Assignment 2: Graph Search

CS 6601

Due September 29 by 9:35 AM

Abstract

You will implement several graph search algorithms with the goal of solving tridirectional search.

1 The Challenge

Graph search is an integral part of AI. You've seen how to find the shortest distance between two points: now, let's try three. One of the trickiest problems that graph search tries to solve is tridirectional search, in which the goal is to find the shortest path between three points. The order of visiting doesn't matter. In this case, we'll be trying to connect three points in the city of Atlanta.

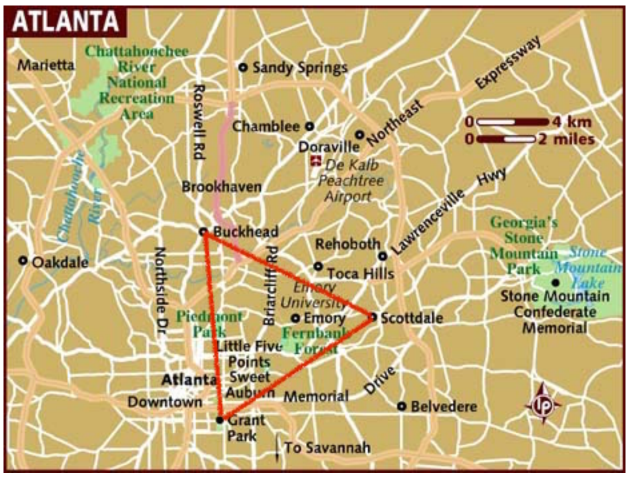

Figure 1: Map of Atlanta
(three points connected)

2 Your Assignment

Your task is to implement several informed search algorithms that will calculate a driving route between three points in Atlanta with a minimal time and space cost.

You will do this in search_notebook.ipynb, and there are tests along the way to help. Your searches should be executed with minimal runtime and memory overhead, as measured by the number of nodes expanded and the amount of time your searches take.

We will provide the following classes and files:

File	Description
search_tests.py	a module containing partial tests for your algorithms
*.pickle	serialized graph files for search tests
visualize_graph.py	a class to visualize search results in Gist

In the pickle files, we've provided you with the graph of the Romania map from Russell and Norvig (Chapter 3) and a graph of the OpenStreetMap of Atlanta for use in testing. Benchmarks for all of the searches you will be implementing have been provided for you for all pairs of nodes in the Romania map and for a subset of the Atlanta map in search_tests.py.

3 Grading

Each section of the assignment is associated with a number of points, as follows (out of 100 points total):

Warmup 1: Implement a priority queue to demonstrate its improvement in performance over an insert-and-sort queue. (5 points)

Warmup 2: Implement breadth-first search. (5 points)

Warmup 3: Implement uniform-cost search. (10 points)

Warmup 4: Implement A* search with a corresponding admissible heuristic. (10 points)

Exercise 1: Implement bidirectional uniform cost search. (15 points)

Exercise 2: Implement bidirectional A* search. (20 points)

Exercise 3: Implement tridirectional uniform cost search. (20 points)

Exercise 4: Implement an improvement on tridirectional search. (15 points)

4 Race!

As long as you meet the above requirements, you're free to implement a custom search method for use in an extra activity. We'll have you race your classmates to connect three new points (with limited time for preprocessing - details to come) and report the winners along with their search strategies.

We will test each agent by searching for paths to a series of random locations in Atlanta. The fastest agents will receive the following rewards as a bonus:

1. First place: 3 points on your final grade
2. Second place: 2 points on your final grade
3. Third place: 1 point on your final grade

Ties, if any, will be broken by assignment submission time.

5 Due date

This assignment is due on T-Square Tuesday September 29th by the start of class (9:35 AM). The deliverables for the assignment are:

- A filled out version of the iPython notebook provided. (search_notebook.ipynb)

6 Resources

If you want to know how the OpenStreetMap data works, check out their wiki.

The software requirements are listed in search_notebook.ipynb. You'll want to skim the networkx documentation before starting.

As always, TAs will hold office hours Monday, Tuesday, Thursday and Friday from 2:00 to 4:00 PM outside TSRB 241.