Maze Searcher

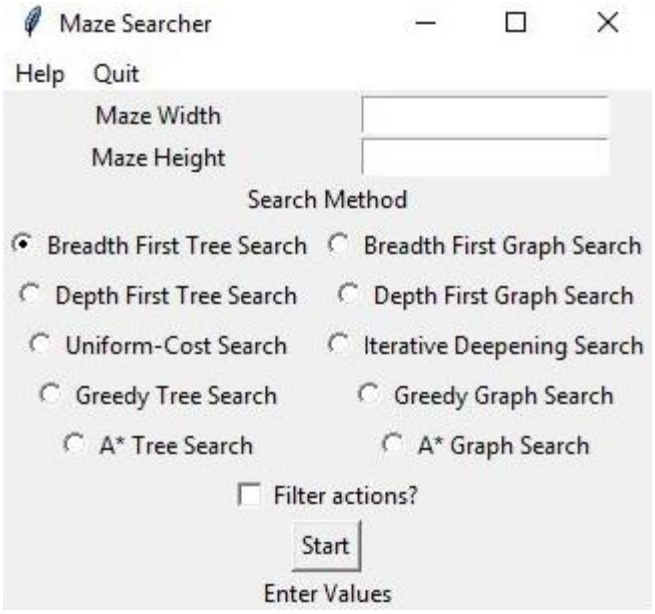
This application is used to test different search algorithms on mazes. The maze generator is forked from a repository from ryukinix on github.

Dependencies

It uses the following import commands:

- Os
- Tkinter
- Webbrowser
- Pygame
- Random
- Time
- Itertools

Using the Application



To use the application, specify a width and height for the maze. Then select which algorithm to use to find the route. Clicking the filter actions checkbox would mean that only actions that are possible are tested. If using informed tree searches like greedy and A* it is best to use filter actions to avoid lots of deadlocks, but you should use graph searches to avoid loops for the informed searches.

Once happy with the specifications press the start button to start finding the solution. A black box will open, and the maze will be displayed on it. While the title of the application is “wait for the application to find a solution” do not try to interact with the application. Once it says “Press space to start the route” then you can press space bar and the green box will make its way to the red square. The title will change to “Following route”. To avoid crashes, try to leave the game open and not interact with anything else.

The size of the maze, time and which method ID you used (1 being breadth first tree search, 2 being breadth first graph search, 3 being depth first tree search and so on) and whether you used filtering of actions (0 if no, 1 if yes) are recorded in the results text file.

Informed graph searches will typically perform the best, so if you want to find a solution to a big maze use one of them. If using a uniformed search to find a big maze it may take some time. To speed up try using filtering of actions and the graph search.

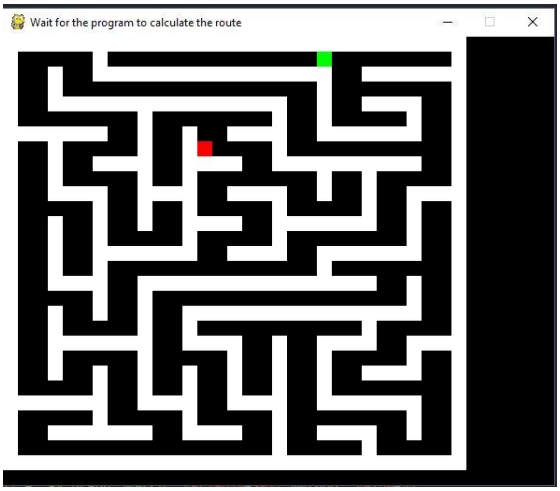


Figure 1 Application finding the route

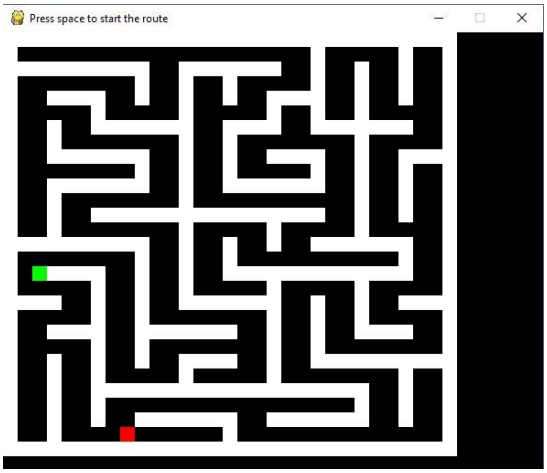


Figure 2 Application calculating the route

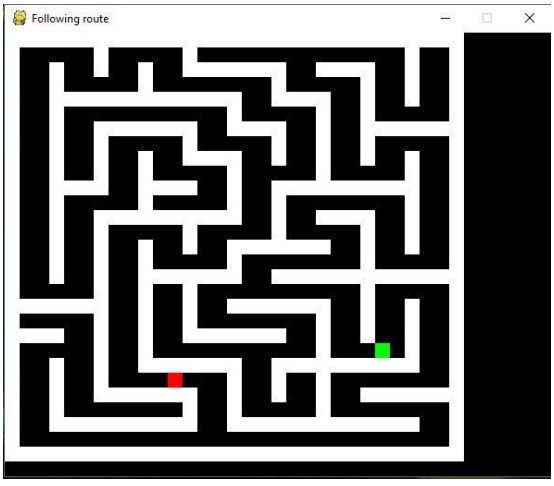


Figure 3 Applicaiton following the route