Programming Assignment 2: Disaster Relief

Specification

(This assignment was partially inspired by Keith Schwarz's 2020 Nifty Assignment.)

Background

When natural disasters strike, governments, relief organizations, and even individual donors must often wrestle with how best to allocate available resources to help those who have been affected. This is generally a very complex decision, balancing countless logistical, economic, political, and other factors. One particular challenge is that travelling between geographic areas affected by the disaster can require different financial or other resources for relief. Organizations sometimes have to make difficult decisions in the hope of helping as many people as possible with the available resources.

In this assignment, you will implement a system to determine how to chart a path through a group of affected regions to help as many people for as little cost as possible.

NOTE: While our simulation will focus on helping the greatest number of people for the least amount of money, this is an oversimplification of the problem of allocating resources in the wake of a disaster, and may not necessarily be the best approach.

Learning Objectives

By completing this assignment, students will demonstrate their ability to:

- Define a solution to a given problem using a recursive approach
- Write functionally correct recursive methods
- Produce clear and effective documentation to improve comprehension and maintainability of a method
- Write methods that are readable and maintainable, and that conform to provided guidelines for style and implementation

System Structure

In our system, we model a partially-connected collection of geographic *regions,* each of which has a population of people who need help within it. Some regions are connected to each other, and there is a cost associated with travelling between those regions. (You can imagine that this cost is related to factors like the distance between the regions and the difficulty of travelling that distance.) Other regions are not connected to each other and relief efforts cannot travel between those regions. (This might be, for example, because there is an impassable geographic feature blocking travel or because the relief organization simply does not have the capability to traverse that distance, such as not having access to air or sea travel.) For our system, we will assume that all connections and costs are symmetric. (That is, the cost to get from region A to region B is the same as the cost to get from region B to region A. If region C is not connected to region D, then region D is also not connected to region C.)

Our goal is to identify a *path* through these regions (from a specified starting point) that will reach and help as many people as possible for the lowest cost. Ideally, we will want to find a path that reaches **all** regions, however that may not be possible. If it is not, we will seek the path that **helps the most people**, not the path that visits the most regions.

⚠ Note that populations are associated with *regions*, but costs are associated with travel *between regions*.

`Region` class

In our system, we will represent areas that may be helped with the following `Region` class (comments and some methods are omitted here; see the full `Region` class in the coding challenge slide for these):

▶ Expand

`Path` class

We will represent a path of regions that will receive resources with the following `Path` class (comments and some methods are omitted here; see the full class in the coding challenge slide for these):

▶ Expand

The two methods `extend` and `removeEnd` can be used to add/remove a `Region` to/from a `Path`. These are the only ways to modify a `Path` -- you cannot add or remove a `Region` to or from the beginning or middle of a path.

Notice that these methods *return a new* `Path` rather than modifying an existing `path`, similar to how `String` methods like `substring` or `toUpperCase` return a new `String` rather than modifying an existing one:

```
Path empty = new Path();
Region one = new Region("Region #1", 50);

Path added = empty.extend(one);
Path removed = added.removeEnd();
```

Make sure you write your implementation accordingly.

Required Methods

For this assignment, you will implement only a single method:

```
public static Path findPath(List<Region> sites)
```

This method takes a list of `Region` objects as a parameter and will compute and return the path through these regions that will result in the most people being helped. If there are multiple paths that result in the most people being helped, return the path that has the lowest cost. If there are multiple paths that help the most people *and* share the same lowest cost, you may return any of these paths. In each path you consider, you should visit each of the sites **at most once**.

For simplicity, you should only consider paths that **start with the first `Region` (the region at index 0) in the given list of regions**. You can think envision this as our organization currently being located at the initial location, and they're interested in figuring out the best route to take from this starting place.

If `sites` is `null`, an `IllegalArgumentException` should be thrown. If `sites` is empty, you should return `null` to indicate that there is no valid path.

You should implement your `findPath` method where indicated in the provided `Client.java` file. You may also implement any additional helper methods you might like. (For example, you will likely want to implement a public-private pair in your algorithm.)

⚠ **WARNING:** We are **requiring** that you do not use the `addConnections()` method within findPath. The `addConnections()` method should only be used for testing your code

Client Program

We have provided a client program that will allow you to test your `findPath` implementation. This client provides two methods that might be useful.

```
public static List<Region> createSimpleScenario()
```

- Manually creates a simple list of regions to represent a known scenario.
  - We have provided one example in the client code, and a few others in the examples below.

```
public static List<Region> createRandomScenario(int numRegions, int minPop, int maxPop, double minC
```

- Creates a scenario with `numRegions` regions by randomly choosing the population, connections, and costs of connections for each region.
  - Populations will be chosen between `minPop` and `maxPop` (inclusive)
  - Costs will be generated by choosing a random value between `minCost` and `maxCost`

(inclusive).
  - Each region will be connected to a random subset of the other regions.
    - Regions are connected both ways. (ex: if Region #1 is connected to Region #2, then Region #2 is connected to Region #1).

You can modify `createSimpleScenario` with different `Region` objects to test your implementation in scenarios of your own design, and/or you can generate random scenarios to try using `createRandomScenario`.

Click "Expand" below to see some example scenarios, their results, and visualizations of ***exploring*** the different possible paths (note that the diagrams do not show the process of choosing the "best" path).

▶ Expand

▶ Expand

▶ Expand

You may create your own client programs if you like, and you may modify the provided client if you find it helpful. However, **your methods must work with the provided files without modification and must meet all requirements below**.

Testing Requirements

For this assignment, you'll be required to implement **three** total JUnit tests. The first two should cover the following cases:

▶ Expand

▶ Expand

The third test should be a test case you come up with on your own. **Our requirement for this third test is that the inputted `sites` contains at least four regions and that there are at least three unique connections between regions.**

All three tests should be placed in their **own methods** within the provided `Testing.java` file. You're welcome to implement tests other than the ones outlined here, but doing so is not required.

Implementation Requirements

To earn a grade higher than N on the Behavior and Concepts dimensions of this assignment, **your algorithm must be implemented *recursively*. You will want to utilize the *public-private pair* technique discussed in class.** You are free to create any helper methods you like, but the core of your algorithm (specifically, building and potentially evaluating possible allocations of relief funds) must be recursive.

Additionally, for this assignment, you should follow the Code Quality guide when writing your code to ensure it is readable and maintainable. In particular, you should focus on the following requirements:

- Avoid recursing any more than you need to. Your method should not continue to explore a path if the current `Path` is no longer viable.
- Watch out for branches of an `if`/`else` statement that shares the same exact code. You should combine the conditionals and write the code only once.
- Make sure that all parameters within a method are used and necessary.
- You should comment your code following the Commenting Guide.
    - Make sure to avoid including *implementation details* in your comments. In particular, for your object class, a *client* should be able to understand how to use your object effectively by only reading your class and method comments, but your comments should maintain *abstraction* by avoiding implementation details.
    - Continuing with the previous point, keep in mind that the client should **not** be aware of what implementation strategy your class/methods utilize.
- All methods present in your class that are not listed in the specification must be private.

Disaster Relief

Download Starter Code

P2_DisasterRelief.zip

Remember that you're required to write three tests, each within their own method in `Testing.java`. More information on this requirement can be found in the spec.

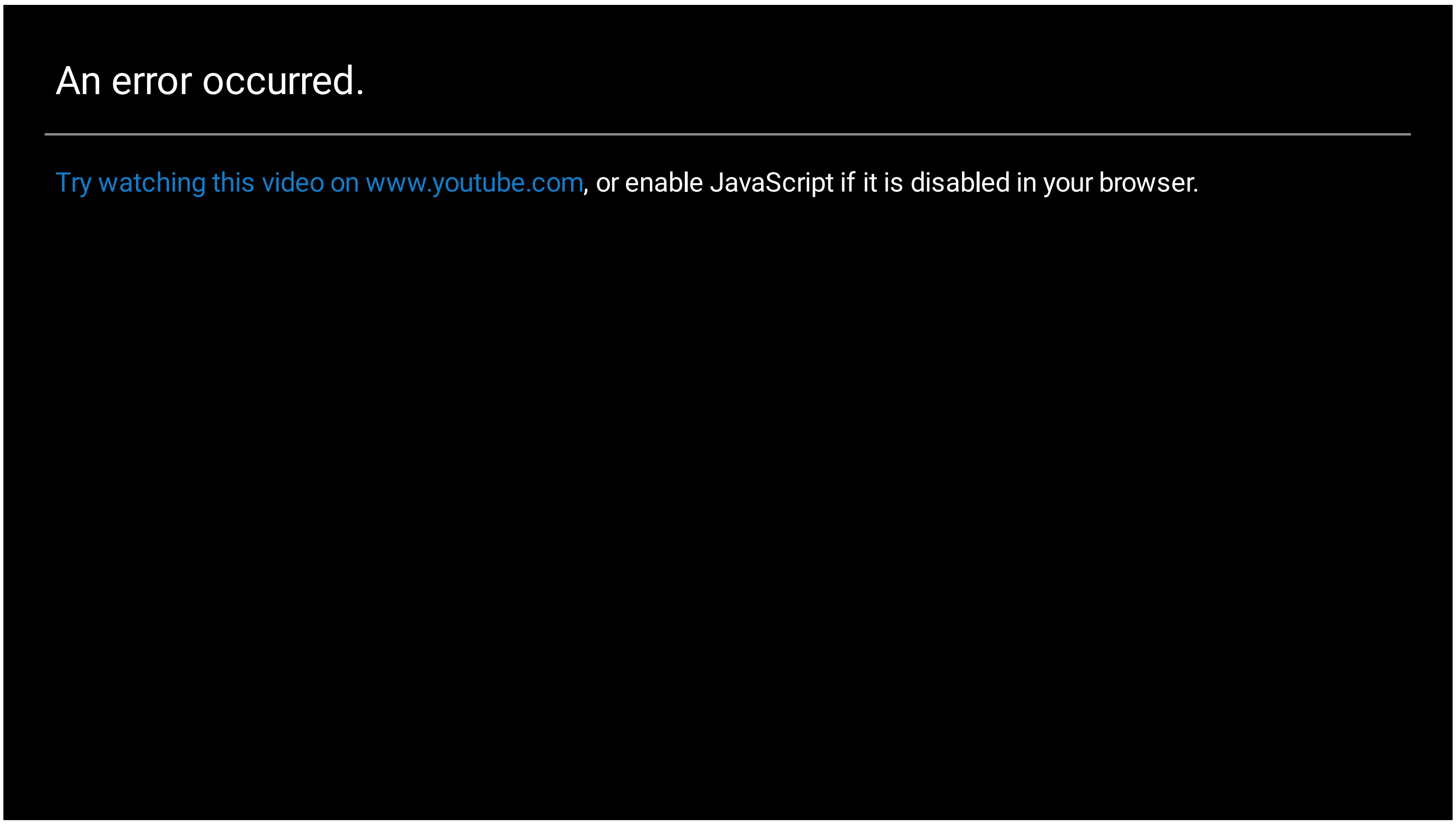
Reflection

The following questions will ask you practice **metacognition** to reflect on the topics covered on this assignment and your experience completing it. For each question, focus on your plan and/or process for working through the assignment along with the CS concepts. Think about things like how you organized your working time, what sorts of things tended to go wrong, and how you dealt with those errors or mistakes.

Please answer all questions.

Question 1

The first 3 questions will require you to reflect about potential benefits and drawbacks in employing algorithms to improve societal welfare. Start by watching a short segment of the following talk from UC Berkeley professor Rediet Abebe (2m 9s to 8m 57s):



An error occurred.

Try watching this video on www.youtube.com, or enable JavaScript if it is disabled in your browser.

(https://youtu.be/h1NqpK4gDrM?t=129)

How do you think measurement challenges (such as data sparsity and inaccurate metrics) affect the effectiveness of algorithms in resource allocation for disaster relief?

No response

Question 2

What are the potential risks of relying on simple metrics (like income or population density) when allocating disaster relief resources, and how might these risks be mitigated?

No response

Question 3

Explain what income shocks are. How might similar 'shocks' or unforeseen events affect disaster relief efforts, and how could an algorithm be designed to handle these sudden needs?

No response

Question 4

Describe how you went about testing your implementation. What specific situations and/or test cases did you consider? Why were those cases important?

No response

Question 5

What skills did you learn and/or practice with working on this assignment?

No response

Question 6

What did you struggle with most on this assignment?

No response

Question 7

What questions do you still have about the concepts and skills you used in this assignment?

No response

Question 8

About how long (in hours) did you spend on this assignment? (Feel free to estimate, but try to be close.)

No response

Question 9

Was any part of the specification or requirements unclear? If so, which part(s), how was it unclear, and how could it have been made more clear?

No response

Question 10

[OPTIONAL] Do you have any other feedback, questions, or comments about this assignment?

(Note that we may not be able to respond to questions here, so please post on the message board if you would like a response!)

No response

󰀀 Final Submission 󰀀

󰀀 Final Submission󰀀

Fill out the box below and click "Submit" in the upper-right corner of the window to submit your work.

Question

I attest that the work I am about to submit is my own and was completed according to the course Academic Honesty and Collaboration policy. If I collaborated with any other students or utilized any outside resources, they are allowed and have been properly cited. If I have any concerns about this policy, I will reach out to the course staff to discuss *before* submitting.

(Type "yes" as your response.)

No response

[SCAFFOLD] Disaster Relief

Download Starter Code

P2_DisasterRelief.zip

Remember that you're required to write four tests, each within their own method in `Testing.java`. More information on this requirement can be found in the spec.

Visualizations

https://docs.google.com/presentation/d/1_TacWWKGg_RrQiUUoRnVcmIdWbugFU2pPMPriZZ9HBM/edit?usp=sharing