SOFE 3720

Introduction to Artificial Intelligence

Winter 2019, Dr. Sukhwant Kaur
Assignment 1

Deadline: Tuesday, February 12, 2019, 11:59PM

Prepared by

100620049 Alexander Hurst
100617713 Samantha Husack
100618756 Darren Chan

Problem Statement

The city of Oshawa is a city in Ontario, Canada, on the Lake Ontario shoreline. We will be using the characteristics of streets and elevation changes to try to find optimal routes for walking through the city of Oshawa. For the basic underlying structure, will use the A* search algorithm to find the “shortest” path, but we will define our terms for shortest and correlate the heuristics to this definition. In particular, developing a cost function and appropriate (admissible) heuristic given that cost function.

Data Requirements

We will be generating two data files, the first an Open Street Map XML of the routes and the second an elevation map.

The Solution

The solution requires a graphical element, which doubles as an aid for the debugging process. The final solution will be able to take two OSM nodes, in some form and return the shortest walking path between them along with the expected time to walk this path. The path will take into account both elevation changes and the distance between the nodes as part of the calculations.

Implementation

Our algorithm implementation will be utilizing Python 3 as our main programming language. We will use additional libraries for mathematical functions, display, and map rendering.

Team

Member	***Tasks***
Alexander	- Main programmer
Darren	- Research, Report, Analysis
Samantha	- Research, Report, Comparison Program for efficiency

Background

The A* algorithm is used to approximate the shortest path between two nodes in real-life situations, like for example on a map.

Given a 2D grid, with cell A(x,y) as the starting point and cell B(m,n) as the target cell, the algorithm chooses the next node using the lowest value of **f**, being the parameter equal to the sum of the two other parameters, **g** and **h**. We take g to be the movement cost from the current cell to the next and h the estimated movement cost to move from the given cell to the target cell - it is also referred to as the heuristic element.

The algorithm does have its limitations, although its pathfinding abilities are efficient because it relies heavily on heuristics, it cannot always find and produce the shortest path. However, A* is a complete algorithm, that will always find a solution provided one exists.

The A* algorithm can be applied to both graphs and grids. When it comes to a graph, like how the data in the OSM files is presented, the same rules apply. If using a graph with edges, the worst case time complexity becomes O(E), where E is the number of edges on the graph. In an auxiliary space, the worst case scenario presents all the edges inside the open list, and the worst case time complexity is calculated as O(V) where V is the total number of vertices.

Implementation

Algorithm Implementation

Given the provided resources, our team wrote a Python 3 program that implemented the A* algorithm. The algorithm resembles that of Dijkstra's algorithm, with the added heuristic component. The program takes in an Open Street Map file containing route information, as well as a Shuttle Radar Topography Mission Data File as an HGT file, with elevation data.

Our implementation utilizes the horizontal distance between two nodes to calculate the cost. If an elevation change occurs between both nodes, an additional multiplier of 1.25 is calculated on the difference of elevation. This elevation cost is added to the horizontal distance cost to get a total cost between both nodes. The cost calculation is done around the heuristic function cost. The heuristic implementation in our program uses the Euclidean distance between the current node and the end node on the map. With this implementation, our program will not waste time wandering around the map in the wrong direction, and will find a minimal cost path.

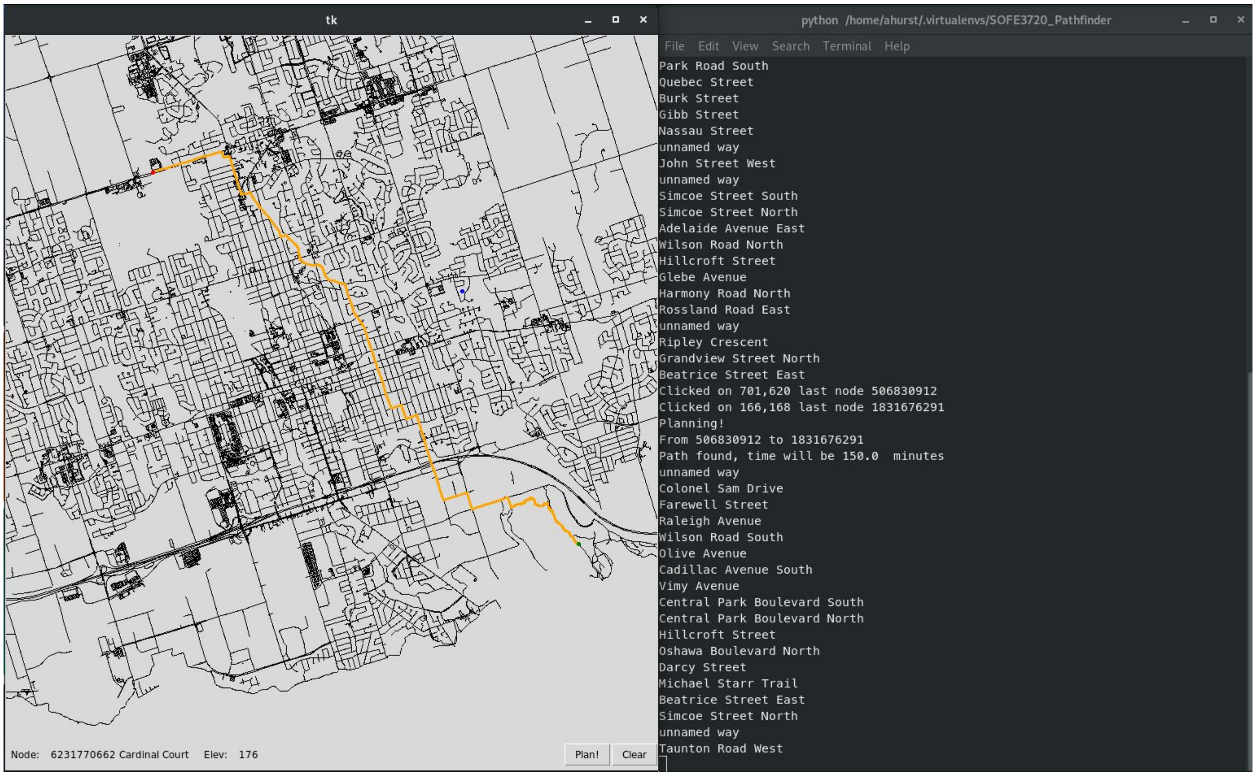
Path Cost Function

The cost of each path is the sum of the individual paths that the A* algorithm takes to the goal, which is $a + 1.25b$ where *a* is our horizontal distance between the two nodes, and *b* is the elevation difference between the two.

Efficiency

Our program runs fairly efficiently. For a long path, our program took around 0.15 seconds to determine a path. Our graphical component is a bit slower, as we have to draw everything out, but is secondary, as the main focus of this assignment is the A* algorithm.

Graphical Outputs



Resources

Patel, A. (2014, July 6). Implementation of A*. Retrieved from https://www.redblobgames.com/pathfinding/a-star/implementation.html

A* Search Algorithm. (2018, September 07). Retrieved from https://www.geeksforgeeks.org/a-search-algorithm/

Haffner, B., & Haffner, B. (2018, April 08). OSMnx Intro and Routing – Bob Haffner – Medium. Retrieved from https://medium.com/@bobhaffner/osmnx-intro-and-routing-1fd744ba23d8

A* search algorithm. (2019, January 09). Retrieved from https://en.wikipedia.org/wiki/A*_search_algorithm

Data Collection

California Institute of Technology. (n.d.). Enhanced Shuttle Land Elevation data. Retrieved from https://www2.jpl.nasa.gov/srtm/

QuickStart with SRTM Data. (n.d.). Retrieved from https://dds.cr.usgs.gov/srtm/version2_1/Documentation/Quickstart.pdf

SRTM3 Data North America. (n.d.). Retrieved from https://dds.cr.usgs.gov/srtm/version2_1/SRTM3/North_America/