RouteFinder

Generated by Doxygen 1.8.8

Fri Nov 7 2014 11:07:02

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 Edge Class Reference

```
#include <Edge.h>
```

Public Member Functions

- Edge (unsigned int id, Node ∗start, Node ∗end, const double weight, const TransportType type)
- unsigned int getID () const
- double getWeight () const
- const Node ∗ getStartNode () const
- const Node ∗ getEndNode () const
- TransportType getType () const
- void setWeight (double w)
- void setType (TransportType t)
- bool operator== (const Edge &e) const
- bool operator!= (const Edge &e) const
- bool operator< (const Edge &e) const
- Edge & operator= (const Edge &e)

Friends

- std::ostream & operator<< (std::ostream &s, const Edge &e)

3.1.1 Detailed Description

Object used for storage of one connections. Includes info about start and end positions, type of connection and weight of it.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Edge::Edge (unsigned int *id*, Node ∗ *start*, Node ∗ *end*, const double *weight*, const TransportType *type*)

Constructor of Edge object.

Parameters

id	Identificator of object.
start	Pointer to Node assigned as starting position.
end	Pointer to Node assigned as ending position.
weight	Given weight.
type	Enumerated value describing type of route.

3.1.3 Member Function Documentation

3.1.3.1 const Node * Edge::getEndNode ( ) const

Returns

Returns pointer to ending Node.

3.1.3.2 unsigned int Edge::getID ( ) const

Returns

Returns id of itself.

3.1.3.3 const Node * Edge::getStartNode ( ) const

Returns

Returns pointer to starting Node.

3.1.3.4 TransportType Edge::getType ( ) const

Returns

Returns TransportType enum value describing type.

3.1.3.5 double Edge::getWeight ( ) const

Returns

Returns weight of itself.

3.1.3.6 bool Edge::operator!= (const Edge & *e*) const

Compares itself id with given edge id.

Parameters

e	Given Edge.

Returns

True if ids are not equal, false otherwise.

3.1.3.7 bool Edge::operator< (const Edge & *e*) const

Compares itself id with given edge id. Method used in Network class.

Parameters

e	Given Edge.

Returns

True if this->id is smaller than e.id, false otherwise.

3.1.3.8 Edge & Edge::operator= (const Edge & *e*)

Assign operator.

Parameters

e	Given Edge.

Returns

Reference to itself.

3.1.3.9 bool Edge::operator== (const Edge & *e*) const

Compares itself id with given edge id.

Parameters

e	Given Edge.

Returns

True if ids are equal, false otherwise.

3.1.3.10 void Edge::setType (TransportType *t*)

Sets type to given.

Parameters

t	Given type value. Should be not equal to UNKNOWN.

3.1.3.11 void Edge::setWeight (double *w*)

Sets weight to given.

Parameters

w	Given weight value. Should be greater than 0.

3.1.4 Friends And Related Function Documentation

3.1.4.1 std::ostream& operator<< (std::ostream & *s*, const Edge & *e*) `[friend]`

Operator used for console debug purposes.

Parameters

s	Stream which is used for output.
e	Edge on which operator is called.

Returns

Given stream.

The documentation for this class was generated from the following files:

- /home/vka/Workspace/RouteFinder/src/Edge.h
- /home/vka/Workspace/RouteFinder/src/Edge.cpp

3.2 Network Class Reference

```
#include <Network.h>
```

Public Member Functions

- Network ()
- Network (std::string f)
- ∼Network ()
- void loadFromFile (std::string f)
- void setSover (Solver ∗s)
- Route ∗ findRouteBetween (const Node ∗start, const Node ∗end)

Friends

- std::ostream & operator<< (std::ostream &s, const Network &n)

3.2.1 Detailed Description

main class, contains information about nodes and edges between them. Should be created from file containing data in GTFS or other format. //todo loadFromFile method should load "db/db.ext" file and save it to inner variables.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Network::Network ( )

Network object constructor. If this constructor is called, loadFromFile method need to be called after.

3.2.2.2 Network::Network (std::string *f*)

Network object constructor in which Network::loadFromFile() method is being called.

Parameters

f	Name of file from which database is loaded.

3.2.2.3 Network::∼Network ( )

Destructs all objects in Network and itself.

3.2.3 Member Function Documentation

3.2.3.1 Route ∗ Network::findRouteBetween (const Node ∗ *start*, const Node ∗ *end*)

Searches for Route beetween two given points.

Parameters

start	Start Node.
end	End Node.

Returns

Pointer to Route between given nodes, NULL if no route can be found.

3.2.3.2 void Network::loadFromFile (std::string *f*)

Load database entries from given file.

Parameters

f	Filename from which database is being loaded.

3.2.3.3 void Network::setSover (Solver ∗ *s*)

Set solved used in Network::findRouteBetween() method.

Parameters

s	Pointer to Solver being used.

3.2.4 Friends And Related Function Documentation

3.2.4.1 std::ostream& operator<< (std::ostream & *s*, const Network & *n*) `[friend]`

Used of debug in console purposes.

Parameters

s	Stream used for output.
n	Reference to Network being printed.

Returns

Given stream.

The documentation for this class was generated from the following files:

- /home/vka/Workspace/RouteFinder/src/Network.h
- /home/vka/Workspace/RouteFinder/src/Network.cpp

3.3 Node Class Reference

```
#include <Node.h>
```

Public Member Functions

- Node (unsigned int id, std::string name, double lon, double lat)
- double getLongtitude () const
- double getLatitude () const
- unsigned int getID () const
- std::string getName () const
- bool operator== (const Node &n) const
- bool operator!= (const Node &n) const
- bool operator< (const Node &n) const
- Node & operator= (const Node &n)

Friends

- std::ostream & operator<< (std::ostream &s, const Node &n)

3.3.1 Detailed Description

primary element. Contains info about position and name of itself.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Node::Node (unsigned int *id*, std::string *name*, double *lon*, double *lat*)

Constructs new Node object.

Parameters

id	Id of an node.
name	Name of node (stop).
lon	Longtitude coord.
lat	Latitude coord.

3.3.3 Member Function Documentation

3.3.3.1 unsigned int Node::getID ( ) const

Returns

Returns id of itself.

3.3.3.2 double Node::getLatitude ( ) const

Returns

Returns latutide as double.

3.3.3.3 double Node::getLongtitude ( ) const

Returns

Returns longtitude as double.

3.3.3.4 std::string Node::getName ( ) const

Returns

Returns string containing name.

3.3.3.5 bool Node::operator!= (const Node & *n*) const

Operator compares id of this and given node.

Parameters

n	Node which is being compared.

Returns

False if ids are equal, true otherwise.

3.3.3.6 bool Node::operator< (const Node & *n*) const

Operator used in sets in Network class. Compares ids.

Parameters

n	Node which is being compared.

Returns

True if this->id is smaller than n.id, false otherwise.

3.3.3.7 Node & Node::operator= (const Node & *n*)

Copies params from given Node to itself.

Parameters

n	Node which is being copied.

Returns

Reference to itself.

3.3.3.8 bool Node::operator== (const Node & *n*) const

Operator compares id of this and given node.

Parameters

n	Node which is being compared.

Returns

True if ids are equal, false otherwise.

3.3.4 Friends And Related Function Documentation

3.3.4.1 std::ostream& operator<< (std::ostream & *s*, const Node & *n*) `[friend]`

Operator used for console debug purposes.

Parameters

s	Stream which is used for output.
n	Node on which operator is called.

Returns

Given stream.

The documentation for this class was generated from the following files:

- /home/vka/Workspace/RouteFinder/src/Node.h
- /home/vka/Workspace/RouteFinder/src/Node.cpp

3.4 Route Class Reference

`#include <Route.h>`

Public Member Functions

- Route ()
- unsigned int getLength () const
- double getWeight () const
- bool validate () const
- bool addEdge (const Edge ∗e)
- bool switchEdge (const Edge ∗e)
- bool switchRoute (Route &r)
- const Node ∗ getStartNode () const
- const Node ∗ getEndNode () const
- bool isNodeIn (const Node ∗n) const
- bool isEdgeIn (const Edge ∗e) const
- bool isConnectionBetween (const Node ∗start, const Node ∗end) const
- std::list< const Edge ∗ >
  ::const_iterator begin ()
- std::list< const Edge ∗ >
  ::const_iterator end ()

Friends

- std::ostream & operator<< (std::ostream &s, Route &r)

3.4.1 Detailed Description

contains information about route between two points Should be used in solver class.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 Route::Route ( )

Constructs object. No data is necessary.

3.4.3 Member Function Documentation

3.4.3.1 bool Route::addEdge (const **Edge** $*$ *e*)

Add one Edge to the end of Route. Given edge must start in Node, in which current route ends.

Parameters

e	Pointer to given edge.

Returns

True if edge could be connected to route, false otherwise.

3.4.3.2 std::list$<$ const **Edge** $*$ $>$::const_iterator Route::begin ( )

Returns

Returns iterator to the begining of route.

3.4.3.3 std::list$<$ const **Edge** $*$ $>$::const_iterator Route::end ( )

Returns

Returns iterator to point after last Edge in route.

3.4.3.4 const **Node** $*$ Route::getEndNode ( ) const

Returns

pointer to Node on the end of Route.

3.4.3.5 unsigned int Route::getLength ( ) const

Returns

Returns length - number of Edge objects.

3.4.3.6 const **Node** $*$ Route::getStartNode ( ) const

Returns

Returns pointer to Node on the begining of Route.

3.4.3.7 double Route::getWeight ( ) const

Returns

Returns sum of weights of Edge objects.

3.4.3.8 bool Route::isConnectionBetween (const Node ∗ *start*, const Node ∗ *end*) const

Checks for subroute between given nodes.

Parameters

start	Pointer to Node of start.
end	Pinter to Node of end.

Returns

True if there is subroute from start to end in route, false otherwise.

3.4.3.9 bool Route::isEdgeIn (const Edge ∗ *e*) const

Parameters

e	Pointer to Edge which is being searched for.

Returns

True if Edge is included in route, false otherwise.

3.4.3.10 bool Route::isNodeIn (const Node ∗ *n*) const

Parameters

n	Pointer to Node which is being checked.

Returns

True if given node is currently in route. False otherwise.

3.4.3.11 bool Route::switchEdge (const Edge ∗ *e*)

Switches given edge with one included in path,

Parameters

e	Pointer to Edge. Its start Node and end Node must be same as start and end nodes of edge included in route.

Returns

True if switch was successful, false otherwise.

3.4.3.12 bool Route::switchRoute (Route & *r*)

Switches part of Route with given route.

Parameters

r	Reference to subroute which needs to be inserted into object. It must to be correct (validate method is being called), and start and end of subroute must have corresponding values as start and end of some subroute inside current object. Length of switched subroutes do not need to be equal.

Returns

True if switch was successful, false otherwise.

3.4.3.13 bool Route::validate ( ) const

Checks if Route does not contain any loops and if all edges are connected. I.e. A->B and then B->C is ok, but A->B and C->D is wrong.

Returns

True if test is passed, false otherwise.

3.4.4 Friends And Related Function Documentation

3.4.4.1 std::ostream& operator<< (std::ostream & *s*, Route & *r*) `[friend]`

Used of debug in console purposes.

Parameters

s	Stream used for output.
r	Reference to Route being printed.

Returns

Given stream.

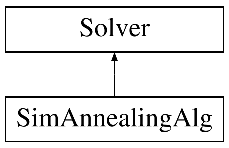
The documentation for this class was generated from the following files:

- /home/vka/Workspace/RouteFinder/src/Route.h
- /home/vka/Workspace/RouteFinder/src/Route.cpp

3.5 SimAnnealingAlg Class Reference

`#include <SimAnnealingAlg.h>`

Inheritance diagram for SimAnnealingAlg:



Public Member Functions

- virtual Route ∗ solve (const Network ∗n)

3.5.1 Detailed Description

Simulated Annealing Algorithm used for finding routes. See doc folder for more information.

3.5.2 Member Function Documentation

3.5.2.1 virtual Route∗ SimAnnealingAlg::solve (const Network ∗ *n*) `[virtual]`

Method used in Network class for finding best connection between points.

Parameters

n	Pointer to Network in which Route is being searched for.

Returns

Pointer to found Route, NULL if no route can be found.

Implements Solver.

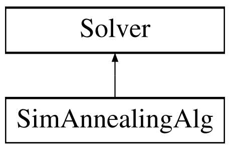
The documentation for this class was generated from the following file:

- /home/vka/Workspace/RouteFinder/src/SimAnnealingAlg.h

3.6 Solver Class Reference

`#include <Solver.h>`

Inheritance diagram for Solver:



Public Member Functions

- virtual Route ∗ solve (const Network ∗n)=0

3.6.1 Detailed Description

wrapper class for solver algorithm. Those shall inherit from Solver class. Solver needs to implement solve method, which gets Network map as

3.6.2 Member Function Documentation

3.6.2.1 virtual Route∗ Solver::solve (const Network ∗ *n*) `[pure virtual]`

Method used in Network class for finding best connection between points. This method need to be implemented in any class inheriting from Solver class.

Parameters

n	Pointer to Network in which Route is being searched for.

Returns

Pointer to found Route, NULL if no route can be found.

Implemented in SimAnnealingAlg.

The documentation for this class was generated from the following file:

- /home/vka/Workspace/RouteFinder/src/Solver.h

Index