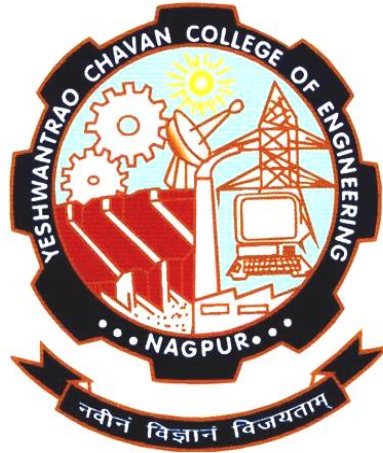


Yeshwantrao Chavan College of Engineering

(An Autonomous Institution affiliated to Rashtrasant Tukadoji Maharaj Nagpur University)
Wanadongri Hingna Road, Nagpur-441 110

Department of Computer Science & Engineering



LABORATORY MANUAL

ARTIFICIAL INTELLIGENCE

By

Dr.Lalit B. Damahe

Ms.Priyanka Pitale

SEVENTH SEMESTER

IVTH YEAR

Artificial Intelligence

Introduction

Artificial Intelligence will define the next generation of software solutions. This computer science course provides an overview of AI, and explains how it can be used to build smart apps that help organizations be more efficient and enrich people's lives. It uses a mix of engaging lectures and hands-on activities to help you take your first steps in the exciting field of AI. Discover how machine learning can be used to build predictive models for AI. Learn how software can be used to process, analyze, and extract meaning from natural language; and to process images and video to understand the world the way we do. Find out how to build intelligent bots that enable conversational communication between humans and AI systems.

COURSE OUTCOMES (CO): (Practical Course)

After Completion of this course students will be able to:

CO1	
CO2	
CO3	
CO4	
CO5	
CO6	

Practical 01

Aim: Write a program to Implement DFS for Water Jug Problem.

Theory:

Water Jug Problem:

This is the water jug problem using simple depth-first search of a graph. Jug X holds 4 liters, and jug Y holds 3 liters. There is a pump, which can be used to fill either Jug. How can you get exactly 2 liters of water into the 4-liter jug?

Assumptions:

- a) We can fill a jug from the pump
- b) We can pour water out of the jug onto the ground
- c) We can pour water from one jug to another
- d) There are no other measuring devices available

To solve the water jug problem, apart from problem statement we also need a control structure that loops through a simple cycle in which some rule whose left side matches the current state is chosen, the appropriate change to state is made as described in corresponding right side and the resulting state is checked to see if it corresponds to a goal state.

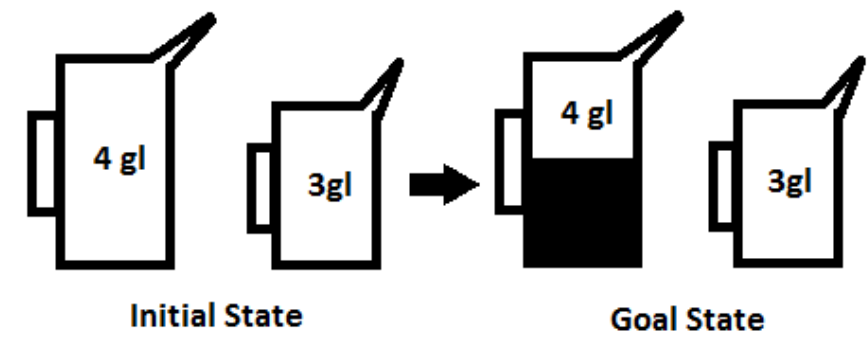


Fig.1: Water Jug: Initial and Goal State

The following table gives the production rule to solve the water jug problem

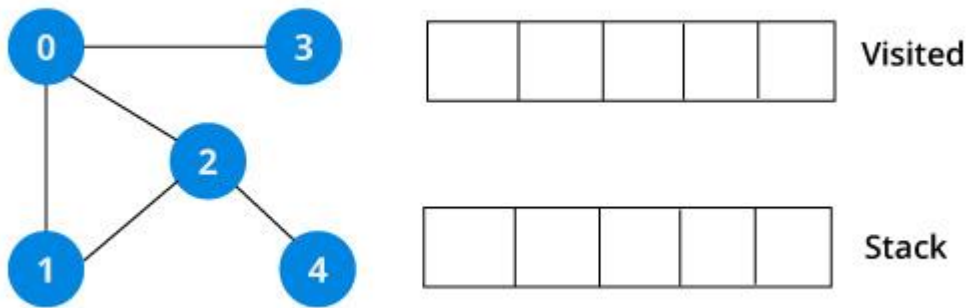
Rule No	Current state	New state	Action
1	(x, y) if $x < 4$	(4, y)	Fill the 4-gallon jug.
2	(x, y) if $y < 3$	(x, 3)	Fill the 3-gallon jug.
3	(x, y) if $x > 0$	(x-d, y)	Pour some water out of 4-gallon jug.
4	(x, y) if $y > 0$	(x, y-d)	Pour some water out of 3-gallon jug.
5	(x, y) if $x > 0$	(0, y)	Empty the 4-gallon jug on ground.
6	(x, y) if $y > 0$	(x, 0)	Empty the 3-gallon jug on ground.
7	(x, y) if $(x+y) \geq 4$ & $(y > 0)$	(4, y-(4-x))	Pour water from 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full.
8	if $(x+y) \geq 3$ & $(x > 0)$	(x-(3-y), 3)	Pour water from 4-gallon jug into the 3-gallon jug until the 3-gallon jug is full.
9	(x, y) if $(x+y) \leq 4$ & $(y > 0)$	(x+y, 0)	Pour all the water from 3-gallon jug into the 4-gallon jug.
10	(x, y) if $(x+y) \leq 3$ & $(x > 0)$	(0, x+y)	Pour all the water from 4-gallon jug into the 3-gallon jug.
11	(0, 2)	(2, 0)	Pour the 2 gallons from the 3-gallon jug into the 4-gallon jug.
12	(2, y)	(0, y)	Empty 2 gallons in the 4-gallon jug on the ground.

Depth First Search :

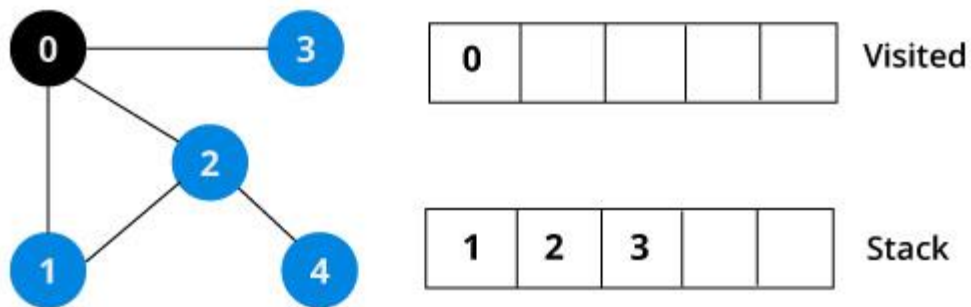
The DFS algorithm works as follows:

1. Start by putting any one of the graph's vertices on top of a stack.
2. Take the top item of the stack and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of stack.
4. Keep repeating steps 2 and 3 until the stack is empty.

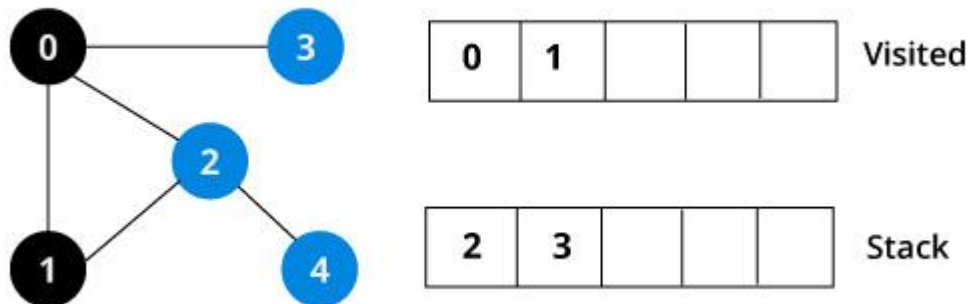
Let's see how the Depth First Search algorithm works with an example. Here Visited is Closelist and Stack is Openlist.



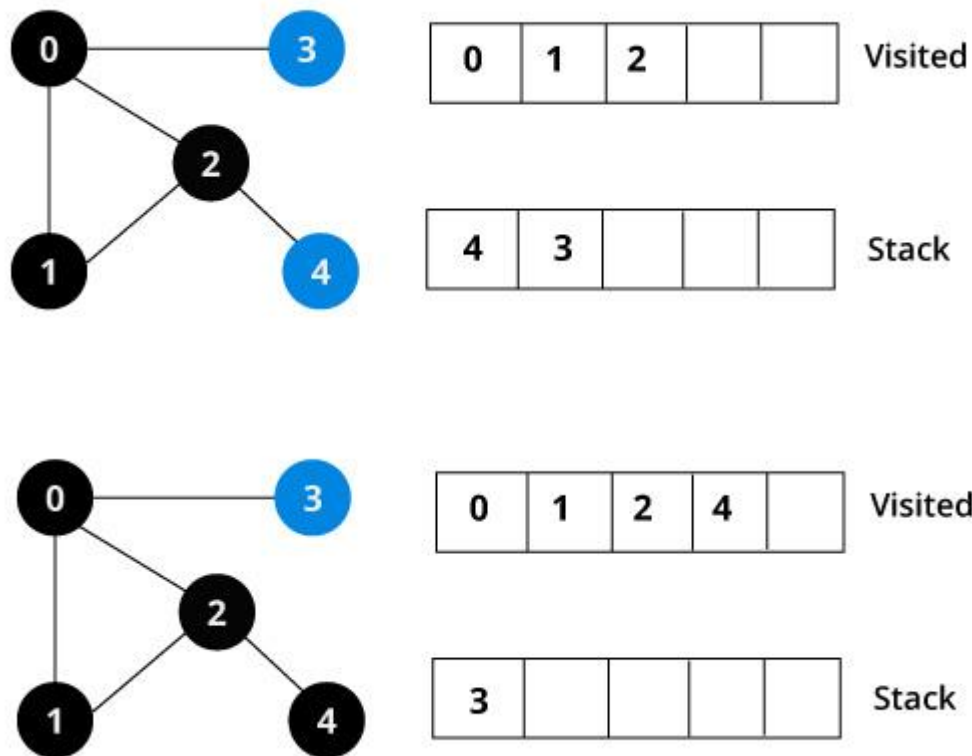
We start from vertex 0, the DFS algorithm starts by putting it in the Visited list and putting all its adjacent vertices in the stack.



Next, we visit the element at the top of stack i.e. 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.



Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

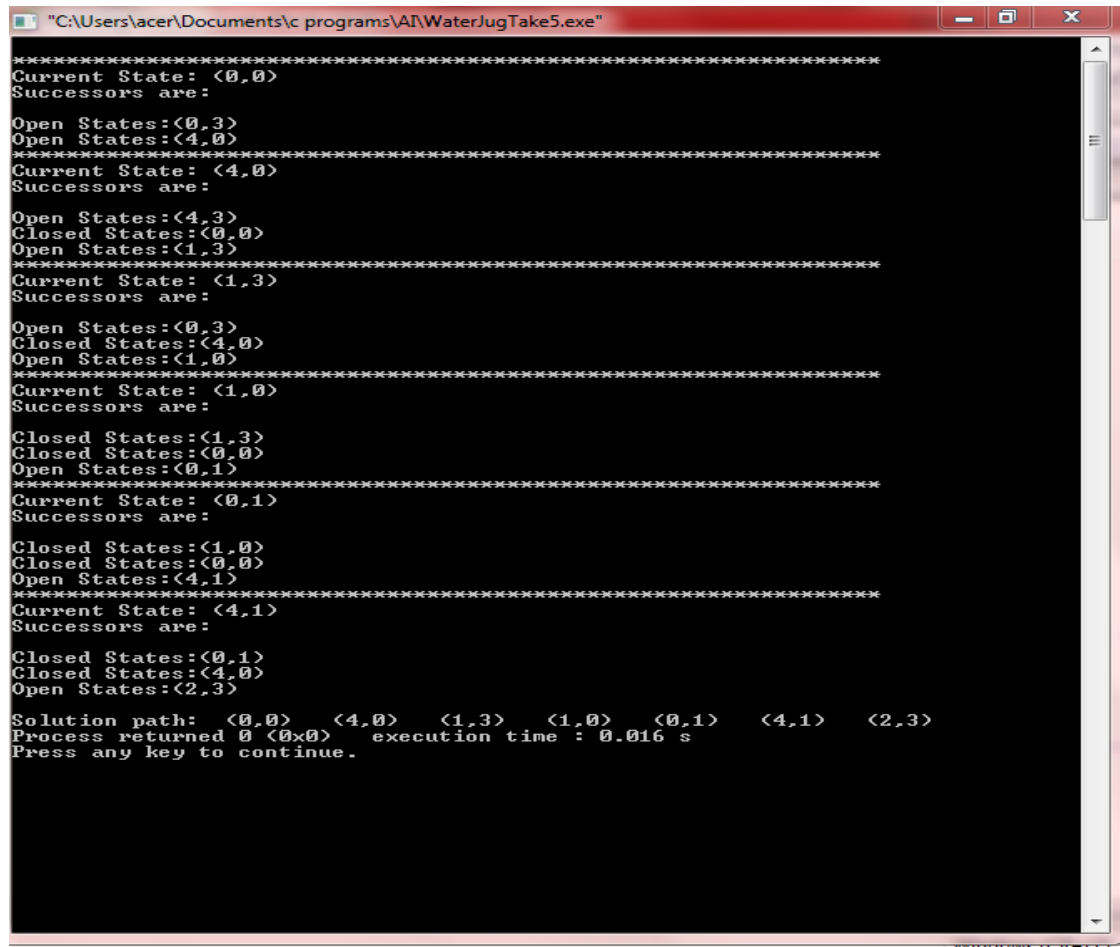


After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Traversal of the graph.

Procedure/Algorithm:

1. Create a structure for storing current state
2. Create a structure for storing non-repeating(Open) successors
3. Create a structure for storing repeating (Close) successors
4. Create a structure for storing all successors of current state
5. Function to generate successors
6. Function for moving from current to next state
7. If generated successor has been already traversed, putting it into closed structure
8. If generated successor is a new state, putting it into open structure
9. Print if goal state Achieved else Backtracking if goal state not achieved

Sample Output:



```
*****
Current State: <0,0>
Successors are:
Open States:<0,3>
Open States:<4,0>
*****
Current State: <4,0>
Successors are:
Open States:<4,3>
Closed States:<0,0>
Open States:<1,3>
*****
Current State: <1,3>
Successors are:
Open States:<0,3>
Closed States:<4,0>
Open States:<1,0>
*****
Current State: <1,0>
Successors are:
Closed States:<1,3>
Closed States:<0,0>
Open States:<0,1>
*****
Current State: <0,1>
Successors are:
Closed States:<1,0>
Closed States:<0,0>
Open States:<4,1>
*****
Current State: <4,1>
Successors are:
Closed States:<0,1>
Closed States:<4,0>
Open States:<2,3>
*****
Solution path: <0,0> <4,0> <1,3> <1,0> <0,1> <4,1> <2,3>
Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

Conclusion: Successfully implemented Depth first Search to solve water jug Problem using given production rules

Review Questions:

1)What is Water Jug Problem

2)How production rule useful to solve problem

3)What is DFS? How it works?

4)How to implement DFS?