```
from collections import defaultdict, deque, Counter
from sklearn.cluster import AgglomerativeClustering
from scipy.spatial.distance import squareform
import numpy as np
from matplotlib import pyplot as plt
from scipy.cluster.hierarchy import dendrogram
from schedule import * #place schedule.py on path
from timeit import default_timer as timer

# Performs Hierarchical Cluster Analysis on geographic locations (pre-loading
distances).
# Call script as HCA.py
# INPUT_FILENAME of company (conanme, datefirst) locations (lat, long), within
spatial temporal units (cbsa, state, year).
# _DISTANCE_FILENAME of source (lat, long), destination (lat,long), dist.
# Copyleft Ed Egan, 2022.

# ATTENTION!!! You must modify hierarchy.py as follows:
# Include the following at line 188:
    # ################## Begin Addition (section 1) ##################
    # _DISTANCE_FILENAME = r'E:\projects\hca\CBSARandom6Locs2006Lookup.txt'
    #
    # _DISTANCES={}
    # with open(_DISTANCE_FILENAME) as f:
    #     next(f) #loose the header row
    #     for line in f:
    #         line=line.rstrip('\r\n')
    #         parts = line.split('\t')
    #         source=(float(parts[0]),float(parts[1]))
    #         dest=(float(parts[2]),float(parts[3]))
    #         dist=parts[4]
    #         if source in _DISTANCES:
    #             _DISTANCES[source][dest]=dist
    #         else:
    #              DISTANCES[source]={}
    #              _DISTANCES[source][dest]=dist
    # ################## End Addition (section 1) ##################

# Comment out the current distance method on ln 732
    # y = distance.pdist(y, metric)

# Include the following at line 188:
    # ################## Begin Addition (section 2) ##################
    #         points=[tuple(row) for row in y]
    #         sqform=np.zeros((len(points),len(points)))
    #         for i,source in enumerate(points):
    #             for j,dest in enumerate(points):
    #                 sqform[i,j]=_DISTANCES[source][dest]
    #         y=distance.squareform(sqform)
    # ################## End Addition (section 2) ##################

# Set _DISTANCE_FILENAME in hierarchy.py ln 189
# Set the INPUT_FILENAME and OUTPUT_FILENAME below.
INPUT_FILENAME = r'E:\projects\hca\SelectCBSARandomForHCA.txt'
OUTPUT_FILENAME =r'E:\projects\hca\SelectCBSARandomForHCA_results.txt'

def parse_colevel_file(path, has_header=True):
    """
    Returns:
        A mapping from a (cbsa, state, year) tuple to a list of (lat, long,
        companyname, datefirstinv) tuples.
    """
    ret = defaultdict(deque)
    with open(path) as f:
```

```
        if has_header:
            next(f)
        for line in f:
            line=line.rstrip('\r\n')
            parts = line.split('\t')
            cbsa, year, lat, lon=parts[0:4]
            ret[(cbsa, year)].append((lat, lon))
    return ret

#Main body
start = timer()
print('Parsing ' + INPUT_FILENAME)
res = parse_colevel_file(INPUT_FILENAME)
print('Found {} unique (city, state, year) tuples in {}'.format(len(res),
INPUT_FILENAME))
print('Generating output in {}'.format(OUTPUT FILENAME))
with open(OUTPUT_FILENAME, 'w', newline='') as c:
    print("cbsa\tyear\tlayer_no\tcluster_no\tlat\tlon",file=c)
    model=AgglomerativeClustering(distance_threshold=0, n_clusters=None)
    for key in res.keys():
        keystr="\t".join(map(str,key)) #Note that this has an issue when key
        is already a str!
        data = res[key]
        print('Agglomerating ' + str(key) + ' with ' + str(len(data)) + '
        points')
        points=[(float(x),float(y)) for (x,y) in data]
        obs_list=[(float(x),float(y)) for (x,y) in data]
        model.fit(points)
        schedule=get_schedule(model)
        for layer_no in dict.keys(schedule):
            for i in range(0,len(obs_list)):
                obs_str="\t".join(map(str,obs_list[i]))
                cluster_no=int(schedule[layer_no][i])

                print(keystr,"\t",layer_no,"\t",cluster_no,"\t",obs_str,file=c
timediff = timer()-start
print('Run Completed in',timediff)
print('Thank Ed and his former McNair minions.')
# No warrantee whatsoever.
```