COMP3210/6210: Report for Assignment 2

1. Group Member Information

Name	Student ID	Email:	Assigned Task
Coordinator: The Vinh Ha	46741208	thevinh.ha@students.edu.au	Task 1
Quang Tri Ha	47382538	quangtri.ha@mq.students.edu.au	Task 3
Thuy Quynh (Kristen) Tran	46325395	thuyquynh.tran@students.mq.edu.au	Task 2

2. Program Execution Requirements

2.1 Program Environment (e.g., OS, database, CPU, etc.)

- Operating system: Windows 10/11, macOS 12
- Programming language: Python 3.8
- CPU requirements: Minimum requirement – Intel i5 or equivalent
- Libraries:
  - NumPy
  - R-Tree
  - SciPy

2.2 Input Files and Parameters (directory settings and other parameters)

- Directory settings:
  - Datasets should be stored in '/Downloads/' directory.
- Input files:
  - Datasets files: 'shop_dataset.txt', 'city_2.txt'
  - Source codes:
    - Task1: 'best_first.py', 'divide_and_conquer.py', 'node.py', 'rtree.py', 'sequential_scan.py'
    - Task2: 'bbs.py', 'dq.py', 'node.py', 'rtree.py', 'sequential_scan.py'
- Parameters:
  - Dataset filename
  - Source code
  - Algorithm selection

2.3 Additional Requirements

- Ensure Python 3.8 is installed to import the required libraries.
- Enough memory and space to handle large datasets.

3. Program Documentation

3.1 Program Organization

If your assignment consists of multiple files and/or classes, please provide brief, high-level descriptions of each file/class within your program, as illustrated below.

Class/File Name	Description (detailed information)

Task 1: ‘best_first.py’ ‘divide_conquer.py’ ‘sequential_scan.py’ ‘node.py’ ‘rtree.py’	- Implement best first search algorithm with the R-Tree data structure to find the nearest neighbor (NN) data point for each query point - Implement best first search algorithm combined with divide and conquer on the R-Tree data structure to find the NN data point for each query point - Implement sequential scan based method to find the NN data point for each query point - Define the Node class for R-Tree data structure - Define the RTree class
Task 2: ‘bbs.py’ ‘dq.py’ ‘sequential_scan.py' ‘node.py’ ‘rtree.py’	- Implement the BBS Algorithm to search the data based on Skyline Search. - Divide the dataset into subsets and search for the data based on Skyline Search. - Implement the sequential scan to find the nearest neighbor point for each query point. - Containing the node file to initiate the R-Tree library. - Construct the R-Tree to implement the Search Algorithm.

3.2 Function Description

Function Name (parameters)	**Description (detailed information)**
‘main’ (located in each function main function file in task 1 and task 2) ‘perimeter’ (self) ‘is_overflow’ (self)	- The main function initialized in the main files to run the nearest neighbor, BF Algorithm and BBS Algorithm. - Calculate half the perimeter of the MBR - Check if the node is overflowing in order to split the entries.

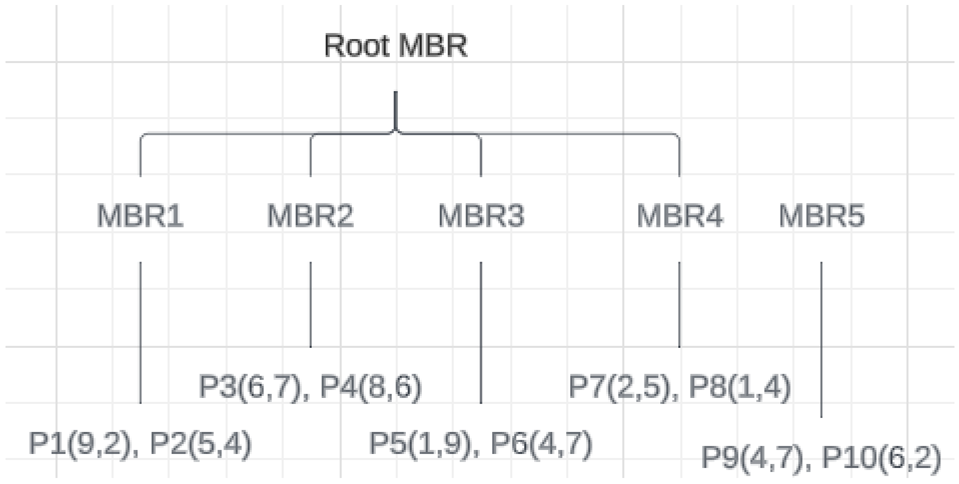
‘is_root’ (self)	- Check if the node is a root node.
‘is_leaf’ (self)	- Check if the node is a leaf node.
‘_init_’ (self, B):	- Initiate a root for the tree.
‘insert’ (self, u, p)	- Inserting a new point to the MBR.
‘choose_leaf’ (self, u, p)	- Find the position of the leaf node to insert the data point.
‘peri_increase’ (self, node, p)	- Calculate the increase of the perimeter after inserting the new data point.
‘handle_overflow’ (self, u)	- Algorithm to handle the overflow when inserting data points.
‘split’ (self, u)	- Split the data point into half.
‘add _child’ (self, node, child)	- Append the root with more MBR child nodes inserted to avoid the overflow.
‘add_data_point’ (self, node, data_point)	- Add a data point into the MBR and update them.
‘update_mbr’ (self, node)	- Update the MBR when splitting or forming a new MBR.
Task 1: ‘construct_RTree’ (input_file, B)	-Construct the R-Tree from the given dataset.
‘bfs (rtree, querypoint)	- Implement the Best First Search Algorithm to find the nearest neighbor using R-Tree.
‘euclidean_distance’ (p1, p2)	- Implement the Euclidean algorithm to find the distance between two points.
‘min_distance’ (query point, mbr)	- Find the minimum distance from a point to a MBR.
‘divide_dataset’ (input_file, dimension = ‘x’)	- Construct the divide and conquer algorithm to split the dataset into half.

‘sequential_scan_base’ (input_file, query_file, output_file)	- Implement the sequential scan to find the nearest neighbor.
Task 2: ‘read_dataset’ (filename)	- Read the downloaded dataset from a file.
‘build_rtree’ (points, B)	- Build an R-Tree from a list of points.
‘is_dominated’ (point, others)	- Check if a point is dominated by any other points for the Skyline Search.
‘bbs_algorithm’ (rtree, points)	- Construct the BBS Algorithm to identify the skyline using R-Tree.
‘divide_dataset’ (points, dimension)	- Divide the dataset into two subsets based on a selected dimension.
‘skyline_search’ (input_file, output_file)	- Implement the Skyline Search Algorithm to find the skyline of the given tree.

4. Analyzing BF Algorithm based NN Search

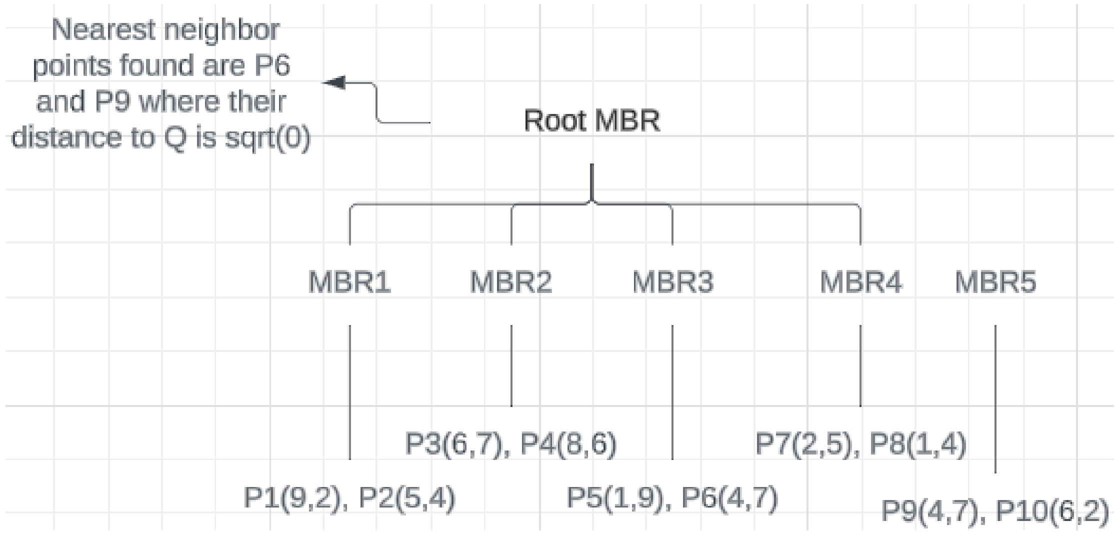
4.1 The Process of R-Tree Construction

- Data insertion: Input the dataset and insert each point into each node.
- For each node, we have to create an MBR that bounds with the defined points.
- Finally, transform the MBRs of each node into higher level nodes. Further initiate the MBR for the nodes until the root node is determined.



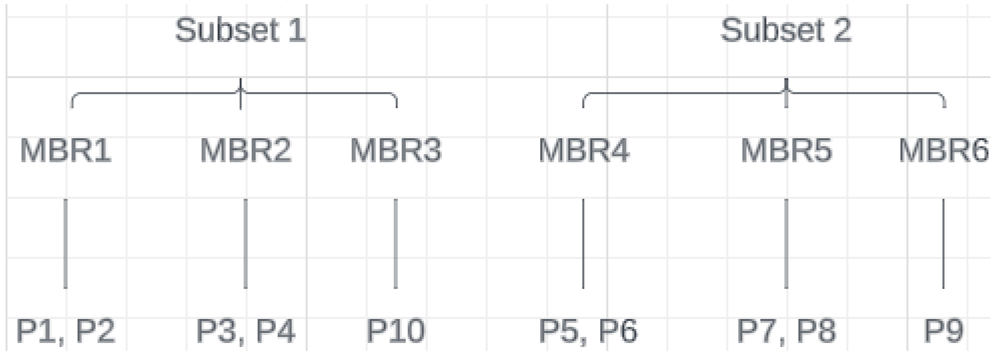
4.2 The Process of BF Algorithm

- To initiate the process of BF Algorithm, we start with the root MBR with the query point at Q(4,7).
- Continue traversing through the R-Tree and check each of the MBRs to calculate the distance between the query point and the traversing node.
- By calculating each point of the MBRs using the Euclidean distance, we can further determine the minimum distance which evokes the nearest neighbor points afterwards.

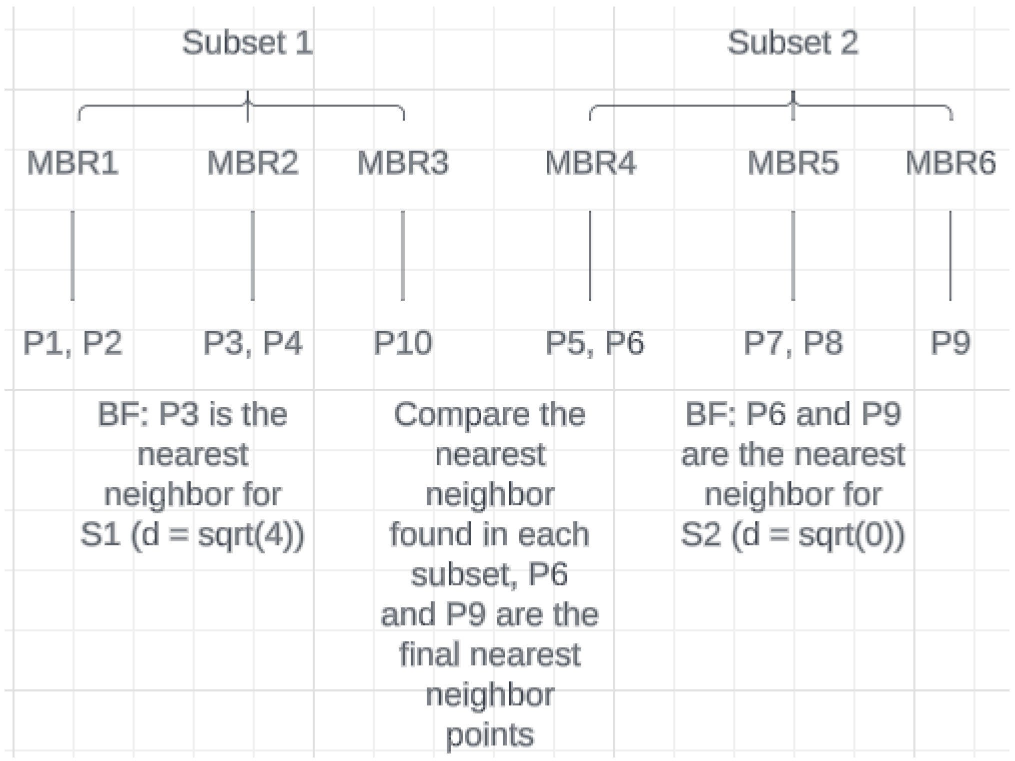


4.3 The Process of Divide-and-Conquer

- First of all, insert every point of the dataset and the query point. Then, we must split the dataset into half of two subsets followed by the median point.
- After constructing an R-tree, each point in the subsets is divided into different children based on the algorithm of R-tree.



- Conduct the BF Algorithm to find the nearest neighbor point between each subset to the selected query point. Afterwards, determine in between each found nearest neighbor point in two subsets to find the final nearest neighbor point.

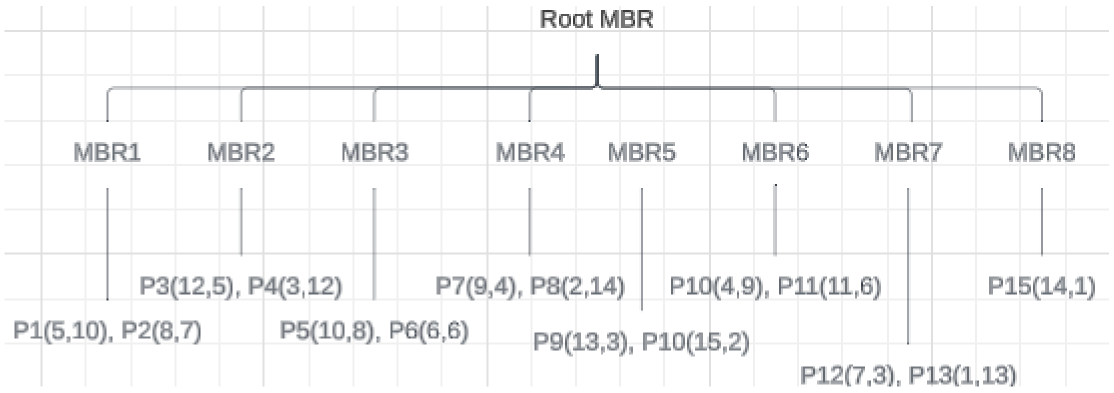


Subset 1
Subset 2
MBR1
MBR2
MBR3
MBR4
MBR5
MBR6
P1, P2
P3, P4
P10
P5, P6
P7, P8
P9
BF: P3 is the nearest neighbor for S1 (d = sqrt(4))
Compare the nearest neighbor found in each subset, P6 and P9 are the final nearest neighbor points
BF: P6 and P9 are the nearest neighbor for S2 (d = sqrt(0))

5. Analyzing the BBS Algorithm based Skyline Search

5.1 The Process of R-Tree Construction

- Data insertion: Input the dataset and insert each point into each node.
- For each node, we have to create an MBR that bounds with the defined points.
- Finally, transform the MBRs of each node into higher level nodes. Further initiate the MBR for the nodes until the root node is determined.

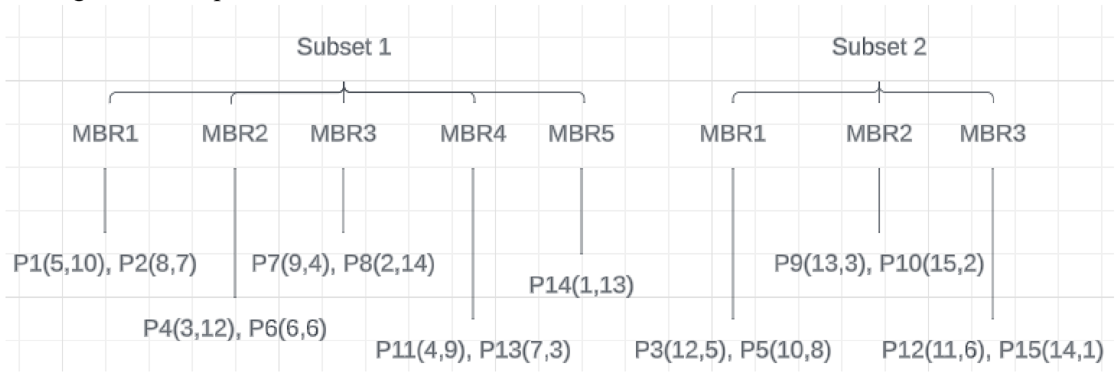


5.2 The Process of BBS Algorithm

- Initiate at the root of the MBRs and take that as a query point.
- Traverse through each point of the tree and determine if that point is part of the skyline. By comparing if x of Q (setter point) is smaller than x of P (comparing point) but Q(y) < P(y), we can tell that Q dominates P such that P is not applying to the skyline.
- By adopting the mechanism, these points below are part of the skyline:
  - P1(5,10), P2(8,7), P4(3,12), P8(2,14), P9(13,3), P15(14,1).

5.3 The Process of Divide-and-Conquer

- First and foremost, we have to halve the dataset into two subsets based on a dimension. In this sample, we will utilize the X-dimension of each point in the MBRs to be able to divide them such that we take the first subset of any point with the x value less than 10, and the second subset will contain the rest of the points with the x value greater or equal to 10.



- After splitting the dataset into each subset, we implement the R-Tree for both subsets such that later we can apply the BBS Algorithm to do Skyline Search.
    - Skyline points of subset 1: P1(5,10), P2(8,7), P4(3,12), P8(2,14), P13(7,3), P14(1,13)
    - Skyline points of subset 2: P3(12,5), P9(13,3)
- By implementing the BBS Algorithm based Skyline Search, we can determine the Skyline points in each of the subset. Thus, merge them together into 1 whole set of Skyline Search.
    - Merged Skyline Search Points: P1(5,10), P2(8,7), P4(3,12), P8(2,14), P13(7,3), P14(1,13), P9(13,3), P10(15,2)