INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

UNIVERSITY OF WESTMINSTER (UOW)

BEng (Hons) in Software Engineering

5SENG003C.2: Algorithms

Assignment title: Algorithms – Coursework (2021/22)

Module Leader: Klaus Draeger

Date of submission: 12/04/2022, 23:59

Student Name: Achintha Jayatilake

IIT Student ID: 2019530

UOW No: w1761374

Sliding Puzzle

Short Explanation of the choice of data structure and algorithm

When deciding on the correct data structure, it is necessary to select the most suitable data structure type based on the given requirement. So mainly in this, it used Ayyay, LinkedList, and Arraylist to manage and store data throughout the program.

When going through the codebase, it can be identified 2D Larry is the first data structure. During the text file conversion, it is using the 2D Array to store all the data sets. The main reason for using a 2D array is because it will be easier to store the data set as a 2D data set. With the use of a 2D array, it can navigate through the data using X and Y cardinalities easily. And there is no shifting between data, no quick fetching, and with that, the most suitable data structure for this scenario is 2D Array.

The next identified main data structure is Linked List. When finding the path from starting position to the end position it is necessary to connect each step sequentially. A linked List is a sequence of links that contain items. Each link contains a connection to another link. This is used to store Node points that are generated through the implementation.

With the use of Linked List it will be very easy to add updates and delete the list without much weight. When doing such operations it won't shift any data but only it will update the pointer. As this implementation contains multiple loops and some complex logic, it is mandatory to think about the performance. So the best data structure for this is this task is the Linked List. And also when defining the Linked List, it is not needed to define the size during the compilation time. And Linked List includes some inbuild important functions such as size, addLast, pollFirst and etc.

The logic mainly contains three parts and they are parser, mover, step finder. Before starting the game, it is needed to set the word file based on the game data map.

```
. . . . . 0 . . . S
. . . . 0 . . . . .
0 . . . . . 0 . . 0
. . . 0 . . . . 0 .
. F . . . . . . 0 .
. 0 . . . . . . . .
. . . . . . . 0 . .
. 0 . 0 . . 0 . . 0
0 . . . . . . . . .
. 0 0 . . . . . 0 .
```

Once the program is started, first it will read the text file through the parser and convert all 0s 1s, and "."s into program readable types. During the execution, it also tracks the end and starting X and Y cardinalities in a static field.

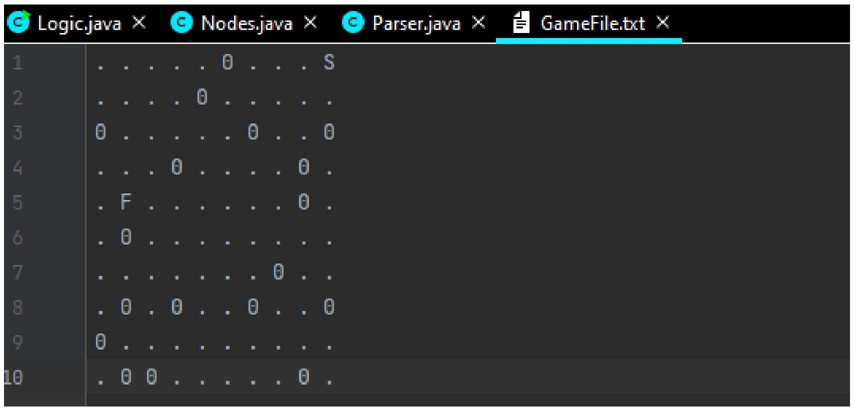
And then it will create the first Node point. Then it creates another 2D data array to store the temp data points.

Once the parser is finished, it will start to execute the generateSteps method and inside this method, it also executes the data mover logic. Mainly the mover logic is for going through all the available four cardinalities (X, -X, Y, -Y). It navigates in all directions from each and every point. And then we will be tracked all the possible successful forward movements. Likewise, it will go through all the possible paths and it will navigate until the last position is met which was captured during the parser execution. And all the possible paths will be saved on Linked List. Here the last part will be the steps generator. Once the endpoint is met, it will start to convert the data points to text format.

Test Samples

Sample 1

Input



```
. . . . . 0 . . . S
. . . . 0 . . . . .
0 . . . . . 0 . . 0
. . . 0 . . . . 0 .
. F . . . . . . 0 .
. 0 . . . . . . . .
. . . . . . . 0 . .
. 0 . 0 . . 0 . . 0
0 . . . . . . . . .
. 0 0 . . . . . 0 .
```

Output

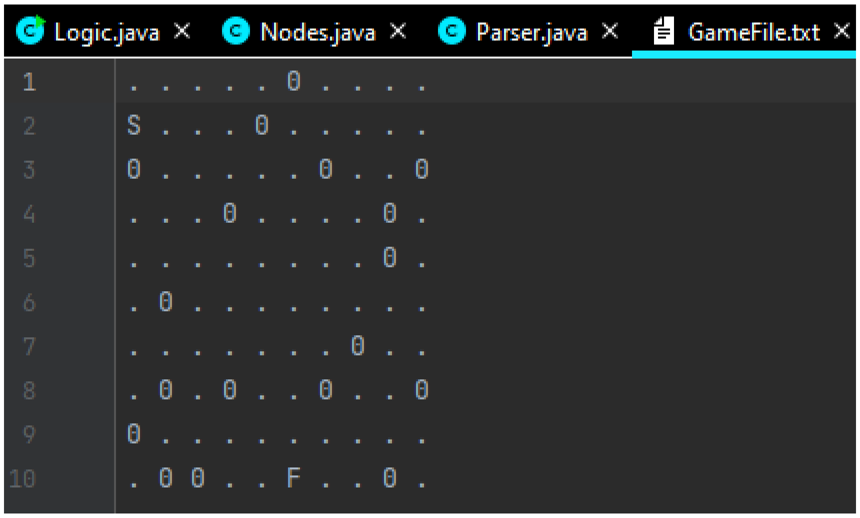
```
Steps for Game
-------------------------


Start at (10,1)
Move Down to (10,2)
Move Left to (6,2)
Move Down to (6,10)
Move Right to (8,10)
Move Up to (8,8)
Move Right to (9,8)
Move Up to (9,6)
Move Left to (3,6)
Move Up to (3,1)
Move Left to (1,1)
Move Down to (1,2)
Move Right to (4,2)
Move Down to (4,3)
Move Left to (2,3)
Move Down to (2,5)
Done


-------------------------
```

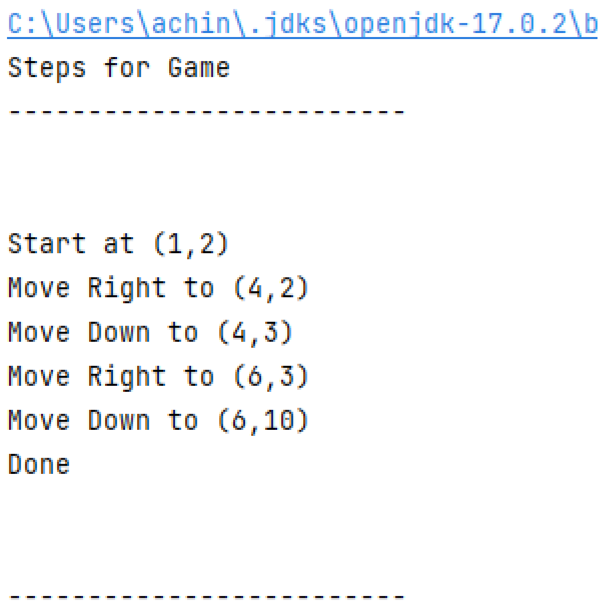
Sample 2

Input



```
. . . . . 0 . . . .
S . . . 0 . . . . .
0 . . . . . 0 . . 0
. . . 0 . . . . 0 .
. . . . . . . . 0 .
. 0 . . . . . . . .
. . . . . . . 0 . .
. 0 . 0 . . 0 . . 0
0 . . . . . . . . .
. 0 0 . . F . . 0 .
```

Output



```
C:\Users\achin\.jdks\openjdk-17.0.2\b
Steps for Game
-------------------------



Start at (1,2)
Move Right to (4,2)
Move Down to (4,3)
Move Right to (6,3)
Move Down to (6,10)
Done



-------------------------
```