Ákos Lévárdy
ID: 121314

Assignment 3

Clustering – Problem C

Agglomerative clustering

Ákos Lévárdy
ID: 121314

Assignment:

The task is to program a clustering algorithm for a 2D space that analyzes a 2D space with all its points and divides this space into **k** clusters.
Different versions of the clustering algorithm, with the following algorithms:

- agglomerative clustering, where the center is the **centroid.**
- agglomerative clustering, where the center is a **medoid (5000 points are enough).**

  - The **centroid** of a cluster in a 2D space is the mean (average) of the coordinates of all points in that cluster.
  - The **medoid** of a cluster is the data point that minimizes the sum of the distances between itself and all other points in the cluster.

We have a 2D space that has X and Y dimensions, ranging from **-5000** to **+5000**.
Fill this 2D space with **20** points, each point having a randomly selected position using X and Y coordinates.
Each point has unique coordinates (no duplicates).

After generating 20 random points, generate another **20 000** points, but these points will not be generated completely randomly, but in the following way:

1. Randomly select one of all points created so far in 2D space. (not only from the first 20)

If the point is too close to the edge, then reduce the corresponding interval, given in the next two steps.

2. Generate a random number **X_offset** in the interval from **-100** to **+100**
3. Generate a random number **Y_offset** in the interval from **-100** to **+100**
4. Add a new point in 2D space that will have the same coordinates as the randomly selected point in step 1, offset by **X_offset** and **Y_offset**

No cluster can have an average distance of points from the center more than **500**.

Agglomerative clustering is a type of **hierarchical clustering**. It goes **bottom up**. Idea in agglomerative clustering: **ensure nearby points end up in the same cluster**.

- Start with a collection **C** of **n singleton clusters** (a cluster that contains just that individual).
  - Each cluster contains one data point: $\boldsymbol{c_i} = \{\boldsymbol{x_i}\}$
- Repeat until only **one cluster** is left (takes **n** steps, where **n** is the number of data points):
  - Find a **pair** of clusters that is **closest** (minimum distance, distance metric over clusters): $\min \boldsymbol{D}(\boldsymbol{c_i}, \boldsymbol{c_j})$
  - **Merge** the clusters $\boldsymbol{c_i}, \boldsymbol{c_j}$ into a new cluster $\boldsymbol{c_{i+j}}$ (take all the elements from both clusters and put them into a larger cluster)
  - **Remove** $\boldsymbol{c_i}, \boldsymbol{c_j}$ from the collection **C**, add $\boldsymbol{c_{i+j}}$
- Produces a **dendrogram**: hierarchical tree of clusters.
  - at the top will be all data in 1 cluster
  - at the bottom will be every data point at a separate leaf cluster (singleton)
- Slow: $\boldsymbol{O}(\boldsymbol{n^2 d} + \boldsymbol{n^3})$ create $\boldsymbol{n^2 d}$ -> to build an initial similarity matrix and initial distance matrix from every individual to every other individual in D dimensions

  $\boldsymbol{n^3}$ -> the algorithm itself takes a cubic number of operations to pursuit

Ákos Lévárdy
ID: 121314

The dissimilarity **between two clusters** is based on the **Euclidean distance** between their **centroids/medoids**.

Euclidian distance: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Cluster distance measures:

- Single link (min D)
- Complete link (max D)
- Average link (average of all pairwise D)
- **Centroids** (distance between **centroids** – **means** of two clusters)
- Ward's method

Algorithms:

Calculation of center_points:

Centroid

The coordinates of the centroid are calculated as the average of the coordinates of the points belonging to the given cluster.
This is an imaginary point.

Medoid

The medoid, unlike the centroid, is not imaginary, and is one of the cluster points. It is a point which's average distance from all other cluster points is the smallest.

The clustering algorithm:

1. Find out which two clusters are closest to each other, from the distance matrix.
2. Connect the points from cluster A with the points from cluster B.
3. Calculate a new center point for cluster B. The goal of the algorithm is to cluster clusters until no cluster has an average distance of points from the center greater than 500. Once any of the clusters exceed this boundary, we split back the cluster that violated this condition, and the algorithm will finish.
4. Cluster A is no longer needed, it will be deleted from the list of clusters, and at the same time its corresponding row is deleted from the matrix.
5. Go through all the rows of the matrix and recalculate the distances of the updated cluster B from other clusters, and a column is deleted in the matrix, which belonged to the deleted cluster A.
6. We calculate the average distance of the points from the center in each cluster, and if there is one, which exceeded it, we return the last cluster connection and print the result (number of clusters). Otherwise, we return to step 1.

Ákos Lévárdy
ID: 121314

Values:

Clusters are in a list of all_clusters, every cluster is represented as one instance of the Cluster class which contains its center_point and all the other coordinates.

```
class Cluster:
    new *
    def __init__(self, coors: [(int,int)], center_point: (int, int)):
        self.coors = coors
        self.center_point = center_point
```

The distances between clusters are kept up to date in the **distance matrix** which is simply a 2-dimensional array. Individual indexes of rows and columns are directly mapped to the list of clusters.

```
def create_matrix(clusters):
    dis_matrix = []
    n  = len(clusters)

    for i in range(n):
        row = []
        for j in range(n):
            dis = dis_between_clusters(clusters[i], clusters[j])
            row.append(dis)
        dis_matrix.append(row)
    return dis_matrix
```

Ákos Lévárdy
ID: 121314

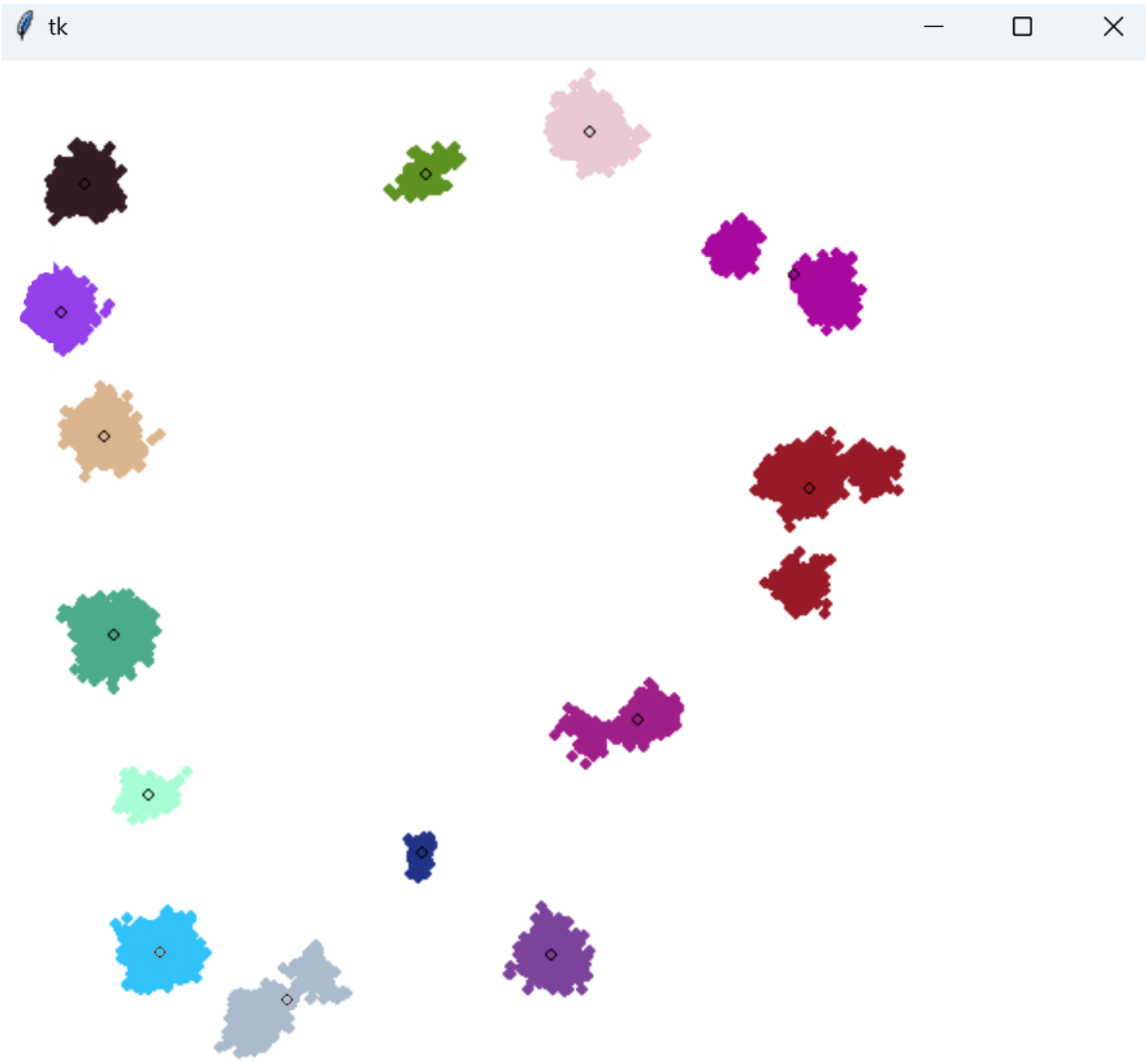
Visualization - Testing:

For testing the clustering algorithm PyPy implementation of Python was used. This helped to speed up the process a lot.

Centroid – 20_000 points:

```
Clustering time for 20000 points:
~ center is centroid:
~ 4414.65 s
~ 73.58 m
~ 1.23 h
```

Num of clusters: 14



Ákos Lévárdy
ID: 121314

Medoid – 5000 points:

```
Clustering time for 5000 points:
~ center is medoid:
~ 81.35 s
~ 1.36 m
~ 0.02 h
```

Num of clusters: 13