MODULE 3

PROBLEM SOLVING Refer Chapter 3 & 4

Solving problem by **SEARCHING** • An agent can find a **sequence of actions** that achieves its goals when no single action will do. • **Problem solving begin**s with precise definitions of PROBLEMS AND THEIR SOLUTIONS

– **Uninformed search algorithms**—algorithms that are given **no information** about the problem other than its definition – **Informed search algorithms**—Can work well given some **guidance** on where to look for solutions.

earc Agortm Termnooges

• **Search:** Searching is a step by step procedure to solve a search-problem in a given search space.

• A search problem can have three main factors:

– **Search Space:** Search space represents a set of **possible solutions**, which a system may have.

– **Start State:** It is a state from where agent **begins the search**.

– **Goal test:** It is a function which observe the current state and returns whether the **goal state is achieved or not.**

• **Search tree:** A tree representation of search problem is called Search tree. • **Actions:** It gives the **description of all the available actions** to the agent. • **Transition model:** Representation of description of what each action does. • **Path Cost:** It is a function which assigns a numeric cost to each path. • **Solution:** It is an action sequence which leads from the start node to the goal node.

• **Optimal Solution:** If a solution has the lowest cost among all solutions.

PROBLEM-SOLVING AGENTS

▪ **GOAL FORMULATION**:- Based on the **current situation** and the agent’s performance measure.

▪ **PROBLEM FORMULATION**:- Process of **deciding what action**s and states to **consider**, given a **GOAL**

▪ “**FORMULATE, SEARCH, EXECUTE**” design for the agent

EXAMPLE PROBLEMS

(Toy and real-world problems)

A TOY PROBLEM is intended to illustrate or exercise various problem solving methods given a **concise, exact description** and hence is usable by different researchers to compare the performance of algorithms.

REAL-WORLD PROBLEM is one whose solutions people actually care about-Eg healthcare

Example Problems

• Toy problems

– Illustrate/test various problem-solving methods

– Concise, exact description

– Can be used to compare performance

**EXAMPLES**: 8-puzzle, 8-queens problem, Cryptarithmetic, Vacuum world, Missionaries and cannibals, simple route finding

• Real-world problem

– More difficult

– No single, agreed-upon specification (state, successor function, edge cost) – *Examples*: Route finding, VLSI layout, Robot navigation, Assembly sequencing

STATE SPACE REPRESENTATION • A problem can be defined formally by **five components**:

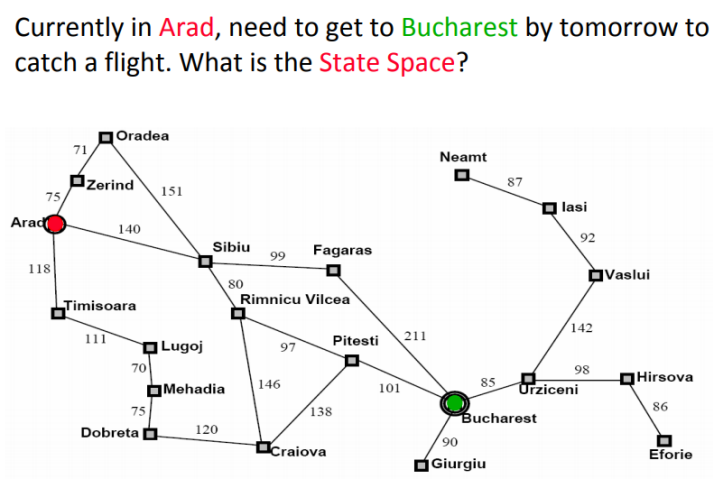
**1. INITIAL STATE** that the agent starts in.

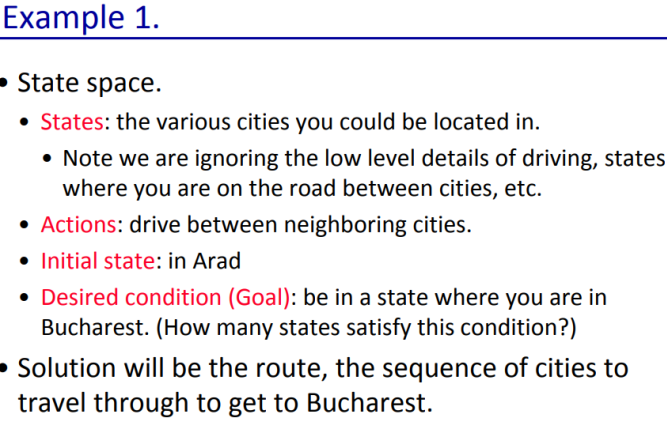
2. Description of the **POSSIBLE ACTIONS** available to the agent. **3. DESCRIPTION of what each action does**. [transition model]

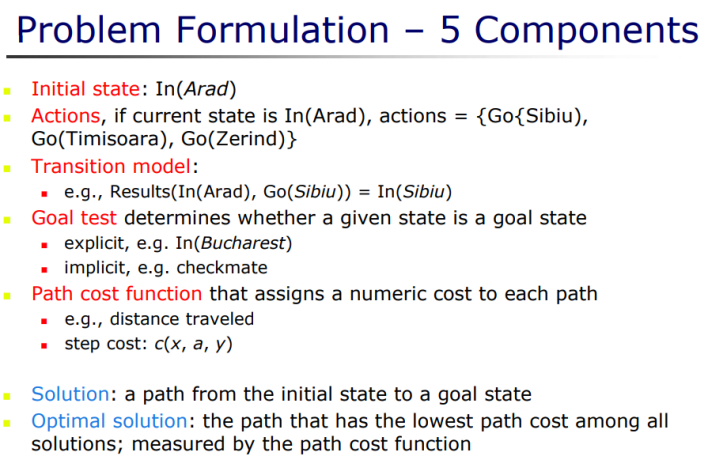
**4. GOAL TEST**, which determines whether a given state is a goal state **5. PATH COST FUNCTION** that assigns a numeric cost to each path **A SOLUTION TO A PROBLEM IS AN ACTION SEQUENCE THAT LEADS FROM THE INITIAL STATE TO A GOAL STATE.**

**Note**: Solution quality = path cost function, & **OPTIMAL SOLUTION** 🡪lowest path cost among all solutions

EXAMPLE 1

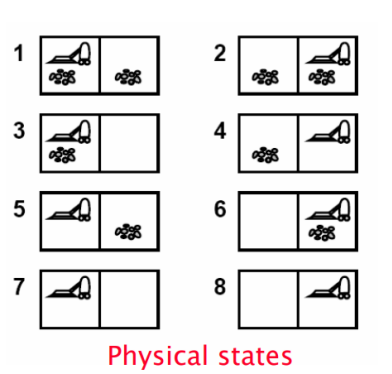






Example 2

Vacuum Cleaner

• The vacuum world 

– The world has only two

*locations*

– Each location may or may not

contain *dirt*

– The agent may be in one

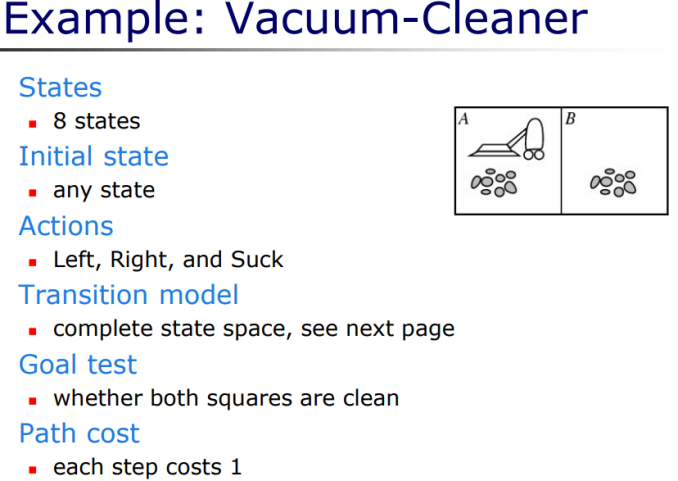
location or the other

– 8 possible *world states*

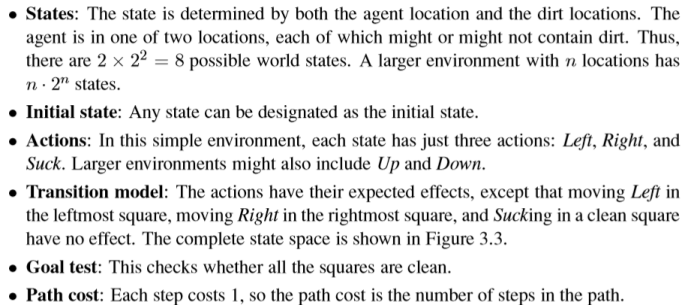
– Three possible actions: *Left,*

*Right, Suction*

– *Goal*: clean up all the dirt



Formulating a Vacuum Agent



• Example 3

Missionaries and Cannibals

**Question:** In this problem, three missionaries and three cannibals must cross a river using a boat which can carry at most two people, under the constraint that, for both banks, that the missionaries present on the bank cannot be outnumbered by cannibals. The boat cannot cross the river by itself with no people on board.

Short Video

Missionaries and cannibals

• There are three missionaries and three cannibals on the left bank of a river. 

• They wish to cross over to the right bank using a boat that can only carry two at a time.

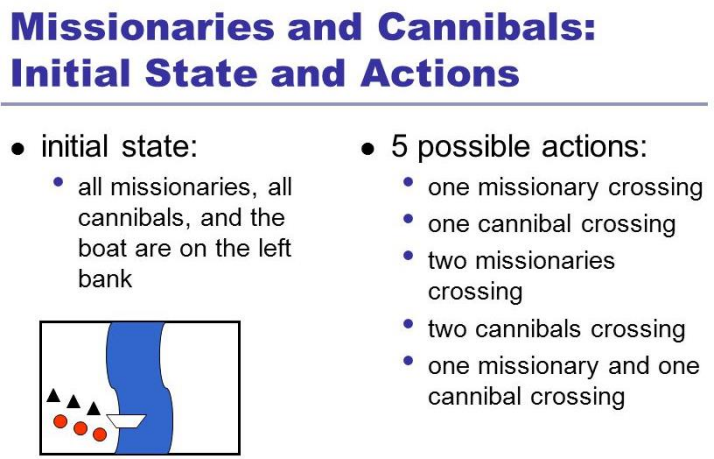
• The number of cannibals on either bank must never exceed the number of missionaries on

the same bank, otherwise the missionaries

will become the cannibals' dinner!

• Plan a sequence of crossings that will take everyone safely across.

25



Solution

First let us consider that both the missionaries (M) and cannibals(C) are on the same side of the river- Left Right

Initially the positions are : 0M , 0C and 3M , 3C (B)

Now let’s send 2 Cannibals to left of bank : 0M , 2C (B) and 3M , 1C • Send one cannibal from left to right : 0M , 1C and 3M , 2C (B) • Now send the 2 remaining Cannibals to left : 0M , 3C (B) and 3M , 0C Send 1 cannibal to the right : 0M , 2C and 3M , 1C (B)

• Now send 2 missionaries to the left : 2M , 2C (B) and 1M . 1C • Send 1 missionary and 1 cannibal to right : 1M , 1C and 2M , 2C (B) • Send 2 missionaries to left : 3M , 1C (B) and 0M , 2C

• Send 1 cannibal to right : 3M , 0C and 0M , 3C (B)

• Send 2 cannibals to left : 3M , 2C (B) and 0M , 1C

• Send 1 cannibal to right : 3M , 1C and 0M , 2C (B)’

• Send 2 cannibals to left : 3M , 3C (B) and 0M , 0C

• Here (B) shows the position of the boat after the action is performed. Therefore all the missionaries and cannibals have crossed the river safely.

Formulating Missionaries & Cannibal Problem

• **State space**: Triple (**x** , **y** , **z** ) with 0 ≤ x , y , z ≤ 3, where x , y , and z represent the number of **missionaries**, **cannibals** and **boats** currently on the original bank.

• **Initial State**: ( 3 , 3 , 1 )

– Successor function: From each state, either bring

• one missionary, one cannibal,

• two missionaries, two cannibals, or

• one of each type to the other bank.

• Note: Not all states are attainable (e.g., (0,0,1)), and some are illegal.

• Goal State: (0,0,0)

• Path Costs: 1 unit per crossing

Example 4

**The 8 Puzzle**

Sliding-block/tile-puzzle(most popular instruments in AI studies) [toy-problem]

• Arrange the tiles so that all the tiles are in the correct positions. – You can move the BLANK tile up, down, left, or right, so long as the following conditions are met:

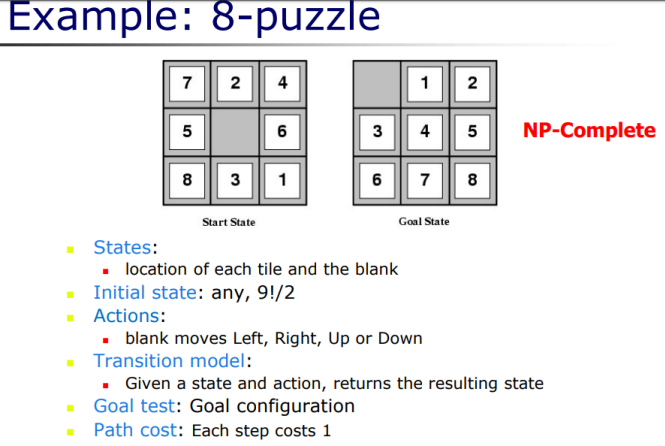
• There’s no other tile blocking you in the direction of the movement;

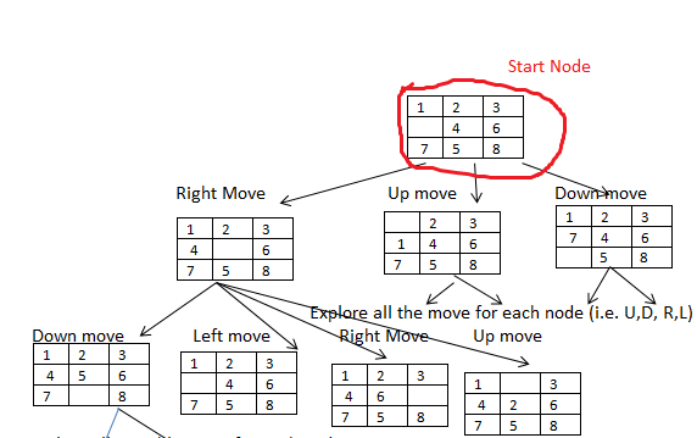
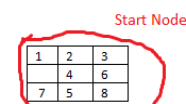
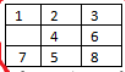
• You are not trying to move outside of the boundaries/edges.

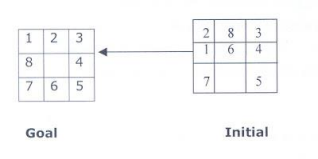
INITIAL GOAL

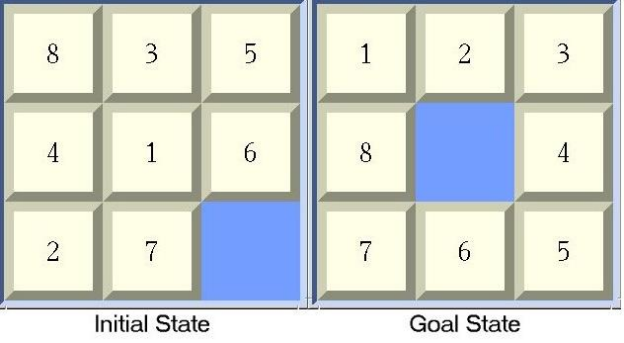
1 2 3 1 2 3 \* 4 6 4 5 6 7 5 8 7 \* 8

Formulating for 8 puzzle problem

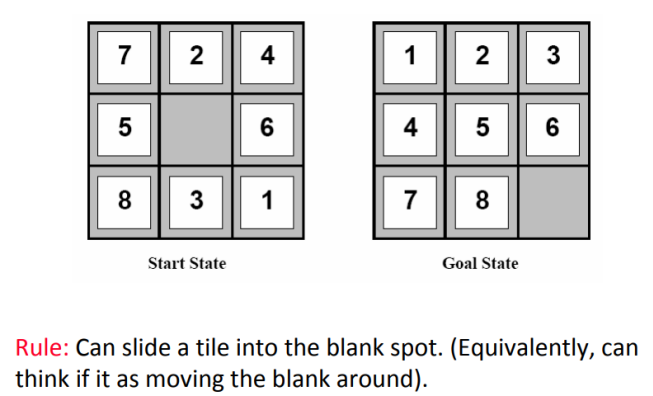








8-puzzle





• Example 5



4 Queen Problem

THE N-QUEEN PROBLEM: Place n queens on an n by n chess board so that no two of them are on the same row, column, or diagonal .



**8-queens problem**

The goal of the 8-queens problem is to place eight queens on a chessboard such that nqueen attacks any other.

(A queen attacks any piece in the same row, column or diagonal.)

Formulating 8 Queens Problem

Draw the State Space tree for 4 Queen Problem





4 Queen

Possible Solution



Example 6

Water Jug Problem

• **Problem:** You are given two 

jugs, a 4-gallon one and a 3-

gallon one . Neither has any

measuring mark on it . There is

a pump that can be used to fill

the jugs with water . How can

you get exactly 2 gallons of

water into the 4-gallon jug.

•

X is 4L and Y is 3L

1. X = 0 , Y = 3 .

2. X = 3 , Y = 0 . (Transfer Y to X )

3. X = 3 , Y = 3 .( Capacity of X being 4)

4. X =4 , Y = 2 . (Fill X up to brim )

5. X=2 , Y= 0 . (Fill X with contents of Y after emptying X)

REAL WORLD PROBLEMS

• Web sites and in-car systems that provide driving directions • Touring Problems

• Travelling Salesman Problem

• Robot Navigation

• Others, such as

– routing video streams in computer networks

– military operations planning

– airline travel planning systems

CONSIDER THE AIRLINE TRAVEL PROBLEMS THAT MUST BE SOLVED BY A TRAVEL-PLANNING WEB SITE



• **A VLSI layout problem** requires positioning millions of components and connections on a chip to minimize area, minimize circuit delays, minimize stray capacitances, and maximize manufacturing yield.

• **PROTEIN DESIGN** in which the goal is to find a sequence of amino acids that will fold into a three-dimensional protein with the right properties to cure some disease.

Defining Problem as a **STATE SPACE SEARCH**1. A state space that contains **all possible configurations** of all relevant objects.

2. Specify one or more states describing all possible situations from which problem solving may start **(Initial States)**

3. Specify one or more states that would be acceptable as solutions to problems **(Goal States).**

4. Specify set of rules that describe **actions** (operators) available.

1. What unstated assumptions are present in the informal problem description?

2. How general should the rules be ?

3. How much of the work required to solve the problem should be pre computed and represented in the rules?

Production Systems

• A set of **rules**(Left side[pattern] determining applicability of rule on right side[describes operation to be performed if rule applied]) • One or more **knowledge base** (Structured)

• **Control strategy**-order of rules applied/compared , way of resolving conflicts.

• **A rule applier**

**S.N** 

**o.Initial State Condition Final state Description of action taken** 1. (x,y) If x<4 (4,y) Fill the 4 gallon jug completely 2. (x,y) if y<3 (x,3) Fill the 3 gallon jug completely

3. (x,y) If x>0 (x-d,y) Pour some part from the 4 gallon jug

4. (x,y) If y>0 (x,y-d) Pour some part from the 3 gallon jug 5. (x,y) If x>0 (0,y) Empty the 4 gallon jug

**S.N** 

**o.Initial State Condition Final state Description of action taken**

6. (x,y) If y>0 (x,0) Empty the 3 gallon jug

7. (x,y) If (x+y)<7 (4, y-[4-x]) Pour some water from the 3 gallon jug to fill the four  gallon jug

8. (x,y) If (x+y)<7 (x-[3-y],3) Pour some water from the 4 gallon jug to fill the 3 gallon jug.

9. (x,y) If (x+y)<4 (x+y,0) Pour all water from 3 gallon jug to the 4 gallon jug 10. (x,y) if (x+y)<3 (0, x+y) Pour all water from the 4 gallon jug to the 3 gallon jug

**S.N** 

**o.Initial State Condition Final state Description of action taken** 1. (x,y) If x<4 (4,y) Fill the 4 gallon jug completely 2. (x,y) if y<3 (x,3) Fill the 3 gallon jug completely 3. (x,y) If x>0 (x-d,y) Pour some part from the 4 gallon jug

4. (x,y) If y>0 (x,y-d) Pour some part from the 3 gallon jug

5. (x,y) If x>0 (0,y) Empty the 4 gallon jug

6. (x,y) If y>0 (x,0) Empty the 3 gallon jug

7. (x,y) If (x+y)<7 (4, y-[4-x]) Pour some water from the 3 gallon jug to fill the fogallon jug 8. (x,y) If (x+y)<7 (x-[3-y],3) Pour some water from the 4 gallon jug to fill the 3  gallon jug.

9. (x,y) If (x+y)<4 (x+y,0) Pour all water from 3 gallon jug to the 4 gallon jug 10. (x,y) if (x+y)<3 (0, x+y) Pour all water from the 4 gallon jug to the 3 gallon j

8 puzzle problem



Problem Characteristics

**1. Is the problem decomposable?** Sub goal

**2. Can solution steps be ignored or undone ?**Eg (8 puzzle ,Chess , Theorem Proving) 1. Ignorable , Recoverable , Irrecoverable (3 classes of problems).

**3. Is the universe predictable?**(Plan Problem Solving?) Eg Lawyer,Controlling Robot arm… **4. Is a Good solution Absolute or relative?** Good Heuristics play imp role. **5. Is the Solution a state or a path?**

**6. What is the role of knowledge?** Eg Chess-Strategy and Tactics, Newspaper reading? **7. Does the task require interaction with a person?** Solitary and Conversational. Eg Medical diagnosis v/s peoples reaction?

**8. Problem Classification**

Issues in designing of search problems

• **Direction of search**(Forwards v/s Backward reasoning) • **Selecting matching** (applicable)r**ules.**

– Efficient procedures to match rules against states in production system.

• **Knowledge Representational Problem**-representing each node of the search process-Eg Chess (Array[small] v/s ??)



Depth

Limited

Search

Iterative

deepening

Depth First

Search

AO\*

**NINFORMED SEARCH STRATEGIES(blind search**“Strategies have **no additional information** about states beyond that provided in the **problem definition**” • All they can do is generate successors and **distinguish a goal state from a non-goal state.**

– Breadth-first search (BFS)

– Depth-first search (DFS)

– Depth Limited search(DLS)

– Iterative Deepening depth first search(IDDS)

BREADTH FIRST SEARCH

Breadth-first search is a simple strategy in which the root node is expanded first, then all the successors of the root node are expanded next, then their successors, and so on.

**ADVANTAGES**:

• BFS will **provide a solution** if any solution exists.

• If there are more than one solutions for a given problem, then BFS will provide the **minimal solution** which requires the least number of steps.

**DISADVANTAGES:**

• It requires **lots of memory** since each level of the tree must be saved into memory to expand the next level.

• BFS needs lots of time if the solution is **far away from the root node.**

DEPTH-FIRST SEARCH

• Depth-first search always expands the deepest node in the current frontier of the search tree.

Root node--->Left node ----> right node.

**Advantages:**

• DFS consumes very less memory space.

• It will reach at the goal node in a less time period than BFS if it traverses in a right path.

• It may find a solution without examining much of search because we may get the desired solution in the very first go.

**Disadvantages**

• This algorithm may not terminate and go on infinitely on one path.

– The solution to this issue is to **choose a cut-off depth.** – If the ideal cut-off is *d*, and if **chosen cut-off is lesser** than *d*, then this algorithm may fail.

– If **chosen cut-off is more** than *d*, then execution time increases. • Its complexity depends on the number of paths • It cannot check duplicate nodes.

**DEPTH LIMITED SEARCH**

• The embarrassing failure of depth-first search in infinite state spaces can be alleviated by supplying depth-first search with a **predetermined depth limit .**

– Nodes at depth are treated as if they have no successors.

• This approach is called depth-limited search

• Depth-limited search can terminate with two conditions: • If the solution is found.

• If there is no solution within given depth limit.

• **PROCESS**: If depth is fixed to 2, DLS carries out depth first search till second level in the search tree.





**Advantages:**

• Depth-limited search is Memory efficient.

**Disadvantages:**

• Depth-limited search also has a disadvantage of incompleteness.

• It may not be optimal if the problem has more than one solution.

• What if solution is deeper than L?

– Increase L iteratively.

**ITERATIVE DEEPENING SEARCH**

This inherits the memory advantage of Depth First search, and is better in terms of time complexity than Breadth first search

ITERATIVE DEEPENING SEARCH

• Iterative deepening depth-first search/ Iterative deepening search [**depth-first tree search that finds the best depth limit**]

– It does this by gradually increasing the limit—first 0, then 1, then 2,and so on—until a goal is found.

• This will occur when the depth limit reaches d, the depth of the shallowest goal node.

• Iterative deepening combines the benefits of depth-first and breadth-first search.

**1'st Iteration-----> A 2'nd Iteration----> A, B, C**

**3'rd Iteration------>A, B, D, E, C, F, G**

**Advantages:**

• It combines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

**Disadvantages:**

• The main drawback of IDDFS is that it repeats all the work of the previous phase.