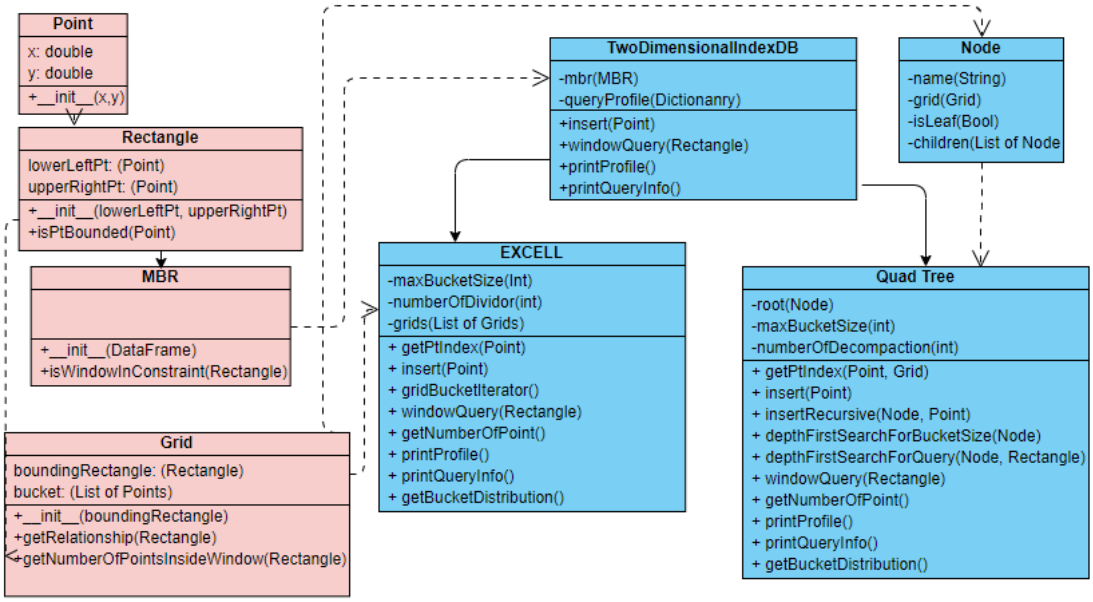
Name: Lam Chun Ting Jeff
SID: 12222973
ITSC: ctjlam

MSBD6000J Spatial and Multimedia Databases
Assignment 1 Report

Task

Given a list of point, we create an index for the points in (r,b) forms, where r is the rectangle, which call the index, and b for the buckets holding all the actual points. In this assignment, EXCELL method and Quad Tree method will be implemented.

System Architecture



(Figure 1. UML Diagram of system structure)

For red rectangle, it is general class definition for implementing the indexing method. For blue rectangle, those are Compulsory implementation.

For red part

Basic class like Point (x,y), and Rectangle (Storing the bottom left, and the upper right) is defined. And there is one operation in rectangle is that checking whether the a given point is inside the rectangle or not.

MBR (Minimum Bounding Rectangle) is a class inherited from Rectangle class. It will find the minimum bounding rectangle by passing a data frame with column X and Y, then find the mbr.

Grid is a definition is that fulfilling the task requirement (r,b) forms. And under that, there is two operation.

1. getRelationship(Rectangle) - is to find the relationship between grid itself with another rectangle. An Enum called RectangleRelationship is defined here. It will return either of the condition that to indicate the relationship.

```
# Define set of rectangle relationship
class RectangleRelationship(Enum):
    WARPPED = 1         # Rectanlge B in A
    CONTAIN = 2         # Some in A and B parts intercept
    NO_RELATION = 3     # Do not overlap
```

(Figure 2. Enum defination)

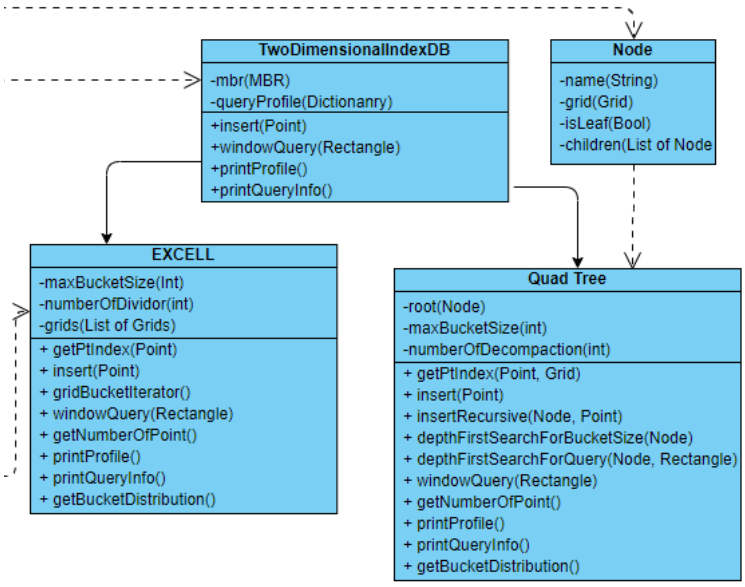
2. getNumberOfPointsInsideWindow(Rectangle) – is aims to do refinement. For a given rectangle like window, it will count the number of points in bucket that bounded by the rectangle. Noted that relationship CONTAIN will only be allowed to use the method, as if WARPPED, then will be the bucket size, and if it is NO_RELATION, then it will be 0.

   Procedure of this function

   1. Compute a new window with the grid constraint
   2. Loop through all point in bucket, use isPtBounded in rectangle to check if it is inside or not

Note: the only reason why grid is not inherited from rectangle is the question requirement, having a (r,b) term, for clearer representation in code.

For blue part, two-dimensional index DB is just a parent abstract class that indicate the method necessary for main program. Therefore, mbr and queryProfile (For query information) will be made, as any kind of indexing method will use these two variables.



(Figure 3. Cropped version of figure 1)

Name: Lam Chun Ting Jeff
SID: 12222973
ITSC: ctjlam

EXCELL:

A one-dimensional array of grids (named girds in class) is maintained. For number of divider be 1, the array length should be one; for 2, then it will be 4, which maintaining a NxN array, but in 1D form. Max Bucket Size is just a parameter that set the limit of each buckets contained (which also share with quad tree, but not necessary for other 2D method).

Insert and decompose:

The following will be the procedure of insert

1. Find the index of the 1D array grid
2. Append the point into that grid bucket
3. If the grid bucket exceeds the max bucket size, decompose

The procedure of decompose

1. Make a copy of old grid buckets
2. Change the 1D array size from (NxN) to (N+1xN+1)
   Note: the new grid will be bottomLeft as 0, topRight as NxN-1, topLeft as N-1.
3. Insert the old grid buckets point into new grids.

Window Query

1. Find the bottomLeftPt index and the upperRightPt index.
2. Loop through the grid with index in between the two index found in step 1.
3. For each grid, find the relationship between the grid and window.
4. If no relationship, directly ignore
5. If warpped, simply get the length of the grid bucket array
6. If contain, do computation of getNumberOfPointsInsideWindow.

Note, for finding index of the 1D grid, an equation has been used instead of brute force search.

Idx =(length of pt.x to left)/(lengthOfMBR in X)*number of divider

Idy =(length of pt.y to bottom)/(lengthOfMBR in Y)*number of divider

The index will be = idx * number of divider + idy

Answer of Task 2

```
Number of dividor: 272
Bucket Distribution: {'0': 62197, '1-25': 9979, '26-239': 1801, '240-255': 7, '256': 0}
```

(Figure 4. Minimal number of dividers and the grids distribution)

Quad Tree:

A tree structure is maintained. Class Node is defined that also contain grid definition, with tree node variable isLeaf, and children that contain the next node. Max Bucket Size is just a parameter that set the limit of each buckets contained (which also share with EXCELL, but not necessary for other 2D method).

Insert and decompose:

The following will be the procedure of insert

1. Start from root node, do recursion insert
2. Check if leaf, if leaf, append
   If node grid bucket exceeds max number of bucket size, do decompose
3. If it is not leaf node
   Find the index that the pt should contain
   For the children node contain that pt, repeat 2

The procedure of decompose

1. Change the current node to non-Leaf Node.
2. Append the node children with 4 grids, separate evenly with the current grid rectangle.
3. For all pts in current bucket, start from current node to perform recursive insertion in insert step 2.
4. Delete the bucket

Window Query

1. Perform Depth First search start from root
2. Check if leaf, if leaf, find the relationship with window
   If warpped, then all pt in bucket will be return
   If contained, then do computation of getNumberOfPointsInsideWindow.
3. If it is not leaf node
   For each node in children, find the relationship with window
   If the relationship is not no relationship, repeat 2 for that node

Answer of Task 3

```
Number of decompaction: 8
Bucket Distribution: {'0': 34, '1-25': 99, '26-239': 1611, '240-255': 27, '256': 3}
```

(Figure 5 Minimal number of decompaction and the grids distribution)

Name: Lam Chun Ting Jeff
SID: 12222973
ITSC: ctjlam

Main Flow of Program

1. Check if EXCELL.obj exist, read the file, or else it will create a EXCELL with reading the whole file, and insert using the algorithm mentioned above. After finishing insert all points into EXCELL, it will create EXCELL.obj, such that it would not take time for creating index when only for window query. The following screenshot will only works when EXCELL.obj exist, or else it will directly goes into create index process

```
========== EXCELL Found Info ==========
Minimum Bounding Rectangle: MBR
(LowPoint: Point (X: 115.435043, Y: 39.439945))
(MaxPoint: Point (X: 117.488743, Y: 41.029975))
Number of dividor: 272
Bucket Distribution: {'0': 62197, '1-25': 9979, '26-239': 1801, '240-255': 7, '256': 0}
Number of Points: 182323
========================================
Would you like to recreate excell? (y/n)
```

(Figure 6. EXCELL.obj found program screenshot, with task 1 MBR and task 2 answer)

```
Inserting into EXCELL: 100%|
Saving excell into ./excell.obj...
========== EXCELL Created Info ==========
Minimum Bounding Rectangle: MBR
(LowPoint: Point (X: 115.435043, Y: 39.439945))
(MaxPoint: Point (X: 117.488743, Y: 41.029975))
Number of dividor: 272
Bucket Distribution: {'0': 62197, '1-25': 9979, '26-239': 1801, '240-255': 7, '256': 0}
Number of Points: 182323
========================================
```

(Figure 7. EXCELL insertion and finish saving program screenshot, with task 1 MBR and task 2 answer)

2. Check if quadTree.obj exist, read the file, or else it will create a Quad Tree with reading the whole file, and insert using the algorithm mentioned above. After finishing insert all points into Quad Tree, it will create quadTree.obj, such that it would not take time for creating index when only for window query. he following screenshot will only works when quadTree.obj exist, or else it will directly goes into create index process

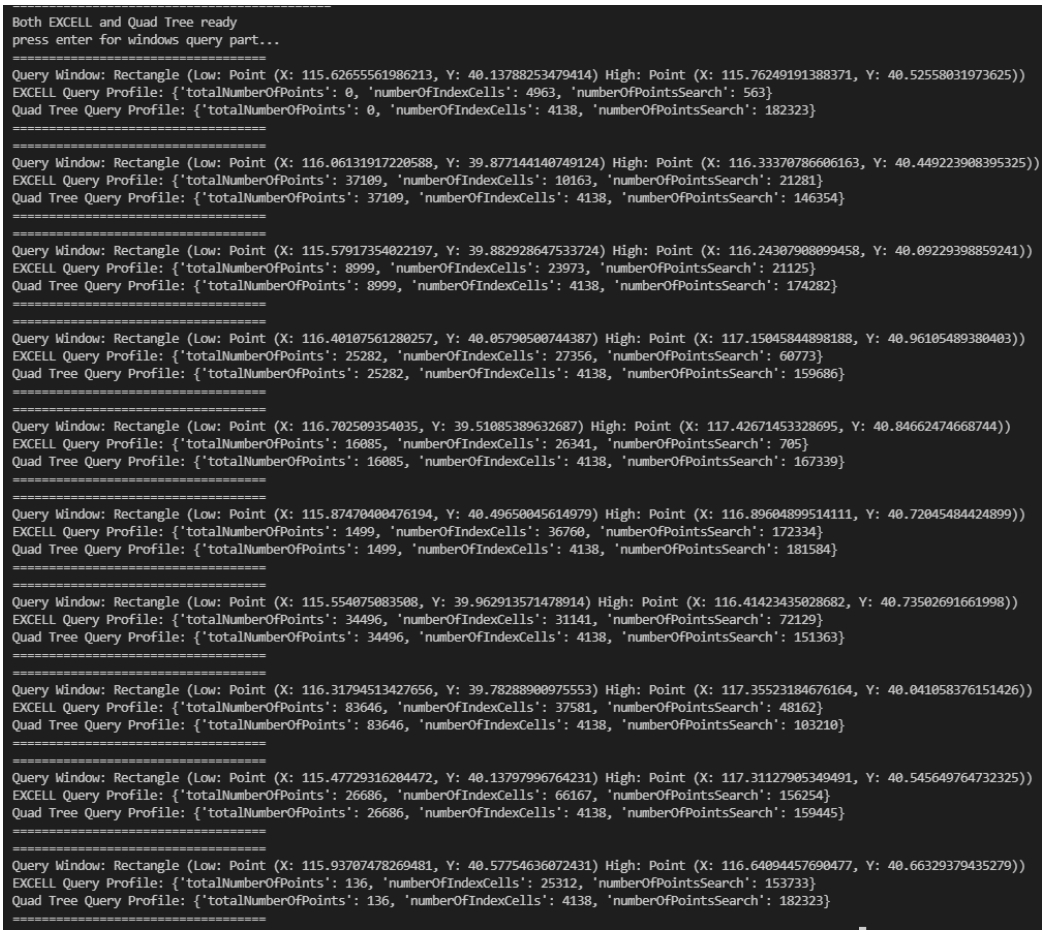
```
========= Quad Tree Found Info ==========
Minimum Bounding Rectangle: MBR
(LowPoint: Point (X: 115.435043, Y: 39.439945))
(MaxPoint: Point (X: 117.488743, Y: 41.029975))
Number of decompaction: 8
Bucket Distribution: {'0': 34, '1-25': 99, '26-239': 1611, '240-255': 27, '256': 3}
Number of Points: 182323
========================================
Would you like to recreate Quad Tree? (y/n)
```

(Figure 8. quadTree.obj found program screenshot, with task 1 MBR and task 3 answer)

```
========================================
Would you like to recreate Quad Tree? (y/n) y
Inserting into quad tree: 100%|          | 182323/182323 [00:10<00:00, 16832.73it/s]
Saving quad tree into ./quadTree.obj...
========= Quad Tree Created Info ==========
Minimum Bounding Rectangle: MBR
(LowPoint: Point (X: 115.435043, Y: 39.439945))
(MaxPoint: Point (X: 117.488743, Y: 41.029975))
Number of decompaction: 8
Bucket Distribution: {'0': 34, '1-25': 99, '26-239': 1611, '240-255': 27, '256': 3}
Number of Points: 182323
========================================
```

(Figure 9. Quad Tree insertion and finish saving program screenshot, with task 1 MBR and task 3 answer)

3. It will generate 10 windows (Rectangle type), within the MBR constraint. Then it will display the result. For each window, both EXCELL and Quad Tree will print the report



(Figure 10. Random Window Query and its profile)

Comparison

	EXCELL	Quad Tree
number of index cells	More (73,984)	Less (1,774)
bucket space utilisation	Inefficient (84% with 0 bucket length)	Efficient (~1% with 0 bucket length)
search costs	High As we have to visit 0 bucket cell, but which is unnecessary	Less As we have only visit the related node, for unrelated, it will be pruned. And all node must have data
index maintenance overhead	High - as it have to rearrange all grids bounded by MBR.	Less, - as it can only decompose the related grids (sub tree) without affecting other grids (any other sub tree or parent tree)