Dijkstra's Algorithm Game Implementation Using Processing

Sebastian Florez	Jacob Koko	Marisa Gomez	Alex Ordonez
Stetson University	Stetson University	Stetson University	Stetson University

Abstract

An implementation of Dijkstra's shortest path algorithm using the Processing language and environment. The purpose of the game is for a user to traverse a city by selecting what they believe would be the shortest path from the start point to the end point.

Introduction

The purpose of this project is to provide users with a visual interfacing medium that allows them to interact with Dijkstra's shortest path algorithm. The approach we decided to take was to create a game that provided users with a fun goal while allowing them to understand the shortest path problem in more detail.

Motivation

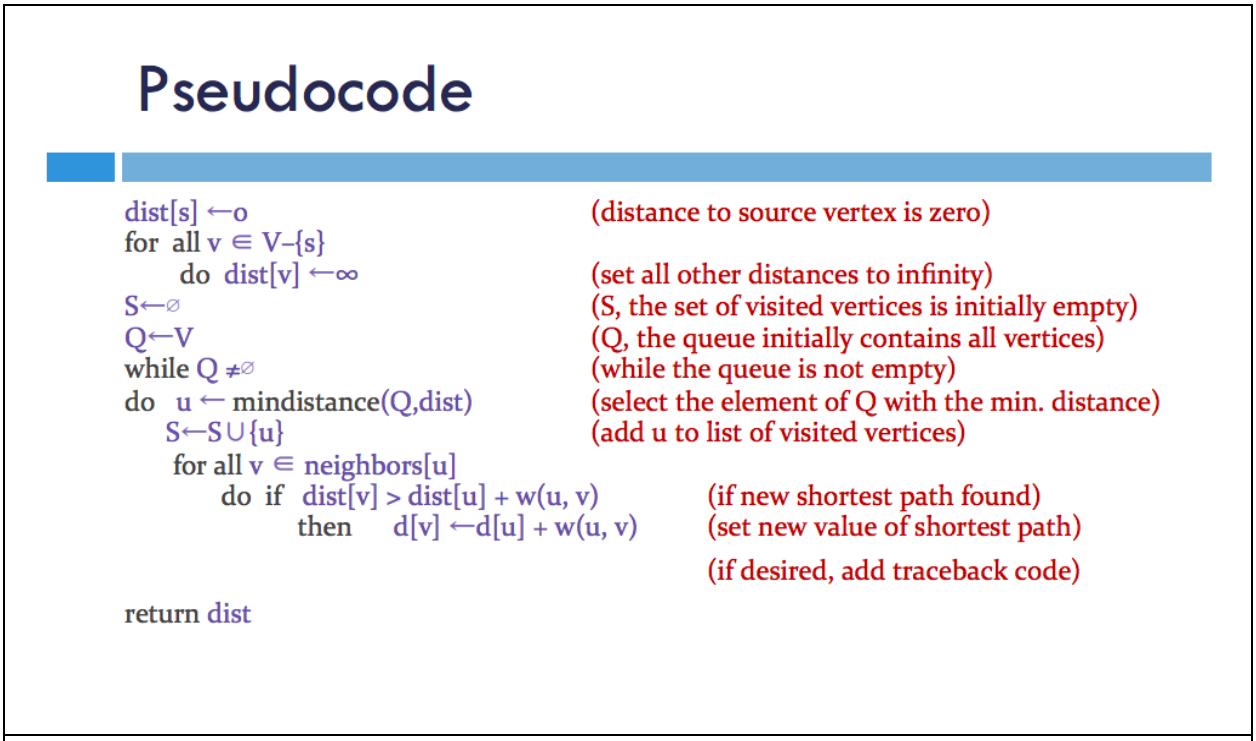
To allow others to learn the shortest path algorithm in an approachable manner.

Background

The algorithm that we used for determining the shortest path of a weighted graph is called Dijkstra's algorithm. Dijkstra's algorithm was first created by Edsger Dijkstra in 1956 [1]. The algorithm is commonly used for routing, which is why we chose it for our game that allows users to predict the shortest route a taxi would take to a single destination point. The city of intersections acts as our weighted graph, and it is a complete graph meaning each pair of distinct vertices has a unique connected edge with weighted values.

Implementation

First, we generate the start and destination vertices dynamically (based on the city's size), as well as the vertices' connected edge weights, to ensure that the shortest path changes each time a user runs a new instance of the game. When the game first compiles, we generate our graph and populate it with vertices linked together by positive weighted edges. We then calculate and store the shortest path for the newly generated grid using Dijkstra's algorithm. Once the user moves the taxi to the destination point, we calculate the length of their path. The user's selected path length is then compared to the length of the shortest path stored at the start of the game to determine if the game has been won or lost.



Pseudo code for Dijkstra's algorithm

```
void computePaths(Vertex source)
{
  source.minDistance = 0.;
  PriorityQueue<Vertex> vertexQueue = new PriorityQueue<Vertex>();
  vertexQueue.add(source);

  while (!vertexQueue.isEmpty())
  {
    Vertex u = vertexQueue.poll();

    // Visit each edge exiting u
    for (Edge e : u.adjacencies)
    {
      Vertex v = e.target;

      double weight = e.weight;
      double distanceThroughU = u.minDistance + weight;
      if (distanceThroughU < v.minDistance)
      {
        vertexQueue.remove(v);
        v.minDistance = distanceThroughU ;
        v.previous = u;
        vertexQueue.add(v);
      }
    }
  }
}
```

Our implementation of Dijkstra's algorithm

Conclusion

Ultimately, our goal was to create a game that would allow users to understand the shortest path problem without having to focus on its intricacies.

References

[1] P. Frana, *An Interview with Edsger W. Dijkstra*, 1st ed. 2001 [Online]. Available: http://conservancy.umn.edu/bitstream/handle/11299/107247/oh330ewd.pdf. [Accessed: 14- Nov- 2014]

[2] Crisler, Nancy, and Walter Meyer. *Shortest Paths*. Lexington, MA: COMAP, 1993. Web.

[3] M., and Melissa. *DIJKSTRA'S ALGORITHM* (n.d.): n. pag. *Math.mit*. Web.