2023 Spring CSE222 Homework8 Report

Dijkstra and BFS algorithms are used to find a path between two points on a graph.

First, let's look at complexity analysis:

Dijkstra's Algorithm: Dijkstra's algorithm has a worst-case time complexity of O((V+E) log V) when using a priority queue, where V is the number of nodes and E is the number of edges. However, this algorithm provides a solution for the shortest path problem in weighted graphs. Dijkstra calculates the shortest path from one starting point to all points in the rest of the graph.

Breadth-First Search (BFS): The BFS algorithm operates at O(V + E) time complexity, usually using a queue data structure. However, BFS only provides a solution for the shortest path problem in unweighted graphs. BFS also calculates the shortest path from the starting point to all points in the rest of the graph.

The time complexity of both algorithms depends on the number of nodes and edges, but Dijkstra's algorithm is generally slower than BFS because the priority queue requires an extra logarithmic factor to find the least weighted edge. However, while Dijkstra's algorithm can work on weighted graphs, BFS provides a valid solution only on unweighted graphs.

On the other hand, both algorithms can be used to find the shortest path from one starting node to all other nodes, but if you just want to find the shortest path to a particular destination, BFS is usually faster because it can stop when it reaches the destination. Dijkstra's algorithm, on the other hand, cannot stop when it reaches the destination because there may be a shorter path on another path.

Consequently, when deciding which algorithm to use, we must consider the characteristics of the graph (weighted or unweighted), the need to solve a particular problem, and performance requirements.

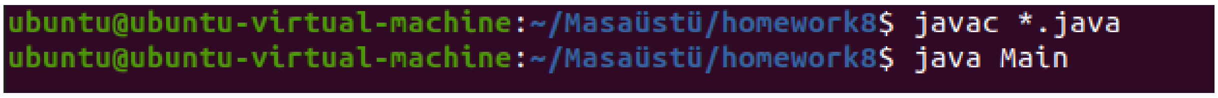
Comment: In the tests, the performances of Dijkstra and BFS algorithms on different maps were compared. When a comparison was made on the calculation times and path lengths of the Dijkstra and BFS algorithms; From a path length perspective, it seems that both algorithms find the same path length for all maps. This shows that both algorithms give accurate and consistent results.

However, in terms of computation time, there is a clear difference between both algorithms. For example, for the map10.txt map, the Dijkstra algorithm gave results in 1381 ms, while the BFS algorithm gave results in 1249 ms, which is faster.

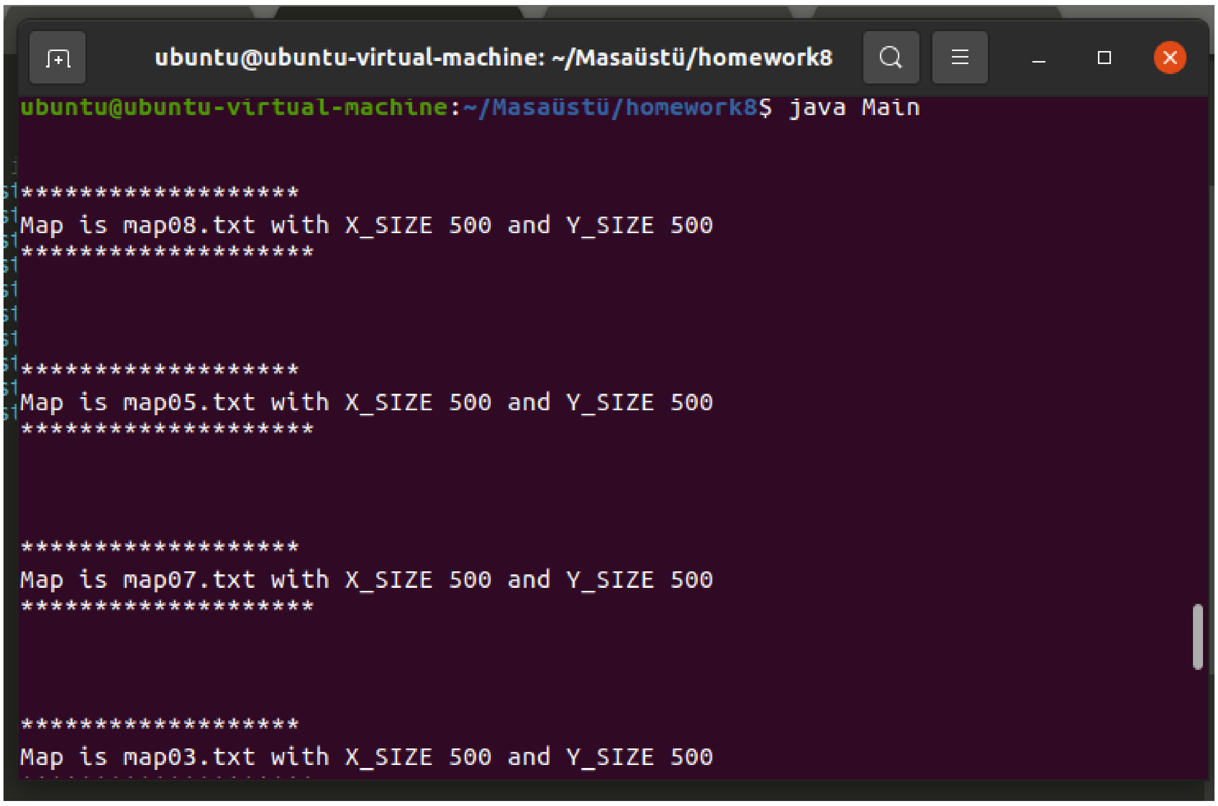
A similar situation applies to all other maps. Dijkstra's algorithm usually takes longer than the BFS algorithm. This is because Dijkstra's algorithm takes more time to calculate the shortest path for each node. In contrast, the BFS algorithm performs a wider search and is generally faster.

In summary, when choosing between BFS and Dijkstra algorithms, it is important to consider factors such as path length and computation time. In general, if computation time is an important factor, the BFS algorithm will usually be the faster option. However, since both algorithms give the same path length, neither algorithm is superior to the other in terms of path length.

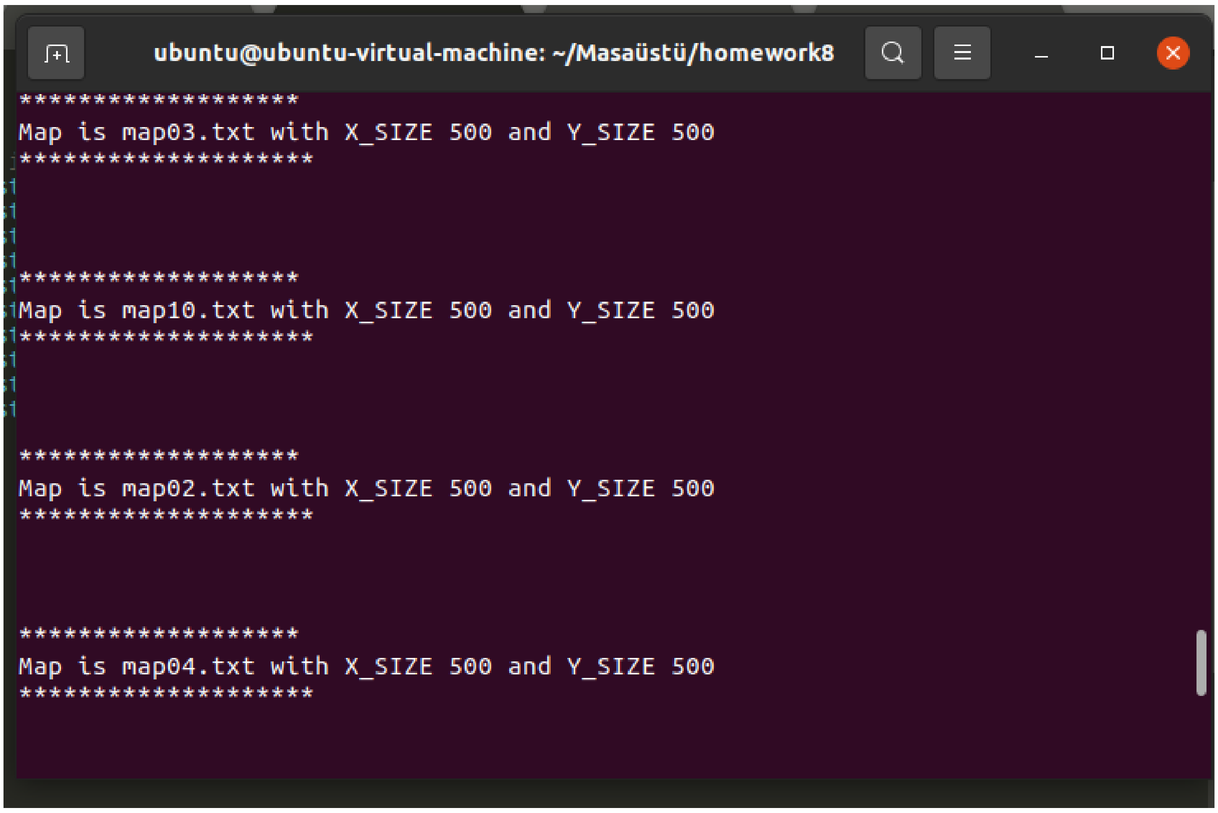
Running Command and Test Results with PNG Files



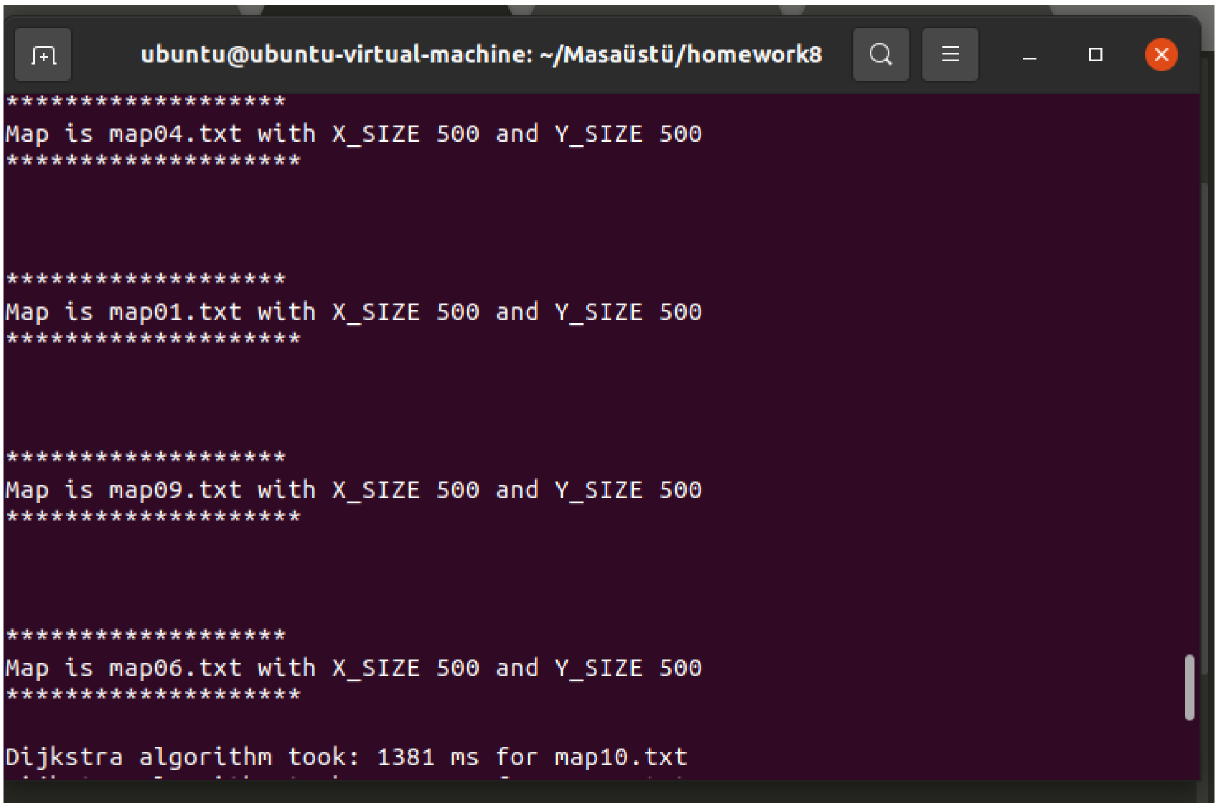
```
ubuntu@ubuntu-virtual-machine:~/Masaüstü/homework8$ javac *.java
ubuntu@ubuntu-virtual-machine:~/Masaüstü/homework8$ java Main
```



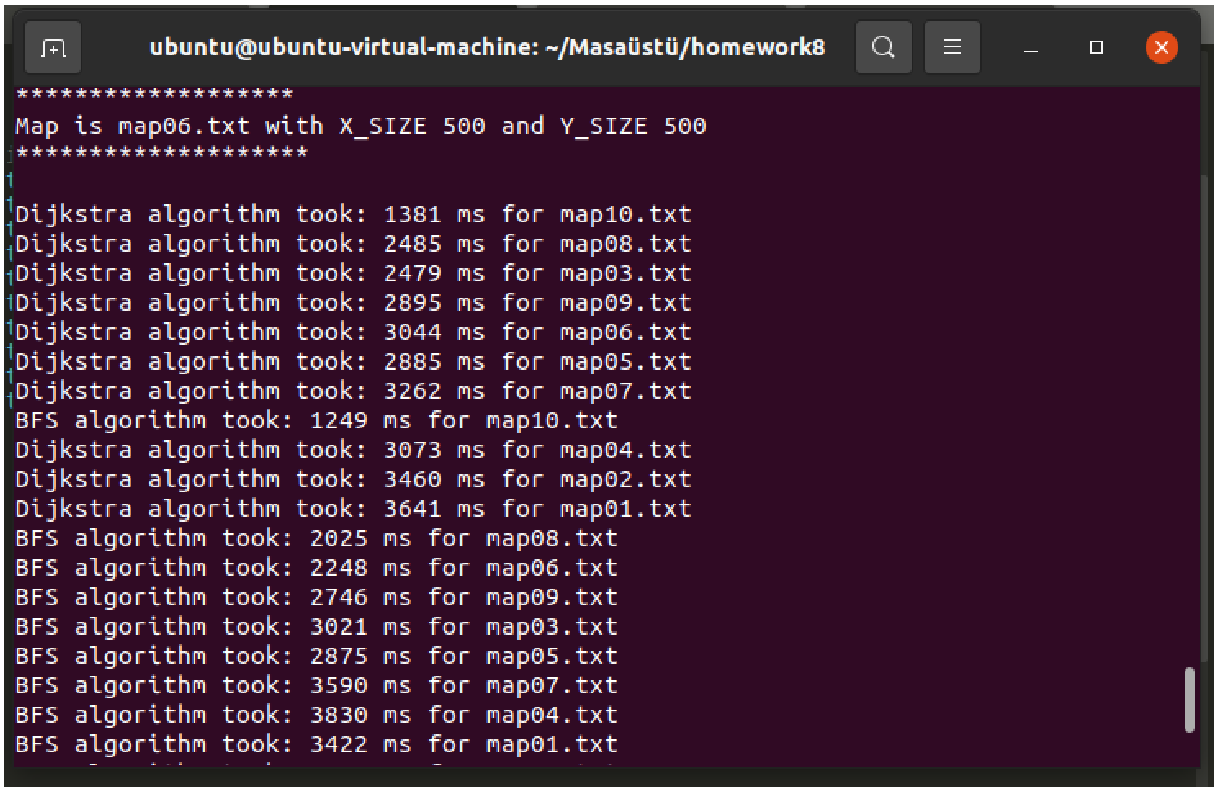
```
ubuntu@ubuntu-virtual-machine:~/Masaüstü/homework8$ java Main


*******************
Map is map08.txt with X_SIZE 500 and Y_SIZE 500
********************



*******************
Map is map05.txt with X_SIZE 500 and Y_SIZE 500
********************



*******************
Map is map07.txt with X_SIZE 500 and Y_SIZE 500
********************



*******************
Map is map03.txt with X_SIZE 500 and Y_SIZE 500
```



```
*******************
Map is map03.txt with X_SIZE 500 and Y_SIZE 500
********************




*******************
Map is map10.txt with X_SIZE 500 and Y_SIZE 500
********************




*******************
Map is map02.txt with X_SIZE 500 and Y_SIZE 500
********************




*******************
Map is map04.txt with X_SIZE 500 and Y_SIZE 500
********************
```

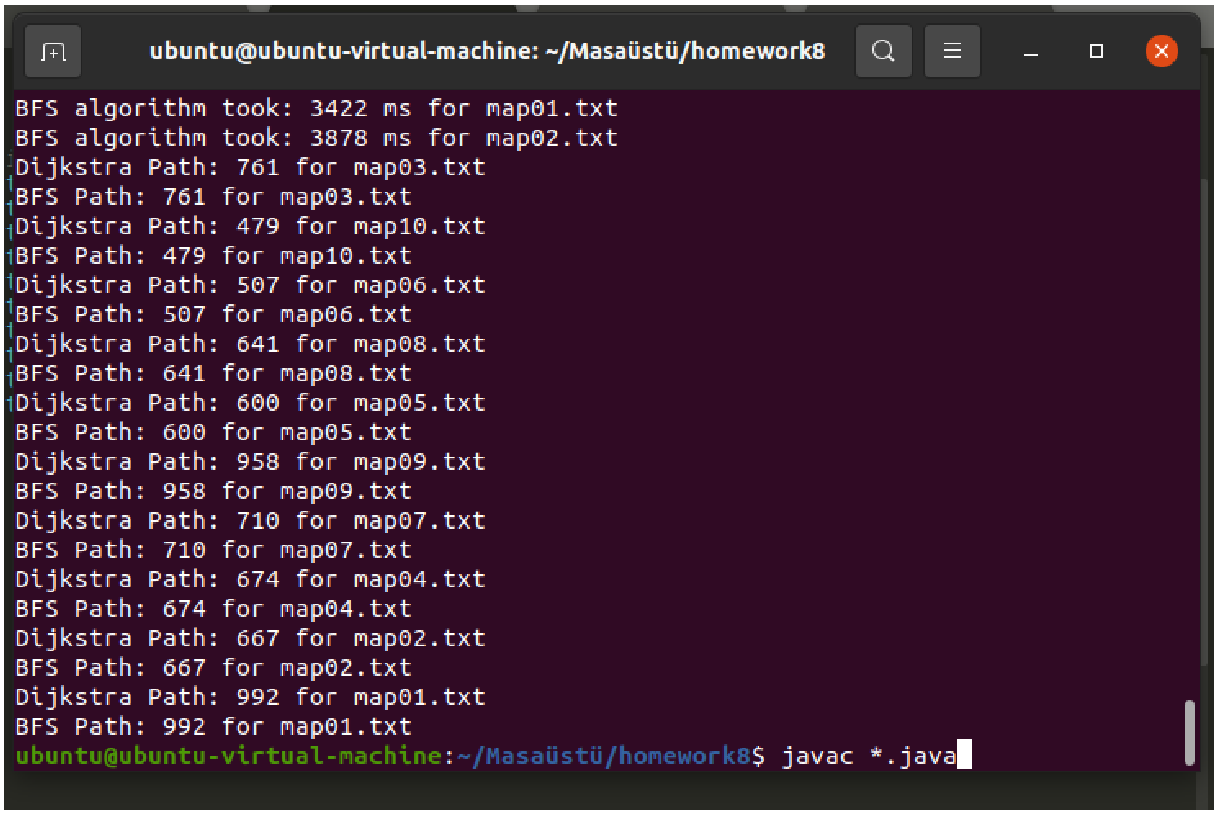


```
*******************
Map is map04.txt with X_SIZE 500 and Y_SIZE 500
********************




*******************
Map is map01.txt with X_SIZE 500 and Y_SIZE 500
********************




*******************
Map is map09.txt with X_SIZE 500 and Y_SIZE 500
********************




*******************
Map is map06.txt with X_SIZE 500 and Y_SIZE 500
********************

Dijkstra algorithm took: 1381 ms for map10.txt
```



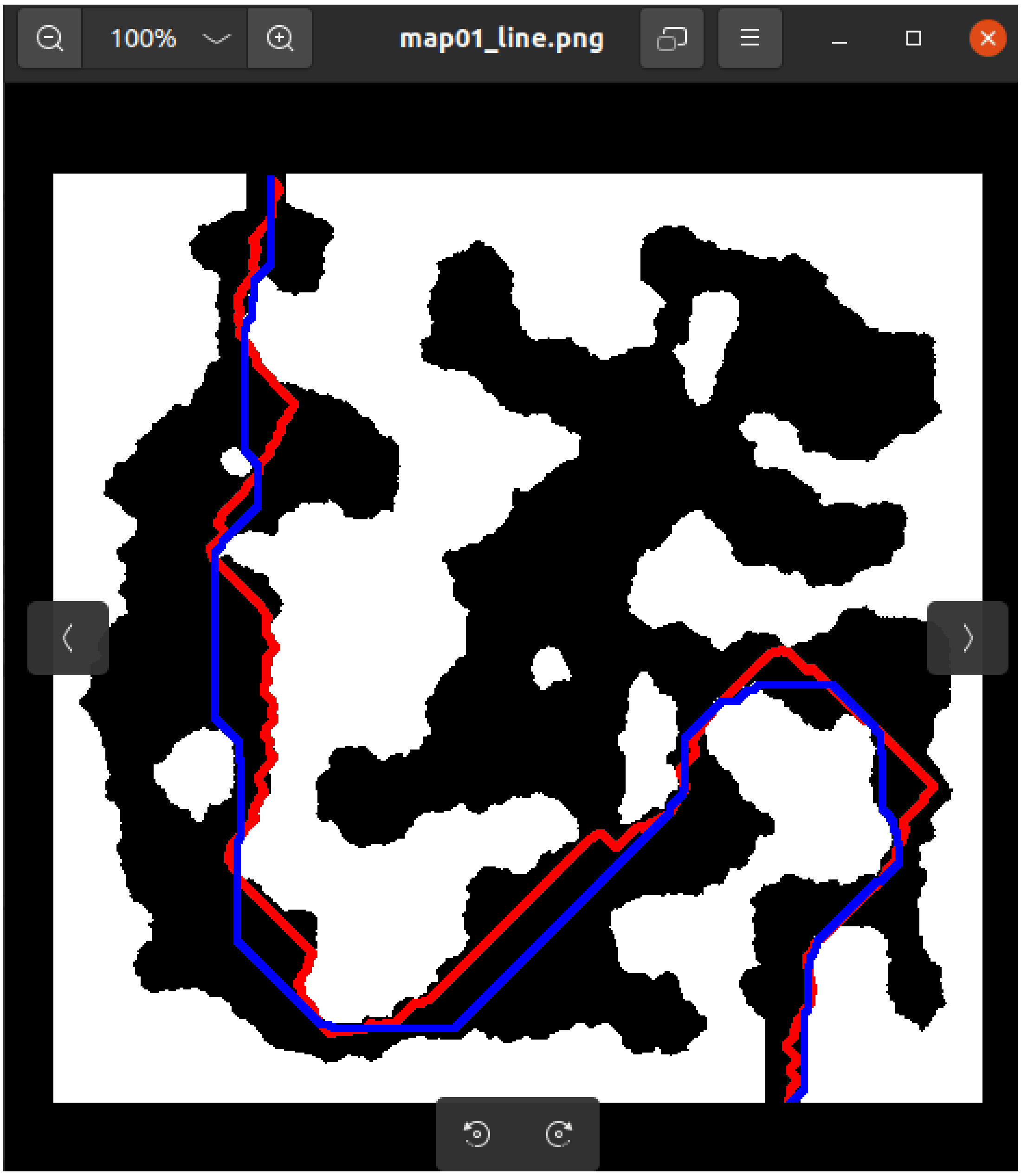
ubuntu@ubuntu-virtual-machine: ~/Masaüstü/homework8

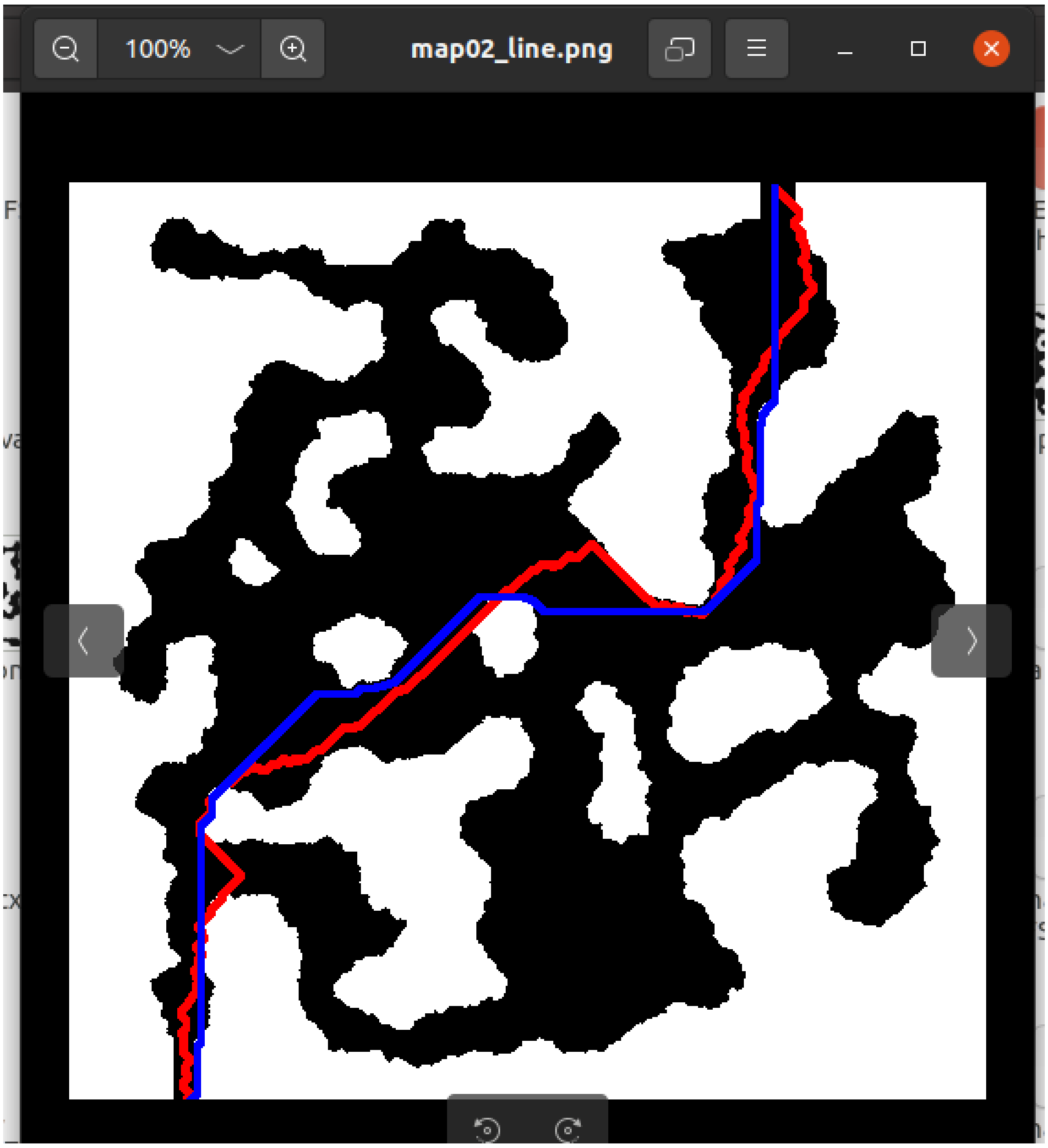
```
*******************
Map is map06.txt with X_SIZE 500 and Y_SIZE 500
********************

Dijkstra algorithm took: 1381 ms for map10.txt
Dijkstra algorithm took: 2485 ms for map08.txt
Dijkstra algorithm took: 2479 ms for map03.txt
Dijkstra algorithm took: 2895 ms for map09.txt
Dijkstra algorithm took: 3044 ms for map06.txt
Dijkstra algorithm took: 2885 ms for map05.txt
Dijkstra algorithm took: 3262 ms for map07.txt
BFS algorithm took: 1249 ms for map10.txt
Dijkstra algorithm took: 3073 ms for map04.txt
Dijkstra algorithm took: 3460 ms for map02.txt
Dijkstra algorithm took: 3641 ms for map01.txt
BFS algorithm took: 2025 ms for map08.txt
BFS algorithm took: 2248 ms for map06.txt
BFS algorithm took: 2746 ms for map09.txt
BFS algorithm took: 3021 ms for map03.txt
BFS algorithm took: 2875 ms for map05.txt
BFS algorithm took: 3590 ms for map07.txt
BFS algorithm took: 3830 ms for map04.txt
BFS algorithm took: 3422 ms for map01.txt
```

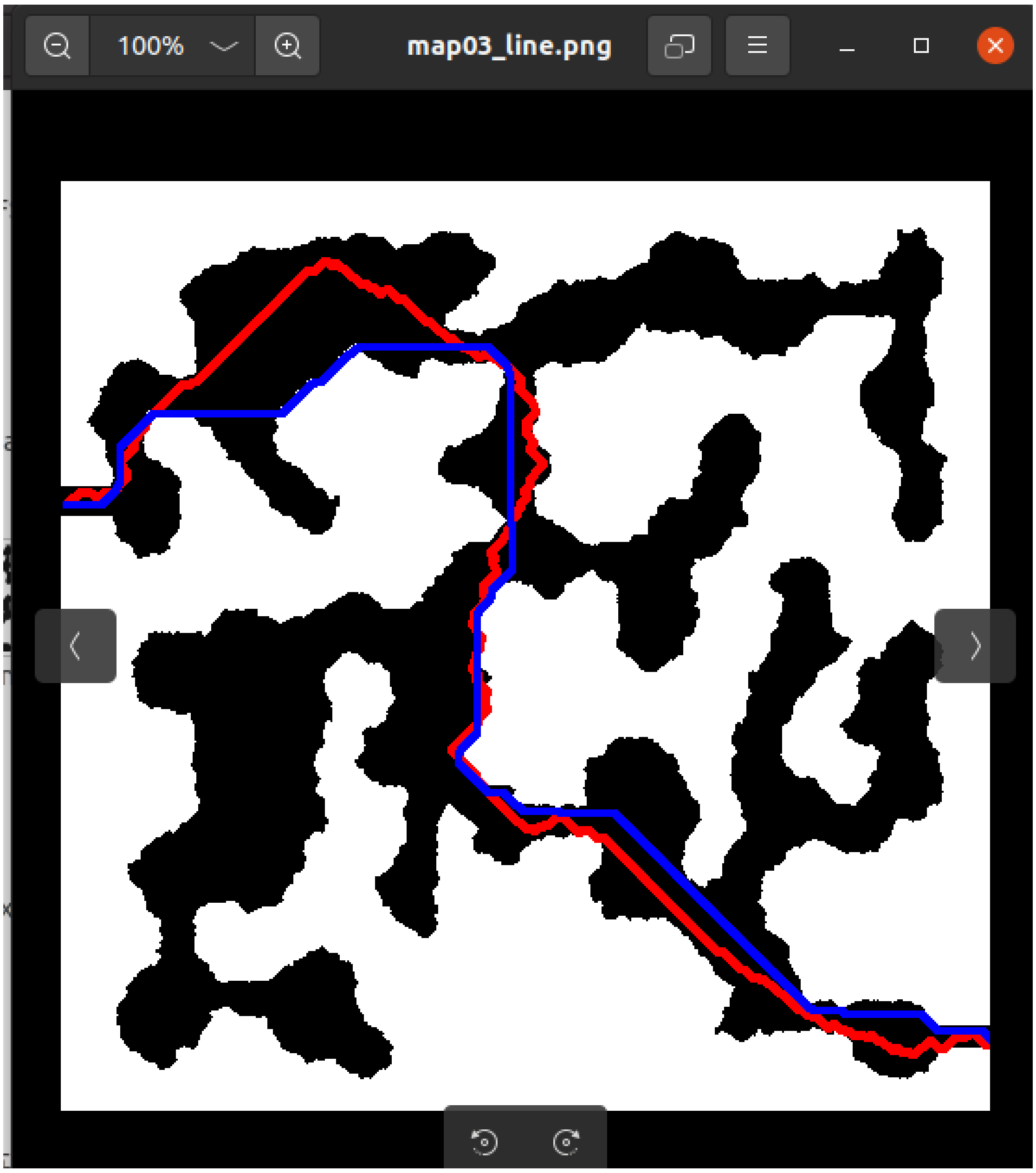


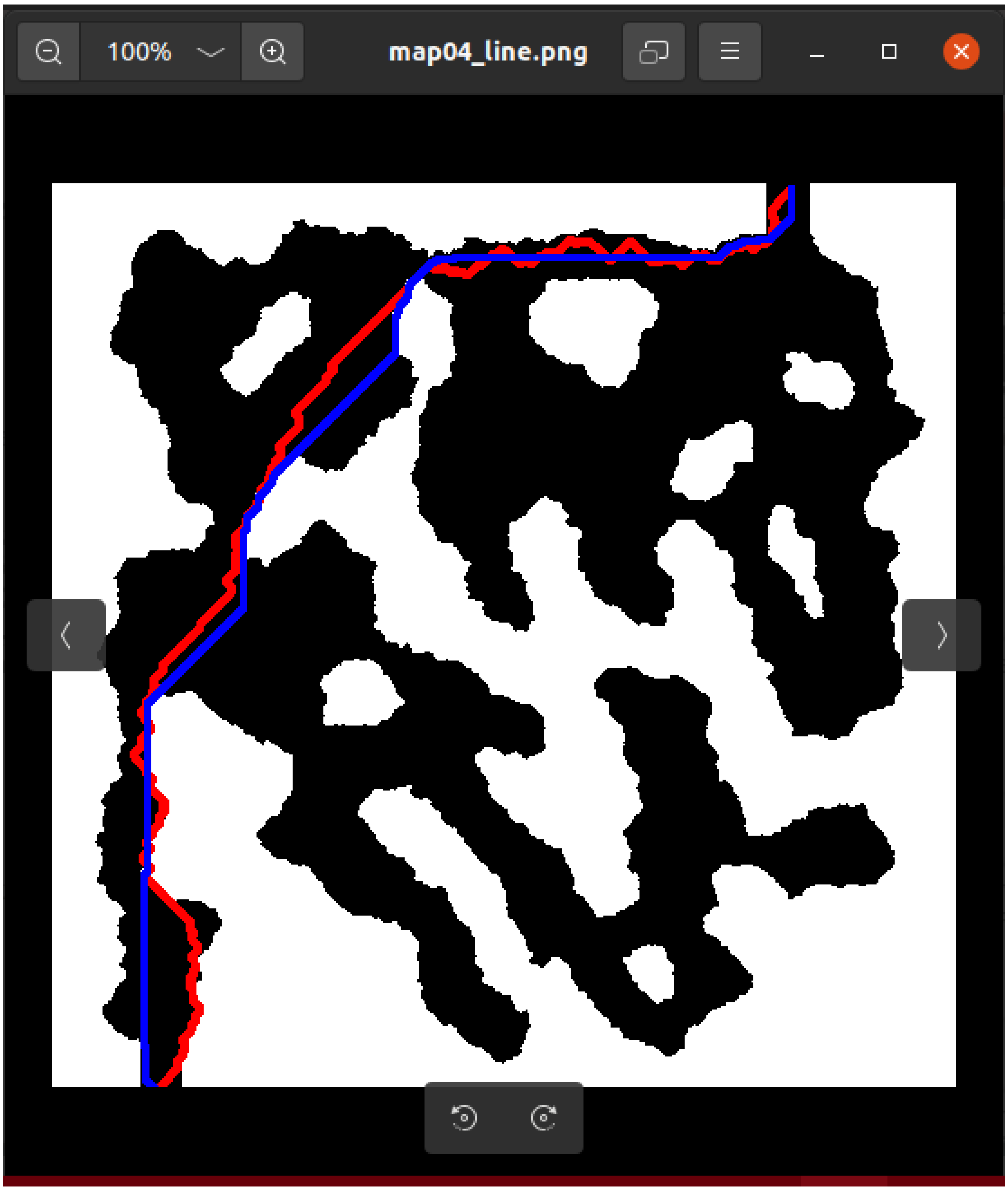
ubuntu@ubuntu-virtual-machine: ~/Masaüstü/homework8

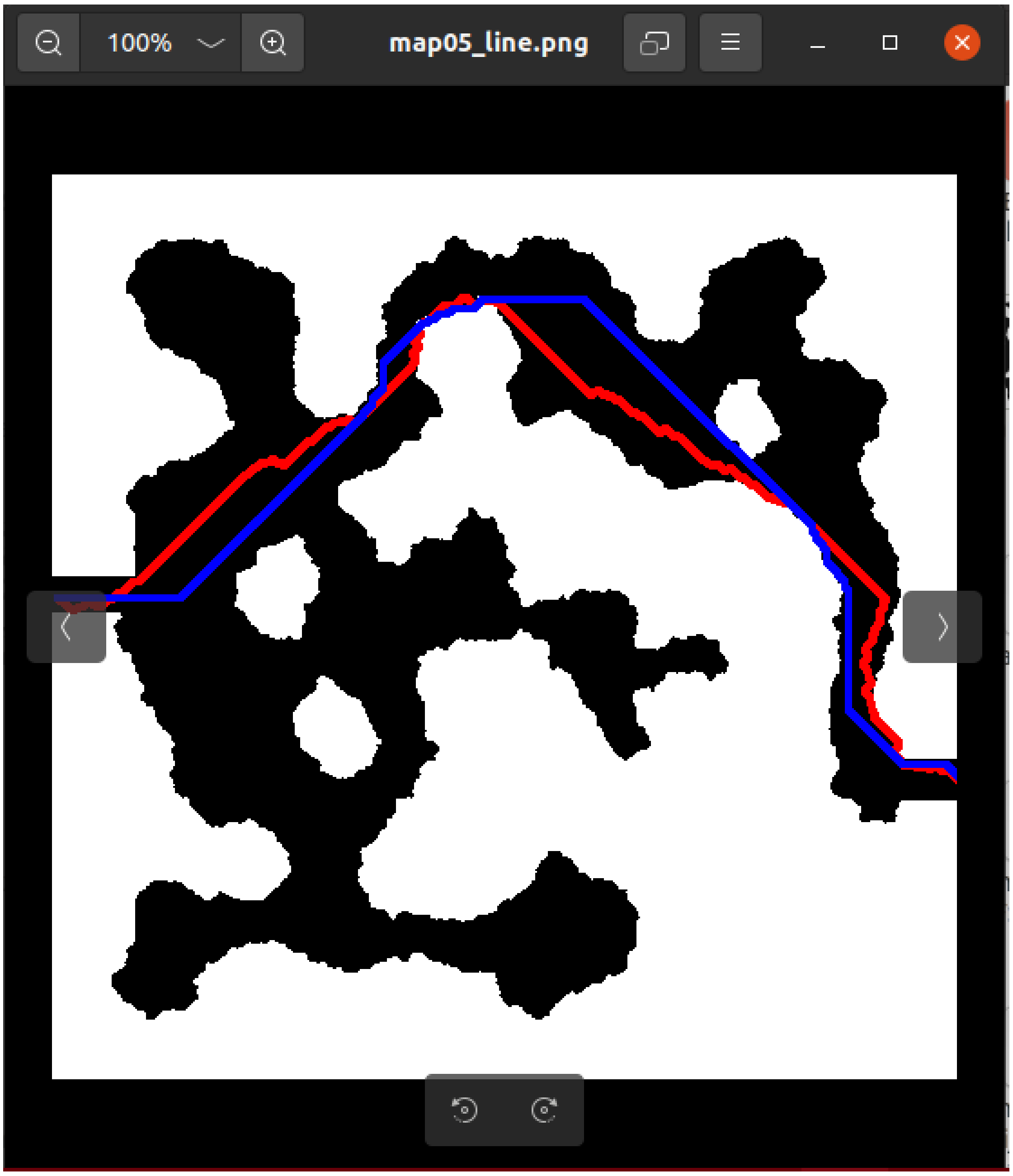
```
BFS algorithm took: 3422 ms for map01.txt
BFS algorithm took: 3878 ms for map02.txt
Dijkstra Path: 761 for map03.txt
BFS Path: 761 for map03.txt
Dijkstra Path: 479 for map10.txt
BFS Path: 479 for map10.txt
Dijkstra Path: 507 for map06.txt
BFS Path: 507 for map06.txt
Dijkstra Path: 641 for map08.txt
BFS Path: 641 for map08.txt
Dijkstra Path: 600 for map05.txt
BFS Path: 600 for map05.txt
Dijkstra Path: 958 for map09.txt
BFS Path: 958 for map09.txt
Dijkstra Path: 710 for map07.txt
BFS Path: 710 for map07.txt
Dijkstra Path: 674 for map04.txt
BFS Path: 674 for map04.txt
Dijkstra Path: 667 for map02.txt
BFS Path: 667 for map02.txt
Dijkstra Path: 992 for map01.txt
BFS Path: 992 for map01.txt
ubuntu@ubuntu-virtual-machine:~/Masaüstü/homework8$ javac *.java
```

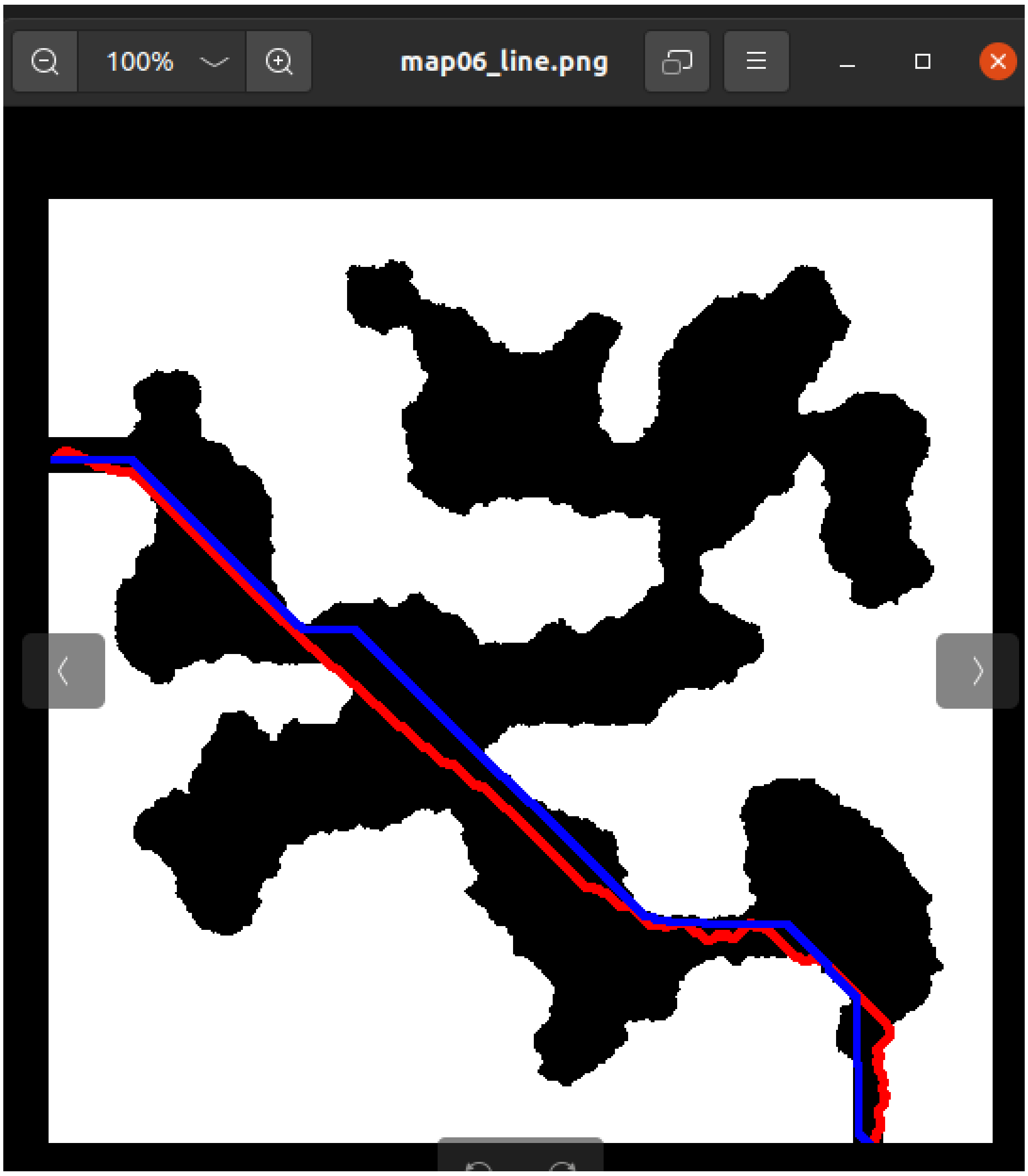

100%
map01_line.png

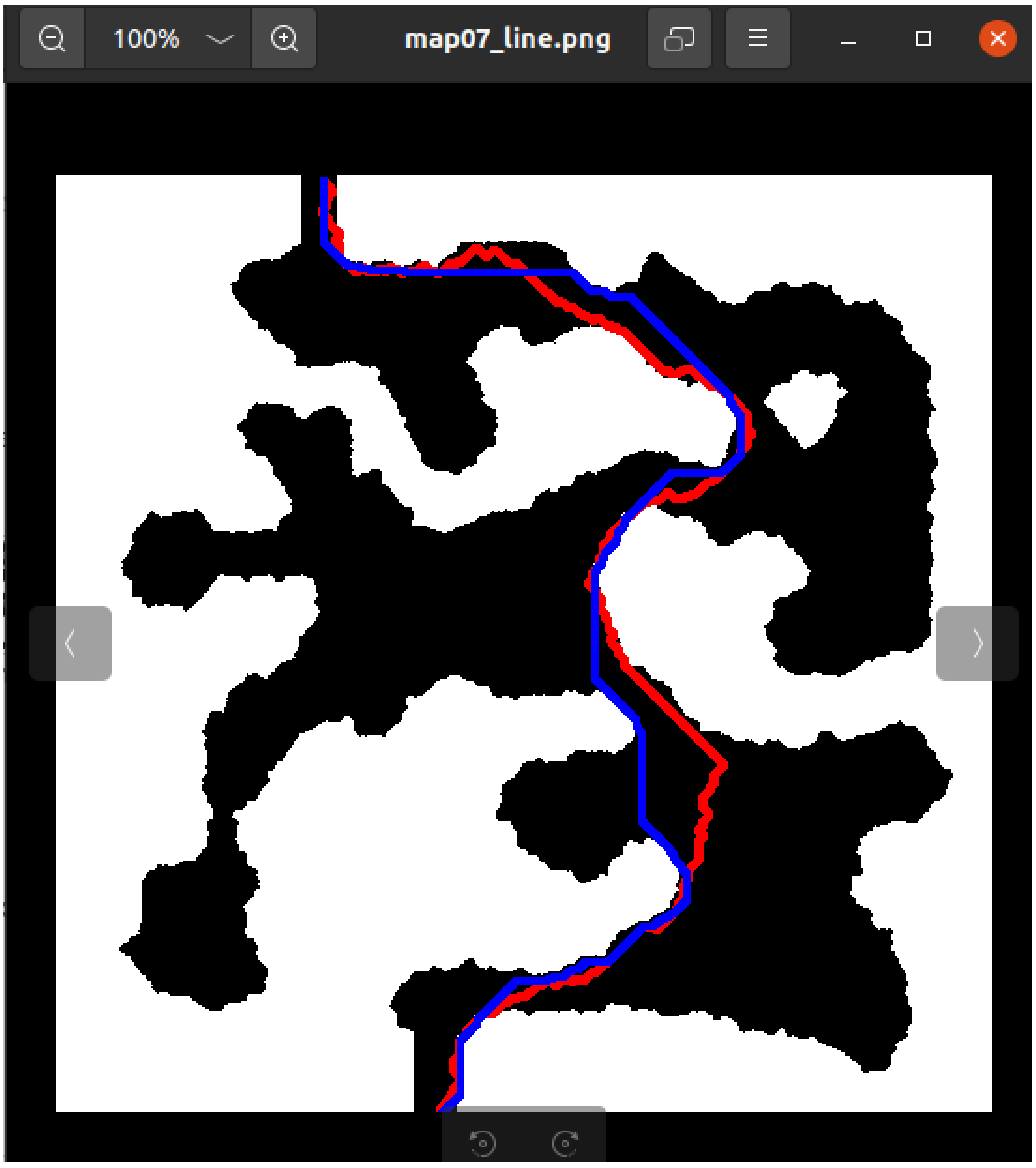

100%
map02_line.png

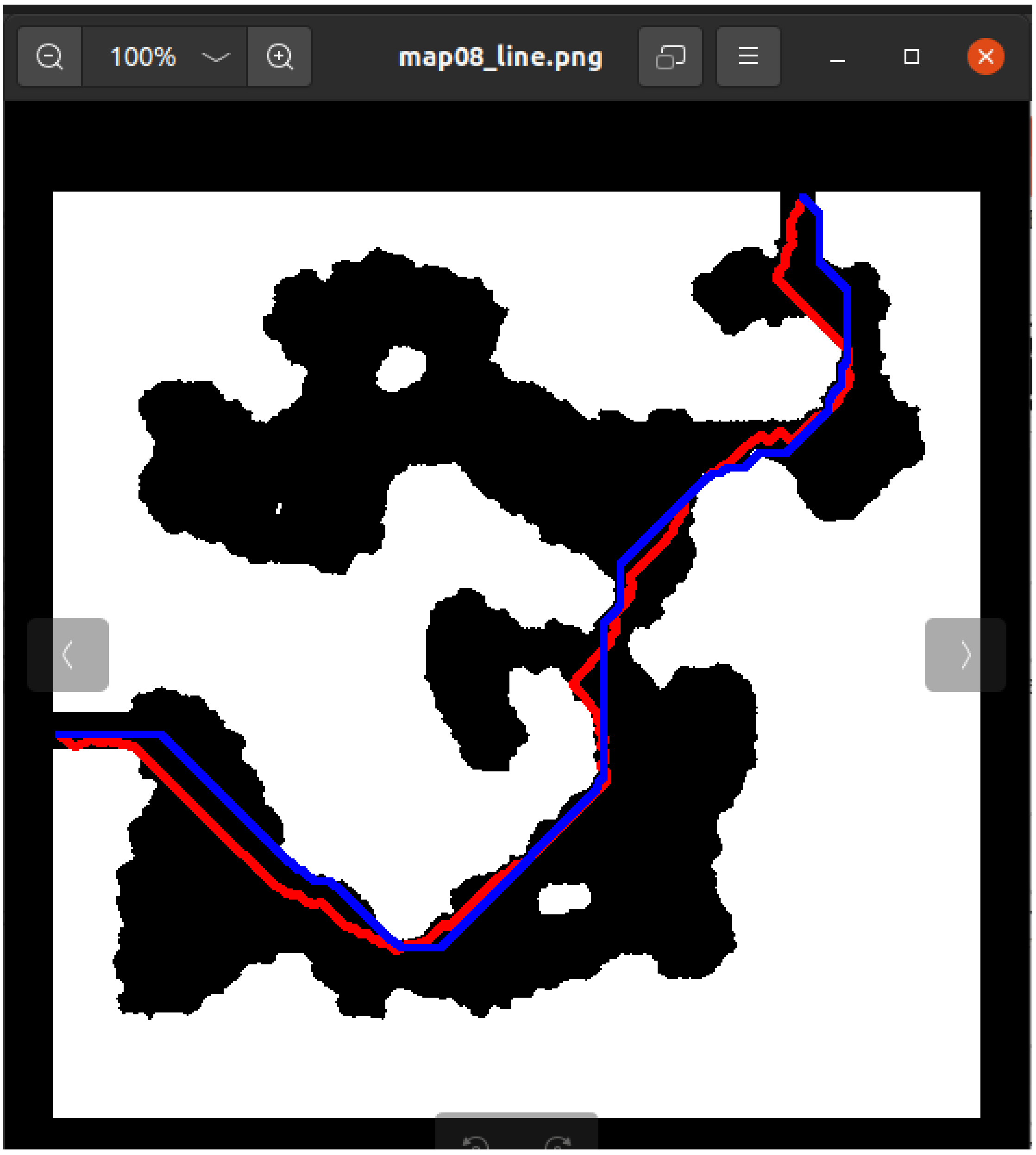

100%
map03_line.png

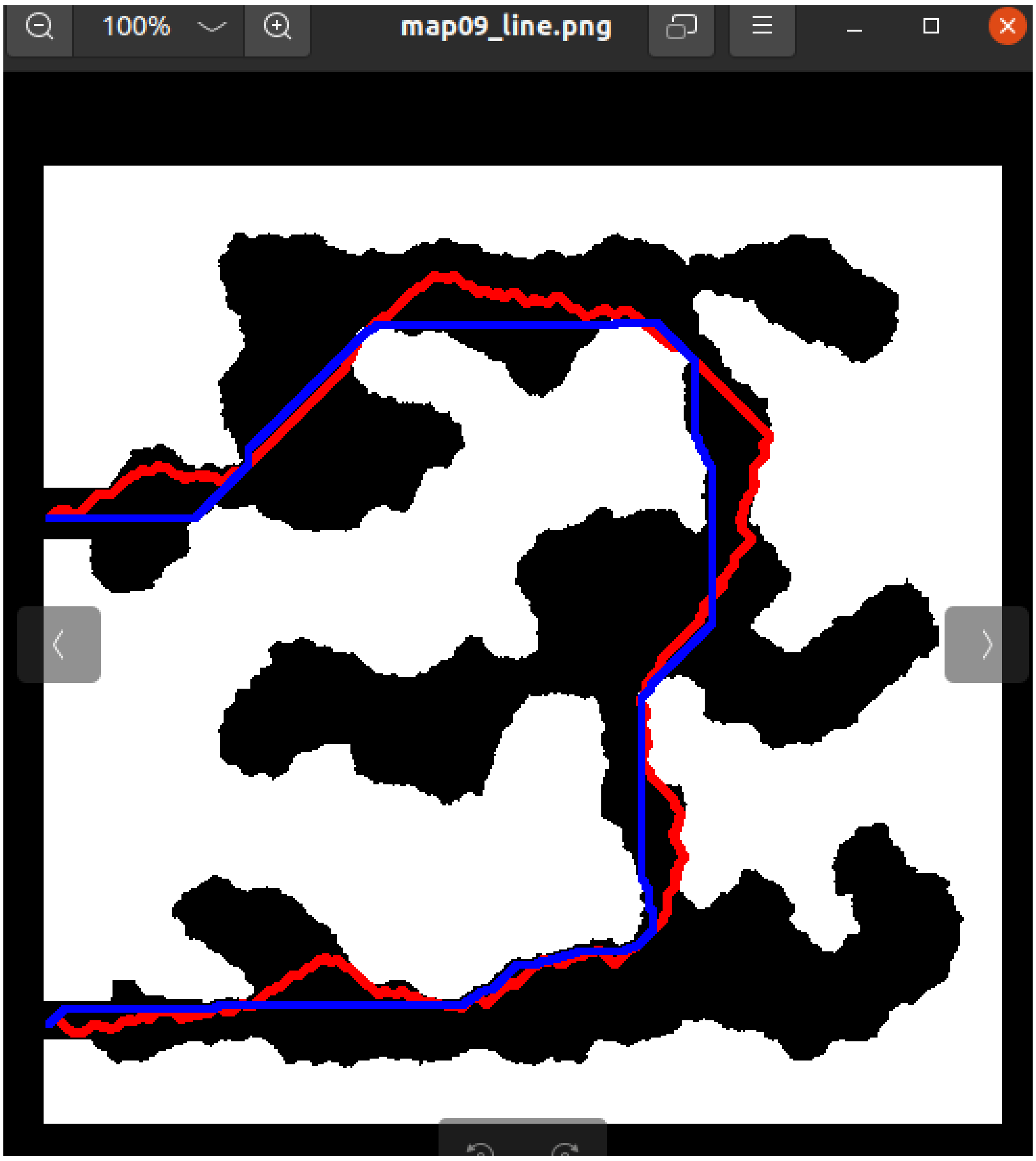

100%
map04_line.png

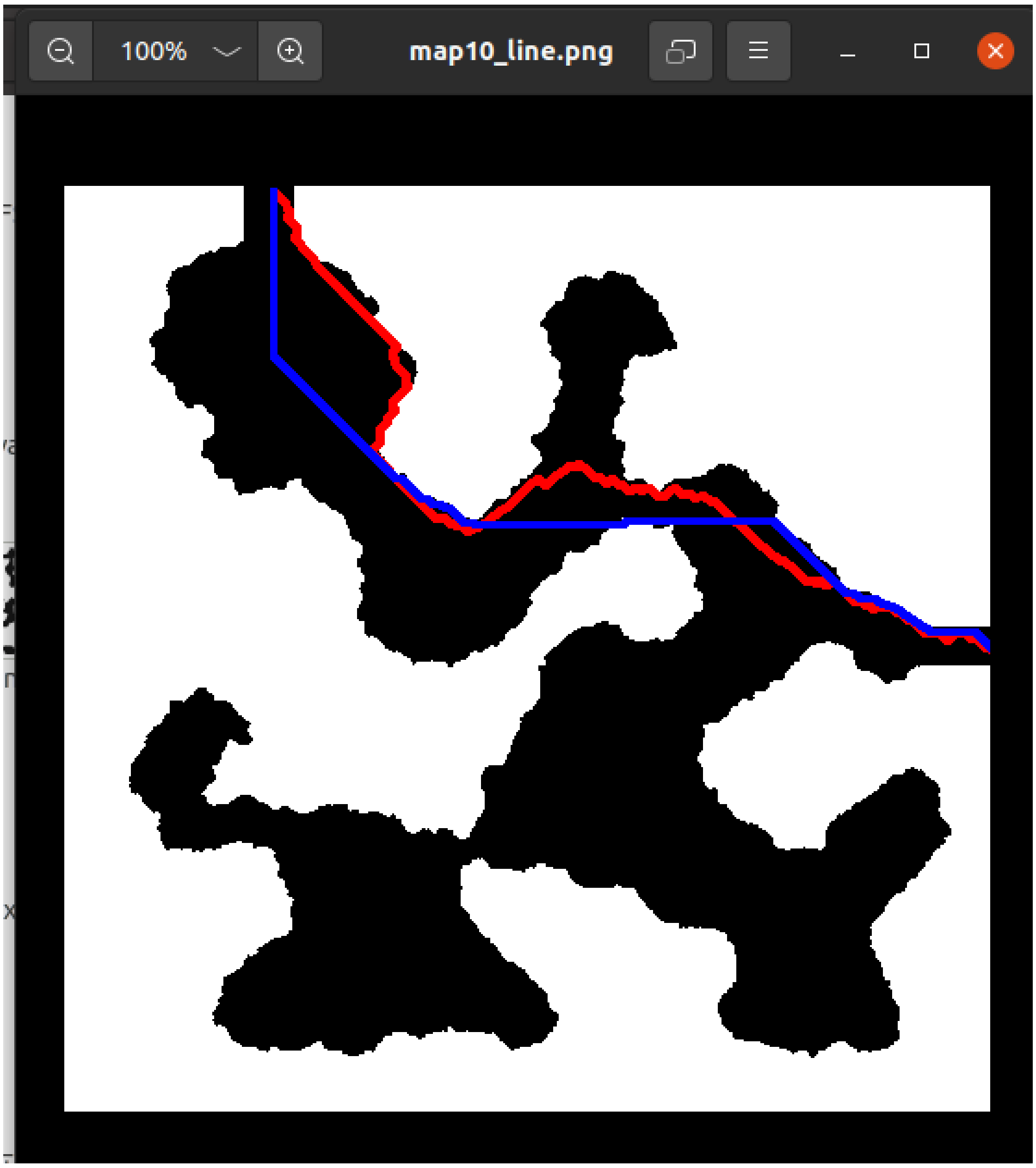

100%
map05_line.png


100%
map06_line.png


100%
map07_line.png


100%
map08_line.png


100%
map09_line.png


100%
map10_line.png