src/Algorithms/PartA_BFS.java

```
 1 packageimportimportimport/**
 2  * Implements the Breadth-First Search (BFS) algorithm for navigating
through a
 3  * graph.
 4  */publicclassPartA_BFS/**
 5      * Performs BFS to find the shortest path from the start node to the
goal node.
 6      *
 7      * @param@param@param@returnpublicstaticbfs(Node start, Node goal, int
newArrayDequenewHashMapnewHashSetnullwhileNodecurrent=ifreturnforifreturnnull
Algorithms;
 8
 9  java.util.*;
10
11  General.Node;
12  General.Utility;
13
14
15    {
16      start      The starting node of the path.
17      *  goal       The goal node of the path.
18      *  planetSize The size of the planet, used to limit the search area.
19      *  A list of nodes representing the path from start to goal if one
20      *         exists, or null if no path is found.
21      */</span>
22       List<Node>  planetSize)</span> {
23        Queue<Node> frontier =  <>();
24        Map<Node, Node> parentMap =  <>();
25        Set<Node> visited =  <>();
26
27        frontier.add(start);
28        parentMap.put(start, );
29
30         (!frontier.isEmpty()) {
31           Utility.printFrontier(frontier);
32              frontier.poll();
33
34           visited.add(current);
35            (current.equals(goal)) {
36              List<Node> path = Utility.constructPath(current,
parentMap);
37              Utility.printPath(path, visited.size());
38               path;
39           }
40
41           List<Node> successors = current.getSuccessors(planetSize,
goal);
42
43           Collections.sort(successors);
44
45            (Node next : successors) {
46                (!visited.contains(next) && !frontier.contains(next)) {
47                   frontier.add(next);
48                   parentMap.put(next, current);
49               }
50           }
51        }
52        Utility.algorithmFails(visited.size());
```

```
53          ;
54      }
55  }
```