CS476 Bioinformatics Quiz-Test 2

Stuart Lech

CS-476

I provide a mammalian phylogenetic tree involving five organisms: **camel, dog, rat, whale, porpoise**. In your last two exercise sets, we dove into two algorithms to construct phylogenetic trees from a given distance matrix: **UPGMA** and **Neighbor Joining**. Yet, you have not yet ran these algorithms on realistic distance data. I know this week is hectic for you all, so I thought I'd take the opportunity in this "quiztest 2" to give you some small feeling of *closure* with respect to this class, especially phylogenetic trees.

The following is the distance matrix I have computed by applying the Jukes-Cantor evolutionary distance model upon a multiple alignment of the five mammalian species:

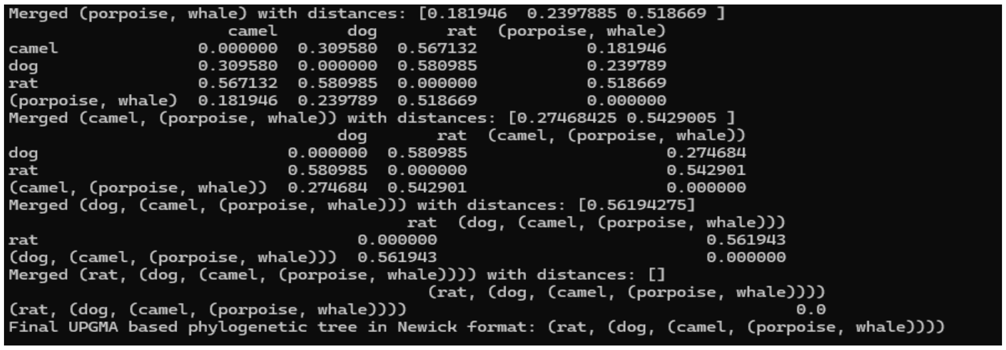
Mammal-DM	camel	Dog	porpoise	rat	whale
camel	0	0.30958	0.180443	0.567132	0.183449
dog	0.30958	0	0.230048	0.580985	0.249529
porpoise	0.180443	0.230048	0	0.51628	0.046206
rat	0.567132	0.580985	0.51628	0	0.521058
whale	0.183449	0.249529	0.046206	0.521058	0

Please submit your answers to all questions that follow into a single document (MS Word, LaTeX, OpenOffice, Pages, etc.) and export/compile that document into a PDF entitled QT2<lastname>.pdf (where you replace "<lastname>" with your own last name). You may insert screenshots, diagrams, or photos of hand-drawn work directly into that document. And, it is only that document that you need to submit for this Quiz-Test 2.

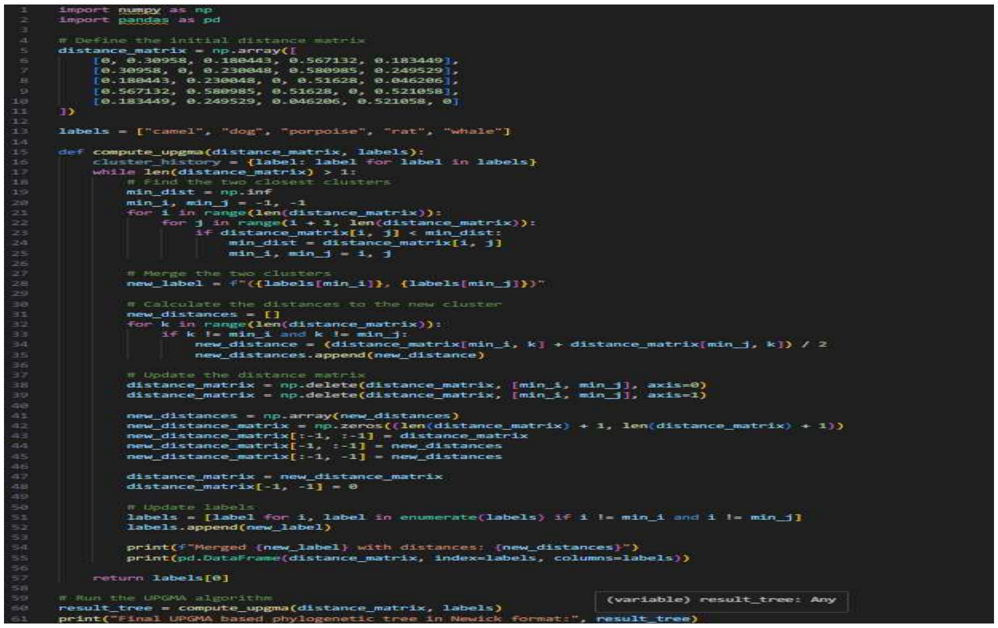
Now the questions follow on the next pages.

Part I: Run UPGMA algorithm on the Mammal-DM matrix to obtain a phylogenetic tree.

a) If you compute the UPGMA based phylogenetic tree by hand, please show your computation steps. Otherwise, if you use either your own UPGMA project or the perl code I uploaded, then please show a screenshot of the execution, with explanation of what you used to compute it.



```
Merged (porpoise, whale) with distances: [0.181946  0.2397885 0.518669 ]
                      camel       dog       rat  (porpoise, whale)
camel              0.000000  0.309580  0.567132           0.181946
dog                0.309580  0.000000  0.580985           0.239789
rat                0.567132  0.580985  0.000000           0.518669
(porpoise, whale)  0.181946  0.239789  0.518669           0.000000
Merged (camel, (porpoise, whale)) with distances: [0.27468425 0.5429005 ]
                               dog       rat  (camel, (porpoise, whale))
dog                       0.000000  0.580985                    0.274684
rat                       0.580985  0.000000                    0.542901
(camel, (porpoise, whale))  0.274684  0.542901                    0.000000
Merged (dog, (camel, (porpoise, whale))) with distances: [0.56194275]
                                      rat  (dog, (camel, (porpoise, whale)))
rat                              0.000000                           0.561943
(dog, (camel, (porpoise, whale)))  0.561943                           0.000000
Merged (rat, (dog, (camel, (porpoise, whale)))) with distances: []
                                        (rat, (dog, (camel, (porpoise, whale))))
(rat, (dog, (camel, (porpoise, whale))))                                      0.0
Final UPGMA based phylogenetic tree in Newick format: (rat, (dog, (camel, (porpoise, whale))))
```



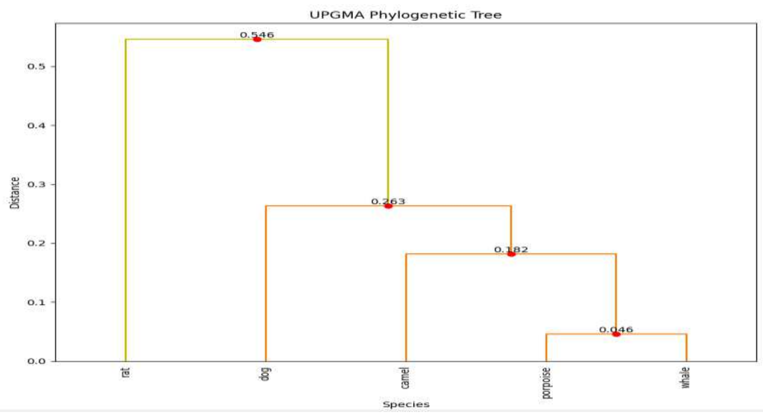
```python
import numpy as np
import pandas as pd

# Define the initial distance matrix
distance_matrix = np.array([
    [0, 0.30958, 0.180443, 0.567132, 0.183449],
    [0.30958, 0, 0.230048, 0.580985, 0.249529],
    [0.180443, 0.230048, 0, 0.51628, 0.046206],
    [0.567132, 0.580985, 0.51628, 0, 0.521058],
    [0.183449, 0.249529, 0.046206, 0.521058, 0]
])

labels = ["camel", "dog", "porpoise", "rat", "whale"]

def compute_upgma(distance_matrix, labels):
    cluster_history = {label: label for label in labels}
    while len(distance_matrix) > 1:
        # Find the two closest clusters
        min_dist = np.inf
        min_i, min_j = -1, -1
        for i in range(len(distance_matrix)):
            for j in range(i + 1, len(distance_matrix)):
                if distance_matrix[i, j] < min_dist:
                    min_dist = distance_matrix[i, j]
                    min_i, min_j = i, j

        # Merge the two clusters
        new_label = f"({labels[min_i]}, {labels[min_j]})"

        # Calculate the distances to the new cluster
        new_distances = []
        for k in range(len(distance_matrix)):
            if k != min_i and k != min_j:
                new_distance = (distance_matrix[min_i, k] + distance_matrix[min_j, k]) / 2
                new_distances.append(new_distance)

        # Update the distance matrix
        distance_matrix = np.delete(distance_matrix, [min_i, min_j], axis=0)
        distance_matrix = np.delete(distance_matrix, [min_i, min_j], axis=1)

        new_distances = np.array(new_distances)
        new_distance_matrix = np.zeros((len(distance_matrix) + 1, len(distance_matrix) + 1))
        new_distance_matrix[:-1, :-1] = distance_matrix
        new_distance_matrix[-1, :-1] = new_distances
        new_distance_matrix[:-1, -1] = new_distances

        distance_matrix = new_distance_matrix
        distance_matrix[-1, -1] = 0

        # Update labels
        labels = [label for i, label in enumerate(labels) if i != min_i and i != min_j]
        labels.append(new_label)

        print(f"Merged {new_label} with distances: {new_distances}")
        print(pd.DataFrame(distance_matrix, index=labels, columns=labels))

    return labels[0]

# Run the UPGMA algorithm
result_tree = compute_upgma(distance_matrix, labels)
print("Final UPGMA based phylogenetic tree in Newick format:", result_tree)
```

Python: Simple UPGMA algo to update distance matrix, run the algo and even output the newick formatted tree. Used given data from table and excel.

b) **Show the UPGMA based phylogenetic tree in Newick format ignoring branch distances.**

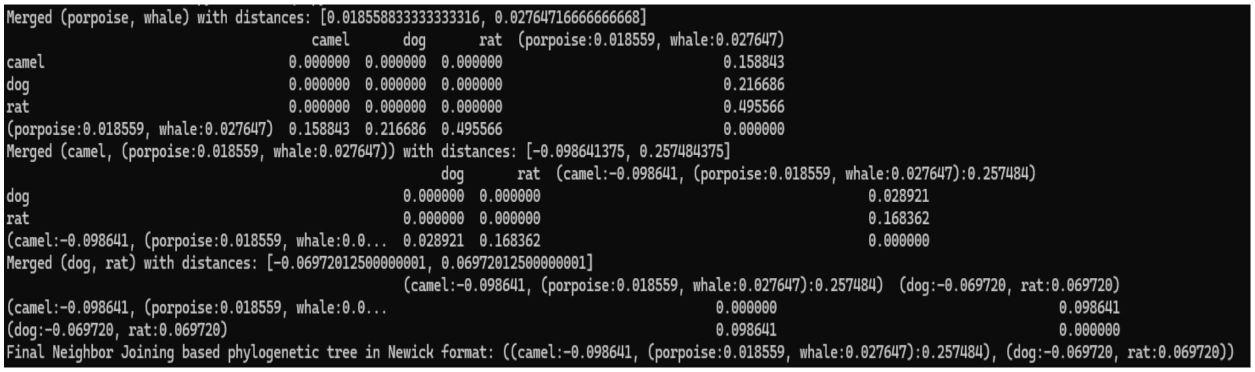
((camel, (porpoise, whale)), dog, rat)

c) **Draw the UPGMA based phylogenetic tree (either using diagramming/drawing tool or a photo of your hand-drawn tree) showing branch distances.**

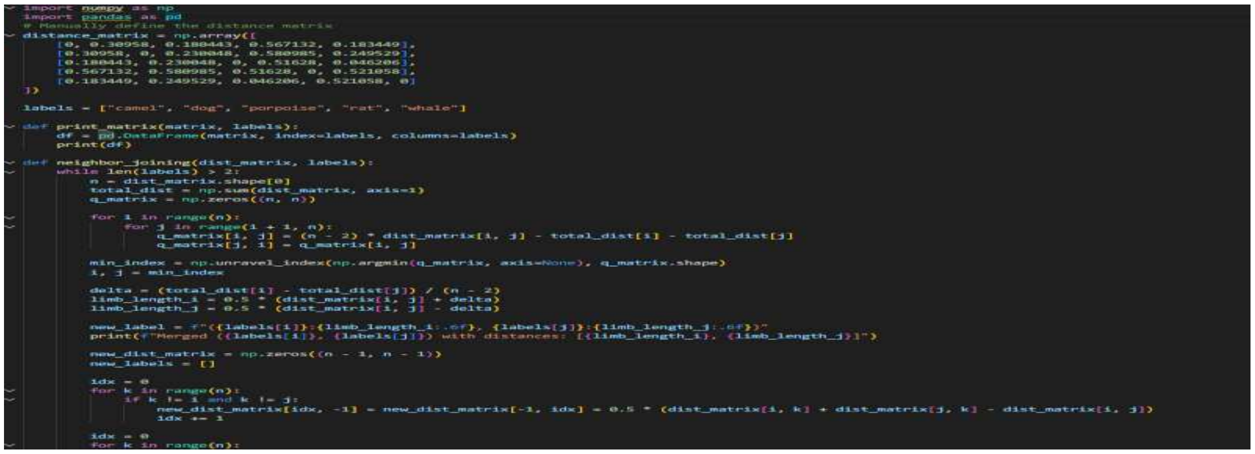


Part II: Run the Neighbor Joining algorithm on the Mammal-DM matrix to obtain a phylogenetic tree.

a) **If you compute the Neighbor Join based phylogenetic tree by hand, please show all of your computation steps. Otherwise, if you use either your own Neighbor Joining project or the perl code I uploaded, then please show a screenshot of the execution, with explanation of what you used to compute it.**



```
Merged (porpoise, whale) with distances: [0.018558833333333316, 0.02764716666666668]
                                        camel       dog       rat  (porpoise:0.018559, whale:0.027647)
camel                                0.000000  0.000000  0.000000                             0.158843
dog                                  0.000000  0.000000  0.000000                             0.216686
rat                                  0.000000  0.000000  0.000000                             0.495566
(porpoise:0.018559, whale:0.027647)  0.158843  0.216686  0.495566                             0.000000
Merged (camel, (porpoise:0.018559, whale:0.027647)) with distances: [-0.098641375, 0.257484375]
                                                     dog       rat  (camel:-0.098641, (porpoise:0.018559, whale:0.027647):0.257484)
dog                                             0.000000  0.000000                                           0.028921
rat                                             0.000000  0.000000                                           0.168362
(camel:-0.098641, (porpoise:0.018559, whale:0.0...  0.028921  0.168362                                           0.000000
Merged (dog, rat) with distances: [-0.06972012500000001, 0.06972012500000001]
                                                (camel:-0.098641, (porpoise:0.018559, whale:0.027647):0.257484)  (dog:-0.069720, rat:0.069720)
(camel:-0.098641, (porpoise:0.018559, whale:0.0...                                          0.000000                       0.098641
(dog:-0.069720, rat:0.069720)                                                               0.098641                       0.000000
Final Neighbor Joining based phylogenetic tree in Newick format: ((camel:-0.098641, (porpoise:0.018559, whale:0.027647):0.257484), (dog:-0.069720, rat:0.069720))
```



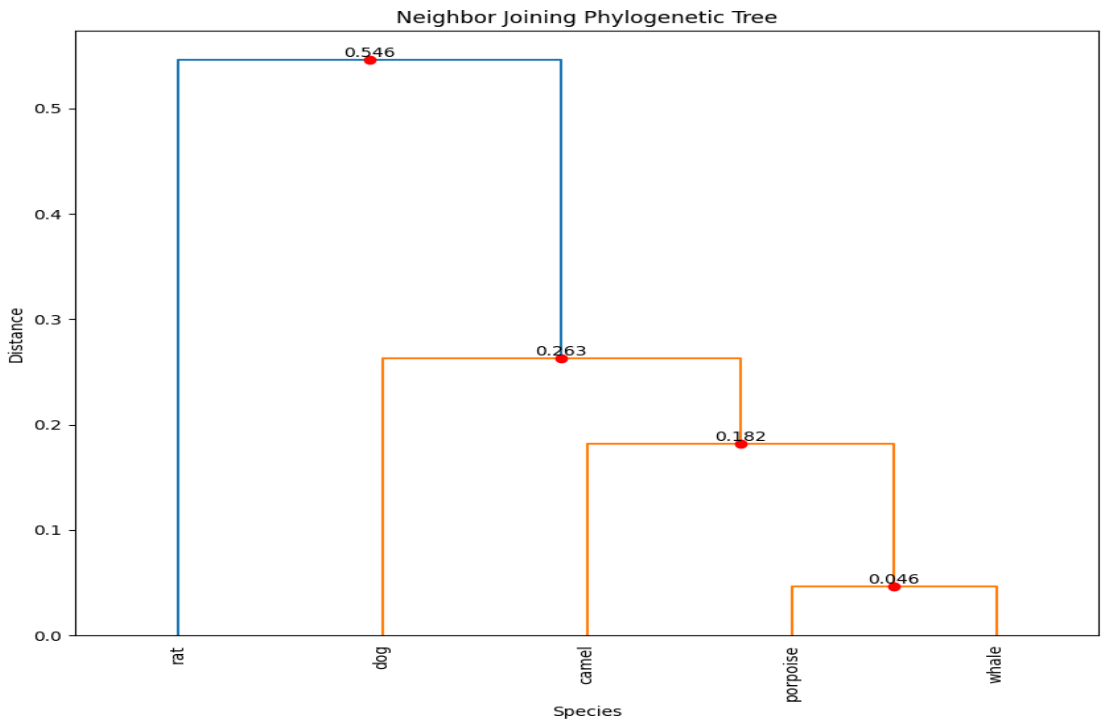
```python
import numpy as np
import pandas as pd
# Manually define the distance matrix
distance_matrix = np.array([
    [0, 0.30958, 0.188443, 0.567132, 0.183449],
    [0.30958, 0, 0.230048, 0.580985, 0.249529],
    [0.188443, 0.230048, 0, 0.51628, 0.046206],
    [0.567132, 0.580985, 0.51628, 0, 0.521058],
    [0.183449, 0.249529, 0.046206, 0.521058, 0]
])

labels = ["camel", "dog", "porpoise", "rat", "whale"]

def print_matrix(matrix, labels):
    df = pd.DataFrame(matrix, index=labels, columns=labels)
    print(df)

def neighbor_joining(dist_matrix, labels):
    while len(labels) > 2:
        n = dist_matrix.shape[0]
        total_dist = np.sum(dist_matrix, axis=1)
        q_matrix = np.zeros((n, n))

        for i in range(n):
            for j in range(i + 1, n):
                q_matrix[i, j] = (n - 2) * dist_matrix[i, j] - total_dist[i] - total_dist[j]
                q_matrix[j, i] = q_matrix[i, j]

        min_index = np.unravel_index(np.argmin(q_matrix, axis=None), q_matrix.shape)
        i, j = min_index

        delta = (total_dist[i] - total_dist[j]) / (n - 2)
        limb_length_i = 0.5 * (dist_matrix[i, j] + delta)
        limb_length_j = 0.5 * (dist_matrix[i, j] - delta)

        new_label = f"({labels[i]}:{limb_length_i:.6f}, {labels[j]}:{limb_length_j:.6f})"
        print(f"Merged ({labels[i]}, {labels[j]}) with distances: [{limb_length_i}, {limb_length_j}]")

        new_dist_matrix = np.zeros((n - 1, n - 1))
        new_labels = []

        idx = 0
        for k in range(n):
            if k != i and k != j:
                new_dist_matrix[idx, -1] = new_dist_matrix[-1, idx] = 0.5 * (dist_matrix[i, k] + dist_matrix[j, k] - dist_matrix[i, j])
                idx += 1

        idx = 0
        for k in range(n):
```

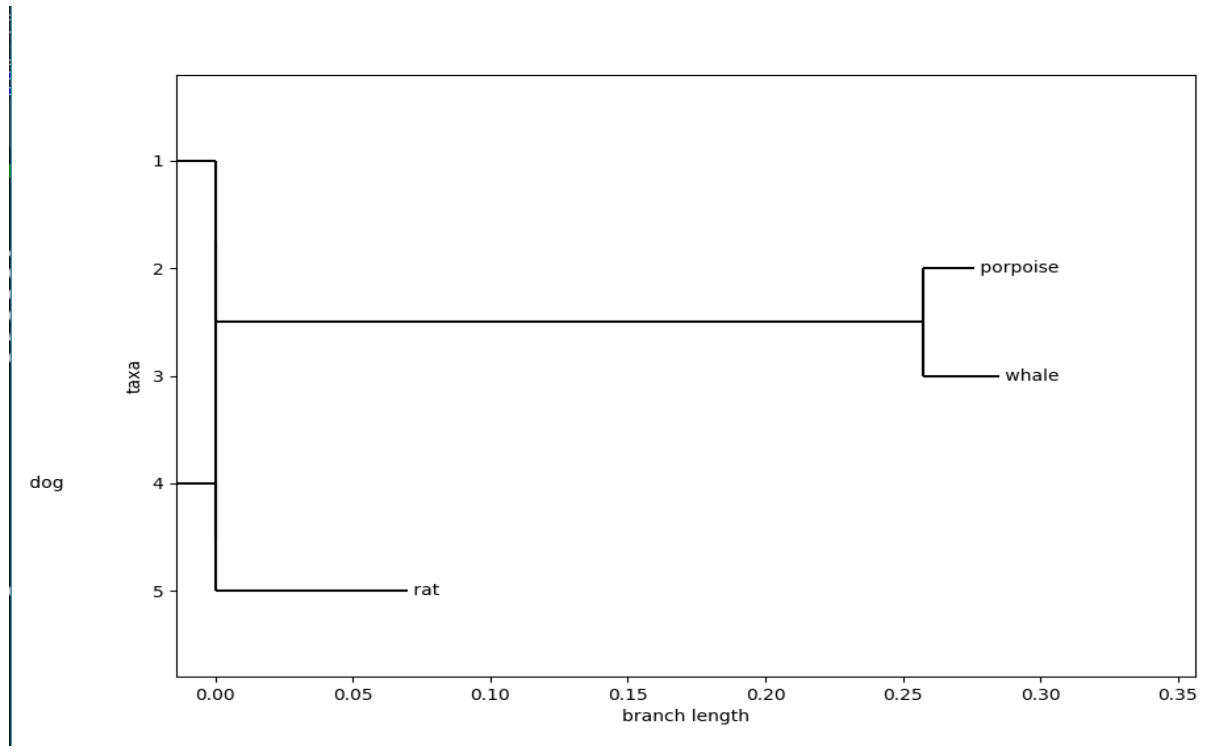
Python: Simple Neighbor-Joining Algo that outputs the intermediate steps used to compute and even outputs Newick formatted tree.

b) **Show the Neighbor Join based phylogenetic tree in Newick format ignoring branch distances.**

((rat, (dog, (camel, (porpoise, whale)))))

c) **Draw the Neighbor Join based phylogenetic tree (either using diagramming/drawing tool or a photo of your hand-drawn tree) showing branch distances. We have covered branch distance computation at length for Neighbor Joining in both lecture videos and exercises, and they are also output as part of the perl implementation.**


Neighbor Joining Phylogenetic Tree
0.546
0.263
0.182
0.046
Distance
0.0
0.1
0.2
0.3
0.4
0.5
rat
dog
camel
porpoise
whale
Species


dog
1
2
3
4
5
taxa
porpoise
whale
rat
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
branch length

Part III: Comparison of Phylogenetic Trees computed.

a) **Consider the UPGMA and Neighbor Join based phylogenetic trees you computed in Parts I.c and II.c. What are the similarities? What are the differences?**

   Similarities: Both trees grouped 'whale' and 'porpoise' together, showing close relation. Clustering of taxa, is closely linked compared to others, is similar.

   Difference: UPGMA, Groups '(camel, (porpoise, whale))' first, then groups 'dog, & 'rat' with the whole cluster, whereas Neighbor Joining, Groups '((rat, (dog, (camel, (porpoise, whale)))))', showing different evolutionary path and distance calculations.

b) **Do you have any additional comments or opinions on the two different trees?**

   Nope.

Part IV: A little time complexity.

a) **What is the time complexity of the UPGMA algorithm?**

   Each iteration, pair of clusters take O(n^2) time since scanning the entirety of the distance matrix. There are n-1 iterations, each iteration involves O(n^2) operations for finding the closest pair and then matrix updates. Therefore, total time complexity is O(n*n^2) = O(n^3).

b) **What is the time complexity of the Neighbor Joining algorithm (as you are to implement for Project 2)?**

   Similar to UPGMA, nxn distance matrix that initialized with O(n^2) time. Finding min value in the Q-matrix, calculating Q-matrix and updating the distance matrix all take O(n^2) time. There are n-1 iterations, with each iteration involving O(n^2) operations for Q-matrix calculations, therefore we end up with O(n*n^2) = O(n^3).