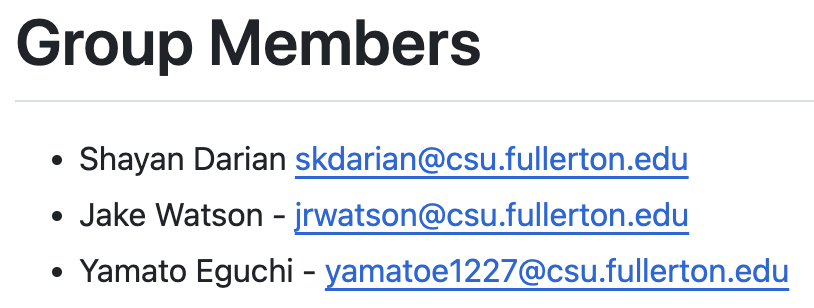
**CPSC 479 Project 2**

K-Means Clustering for Baltimore City Crime Data



Pseudocode:

input: k\_clusters, the number of clusters

data = x,y points to be clustered

main(data, k\_clusters):

k = arr of size k\_clusters //initialized from command line

for i in range(k)

k[i] = randomly selected point

cluster\_centers = arr of k\_clusters pairs with x, y vals

for i < max\_iterations, i++

cluster = find\_centroids(data, k)

//group data points by cluster

for cluster\_index in range(cluster):

cluster\_centers[cluster[cluster\_index]]->left += data[cluster\_index].x

cluster\_centers[cluster[cluster\_index]]->right += data[cluster\_index].y

temp\_cl = k.copy()

for cl in range(cluster\_centers):

k[cl] = cluster\_centers[cl] / size(cluster\_centers[cl])

//convergence check

bool change = False

for v in temp\_cl:

if k[v] != temp\_cl:

change = True

break

if not change:

break

return k

//GPU parallel section

find\_nearest\_centriod(data, k):

cluster = arr of len data //the corresponding entry will be the cluster for the x,y pair

for i entry in data:

cluster[i] = min(sq\_euclidian\_distance(i, k))

return cluster

sq\_euclidian\_distance(val, k):

min = infinity

for iteration iter in k:

distance = x^2 + y^2

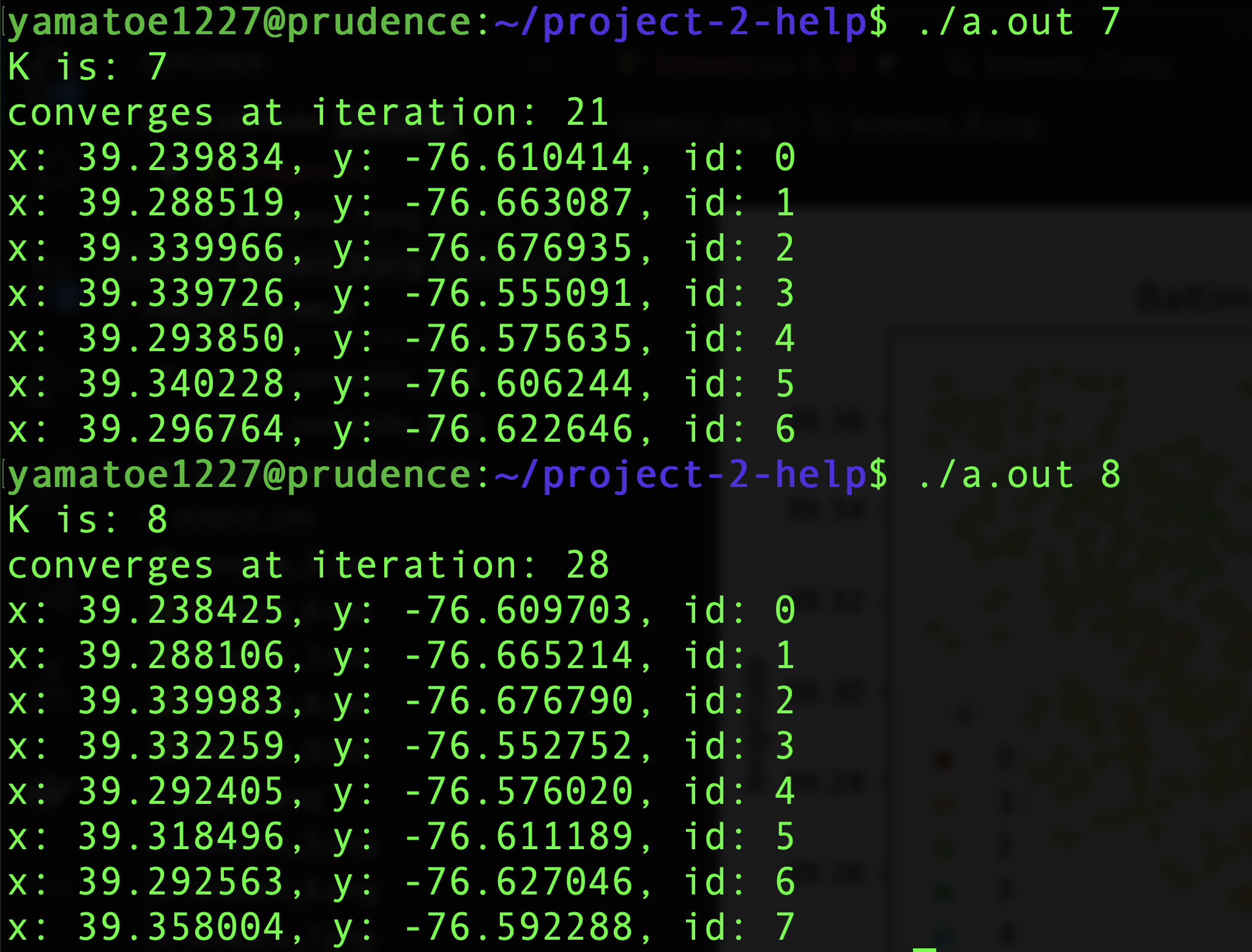
if distance < min:

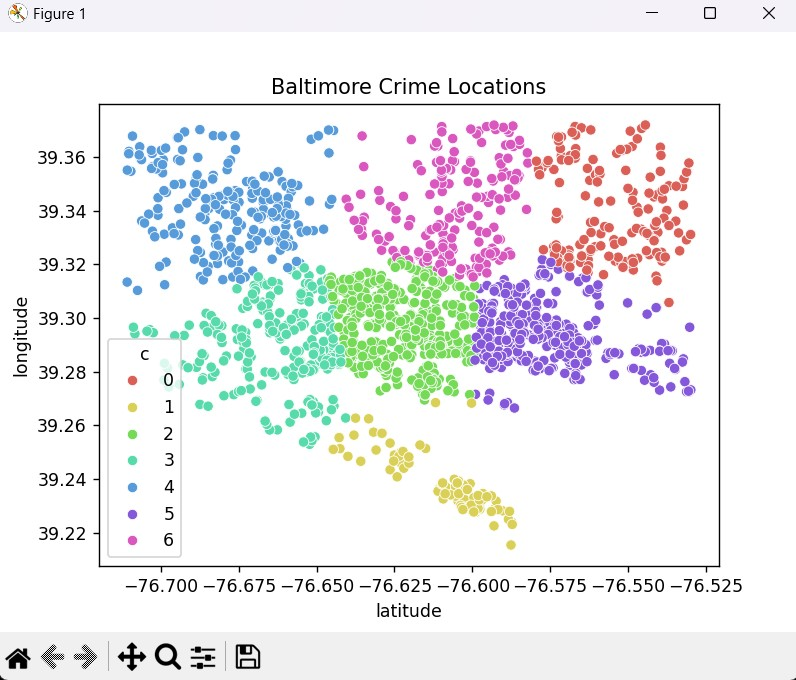
min = distance

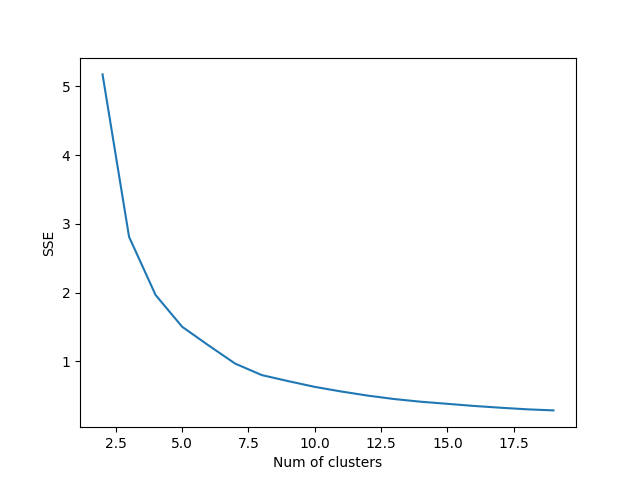
return min

To run the code, first compile it with “nvcc kmeans.cu”. Then type ./a.out k

Where k specifies the number of clusters k for the k-means clustering.



Output of visualize\_kmeans.py

Output of clustering.py

Screenshot of group members:



From the [GitHub](https://github.com/CSUF-CPSC-Fall-2023/project-2-help/tree/main).