Travelling salesman person

Travelling salesman person person implemented for purposes of DD2440 - Advanced Algorithms course on KTH, Stockholm.

Installing

Run this command to compile project

```
g++ -g -O2 -static -std=gnu++14 -o TSP.out main.cpp christ
ofides.cpp opt2local.cpp sia.cpp graph.cpp graph.h opt2loc
al.h sia.h christofides.h branch_bound.cpp branch_bound.h
ant_colony_optimization.cpp ant_colony_optimization.h rand
om_provider.cpp random_provider.h  simulated_annealing.cpp
 simulated_annealing.h greedy_algorithm.cpp greedy_algorit
hm.h utility.cpp utility.h
```

And output should look like this:

```
TSP.out
```

Running

There are three ways of running algorithm:

Running with Command line (std IO) input:

This type of running expects input to start with number of cities followed by their coordinates.

```
./TSP.out
```

Running with Command line (std IO) input with specific algorithm:

Also requires to be ran from command line, passing the command line argument as the name of algorithm.

```
./TSP.out <algorithm>
```

Where <algorithm> can be:

- CHRISTOFIDES - Christofides’ algorithm
- SIA - Simple immunological algorithm on top of Christofides construction
- ACO - Ant colony optimization algorithm
- SA - Simulated annealing on top of Christofides
- BNB - Branch and Bound algorithm for small instances

Running with File input with specific algorithm:

File should be stored in /samples folder and is used as input to algorithm with each line containing city coordinates. That file is defined as <instance>. Also, algorithm input should be provided same as in previous section. Last argument is optional, meaning if you include argument **v**, **Python 3** script would be started to draw solution. Started trial is going to be logged into /logs folder.

```
./TSP.out <instance> <algorithm> v
```

Authors

- **Matak, Josip**
- **Mrđen, Josip**

Acknowledgments

- Code lacks inspection and it is developed for purposes of obtaining better results
- Code lacks usage of common design patters