**ADS PROJECT**

**Missionaries and Cannibals**

**Project by:**

**1)Kshitij Shah (UID:2017140048)**

**2)Tirth Parekh (UID:2017140037)**

**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.

**Files in the Project:**

1.Menu.java

2.Play.java

3.State.java

4.Ads.java

5.Instruction.java

6.UCS.java

7.DLS.java

8.IDS.java

9.Gui.java

**Instruction to Run the program:**

1.There are 9 files in the same package named ads1

2.Open the project folder in NetBeans with these 2 files or any other editor but keep the file in a foldername as the same as the package name

3.Main method is in Ads.java so compile it and run it.One should get the output as follow

**Search Methods Used:**

Various graph traversal techniques have been used. Each state is represented as a node of the graph with the initial state being 3 missionaries and 3 cannibals on the left hand side. The goal state is achieved when there are 0 missionaries and 0 cannibals present on the left. Various nodes are not present in the graph because they don’t satisfy the condition of number of missionaries > number of cannibals. Here are the traversal techniques used:

**1.Uniform Cost Search(Breadth First Search):**

Uniform cost search is a search algorithm to find the shortest path. It does not involve any heuristic.It makes use of weights of each edge. Since in this problem weight of all edges is the same Uniform cost search is indirectly Breadth first Search.

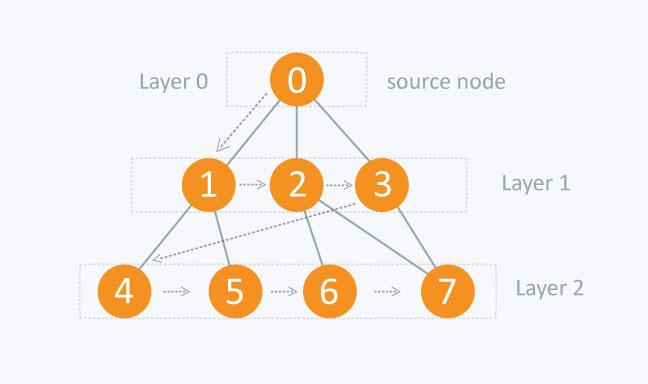
In BFS, one should start traversing from the source node and traverse the graph layerwise thus exploring the neighbor nodes(nodes which are directly connected to the source). BFS ends when goal State is reached. The time complexity of BFS is O(V+E) where V is the number of vertices and E is the number of edges.

ALGORITHM:

1.Insert the source node in a queue

2.Till the queue is not empty, dequeue front element and visit all its child.If he child is not explored then add it to the queue.

3.End when the goal state is reached



**2.Depth Limited Search**

Depth limited Search is used when we have prior knowledge of the problem. If DFS is used in an unbounded tree then we will not get a solution. Hence DLS is used in which traversal does not take place beyond the assigned limit. If the solution is present before the limit then we will get the output else we wont get the output. This is the only disadvantage of DLS.

Algorithm is similar to DFS in which traverse along a particular path till the end and then we backtrack till we reach a goal state.In DLS only difference being it backtracks once limit is reached thus it can be used in unbounded tree

The space requirements for depth limited search are similar to depth first search. That is, O(*bl*).

The time complexity is O(*bl* )

**3.Iterative Deepening Search**

The problem with depth limited search is deciding on a suitable depth parameter. To overcome this problem there is another search called iterative deepening search.

This search method simply tries all possible depth limits; first 0, then 1, then 2 etc., until a solution is found.

What iterative deepening search is doing is combining breadth first search and depth first search. IDDFS calls DFS for different depths starting from an initial value. In every call, DFS is restricted from going beyond given depth. So basically we do DFS in a BFS fashion.

An important thing to note is, we visit top level nodes multiple times. The last (or max depth) level is visited once, second last level is visited twice, and so on. It may seem expensive, but it turns out to be not so costly, since in a tree most of the nodes are in the bottom level. So it does not matter much if the upper levels are visited multiple times.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Evaluation** | **Breadth First** | **Uniform Cost** | **Depth First** | **Depth Limited** | **Iterative Deepening** |
| Time | BD | BD | BM | BL | BD |
| Space | BD | BD | BM | BL | BD |
| Optimal? | Yes | Yes | No | No | Yes |
| Complete? | Yes | Yes | No | Yes, if L >= D | Yes |

**Few Snippets of the Output:**

A screenshot of a social media post

Description generated with very high confidence

A screenshot of a social media post

Description generated with very high confidence

A screenshot of a computer

Description generated with very high confidence

A screenshot of a computer

Description generated with very high confidence

A screenshot of a computer

Description generated with very high confidence

A screenshot of a computer

Description generated with very high confidence

A screenshot of a computer

Description generated with very high confidence