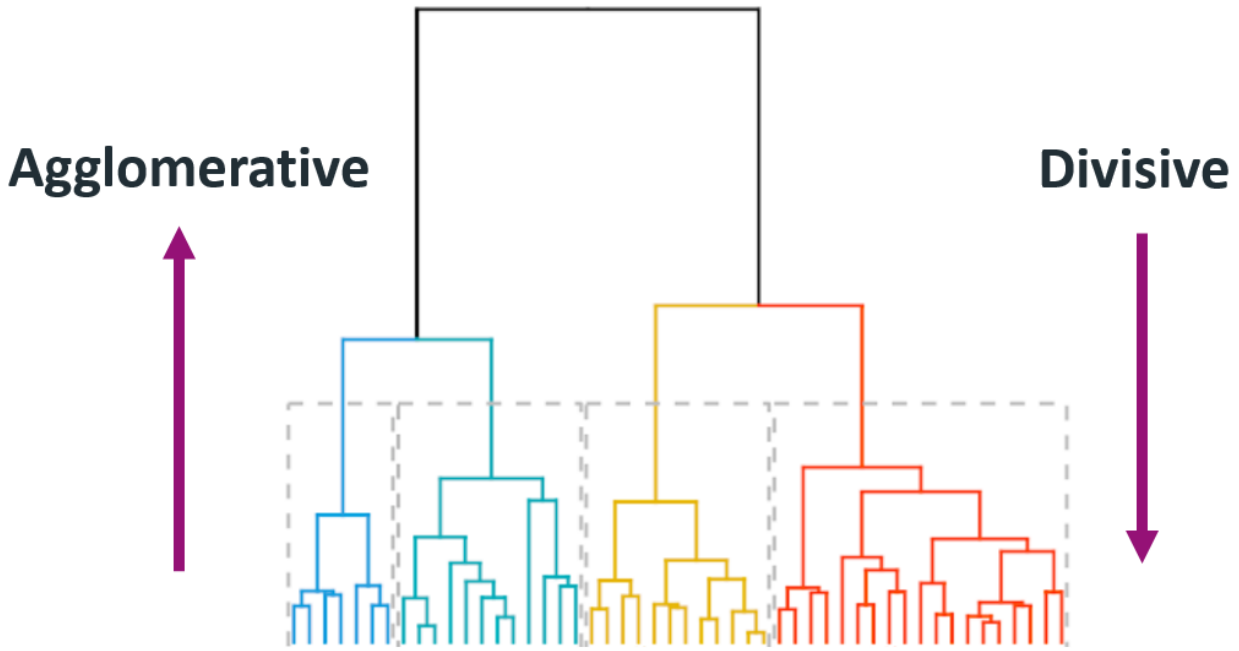
Atelier bioinformatique

Hierarchical clustering
Module 3
GUIHARD Fabien



Motivation: In this project, we want to compare a gene and its associated proteins across multiple species. This will happen in a series of steps. I got as far as step 22 concluding the alignment matrix part.

1) Compare sequences

The first class **Sequence** is pretty concise, we have implemented the quickest way to compare two sequences using their proportion of pairwise different elements.

2) Hierarchical clustering

The beginning of the class **ClusterOfSequences** is still pretty straightforward with the three different constructors. I've added code comments so it is easy to understand.

The first interesting thing in this class is the getNewickIntermediate() function. It returns a string like `“(ATTACG,ATATCG,ACCCCG,GCCGAG,TCCCCG);”` before clustering.

```
private String getNewickIntermediate() {
    String res = "";
    if(elements.size()==1) res += elements.get(0);
    else {
        int index = 0;
        res += "(";
        while(subClusters.size() > index) {
            if(subClusters.get(index).elements.size() >= 1) {
                res += subClusters.get(index).getNewickIntermediate();
            }
            if(subClusters.size()-1 > index) res += ",";
            index++;
        }
        res += ")";
    }
    return res;
}
```

We recursively print all the elements of the ‘subClusters’ list until we arrive at the end.

Then, we have computed the distance between two clusters with the function linkage(). We use a helper function that calculates the average of the elements in a list. For now, we only use the crude distance.

```
//the average of the distance for each combination of sequence from each cluster
public double linkage(ClusterOfSequences anotherCluster) {
    ArrayList<Double> distList = new ArrayList<Double>();
    for(Sequence i : elements) {
        for(Sequence j : anotherCluster.elements) {
            distList.add(i.distance(j));
        }
    }
    return average(distList);
}
```

And last but not least, we now use this linkage function to **perform clustering** using the agglomerative method.
The overall idea of this agglomerative clustering is to ensure nearby points end up in the same cluster. Agglomerative clustering is typically **bottom-up** as opposed to the divisive clustering as shown on the diagram above. We then start by having a set of simple clusters and for each point we create a cluster that only contains that particular point. Next, we look at the collection of clusters and we try to find a pair of clusters that is closest to each other using the linkage.
If cluster A and cluster B are closest, we merge them and create a new cluster C with all the points of cluster A and cluster B. What we do next is that we delete cluster A and cluster B from the initial collection of simple clusters and add the cluster C into this initial collection.
We repeat this process until there is only one cluster left.

If we take a look at the complexity of this clustering algorithm, we know that each time we need to find the nearest pair of clusters, which takes N^2 steps. The overall complexity of it is therefore $O(N^2)$. N being the total number of points we have.

Here is an example of instructions for solving the clustering problem following my algorithm approach.

Set of simple clusters:

- cluster 1 = ATTACG
- cluster2 = ATATCG
- cluster 3 = ACCCCG
- cluster4 = GCCGAG
- cluster 5 = TCCCCG

→ cluster1 and cluster2 are closest so we merge them into a new cluster:

cluster 12 = ATTACG, ATATCG

We now remove cluster1 and cluster2 from the set of simple clusters and add cluster 12:

- cluster 3 = ACCCCG
- cluster 5 = TCCCCG
- cluster4 = GCCGAG
- cluster 12

→ cluster 3 and cluster 5 are closest so we merge them into a new cluster:

cluster 35 = ACCCCG, TCCCCG

We now remove cluster 3 and cluster 5 from the set of simple clusters and add cluster 35:

- cluster 4 = GCCGAG
- cluster 35
- cluster 12

→ cluster 4 and cluster 35 are closest so we merge them into a new cluster:

cluster 354 = ((ACCCCG, TCCCCG), GCCGAG)

We now remove cluster 4 and cluster 35 from the set of simple clusters and add cluster 354:

- cluster 354
- cluster 12

→ cluster 354 and cluster 12 are closest so we merge them into a new cluster:

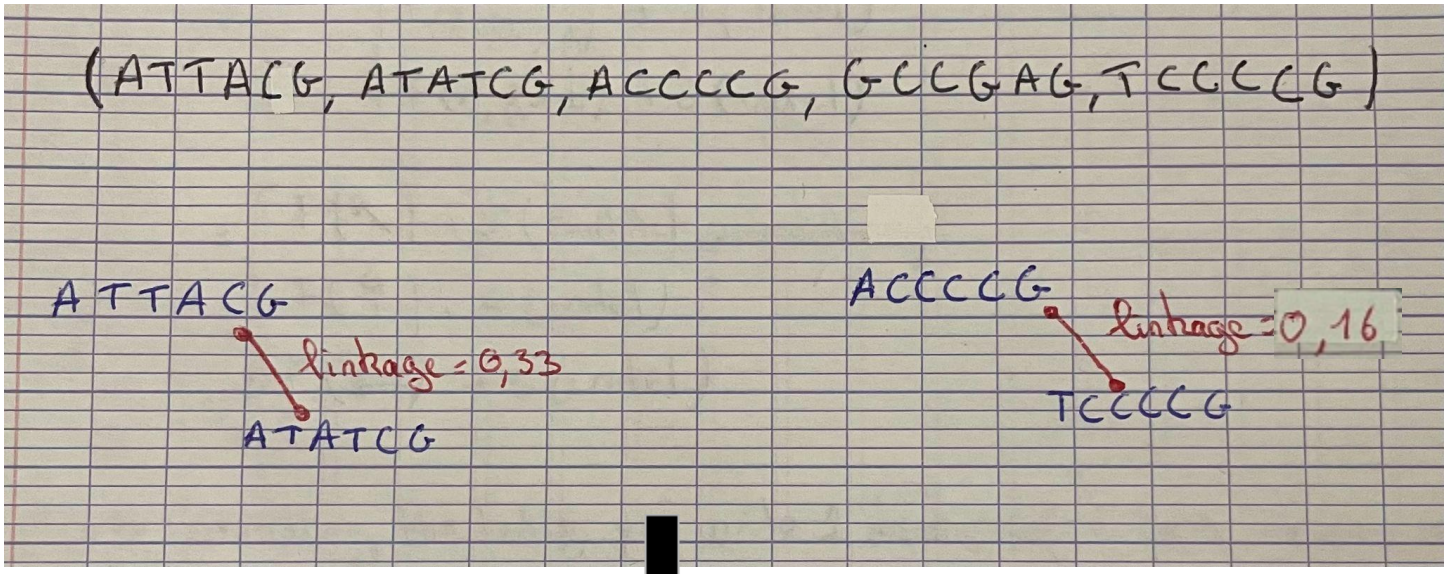
clusterFinal = ((ATTACG, ATATCG), ((ACCCCG, TCCCCG), GCCGAG))

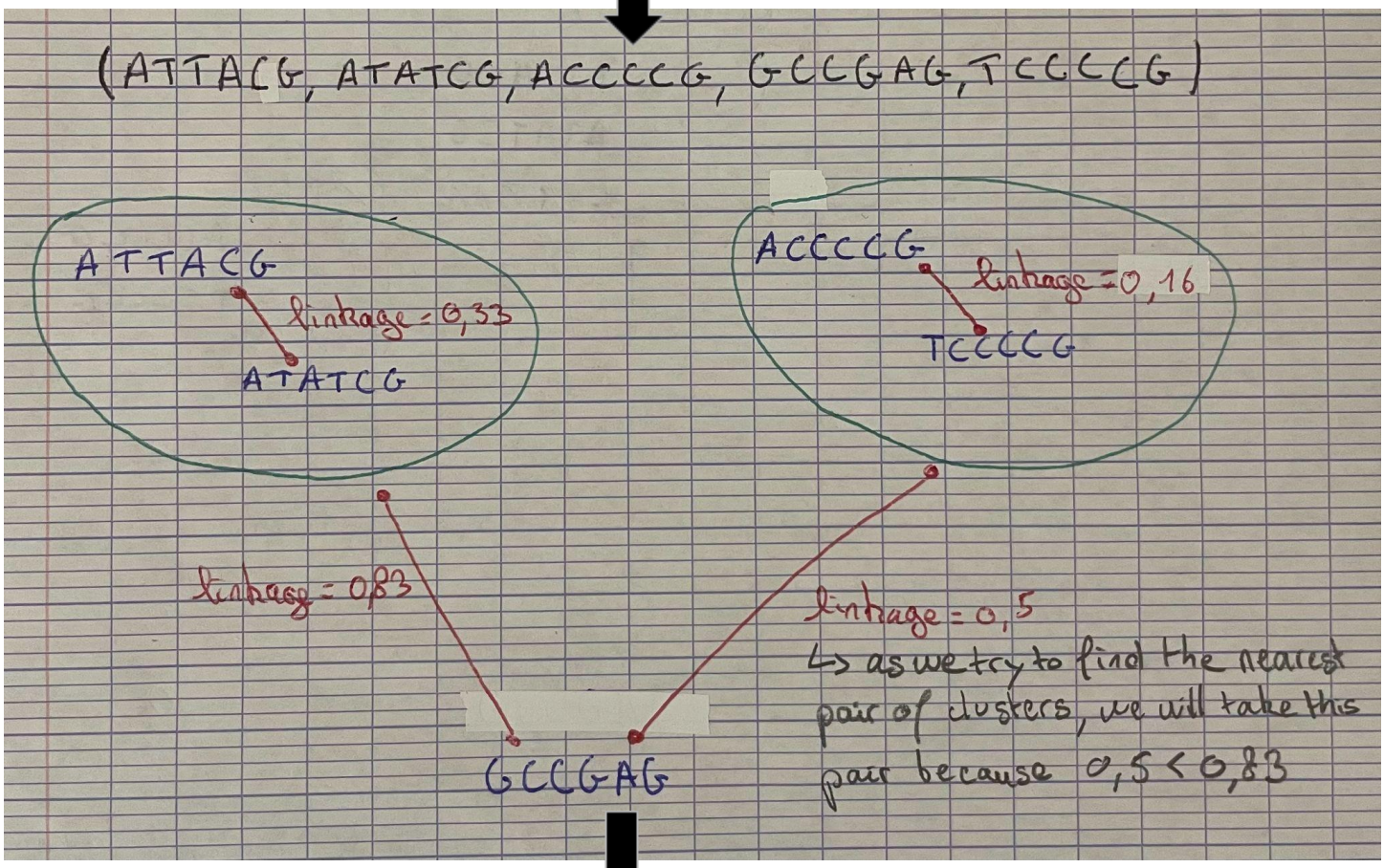
We now remove cluster 354 and cluster 12 from the set of simple clusters and add clusterFinal:

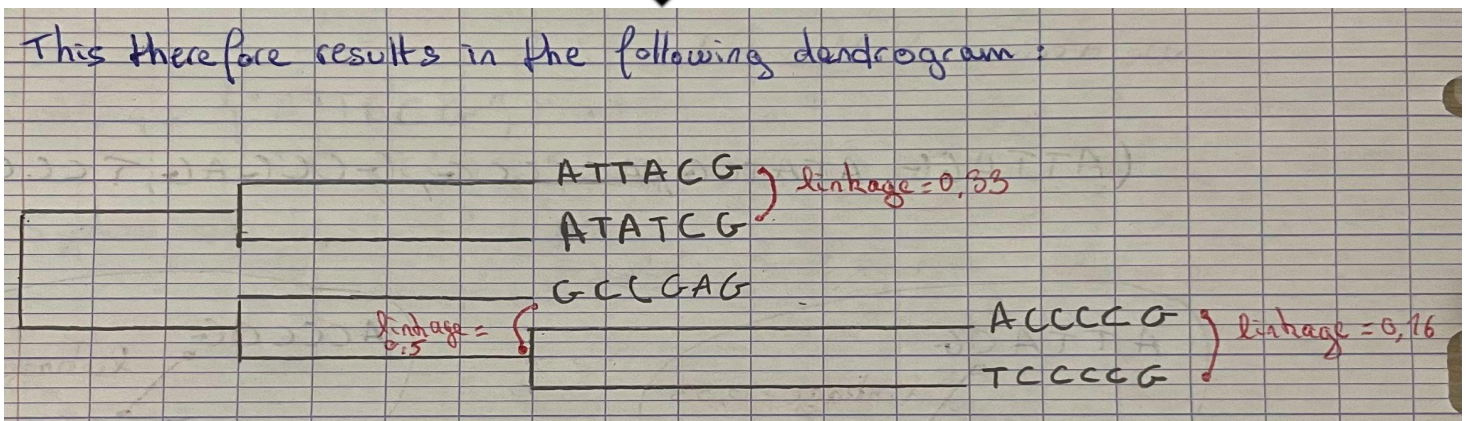
- clusterFinal

=> There is only one cluster left so it is therefore the final result of the clustering!

Here are some deeper explanatory diagrams in order to illustrate the whole process with the same clustering example :







3) Application to hemoglobin

I get the same dendrograms for DNA sequences and Protein sequences as those given in example (Figure 5 and 6).

4) Optional extensions

I've created the class AlignmentNW in order to use the global alignment strategy. My algorithm approach follows the example illustrated in the following video: https://www.youtube.com/watch?v=BYdTqq8AGgc

credit image : https://harshsharma1091996.medium.com/hierarchical-clustering-996745fe65