Lab 8 documentation

Problem: Given a directed graph, find a Hamiltonian cycle, if one exists. Use multiple threads to parallelize the search.

The algorithm used to solve this problem starts from the first vertex in the graph and then goes to all his neighbors, and each neighbor goes to all its neighbors that are not already in the resulted path and so on until the length of the path has its length equal to the number of vertices and a vertex has as its neighbor the first vertex, or a solution was not found and all possible paths were searched. Each time we want to search on a new path a new thread is created. Given that we perform search on a graph there was no need for synchronization, but I used the graph as a volatile variable to ensure that the algorithm does not search same path twice.

Performance measurements:

1. Graph with 8 vertices, time: 202 ms.

0 1 0 1 1 0 0 0

1 0 1 0 0 1 0 0

0 1 0 1 0 0 1 0

1 0 1 0 0 0 0 1

1 0 0 0 0 1 0 1

0 1 0 0 1 0 1 0

0 0 1 0 0 1 0 1

0 0 0 1 1 0 1 0

2. Graph with 15 vertices, time: 11721 ms.

0 1 0 1 1 0 0 0 1 0 1 1 0 0 0

1 0 1 0 0 1 0 0 0 1 0 0 1 0 0

0 1 0 1 0 0 1 0 1 0 0 0 1 1 0

1 0 1 0 0 0 0 1 0 0 0 0 1 0 1

1 0 0 0 0 1 0 1 0 1 1 1 0 0 0

0 1 0 0 1 0 1 0 0 0 0 0 0 0 1

0 0 1 0 0 1 0 1 1 0 0 1 0 0 1

0 0 0 1 1 0 1 0 0 0 1 0 1 0 1

1 0 1 0 0 0 1 0 0 0 0 1 0 0 0

0 1 0 0 1 0 0 0 0 0 0 0 0 0 0

1 0 0 0 1 0 0 1 0 0 0 0 0 0 0

1 0 0 0 1 0 1 0 1 0 0 0 1 0 1

0 1 1 1 0 0 0 1 0 0 0 1 0 1 0

0 0 1 0 0 0 0 0 0 0 0 0 1 0 0

0 0 0 1 0 1 1 1 0 0 0 1 0 0 0