**AI project 3**

**b) K-means and Divisive clustering**

Márk Bartalos

Seminars with**: Ing. Martin Komák, PhD.**Seminars on**: Monday 9:00-10:50**

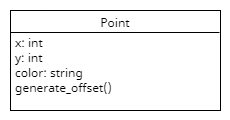
# Introduction

As task we got the implementation of K-means and Divisive clustering algorithms. For K-means centroid and medoid type clustering had to be implemented and for Divisive clustering the centroid type.

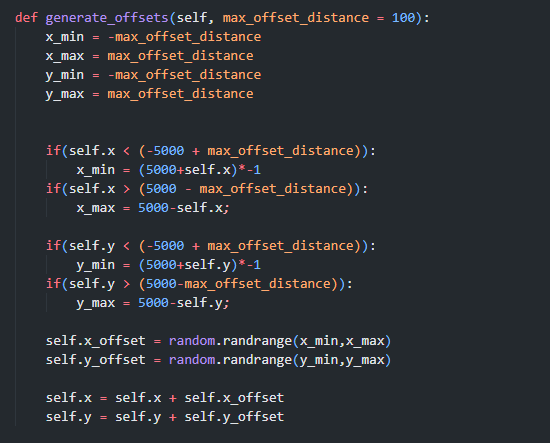
# Used data structures

## Point

The only data structure used was a point which has an X, Y position and a color.

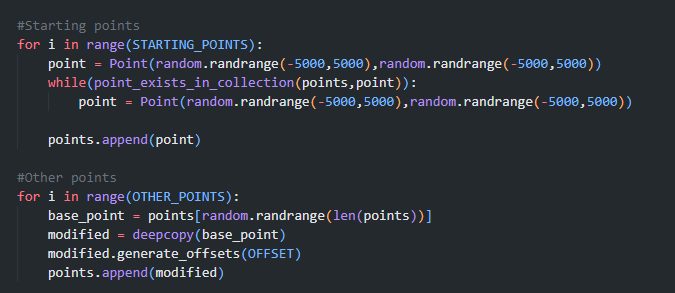


It also has a method called generate offset which is used to generate offset of the position from the original position which is inside the bounds of the diagram.



# Base initialization

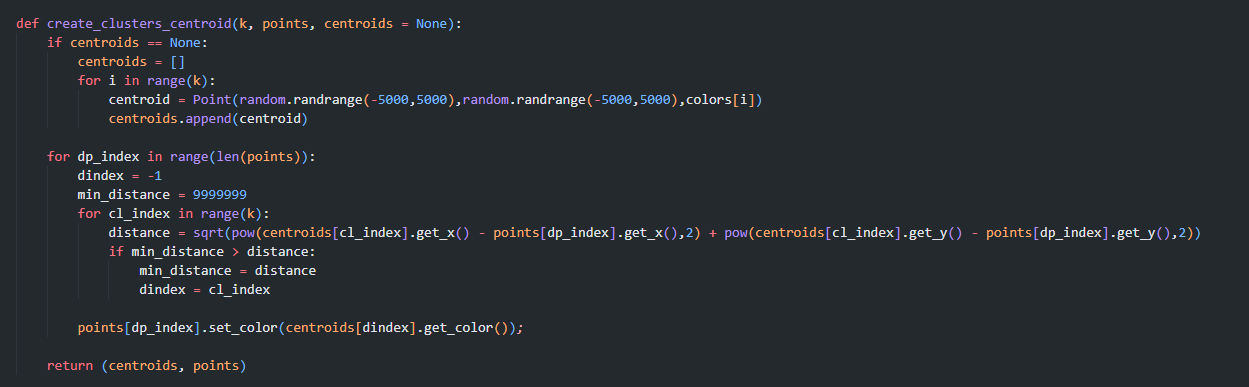
As initialization I generated n (20) amounts of points, which do not overlap and then generated 4000 more points near the previously generated points. These base points will be used during the algorithm.



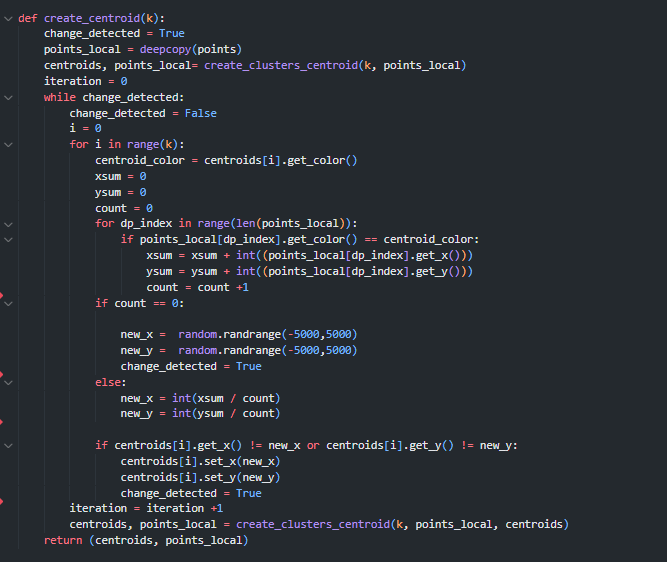
# K-means centroid

First, I generated K points in random positions to each I assigned a color, these will be the centroids. Then to each point I assigned a centroid which is closest to them by assigning its color. Then I went through each centroid and each point in each cluster and calculated the new center point for the cluster based on the average distance from each point. If it was done for all clusters and there was a change, we go back to the recolorization step, where we set the color of the points based on where the new centroids are and we do it all over until all centroids are constant and cannot be adjusted further.

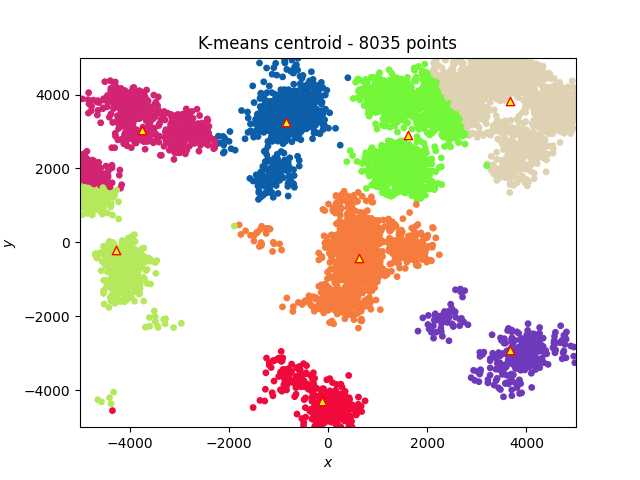
First, we create the clusters randomly.



Then we adjust the centroids iteration by iteration. If a centroid doesn’t have assigned points, we simply regenerate it randomly.



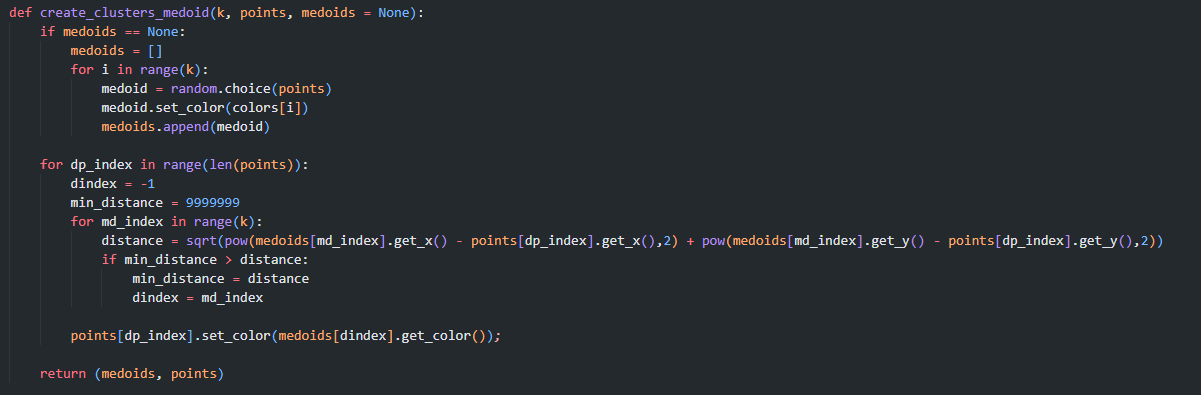
## Graph – 8035 points



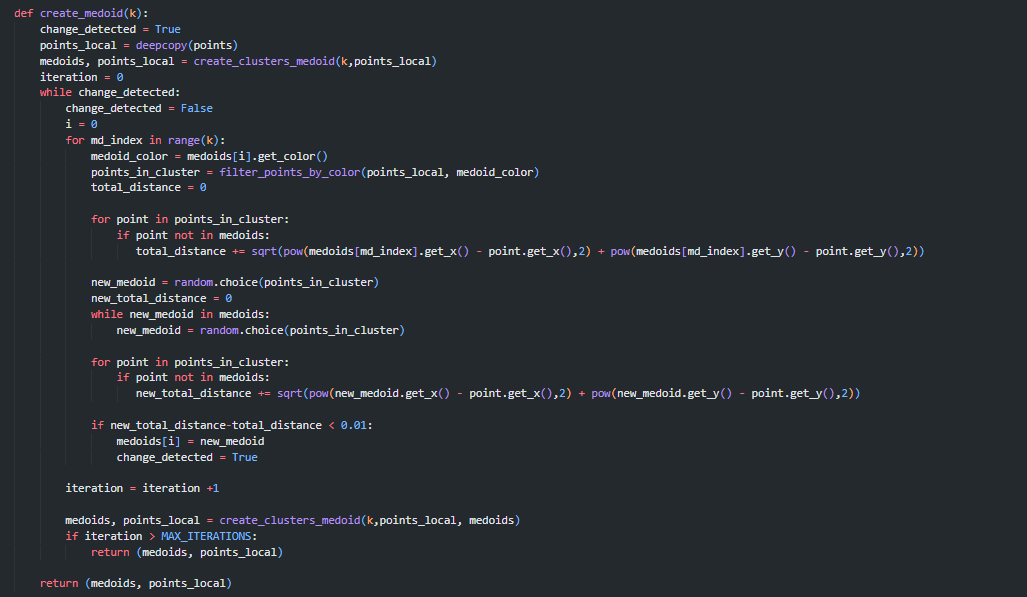
# K-means medoid

Here I first selected K already existing points, assigned each color, these will be the medoids. Then I assigned each point to a medoid which was closest to them. Then I went through each medoid and calculated the total distance between the medoid and all the points in the cluster then select a new random point from the cluster (which will be the new medoid) and calculate the total distance of all the points in the cluster from the new medoid. If the new total distance is smaller replace the previous medoid with the new one, do this to all medoids and re do from the recolorization step until there are no better points found or the iteration limit is not reached.

First, we create the medoid clusters.



Then we calculate the distances in the medoids and try to find new better medoids.

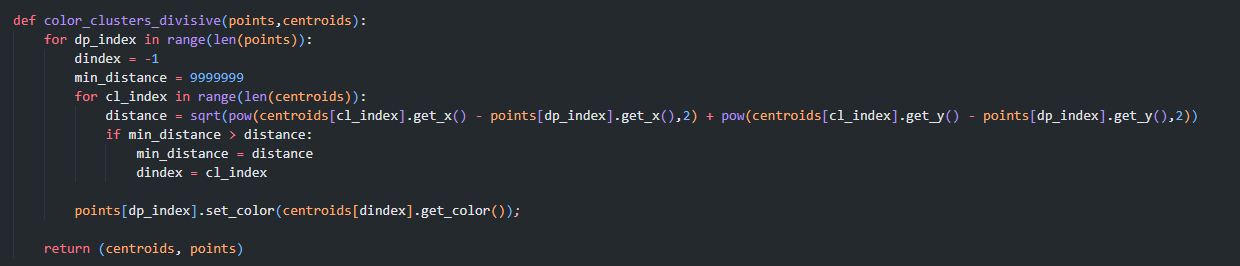


## Graph – 8035 points, 5000 max iterations

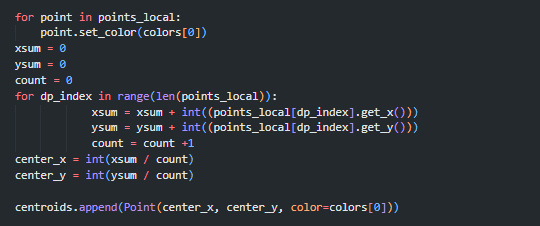
# Divisive centroid clustering

Here we start with a centroid of all points, then we split these points to two clusters, to each cluster we assign a centroid based on its points, we recolor the points based on their associated clusters and we continue splitting the clusters until we have the desired k amount.

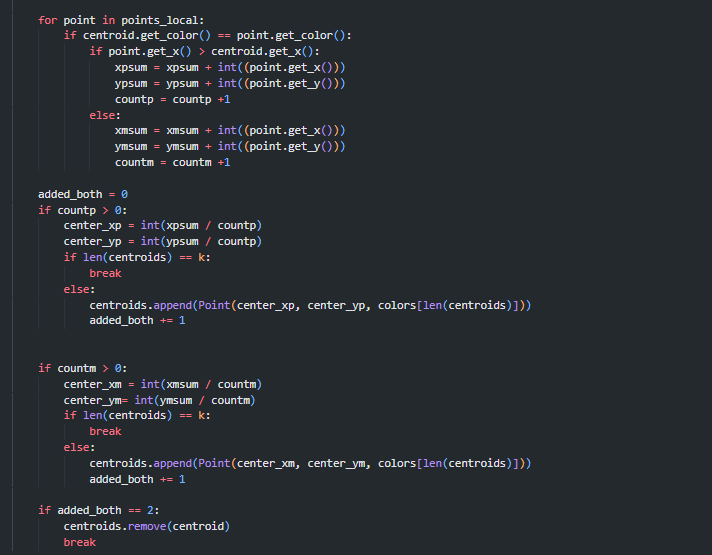
Colorization of the clusters



Calculate the first centroid (of all points)



Split a cluster into two based on the x axis:



## Graph