Project Number: 3

Project Name: Rat in a Maze

Team Members[[1]](#footnote-2):

|  |  |  |
| --- | --- | --- |
| Grade | Team member name (**in Arabic**) | Team Member ID |
|  | رواء رجب محمود مغازى | 201900309 |
|  | احمد حاتم فتحى خضر | 201900022 |
|  | حلا تاج السر ربيع عبد العزيز | 201900274 |
|  | امنيه سيد حامد عبد الرحمن | 201900174 |
|  | جورج الامير مكرم سليم | 201900245 |
|  | مؤمن عماد الدين عبد الفتاح | 201900863 |

**Evaluation Criteria**

General Criteria

|  |  |  |
| --- | --- | --- |
| Criteria | | Grade |
| **Multithreading (5)** | No multithreading ( 2 out of 5 ) | **…………………………………………………………………..** |
| Threads in serial ( 3 out of 5)  Correct usage of threads, and synchronization mechanisms |
| Multithreading (4 or 5 out of 5)  Correct usage of threads, and synchronization mechanisms |
| **GUI (2)** | No GUI (0 out of 2) | **…………………………………………………………………..** |
| GUI without thread communication or realtime update (1 out of 2) |
| GUI with correct I/O and Thread communication or realtime update   (2 out of 2) |
| **Documentation (1)** |  |  |
| **Understanding (2)** |  |  |

رواء رجب محمود مغازى

Rat in a Maze Problem

A Maze is given as𝑁×𝑁binary matrix of blocks where source block is the upper left most block i.e., maze [0][0]and destination block is lower rightmost block i.e., maze[N-1] [N-1]. A rat starts from source and must reach the destination. The rat can move only in two directions: forward and down. In the maze matrix, 0means the block is a dead end and 1 means the block can be used in the path from source to destination. Use Multi-threading to solve this problem. You should design a multithreaded JAVA program with the following features:

- You should enter the dimensions of the maze, then a grid is generated.  
- You should use the grid to specify dead blocks on runtime.

We used Swing for GUI and depth first search algorithm to find the path from start (north-west) to end (south-east) regarding the dead blocks.  
Start frame opens the first panel which let user enter the dimensions of the grid Then create new frame with the grid as 2d array of JButtons. The buttons start with white color and the user click a button to make it dead block with black color. When click end, the DFS algorithm works to find the path with multithreading, maximum 1024 threads for the program and each thread checks a path, if found a path exit and return it to the third frame which views the path. Green for path, white for not used blocks and black for dead blocks. If the user clicks start, he will start the game again and if clicked end will exit the program.

|  |  |
| --- | --- |
| Member | Role |
| Ahmed Hatem | GUI (Second page) |
| Mo’men Emad | Algorithm |
| George Elamir | Algorithm |
| Rewaa Ragab | GUI (Third page) |
| Hala Tag Elser | GUI (First page) + documentation |
| Omnia Sayed | Algorithm |

**Regarding the algorithm that was used:**

The main idea was to use recursive depth first search to find a path from the starting point to the finish point, the maze was represented as a graph in the implemented code with each node having two, one or no children since we can only traverse to the right or to the bottom of the current node.

We used multithreading to help reach an answer faster while using the maximum capacities of the computers’ hardware, with the implementation we provide negatives of using multithreading were limited since there was no shared data between each thread that would affect the ability to find an answer.

**Code documentation:**

The algorithm is located in the DFS\_class.java file, it includes the global variables needed to find the answer with depth first search in mind:



We note that we made the maximum number of concurrent software threads to be 1024 for performance reasons.

The constructor for the class is responsible for initiating the global variables as well as starting the DFS operation, it includes a while loop that makes sure all possible path were taken before proceeding to output the answer.

The realDFS class is what we use for our multithreading, it is used as a recursive manner but with different set of values for every iterate and as long as we have not reached the maximum number of concurrent software threads then we are free to thread otherwise we use typical DFS methods of finding a path until a thread finish running then another uses its spot and so on until an answer is reached.



1. 1st team member should be the same one in project schedule

   إسم العضو الأول في الفريق يجب أن يكون نفس الاسم المعلن في جدول المناقشة [↑](#footnote-ref-2)