CS141 – Intermediate Algorithms and Data Structures
Assignment 2 – All Pairs Shortest Path

Anthony Martinez

5/27/15

Abstract

We will be comparing the run time of two separate algorithms for determing the shortest distance between all pairs of vertices in a graph. The first algorithm is Bellman-Ford's, which was originally developed for finding the shortest path between a single source vertex and all other vertices. We will extend this algorithm to find all pairs shortest paths. The next algorithm is Floyd-Warshall's, which was created to solve this problem faster than the extended Bellman-Ford algorithm.

1 Assignment

- Implement Bellman-Ford's algorithm to find the length of the path between a source (s) and all other vertices.
  - Extend Bellman-Ford's to find the length of the paths between all pairs of vertices.
- Implement Floyd-Warshall's algorithm to find the length of the paths between all pairs of vertices.
- Calculate the run-time of all of the implemented algorithms.
- Run all of the benchmarks on both "all pairs shortest path" algorithms.
- Fill in the report analyzing the algorithms and their run-time.
- You may remove the Assignment section (section 1) before turning in the final report

2 Introduction

- What is the problem that you are solving?

  Solving the problem of finding the shortest path between all pairs of points in a given graph, passed in as a file with info about the vertices and edges for the graph, using two known algorithms for doing so and comparing the resulting run times of those algorithms.

- What methods are you going to use to solve the problem?

  The solution will involve impliment ing both algorithms and running them on the same data sets while a timer collects information about the duration of the functions. The algorithms are the Bellman-Ford and Floyd-Warshall algorithms for finding the shortest path in a graph.

- Why are these good methods to use?

  They can handle negative edge weights and/or detect them in order to apropriately handle them.

- Why are you going to be using both of them?

  We use both to compare the run times and see for which types of situations is one better than the other.

3 Bellman-Ford

- What is the Bellman-Ford algorithm?

  Does Edge relaxation $|V| - 1$ times values for distance shold converge to actual minimum distance between two vertices. Does one more time. If values change. There is a negative cyle in the graph.

- Why are you using it?

  It allows us to solve for the minimum distance between a vertex to all other vertices. A slight adjustment would allow it to be used for our intended purposes.

- How did you adapt it to work for all-pairs as opposed to single source?

  Put the single source version in a loop that iterates through all vertices. Updating the results as necessary.

- What is the run-time of the algorithm before and after your adaptation?

  The theoretical run time before the adaptation is $O(|V|*|E|)$ and the runtime after is $O(|V|^2*|E|)$

4 Floyd-Warshall

- What is the Floyd-Warshall algorithm?

  A dynamic programming algorithm that finds the shortest distance between every pair of vertices in a graph by comparing the value of the distance using one path to the value if another path was used, updating the values accoringly.

- Why are you using it?

  It finds the shortest path between all pairs of points in a graph.

- How is it better than the Bellman-Ford algorithm?

  It already solves our problem without modification.

- What is the run-time of the algorithm?

  The theoretical run-time of the algorithm is $O(|V|^3)$

5 Results

- Compare and contrast the two algorithms? What makes one more suited for this problem?

  Both the Floyd-Warshall and Bellman-Ford can detect negative cycles and neither one can find the shortast distance of one exists. The FW algo is able to find the shortest path between all vertices without modification where as the BF requires some modification to be able to handle this problem. The run time are different but if the graph in question has more vertices than edges the BF is preferred. If the graph contains alot of edges on the other hand the FW is the preferred algorithm according to run times.

- What are their theoretical run-times (from the previous sections) and how do they compare? The bellman-Ford theoreticly runs in $O(|V|^2 * |E|)$ for this type of problem and the Floyd-Warshall algorithm runs in $O(|V|^3)$

- What are the actual run-times that you computed? Which method is better? Why?

  For these problems the Floyd-Warshall is actually better since for every examle there is actually more edges than vertices which benefits FW algo since it doesnt depend on the number of edges and the BF algo does and for the BF the number of edges is actually the dominant term.

- Fill in table 1 with your results

6 Conclusions

- What did you find difficult about the assignment?

  The implimentation was easy. The difficult part was the python syntax and organizing the data for the graph to be used in the algorithms.

Table 1: Run-Time Comparison

Benchmarks	# Edges	Bellman-Ford Actual	Floyd-Warshall Actual
input4.txt	5	0.000111	1e-06
input5.txt	8	0.000203	2e-06
input10.txt	16	0.00158	1e-06
input25.txt	43	0.024964	1e-06
input50.txt	93	0.208175	1e-06
input100.txt	192	1.857163	3.99999999989e-06
input250.txt	730	41.177519	3.99999999701e-06
input500.txt	1532	339.405867	3.9999999899e-06
input1000.txt	2985	2633.958733	5.99999998485e-06

- What did you learn?

  The running time of algorithms can depond on a number of factors and there might not only be a single solution to a problem. Though some solutions are better than others depending on the problem.

- What is one real-world problem that you think each of these problems would be good at solving?

  The Bellman-Ford algorithm can be used to find the closest Mcdonalds to my given location and the Floyd-Warshall can be used to find closest neighboring cities and building maps.