Northeastern University
College of Engineering
Department of Electrical and Computer Engineering
EECE 2560: Fundamentals of Engineering Algorithms
Fall 2024

Project: Real Time Emergency Response System

Name: Tyler Myers & Alex Hunt

Introduction

The purpose of this project is to develop a program to dispatch emergency services to various locations on Northeastern University's campus based on the urgency of the situation, route, terrain, and distance.

Introduction

The project's main objectives are:

- To find optimal paths for emergency services based on their location on campus and the incident location.
- To respond to incidents as quickly as possible by dispatching the nearest available emergency services based on the incident type.
- To dynamically deploy emergency services to locations on campus that are incident hot spots.

Literature Review

- NUPD already has an emergency response procedure in place
- Used Google API to do the routing of all personnel

Literature Review

- Nlohmann/json used to extract needed data from API response

Methodology

Methods and Techniques
Data preprocessing

1. Parse past data from CSV
   - Store data about locations, emergencies, and items in arrays
   - Dynamic officer allocation on campus for use dispatching later
2. When receiving an emergency:
   - Get route from Google API
   - Algorithms to allocate officer personnel and resources.

Methodology

Routing Code

```
std::string autoURL = "https://maps.googleapis.com/maps/api/place/autocomplete/json"
                      "?input=" + location +
                      "&location=Northeastern+University"
                      "&radius=3000" // Limit search radius to within 3km of Northeastern
                      "&key=" + apiKey;
PlaceID = jsonResponse["predictions"][0]["description"].get<std::string>();
```

```
std::string polyUrl = "https://maps.googleapis.com/maps/api/directions/json?"
                      "origin=" + origin +
                      "&destination=" + destination +
                      "&mode=walking"
                      "&location=Northeastern+University" // limit search results to northeastern area
                      "&radius=3000" // Limit search radius to within 3km of Northeastern
                      "&key=" + apiKey;
polyline = jsonResponse["routes"][0]["overview_polyline"]["points"].get<std::string>()
```

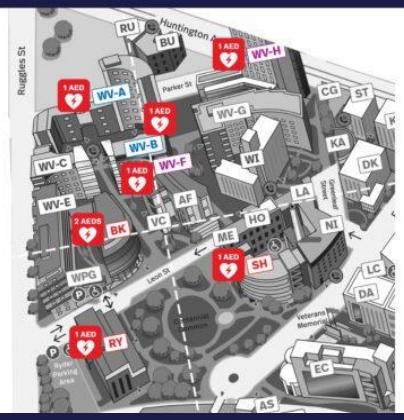
```
std::string url = "https://maps.googleapis.com/maps/api/staticmap?"
                  "size=600x600"
                  "&markers=color:blue%7Clabel:S%7C" + origin + // Add NEU to address for accuracy
                  "&markers=color:red%7Clabel:D%7C" + destination + // Add NEU to address for accuracy
                  "&path=enc:" + getPolyLine(origin, destination) + // Shows actual path for officers to take
                  "&maptype=satellite" // Change map type to satellite
                  //"&zoom=17" // Adjust zoom level to focus on the path
                  "&key=" + this->apiKey;
```

Methodology

Dynamic Allocation

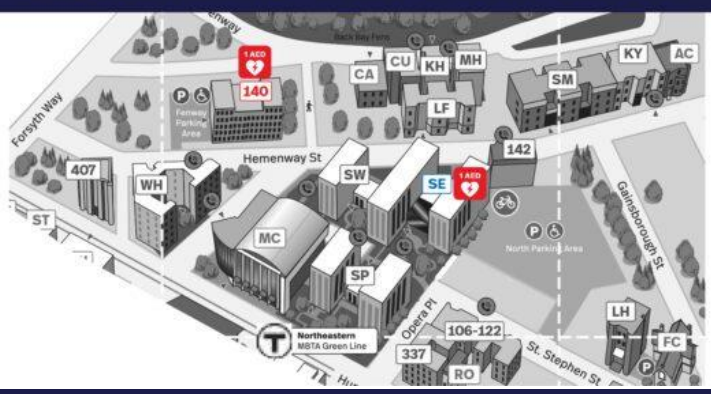
```
PastEmergencies[0] ~ Columbus Place (Columbus South Sector)
PastEmergencies[1] ~ Behrakis (West Campus Sector)
PastEmergencies[2] ~ Curry (Academics Sector)
PastEmergencies[3] ~ Marino (East Fenway Sector)
```

AED LOCATIONS - WEST CAMPUS SECTOR

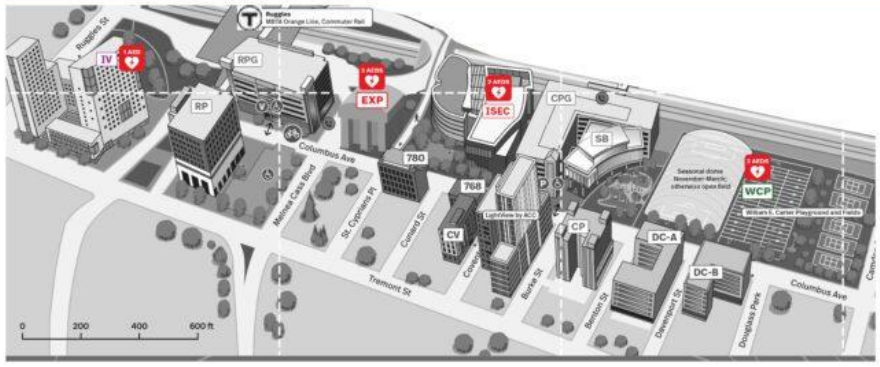


BK Behrakis Health Science Center – 1
• First floor – right of main entrance.
Behrakis – Exercise Physiology Lab - 1
RY Ryder Hall – 1
• Enter first floor, on the left.
SH Shillman Hall – 1
• Next to women's bathroom first floor.
WVA West Village A-North – 1
• Near elevator first floor.
WVB West Village B – 1
• Near elevator first floor.
WVF West Village F Visitor's Center – 1
• Main Lobby Near Side Entrance.
WVH West Village H – 1
• Near mailboxes first floor

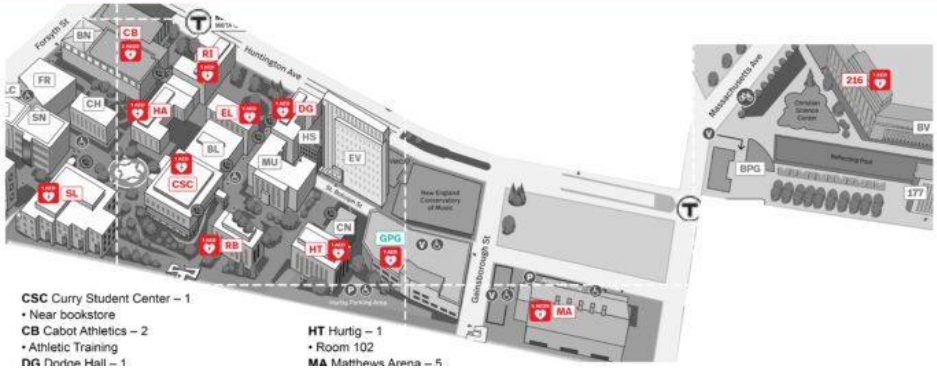
AED LOCATIONS - EAST FENWAY SECTOR



AED LOCATIONS - COLUMBUS SOUTH SECTOR



AED LOCATIONS - ACADEMICS SECTOR



CSC Curry Student Center – 1
• Near bookstore
CB Cabot Athletics – 2
• Athletic Training
DG Dodge Hall – 1
HT Hurtig – 1
• Room 102
MA Matthews Arena – 5

Methodology

Pseudo Code
Dynamic Allocation

Algorithm 1: DynamicOfficerAllocation

```
Input: numOfficers
Output: None
numOfficers ← numOfficers - 4
while numOfficers > 0 then
    i ← 1
    worstZone ← 0
    while pastEmergencies[worstZone] == 0 then
        worstZone++
    end
    while i < 4 then
        if PastEmergencies[i] != 0 then
            if OfficersAllocated[i]/PastEmergencies[i] <
             OfficersAllocated[worstZone]/PastEmergencies
             then                                    [worstZone]
                worstZone ← i
            end
        end
        i++
    end
    OfficersAllocated[worstZone]++
    numOfficers–
end
End Algorithm: DynamicOfficerAllocation
```

Methodology

Pseudo Code
Greedy Algorithm

Algorithm 1: GreedilyDeployOfficer

```
Input: A list of officer objects, officers. The location of the emergency,
 emergencyLocation
Output: Nearest officer deployed to location
Begin Algorithm: GreedilyDeployOfficer
shortestPath ← infinity
nearestOfficer ← NULL
for each officer in officers do
    if officer is available then
        pathLength ← Distance of officer to emergencyLocation
        if pathLenth<shortestPath then
            shortestPath ← pathLength
            nearestOfficer ← officer
        end
    end
end
if nearestOfficer != NULL then
    Deploy officer to emergencyLocation
    nearestOfficer.isAvailable ← false
end
else
    print("No available officers.")
end
End Algorithm: GreedilyDeployOfficer
```

Methodology

Pseudo Code
0-1 Knapsack

Algorithm 1: solveKnapsack

Input: A list of equipment objects, *items*. The maximum weight of the knapsack, *maxWeight*.
Output: A list of strings representing the names of the items chosen, *maxWeight*.

```
Begin Algorithm: solveKnapsack
n ← Number of items
dp ← A 2D array of size (n+1)x(maxWeight+1) initialized to 0
for i = 1 to n do
    for w = 1 to maxWeight do
        if items[i-1].weight≤ w then
            dp[i][w] = max(dp[i-1][w],
              dp[i-1][w-items[i-1].weight+items[i-1].importance])
        end
        else
            dp[i][w] = dp[i-1][w]
        end
    end
end
chosenItems ← empty list
w ← w-items[i-1].weight
for i = n decrementing to 1 do
    if dp[i][w] != dp[i − 1][w] then
        Add items[i-1].name to chosenItems
        w ← w − items[i-1].weight
    end
end
End Algorithm: solveKnapsack
```

Methodology

Time Complexity

- Dynamic Allocation O(n)
- Greedy Algorith O(n)
- 0-1 Knapsack O(n)

Methodology
Data Structures

1. All data stored is done so in CSV file and parsed into arrays.
2. Indexes correlate to the item
3. Equipment needed for emergency types stored in unordered map

```
/* The first location is where the officers are stationed
   The second location is the AED zone the station point is based upon
   Zone 1 ~ Columbus Place (Columbus South Sector)
   Zone 2 ~ Behrakis (West Campus Sector)
   Zone 3 ~ Curry (Academics Sector)
   Zone 4 ~ Marino (East Fenway Sector)
*/
```

Analysis and Results

- Live Demonstration Here

Discussion

- Project was limited by the deadline
- Large potential for expansion, but good MVP

Conclusion

- Our project is viable
- Improvements:
  - Number of emergency types and resources available
  - Flush out the GUI

References

- List of All Cited Sources
- https://console.cloud.google.com/apis/library