Project Scope and Plan: Library System

Fausta and Druhi

November 18, 2024

1 Project Scope

The aim of this project is to design and develop a food resource map/system for the Greater Boston area. The system will take different factors such as transportation and grocery stores in the area, distribution center proximity, etc. in account. The projects main objectives are:

- use stats of Boston Neighborhoods to understand food allocation/ accessibility
- possible statistics:
  - number of Soup Kitchens / Boston Eats (Data 2019)
  - redlining grade
  - median income
- provide a path for mobile market to travel in order to allocate food
-

The expected outcomes include a functioning food recourse map, with the user being able to access a "best route" for mobile food recourse markets to follow. It can consider a user input for starting location (with a default starting location if the user opts out), and then given statistics to determine the following route.

2 Project Plan

2.1 Timeline

The overall timeline for the project is divided into phases:

- **Week 1 (October 7 - October 13)**: Define project scope, establish team roles, and outline skills/tools.
- **Week 2 (October 14 - October 20)**: Begin development, set up the project repository, and start coding basic system functionalities.
- **Week 3 (October 21 - October 27)**: Continue coding, work on backend integration, and start writing technical documentation.
- **Week 4 (October 28 - November 3)**: Complete the backend. Begin PowerPoint presentation.
- **Week 5 (November 4 - November 10)**: Finalize the system, conduct testing, and continue with the report.
- **Week 6 (November 11 - November 17)**: Revise and finalize the technical report and the PowerPoint presentation.
- **Week 7 (November 18 - November 28)**: Final presentation, report submission, and project closure.

2.2 Milestones

Key milestones include:

- Project Scope and Plan (October 7).
- GitHub Repository Setup and Initial Development (October 9).
- Backend Completion (October 28).
- Final System Testing and Report Draft (November 10).
- Final Presentation and Report Submission (November 28).

2.3 Team Roles

- **Druhi:** Backend developer, develop data structures and Class functionality
- **Fausta:** Backend developer, develop location pinpointing system and ranking algorithm
- **Collectively:** Documentation and report writing, including Overleaf and GitHub management. Testing and quality assurance of the library system.

These roles are flexible and may change as the project progresses.

3 Team Discussion Summary (October 7)

3.1 Skills Assessment

Team members need the following skills to complete the project:

- Familiarity with C++
- Familiarity with GitHub for version control.
- Use of Overleaf for writing the final technical report.

3.2 Tools & Technologies

- **Programming Languages:** C++
- **Database:** Researching potential databases to utilize
- **Collaboration Tools:** GitHub for version control, Overleaf for report writing.

3.3 Team Responsibilities

Each team member will focus on specific areas, but roles may adapt as the project progresses:

- Backend Development.
- Technical Documentation.
- System Testing.

4 Skills & Tools Assessment (October 8)

4.1 Skills Gaps & Resource Plan

We identified a few gaps in backend programming and handling databases/how to apply them to our project. To address these, we plan to research potential tools and methods to implement data found online.

4.2 Tools

We will use the following tools:

- C++ for backend development.
- GitHub for version control and collaboration.
- Overleaf for the technical report.

5 Initial Setup Evidence (October 9)

5.1 Project Repository

The project repository has been created on GitHub, accessible by all team members. The repository can be found at `https://github.com/druhib/algoproject`.

5.2 Setup Proof

The development environment has been successfully set up.

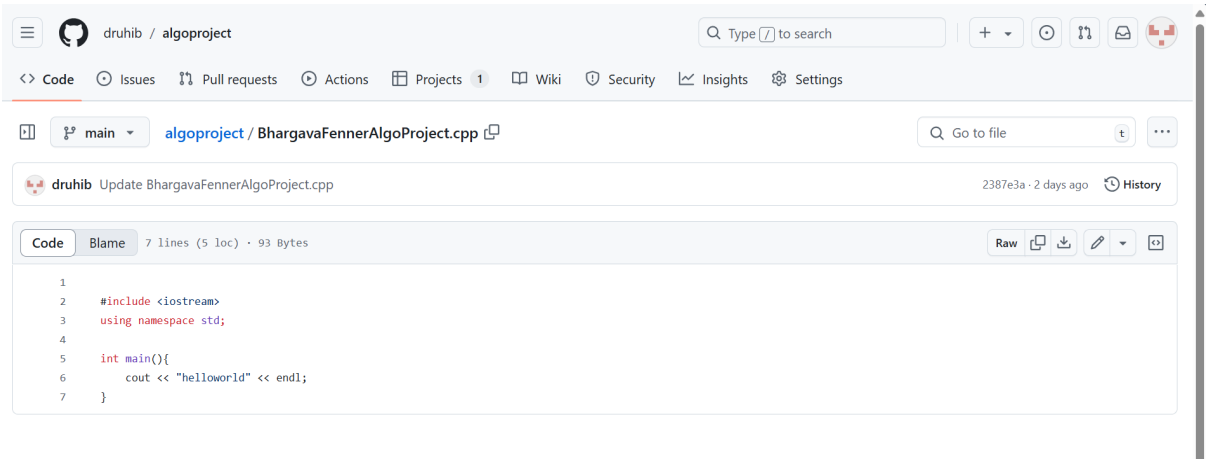

Figure 1: Github Repository

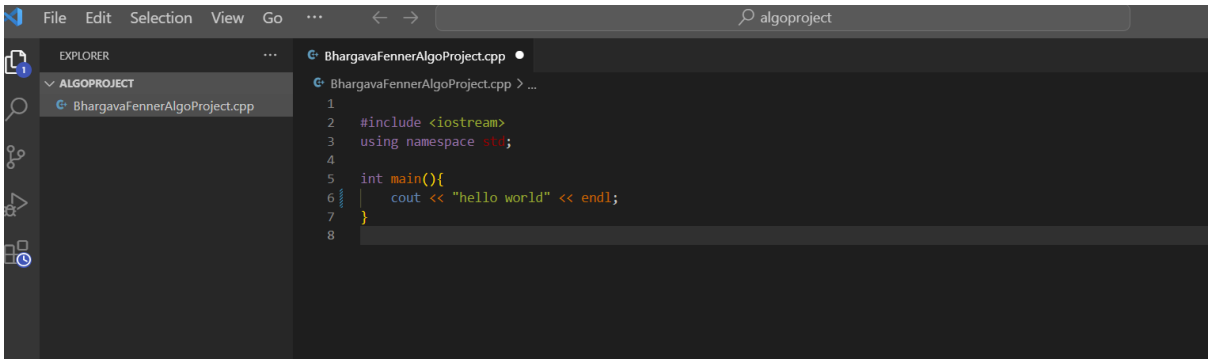

Figure 2: C++ File

6 Progress Review (October 10)

6.1 Progress Update

The team successfully completed the initial setup of the development environment and repository. Research has also begun to flesh out how to implement our ideas.

6.2 Issues Encountered

We encountered issues with the Github environment, but we are resolving this by watching Github tutorials.

7 Revised Project Plan (October 10)

7.1 Updated Plan

We plan to research how to incorporate outside data in neatly organized data structures and begin backend development.

7.2 Justification for Changes

No large changes were made to the initial timeline and plan.

8 Progress Review (October 27th)

8.1 Final Project Scope + Objectives

The aim of this project is to design and develop a food resource map/system for the Greater Boston area. The system will take different factors such as transportation and grocery stores in the area, distribution center proximity, etc. in account. The projects main objectives are:

- use stats of Boston Neighborhoods to understand food allocation/ accessibility
- possible statistics:
  - number of Soup Kitchens / Boston Eats (Data 2019)
  - redlining grade
  - median income
- provide a path for mobile market to travel in order to allocate food

The expected outcomes include a functioning food recourse map, with the user being able to access a "best route" for mobile food recourse markets to follow. It can consider a user input for starting location (with a default starting location if the user opts out), and then given statistics to determine the following route.

8.2 Algorithm Design

The next steps for this project involve actually implementing the algorithm for mapping out a path for mobile markets. In Iterations 3 and 4, we focused on researching data we could use for each neighborhood and creating the data structures needed for the algorithm. The current plan is: Allow a program user the option to enter a high priority neighborhood or node. This is where the mobile market would start its route. If the user chooses not to enter a starting point, the program will default to one. Next, using a graphing algorithm (either DFS or BFS), the program will map out a path for the mobile market, so that it eventually reaches each neighborhood. In order to determine the next node (if two are equidistant), it will use an equation based on factors such as Redlining Grade and median income to determine which neighborhood to branch to next.

8.3 Data Structures

The code for this implementation consists of two main data structures. Firstly, a type Node which encompasses a neighborhood. Each neighborhood is a distinct Node, with the attributes Grade (Redlining grade), Soup Kitchens (number of soup kitchens/Boston Eats data), Median Income (the median income of the area), Transport (transport lines in the area), and connections (a list of connections to other neighborhoods or nodes). Second, there is a Graph data type, which allows you to create the graph actually connecting the nodes. You instantiate it by giving the number of neighborhoods. Then, you can do things such as add neighborhoods (given all the data required), add edges (connections between neighborhoods storing them in a list), or view the existing data.

8.4 Testing Results

So far, we have created the data structures and testing implementing the actual graph (creating the neighborhoods and importing data).

```
Neighborhood: Roxbury
Grade: D
Soup Kitchens: 1
Median Income: 32.5k
Transport Lines: Orange
Connected to: Jamaica Plain Roslindale Mattapan Dorchester
```

Figure 3: Figure 3: Node Implementation

```
int main() {
    Graph graph(17);
    graph.addNeighborhood("East Boston", 'D', 8, 54.9, {"Blue"});
    graph.addNeighborhood("Boston", 'D', 1, 58.5, {"Red", "Green"});
    graph.addNeighborhood("South Boston", 'D', 3, 89.1, {"Red", "Blue"});
    graph.addNeighborhood("South End", 'D', 4, 69.9, {"Orange", "Green"});
    graph.addNeighborhood("CharlesTown", 'D', 1, 94.6, {"Orange"});
    graph.addNeighborhood("Allston", 'C', 0, 52.1, {"Green"});
    graph.addNeighborhood("Brighton", 'C', 0, 52.1, {"Green"});
    graph.addNeighborhood("Fenway/Kenmore", 'C', 3, 65.7, {"Green"});
    graph.addNeighborhood("Roxbury", 'D', 1, 32.5, {"Orange"});
    graph.addNeighborhood("Jamaica Plain", 'D', 5, 32.3, {"Orange"});
    graph.addNeighborhood("Dorchester", 'C', 1, 0, {"Red"}); // median income is split
    graph.addNeighborhood("Roslindale", 'C', 6, 77.9, {"Orange"});
    graph.addNeighborhood("West Roxbury", 'C', 4, 90.5, {"Orange"});
    graph.addNeighborhood("Mattapan", 'C', 8, 43.5, {"Red"});
    graph.addNeighborhood("Hyde Park", 'C', 1, 64.9, {"Red"});
    graph.addNeighborhood("North Dorchester", 'C', 3, 59.7, {"Red"});
    graph.addNeighborhood("South Dorchester", 'C', 4, 62.2, {"Red"});

    graph.addEdge("Roxbury", "Jamaica Plain");
    graph.addEdge("Roxbury", "Roslindale");
    graph.addEdge("Roxbury", "Mattapan");
    graph.addEdge("Roxbury", "Dorchester");
```

Figure 4: Figure 4: Main

8.5 Next Steps

Our next goal is to implement the algorithm. Firstly, we plan to set up Breadth First Search within the graph, and second, we want to create the equation that will determine node order.

9 Iteration 5 (November 10th)

9.1 Identifying Tasks

- 1. Analyze research and define relevant data
- 2. Import data into final project
- 3. Implement BFS
- 4. Implement sorting algorithm

9.2 Analyze research and define relevant data:

The first thing we needed to do was make the final decisions on what data we were going to use/what was going to be relevant to our project. We had a plethora of recourses/databases we had found, but not all of it was needed in order to implement our program. We decided on a database that contained data for all of the neighborhoods we were analyzing, and mainly focused on access to grocery stores and access during a storm surge (as well as average distance to a grocery, poverty rate, median income, etc). The data can be viewed here: https://northeastern-my.sharepoint.com/:x:/g/personal/bhargava$_{dn}$ortheastern$_{e}$du/EQ0AcveXq3dKglNb
688c1z0cqcErXbkXww?e = 1ffUkw.

9.3 Import data into final project:

Next, we needed to import all of this data into our final project so that we could work with it/test our algorithms using actual data. We did this by creating the graph stucture, and then adding each data point via main.

```
graph.addNeighborhood("Allston", 'C', 3, 0.1, 0, 12, 0.41, 0, 1, 0.37, 17.7, 8.1, 0.16, 52.1, {"Green Line"});
graph.addNeighborhood("Backbay", 'C', 2, 0.1, 0, 5, 0.28, 0, 3, 0.23, 7.1, 4.1, 0.14, 97.8, {"Green Line"});
graph.addNeighborhood("Beacon Hill", 'A', 0, 0, 0, 3, 0.33, 0, 1, 0.24, 5.8, 2.5, 0.14, 102.2, {"Green Line"});
graph.addNeighborhood("Brighton", 'C', 2, 0.04, 1, 15, 0.33, 0, 0, 0.73, 13.3, 11.4, 0.19, 65.7, {"Green Line"});
graph.addNeighborhood("CharlesTown", 'D', 1, 0.06, 0, 6, 0.36, 0, 3, 0.37, 19.4, 17, 0.1, 94.6, {"Green Line"});
graph.addNeighborhood("Dorchester", 'C', 5, 0.04, 2, 47, 0.41, 0, 2, 0.58, 22.6, 29.3, 0.13, 60.05, {"Green Line"});
graph.addNeighborhood("Downtown", 'N', 0, 0, 0, 23, 2.05, 1, 8, 0.38, 20.3, 12.8, 0.1, 168.6, {"Green Line"});
graph.addNeighborhood("East Boston", 'D', 1, 0.02, 1, 28, 0.69, 1, 12, 0.82, 16.5, 22.1, 0.11, 54.9, {"Green Line"});
graph.addNeighborhood("Fenway", 'C', 2, 0.06, 2, 14, 0.41, 0, 10, 0.28, 19.7, 11.8, 0.15, 37.9, {"Green Line"});
graph.addNeighborhood("Hyde Park", 'C', 5, 0.16, 0, 9, 0.29, 0, 0, 0.48, 9.4, 15.2, 0.29, 64.9, {"Green Line"});
graph.addNeighborhood("Jamaica Plain", 'A', 3, 0.08, 0, 9, 0.24, 0, 0, 0.45, 16.9, 13.8, 0.19, 84, {"Green Line"});
graph.addNeighborhood("Longwood Medical Center", 'C', 0, 0, 0, 0, 0, 0, 0, 0.29, 7, 20.1, 0.27, 38.5, {"Green Line"});
graph.addNeighborhood("Mattapan", 'C', 2, 0.09, 0, 8, 0.35, 0, 0, 0.6, 21.5, 28.8, 0.28, 43.5, {"Green Line"});
graph.addNeighborhood("Mission Hill", 'D', 1, 0.06, 0, 2, 0.12, 0, 0, 0.29, 28.8, 22.5, 0.16, 37.3, {"Green Line"});
graph.addNeighborhood("North End", 'D', 0, 0, 0, 7, 0.69, 0, 2, 0.64, 4.2, 1.6, 0.05, 98.5, {"Green Line"});
graph.addNeighborhood("Roslindale", 'C', 2, 0.07, 0, 5, 0.17, 0, 0, 0.46, 11.4, 15.3, 0.18, 77.9, {"Green Line"});
graph.addNeighborhood("Roxbury", 'D', 2, 0.04, 0, 20, 0.41, 0, 4, 0.37, 34.9, 40, 0.14, 32.3, {"Green Line"});
graph.addNeighborhood("South Boston", 'D', 2, 0.06, 0, 10, 0.3, 0, 3, 0.35, 17.9, 13.4, 0.16, 89.1, {"Green Line"});
graph.addNeighborhood("South Boston Waterfront", 'N', 0, 0, 0, 2, 1.06, 0, 1, 0.56, 3.7, 8.4, 0.55, 0, {"Green Line"});
graph.addNeighborhood("South End", 'D', 4, 0.16, 3, 10, 0.41, 0, 9, 0.25, 19.6, 17.9, 0.08, 69.9, {"Green Line"});
graph.addNeighborhood("West End", 'D', 1, 0.25, 0, 1, 0.25, 0, 0, 0.21, 10.3, 6.4, 0.18, 83.8, {"Green Line"});
graph.addNeighborhood("West Roxbury", 'B', 2, 0.07, 0, 4, 0.13, 0, 0, 0.33, 5.8, 7.5, 0.33, 90.5, {"Green Line"});
```

Figure 5: Creating neighborhoods in main

9.4 Implement BFS:

To complete the main structure of the code, we needed to finish our Breadth First Search algorithm. To do this, we utilized a queue and its FIFO structure. Elements enter the queue, along with the list of their connections. Elements then exit the queue in the order that they were added (So first an element

would exit, then its connections, then its connections connections, etc). We tested our algorithm with our data points and it was proved successful.

9.5 Implement sorting algorithm:

For our last goal for this iteration, we wanted to create a way for an elements connections to be sorted before they enter the queue (so that there was a system of sorting the connections/going to the next node instead of it simply being the order that the connections/edges were created). In order to do this, we created a sorting algorithm with the list "connections" as an input. The sorting algorithm then uses the data attached to each node to sort the list (example- sort by poverty rate).

9.6 Moving Forward + Final Steps:

For our final steps, the goals that we want to complete this week are:

- 1. Clean up the code
- 2. Ensure data types are consistent across the program (for example- change the lists to queues)
- 3. Add a menu for easier user interaction

9.7 Github Link

Link to Github: https://github.com/druhib/algoproject