src/Algorithms/PartA_DFS.java

```
packageimportimportimportimportimportimportimportimportimportimport/**
 * This class implements the Depth-First Search (DFS) algorithm for graph
 * traversal.
 */publicclassPartA_DFS/**
     * Executes the Depth-First Search (DFS) from a start node to a goal
node within
     * a given planet size.
     *
     * @param@param@param@returnpublicstaticdfs(Node start, Node goal, int
newStacknewHashMapnewHashSetnullwhileNodecurrent=ifcontinueifreturnforinti=10
Nodenext=ifreturnnull/**
     * Prints the current state of the frontier (nodes to be visited).
     *
     * @paramprivatestaticvoidprintFrontier(Stack<Node> frontier)ifString
result=""forinti=10Nodenode=",""["01"]" Algorithms;

 java.util.Collections;
 java.util.Comparator;
 java.util.HashMap;
 java.util.Map;
 java.util.Set;
 java.util.HashSet;
 java.util.List;
 java.util.Stack;

 General.Node;
 General.Utility;


   {

     start      The starting node of the search.
     *  goal       The target node to find.
     *  planetSize The size of the planet, which influences the bounds of
the
     *                   search area.
     *  A list of nodes representing the path from the start to the goal if
     *         found, otherwise null.
     */</span>
      List<Node>  planetSize)</span> {
        Stack<Node> frontier =  <>();
        Map<Node, Node> parentMap =  <>();
        Set<Node> visited =  <>();

        frontier.push(start);
        parentMap.put(start, );

         (!frontier.isEmpty()) {
           printFrontier(frontier);
              frontier.pop();

            (visited.contains(current)) {
               ;
           }

           visited.add(current);
            (current.equals(goal)) {
               List<Node> path = Utility.constructPath(current,
parentMap);
                Utility.printPath(path, visited.size());
                 path;
            }

            List<Node> successors = current.getSuccessors(planetSize,
goal);
            Collections.sort(successors,
Comparator.comparingInt(Node::getD).thenComparingInt(Node::getAngle));

             (   successors.size() - ; i >= ; i--) {
                   successors.get(i);
                 (!visited.contains(next) && !frontier.contains(next)) {
                    frontier.push(next);
                    parentMap.put(next, current);
                }
            }
        }
        Utility.algorithmFails(visited.size());
         ;
    }

     frontier The stack containing the nodes currently in the frontier.
     */</span>
        {
         (!frontier.isEmpty()) {
               ;
             (   frontier.size() - ; i >= ; i--) {
                   frontier.get(i);
                result += node.toString() + ;

            }
            System.out.println( + result.substring(, result.length() - )
+ );
        }
    }
}
```