Shortest Path in Real Maps

BRIDGES Team

SIGCSE 2019

Outline

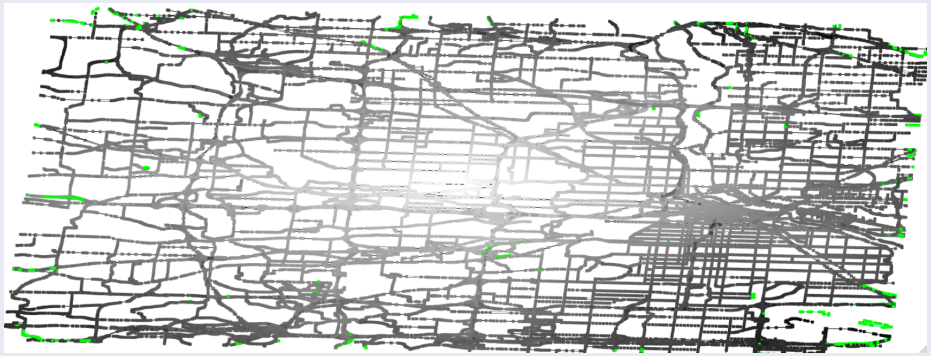
1. Presentation of the Problem and Overview
2. An Data Structures/Algorithm problem
3. Variants and Reflection

GPS Routing Application

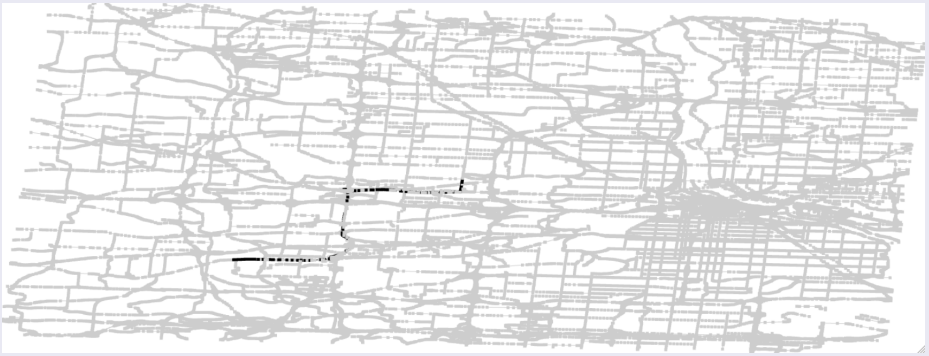
Algorithms

- Identify closest vertex to coordinate.
- Compute Single Source Shortest Path.
- Highlight distances.
- Follow and Highlight path.

Distances in Minneapolis



Path in Minneapolis



What does BRIDGES do for you?

Engagement

Enables student to get maps for any location.
Access real data of sizable scale.
Build a real application.

Getting Maps (through Open Street Map)

```
DataSource ds (&bridges);
OSMData osm_data = ds.getOSMData("minneapolis");
GraphAdjList<int, OSMVertex, double> graph;
osm_data.getGraph (&graph);
```

Styling Graphs

```
ElementVisualizer* elvis = graph.getVertex(vertID)->getVisualizer();
elvis->setColor(Color(0,0,0,255));
```

Outline

1 Presentation of the Problem and Overview

2 An Data Structures/Algorithm problem

3 Variants and Reflection

Bird's eye view of the Shortest Path assignment

Getting the data

Get data from an API into the program.
Topics: API Usage.

Finding a Source

Identifying the vertex the closest to a coordinate.
Topics: For loops/Reduction, Spatial Data Structure.

Single Source Shortest Path

Computing distance from a source to all vertices.
Topics: Dijkstra Graph Algorithms.

Single Pair Shortest Path

Identifying the path between a source and a destination.
Topics: Graph Algorithms, Pointer Chasing.

Getting the data (Topics: API usage)

Get a graph from an API into the program, and visualize it.

In C++

```
DataSource ds (&bridges);
OSMData osm_data = ds.getOSMData("minneapolis");
GraphAdjList<int, OSMVertex, double> graph;
osm_data.getGraph (&graph);
```

In Java

```
OsmData osm_data = bridges.getOsmData("uncc_campus");
GraphAdjList<Integer, OsmVertex, Double> graph = osm_data.getGraph ();
```

In Python

```
osm_data = data_source.get_osm_data("uncc_campus")
gr = osm_data.get_graph()
```

Finding a Source (Topics: For Loops, Quad Trees)

Find the vertex the closest to the center of the map, and style it.

In C++

```
const OSMVertex& vertdata = graph.getVertexData(vertID);
vertdata.getLatitude(); // vertdata.getLongitude();
graph.getVisualizer(vertID)->setColor(Color(255,0,0,255));
```

In Java

```
OsmVertex v = graph.getVertex(i).getValue();
double d1 = v.getLatitude(); // v.getLongitude()
ElementVisualizer elvis = graph.getVertex(root).getVisualizer();
elvis.setColor(new Color(255, 0, 0, 1.0));
```

In Python

```
theosmvertex = gr.get_vertex(k).get_value()
vlat = theosmvertex.latitude # .longitude
gr.get_visualizer(vertID).set_color(0,0,0)
```

Single Source Shortest Path (Dijkstra, Priority Queues)

Dijkstra's algorithm is a good algorithm for student to implement. Opens questions about Priority Queues. (Scaffolded here.)
Style as a function of distance.

In C++

```
std::unordered_map<int, double> distance;
dijkstra(graph, source, distance);
```

In Java

```
HashMap<Integer, Double> distance= new HashMap<Integer, Double>();
dijkstra(graph, closest, distance);
```

In Python

```
distance = dijkstra(gr,root)
```

Single Pair Shortest Path (Topics: Graph Algorithms)

Modify Dijkstra's implementation to add parent pointer. Find a destination. Style the graph to show the path.

In C++

```
ElementVisualizer* elvis = graph.getVertex(vertID)->getVisualizer();
elvis->setColor(Color(0,0,0,255));
LinkVisualizer* livis = graph.getLinkVisualizer(src, dest);
if (livis != nullptr) livis->setColor(Color(0,0,0,255));
```

In Java

```
ElementVisualizer elvis = graph.getVertex(vertID).getVisualizer();
elvis.setColor(0, 0, 0, 1.0f);
try { LinkVisualizer livis = graph.getLinkVisualizer(from, to);
    livis.setColor(0, 0, 0, 1.0f);
} catch (Exception e) {} //exception is thrown no (from,to) edge
```

In Python

```
gr.get_visualizer(vertID).set_color(0,0,0)
gr.get_link_visualizer(v, nei).set_color(0, 0, 0, 1.0)
```

Outline

1. Presentation of the Problem and Overview
2. An Data Structures/Algorithm problem
3. Variants and Reflection

Complexity Questions

Spatial queries

- Linear Search
- Quad trees
- k-d trees

Priority queues

- Sorted Arrays
- Min-Heap
- Fibonnacci Heap

Different Size

- Campus
- Downtown
- Whole City
- Metro Area

Other Problems

- Spanning Tree

Questions from the room?