A Simple Map Functionality : ECE650 Assignment 3

Group Members : Anson Jacob(20595866), Niyaz Murshed(20641996), Minita Dabhi(20498102)

This project aims to develop a simple map utility to find shortest path beween two places. The shortest path is found by using Dijkstra's algorithm.

The map is implemented in **roadmap.c** and is provided with a header file **roadmap.h**.

Functionalities Supported

The roadmap supports adding:

- Vertex
- Edge
- Add an Edge Event
- Name a Road between edges
- Lookup for suggestions for Vertex Name in the Map
- Find the shortest distance between two vertex
- Store the map into a file
- Retreive the map from a file previously saved using Store
- Print the Graph Info

A sample file is also provided named **main.c** which showcases the usage of the different APIs

For testing purposes, two sample maps are created in **sample1.c** and **sample2.c**. Scripts are created to run them: ./run-sample1.sh and ./run-sample2.sh

Documentation

Run **doxygen Doxyfile**

Compile

Run **make**

Run

./runmap

Learning

The project aided the group to explore, and learn:

- Dynamic Data Structures in C
- Dijkstra's algorithm using directed weighted graph
- Levenshtein distance
- Development in a *nix Machine (Ubuntu/Mac)
- API in C and use of header file

- Vim Editor
- Makefile
- Bash scripts
- Git
- Doxygen
- Markdown

This document was generated using Markdown and Doxygen.

---

roadmap.h File Reference

Roadmap API Documentation. More...

Go to the source code of this file.

Macros

#define	**VERTEX_NAME_SIZE** (20)
#define	**EDGE_NAME_SIZE** ((2*VERTEX_NAME_SIZE) + 1)
#define	**ROAD_NAME_SIZE** (3*VERTEX_NAME_SIZE)
#define	**FILENAME** "graphInfo.txt"

Enumerations

enum	**edge_type** { **ONE_WAY**, **TWO_WAY** }
enum	**vertex_type** { **POINT_OF_INTEREST**, **INTERSECTION**, **MAX_VERTEX_TYPE** }
enum	**edge_event** { **EVENT_NORMAL**, **EVENT_ACCIDENT**, **EVENT_ROADBLOCK**, **MAX_EDGE_EVENT** }

Functions

int	**addVertex** (enum **vertex_type** type, char *name) Adds a Vertex the Map. More...
int	**addEdge** (char *vertex1, char *vertex2, int directional, double speed, double length) Adds an Edge to the Map. More...
int	**edgeEvent** (char *edgename, enum **edge_event** event) Adds an edge event on the Map. More...
int	**road** (char edges[][**EDGE_NAME_SIZE**], int edgescount, char *roadname) Adds a Vertex the Map. More...
int	**trip** (char *vertex1, char *vertex2) Calculates the shortest feasible path between two vertex by using Dijkstra's Algorithm. More...
char *	**vertex** (char *vertex) Returns the Map identifier for the closest match of the vertex name using Levenshtein distance. More...
int	**store** (char *filename) Store the current state of the map into a text file in a custom csv format. More...
int	**retrieve** (char *filename) Retrieves a Map saved into a file using **store()**. More...
void	**printGraph** () Prints the Map info to stdout. More...
void	**freeGraph** () Call this API to clear all the data structures used by the Map. More...

Detailed Description

Roadmap API Documentation.

Author

Anson Jacob

Niyaz Murshed

Minita Dabhi

Date

20/11/2016

Macro Definition Documentation

§ EDGE_NAME_SIZE

#define EDGE_NAME_SIZE ((2***VERTEX_NAME_SIZE**) + 1)

Maximum number of character for Edge Name.

§ FILENAME

#define FILENAME "graphInfo.txt"

Arbitary filename for saving and retreiving map.

§ ROAD_NAME_SIZE

#define ROAD_NAME_SIZE (3***VERTEX_NAME_SIZE**)

Maximum number of character for Road Name.

§ VERTEX_NAME_SIZE

#define VERTEX_NAME_SIZE (20)

Maximum number of character for Vertex Name.

Enumeration Type Documentation

§ edge_event

enum **edge_event**

Events at Edge

Enumerator	
EVENT_NORMAL	Edge is functional.
EVENT_ACCIDENT	Edge has an accident.
EVENT_ROADBLOCK	Edge has a road block.
MAX_EDGE_EVENT	Maximum numver of edge events.

§ edge_type

enum **edge_type**

Edge Direction

Enumerator	
ONE_WAY	One-way edge / road
TWO_WAY	Two-way edge / road

§ vertex_type

enum **vertex_type**

Vertex Types

Enumerator	
POINT_OF_INTEREST	Point of Interest.
INTERSECTION	Intersection.
MAX_VERTEX_TYPE	Maximum number of vertex types.

Function Documentation

§ addEdge()

```
int addEdge ( char *  vertex1,
              char *  vertex2,
              int     directional,
              double  speed,
              double  length
            )
```

Adds an Edge to the Map.

Usage:

```
addEdge("DC", "MC", ONE_WAY, 30, 6.0);
addEdge("MC", "SLC", TWO_WAY, 35, 4.2);
```

Parameters

vertex1	Character pointer to the name of the first edge.
vertex2	Character pointer to the name of the second edge.
directional	0: For single and 1: bi-directional edge.
speed	Maximum Speed limit on the edge in KM/Hr.
length	Distance between two edges in KM.

Returns

Success: 0.

Failure: -1.

See also

addVertex

freeGraph

By default, Edge event is set to EVENT_NORMAL.

§ addVertex()

int addVertex (enum **vertex_type** type,
char * name
)

Adds a Vertex the Map.

Usage:

```
addVertex(POINT_OF_INTEREST, "DC");
addVertex(POINT_OF_INTEREST, "MC");
addVertex(INTERSECTION, "SLC");
```

Parameters

type Type of Vertex.

name Pointer to a character array which holds the name of the vertex.

Returns

Success: Total number of vertex in the Map.

Failure: -1.

See also

addEdge

freeGraph

§ edgeEvent()

int edgeEvent (char * edgename,
enum **edge_event** event
)

Adds an edge event on the Map.

Usage:

```
edgeEvent("DC-RR", EVENT_NORMAL);
edgeEvent("SLC-MC", EVENT_ROADBLOCK); // One-way roadblock
```

Parameters

edgename Name of edge formed by concatenating two vertex names by a '-'.

event type.

Returns

Success: 0.

Failure: -1.

See also

freeGraph

§ freeGraph()

void freeGraph ( )

Call this API to clear all the data structures used by the Map.

Usage:

```
freeGraph();
```

§ printGraph()

void printGraph ( )

Prints the Map info to stdout.

Usage:

```
printGraph();
```

§ retrieve()

int retrieve (char * filename)

Retrieves a Map saved into a file using **store()**.

Usage:

```
retrieve(FILENAME);
```

Parameters

filename Pointer to a character array which holds the filename.

Returns

Success: 0.

Failure: -1.

See also

store

freeGraph

If there are errors while adding a certain object, an error message is displayed, but the process continues until the whole file is parsed.

§ road()

```
int road ( char    edges[][EDGE_NAME_SIZE],
           int     edgescount,
           char *  roadname
         )
```

Adds a Vertex the Map.

Usage:

```
#define NUM_OF_EDGES 1

char roadInfo[NUM_OF_EDGES][EDGE_NAME_SIZE];
char roadname[ROAD_NAME_SIZE] = "Ring Road";

memcpy(roadInfo[0], "DC-RR", EDGE_NAME_SIZE);

road(roadInfo, NUM_OF_EDGES, roadname);
```

Parameters

edges	A 2-D char array of Name's of edges formed by concatenating two vertex names by a '-'.
edgescount	Number of rows in the 2-D array.
roadname	Pointer to a character array which holds the name of the road.

Returns

Success: Returns the number of road names added.

Failure: -1.

See also

freeGraph

§ store()

int store (char * filename)

Store the current state of the map into a text file in a custom csv format.

Usage:

```
store(FILENAME);
```

Parameters

filename Pointer to a character array which holds the filename.

Returns

Success: 0.

Failure: -1.

See also

retrieve

freeGraph

§ trip()

int trip (char * vertex1,
char * vertex2
)

Calculates the shortest feasible path between two vertex by using Dijkstra's Algorithm.

Usage:

```
char* src;
char* dest;

src = vertex("DC");
dest = vertex("SLC");

trip(src, dest);
```

Parameters

vertex1 Pointer to a character array which holds the name of the source vertex

vertex2 Pointer to a character array which holds the name of the destination vertex

Returns

Success: 0.

Failure: -1.

See also

freeGraph

§ vertex()

char* vertex (char * vertex)

Returns the Map identifier for the closest match of the vertex name using Levenshtein distance.

Usage:

```
char* ver = NULL;

ver = vertex("DC");
if(ver == NULL)
 printf("Vertex not found\n")
```

Parameters

vertex Pointer to a character array which holds the name of the vertex

Returns

Success: Returns the pointer to vertex name.

If Vertex is not found: NULL.

See also

freeGraph

roadmap.h

Go to the documentation of this file.

```
  1
 10 #ifndef ROADMAP_H
 11 #define ROADMAP_H
 12
 14 #define VERTEX_NAME_SIZE        (20)
 15
 16 #define EDGE_NAME_SIZE          ((2*VERTEX_NAME_SIZE) + 1)
 17
 18 #define ROAD_NAME_SIZE          (3*VERTEX_NAME_SIZE)
 19
 20 #define FILENAME            "graphInfo.txt"
 21
 23 enum edge_type {
 24     ONE_WAY,
 25     TWO_WAY
 26 };
 27
 29 enum vertex_type {
 30     POINT_OF_INTEREST,
 31     INTERSECTION,
 32     MAX_VERTEX_TYPE
 33 };
 34
 36 enum edge_event {
 37     EVENT_NORMAL,
 38     EVENT_ACCIDENT,
 39     EVENT_ROADBLOCK,
 40     MAX_EDGE_EVENT
 41 };
 42
 59 int addVertex(enum vertex_type type, char* name);
 60
 81 int addEdge(char* vertex1, char* vertex2, int directional, double speed, double length);
 82
 97 int edgeEvent(char* edgename, enum edge_event event);
 98
120 int road(char edges[][EDGE_NAME_SIZE], int edgescount, char* roadname);
121
143 int trip(char* vertex1, char* vertex2);
144
164 char* vertex(char* vertex);
165
179 int store(char* filename);
180
197 int retrieve(char* filename);
198
207 void printGraph();
208
217 void freeGraph();
218
219 #endif
```

Test cases for API

API	Test Case / Steps	Input	Expected Output	Actual result / Message	Pass / Fail
1	**addVertex(type, label) API**				
	Check if the API is adding vertex with its type	addVertex(PointOfInterest,"DC")	Valid,Message		Pass
	Negative test of inputs				
	(a)Input name of the vertex with wrong syntax	addVertex(PointOfInterest,DC)	Invalid,Error Message	sample2.c:174:2: error: expected '}'	Fail
	(b)Input random string instead of POI or Intersection	addVertex(AXX, "DC")	Invalid,Error Message	error: use of undeclared identifier 'AXX' addVertex(AXX, "Downsview");	Fail
	(c)Input numeric string as vertex name and type	addVertex(123,45)	Invalid,Error Message	warning: incompatible integer to pointer conversion passing 'int' to parameter of type 'char *' [-Wint-conversion] addVertex(123, 45);	Fail
2	**addEdge(vertex1, vertex2, directional, speed, length) API**				
	Check if API is adding edge between vertex1 and vertex2 with attributes direction,speed,length	addEdge("DC","MC",1,12,60)	Valid,Message		Pass
	Negative test of inputs				
	(a)Input vertex name without "",wrong syntax	addEdge(dc,"MC",1,12,60)	Invalid,Error Message	error: use of undeclared identifier 'Downsview'	Fail
	(b)Input length of edge 0	addEdge("DC","MC",1,0,60)	Invalid,Error Message	Length can't be negative or zero	Pass
	(c)Input speed of driving 0	addEdge("DC","MC",1,12,0)	Invalid,Error Message	Speed can't be negative or zero	Pass
3	**edgeEvent(edge, eventType) API**				
	Check if API is adding the event(Road closure or open) to the edge or not	edgeEvent("DC-RR",0)	Valid,Message		Pass
	Negative test of inputs				
	(a)Input edge name with wrong syntax	edgeEvent(DC-RR,0)	Invalid,Error Message	use of undeclared identifier 'Warden' edgeEvent(DC-RR EVENT_ROADBLOCK); ^	Fail
	(b)Input edge name with invalid event type	edgeEvent("DC-RR",5)	Invalid,Error Message	Edge event (5) not found	Pass
4	**vertex(point-of-interest) API**				
	Check if API returns the int value related with Point Of Interest from the list	vertex("DC")	Valid,Message,returns int value		Pass
	Negative test of inputs				
	(a)Input edge name with wrong syntax	vertex(DC)	Invalid,Error Message	error: use of undeclared identifier 'DC'	Fail
	(c) Input name with variation in cases of letters	vertex("dc")	Valid		Pass
5	**road(edges[],arraysize,label) API**				
	Test if API is assigning label to the collection of edges	road(roadinfo,2,roadname),where roadinfo is the array of edges,roadname is the variable with name assigned	Valid,Message		Pass
	Negative test of inputs				
	(b)Input the alphabetical strings in no.of edges	road(roadinfo,abc,roadname),	Invalid,Error Message	Compile error	Fail
	(c)Input the invalid roadname	road(roadinfo,2,roadname), roadname - #$!@	Invalid,Error Message	Compile warning	Fail
6	**trip(fromVertex, toVertex) API**				
	Test if API is returning shortest path for the trip or not	trip("DC","RR")	Valid,Returns shortest path		Pass
	Negative test of inputs				
	(a)Input numerical string	trip("123","RR")	Invalid,Error Message	Vertex 123 not found	Pass
	(b)Input the POI without """"	trip(DC,RR)	Invalid,Error Message	Vertext1 or Vertex2 invalid	Pass
	(c)Input the same POI instead of 2 different POIs	trip("DC","DC")	Invalid,Error Message	You are already at DC	Pass
	(d)Input 1 valid POI and other POI with no edge	trip("DC","ABC")	Invalid,Error Message	Vertex ABC not found	Pass
7	**store(filename) API**				
	Test if API stores the data in the file with filename given	store(FILENAME), where FILENAME- GraphInfo.txt	Valid	pass	Pass
	Negative test of inputs				
	(a) Input space between 2 words	FILENAME = graph info.txt	Valid	No message	Pass
	(b)Input alphanumeric string as filename	FILENAME = gr@ph12.txt	Valid	No message	Pass
8	**retrive(filename) API**				
	Test if API return/reads the data stored previously in the file named FILENAME	retrieve(FILENAME),where FILENAME - GraphInfo.txt	Valid,Message	pass	Pass