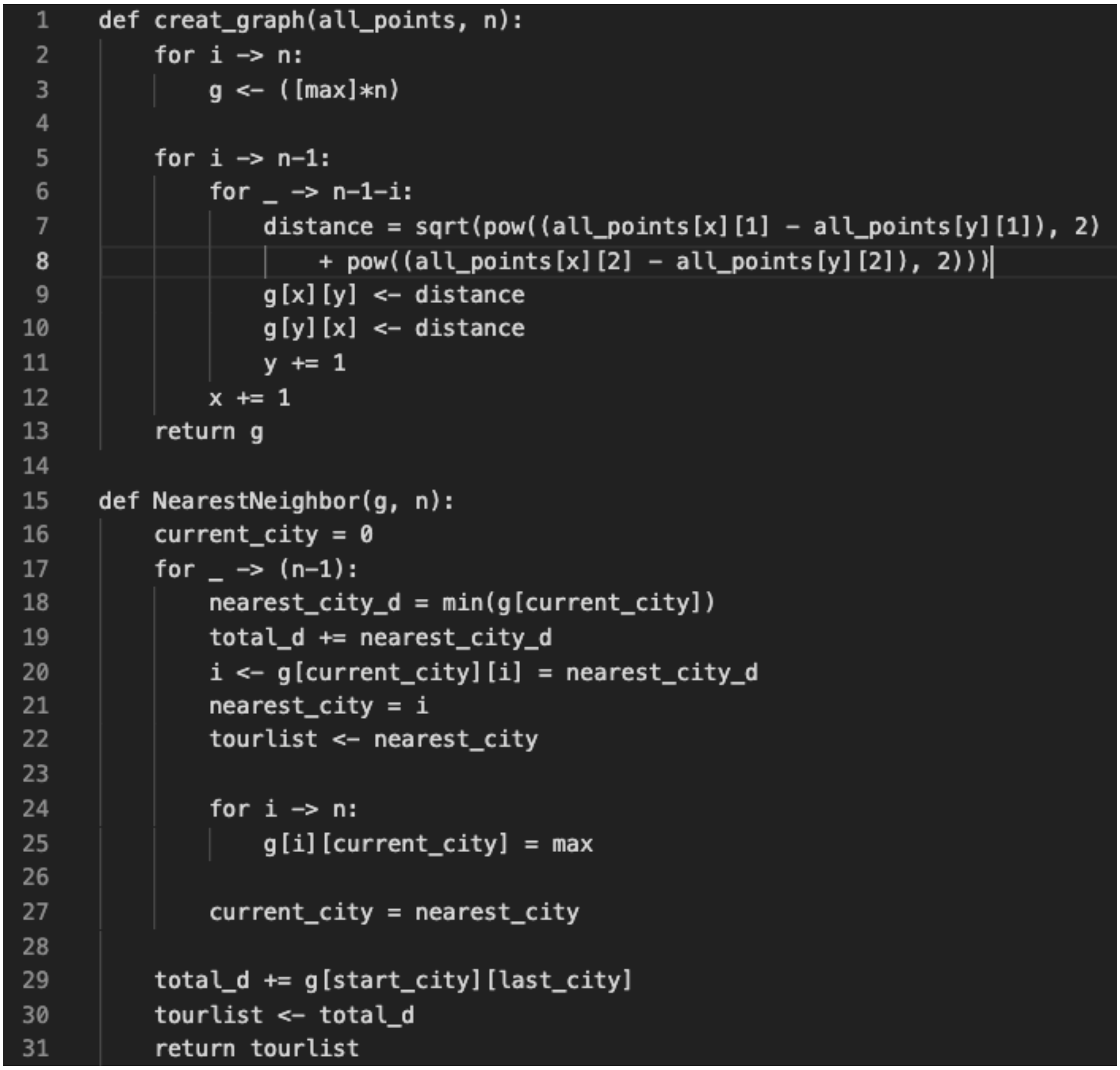
CS325-Homework6

1. **Description:** For this program, I used the nearest neighbor algorithm to “solve” the TSP. Nearest neighbor algorithm is greedy approximations. Each time, the city closest to the current city is selected. In this way, the merchant takes the shortest distance every time.

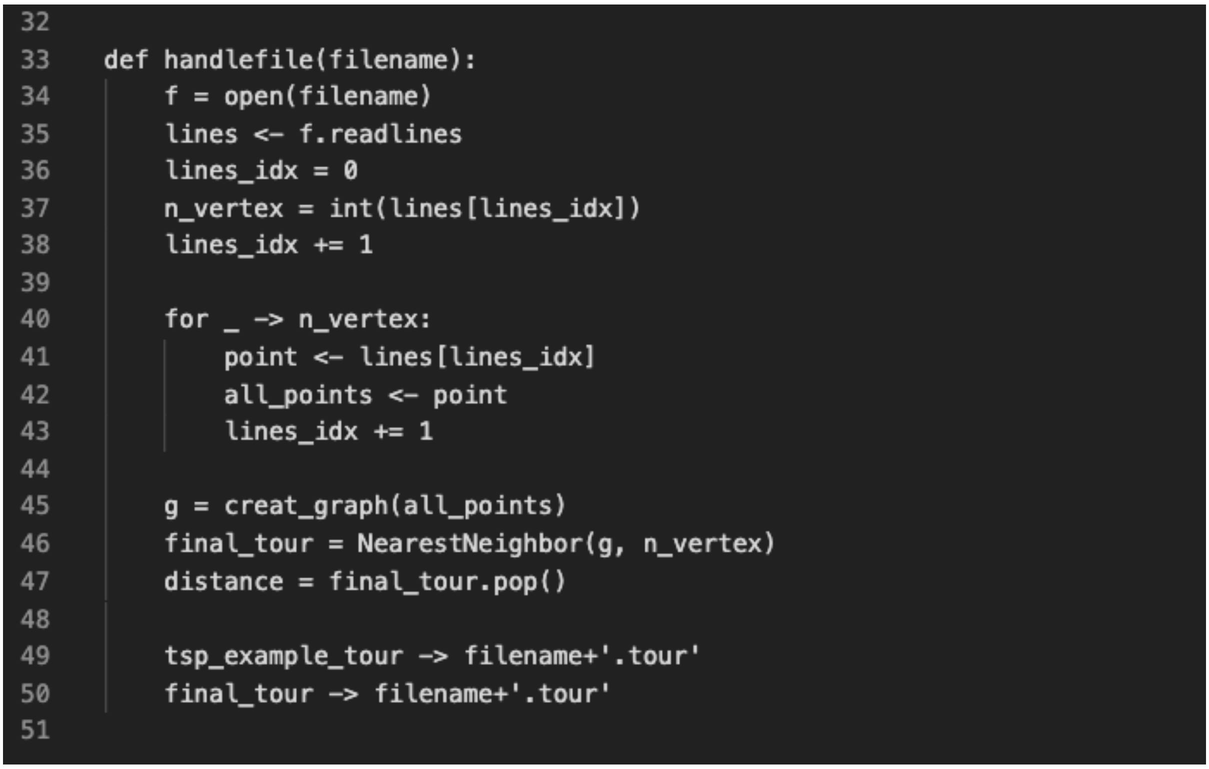
   For this algorithm, in order to ensure that merchants do not go to duplicate cities, every time a merchant arrives in a new city, the distance from all other cities to the previous city needs to be set to a maximum value. In this way, when using the greedy algorithm, the merchants will not pass through the cities that they have visited before.

   In my code, I used tuple list to store the graph at first. But when I look for the distance from all other cities to the current city, I need to perform a for loop. And I have to sort their distance. This greatly increases the running time. Then I used a two-dimensional array to store the graph. This makes it easy to find the distance between two cities.

2. **Pseudocode:**



```
def creat_graph(all_points, n):
    for i -> n:
        g <- ([max]*n)

    for i -> n-1:
        for _ -> n-1-i:
            distance = sqrt(pow((all_points[x][1] - all_points[y][1]), 2)
                + pow((all_points[x][2] - all_points[y][2]), 2)))
            g[x][y] <- distance
            g[y][x] <- distance
            y += 1
        x += 1
    return g

def NearestNeighbor(g, n):
    current_city = 0
    for _ -> (n-1):
        nearest_city_d = min(g[current_city])
        total_d += nearest_city_d
        i <- g[current_city][i] = nearest_city_d
        nearest_city = i
        tourlist <- nearest_city

        for i -> n:
            g[i][current_city] = max

        current_city = nearest_city

    total_d += g[start_city][last_city]
    tourlist <- total_d
    return tourlist
```



```

def handlefile(filename):
    f = open(filename)
    lines <- f.readlines
    lines_idx = 0
    n_vertex = int(lines[lines_idx])
    lines_idx += 1

    for _ -> n_vertex:
        point <- lines[lines_idx]
        all_points <- point
        lines_idx += 1

    g = creat_graph(all_points)
    final_tour = NearestNeighbor(g, n_vertex)
    distance = final_tour.pop()

    tsp_example_tour -> filename+'.tour'
    final_tour -> filename+'.tour'

```

3. Theoretical running time: O(N^2)

   Two nested loops in function “NearestNeighbor”

   2-approximation algorithm for TSP

4. **Summary of results:**

   Ratio 0: 1.00

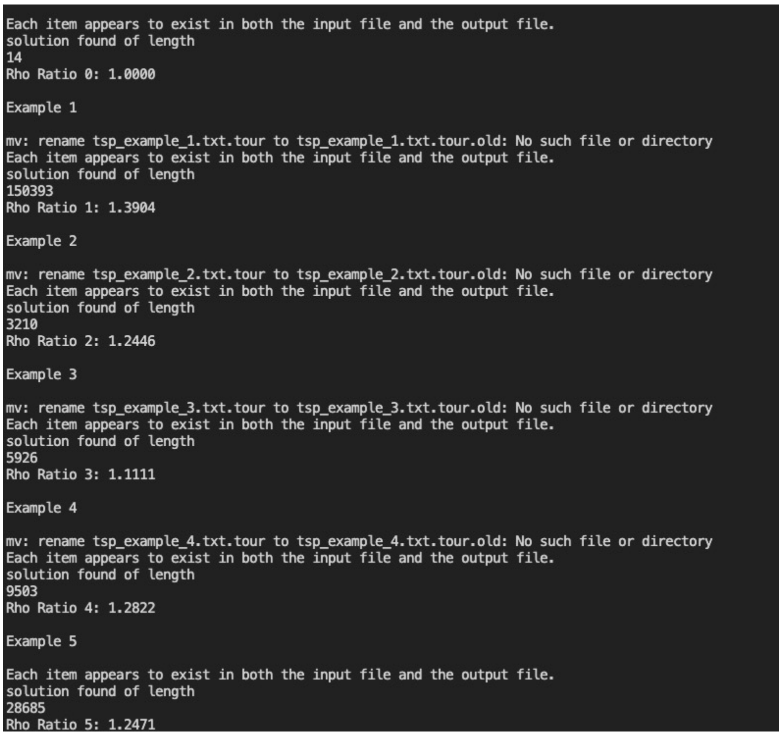
   Ratio 1: 1.39

   Ratio 2: 1.24

   Ratio 3: 1.11

   Ratio 4: 1.28

   Ratio 5: 1.25



```
Each item appears to exist in both the input file and the output file.
solution found of length
14
Rho Ratio 0: 1.0000

Example 1

mv: rename tsp_example_1.txt.tour to tsp_example_1.txt.tour.old: No such file or directory
Each item appears to exist in both the input file and the output file.
solution found of length
150393
Rho Ratio 1: 1.3904

Example 2

mv: rename tsp_example_2.txt.tour to tsp_example_2.txt.tour.old: No such file or directory
Each item appears to exist in both the input file and the output file.
solution found of length
3210
Rho Ratio 2: 1.2446

Example 3

mv: rename tsp_example_3.txt.tour to tsp_example_3.txt.tour.old: No such file or directory
Each item appears to exist in both the input file and the output file.
solution found of length
5926
Rho Ratio 3: 1.1111

Example 4

mv: rename tsp_example_4.txt.tour to tsp_example_4.txt.tour.old: No such file or directory
Each item appears to exist in both the input file and the output file.
solution found of length
9503
Rho Ratio 4: 1.2822

Example 5

Each item appears to exist in both the input file and the output file.
solution found of length
28685
Rho Ratio 5: 1.2471
```