CSCI 3901: Project: High-Level Analysis

Waleed R. Alhindi (B00919848)

Problem Overview:

The project involves the implementation of a program that simulates Nova Scotia Power's management and reporting system. Namely, the program is used as an interface to store, manage, and report information about how power is being distributed throughout the province and the impact of power outages. Postal codes define the province's composite regions and denote the population within that area. Each province is serviced by one or more power distribution hubs which are represented as a UTM coordinate. By providing this information as input, the program will map postal codes to distribution hubs, defining how many people the hub services. Additionally, in the case of outages, the program stores the amount of time needed to repair distribution hubs. Using these pieces of information together, the program will calculate the impact of power outages as the number of people effected by damaged hubs and will predict the restoration plan using the information provided about each damaged hub's location (UTM coordinate), repair time, and the number of people who will get their power back by repairing it. Additionally, since the program maps power hubs to postal codes, it will also be able to report postal codes that are being underserved by population and/or area, which gives NS Power insight into where new power hubs should be placed.

Data:

To facilitate its operations, the program will need data about each postal code and distribution hub. Additionally, it will need information about how these two are linked. Thus, the minimum amount of data needed is as follows:

- Postal codes:
  - Their identifiers (i.e., B3J 1H6)
  - Their populations
  - Their areas

- Distribution hubs:
  - Their identifiers
  - The postal codes they service
  - Whether they are in-service:
    - Number of people effected by their outages
    - Time needed to repair them

Using these pieces of information, the program should be able to calculate the number of people effected by outages, the optimal route to restore distribution hubs, and the rate at which people get their power back.

Additionally, this data needs to survive between program executions. This means that the data must be stored in some external resource like a database or file. In the case of a database, a relational database would provide a convenient way of linking postal codes and distribution hubs

but will still need a data structure to store, organize, and operate on the fetched data. One benefit provided by using a relational database is being able to use structured queries to quickly retrieve related pieces of data. For example, instead of implementing an algorithm to retrieve the hubs in a particular radius and the postal codes they service, the program can just issue an SQL query and have the database, which is optimized to do such retrievals, handle it. On the other hand, an external file can be used to store and fetch serialized data structures between executions. However, this means that the program must fetch that data each time it starts execution and must update the file frequently to reflect changes made to the data. This introduces overhead in the form of frequent I/O calls during execution. However, this approach gives the program full control over how it structures its data since it is not constrained by the rules, syntax, and logic of a database.

Algorithms:

By examining the project's problem, we can preliminarily identify the key algorithms that need to be implemented. Although it is almost certain that additional support algorithms need to be implemented, the following is an outline of the fundamental algorithms the program needs to implement:

- An algorithm that adds a postal code to the existing pool of postal codes. This also involves checking whether that postal code already exists.

- An algorithm that adds a distribution hub to the existing pool of distribution hubs. This also involves checking whether that distribution hub already exists. Additionally, this algorithm must also link the newly added hub to the postal codes it services. Thus, it must also check that the postal codes associated with it exist and, if they do, it must create a link between them.

- An algorithm that updates a distribution hub's status and repair time. This algorithm is key in reporting damages and tracking repair progress. Consequently, it sets up the data needed to calculate the number of people effected by outages which dictates the optimal restoration plan in additional to calculating the most damaged postal codes based on the time needed to repair each hub that services it.

- An algorithm to calculate the total number of people effected by the collective outages which is also used as a part of calculating the most impactful hub outages. This is also essential in determining which postal codes are underserved by population and when calculating the ideal rate of service restoration.

- An algorithm that will calculate the most underserved areas. This is one of the simpler algorithms as it just involves retrieving each postal code and dividing their areas by the number of hubs that service them, then comparing these values to find the postal codes that have larger areas per hub than others. (This can also be calculated by dividing the number of hubs that service each postal code with its area and comparing the values to find the smallest hub per area value).

- An algorithm that determines the optimal order of fixing hubs in a given area range. This involves determining the impact of restoring the hubs in that range and routing a monotone path between the starting and ending hub, where that route will also stop by and repair the most optimal set of hubs between the two in an x or y monotone path. This algorithm will be the most difficult to implement as it involves many moving parts and can plausibly involve state space exploration in determining the set of intermediate hubs that yield that highest benefit when restored.

Output:

Using the data and algorithms described above, the program's reporting methods output the following data:

- The number of people effected by hub outages.
- A descending list of the most damaged postal codes based on the time needed to fully repair all the hubs that service the postal code.
- A descending list of the most impactful hub outages based on the number of people who will regain power as it is repaired.
- An ascending list of repair times that correspond to the number of hours needed to increment the percentage of people without power.
- An order list of damaged hubs within a given area range that denote the optimal path to take between them to restore power to the most amount of people in the allotted time.

Target Environment:

The program will be tested on a Linux server in the form of either Timberlea or Bluenose. Thus, it will not include a user interface and must not utilize packages that the server might not have. Additionally, this means that it should be regularly run on the Linux server to identify environment-specific bugs during development. This is important because some Java methods, especially those that involve I/O, have some quirks that are specific to Linux servers that need to be accounted for. For example, during assignment 1, I found that the Scanner class needed some extra parameters to correctly read files on a Linux server.

Target Users:

Since this is a course project, the users will primarily be the course's instructor and TA. Furthermore, the program's methods will most likely be run through Junit tests, meaning that the return values of the methods will be examined rather than any printing or UI.

Assumptions:

The project instructions grant the assumption that postal codes will not overlap, and that postal code will not be deleted. However, some preliminary implicit assumptions can be made (note that items in blue need further clarification):

- Adding an existing postal code in addPostalCode will return false.
- Passing a population of zero or less in addPostalCode will return false.
- Passing an area of zero or less in addPostalCode will return false
- Adding an existing distribution hub in addDistributionHub will return false.
- Passing an empty set as servicedAreas in addDistributionHub will return false.
- Passing at least one postal code that does not exist in servicedAreas to addDistributionHub will return false.
- Passing a hub that does not exist in hubDamage will throw an exception.
- Passing a repair time of zero or less in hubDamage will throw an exception.
- Passing a hub that does not exist in hubRepair will throw an exception.
- Passing a hub that is not damaged (has no repair time left) in hubRepair will throw an exception.
- Passing a repair time that exceeds the remaining repair time of a hub in hubRepair will throw an exception.
- The inService parameter should only be supplied as true if the repair time supplied would reduce the hub's remaining repair time to 0 in hubRepair, else it will throw an exception.
- Ties at the limit of mostDamagedPostalCodes list should be added to the list as was done in assignment 3's busyClasses.
- Ties at the limit of the fixOrder list should be added to the list as was in done in assignment 3's busyClasses.
- An increment of 0 or less in rateOfServiceRestoration will throw an exception.
- Calling rateOfServiceRestoration when all hubs are in-service will return an empty list.

No assumptions can be made about repairPlan yet, but uncertainties and assumptions will very likely be identified during design and implementation.