Assignment: We Are All Connected

Avraham Leff

March 9, 2023

1 Assignment-Specific Packaging

The general packaging is unchanged from the basic "Homework Policies" document (see Piazza posting).

This assignment's "DIR" **must be named** *WeAreAllConnected*; the write-up file **must be named** `WeAreAllConnected.pdf`.

2 Motivation

The purpose of this assignment is to give you more practice modifying classic algorithms to solve (slightly) different requirements; also to use your order-of-growth knowledge to analyze algorithm behavior.

3 Background

Your client is a state that developed a communications system which connects n cities using point-to-point communication links. Each segment in the system is associated with a *duration*: the amount of time for a message to travel from one end of the segment to the other. Duration isn't affected by the "direction" in which the message travels along a segment.

As a good state should, your client wants to make its citizens ever more happy. To that end, it is considering a set of possible new segments that it will add to the current system. Given budget considerations, **only one segment** from the set of possibilities will be selected and deployed.

You propose various selection criteria, but your client is firm: the selection criterion is the segment that *most improves the current total duration*. Total duration is defined as the *sum of durations between all pairs of cities*.

This criterion is motivated by *fairness*: "no two cities are more important than any other two cities, and the amount of actual communication traffic between cities is irrelevant".

3.1 The Problem

Develop an algorithm that solves the above problem.

3.2 My Expectations

In lecture (last semester) we touched on a classic algorithm that is highly relevant to this problem (and you may research the space of such algorithms for this assignment). I consider that algorithm to be the basis of a "brute force" solution to this problem.

I'm expecting you to provide a solution that is an improvement over the "brute force" solution. However: **I don't require an algorithm that is an improvement from an order-of-growth perspective**! It suffices for your algorithm to improve performance from a "lower-order terms" perspective.

4 Requirements

The assignment has two components: a programming component (70%), and a write-up (30%).

4.1 Write-Up

Supply, in the **following order**:

1. How does your algorithm "model" the problem? Does this "modeled problem" have a name?
2. Describe (ideally by reference to a classic algorithm) a "brute force" algorithm that solves the problem. In formal terms, what is its order of growth?
3. Describe your "better" algorithm that solves the problem. Why is it an improvement of the "brute force" algorithm? In formal terms, what is its order of growth?

No more than one-and-a-half pages! Don't hesitate to use diagrams if they help with your explanation.

4.2 Programming Component

Please review the general requirements for a programming assignment! I've tried to reduce the chances of "mistakes" occurring through the use of a "skeleton class", but ultimately, **you are responsible** for ensuring that I can compile and test your code without incident.

Begin by downloading the code from the *WeAreAllConnected* directory in this git repository. Then implement the API and compute the required doubling ratio.

Suggestion: consider constructing your own customized data-structures rather than something with more features than you really need.

4.2.1 Sample API Usage

```java
import edu.yu.da.WeAreAllConnected;
import edu.yu.da.WeAreAllConnectedBase;
import static edu.yu.da.WeAreAllConnectedBase.SegmentBase;
import static edu.yu.da.WeAreAllConnected.Segment;

final WeAreAllConnectedBase waac = new WeAreAllConnected();
logger.info("n = {}", n);
logger.info("current={}", Arrays.toString(current));
logger.info("possibilities={}", Arrays.toString(
   possibilities));
final SegmentBase retval =
  waac.findBest(n, Arrays.asList(current), Arrays.asList(
      possibilities));
logger.info("Implementation returned {}", retval);
return retval;
```