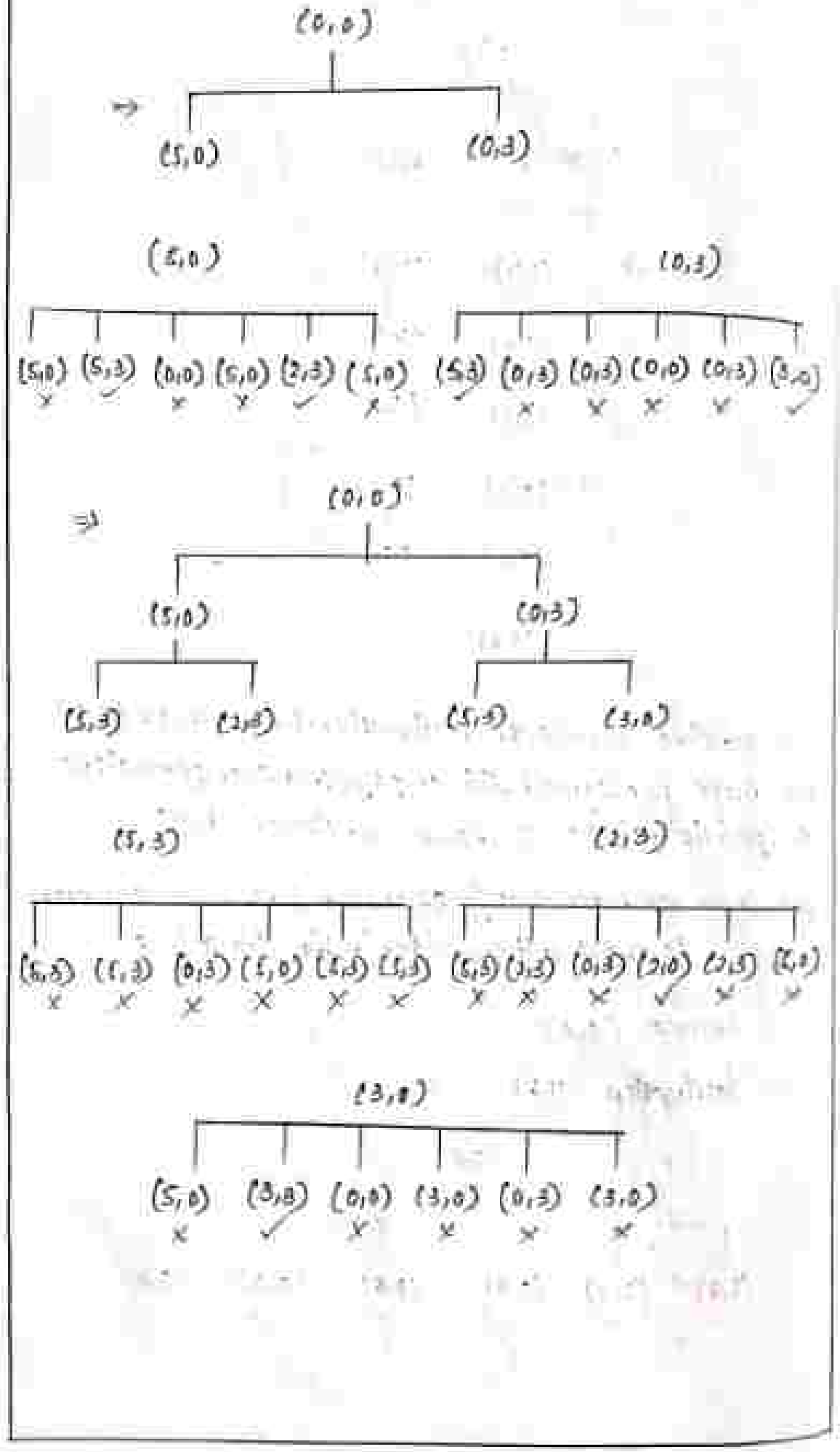
To achieve source to destination i.e (0,0) to (3,0) we have performed fill, empty, transfer operations at specific points as shown in figure above

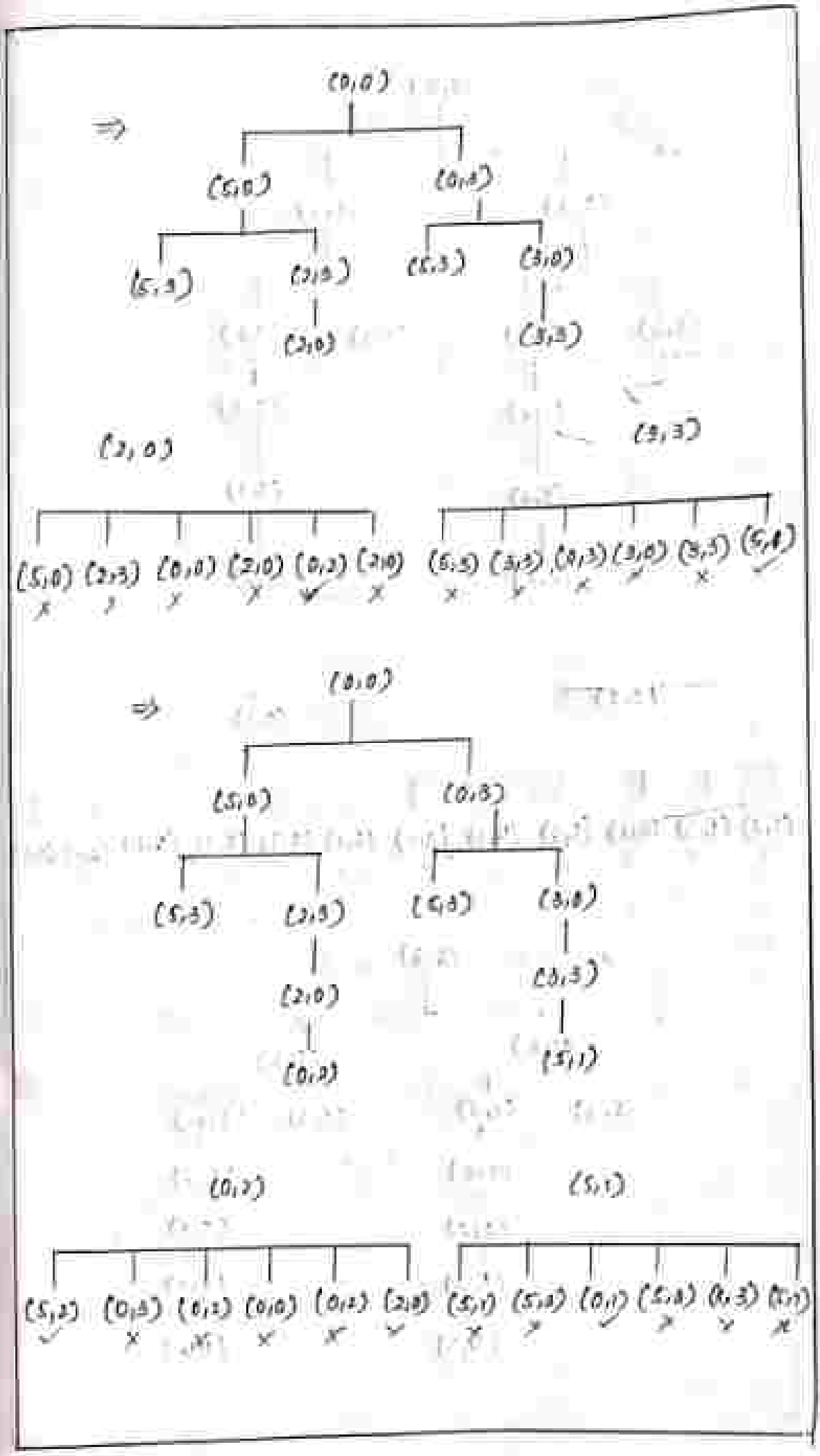
problem: We have two jugs a 5L and other 3L. Our task is to get 4 liters water in the five lt jug

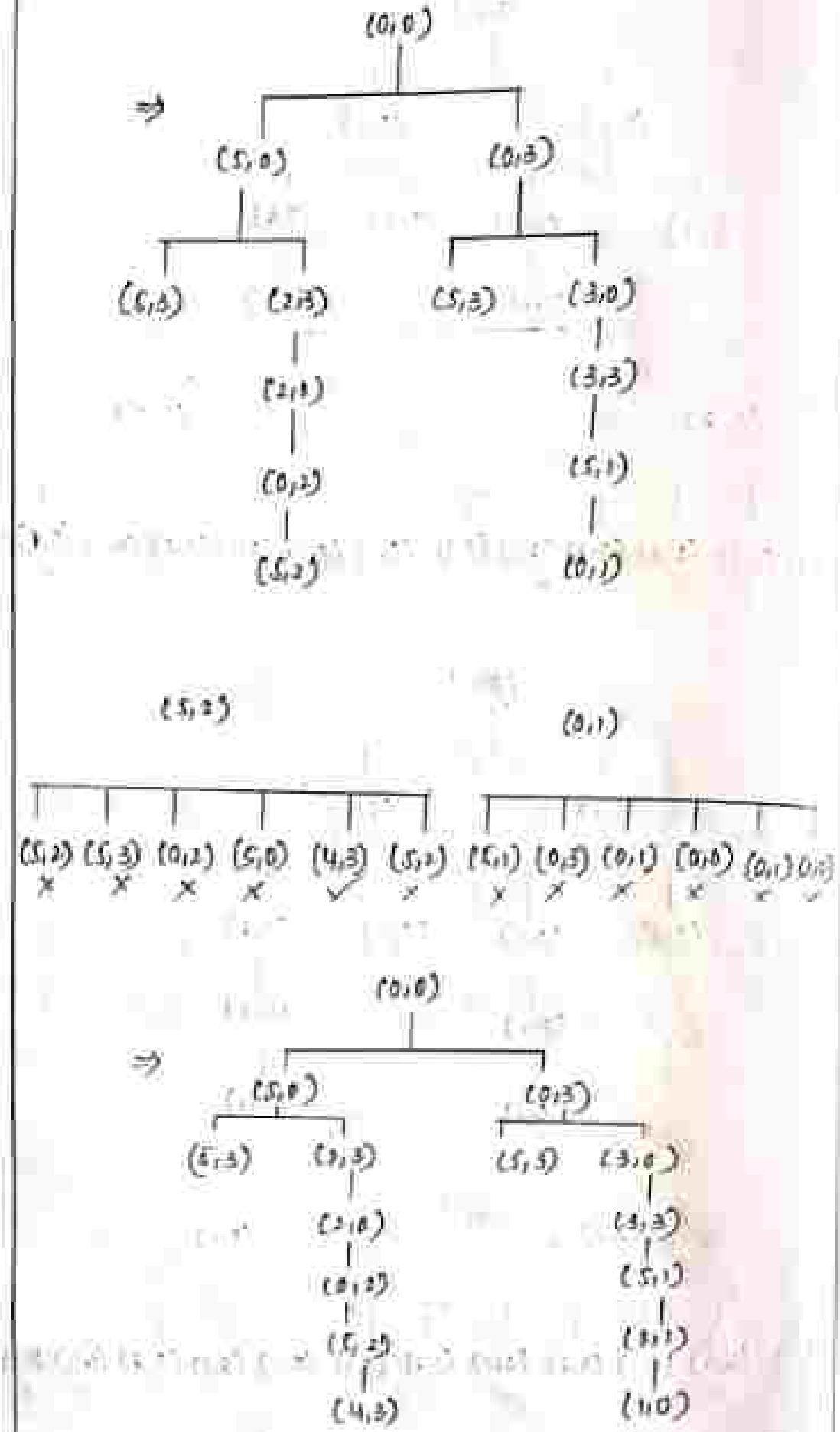
Source (0,0)

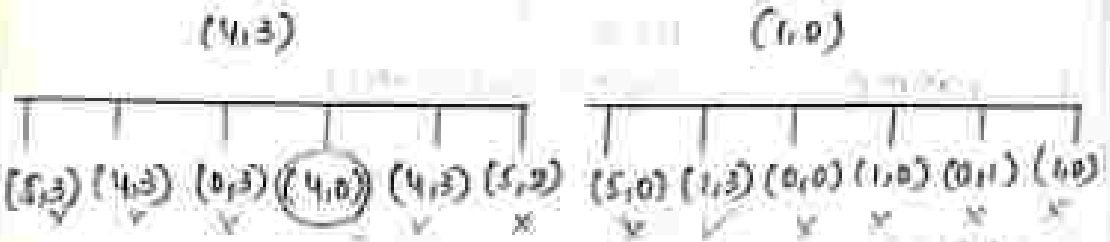
destination (4,0)

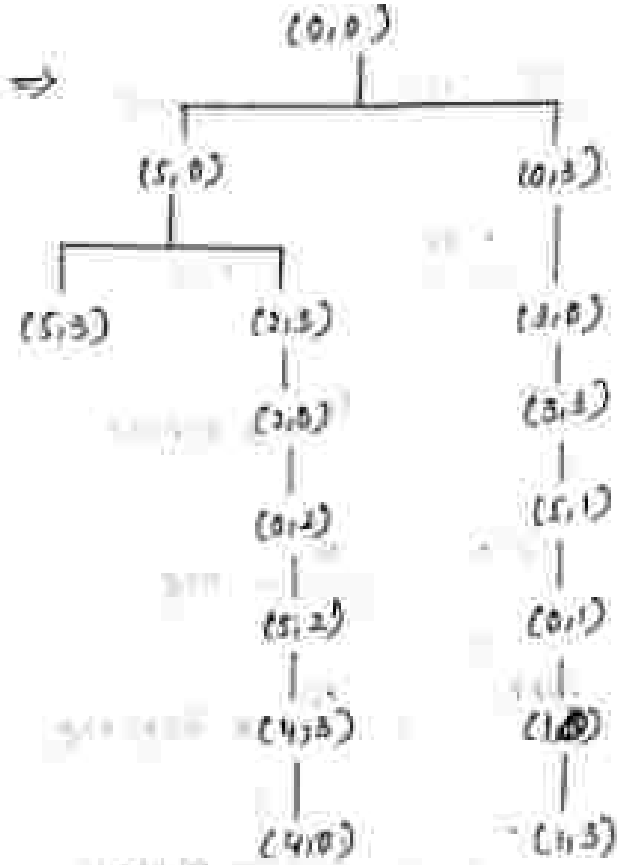



(0,0)
(5,0)
(0,3)
(5,0)
(0,3)
(5,0) (5,3) (0,0) (5,0) (2,3) (5,0)
(5,3) (0,3) (0,3) (0,0) (0,3) (3,0)
(0,0)
(5,0)
(0,3)
(5,3)
(2,3)
(5,3)
(2,3)
(5,3) (1,3) (0,3) (5,0) (1,3) (5,3)
(5,5) (3,3) (0,3) (2,0) (2,3) (2,3)
(3,0)
(5,0) (3,3) (0,0) (3,0) (0,3) (3,0)


(0,0)
⇒
(5,0)
(0,3)
(5,3)
(2,3)
(2,0)
(3,0)
(3,3)
(3,3)
(2,0)
(5,0) (2,3) (0,0) (2,0) (0,2) (2,0)
(0,3) (3,3) (3,0) (3,5) (5,1)
(0,0)
⇒
(5,0)
(0,3)
(5,3)
(2,3)
(2,0)
(0,2)
(5,3)
(3,0)
(3,3)
(5,1)
(0,2)
(5,1)
(5,2) (0,3) (0,2) (0,0) (0,2) (2,0)
(5,1) (5,0) (0,1) (2,3) (3,3) (5,1)


(0,0)
(5,0)
(0,3)
(5,3)
(3,0)
(3,3)
(5,1)
(0,1)
(2,3)
(2,0)
(0,2)
(5,2)
(5,2)
(0,1)
(5,1) (5,3) (0,2) (5,0) (4,3) (5,2)
(5,1) (0,3) (0,1) (0,0) (0,1) (0,3)
(0,0)
(5,0)
(0,3)
(5,3)
(3,3)
(3,0)
(3,3)
(5,1)
(2,0)
(0,2)
(5,2)
(4,3)
(1,0)





To achieve source to destination ie (0,0) to (4,0) we perform fill, transfer, Empty operations as shown

Missionaries and cannibals problem

Condition: m ≥ c

1. MMMC —Bcc→ Bcc

2. MMMCC ←cB— C

3. MMM —cc B→ Bccc

4. MMMC ←cB— cc

5. MC —MM B→ MMCC

6. MMCC ←MC B— MC

7. CC —MM B→ MMMC

8. CCC ←C B— MMM

9. C —CC B→ MMMCC

10. CC ←C B— MMMC

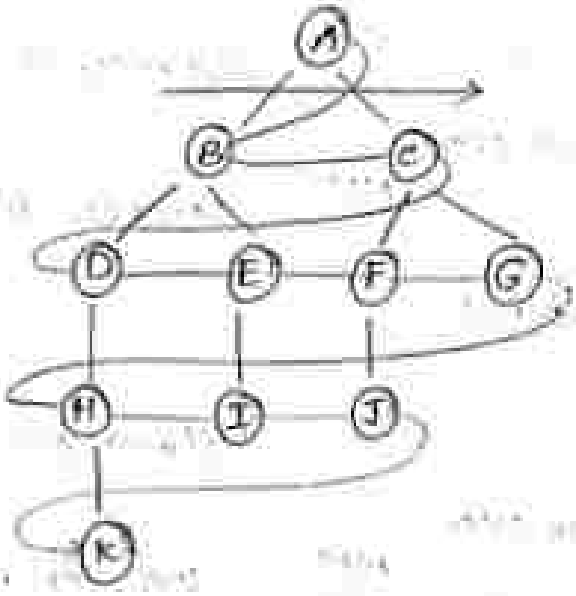
11. —CC B→ MMMCCC

left hand	Right hand
10) one cannibal return back left over: ccc	c → left over: cmmmc
11) two cannibals cross over left over: 0	cc → left over: mmmccc

Breadth First Search (BFS):

This algorithm searches or explores nodes or points breadth wise. i.e only after exploring / searching the entire points on same level it moves to the next level

Ex:



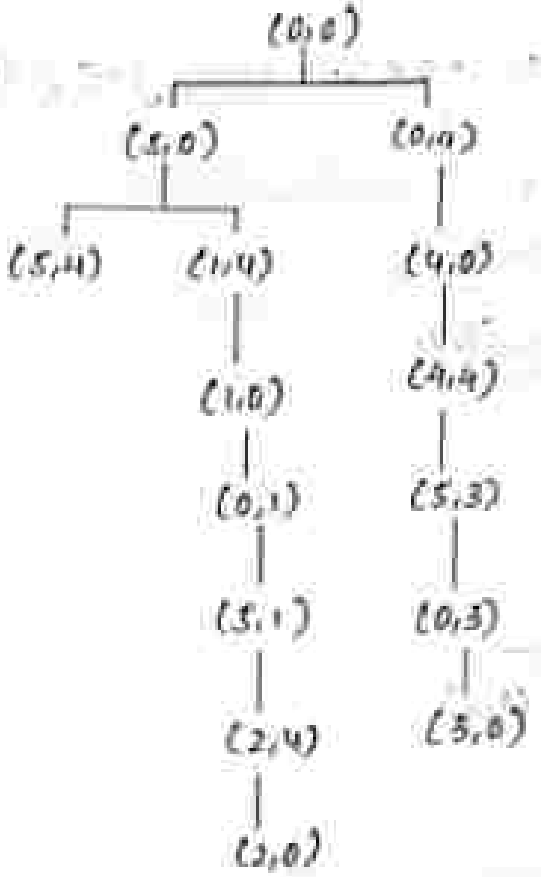
Here our source point is A and the goal is K

for that we explored the points as shown below by using Breadth-first search

A → B → C → D → E → F → G → H → I → J → K

We can also solve water jug problem by using BFS

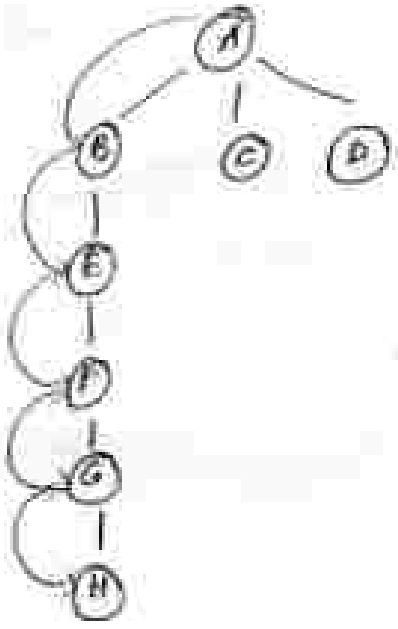
Here source point is (0,0) and target (2,0)



Depth first search (DFS):

Here exploration was done on the depth/vertically

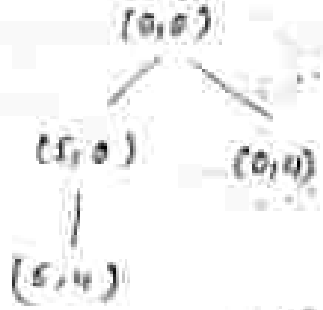
Ex:



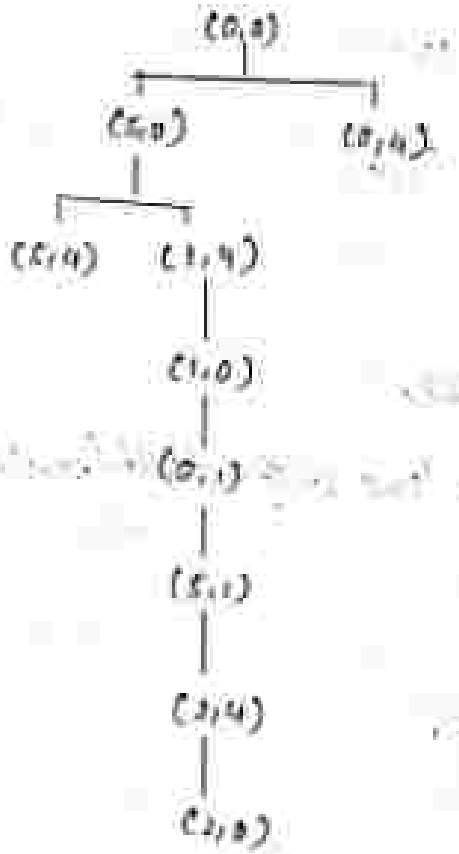
Here we may take less time to reach the destination but there is a disadvantage of getting the infinite loop.

Here DFS are used to check the water jug problem.

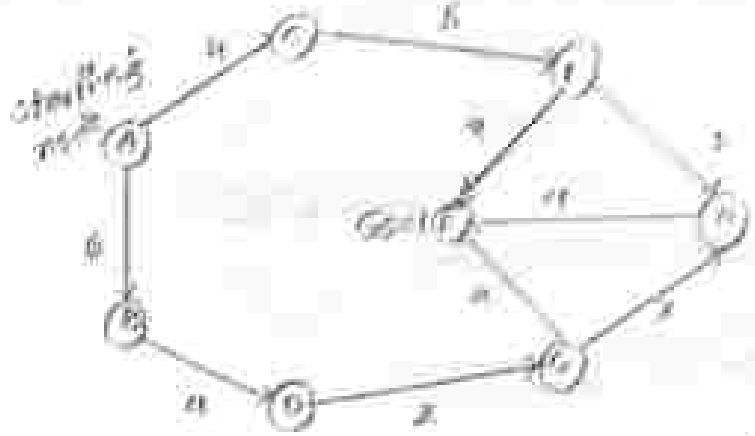
Ex:



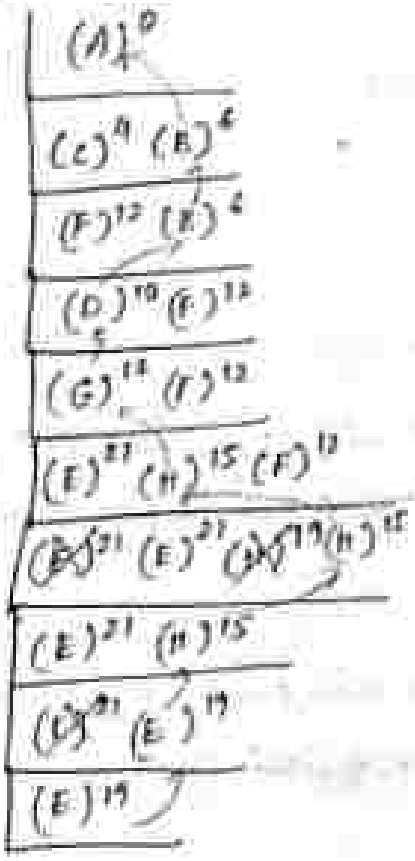
Ex:

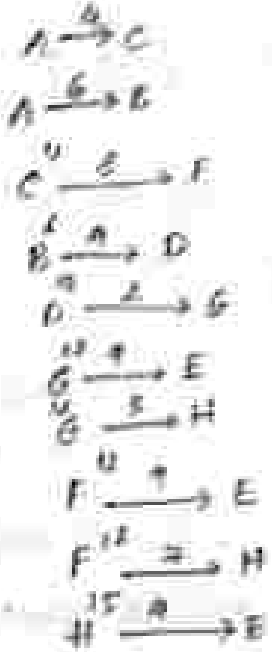


Uniform cost search



Open list:

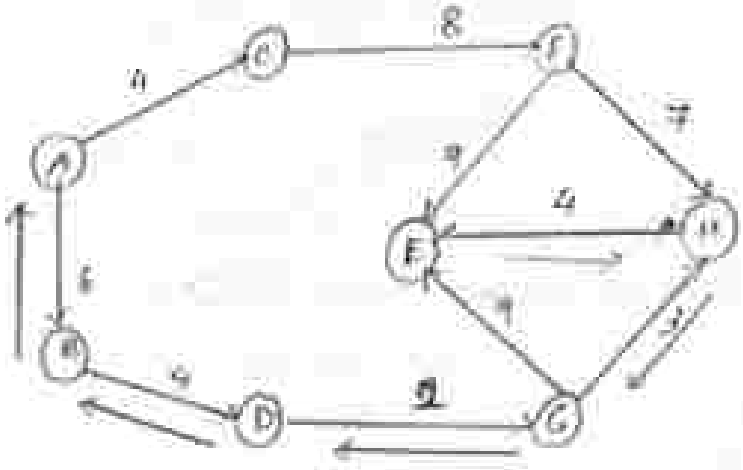




close list:

(A)0 (C)9 (B)6 (D)10 (G)12 (F)12 (H)15

The backtrack is provided as follows



Order / path of given

A – B – D – G – H – E

path cost:

6 + 4 + 2 + 3 + 4 = 19

Uniform cost search not only find one path from initial node to goal node but it will find all the alternate path.

Uniform cost search will provide multi ways to reach goal but with the lowest cost.

Bidirectional Search

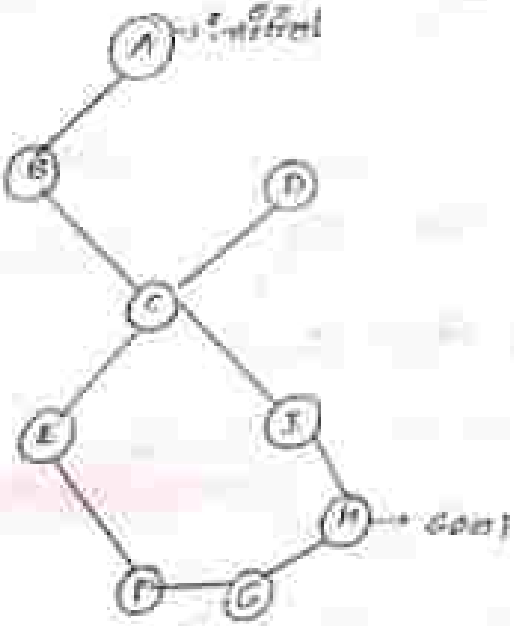
It means searching from two directions i.e

forward direction

Backward direction

Forward: Searching from root/initial node we perform BFS in forward search

Backward: Searching from goal node we perform DFS in backward direction



forward:

A - B

A - B - C

Backward

H - I

H - I - C

A - B - C
H - I - C

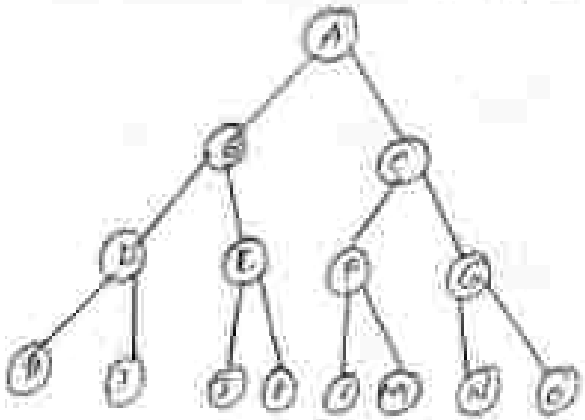
It means there exists a path to reach goal. i.e if we reach the common node.

DLS (Depth Limit Search):

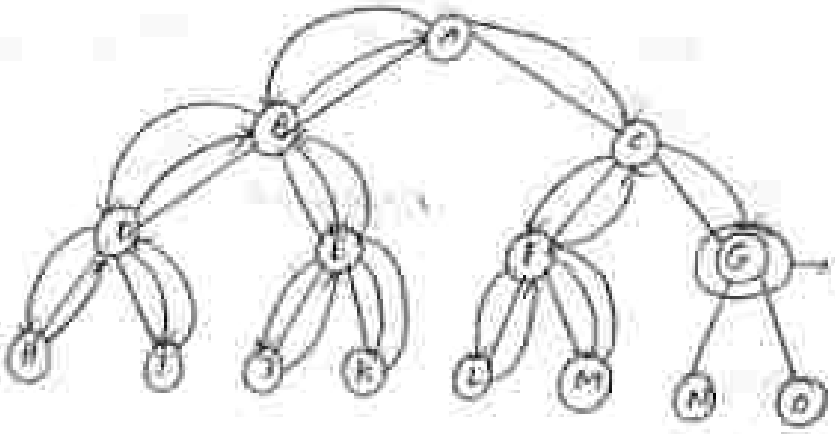
This algorithm was developed to overcome the drawback of DFS.

Below we are elaborate an example to reach goal state from initial state by DFS making use of backtracking

DFS:



backtracking:

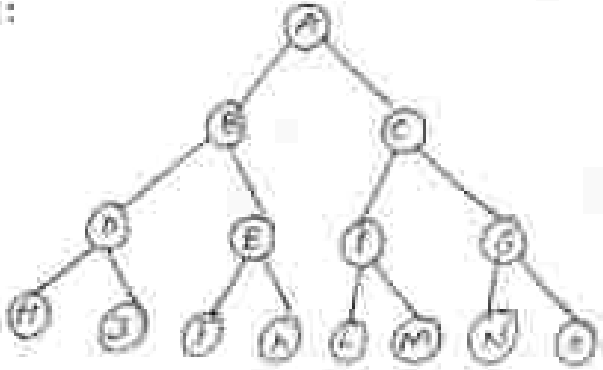


Path: A B D H I E J K C F L M G

DLS:

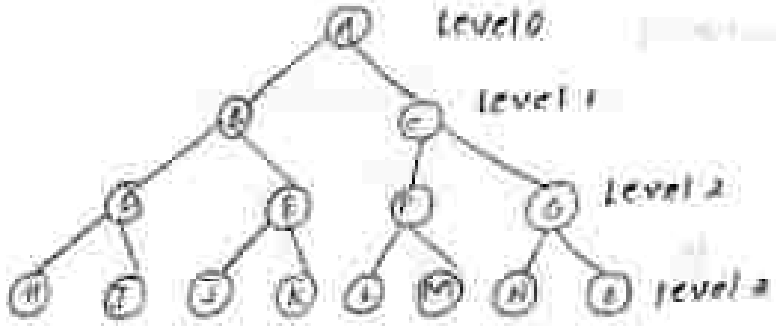
To overcome the drawback of DFS we using the searching strategy called Depth limit search

Ex:

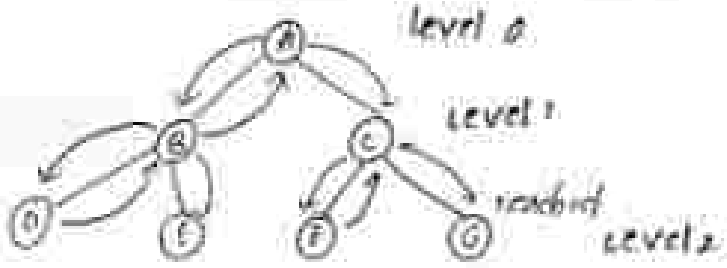


Here our limit - 2

i.e we search for the goal node upto level-2



limit=2

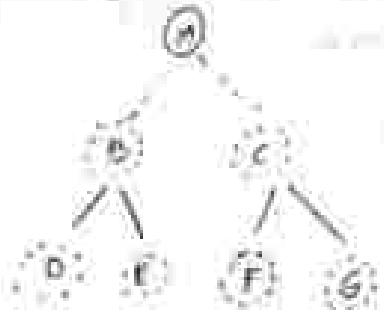


The searching for the goal node by backtracking is shown above

The drawback here is if our goal node is not within the limit then we can't reach goal node

thus we say if in some situations depth limit search is incomplete.

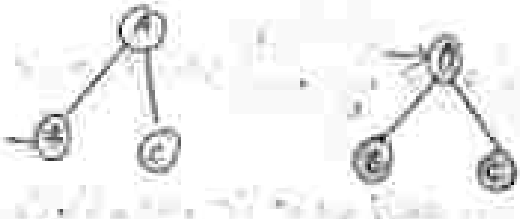
Iterative Deepening DFS:



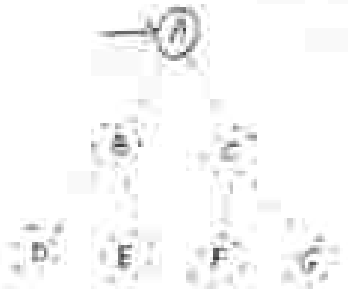
Limit - 0

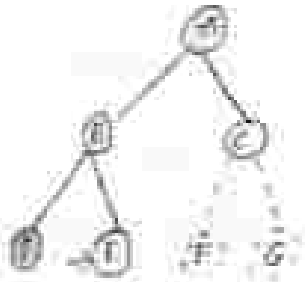
completed

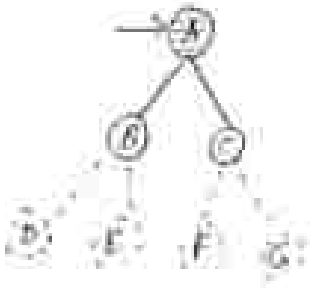
Limit - 1

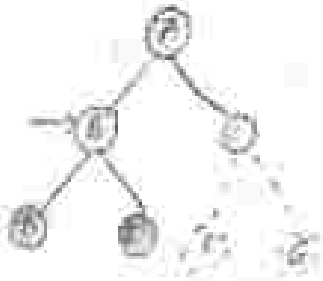


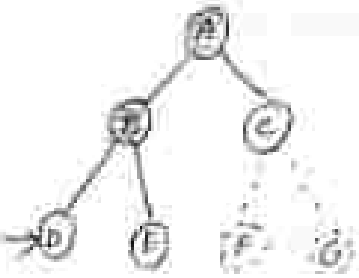
Level-1 :

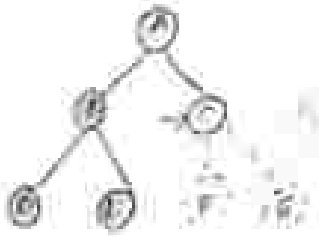


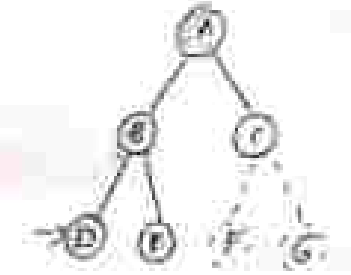


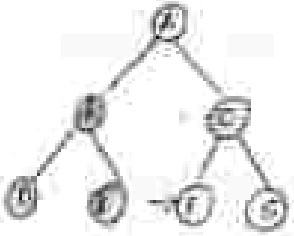


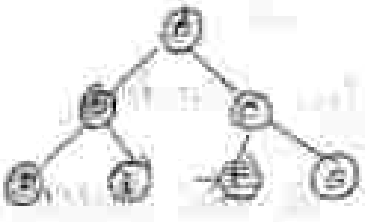


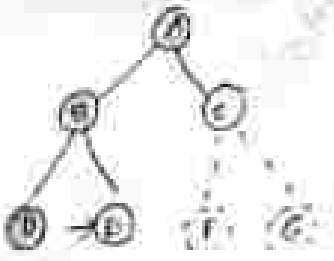


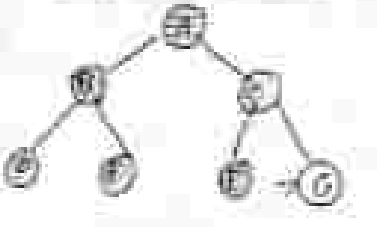


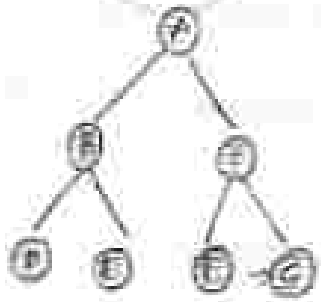


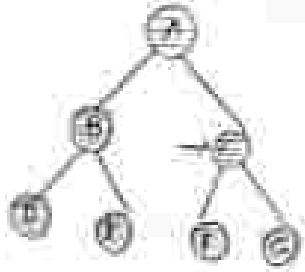


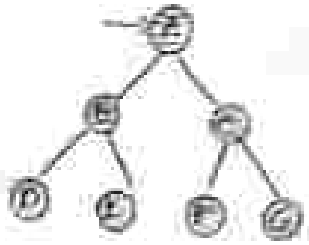


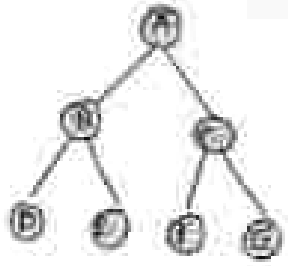








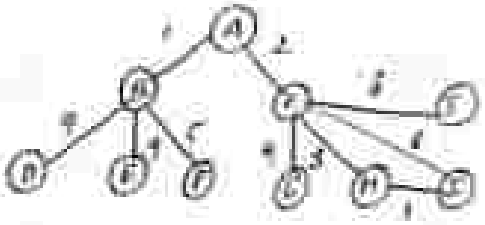




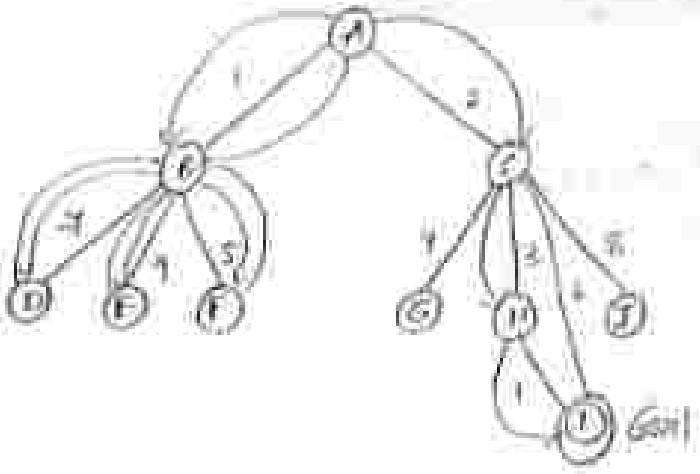
Best-first Search

Using this searching strategy we reach the next node which has best cost i.e low cost

Ex:



Finding goal state from initial state using Best-first searching strategy by backtracking is depicted below



Here the path to reach the goal node is

A - B - F - D - E - C - H - I

path cost = 1+5+3+9+2+3+1 = 26

Whereas intuition or still the utmost minimum cost to reach gold node is

A - C - H - I => 2+3+1 = 6

But by using Best-first search the path cost is too high to reach goal state.

Greedy Best First search

Here we obtain the path cost based on heuristic function.

Heuristic function $f(n) = h(n)$

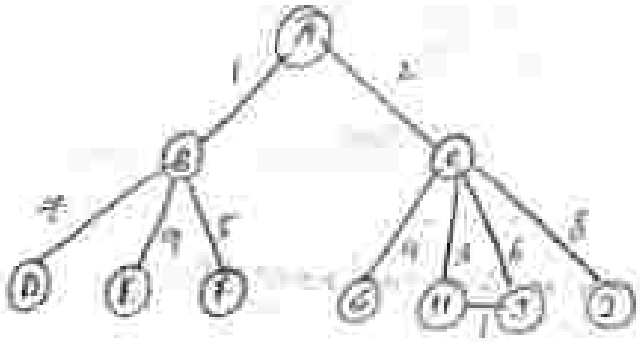
$h(n)$ = estimated cost of cheapest path from any one 'n' to the goal node.

Straight line distance:

Distance b/w center of one node to center of another node as depicted below

h value based on SLD.

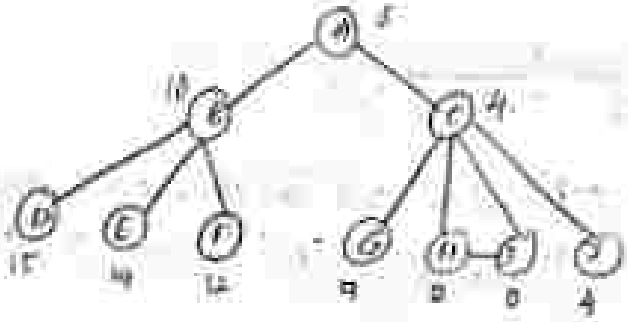
Ex:



and given (h_{SLD}) table were given as

A	8	F	12
B	10	G	3
C	4	H	2
D	15	I	0
E	14	J	4

By using the Greedy First search with the help of backtracking we reach the goal node as shown below



Here the path determined using heuristic function is

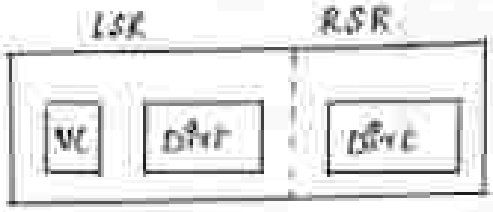
A C I

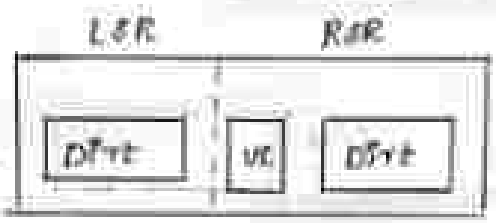
path cost : 2 + 6 = 8

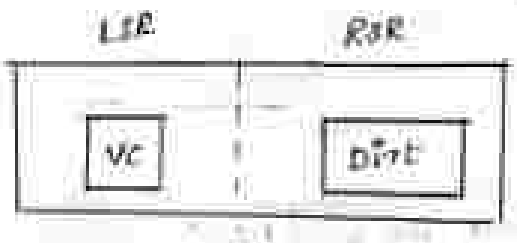
with two exception or still at most min cost to reach goal it
that means it doesnot eliminate the disadvantage of best
first search completely but partially

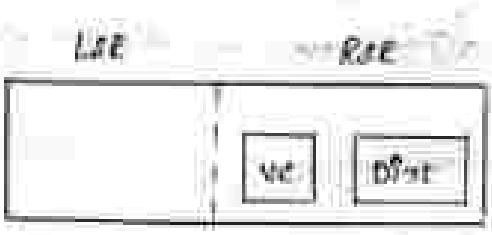
Vaccum cleaner problem

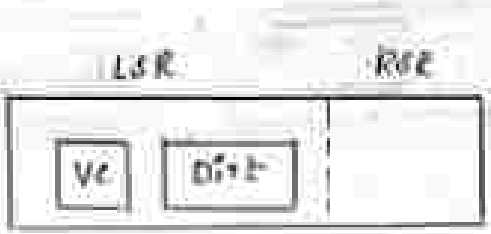
there are 8 states for vaccum cleaner problem

1.


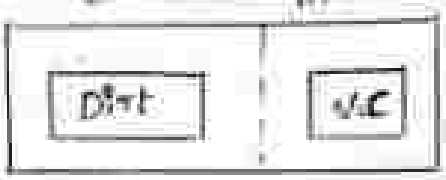
2.


3.


4.


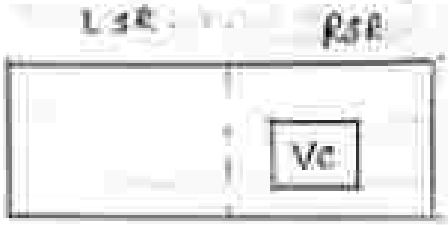
5


6.

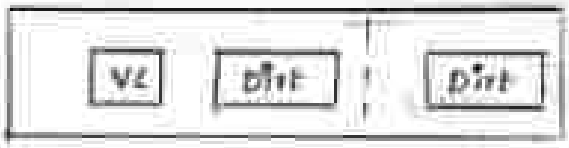


7

8



Here below drawn is the initial state

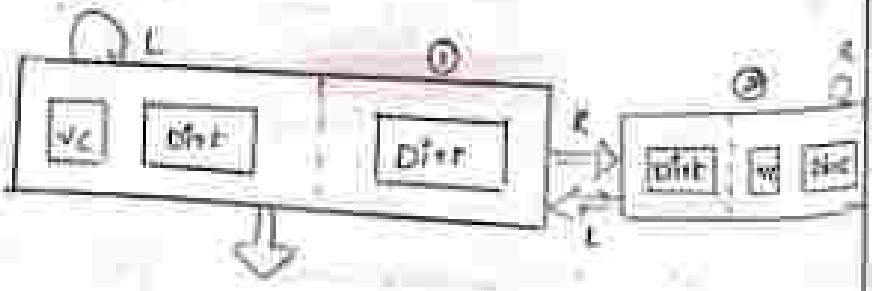


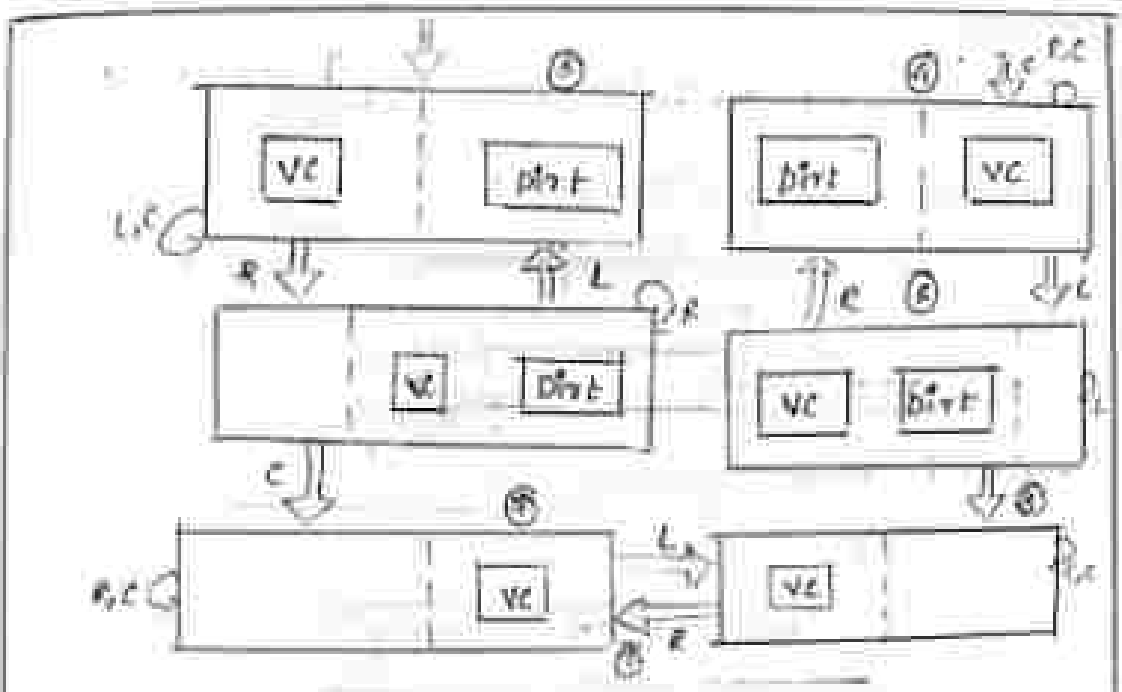
Applying three operations

Move left
Move right
clean

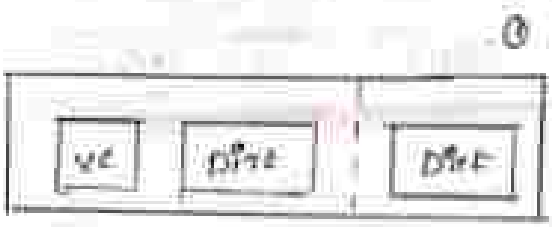
on any state to reach the goal node.

Here the below figure depicts how we achieve goal state from initial state.





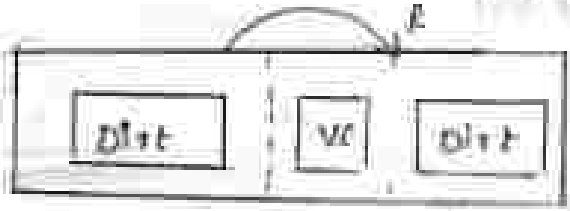
At State 1 which is given below



Move left action

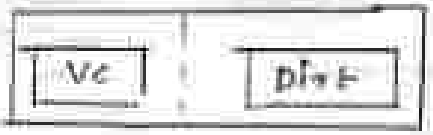
As vacuum cleaner is in left side room on performing move left action there will be no change of state.

Move right



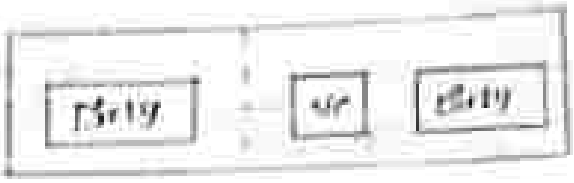
On performing right operation vaccum cleaner will be moved to right side room

clean



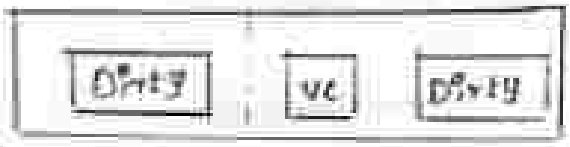
By clean left room clean at vc present in 1st

At state ① which is given below:

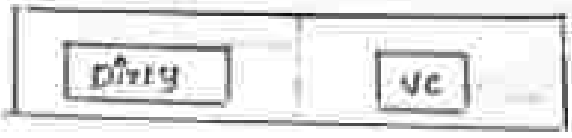


move left action

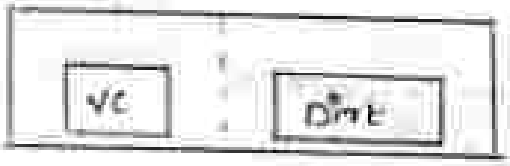
Move right



clean



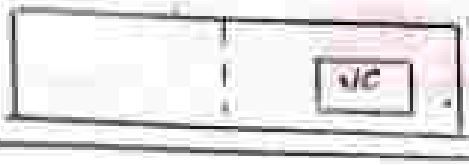
At state ②



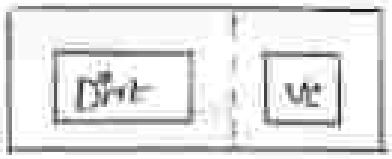
Move left

Move right

clean

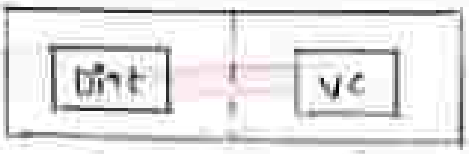


At state (A)

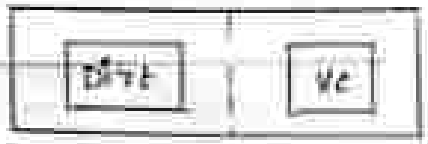


Move left

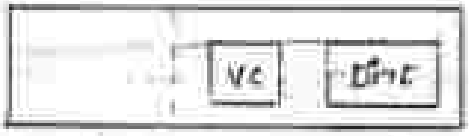
Move right



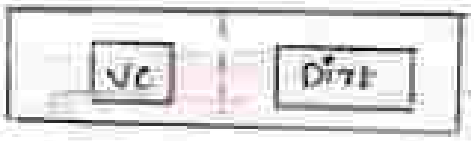
clean



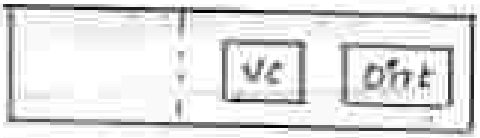
At state (5)



Move left



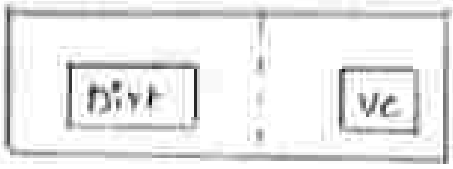
Move right



clean

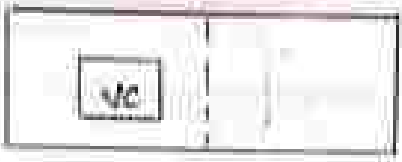
At state (2)

Move right

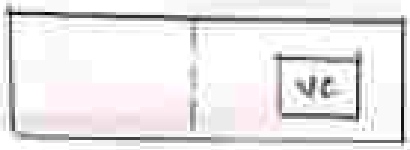


Move right

clean



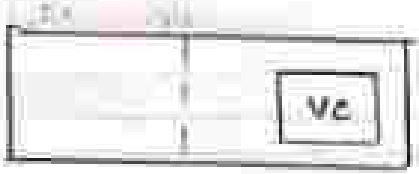
At stage (7)



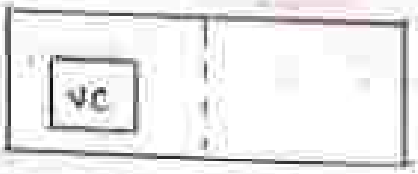
Move left

Move right

clean

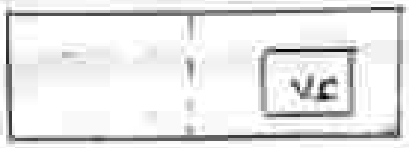


At stage (8)

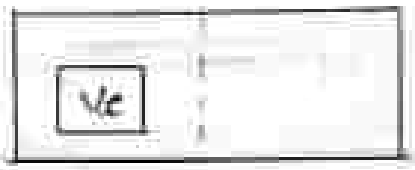


Move left

Move right

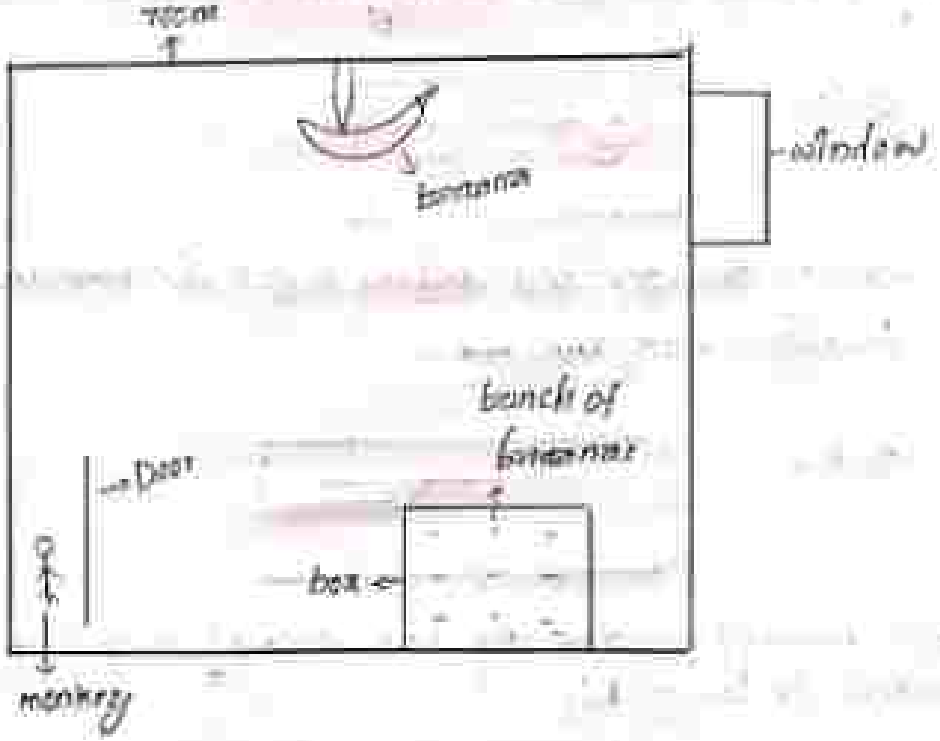


Clean



Monkey - Banana problem

Here the initial state is monkey waiting at the door of room and the goal state is monkey getting the banana as shown below.



Here I'm choosing climb the box and get the bananas

the actions performed to achieve the goal state from initial state are

walk
push
climb
grasp

the followings are the states showed to achieve the goal state are shown below

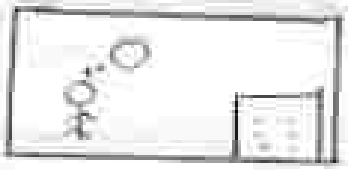
initial state

Here initially monkey is at the door

State 1

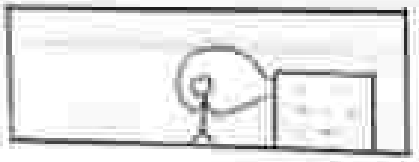
By pushing the door monkey enters into the room

State 2



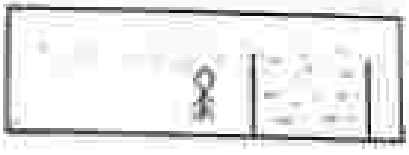
Monkey thought and decided to get the banana beneath/under the box

State 3



Here monkey pushes the box strongly so that the box reverse or somersalt.

State 4



As the monkey pushed the box, the box controlling bananas was reverse-

State-5

Here the monkey climbed the box.

State-6

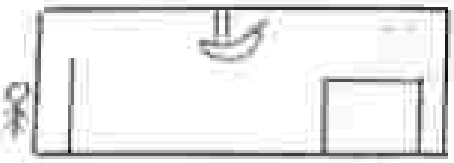
Here the monkey grasped the bananas and in turn it got the bananas.

In other case monkey reaches the bananas by climbing bottom or have to perform-four actions

walk
push
climb
grasp

The states to reach the goal state from the initial state

State-1



Here the monkey walks at the door

State-2

Here the monkey entered the room by pushing the door

State-3

Here the monkey thought and decided to eat the banana hanging to the ceiling of the room.

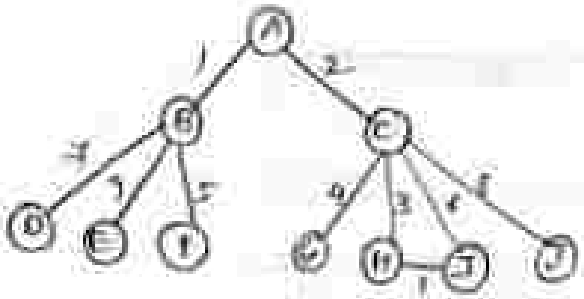
State-4

Monkey climbed the ceiling

State-5

Monkey grasped the banana thus monkey got the banana

A* search



A	8	F	12
B	10	G	7
C	4	H	2
D	15	I	0
E	14	J	4

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

A - $f(n) = g(n) + h(n) = 0 + 6 = 6$

B - $f(n) = g(n) + h(n) = 1 + 10 = 11$

C - $f(n) = g(n) + h(n) = 2 + 4 = 6$

D - $f(n) = g(n) + h(n) = 3 + 15 = 23$

E - $f(n) = g(n) + h(n) = 10 + 14 = 24$

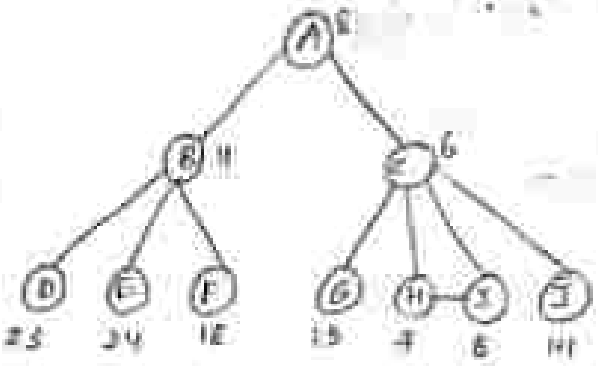
F - $f(n) = g(n) + h(n) = 6 + 12 = 18$

G - $f(n) = g(n) + h(n) = 8 + 7 = 13$

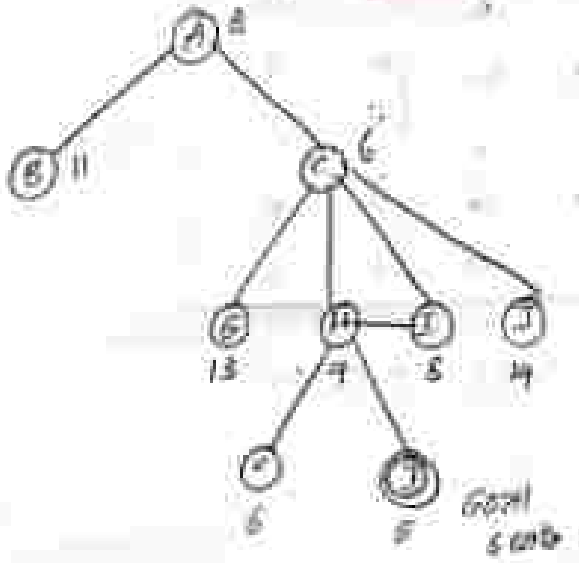
H - $f(n) = g(n) + h(n) = 5 + 2 = 7$

I - $f(n) = g(n) + h(n) = 8 + 0 = 8$

J - $f(n) = g(n) + h(n) = 4 + 10 = 14$



Now by the heuristic function based the

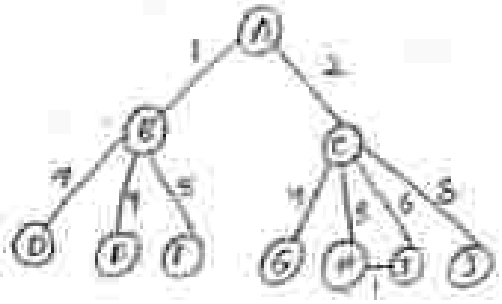


In A* algorithm we reach the next node best / low heuristic value. Here the path to achieve goal node from initial node depends on heuristic function. We calculate the path cost by seeing the given graph.

Here we achieve the goal state from initial state with minimum path cost thus eliminating the drawback of Best-first search and Greedy Best-first search.

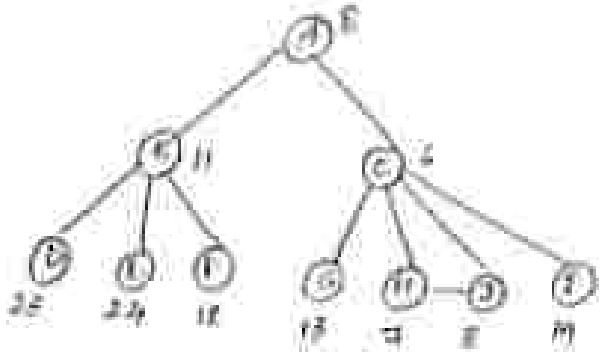
The drawback of A* search is we expand more & more nodes, causes to wastage of memory and time

Memory Bounded A* (MA*) search

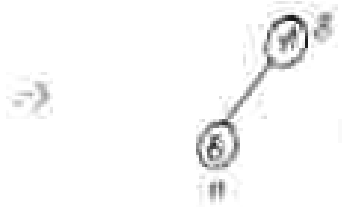


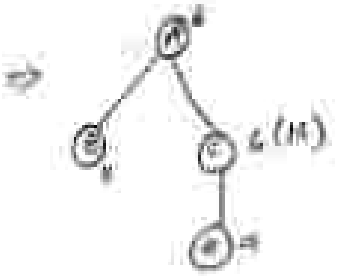
A	8	E	12
B	10	G	7
C	4	H	2
D	15	I	0
E	19	J	4

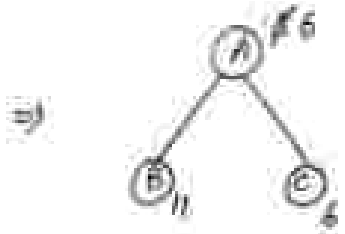
$f(n) = g(n) + h(n)$

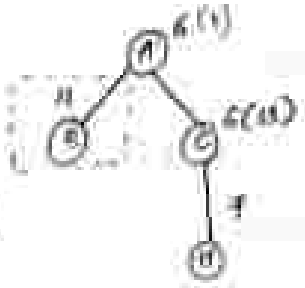


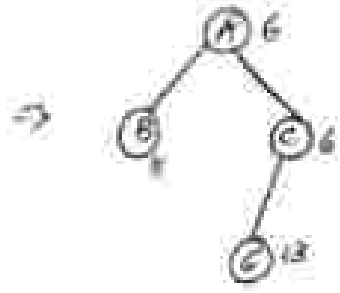
Now by MAX search

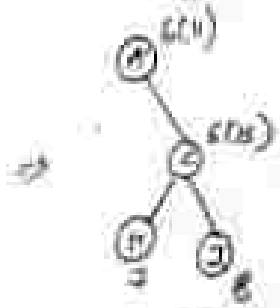


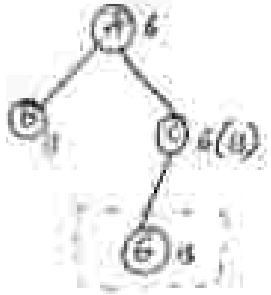


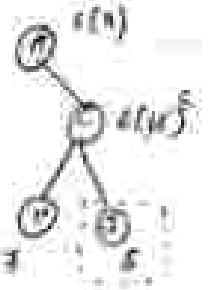


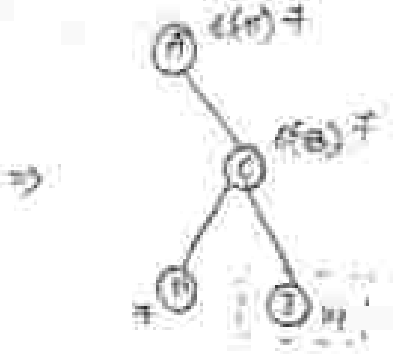


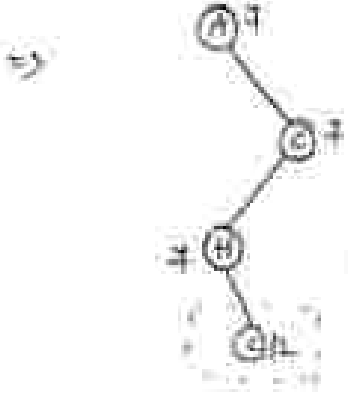


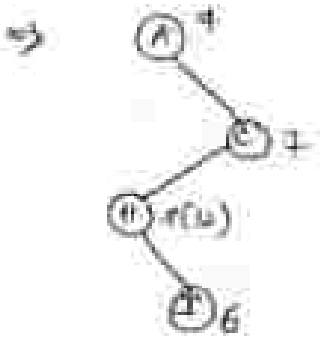


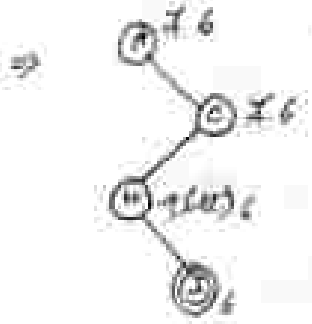












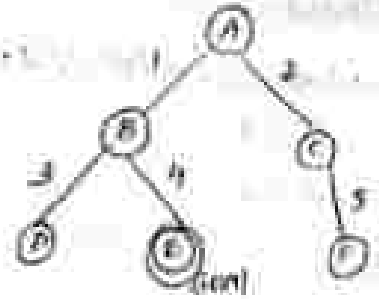
In Memory bounded A* search strategy will eliminate the drawback of A* i.e expanding of more nodes to achieve the goal state in turn occupying longer memory.

In MA* we put the limit on the no. of nodes to be present in the graph while achieving the goal node.

In our given example we set the bound or limit = 4 i.e while searching for the goal node there should be a maximum of four nodes in any state of the graph.

While nodes = 4 in our present graph, if we want to add new node then the condition is we will remove the existing node in the graph which has highest heuristic value of it node.

SMA* (Simplified Memory A*)

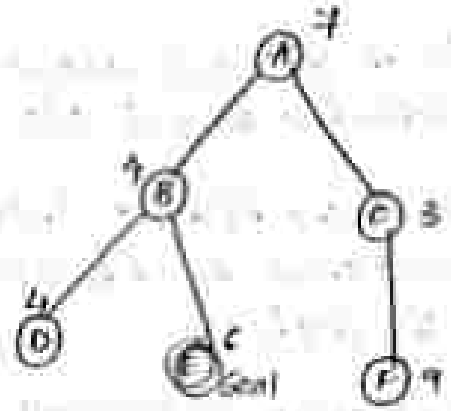


A	7
B	3
C	4
D	6
E	1
F	2

- $f(n) = g(n) + h(n)$
- $f(A) = 0 + 7 = 7$
- $f(B) = 1 + 3 = 4$
- $f(C) = 2 + 1 = 3$
- $f(D) = 4 + 6 = 4$

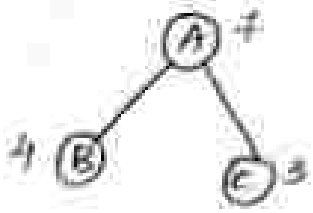
$f(E) = 5+1 = 6$

$f(F) = 7+2 = 9$



f-score graph

By IDA*

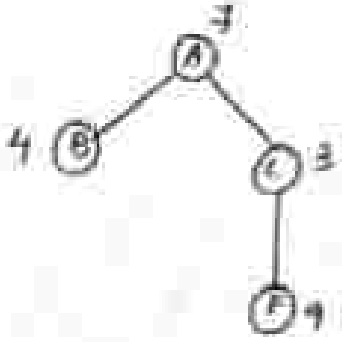


threshold = 3

threshold ≥ f-score of a node

$3 \geq 4$

$3 \geq 3$

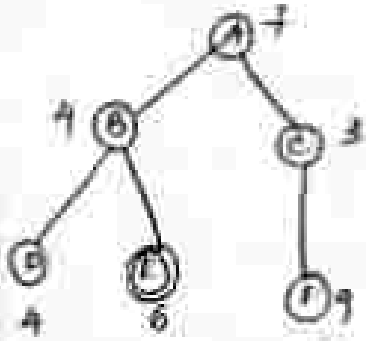


∴ c expanded

the expansion over for c

then go for B

$4 \geq 4$



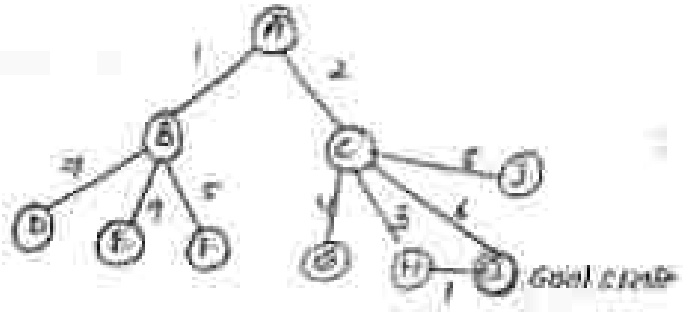
∴ B expanded

In IDA* we will expanded the next node first then we will set the threshold value nothing but minimum f-score of any node in that current state of graph

the condition to expand any node is

Threshold ≥ f-score of any node.

Recursive Best First Search (RBFS):

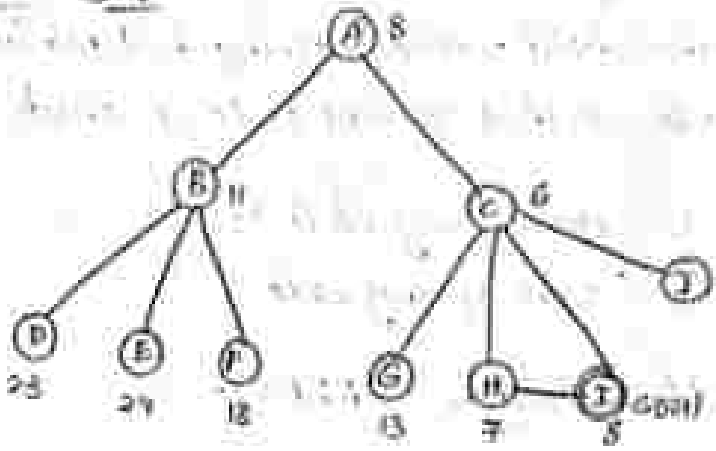


A	8	F	12
B	10	G	7
C	4	H	2
D	15	I	0
E	14	J	4

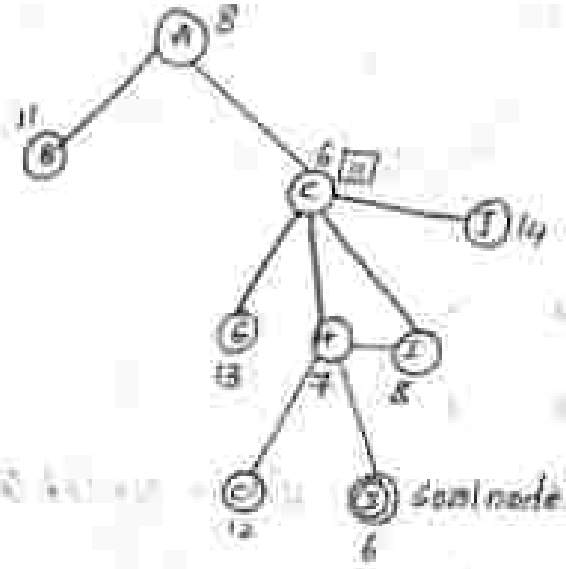
$f(n) = g(n) + h(n)$

- $f(A) = 0 + 8 = 8$
- $f(B) = 1 + 10 = 11$
- $f(C) = 2 + 4 = 6$
- $f(D) = 8 + 15 = 23$
- $f(E) = 10 + 14 = 24$
- $f(F) = 6 + 12 = 18$
- $f(G) = 6 + 7 = 13$
- $f(H) = 5 + 2 = 7$
- $f(I) = 6 + 0 = 6$
- $f(J) = 10 + 4 = 14$

-f-score graph



By RBFS



In RBFS strategy we will look for the alternative path. The heuristic value of the alternative path is written on the node expanded enclosed in a box. The childs of that node which was expanded must satisfie the condition

f-score value ≤ f-score value of alternative path

Problem solving

Area for finding answers to unknown situations. There are four steps for problem solving as follows:

1. Understanding
2. Representation
3. Formulation
4. Solving

1. Understanding:

Analyzing initial and final states

2. Representation:

It is nothing but writing all the possible states which we come across while solving the problem diagrammatically or in the form of Images and integrating the initial and final states.

3. Formulation:

The methods or actions used to solve the problem.

4. solving:

Implementation of actions/methods on different states on the graph. We solve the given problem.

Ex: Vacuum cleaner

For problem solving we need 4 steps

1. understanding:

Initial state:

One house there are two rooms where both are dirty and vaccum cleaner is present either of one room.

Goal state:

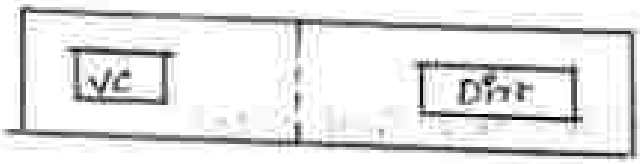
Both rooms must be clean and vaccum cleaner is present

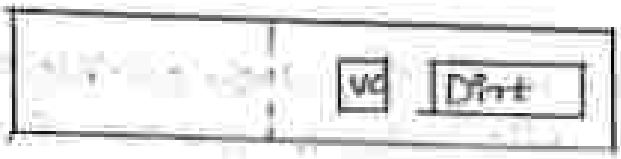
in either of the two rooms.

2. Representation:

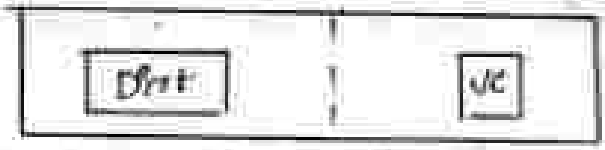
1. 

2. 

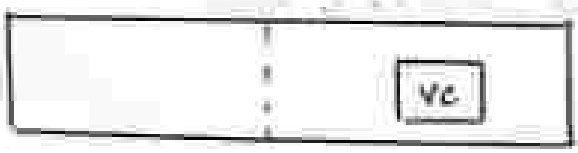
3. 

4. 

5. 

6. 

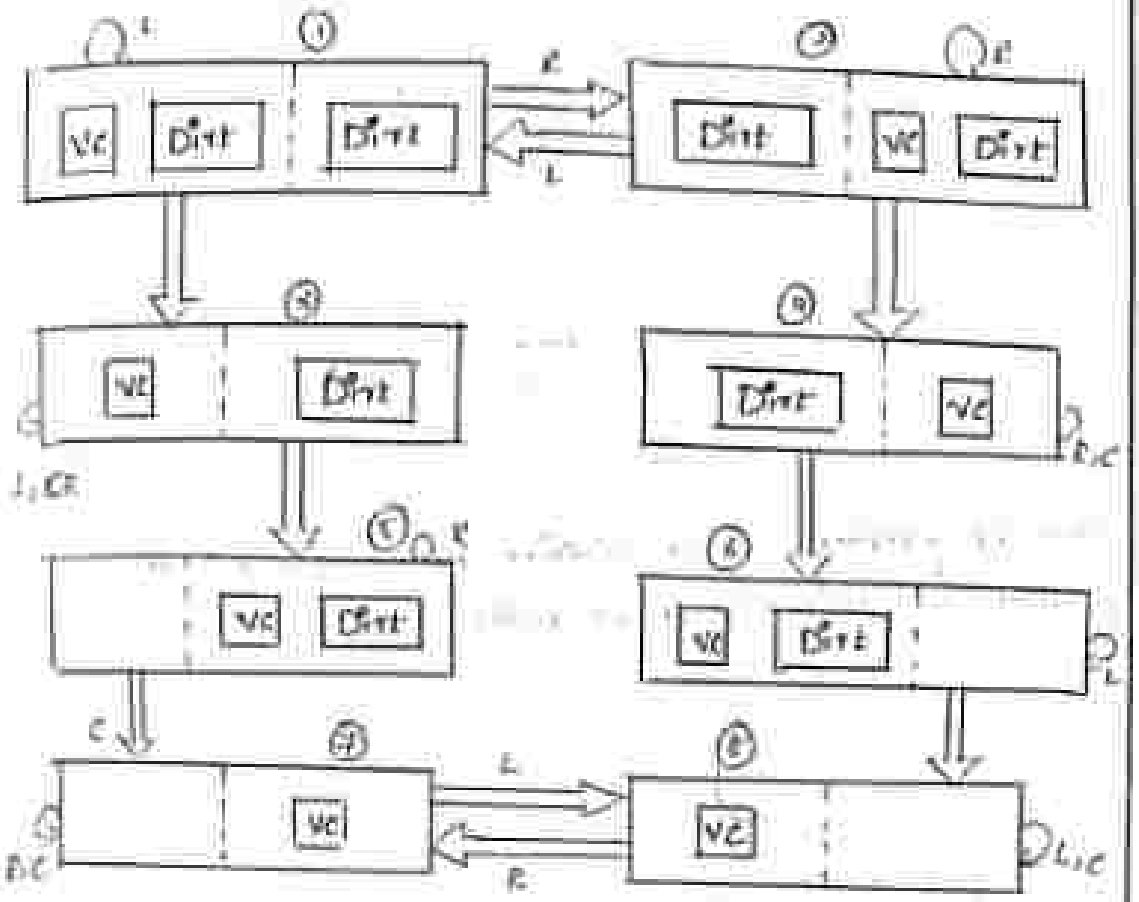
7. 

8. 

3. Formulation:

Here we use three methods to access the
MoveLeft, Move Right, clean

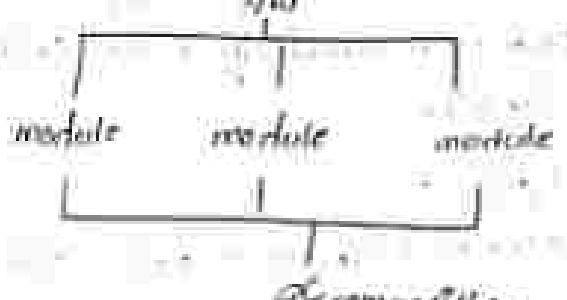
4. Solving:



Problem Reduction

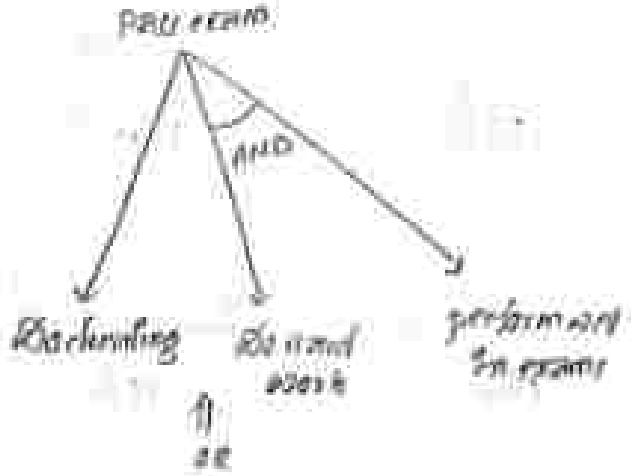
The process of decomposing a complex problem or task into a set of sub problems, solving them and then integrating all the sub problem to get the soln of the problem.

Ex: Developing the s/w for our problem we first decompose into modules, develop them and combine all the modules to get the required s/w as shown below

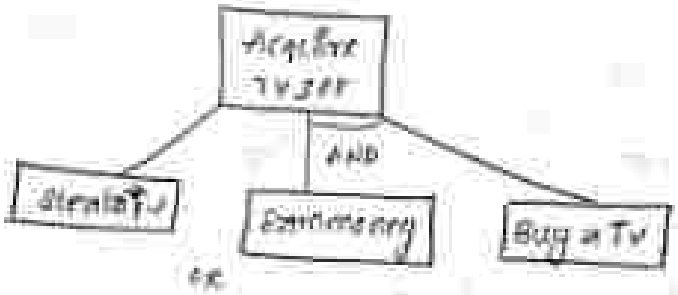


Decomposition

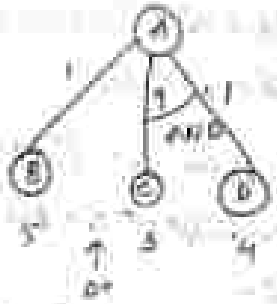
Here the problem given is to pass the exam for that the problem reduction is done as shown below:



Here the problem is to acquire TV set for that the problem reduction is shown below



for some graph as shown below



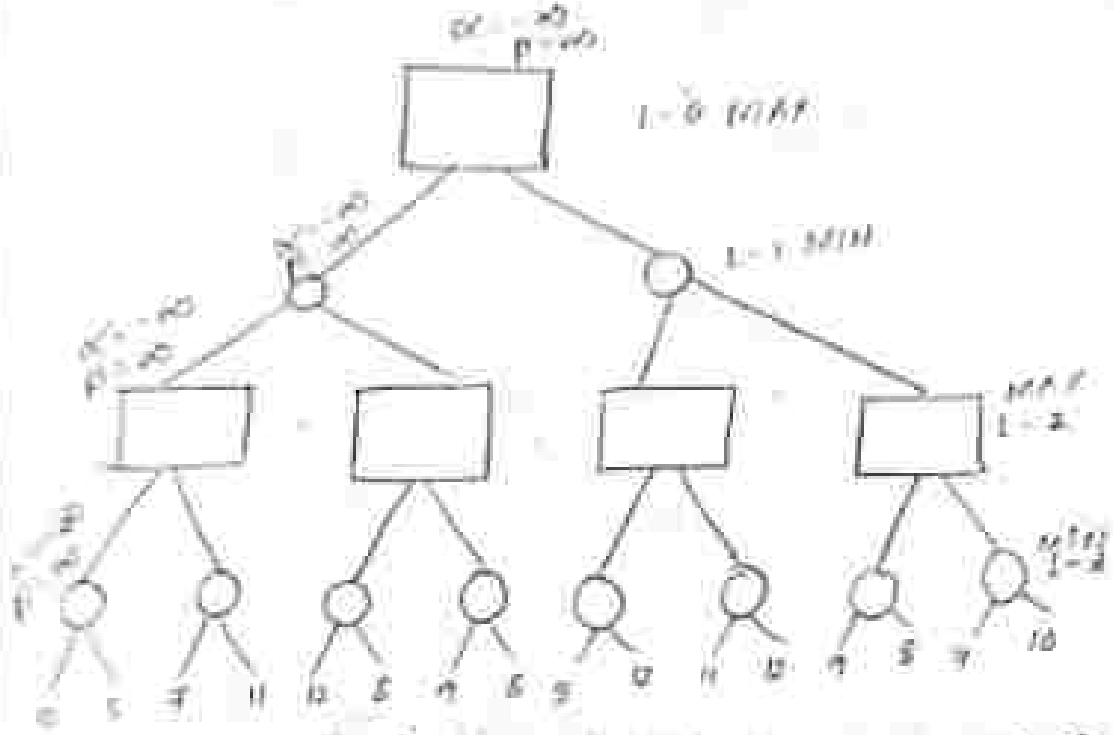
Here to obtain the goal state we can expand B individually or we can expand both C & D combinedly based on the best value of heuristic function.

$B : f(n) = 6$

$C,D : f(m)(C) + f(n)(D) = 12 + 5 = 17$

So, heuristic value of B i.e. B thus we expand B

Alpha beta pruning

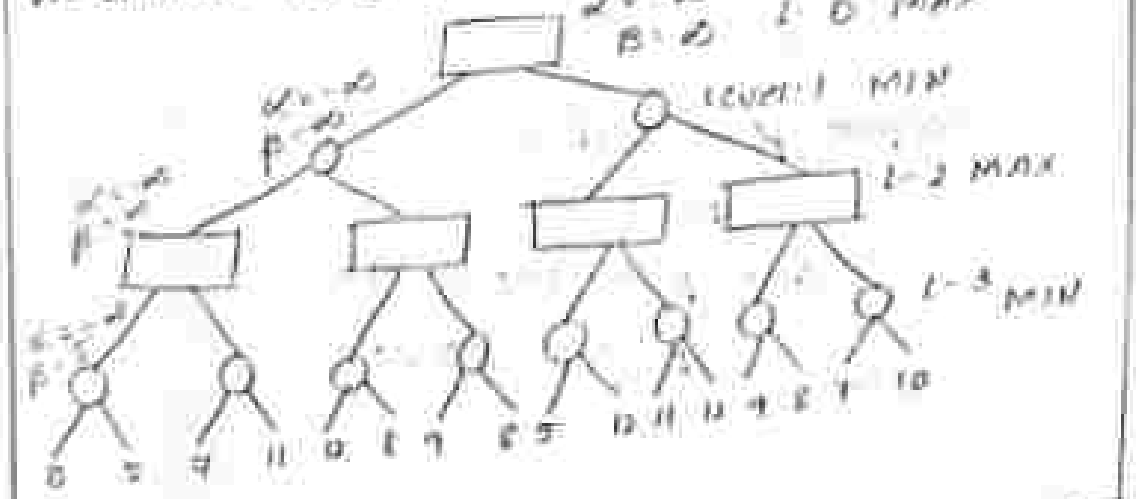


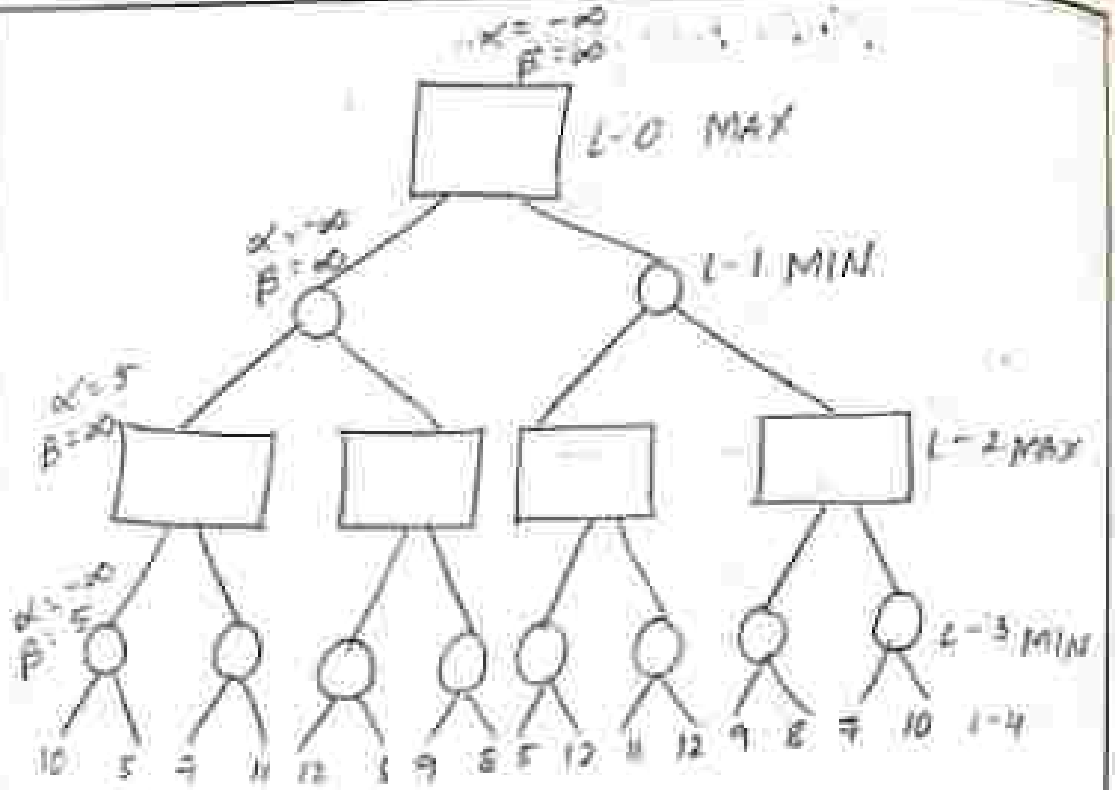
In alpha beta pruning there are two terms α and β α means maximum α - max, β is minimum i.e β - min

Initially we take $\alpha = -\infty$, $\beta = \infty$

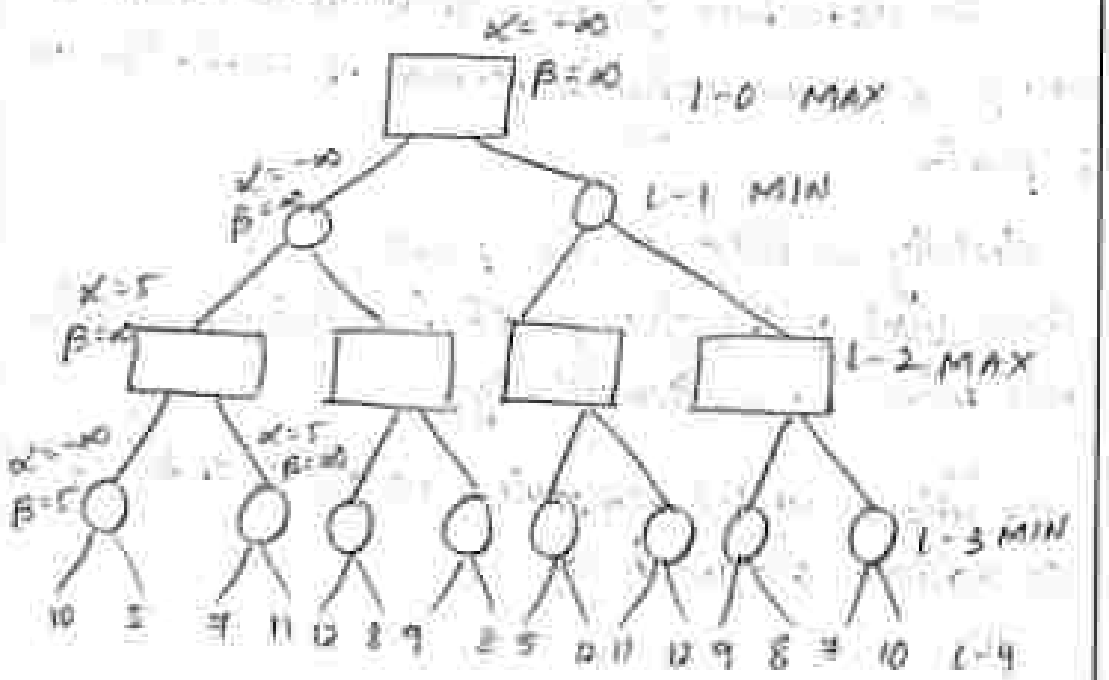
Every level of the given graph have value max, min, max, min, ... alternatively

From above at L-3 comparing node with their childs we achieve the graph as





Next we have to compare the nigh child

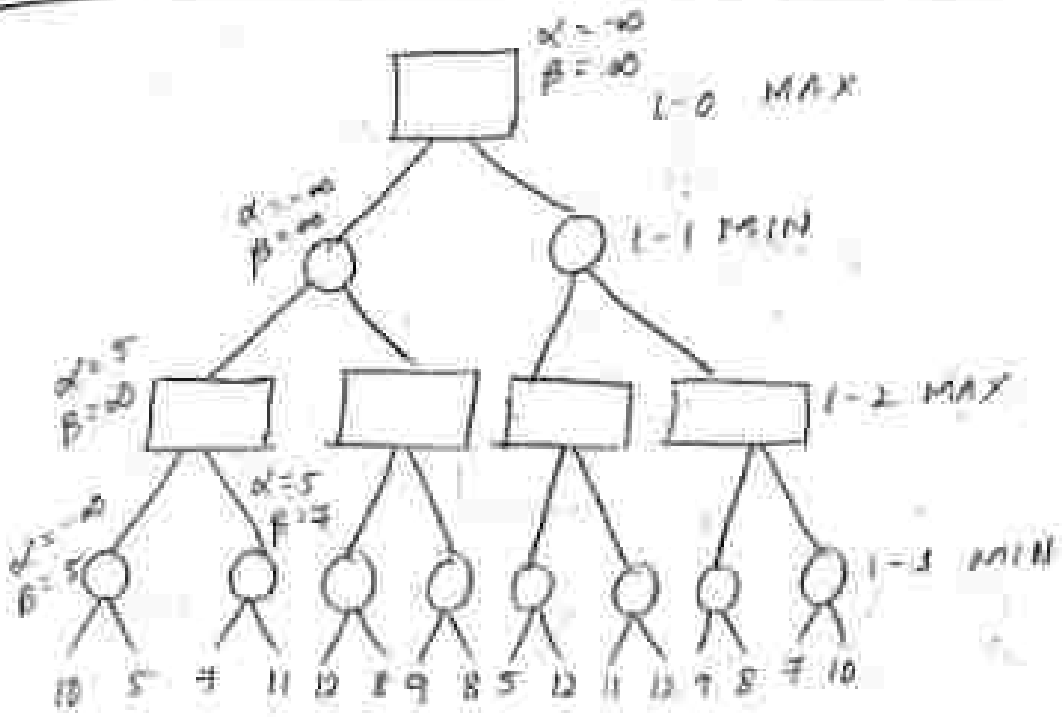


compare $\beta = \infty$ with 7

$\beta = 7$

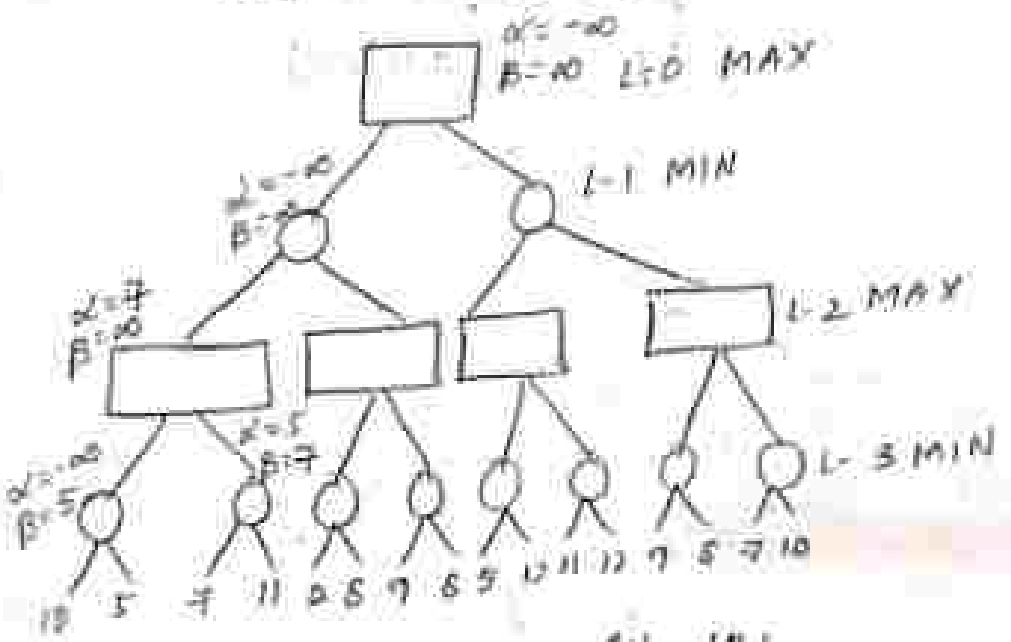
then $\beta = 9$ compare 11

$\beta = 7$ ($\because 7 < 11$)



Now $\alpha = 5$ compare with right child 5, 7

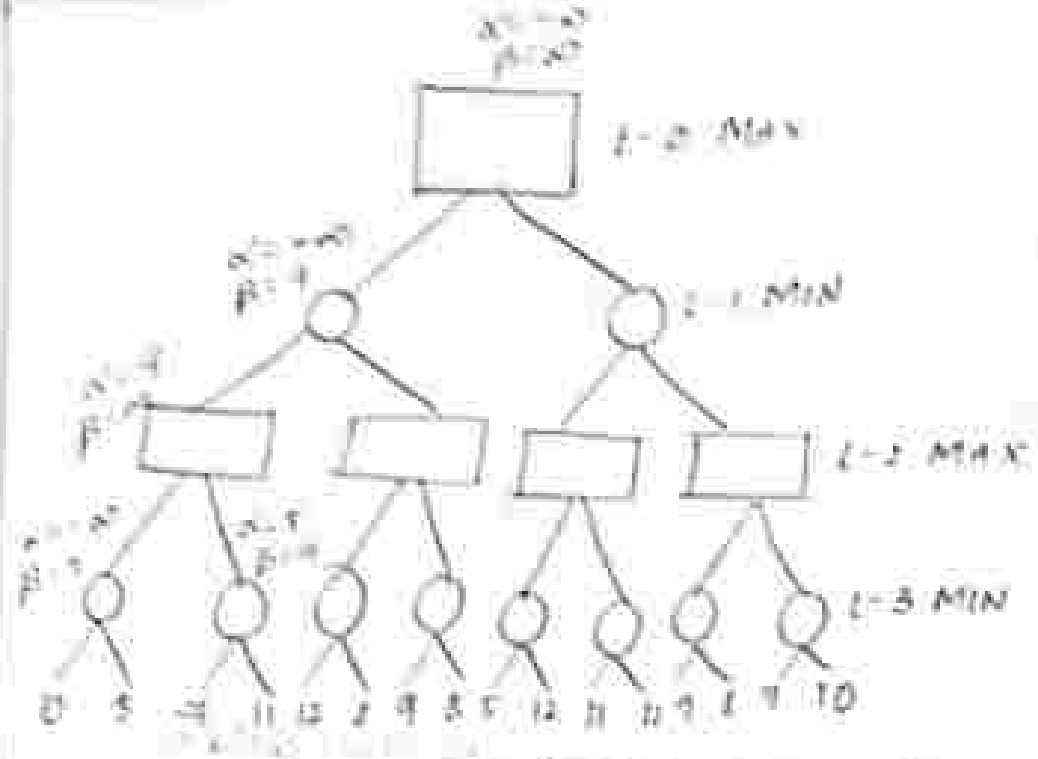
$\alpha = 7$ ($\because 5 < 7$)



Now compare the above with child

$\beta = \infty$ compare with $\infty, 7$

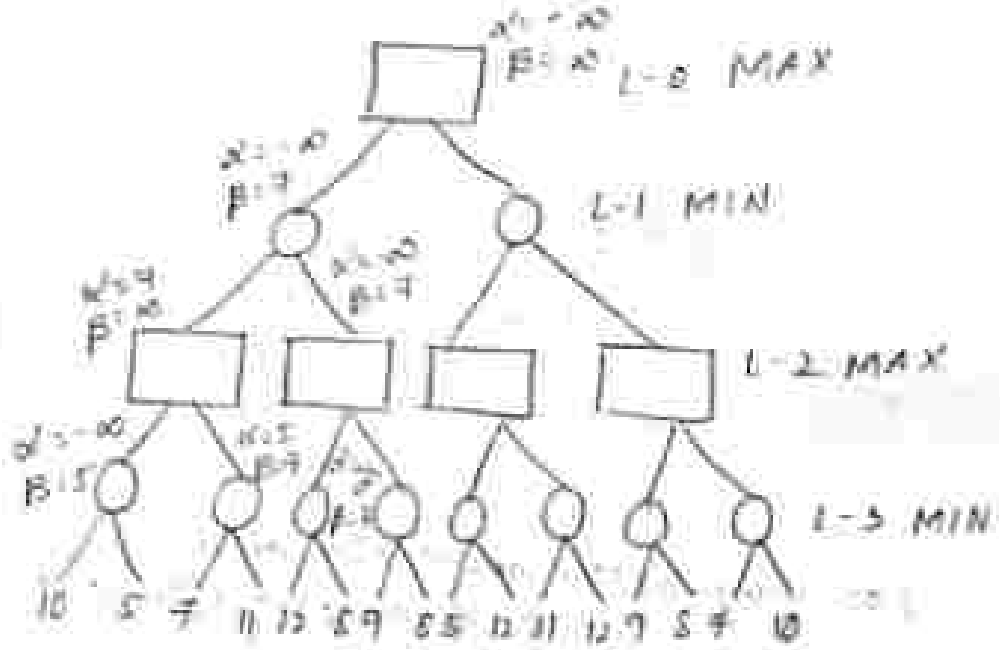
$\beta = 7$



compare the right child now

$\beta = 7$ compare 8, 12

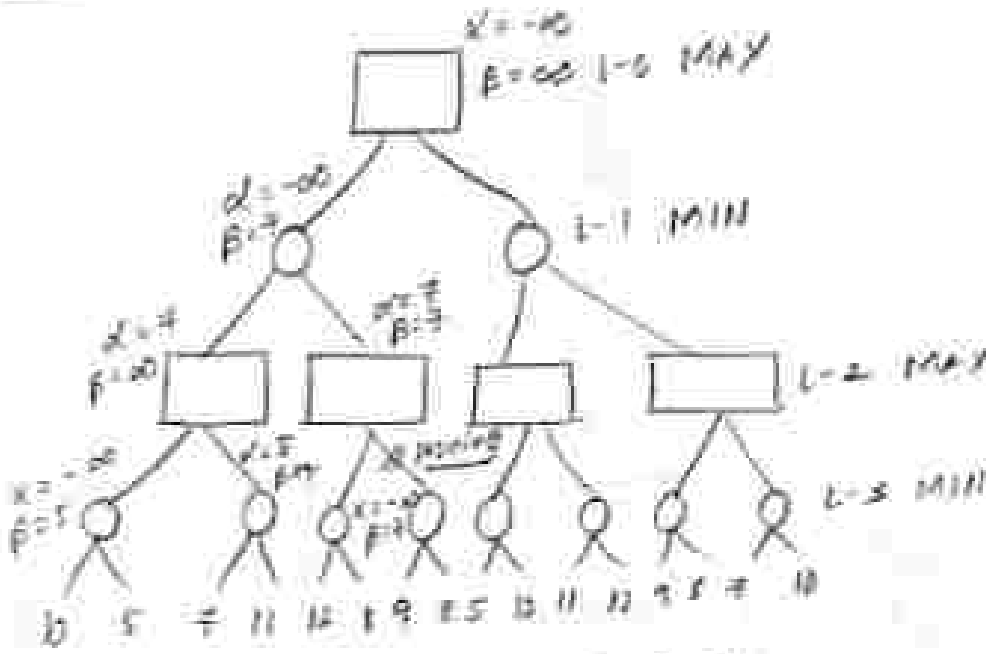
$\therefore \beta = 7 \quad (8, 12 > 7)$



go to top now

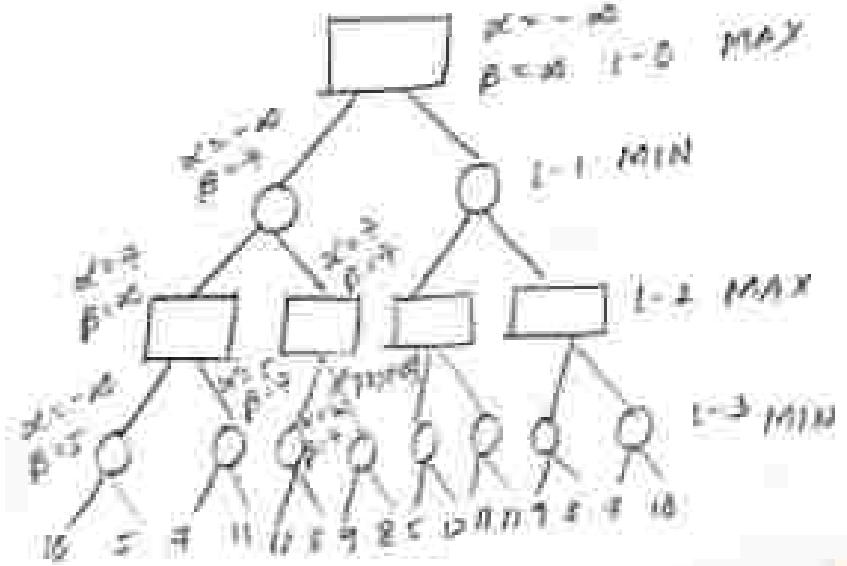
$\alpha = -\infty$ compare $-\infty, 7$

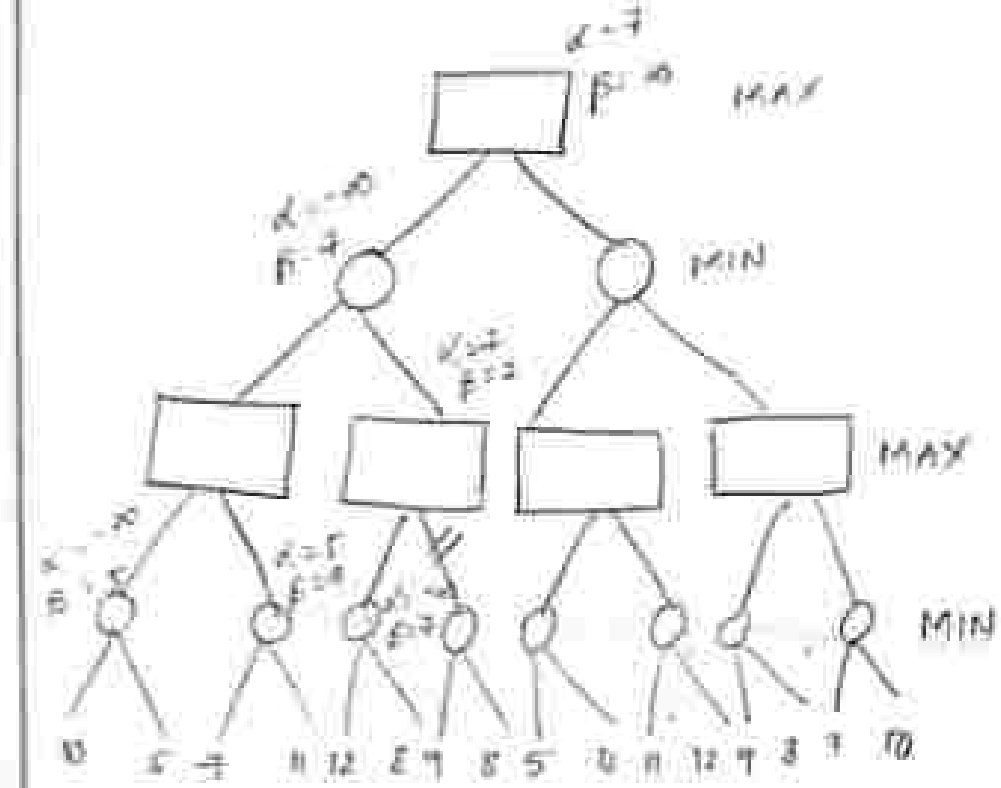
$\alpha = 7 \quad (-\infty < 7)$

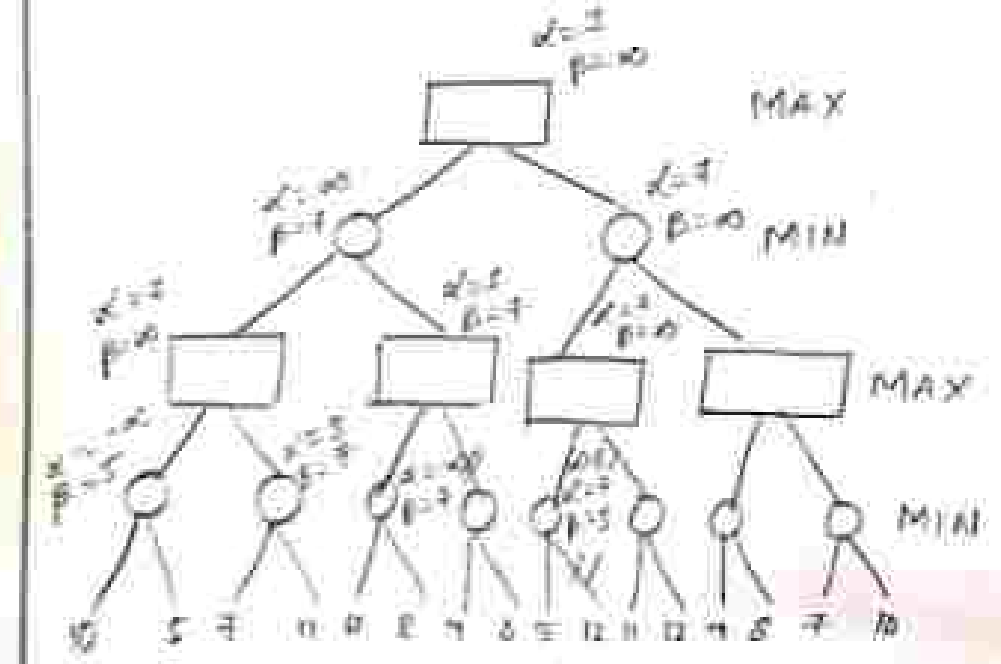


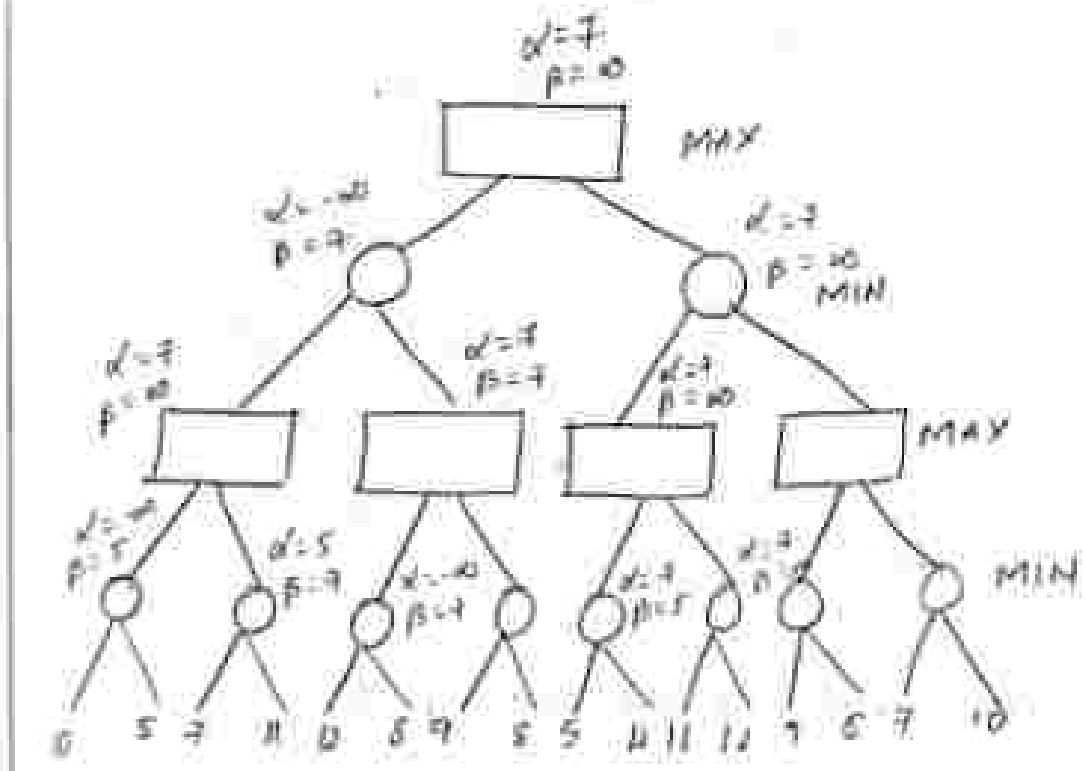
Now compare $\beta = 5$ with $\alpha = 7$ & $\beta = 7$

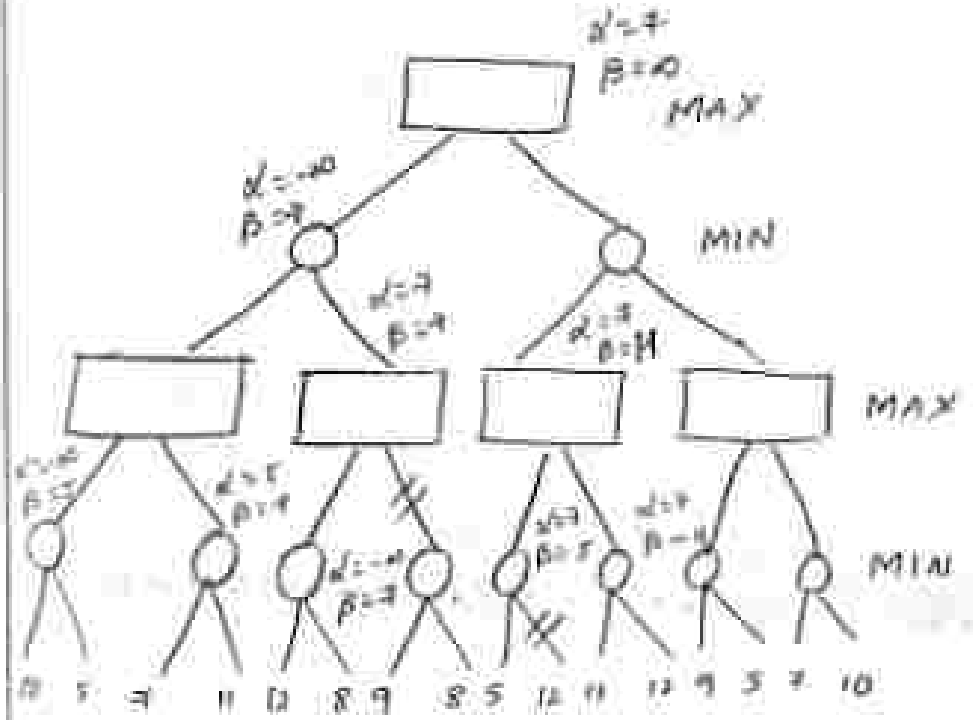
$\therefore \beta = 7$

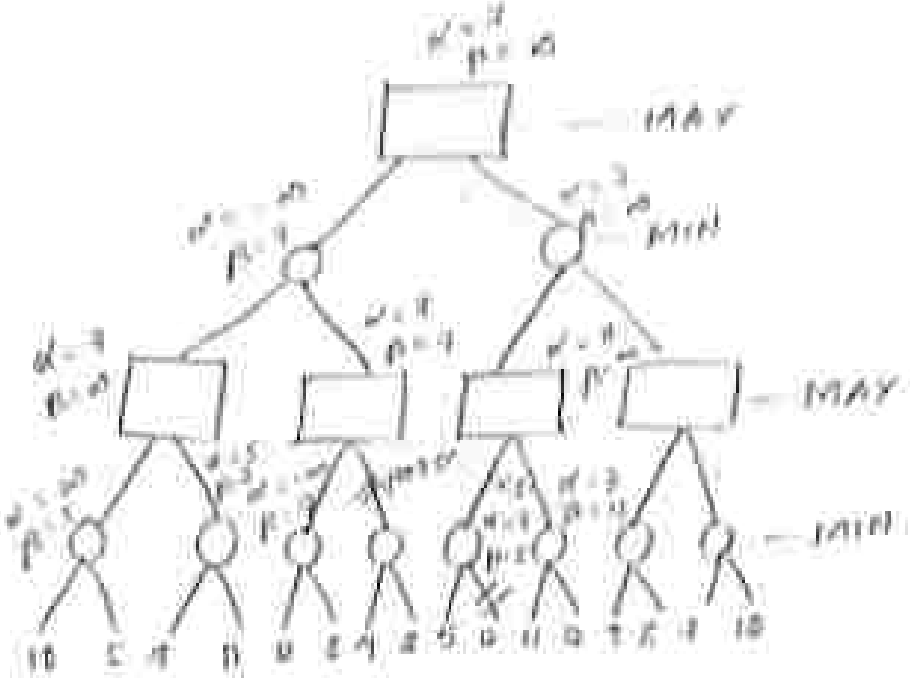



MAX
MIN
MAX
MIN

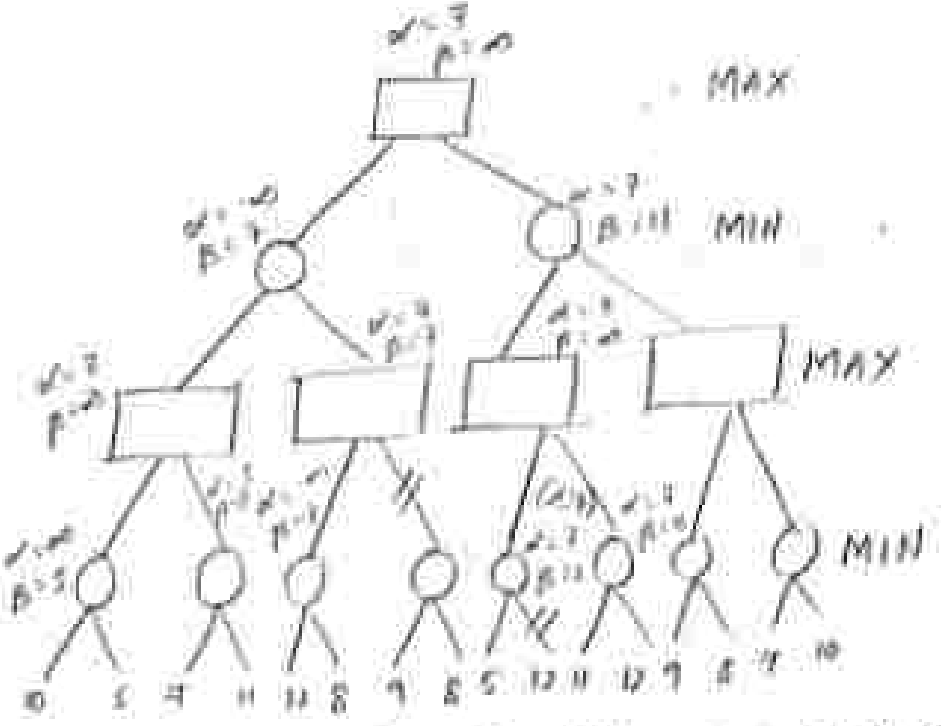

MAX
MIN
MAX
MIN


MAX
MIN
MAX
MIN

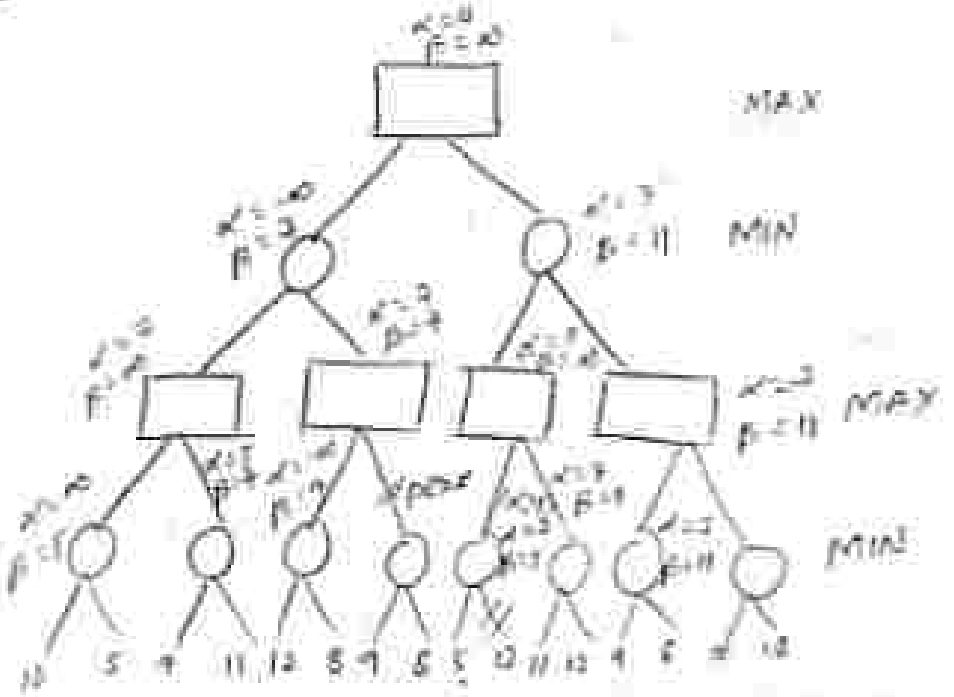

MAX
MIN
MAX
MIN

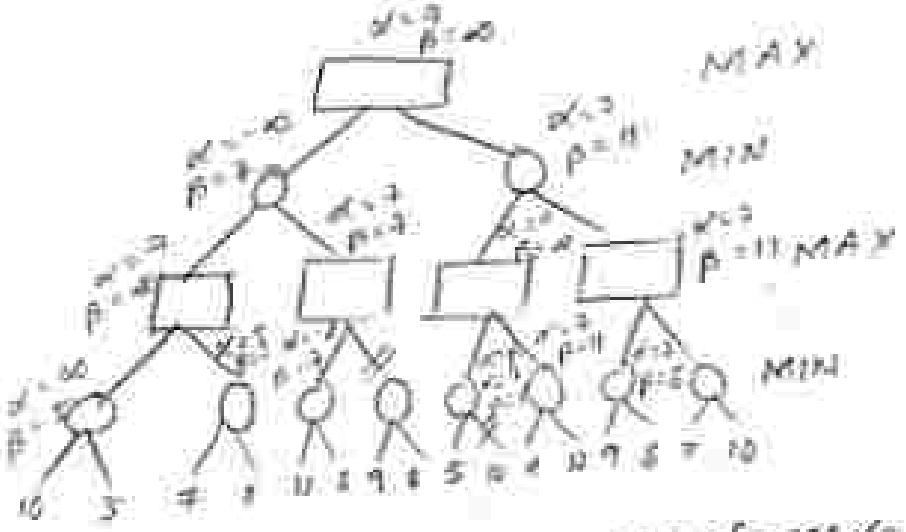


Now compare β with $11, \infty \therefore \beta = 11$

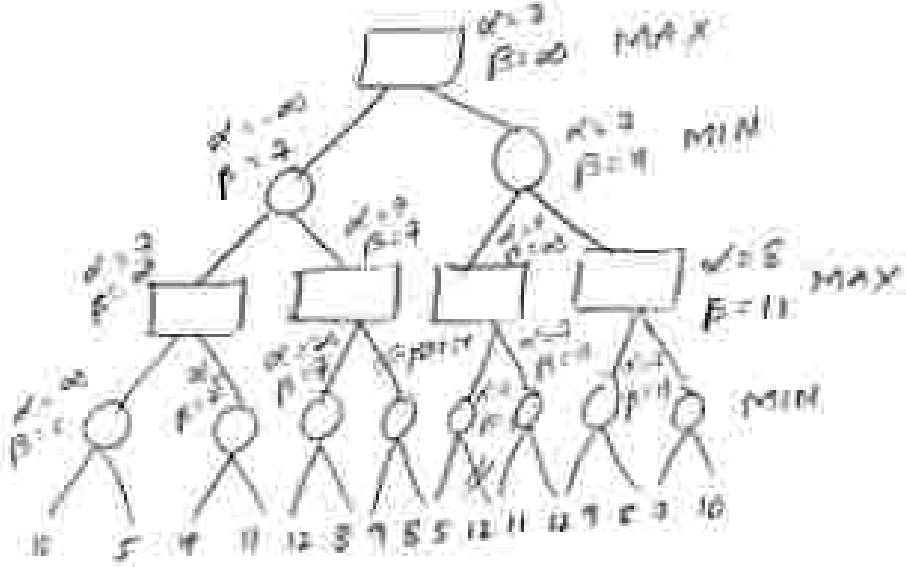


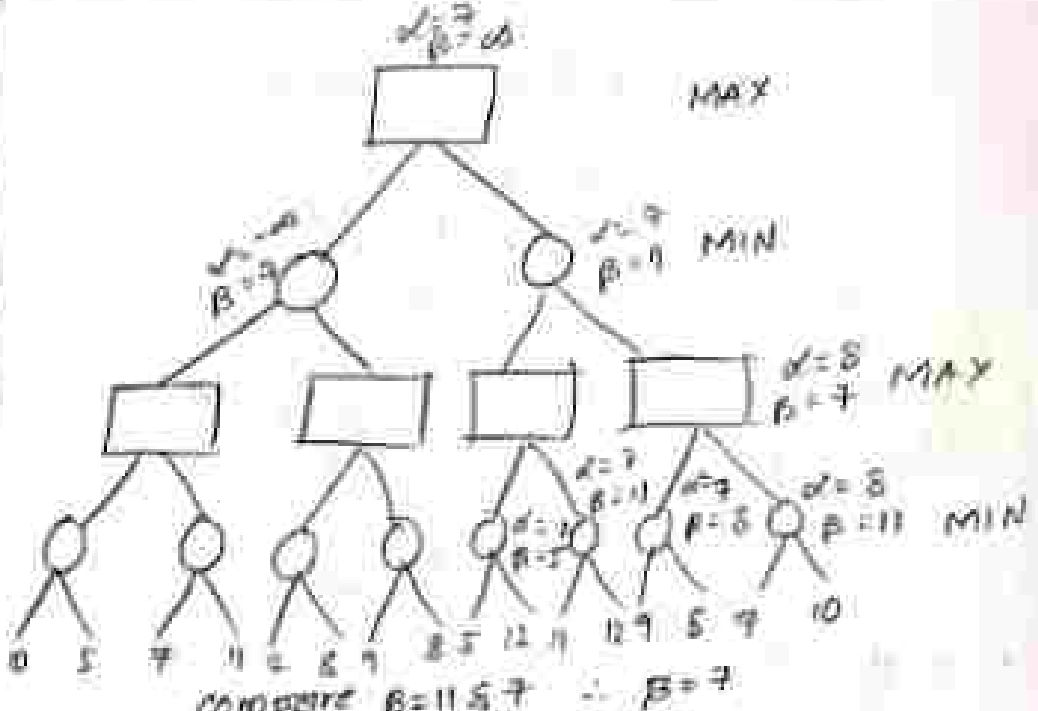
Now we update the child values of left side



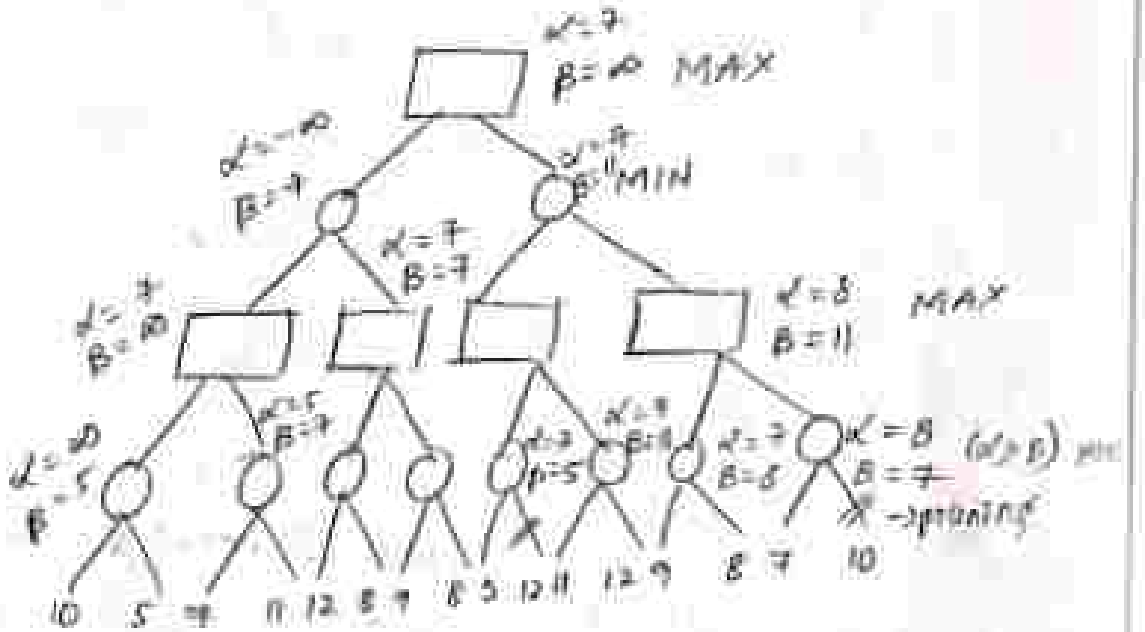


Now change $\alpha <= 5$





compare $\beta = 11$ & 7 $\therefore \beta = 7$



Game playing

Game playing can be defined by the following four steps

1. Initial state
2. Successor function
3. Goal state
4. path cost

Artificial intelligence has continued to improve with

Aim - that we unable to tell the difference b/w computer and human being

A game must-fill natural if it follows the following

1. Obey laws of the game
2. character names
3. path-finding
4. Decision making
5. planning

Different types of Computer Games

→ Strategy game Ex: chess, ludo
It is a turn based game

Roll player game: single and multi player game
Ex: tennis, Starcraft

Action game: Ex: DOOM, zombie

Sports game: basket ball, tennis, cricket, etc

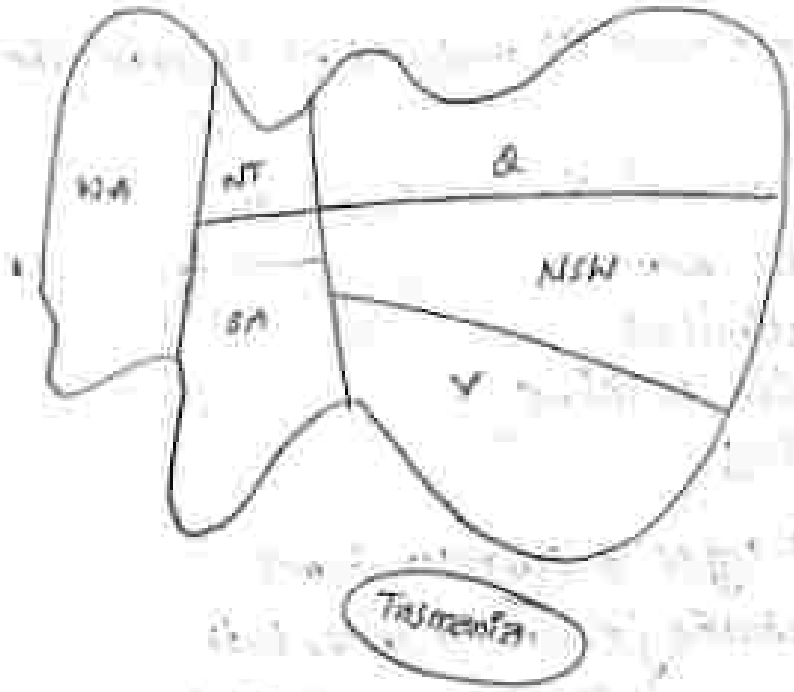
Simulation: pubg, free fire, call of duties

Adventure game: subway surfs, temple run

puzzle game: tick-tac-toe, brain bloom, sudoku

Constraint Satisfaction

WA - Western Australia
NT - Northern Territory
T - Transelvania
Q - Queens land
NSW - New south wales
SA - South Australia
V - Victoria



Problem:

We need to color each region with R, G, B such a way that no neighbouring reasons have same color

Solution:

Here the constraint is that no neighbouring reasons have same color.

Variables are: WA, NT, SA, Q, NSW, V, Tasmania

Domain: {R, G, B}

Here we provide the soln by satisfying constraints as shown below in the table

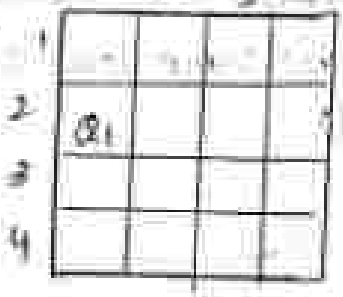
Domain	WA	NT	SA	Q	NSW	V	T
R, G, B	B	R	G	B	R	B	R V B V G

4-Queen problem

Q. The 4-Queens problem consists of placing 4-Queens on 4x4 chess board so that no two queens can capture each other i.e no two queens are allowed to be placed on the same row, same column and same diagonal.)

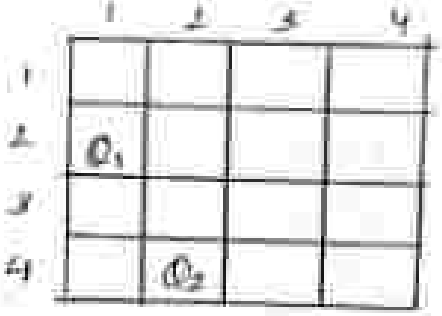
Sol:

Step-1:



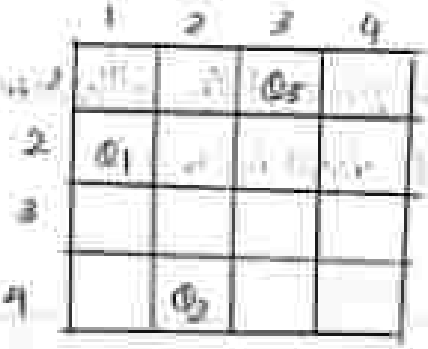
Here we place the Queen in 1st colmn & 2nd row

Step-2:



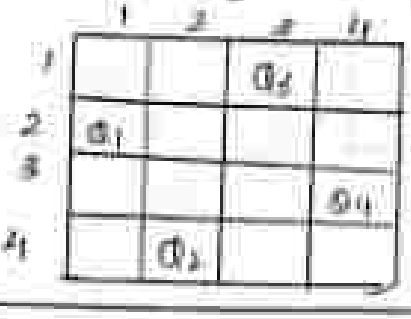
we place Queen in 2nd colmn and 4th row

Step-3



we place the Queen Q_3 in 3rd column & first row

Step-4



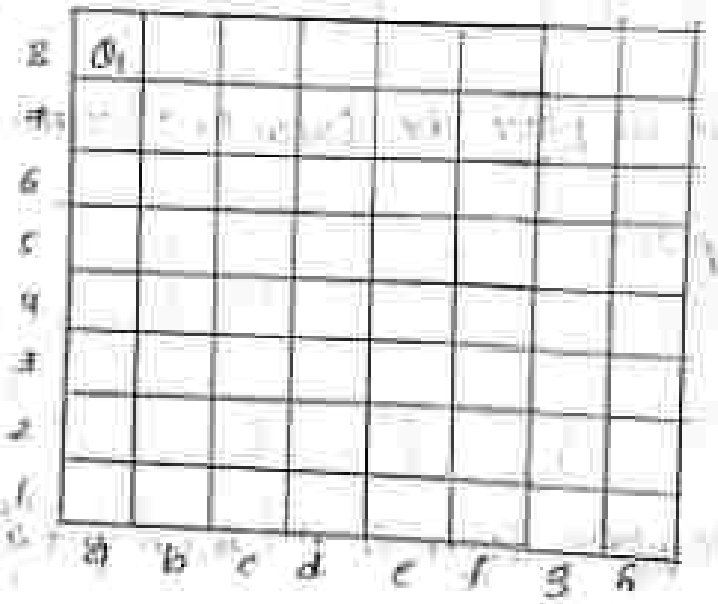
we place the Queen Q_1 in 1^{th} column, 8^{nd} row

8 Queens problem

Problem: The 8-Queens problem is the problem of placing 8 Queens on 8x8 chess board such that none of them attack one another (i.e no two queens are in the same row, same column and diagonal).

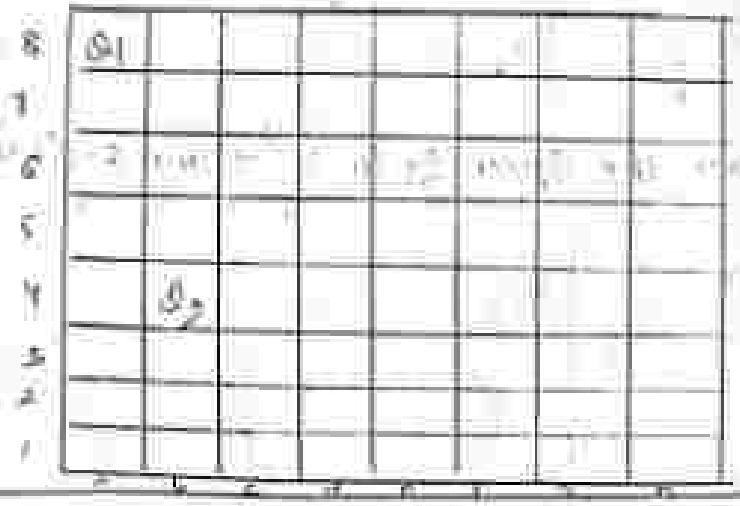
Sol:

Step-1:



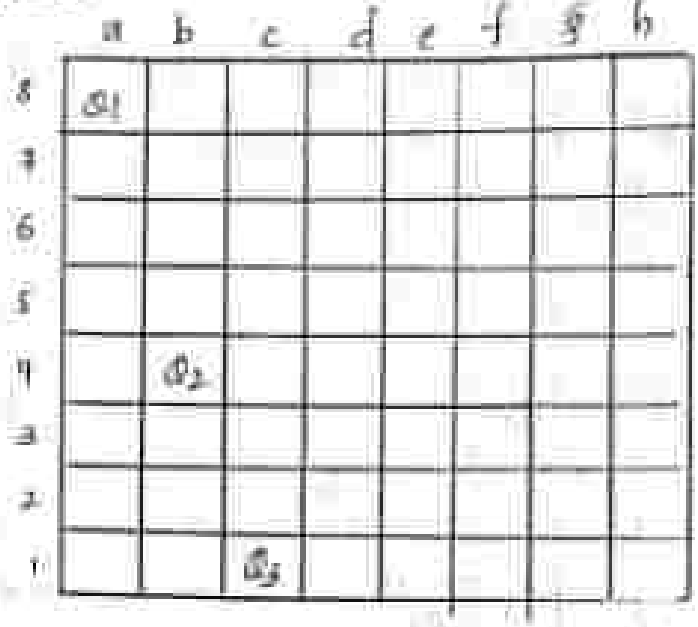
Here first Queen is placed in 1^{th} column and 8^{th} row in 8x8 Grid chess board

Step-2:



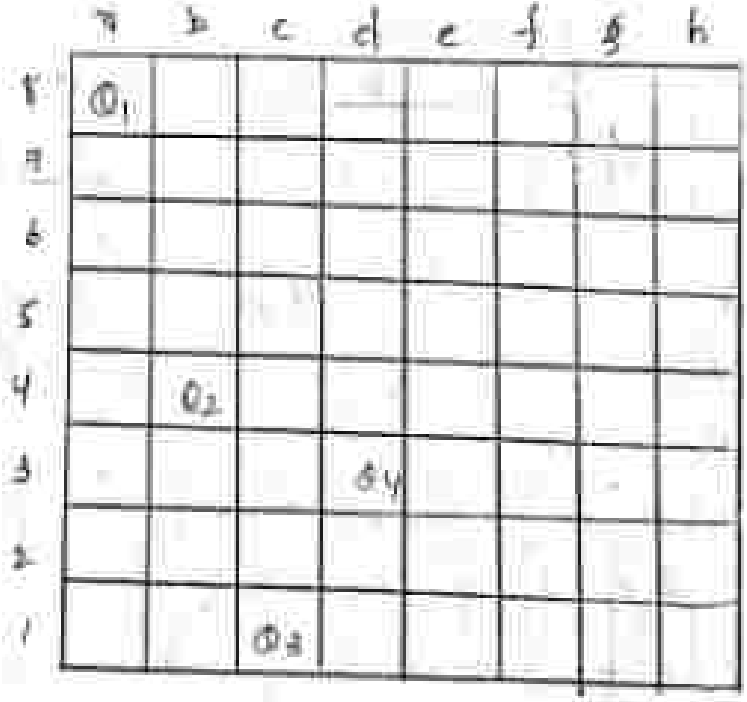
Here 2nd Queen is placed on 6th column and 4th row on 8x8 chess board.

step-3



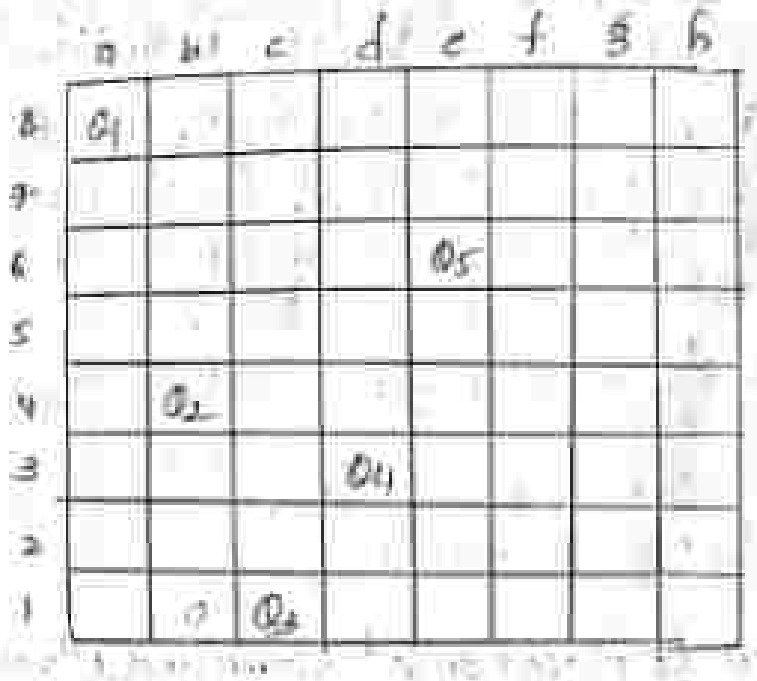
Here Queen Q_3 placed on 3rd column and 1st row on 8x8 chess board.

step-4:



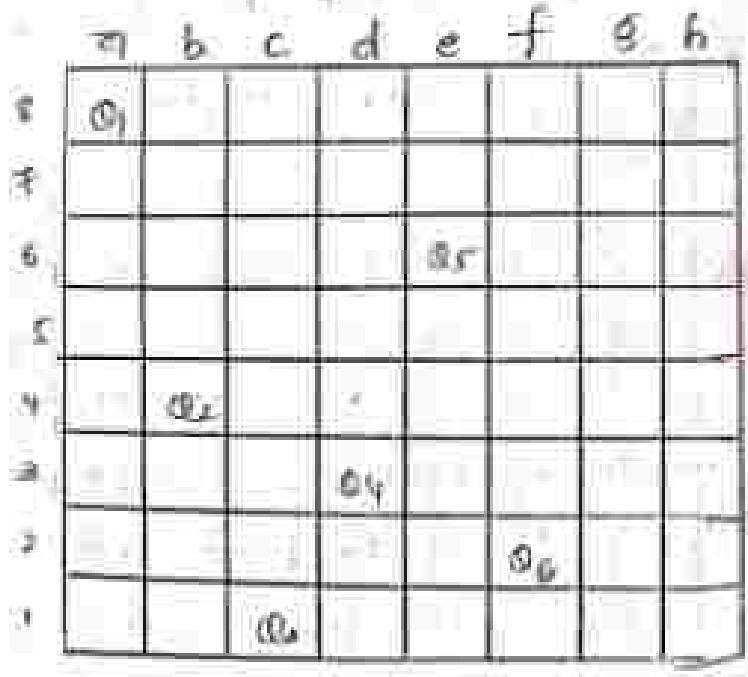
Here Queen Q_4 placed on 4th column and 2nd row on 8x8 chess board.

Step-5:



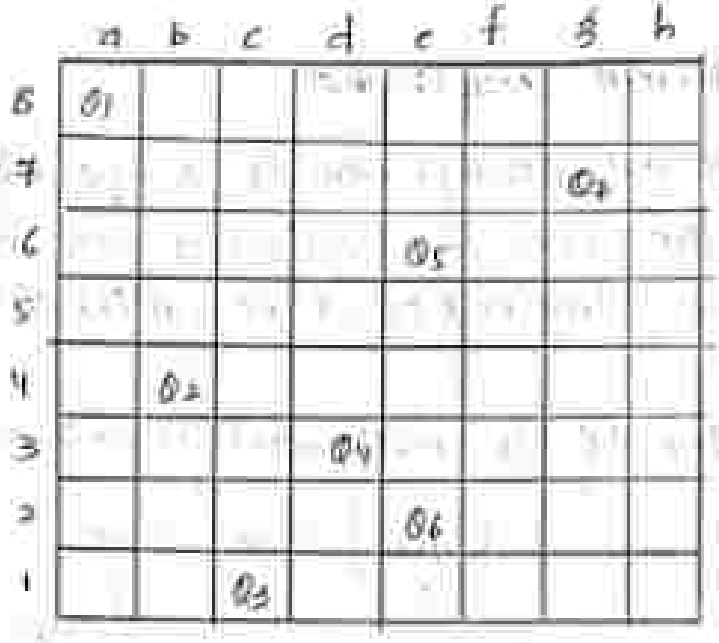
Here we place the 5th Queen on e th column and 6th row on 8x8 chess board.

Step-6:



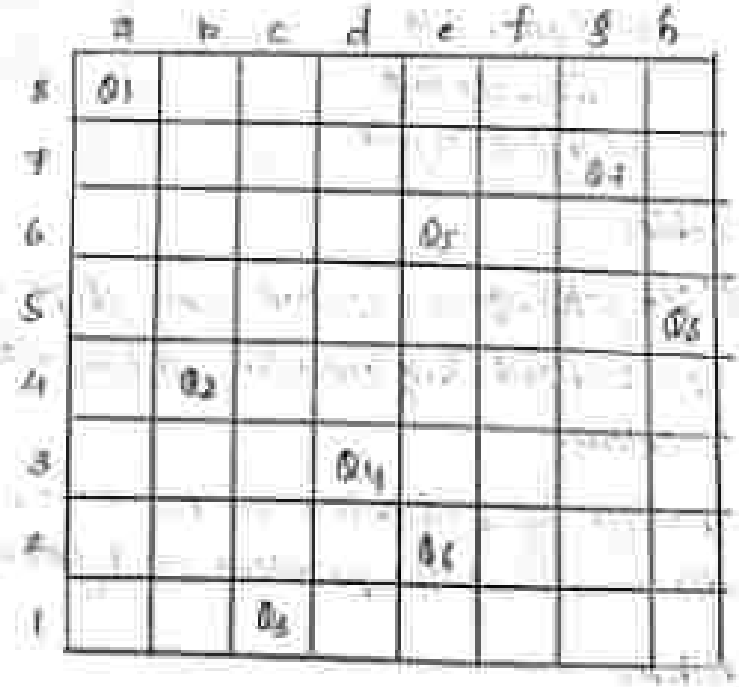
Here we place 6th Queen on f th column and 2nd row on 8x8 chess board.

step-7:



Here we place the 7th Queen on 8th column, 7th row on 8x8 chess board.

step-8:



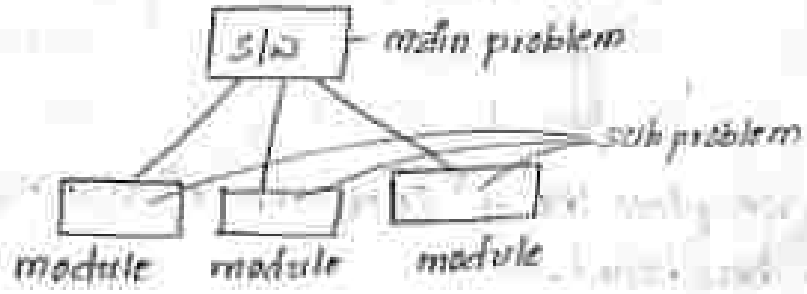
Here we place the Queen 8 Q8 on h^th column and 5th row 8x8 chess board

Problem characteristics

1. Is problem decomposable?

Here the main problem is decomposable into sub problems where the solutions of these into sub problems are combined to get the soln for the main problem.

Ex: problem is s/w development as shown below



2. Can solution steps can be ignored or undone?

there we have three things

1. Ignorable
2. Recoverable
3. Irrecoverable

3. Ignorable:

While solving a problem some steps can be ignored i.e without implementing them also we can solve the problem.

Ex: Declaration of variables are not does not cause any effect solving the problem in python.

2. Recoverable:

If we can do the done work 'undone' is called recoverable.

Ex: Filling online forms

3. Irrecoverable:

Here the 'done' work cannot be 'undone'

Ex: Net booking, pl. payment

4. Is the universe predictable?

Here we say the people living in the universe can predict the soln for some problems and cannot predict the soln for some problems based on that we divide the problem as

1. Certain outcome
2. Uncertain outcome.

1. Certain Outcome:

Here the problem which has a soln is called problems with certain outcome.

Ex: Tic-Tac-toe

2. Uncertain Outcome:

The problem which may (or) may not be solved come under problems with uncertain outcome.

Ex: Scientific evaluation (or) research

5. Is good solution is absolute (or) relative:

Absolute:

Here finding the goal node with minimum path cost.

Relative:

Finding the goal node by any way (or) any means.

6. Is solution is a state (or) path?

State: Other than initial state there will be a change

of states when particular actions are performed on any of the states.

path:

path is nothing but collection of states - path is a solo but not the state for any given problem

6. Does the task require interaction with a person?

Not all situations need conversation with human but some task like human diagnosis required a conversation between human and machine.

Two-player game performance

The two-player game follows the zero sum approach

Zero sum:

If one player wins that shows the other player was lost and viceversa. This concept we called it as zerosum Agent.

The element which performs project/work is known as Agent.

In two-player game there will be two agents

The ex for two-player game is tic-tac-toe as explained below.

Ex: tic-tac-toe