In [1]:

```
from pathlib import Path
from geopy import distance
from geopy import Nominatim
import numpy as np
import geopy
import pandas as pd
from Assignment_2_Functions import *
```

In [2]:

```
geolocator = Nominatim(user_agent='stat159-group3_assignment2')
origin_location = geolocator.reverse("37.905098, -122.272225",timeout = 10)
```

In [3]:

```
origin_lat, origin_lon = 37.905098, -122.272225
```

In [5]:

```
optimal = find_optimal(origin_lat, origin_lon, 5)
optimal1 = optimal[0]
optimal2 = optimal[1]
```

In [6]:

```
optimal1
```

Out[6]:

```
0.056713928273561295
```

In [7]:

```
optimal2
```

Out[7]:

```
0.09090909090909091
```

In [10]:

```
#### IMPORTANT!!!!
#### Since this is extremely time consuming and there is no guarantee geopy will
not shut you down, we decided to save it to a csv file for the first time runnin
g, so that we don't need to run it everytime.
#### Change timeout option if there is timeout error!!!

grid_names = []
grid_lat = []
grid_lon = []
for n in range(15):
    origin_lat = float(origin_location.raw['lat'])
    origin_lon = float(origin_location.raw['lon'])
    origin_lon = float(origin_location.raw['lon']) + n * optimal2
    for i in range(10):
        print(str(n) + '-' + str(i))
        lat = origin_lat - i * optimal1
        lon = origin_lon + i * optimal1
        try:
            location = geolocator.reverse(f'{lat, lon}', timeout = 1)
        except:
            print("You are blocked by geopy. Use our grid_points_alameda.csv dir
ectly. Or try changing timeout to greater value to avoid being blocked")
        grid_lat.append(lat)
        grid_lon.append(lon)
        grid_names.append(location.raw['display_name'])
d = {'name' : grid_names,
     'lat' : grid_lat,
     'lon' : grid_lon}
locations = pd.DataFrame(data = d)
```

```
0-0
0-1
0-2
0-3
0-4
0-5
0-6
0-7
0-8
0-9
1-0
1-1
1-2
1-3
1-4
1-5
1-6
1-7
1-8
1-9
2-0
```

```
2-1
2-2
2-3
2-4
2-5
2-6
2-7
2-8
2-9
3-0
3-1
3-2
3-3
3-4
3-5
3-6
3-7
3-8
3-9
4-0
You are blocked by geopy. Use our grid_points_alameda.csv directly.
Or try changing timeout to greater value to avoid being blocked
4-1
4-2
4-3
4-4
4-5
4-6
4-7
4-8
4-9
5-0
5-1
5-2
5-3
5-4
5-5
5-6
5-7
5-8
5-9
6-0
6-1
6-2
6-3
6-4
6-5
6-6
6-7
6-8
6-9
7-0
7-1
```

7-2

7-3

7-4

7-5

7-6

7-7

7-8

7-9

8-0

8-1

8-2

8-3

8-4

8-5

8-6

8-7

8-8

8-9

9-0

9-1

9-2

9-3

9-4

9-5

9-6

9-7

9-8

9-9

10-0

10-1

10-2

10-3

10-4

10-5

10-6

10-7

10-8

10-9

11-0

11-1

11-2

11-3

11-4

11-5

11-6

11-7

11-8

11-9

12-0

12-1

12-2

12-3

12-4

```
12-5
12-6
12-7
12-8
12-9
13-0
13-1
13-2
13-3
13-4
13-5
13-6
13-7
13-8
13-9
14-0
14-1
14-2
14-3
14-4
14-5
14-6
14-7
14-8
14-9
```

filter out points that are not in Alameda

In [13]:

```
locations = pd.DataFrame(data = d)
```

In [15]:

```
locations.head()
```

Out[15]:

	name	lat	lon
0	399, Vassar Avenue, Cragmont, Berkeley, Alamed...	37.904360	-122.272787
1	6599, Gwin Road, Oakland, Alameda County, Cali...	37.847646	-122.216073
2	Denton Place, Oakland, Alameda County, Califor...	37.790932	-122.159359
3	Towhee Trail, Ashland, Alameda County, Califor...	37.734218	-122.102645
4	24499, Sarita Street, Fairview, Alameda County...	37.677504	-122.045931

In [16]:

```
#number of grid points in Alameda
sum(['Alameda' in location for location in locations.name])
```

Out[16]:

```
31
```

In [17]:

```
#total number of grid points
len(locations)
```

Out[17]:

```
150
```

In [18]:

```
#Extract grid points in Alameda into a dataframe called 'locations_alameda'
locations_alameda = locations.loc[['Alameda' in location for location in locations.name]]
locations_alameda.head()
locations_alameda.to_csv("grid_points_alameda.csv")
```