Open House Route Planner

Alexander Jansing

SUNY Polytechnic Institute
Department of Computer Science

29 April 2019

Inspiration

I have been looking for houses. When I add open houses to my Google Calendar, I am able to request direction to whatever house is open next in time, but I was thinking, *"What if two houses are significantly far apart, open at similar times, and there are other houses in each of their respective neighborhoods that open at different times? Is there a way I can plan my day of house hunting so that I can attend all of the open houses?"*

Inspiration

I have been looking for houses. When I add open houses to my Google Calendar, I am able to request direction to whatever house is open next in time, but I was thinking, *"What if two houses are significantly far apart, open at similar times, and there are other houses in each of their respective neighborhoods that open at different times? Is there a way I can plan my day of house hunting so that I can attend all of the open houses?"*
The answer to this question is, "yes, within reason."

Objective

Given a series of open houses the application should find routes that will allow the user to visit the maximum number of open houses given the constraints of *travel time* and *when the open houses are open*.
After stating the problem, it was divided up into several part:

- where the houses were with respect to each other,
- when the open houses were,
- and try to determine the path I needed to take to visit as many open houses as possible.

I will describe how each of these tasks were accomplished and what other work needed to be done to facilitate that work.

Requirements

As with any project, it is difficult to just dive right into working on the solution.

Requirements

As with any project, it is difficult to just dive right into working on the solution. Data needed to be

- gathered,
- parsed,
- geocoded,
- transformed for repeatable use,
- and cached.

ETL

Extract - ICS files [2] was manually gathered from `Trulia.com` [7].

ETL

Extract - ICS files [2] was manually gathered from `Trulia.com` [7].

Transform -

- The ICSParser (see slide 10) parsed ICS files to JSONs that python could handle.
- The Esri Flask App (see slide 10) gathered data from the Esri Developer API to geocode addresses to locations and fetched directions between two points so we could see how long it would take to travel between houses.
- And the directions matrix (see slide 8) were used to manipulate the data into a form that would be useful for the OpenHouseGraph.

ETL

Extract - ICS files [2] was manually gathered from `Trulia.com` [7].

Transform -

- The ICSParser (see slide 10) parsed ICS files to JSONs that python could handle.
- The Esri Flask App (see slide 10) gathered data from the Esri Developer API to geocode addresses to locations and fetched directions between two points so we could see how long it would take to travel between houses.
- And the directions matrix (see slide 8) were used to manipulate the data into a form that would be useful for the OpenHouseGraph.

Load - MongoOps (see slide 10) performed several roles:

- querying for existence of geocoded addresses and directions,
- storing data in a MongoDB instance,
- fetching data from a MongoDB instance,
- and requesting information from the Esri Flask App if the data did not exist in its database.

Infrastructure

All of this had to be supported by an underlying infrastructure.

Infrastructure

All of this had to be supported by an underlying infrastructure.
For this, I used docker-compose [6].
My docker-compose network consisted of three containers:

- esri [1],
- mongo [4],
- and routefinder [3].

The docker-compose.yml can found here.

Definitions

- *Safely Query* - to query for existence in MongoDB before querying ArcGIS API for information. This applies to both *geocoding* an address its coordinates on the planet and to gather the directions from one point to another.

Definitions

- *Safely Query* - to query for existence in MongoDB before querying ArcGIS API for information. This applies to both *geocoding* an address its coordinates on the planet and to gather the directions from one point to another.
- geocoding - the process or converting addresses to coordinates on the globe.

Definitions

- *directions matrix* - an array of JSONs containing vertex and edge information. While not strictly a matrix, it supplies the information to populate a matrix of travel times. It is a good mental image to have when thinking about the OpenHouseGraph.

```
    [{'_id': ObjectId('5cad42f3671c850b358ab86b'), 'url': REDACTED, 'dtstart':
        ↪ '20190414T153000Z', 'dtend': '20190414T170000Z', 'summary': REDACTED,
        ↪ 'description': REDACTED, 'location': {'geometry': {'x': x_0, 'y': y_0,
        ↪ 'spatialReference': {'wkid': 4326, 'latestWkid': 4326}}, 'attributes':
        ↪ {'Loc_name': 'World', 'Status': 'M', 'Score': 100, ... 'X': x_0, 'Y':
        ↪ y_0, 'DisplayX': x_{d0}, 'DisplayY': y_{d0}, 'Xmin': x_{min0}, 'Xmax':
        ↪ x_{max0}, 'Ymin': y_{min0}, 'Ymax': y_{max0}, 'ExInfo': '', 'OBJECTID':
        ↪ 1}, 'address': REDACTED}, 'address_hash': sha1(location0), 'durations':
        ↪ [[1, 13.85]]}
{'_id': ObjectId('5cac003a671c85002d41afb9'), 'url': REDACTED, 'dtstart':
    ↪ '20190413T150000Z', 'dtend': '20190413T170000Z', 'summary': REDACTED
    ↪ 'description': 'REDACTED, 'location': {'geometry': {'x': x_1, 'y': y_1,
    ↪ 'spatialReference': {'wkid': 4326, 'latestWkid': 4326}}, 'attributes':
    ↪ {'Loc_name': 'World', 'Status': 'M', 'Score': 100, ... 'X': x_1, 'Y': y_1,
    ↪ 'DisplayX': x_{d1}, 'DisplayY': y_{d1}, 'Xmin': x_{min1}, 'Xmax': x_{max1},
    ↪ 'Ymin': y_{min1}, 'Ymax': y_{max1}, 'ExInfo': '', 'OBJECTID': 1},
    ↪ 'address': REDACTED},, 'address_hash': sha1(location1), 'durations': [[0,
    ↪ 14.15]]}]
```

Figure 1: Simplified Directions Array (Matrix).

Definitions

- *directions matrix* - an array of JSONs containing vertex and edge information. While not strictly a matrix, it supplies the information to populate a matrix of travel times. While a matrix is no longer used, it is a good mental image to have when thinking about the OpenHouseGraph.
  - Figure 2 is an implied matrix of travel times between locations from Figure 1. Where $t_{i,j}$ is the travel time between house i and house j. The travel time from house i to house i is given the value -1 as a guard against reflexive traveling.

$$\begin{pmatrix} -1 & t_{0,1} & \cdots & t_{0,n-1} \\ t_{1,0} & -1 & \cdots & t_{1,n-1} \\ \vdots & \ddots & \ddots & \vdots \\ t_{n-1,0} & \cdots & \cdots & -1 \end{pmatrix}$$

Figure 2: Matrix of travel times between locations.

Definitions

- ICSParser - Class for parsing ICS files to JSON to be geocoded.

Definitions

- ICSParser - Class for parsing ICS files to JSON to be geocoded.
- Esri Flask App - REST endpoint that is designed to accept information from MongoOps and query the ArcGIS Developer API for geocoded address information or directions between two geometry points.

Definitions

- ICSParser - Class for parsing ICS files to JSON to be geocoded.
- Esri Flask App - REST endpoint that is designed to accept information from MongoOps and query the ArcGIS Developer API for geocoded address information or directions between two geometry points.
- MongoOps - Class for:
  - loading data to database,
  - querying the database for geocoded address,
  - querying the database for directions,
  - querying the ArcGIS Developer API for geocoded address,
  - and querying the ArcGIS Developer API for directions.

Definitions

- ICSParser - Class for parsing ICS files to JSON to be geocoded.
- Esri Flask App - REST endpoint that is designed to accept information from MongoOps and query the ArcGIS Developer API for geocoded address information or directions between two geometry points.
- MongoOps - Class for:
  - loading data to database,
  - querying the database for geocoded address,
  - querying the database for directions,
  - querying the ArcGIS Developer API for geocoded address,
  - and querying the ArcGIS Developer API for directions.
- OpenHouseGraph - A graph data structure used for computing routes one might take while visiting open houses.
  - Inspired by: Data Scientists, The one Graph Algorithm you need to know [5] - Basis for the OpenHouseGraph class.

Example Houses

ICS files contain more information than is provided. Some information has been omitted for brevity and privacy.

```
DTSTART:20190427T160000Z
DTEND:20190427T173000Z
LOCATION:location0
      ↪ (Clinton)
```

```
DTSTART:20190428T173000Z
DTEND:20190428T190000Z
LOCATION: location1
      ↪ (Utica)
```

```
DTSTART:20190428T160000Z
DTEND:20190428T183000Z
LOCATION: location2
      ↪ (New Hartford)
```

```
DTSTART:20190428T153000Z
DTEND:20190428T170000Z
LOCATION: location3
      ↪ (New Hartford)
```

```
DTSTART:20190501T200000Z
DTEND:20190501T220000Z
LOCATION: location4
      ↪ (Sherrill)
```

```
DTSTART:20190427T140000Z
DTEND:20190427T160000Z
LOCATION: location5
      ↪ (Morrisville)
```

```
DTSTART:20190427T140000Z
DTEND:20190427T160000Z
LOCATION: location6
      ↪ (Oneida)
```

Figure 3: Shortened ICS files.

Example Houses

```
[{'ID': 0, 'address': location0 (Clinton),
  'edges': [[1, 14.88], [2, 7.42], [3, 8.51], [4, 21.55], [5, 31.54], [6, 25.19]],
  'end': 810, 'start': 720},
 {'ID': 1, 'address': location1 (Utica),
  'edges': [[0, 15.47], [2, 8.54], [3, 7.85], [4, 31.88], [5, 42.08], [6, 35.73]],
  'end': 900, 'start': 810},
 {'ID': 2, 'address': location2 (New Hartford),
  'edges': [[0, 8.15], [1, 8.34], [3, 2.05], [4, 24.24], [5, 33.74], [6, 27.41]],
  'end': 870, 'start': 720},
 {'ID': 3, 'address': location3 (New Hartford),
  'edges': [[0, 10.03], [1, 8.88], [2, 3.36], [4, 26.72], [5, 36.22], [6, 29.85]],
  'end': 780, 'start': 690},
 {'ID': 4, 'address': location4 (Sherrill),
  'edges': [[0, 21.93], [1, 31.39], [2, 23.43], [3, 25.09], [5, 15.65], [6, 9.24]],
  'end': 1080, 'start': 960},
 {'ID': 5, 'address': location5 (Morrisville),
  'edges': [[0, 32.63], [1, 42.56], [2, 34.11], [3, 35.71], [4, 16.2], [6, 14.71]],
  'end': 720, 'start': 600},
 {'ID': 6, 'address': location6 (Oneida),
  'edges': [[0, 25.42],[1, 35.35],[2, 26.93],[3, 28.53],[4, 8.9],[5, 15.26]],
  'end': 720, 'start': 600}]
```

Figure 4: Shortened Parsed ICS files.

Walkthrough

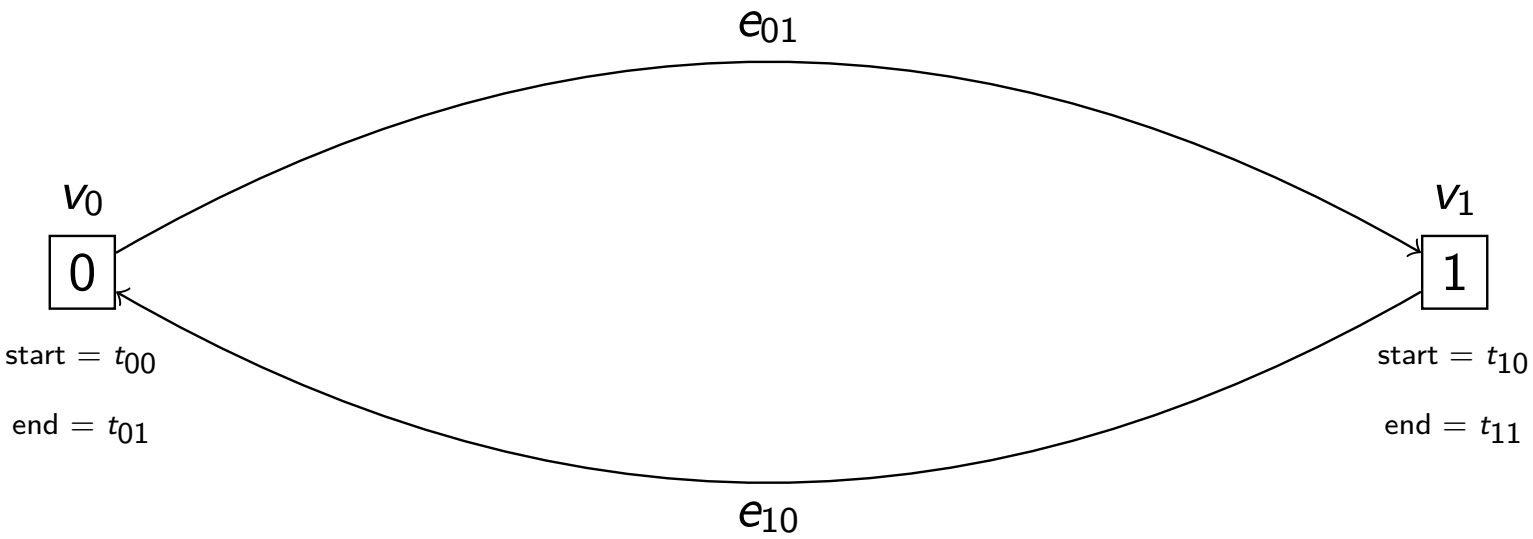


Figure 5: A trivial case of the Open House Graph.

Walkthrough

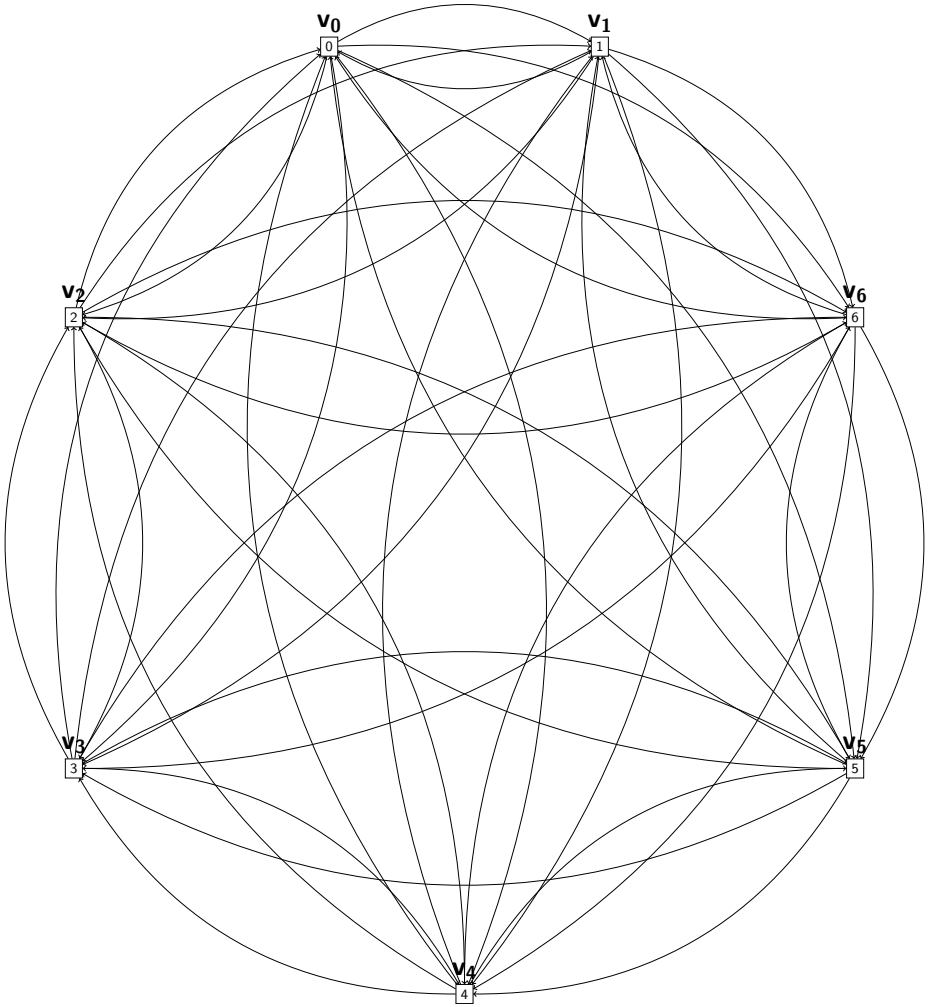


Figure 6: Open House Graph containing seven houses.

Walkthrough

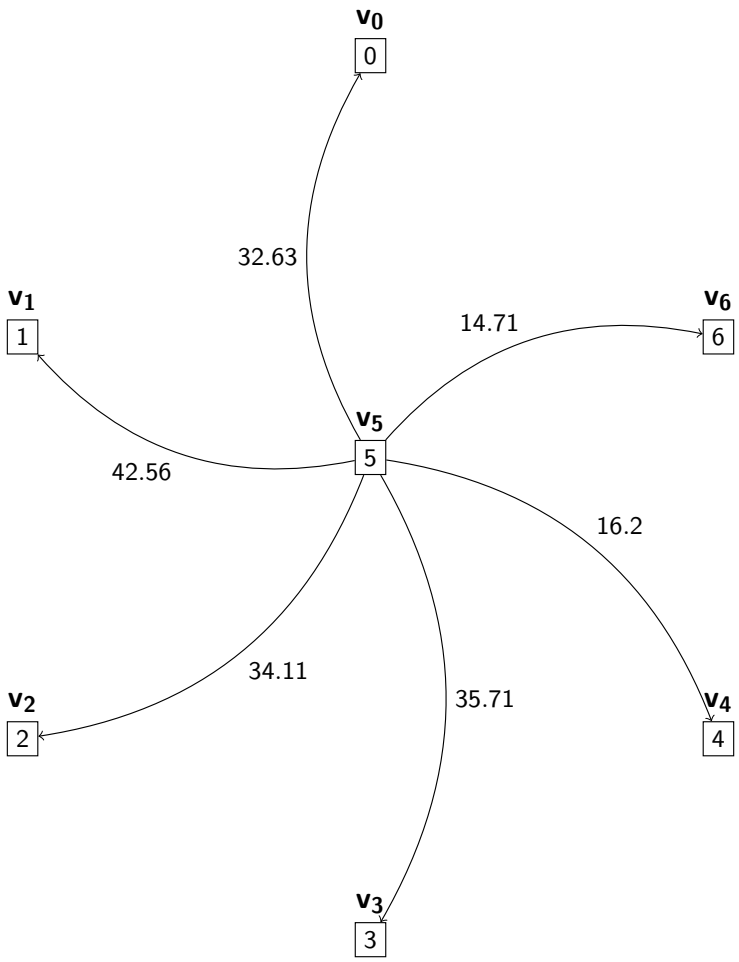
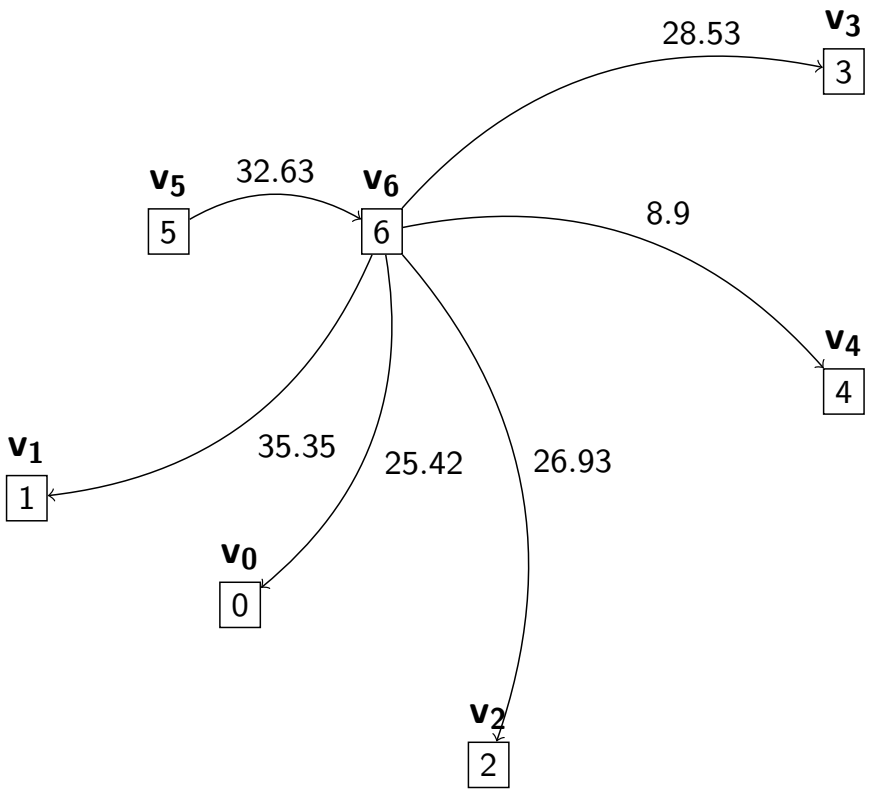


Figure 7: Step 1 - 'ID': 5, 'end': 720, 'start': 600.

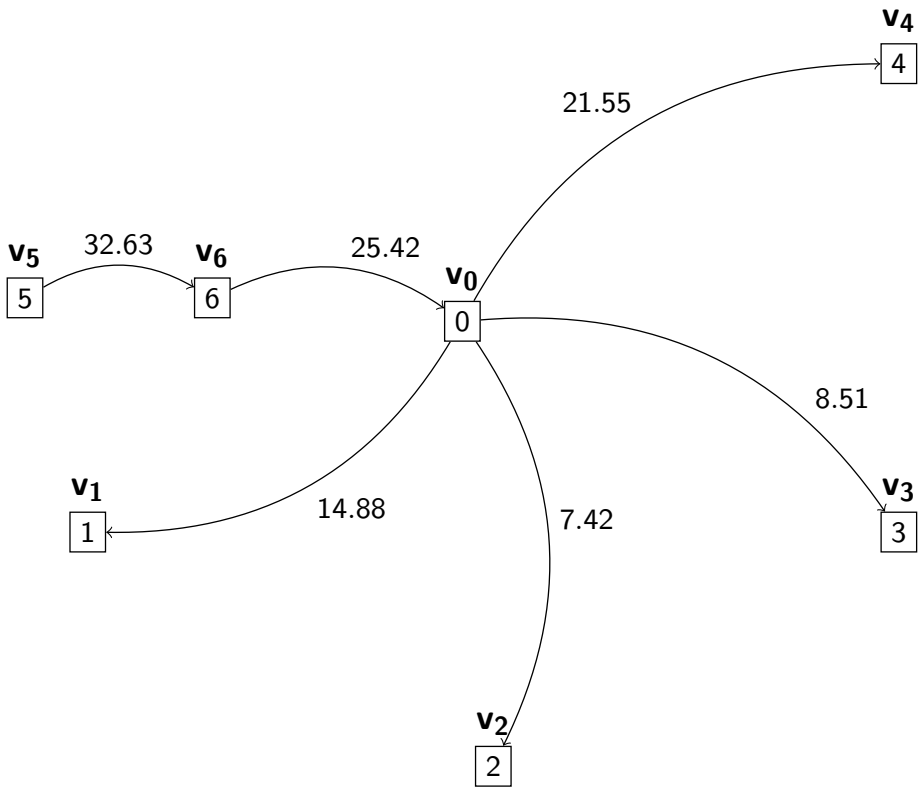
Walkthrough



Current Time = 662.63 Minutes from Midnight (600 + 30 + 32.63)

Figure 8: Step 2 - 'ID': 6, 'end': 720, 'start': 600.

Walkthrough



Current Time = 720 (*718.05*) Minutes from Midnight (662.63 + 30 + 25.42)

Figure 9: Step 3 - 'ID': 0, 'end': 810, 'start': 720.

Walkthrough

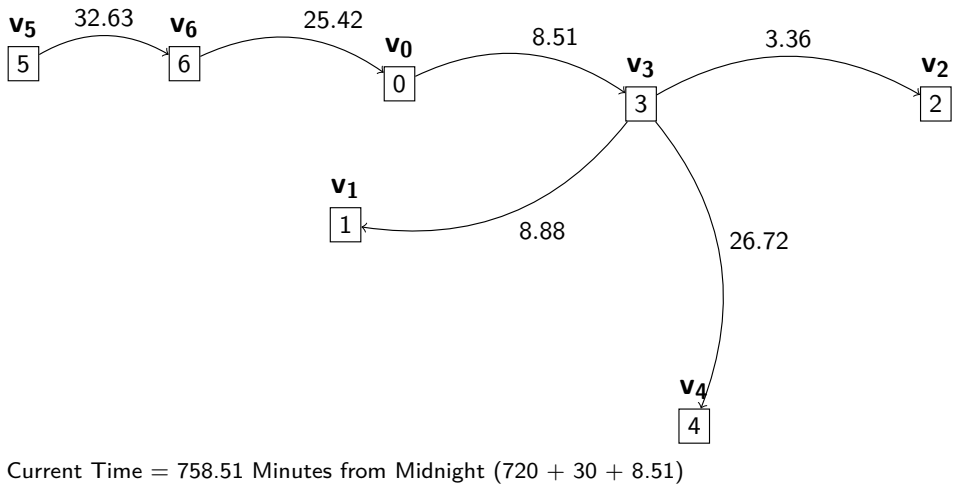


Figure 10: Step 4 - 'ID': 3, end': 780, 'start': 690.

Walkthrough

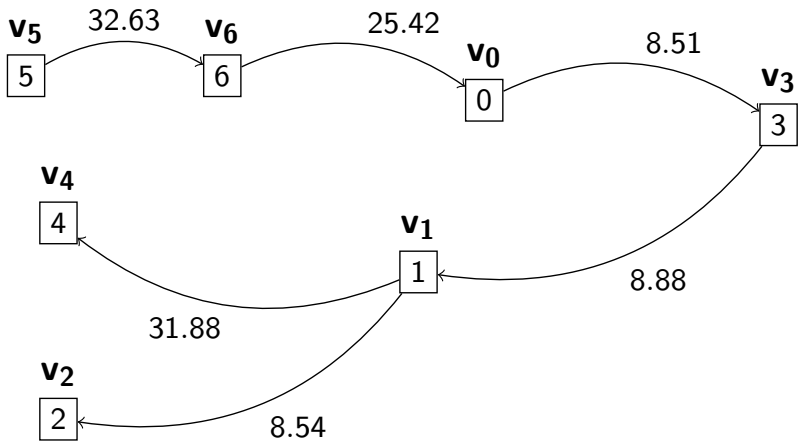


Figure 11: Step 5 - 'ID': 1, 'end': 900, 'start': 810.

Walkthrough

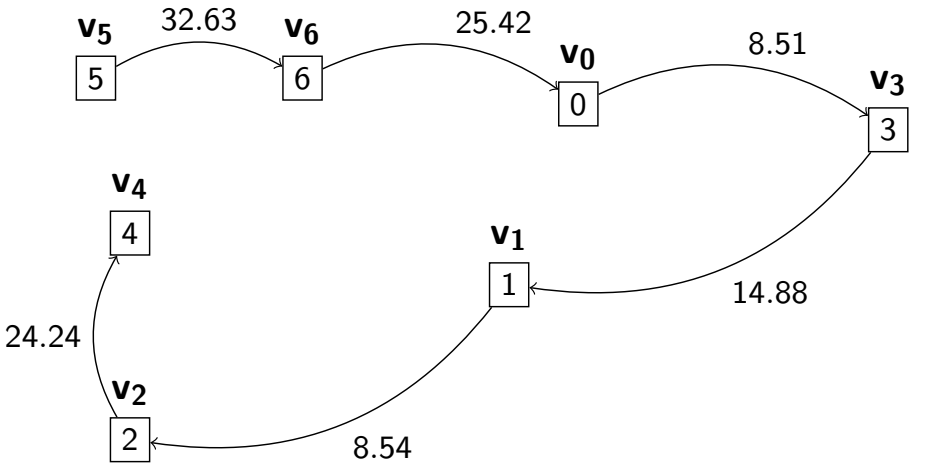


Figure 12: **Close** Step 6 - 'ID': 2, 'end': 870, 'start': 720.

Walkthrough

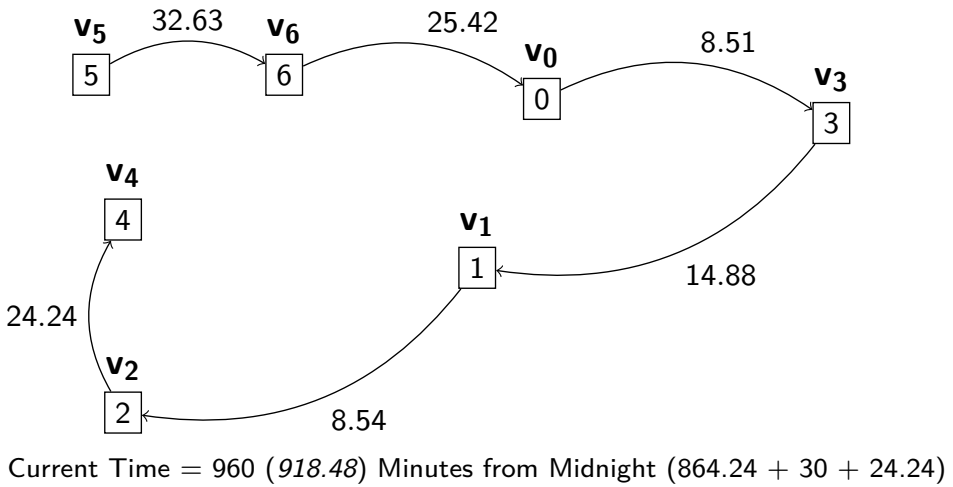


Figure 13: Step 7 - 'ID': 4, 'end': 1080, 'start': 960

Proposed Path

```
 [{'ID': 5, 'address': location5 (Morrisville),
   'edges': [[0, 32.63], [1, 42.56], [2, 34.11], [3, 35.71], [4, 16.2], [6, 14.71]],
   'end': 720, 'start': 600},
 {'ID': 6, 'address': location6 (Oneida),
   'edges': [[0, 25.42],[1, 35.35],[2, 26.93],[3, 28.53],[4, 8.9],[5, 15.26]],
   'end': 720, 'start': 600},
 {'ID': 0, 'address': location0 (Clinton),
 'edges': [[1, 14.88], [2, 7.42], [3, 8.51], [4, 21.55], [5, 31.54], [6, 25.19]],
 'end': 810, 'start': 720},
 {'ID': 3, 'address': location3 (New Hartford),
  'edges': [[0, 10.03], [1, 8.88], [2, 3.36], [4, 26.72], [5, 36.22], [6, 29.85]],
  'end': 780, 'start': 690},
{'ID': 1, 'address': location1 (Utica),
 'edges': [[0, 15.47], [2, 8.54], [3, 7.85], [4, 31.88], [5, 42.08], [6, 35.73]],
 'end': 900, 'start': 810},
{'ID': 2, 'address': location2 (New Hartford),
 'edges': [[0, 8.15], [1, 8.34], [3, 2.05], [4, 24.24], [5, 33.74], [6, 27.41]],
 'end': 870, 'start': 720},
{'ID': 4, 'address': location4 (Sherrill),
 'edges': [[0, 21.93], [1, 31.39], [2, 23.43], [3, 25.09], [5, 15.65], [6, 9.24]],
 'end': 1080, 'start': 960}]
```

Figure 14: Proposed Path.

Other Paths

```
[{'ID': 0, 'address': location0 (Clinton), 'end': 810, 'start': 720},
{'ID': 1, 'address': location1 (Utica), 'end': 900, 'start': 810},
{'ID': 2, 'address': location2 (New Hartford), 'end': 870, 'start': 720},
{'ID': 3, 'address': location3 (New Hartford), 'end': 780, 'start': 690},
{'ID': 4, 'address': location4 (Sherrill), 'end': 1080, 'start': 960},
{'ID': 5, 'address': location5 (Morrisville), 'end': 720, 'start': 600},
{'ID': 6, 'address': location6 (Oneida), 'end': 720, 'start': 600}]
```

Figure 15: Locations and Times.

```
[[5, 6, 0, 3, 1, 2, 4], [5, 6, 0, 3, 2, 1, 4], [5, 6, 2, 3, 0, 1, 4],
[5, 6, 3, 0, 1, 2, 4], [5, 6, 3, 2, 0, 1, 4], [6, 5, 0, 3, 1, 2, 4], [6, 5, 0, 3, 2, 1, 4],
[6, 5, 2, 3, 0, 1, 4], [6, 5, 3, 0, 1, 2, 4], [6, 5, 3, 0, 2, 1, 4], [6, 5, 3, 2, 0, 1, 4]]
```

Figure 16: Potential Paths.

Questions

Questions?

Acknowledgements

This work is dedicated to my parents. Without their guidance, I would not be the person I am today. I joined the United States Air Force at their suggestion, where I gained a greater appreciation for my education and the opportunities afforded to me.
I would like to thank each of my committee members: Doctors Bruno Andriamanalimanana, Roger Cavallo, and Jorge Novillo.
Gerard Aiken supplied the Esri ArcGIS Developer credits required to do this work.
I would also like to acknowledge Jennifer Tran, Sylvia Pericles, Zhushun Cai, and Oliver Medonza for aiding the initial code base where we won the Esri API Prize and the Grand Prize at Hack Upstate XI.
This work was funded by Booz Allen Hamilton tuition assistance program.


esridocker/arcgis-api-python-notebook.
https://hub.docker.com/r/esridocker/arcgis-api-python-notebook, 2019.


icalendar — wikiwand.
https://www.wikiwand.com/en/ICalendar, 2019.


jupyter/pyspark-notebook.
https://hub.docker.com/r/jupyter/pyspark-notebook, 2019.


mongo.
https://hub.docker.com/_/mongo, 2019.


R. Agarwal.
To all data scientists - the one graph algorithm you need to know.
https://towardsdatascience.com/to-all-data-scientists-the-one-graph-algorithm-you-need-to-know-59
2019.


Docker Documentation.
Install docker compose.
https://docs.docker.com/compose/install/, 2019.


Trulia.
Trulia: Real estate listings, homes for sale, housing data.
https://www.trulia.com/, 2019.