Rapport to arbeidskrav 9. IDATT2101.

Note: I use JXMapViewer implementation here, so if you want to get map, you need to implement maven dependency. I will send also my pom.xml file

Dijktra`s algorithm

To make dijkstra work without output to map, I use method dijkstraAlgFromTo(int from, int to).

To make Dijkstra output map with shortest path, I use method createMapDij(DijkstraGraph dijkstraGraph,int start,int finish).

Kårvåg–Gjemnes

Algorithm without showing path on the map

Code in main:

```
public static void main(String[] args) throws IOException {

    DijkstraGraph util = new DijkstraGraph( vertexes: 1);
    DijkstraGraph original = util.graphFromFile( nodetxt: "noder.txt",  kanttxt: "kanter.txt",  interetxt: "interessepkt.txt");
    Map sample2 = new Map();
    int start = original.getNrByName( gg: "\"Kårvåg\"");
    int finish = original.getNrByName( gg: "\"Gjemnes\"");
    long startTime = System.currentTimeMillis();

    //sample2.createMapDij(original,start,finish);
    ArrayList<Integer> result = original.dijkstraAlgFromTo(start,finish);
    System.out.println("Nodes in rute:" + result.size());


    long endTime = System.currentTimeMillis();
    System.out.println("That took " + (endTime - startTime) + " milliseconds ");

}
```

Output:

```
Suksess nodes
Success vertexes
Dijkstra algorithm:
Estimatite driving time: 0:40:46
Polling from the queue:14031
Nodes in rute:329
That took 71 milliseconds
```

«Success nodes and vertexes» in output is only to show that graph is ready

Algorithm with showing path on the map

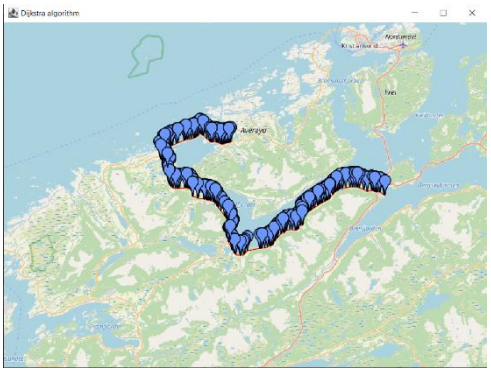
Code in main:

```
public static void main(String[] args) throws IOException {

    DijkstraGraph util = new DijkstraGraph( vertexes: 1);
    DijkstraGraph original = util.graphFromFile( nodetxt: "noder.txt",  kanttxt: "kanter.txt",  interetxt: "interessepkt.txt");
    Map sample2 = new Map();
    int start = original.getNrByName( gg: "\"Kårvåg\"");
    int finish = original.getNrByName( gg: "\"Gjemnes\"");
    // long startTime = System.currentTimeMillis();

    //sample2.createMapDij(original,start,finish);
    // ArrayList<Integer> result = original.dijkstraAlgFromTo(start,finish);
    // System.out.println("Nodes in rute:" + result.size());
    sample2.createMapDij(original,start,finish);


    //long endTime = System.currentTimeMillis();
    //System.out.println("That took " + (endTime - startTime) + " milliseconds ");

}
```

Output:

```
Suksess nodes
Success vertexes
Dijkstra algorithm:
Estimatite driving time: 0:40:46
Polling from the queue:14031
That took 101 milliseconds
Nodes in rute:329
```

Map:



Stavanger - Tampere

Algorithm without showing path on the map

Code in main:

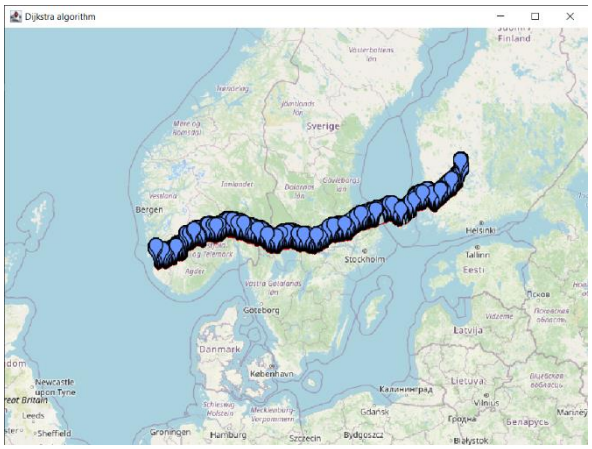
```
public static void main(String[] args) throws IOException {

    DijkstraGraph util = new DijkstraGraph( vertexes: 1);
    DijkstraGraph original = util.graphFromFile( nodetxt: "noder.txt",  kanttxt: "kanter.txt",  intertxt: "interessepkt.txt");
    Map sample2 = new Map();
    int start = original.getNrByName( gg: "\"Stavanger\"");
    int finish = original.getNrByName( gg: "\"Tampere\"");
    long startTime = System.currentTimeMillis();
    //sample2.createMapDij(original,start,finish);
    ArrayList<Integer> result = original.dijkstraAlgFromTo(start,finish);
    System.out.println("Nodes in rute:" + result.size());
    //sample2.createMapDij(original,start,finish);
    long endTime = System.currentTimeMillis();
    System.out.println("That took " + (endTime - startTime) + " milliseconds ");

}
```

Output:

```
Suksess nodes
Success vertexes
Dijkstra algorithm:
Estimatite driving time: 19:51:8
Polling from the queue:6641366
Nodes in rute:5317
That took 3390 milliseconds
```

Algorithm with showing path on the map:

Map:



Output:

```
Suksess nodes
Success vertexes
Dijkstra algorithm:
Estimatite driving time: 19:51:8
Polling from the queue:6641366
That took 3700 milliseconds
Nodes in rute:5317
```

Code in main

ALT algorithm

To make ALT-algorithm work without output to map, I use method dijkstraAlgFromTo(int from, int to).

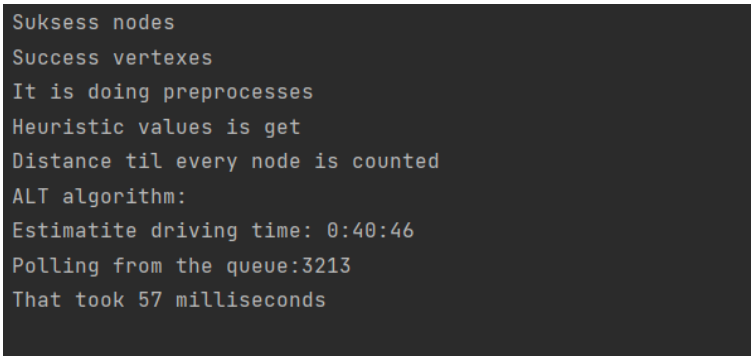
To make Dijkstra output map with shortest path, I use method createMapDij(DijkstraGraph dijkstraGraph,int start,int finish).

Kårvåg–Gjemnes

Alt without showing the map:

Code in main:

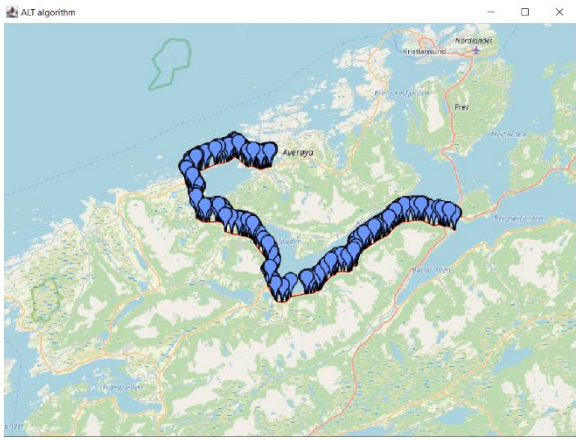
Output:



All strings before «ALT algorithm:» is only to monitore how is processes going in programm

Alt with showing the map:

Map:



Output:

```
Suksess nodes
Success vertexes
It is doing preprocesses
Heuristic values is get
Distance til every node is counted
ALT algorithm:
Estimatite driving time: 0:40:46
Polling from the queue:3213
That took 57 milliseconds
```

Code in main:

```
public static void main(String[] args) throws IOException {

    DijkstraGraph util = new DijkstraGraph( vertexes: 1);
    DijkstraGraph original = util.graphFromFile( nodetxt: "noder.txt",  kanttxt: "kanter.txt",  intertxt: "interessepkt.txt");
    Map sample2 = new Map();
    int start = original.getNrByName( gg: "\"Kårvåg\"");
    int finish = original.getNrByName( gg: "\"Gjemnes\"");
    //long startTime = System.currentTimeMillis();
    sample2.createMapALT(original,start,finish, landmark: 7314532);
    //ArrayList<Integer> result = original.altAlg(start,finish,7314532);
    //long endTime = System.currentTimeMillis();
    //System.out.println("That took " + (endTime - startTime) + " milliseconds ");


}
```

Stavanger–Tampere

Here ALT takes a lot of memory for preprocesses so I use -Xmx4g like VM variable

ALT without showing map:

Code:

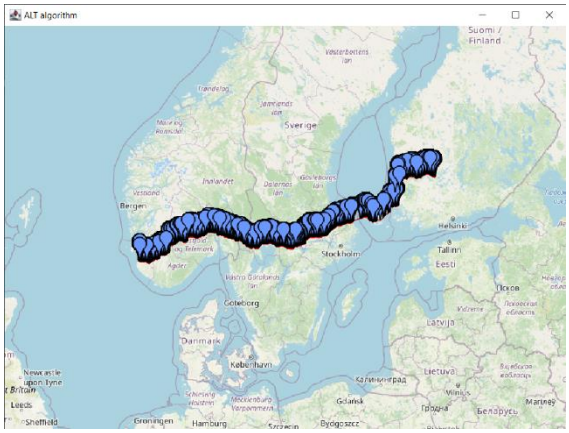
```
public static void main(String[] args) throws IOException {

    DijkstraGraph util = new DijkstraGraph( vertexes: 1);
    DijkstraGraph original = util.graphFromFile( nodetxt: "noder.txt",  kanttxt: "kanter.txt",  intertxt: "interessepkt.txt");
    Map sample2 = new Map();
    int start = original.getNrByName( gg: "\"Stavanger\"");
    int finish = original.getNrByName( gg: "\"Tampere\"");
    //sample2.createMapALT(original,start,finish,6857624);
    ArrayList<Integer> result = original.altAlg(start,finish, landmark: 6857624);


}
```

OutPut:

```
Suksess nodes
Success vertexes
It is doing preprocesses
Heuristic values is get
Distance til every node is counted
ALT algorithm:
Estimatite driving time: 19:51:8
Polling from the queue:2622309
That took 8053 milliseconds
```

ALT with showing the map:

Map:



Output:

```
Suksess nodes
Success vertexes
It is doing preprocesses
Heuristic values is get
Distance til every node is counted
ALT algorithm:
Estimatite driving time: 19:51:8
Polling from the queue:2622309
That took 5974 milliseconds
```

Code:

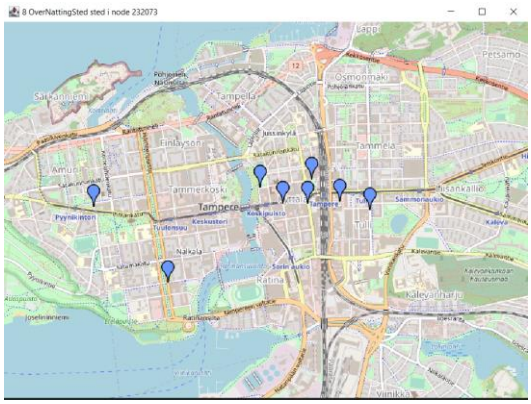
```java
public static void main(String[] args) throws IOException {

    DijkstraGraph util = new DijkstraGraph( vertexes: 1);
    DijkstraGraph original = util.graphFromFile( nodetxt: "noder.txt",  kanttxt: "kanter.txt",  intertxt: "interessepkt.txt");
    Map sample2 = new Map();
    int start = original.getNrByName( gg: "\"Stavanger\"");
    int finish = original.getNrByName( gg: "\"Tampere\"");
    sample2.createMapALT(original,start,finish, landmark: 6857624);


}
```

Bonus: How to find 8 interesse punkter?

For this I use createMapInteressePunkt(DijkstraGraph dijkstraGraph,int start,String modificator) method in Map class.

Example:find 8 overnatting sted in Tampere

Map:



Output:

```
Suksess nodes
Success vertexes
That took 31 milliseconds
Their name from closest:
"Forenom
"Omenahotelli
"Scandic
"Holiday
"Scandic
"Scandic
"Lapland
"Hotel
```

Code:

```
public static void main(String[] args) throws IOException {

    DijkstraGraph util = new DijkstraGraph( vertexes: 1);
    DijkstraGraph original = util.graphFromFile( nodetxt: "noder.txt",  kanttxt: "kanter.txt",  intertxt: "interessepkt.txt");
    Map sample2 = new Map();
    //int start = original.getNrByName( gg: "\"Stavanger\"");
    int finish = original.getNrByName( gg: "\"Tampere\"");
    sample2.createMapInteressePunkt(original,finish, modificator: "OverNattingSted");


}
```

Conclusion

ALT gives us less polling from priority queue, but takes more memory for all preprocesses` stuff. In speed performance my ALT loses to my Dijkstra`s, but it only because I have one landmark in algorithm. If I will implement 3-4 landmark estimators will be better, but it will take more memory.

PS: In my program you can see different variation of Dijkstra algorithm (one for Dijkstra, second for ALT and 3rd for find interesse punkter). I made a big work during this arbeidskrav and tried to implement different versions of Dijkstra`s that will suit for my method or optimize algorithms. About JXMap I took implementation from github and customize it for myself, but it is the only part that I took from the internet.