COMP3210/6210 Report for Assignment 3

46168249_Chowdhury

1 Program execution requirements

1.1 Program environment

Language: C#
Framework: .NET Core.

How to run the program:

To run the program by compiling the source:
- Download .NET Core SDK (version 3.1.x) from dot.net/get-core
- Navigate to `src` folder, which contains the `RTree.csproj` file and the source files.
- Run the following command:

```
$ dotnet run [path-to-dataset-file] [path-to-query-file]
```

1.2 Input files and parameters

A file containing the dataset (points) and a file containing queries (rectangle coordinates) are needed. The coordinates should be separated by a space.

2. Program documentation

2.1 Program organisation

Class/File Name	Description
`Program`	The class that executes the program.
`Point`	The class that represents a 2D point. Defined in `Geometry.cs` file.
`Rectangle`	The class that represents a rectangle (or a query area). Defined in `Geometry.cs` file.
`Constants`	Static class that contains the constant B.
`Node`	The class contains the model of a node of the r-tree.
`RTree`	The class for building the r-tree. Class definition is divided in two files: `RtreeModel.cs` and `HelperFunctions.cs`

2.2 Function description

Function Name (parameters)	Description
Class: Node	
`int Perimeter()`	Calculates half perimeter of the node's MBR
`bool IsUnderflow()`	Checks if the node is underflowing.
`bool IsOverflow()`	Checks if the node is overflowing.

`bool IsRoot()`	Checks if the node is a root node.
`bool IsLeaf()`	Checks if the node is a leaf node.
Class: `RTree` (Instance methods in the `RtreeModel.cs` file)	
`bool Insert(Node node, Point point)`	Inserts a data point to a node.
`void HandleOverflow (Node node)`	Handles overflow of a node by splitting it into two.
`Node ChooseSubtree (Node node, Point point)`	Chooses a subtree, a child of the node, whose MBR needs the least amount of increased perimeter.
`int Query (Node node, Rectangle query)`	Queries a node with a rectangle and returns the number of data points that are inside the query area.
`(Node, Node) Split(Node node)`	Finds the best split of the node and returns a tuple of the nodes. Its parents are also split if they overflow.
Class: `RTree` (static methods in the `HelperFunctions.cs` file)	
`int IncreasePerim (Node node, Point point)`	Calculates the increase to perimeter of the node's MBR due to adding the point to the node's data points.
`bool IsCovered (Point point, Rectangle query)`	Checks if the point is covered by the query area.
`bool IsIntersect (Node node, Rectangle query)`	Checks if a node's MBR is intersect with the query area.
`void AddChild(Node node, Node child)`	Adds the child node to the node's children and modifies the node's MBR accordingly.
`void AddDataPoint (Node node, Point dataPoint)`	Adds the data point to the node's list of data points and modifies the node's MBR accordingly.
`void UpdateMBR (Node node)`	Updates the node's MBR by comparing the node's datapoints (if it is a leaf node) or its children's MBRs (otherwise) and taking the minimum of the lower bounds and the maximum of the upper bounds.
`int Max(int a, int b, int c)`	Returns the maximum of three integers.
`int Min(int a, int b, int c)`	Returns the minimum of three integers.
Class: `Program`	
`int SequentialQuery (List points, Rectangle query)`	Sequentially runs, by brute force, a query through a list of points.