EECE 2560

# Smart City: **Mobile Market Resource Allocation**

**Druhi Bhargava and Fausta Fenner**

# Introduction

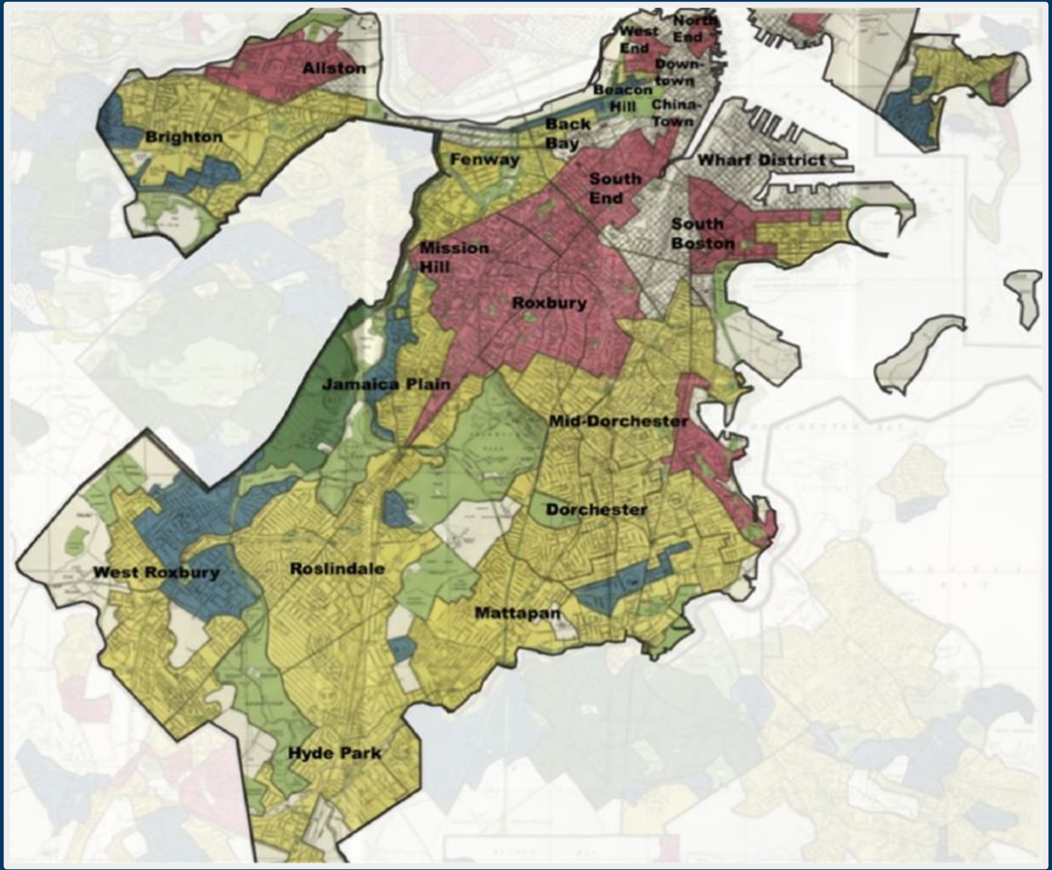
**Goals and Project Scope**:

- To display food resources and data for 17 neighborhoods in Boston
- Provide path for food allocation based on current resources and neighborhood needs
- Provide a path for food allocation if a natural disaster (ex: Flood) impacts a neighborhood

# Redlining Data

Redlining: (1930-1968)

- Discriminatory practice by banks and institutions of denying loans or insurance to people based on race or ethnicity



# Neighborhood Resilience Data

| Poverty Rate | SNAP Rate | Number of Grocery Stores | Grocery Stores Per 1,000 | Number of Grocery Stores in 7.5 Foot Storm Surge | Number of Corner Stores | Corner Stores Per 1,000 | Number of Corner Stores in Flood-plain | Number of Corner Stores in 7.5 Foot Storm Surge | Average Distance to Closest SNAP Outlet (Miles) | Average Distance to Closest Grocery Store (Miles) |
|---|---|---|---|---|---|---|---|---|---|---|

# Literature Review

- Databases:
  - Correlations of Historic Redlining to Poverty and Food Insecurity
    - Redlining and Present-Day Neighborhood Opportunity in the Boston Area | diversitydatakids.org
    - https://www.researchgate.net/publication/371128165/figure/fig2/AS:11431281162565562@1685363396691/Bostons-Redlining-Map-Source-33.png
    - Info on Redlining: What is Redlining? Redlining Explained, A Brief History
  - Median Income: The Demographic Statistical Atlas of the United States - Statistical Atlas
  - Current Food Resources: ICIC_Food_Systems_final_revised_post.pdf

# Methodology

**Data Structures Utilized:**

- Vertex: custom data structure for each neighborhood to store individual attributes.
- Hash table: mapping key and value pair data
  - Ex: neighborhood attributes and user weights input.
- List: For storing connections between neighborhoods.
- Queue: Used in BFS for processing Vertices.
- Vector: For sorting neighbors based on weight.

# Methodology

**Description of Methods and Techniques Used:**

- **Graphing Algorithm (BFS)**
  - Using a starting node input by the user, provide a list of neighborhoods to visit using BFS
- **Sorting Algorithm**
  - Used built in vector sorting function with a custom comparison algorithm called get_weight
  - get_weight → calculates a total weight for each neighborhood based on the selected data + user input weight

# Methodology

## Pseudocode for Techniques Used: BFS

```
void bfsWithWightedOrder(start, choice):
    visited → empty map
    bfsQueue → empty queue
    Push start to bfsQueue
    Add start to visited

    While bfsQueue is not empty:
        Dequeue currentNode from bfsQueue
        Print: "visiting" + currentNode
        Sort currentNodes neighbors by weight:
            If choice ==1, sort by general, if choice == 2 sort by natural disaster
        For neighbor in sorted neighbors:
            If neighbor not visited:
                Add to visited and enequeue to bfsQueue
```

# Methodology

## Pseudocode for Techniques Used: Get Weight

```
void getWeight_gen():
    totalWeight = 0
    If “poverty_weight” exists in factorWeights:
        Add poverty_rate weight * poverty_rate to totalWeight

    If “median_income” exists in factorWeights:
        Add median_incomeweight * median_incometo totalWeight

    If “avg_distance_grocery” exists in factorWeights:
        Add avg_distance_grocery weight * avg_distance_grocery to totalWeight

    Return totalWeight
```

# Methodology

## Pseudocode for Techniques Used: Get Weight (Natural Disaster)

```
void getWeight_natural_disaster():
    totalWeight = 0
    If "num_grocery_7ft_surge" exists in factorWeights:
        Add num_grocery_7ft_surge weight * num_grocery_7ft_surge to totalWeight

    If "num_corner_in_floodplain" exists in factorWeights:
        Add num_corner_in_floodplain weight* num_corner_in_floodplain totalWeight

    If "num_corner_7ft_surge" exists in factorWeights:
        Add num_corner_7ft_surge weight * num_corner_7ft_surge to totalWeight

    If "poverty_rate" exists in factorWeights:
        Add poverty_rate weight * poverty_rate to totalWeight

    Return totalWeight
```

# Methodology

## Time Complexity Estimation

```
void bfsWithWightedOrder(start, choice):                                  →  O(n^2logn)
    visited → empty map
    bfsQueue → empty queue
    Push start to bfsQueue
    Add start to visited

    While bfsQueue is not empty:
        Dequeue currentNode from bfsQueue
        Print: currentNode
        Sort currentNodes neighbors by weight:
            If choice ==1, sort general, else sort by natural disaster
        For neighbor in sorted neighbors:
            If neighbor not visited:
                Add to visited and enequeue to bfsQueue
```

# Analysis and Results:

Menu Bar:

```
Please select (1-4)
1.Display a Neighborhoods Data
2.General Mobile Market Route
3.Mobile Market Route During a Natural Disaster
4.End Program
 Number Entered: 1
```

# Analysis and Results:

Searching Data for a Neighborhood:

```
 Number Entered: 1
Enter a Neighborhood: Roxbury
Neighborhood: Roxbury
grade: D
number of grcoery stores: 2
grocery stores per 1000: 0.04
number of grocery stores 7ft surge: 0
number of corner stores: 20
number of corner stores per 1000: 0.41
number of corner in floodplain: 0
number of corner in 7ft surge: 4
poverty rate: 34.9
snap rate: 40
avg distance to snap: 0.14
median_income: 32.3
```

# Analysis and Results:

## General Sorting Algorithm with Weighted Factors and BFS

Factors Used:

- poverty rate
- median income
- avg distance to a grocery store (miles)

```
Enter a Neighborhood to start the route at: Roxbury
Enter the weight for poverty_rate (0.0 to 1.0): .7
Enter the weight for median_income (0.0 to 1.0): .5
Enter the weight for avg_distance_grocery (the average distance to a grocery
 store) (0.0 to 1.0): .6
Visiting neighborhood: Roxbury
Visiting neighborhood: Jamaica Plain
Visiting neighborhood: Roslindale
Visiting neighborhood: Dorchester
Visiting neighborhood: Mattapan
Visiting neighborhood: South End
Visiting neighborhood: Mission Hill
Visiting neighborhood: Hyde Park
```

# Analysis and Results:

Natural Disaster Algorithm

Factors Used:

- # corner stores in 7.5 ft storm surge, floodplain
- # grocery stores in 7.5 ft storm surge
- poverty rate

```
Number Entered: 3
Enter a Neighborhood to start the route at: Roxbury
Enter the weight for num_grocery_7ft_surge (Number of accessible grocery stores in a 7.5 foot storm surge) (0
.0 to 1.0): .7
Enter the weight for num_corner_in_floodplain (Number of corner stores in a floodplain) (0.0 to 1.0): .5
Enter the weight for num_corner_7ft_surge (Number of corner stores accessible during a 7.5 foot storm surge)
(0.0 to 1.0): .5
Enter the weight for poverty_rate (0.0 to 1.0): .8
Visiting neighborhood: Roxbury
Visiting neighborhood: Dorchester
Visiting neighborhood: Mattapan
Visiting neighborhood: Jamaica Plain
Visiting neighborhood: Roslindale
Visiting neighborhood: Mission Hill
Visiting neighborhood: South End
Visiting neighborhood: Hyde Park
```

# Analysis and Results:

Ending Program

```
Please select (1-4)
1.Display a Neighborhoods Data
2.General Mobile Market Route
3.Mobile Market Route During a Natural Disaster
4.End Program
 Number Entered: 4
Ending Program
```

# Analysis and Results

**Duration of Project**: 1 month, 3-5 hours per week

# Discussion

**Key Findings and Implications:**

- Some neighborhoods currently have less available food resources than others, and this algorithm could be used to help with decision making for allocating resources and understanding current resources of Neighborhoods in Boston
- Help a mobile market route a service to provide additional resources based on a starting point

**Project Limitations:**

- Data is limited to what is readily available on the internet, it is not constantly updating

# Conclusion

**Conclusions from the Project:**

- Many different factors can be considered when looking at food resource allocation
- Graphing algorithms can be utilized in combination with data to create ordered lists or routes

**Recommendations for Future Work:**

- Addition of a more visual UI
- Increase flexibility by allowing the user choose which factors they want to weigh

# Additional References


- Graphing Algorithms:
  - https://www.youtube.com/watch?v=LG_KDNd5BQI
  - https://www.youtube.com/watch?v=v78tWnjklio
- Inspiration of Concept Piece:
  - Food Insecurity in Boston
  - Harvard_JCHS_mapping_neighborhood_change_boston_january_2019.pdf