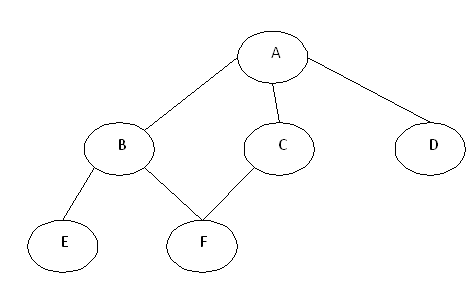
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Finolex Academy of Management and Technology, Ratnagiri** | | | | | | | | | |
| **Department of Information Technology** | | | | | | | | | |
| Subject name: Intelligent Systems | | | | | | | | Subject Code: BEITC703 | | | |
| Class | | BE IT | | Semester – VII (CBGS) | | | | Academic year: 2018-19 | | | |
| Name of Student | |  | | | | | **QUIZ Score :** | | | | |
| Roll No | |  | | | Assignment/Experiment No. | | | | | 01 | |
| Title:  **To implement water jug problem using BFS and DFS.** | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **1. Course objectives applicable:**  **COB2**.Understand the different searching techniques to solve the different AI problems on the basis of knowledge of agent | | | | | | | | | | | |
| **2. Course outcomes applicable:**  **CO2** –Solve the problems based on searching techniques. | | | | | | | | | | | |
| **3. Learning Objectives:**   1. To understand concept of DFS and BFS. 2. To understand water jug problem 3. To program water jug problem for solution no 2 4. To get the output which will give the correct rule applied for required steps | | | | | | | | | | | |
| **4. Practical applications of the assignment/experiment:** Euler circuits,Bioconnected graphs | | | | | | | | | | | |
| **5. Prerequisites**:   1. To learn the use of intelligent agents in uninformed search. 2. To understand the programming methodology for water jug problem. | | | | | | | | | | | |
| **6. Hardware Requirements**:   1. PC with minimum 2GB RAM   **7. Software Requirements:**  1. Windows installed  2. JDK/Net beans | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **8. Quiz Questions (if any): (Online Exam will be taken separately batchwise, attach the certificate/ Marks obtained)**   1. What is AI? 2. What is an Agent? 3. What is PEAS? 4. What is not an Uninformed search technique? | | | | | | | | | | | |
|  | | | | | | | | | | | |
| **9. Experiment/Assignment Evaluation:** | | | | | | | | | | | |
| **Sr. No.** | **Parameters** | | | | | | | | **Marks obtained** | | **Out of** |
| **1** | Technical Understanding (Assessment may be done based on Q & A **or** any other relevant method.) Teacher should mention the other method used - | | | | | | | |  | | 6 |
| **2** | Neatness/presentation | | | | | | | |  | | 2 |
| **3** | Punctuality | | | | | | | |  | | 2 |
| **Date of performance (DOP)** | | |  | | | **Total marks obtained** | | |  | | **10** |
| **Date of checking (DOC)** | | |  | | | **Signature of teacher** | | | | | |

**10. Theory:**

**Breadth First Search (BFS)**

This is a very different approach for traversing the graph nodes. The aim of BFS algorithm is to traverse the graph as close as possible to the root node. Queue is used in the implementation of the breadth first search. Let’s see how BFS traversal works with respect to the following graph:

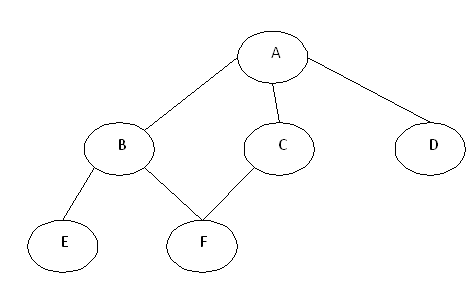


If we do the breadth first traversal of the above graph and print the visited node as the output, it will print the following output. “A B C D E F”. The BFS visits the nodes level by level, so it will start with level 0 which is the root node, and then it moves to the next levels which are B, C and D, then the last levels which are E and F.

**Depth First Search (DFS)**

**Depth-first search** (**DFS**) is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) for traversing or searching [tree](https://en.wikipedia.org/wiki/Tree_data_structure) or [graph](https://en.wikipedia.org/wiki/Graph_(data_structure)) data structures. One starts at the [root](https://en.wikipedia.org/wiki/Tree_(data_structure)#Terminology) (selecting some arbitrary node as the root in the case of a graph) and explores as far as possible along each branch before [backtracking](https://en.wikipedia.org/wiki/Backtracking).

The aim of DFS algorithm is to traverse the graph in such a way that it tries to go far from the root node. Stack is used in the implementation of the depth first search. Let’s see how depth first search works with respect to the following graph:



As stated before, in DFS, nodes are visited by going through the depth of the tree from the starting node. If we do the depth first traversal of the above graph and print the visited node, it will be “A B E F C D”. DFS visits the root node and then its children nodes until it reaches the end node, i.e. E and F nodes, then moves up to the parent nodes.

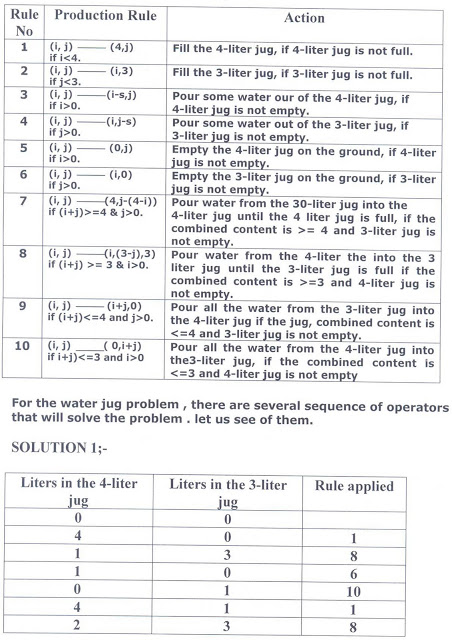
**Difference between BFS and DFS**

|  |  |  |
| --- | --- | --- |
| **Parameter** | **BFS** | **DFS** |
| **Time Complexity** | **O(bd)->**if we travesing till the maximum depth.  **O(bd+1)->**next level. | **O(bd)**  d-Maximum depth |
| **Space Complexity** | **O(bm)**  bm->linear space | **O(bm)**  bm->linear space |
| **Completeness** | **Yes** | **No** |
| **Optimal** | **Yes** | **No** |

**Properties BFS/DFS**

* BFS and DFS originating from v create a tree, whose nodes are visited by the algorithm and whose arcs are those traversed. Both create a forest of trees which spans G.
* If w is reachable by DFS and v is visited before, then w is a descendent of v in the DFS forest.
* An arc of a digraph G is a tree arc if it belongs to one of the trees of the forests formed by BFS or DFS.
* DFS may also be used to collect a [sample](https://en.wikipedia.org/wiki/Sample_(statistics)) of graph nodes. However, incomplete DFS, similarly to incomplete [BFS](https://en.wikipedia.org/wiki/Breadth-first_search#Bias_towards_nodes_of_high_degree), is [biased](https://en.wikipedia.org/wiki/Bias) towards nodes of high [degree](https://en.wikipedia.org/wiki/Degree_(graph_theory)).

**WaterJug Problem Rules:**



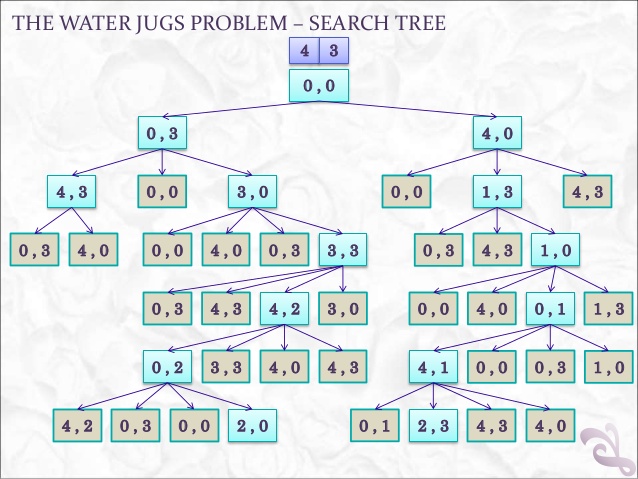
**Solution1:**

|  |  |  |
| --- | --- | --- |
| Liters in the 4-liter jug | Liters in the 3-liter jug | Rule applied |
| 0 | 0 |  |
| 4 | 0 | 1 |
| 1 | 3 | 8 |
| 1 | 0 | 6 |
| 0 | 1 | 10 |
| 4 | 1 | 1 |
| 2 | 3 | 8 |

**Solution2:**

|  |  |  |
| --- | --- | --- |
| Liters in the 4-liter jug | Liters in the 3-liter jug | Rule applied |
| 0 | 0 |  |
| 0 | 3 | 2 |
| 3 | 0 | 9 |
| 3 | 3 | 2 |
| 4 | 2 | 9 |
| 0 | 2 | 5 |
| 2 | 0 | 9 |

**WaterJug Problem-Search Tree:**



**11. Precautions**:

1. Find all available solution paths and represent it in tree.

2. Define rules and actions applicable for each rule.

**12. Installation Steps / Performance Steps –**

**Program.java**

public class Program

{

public static void main(String args[])

{

WaterJug w = new WaterJug();

w.checkGoal();

}

}

**WaterJug.java**

import java.util.\*;

public class WaterJug

{

int a\_max = 4;

int b\_max = 3;

int a = 0;

int b = 0;

int goal = 2;

void checkGoal()

{

int fin = 0;

while(fin != 1)

{

if((this.a == this.goal) || (this.b == this.goal)) { fin = 1; }

if(this.a==0)

{

fillA();

} else if ((this.a > 0) && (this.b != this.b\_max)) {

transferAtoB();

} else if ((this.a > 0) && (this.b == this.b\_max)) {

emptyB();

}

}

}

void fillA()

{

this.a = this.a\_max;

System.out.println("{" + this.a + "," + this.b + "}");

}

void fillB()

{

this.b = this.b\_max;

System.out.println("{" + this.a + "," + this.b + "}");

}

void transferAtoB()

{

int fin = 0;

while(fin != 1) {

this.b += 1;

this.a -= 1;

if((this.b == this.b\_max) || (this.a == 0)) { fin = 1;}

}

System.out.println("{" + this.a + "," + this.b + "}");

}

void emptyA() {

this.a=0;

System.out.println("{" + this.a + "," + this.b + "}");

}

void emptyB() {

this.b=0;

System.out.println("{" + this.a + "," + this.b + "}");

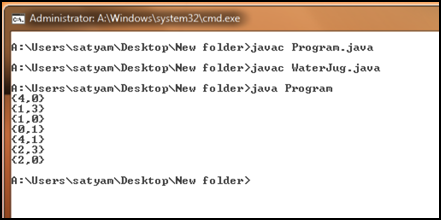
}

}

**13. Observations**

1. The output will give the sequence of nodes to find the solution path or goal node.

**14. Results:**

****

**15. Learning Outcomes Achieved**

1. Understanding the concept of uninformed search.
2. Understanding the water jug problem solved by uninformed search technique.

**16. Conclusion:**

1. **Applications of the studied technique in industry**
   1. DFS and BFS algorithms used to develop intelligent systems which solve mathematical problems based on graph.
2. **Engineering Relevance** 
   1. Such algorithms are very useful in searching techniques where number of solutions are more than one.
3. **Skills Developed**
   1. Implementation of BFS for water jug problem.

**17. References** :

[1] G. Görz, C.-R. Rollinger, J. Schneeberger (Hrsg.) “Handbuch der künstlichen

Intelligenz” Oldenbourg Verlag, 2003, Fourth edition

• [2] Turing, A. "Computing Machinery and Intelligence", Mind LIX (236): 433–460,

Ocotober, 1950.

• [3] Aristotle “On Interpretation”, 350 B.C.E, see:

http://classics.mit.edu/Aristotle/interpretation.html

• [4] Newell, A., Simon, H.A. “Human Problem Solving” Englewood Cliffs, N.J.: Prentice

Hall, 1972

• [5] Newell, A. “The Knowledge Level”, AI Magazine 2 (2), 1981, p. 1-20.