Assignment: Using Network-Flow To Relieve A Famine

Avraham Leff

April 27, 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** *OverfullGranaries*, and your file **must be named** *OverfullGranaries.pdf*

2 Motivation

This assignment will give you more experience with:

- Using network-flow graphs to model (and solve) "real-world" problems
- Getting "hands-on" experience solving network-flow problems through a concrete implementation.

3 The Problem

> As usual, the requirement details are specified in the API (see below): this section is only intended to motivate the problem at a high level.

You're involved with a crucial project to supply grain to a country experiencing a major famine. Currently, there are x overfull granaries and y

underfull granaries. The project manager, *Yosef*, wants to move **as quickly as possible** a total of $10,000$ bushels from the overfull granaries to the underfull ones. He tells you that he doesn't care how much of the grain comes from each of the x individual granaries: all that matters is that the total number moved is $10,000$. In addition, *Yosef* tells you that he doesn't care how much grain arrives at each of the y granaries: all that matters is that the total number that arrives is $10,000$.

The good news: you have a map of one-way roads connecting the granaries in the form of a directed graph. Each road has a maximum capacity in *"bushels per hour"*.

> Note: you've seen the project spread-sheet, the x granaries have $\geq 10,000$ bushels between them. The issue is how to get the surplus moved as **fast as possible**.

Your task: find a way to deliver the grain such that it takes the **minimum amount of time**.

Specifically:

- Determine how many bushels will be delivered on which roads and
- How much time the end-to-end delivery will take

4 Requirements

> You **must solve** this problem using a *network flow graph* technique.

This assignment contains both non-programming and programming components.

4.1 Non-Programming Work

This component is weighted as 50% of your grade.

Your writeup file must contain, in the following order, the following sections.

1. I've already stated that this problem must be solved as an *Network Flow* problem. Explain how you propose to map this problem to a *Network Flow* model.

   If anything is missing from a straightforward reduction to an *Network Flow* problem, be sure to state what is missing explicitly!

   (No more than three sentences!)

2. If your reduction of the problem to an *Network Flow* model requires something(s) "extra", state how you supplied the missing pieces.

   (No more than five sentences!)

   You **must also** draw the corresponding network-flow graph for a small, but not trivial, sample problem: it must clearly show all relevant portions of the reduction to the *Network Flow* model.

3. Prove that your algorithm is correct.

   For *Network Flow* problems, the proof is usually just

   - Showing that your reduction to an *Network Flow* problem respects this problem's requirements and characteristics (you should not prove the correctness of the "basic" *Network Flow* algorithm(s))!
   - Showing that any non-standard modeling in your reduction doesn't violate the *Network Flow* algorithm's requirements

Your write-up **cannot exceed** 1.5 pages of size 11 font print. Diagrams are welcome!

4.2 Programming Work

This component is weighted as 50% of your grade.

4.2.1 Programming A Solution

> 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 `OverfullGranaries.java` (in the *OverfullGranaries* directory) from this git repository. Then, implement the stubbed methods.

The following test code snippet should guide you as to how I will invoke your code.

```
final String[] X = {"One"};
final String[] Y = {"Two"};
final OverfullGranaries ofg = new
   OverfullGranaries(X, Y);
ofg.edgeExists("One", "Two", 10);
final double actualValue = ofg.solveIt();
final List<String> actualMinCut = ofg.minCut();
```

- You may (encouraged to) "research" the concept of "network flow" algorithms in general.
- You may not "research" this specific problem in any way!
- **For this assignment only**, you may copy and use any generic *Network Flow* code in your implementation. If you do so, you **must add** (in your writeup) a section that states your code source(s).

  You may only use code written by yourself (and the JDK) to implement the specific *Network Flow* solution for this assignment.