COMP7106 Big Data Management
Assignment 2 – Spatial Data

Xin Hong – 3036031914

1. Project Structure

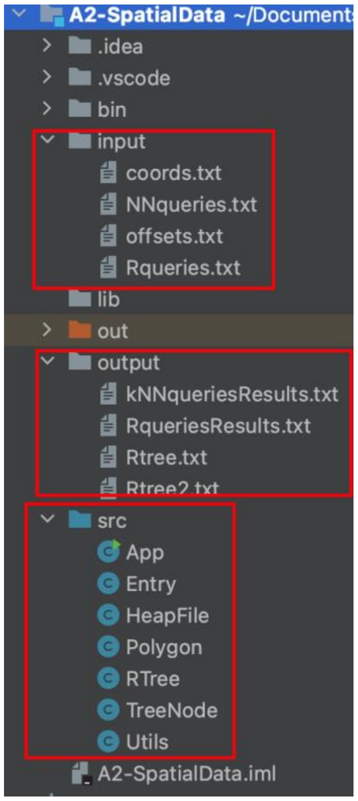


Figure 1.1 project structure

"/src" is the folder that contains the source code files.
"/input" is the folder where you can put the input files: coords.txt, offsets.txt, Rqueries.txt, and NNqueries.txt.
"/output" is the folder containing the output of the program: Rtree.txt.

2. Run the Program

To run 3 different parts of the program using command-line arguments, I have saved three configurations (Figure 2.1) for the program running in ***IntelliJ IDEA***.

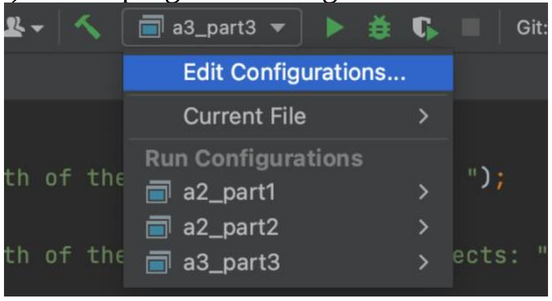
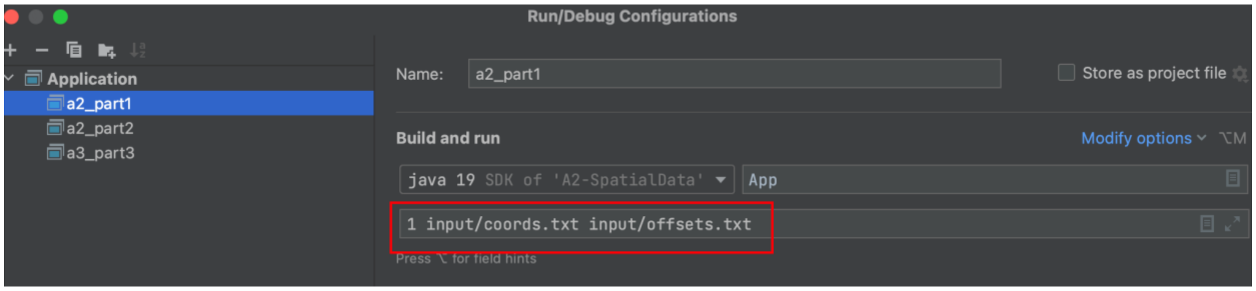


Figure 2.1

a. Part 1



Name: a2_part1

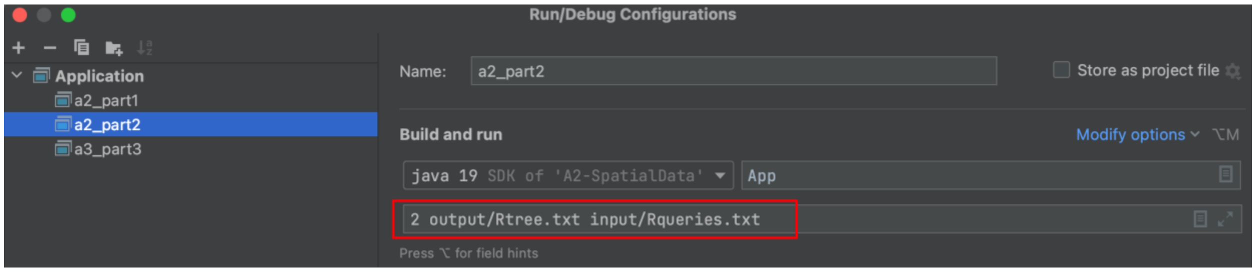
java 19 SDK of 'A2-SpatialData' App

```
1 input/coords.txt input/offsets.txt
```

Figure 2.2 part 1 arguments

The argument to run the part 1 program is "1 input/coords.txt input/offsets.txt". The first argument indicates which part of the program you want to run. The second argument is the file path of the coordinates of points and the third argument is the file path of the offsets of the records.

b. Part 2



Name: a2_part2

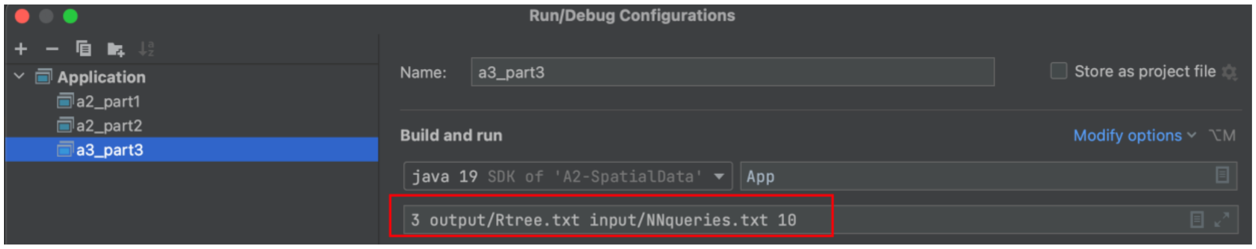
java 19 SDK of 'A2-SpatialData' App

```
2 output/Rtree.txt input/Rqueries.txt
```

Figure 2.3 part 2 arguments

The argument to run the part 2 program is "2 output/Rtree.txt input/Rqueries.txt". The first argument indicates which part of the program you want to run. The second argument is the file path of the Rtree file created in part 1, and the third argument is the file path of the range queries.

c. Part 3



Name: a3_part3

java 19 SDK of 'A2-SpatialData' App

```
3 output/Rtree.txt input/NNqueries.txt 10
```

Figure 2.4 part 3 arguments

The argument to run the part 1 program is "3 output/Rtree.txt input/NNqueries.txt 10". The first argument indicates which part of the program you want to run. The second argument is the file path of the Rtree file created in part 1. The third argument is the file path of the kNN queries, and the fourth argument is the parameter k for kNN search.

3. Core Methods of the Classes

a. Polygon.java

Used to represent the polygon objects.

Methods	Description
`public void addPoint(Double x, Double y)`	Incrementally add a point to the polygon, and update the MBR of the polygon object.
`public void calZOrderCode()`	Calculate the z-order code for the center point of the polygon object.

b. HeapFile.java

A data structures to store all the polygons.

Methods	Description
`public void readFromFiles(String coordsFileName, String offsetFileName)`	Construct all the polygon objects by reading data from coordinate points and offsets files, and store them in a list which are sorted by their z-order code.

c. Entry.java

R-tree' entry, where contains MBR and node-id or object-id.

Methods	Description
`public boolean isOverlap(ArrayList<Double> window)`	Determine whether the range query area is overlap with the MBR of the entry.
`public double distance(double x, double y)`	Calculate the distance between the point (x,y) and the entry's MBR.

d. TreeNode.java

Tree node of the R-tree, where contains entries.

Methods	Description
`private void updateMBR(Entry entry)`	When appending an entry or prepending an entry to the node, MBR is needed to update.
`private void recalMBR()`	When deleting an entry of the node, MBR is required to recalculate.

e. Rtree.java

R-tree.

Methods	Description
`public void bulkLoading(HeapFile disk)`	Bulk-Loading of the R-tree.
`public void writeTree(String fileName)`	Write the R-tree to a file.
`public void readTreeFromFile(String fileName)`	Read an R-tree from the R-tree structure file.

`public void rangeQuery(String fileName, String type)`	Read range queries from file and do range search using the R-tree. Type indicate which version of implementation you want to use: “r” is the recursive version and “i” is the iterative version.
`public void kNNQueries(String fileName, int k)`	Read kNN queries from file and do kNN search using incremental best-first search.