# My Project

Generated by Doxygen 1.9.5

# Chapter 1

# INF205 Paths in labelled graphs

## 1.1 Programming project in INF205 Resource-efficient programming at NMBU

The point of the project is to take in a labelled graph *g* and two sequences *p* and *q* of edge labels and check if there exist a path *p* and a path *q* that contains the same start node and end node.

## 1.2 The code

### 1.2.1 src

The folder containing the whole directory

**1.2.1.0.1 main.cpp** You run this code and it will run the most important file for comparing the paths. Running Makefile in src-folder is better.

**1.2.1.0.2 comparing-paths.cpp** Compares the paths that is read from results.dat

**1.2.1.0.3 time-comparing-paths.cpp** Measures the run time for comparing-paths.cpp nevn hvordan den funker og hva som blir tatt tid på

#### 1.2.1.1 results

Folder containing all the different results

- Results with different amounts of paths found from 2^4 to 2^13
- Results with different amount of equal paths found: 0%, 25%, 50%, 75% and 100%
- plot-timing.py - code for plotting the results
- 6 different plots saved as png
- 5 different .dat-files with the time it took to benchmark-testing
- kbt.dat and queryt.dat which contains data used to benchmark run-graph.cpp

#### 1.2.1.2 directed-graph

Folder containing the code for creating the graph and finding valid paths to use in comparing-paths

##### 1.2.1.2.1 graph-benchmark.cpp

##### 1.2.1.2.2 query.cpp

##### 1.2.1.2.3 graph.cpp

##### 1.2.1.2.4 run-graph.cpp

run-graph.cpp

##### 1.2.1.2.5 time-run-graph.cpp

#### 1.2.1.3 threading_pkg

##### 1.2.1.3.1 src

**threading-paths.cpp**

## 1.3 How does it work and how to use it

## 1.4 Performance

#### 1.4.0.1 Run time for run-graph.cpp with different amount of nodes in the graph

First the time run time was tested for run-graph.cpp with different amount of nodes in the graph in every plot, and each plot had different query size was. The run time appears to be linear with run time O(n), and the slope increases slightly when the query size increases.

#### 1.4.0.2 Run time for run-graph.cpp with different query sizes

Second the time run time was tested for run-graph.cpp with different query size in every plot, and each plot had different amount of nodes in the graph. There can not be a query size that is longer than the amount of nodes in the graph. Therefore each plot has more and more points in accordance with the graph size. The run time seems to be exponential O(n^2), and it takes longer with a bigger graph.

#### 1.4.0.3 Run time for comparing the paths with different amounts of paths found

The run time for comparing the paths with different amounts of paths found was tested with and without threading. The run time is linear O(n). When there is a small amount of paths found the code with threading is slower, but when there is a large amount of paths found the code is slightly faster, but barely noticeable. If we tested on larger numbers we would probably be able to see a bigger difference. This could be something for further work.

#### 1.4.0.4 Run time for comparing the paths with different amounts of equal paths found

The run time for comparing the paths with different amounts of equal paths found was tested with and without threading. Without threading the run-time seems to be exponential (O(n^2)), but linear (O(n)) with threading. The code is noticeably faster with threading.

## 1.5 Concurrency

## 1.6 Further work