Lab 3: Sampling spatial data using Quadtrees

Auke-Dirk Pietersma

Deadline: 10 December at 12:00h

1 Problem definition

GIS databases, typically, consists of thousands of geometries. While it it can fetch such geometries efficiently, sending them to the client often leads to bandwidth problems. This is also due to more and more queries being performed through mobile devices. One approach to address this challenge, and on which we will focus during this lab, is to sub sample the result of the query. Our goal is to improve on naive sampling (first n elements) as seen in Fig. 2, such that it better resembles our total set, Fig. 1.

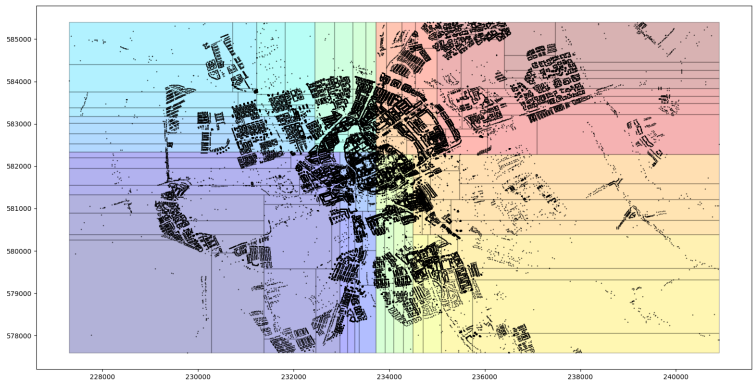


Figure 1: Illustration of an entire data set.

2 Lab

The student must improve the provided code such that sampling through Quadtrees is possible. In the final program there should be two variables controlling the final sampling:

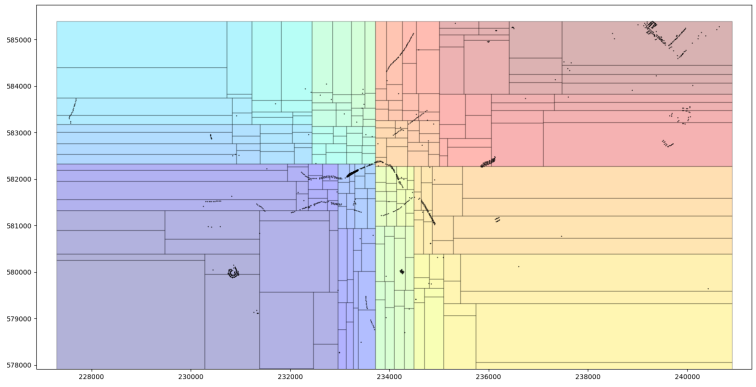


Figure 2: Naive sampling selecting the first N items

- (Quadtree) quadtree : add a QuadTree with certain depth
- (Plotter) quadlevel: at which level the points should be sampled.

In order to achieve the above the student will first need to implement the following functions:

KDTree	def closest(self, point, sidx = si.StorageIndex())
QuadTree	def recurse(self,bbox, depth)

3 The role of each class

3.1 Documentation

The classes Database, QuadTree, KDTree, and BoundingBox come with full documentation and test cases. The documentation can be viewed in the code, or with ipython. All classes contain a main entry point such that they can can be tested/viewed independently.

```
python kdtree.py
{'index': 1, 'depth': 0, 'partition': 5, 'axis': 0}
{'index': 2, 'depth': 1, 'partition': 4, 'axis': 1}
{'index': 3, 'depth': 1, 'partition': 2, 'axis': 1}
{'index': 4, 'depth': 2, 'elements': array([1, 2]), 'axis': 0}
..
```

4 Tasks to be implemented by students

4.1 Read all documentation

Before starting to implement the missing code please take some time to read the respective documentation of the classes: Database, QuadTree, KDTree, BoundingBox.

4.2 Implementing the QuadTree [2.5 pts]

The QuadTree is a recursive data structure, where in each new iteration, the current BoundingBox needs to be split into four equally sized BoundingBoxes. These newly formed boxes should fill the exact space as their predecessor.

The following code can be used to test your QuadTree:

```
python3 plot_kdtree.py --quadtree 4 --quadshow True
```

With the above statement, you should see an 8×8 chessboard pattern.

4.3 Implementing the KDTree [2.5 pts]

The KDTree is used to partition the data points into several bounding boxes, after which the tree can be queried to obtain a collection of such data points. The range-query, which finds all bounding boxes/Leaf-nodes that intersect with a given BoundingBox and return their data point collection, is already implemented and given to the student. The closest function , however, is left for the student. The closest function must traverse through the tree similar to that of the rquery instead it operates on a single point (x, y), and can thus only traverse left or right. After reaching the leafnode, closest should return all the data points within the leafnode.

4.4 Using the QuadTree depth to sub sample the KDTree [2.5 pts]

In the final section we will need to add/update the field 'quad' for every record inside our database. This field represents whether or not a record should be sampled. For example: if we wish to sample all records up to quad-level 4, then we simply check whether or not a record has 'quad' field is lower or equal to 4 (The higher the level the more the data).

The student needs to implement the latter in 'plot_kdtree.py' at line 81:

#To be implemented by the student

The following steps are needed:

- initially set the quad field for all records to the max quad-level

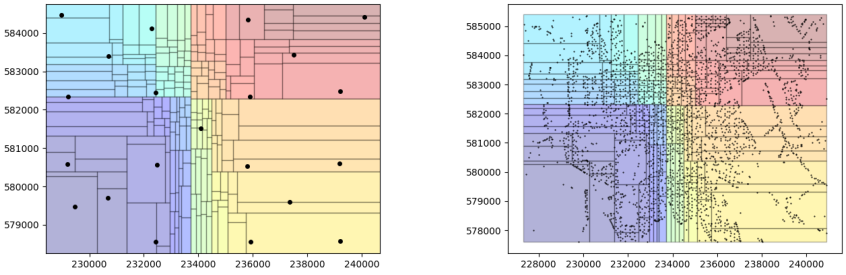
- iterate through all levels of the quadtree in reverse order.
- for every box inside that level obtain its centroid.
- use the centroid to obtain the list of the closest points inside the KDTree
- find the closest point inside that list
- update the ‘quad‘ field corresponding to that point to the current quad-level

4.5 Final result [1.5 pts]

Execute the final program with different values of ‘quadtree‘ and ‘quadlevel‘:

```
python3 plot_kdtree.py
        --filename ../data/addressen_groningen/adressen_groningen.shp
        --bbox-depth 8
        --max-depth 9
        --plot kdtree-bb
        --quadtree 8
        --quadlevel 7
```

should result in sampling as in Fig. 3a and Fig. 3b.



(a) (b)

Figure 3: Sampling using a quad tree. (a) Depth 4 and level 3, and (b) depth 8 and level 7.

5 Deliverable

Please submit a zip file containing the implemented code and a report where each step is clearly explained. The clarity of the report contributes to **1 point**.