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?

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

- Why are these good methods to use?

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

3   Bellman-Ford

- What is the Bellman-Ford algorithm?

- Why are you using it?

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

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

4   Floyd-Warshall

- What is the Floyd-Warshall algorithm?

- Why are you using it?

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

- What is the run-time of the algorithm?

Table 1: Run-Time Comparison

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

5   Results

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

- What are their theoretical run-times (from the previous sections) and how do they compare?

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

- Fill in table 1 with your results

6   Conclusions

- What did you find difficult about the assignment?

- What did you learn?

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