* **Program execution requirements**

**1.1** **Program environment**

The program was developed using Python programming language. The Python version 3.7.4 (64-bit) was used for this purpose. The program was executed and tested on a Window 10 (64-bit) HP (Hewllet Packard) laptop with Intel i5 core processor and 4GB RAM

**1.2** **Input files and parameters**

The program takes two input files; the dataset file and the query file as command line argument respectively. After complete execution, the program generates two output files; “SequentialSearchResult.txt” and “RTreeResult.txt”. These files hold the data of number of points retrieved by each query for both sequential and Rtree program respectively. Please refer the screen shot attached below for the command syntax.

**1.3** **Other requirements**

The program uses relative path for both dataset and query files. This makes it necessary to place both the files in the same directory as the “BigdataAlgorithms.py” python file.

1. **Program documentation**

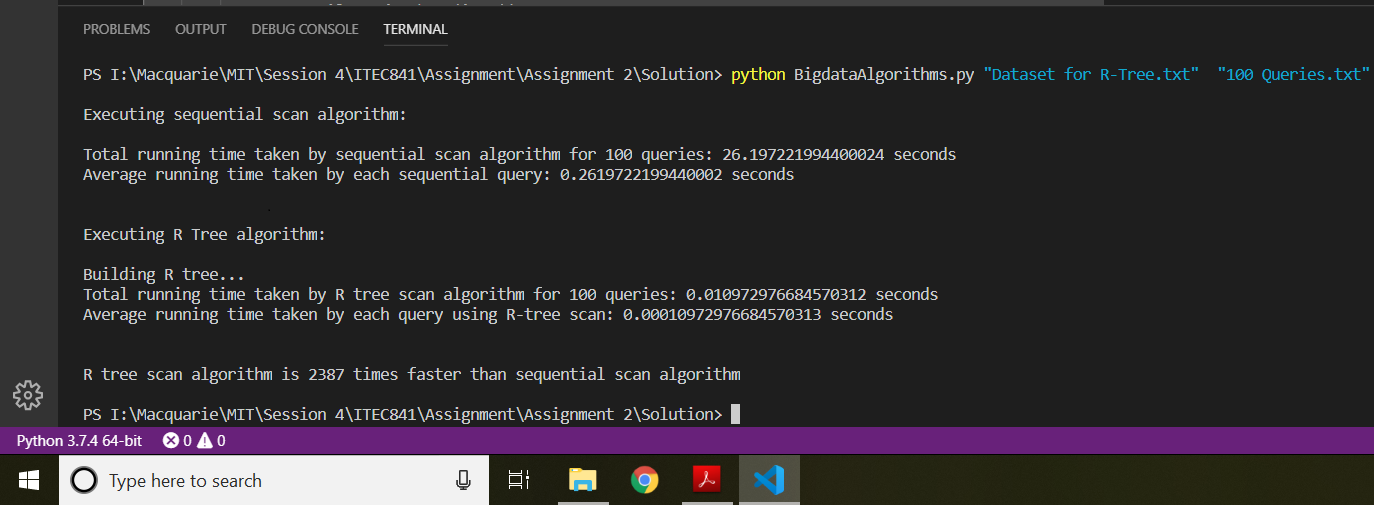
**2.1 Program organisation**

|  |  |
| --- | --- |
| **Class/File Name** | **Description** |
|  |  |
| BigdataAlgorithms.py | This file holds a total of 3 functions. When executed, the main function is called which in turn calls the functions implementing of both Sequential scan and Rtree scan algorithms. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

* 1. **Function description**

|  |  |
| --- | --- |
| **Function Name (parameters)** | **Description** |
|  |  |
| main() | The main function retrieves the dataset filename and query filename provided by the user as command line argument. It then reads the data stored in both the files and loads them into dataset and queryDataPoints lists respectively. It then calls the SequentialSearch and RTreeImplentation and passes both dataset and queryDataPoint as arguments. Finally, it computes the efficiency of Rtree algorithm with respect to Sequential algorithm and displays it on the output screen. |
| SequentialSearch(dataset, queryDataPoints) | This function sequentially scans the dataset for each query in the queryDataPoints list. For every query, it maintains a counter which is incremented for each point that satisfies the given query. A point is said to have met a query if and only if the both the point’s x and y coordinate lie between the query’s xMax and xMin and also yMax and yMin respectively. After executing the algorithm, it displays the total and average running time taken by each query. For this purpose, the program uses built-in time module’s now() function provided by Python. It finally generates an output file “SequentialSearchResult.txt” that contains the number of points retrieved for each query. The function also returns the average running time of each query to the main function. |
| RTreeImplentation(dataset, queryDataPoints) | This function uses the Rtree library (version 0.8.3) that was downloaded from <https://pypi.org/project/Rtree/> . It provides two functions insert and intersection that are used to build and query the Rtree respectively. The insert function creates a bounding box for every point in the given dataset. The insertion function also creates a bounding box and identifies all other bounding boxes it intersects. This significantly reduces the search space when identifying the points that satisfies the query. After executing the algorithm, it displays the total and average running time taken by each query. Similar to SequentialSearch function it uses the time module’s now() function for this purpose. It finally generates an output file “RTreeResult.txt” that contains the number of points retrieved for each query. The function also returns the average running time of each query to the main function. |

**Snapshot of the program output**

****