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

MSBD6000J Spatial and Multimedia Databases
Assignment 2 Report

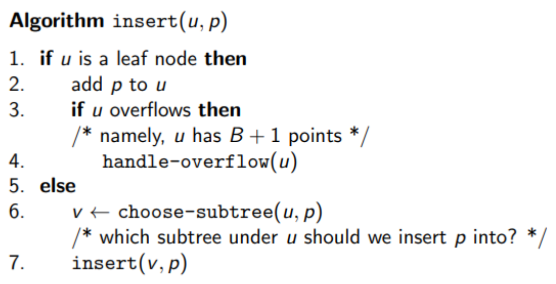
Task

Build an R-Tree, and perform nearest-neighbor search with pruning.

Task 1 (2) – Implementation of R-Tree

The code retrieved from the following GitHub repository by LBDM2707[1] and followed the implementation from CUHK Prof Tao Yu Fei slides [2]. All pseudocode is retrieved from the slides.

Basically, the construction of R-Tree starts from insert function.

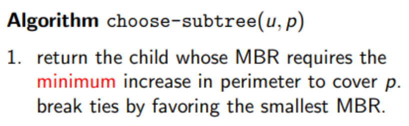


```
Algorithm insert(u, p)
1. if u is a leaf node then
2.     add p to u
3.     if u overflows then
       /* namely, u has B + 1 points */
4.         handle-overflow(u)
5. else
6.     v ← choose-subtree(u, p)
       /* which subtree under u should we insert p into? */
7.     insert(v, p)
```

There are two overflow conditions of the node

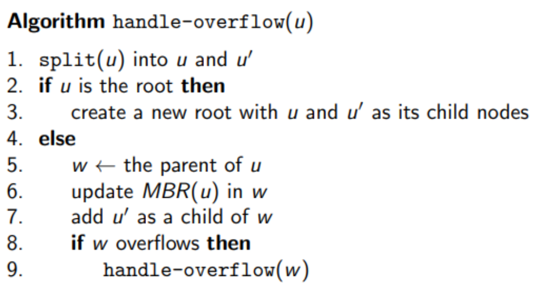
1. Data points > max bucket size (b) (Leaf node)
2. Child nodes > Sub Tree (d) (Non-Leaf node)

The way of choosing the sub-tree in non-leaf node is by finding the minimum increase in perimeter of the MBR.



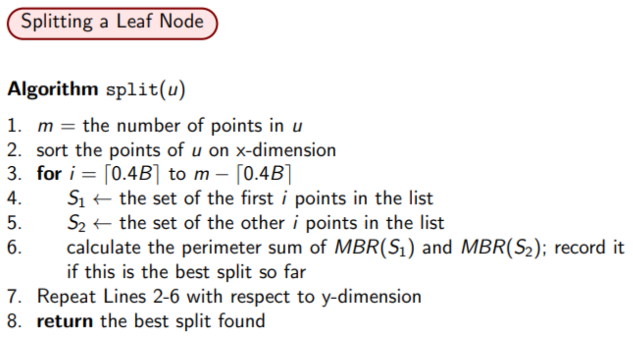
```
Algorithm choose-subtree(u, p)
1. return the child whose MBR requires the
   minimum increase in perimeter to cover p.
   break ties by favoring the smallest MBR.
```

The way of overflow handling.



```
Algorithm handle-overflow(u)
1. split(u) into u and u'
2. if u is the root then
3.     create a new root with u and u' as its child nodes
4. else
5.     w ← the parent of u
6.     update MBR(u) in w
7.     add u' as a child of w
8.     if w overflows then
9.         handle-overflow(w)
```

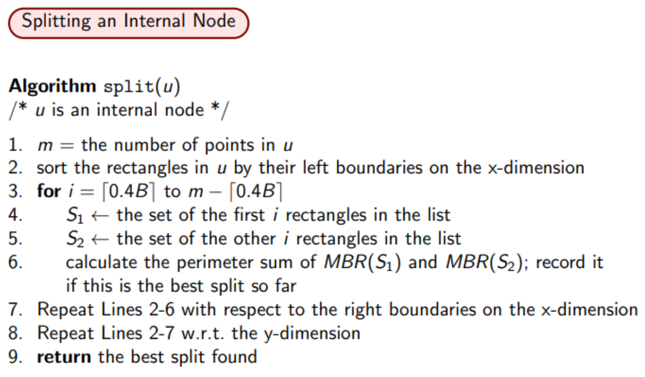
Splitting a leaf node when overflow happen



Splitting a Leaf Node

```
Algorithm split(u)
1. m = the number of points in u
2. sort the points of u on x-dimension
3. for i = ⌈0.4B⌉ to m − ⌈0.4B⌉
4.     S_1 ← the set of the first i points in the list
5.     S_2 ← the set of the other i points in the list
6.     calculate the perimeter sum of MBR(S_1) and MBR(S_2); record it
       if this is the best split so far
7. Repeat Lines 2-6 with respect to y-dimension
8. return the best split found
```

Splitting a non-node when overflow happen



Splitting an Internal Node

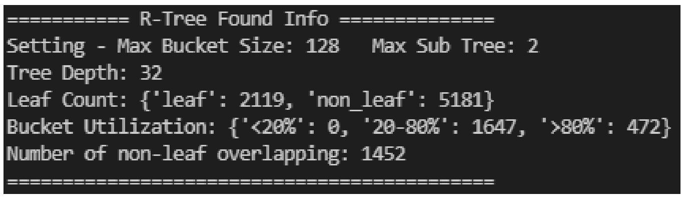
```
Algorithm split(u)
/* u is an internal node */
1. m = the number of points in u
2. sort the rectangles in u by their left boundaries on the x-dimension
3. for i = ⌈0.4B⌉ to m − ⌈0.4B⌉
4.     S_1 ← the set of the first i rectangles in the list
5.     S_2 ← the set of the other i rectangles in the list
6.     calculate the perimeter sum of MBR(S_1) and MBR(S_2); record it
       if this is the best split so far
7. Repeat Lines 2-6 with respect to the right boundaries on the x-dimension
8. Repeat Lines 2-7 w.r.t. the y-dimension
9. return the best split found
```

Then for each point in dataset, it will go from the insert function. It will directly traverse to the sub tree in a recursive manner. Once it reaches the leaf node, then it will add the point. Check if any overflow happened, then it performs split in its parent layer first. If parent layer still happened overflow, then it will continue traverse up until it is balance and no overflow happened.

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

Task 1 (3) – Result of the constructed R-Tree

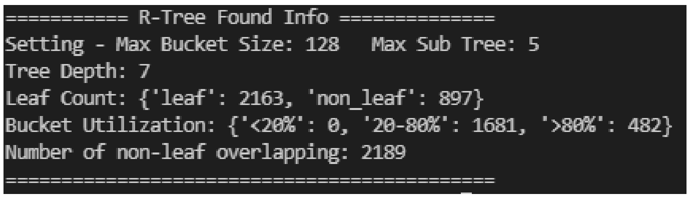
R Tree setting (n=128, d=2)



```
=========== R-Tree Found Info ==============
Setting - Max Bucket Size: 128   Max Sub Tree: 2
Tree Depth: 32
Leaf Count: {'leaf': 2119, 'non_leaf': 5181}
Bucket Utilization: {'<20%': 0, '20-80%': 1647, '>80%': 472}
Number of non-leaf overlapping: 1452
============================================
```

There is a tree depth (height) 32, 2119 leaf node and 5181 non-leaf nodes. For each bucket in leaf node containing points within n, there is no bucket under 20%, 1647 within 20%-80%, and 472 over 80%. And there are 1452 pairs of non-leaf node have an overlapping region.

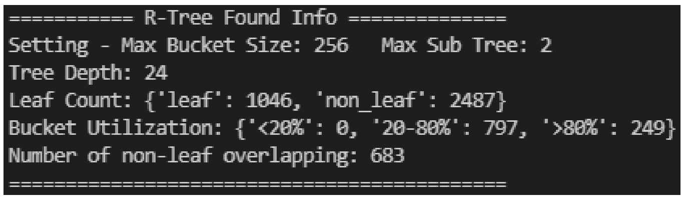
R Tree setting (n=128, d=5)



```
=========== R-Tree Found Info ==============
Setting - Max Bucket Size: 128   Max Sub Tree: 5
Tree Depth: 7
Leaf Count: {'leaf': 2163, 'non_leaf': 897}
Bucket Utilization: {'<20%': 0, '20-80%': 1681, '>80%': 482}
Number of non-leaf overlapping: 2189
============================================
```

There is a tree depth (height) 7, 2163 leaf node and 897 non-leaf nodes. For each bucket in leaf node containing points within n, there is no bucket under 20%, 1681 within 20%-80%, and 482 over 80%. And there are 2189 pairs of non-leaf node have an overlapping region.

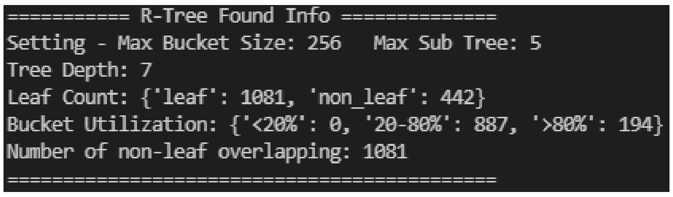
R Tree setting (n=256, d=2)



```
=========== R-Tree Found Info ==============
Setting - Max Bucket Size: 256   Max Sub Tree: 2
Tree Depth: 24
Leaf Count: {'leaf': 1046, 'non_leaf': 2487}
Bucket Utilization: {'<20%': 0, '20-80%': 797, '>80%': 249}
Number of non-leaf overlapping: 683
============================================
```

There is a tree depth (height) 24, 1046 leaf node and 2497 non-leaf nodes. For each bucket in leaf node containing points within n, there is no bucket under 20%, 797 within 20%-80%, and 249 over 80%. And there are 683 pairs of non-leaf node have an overlapping region.

R Tree setting (n=256, d=5)



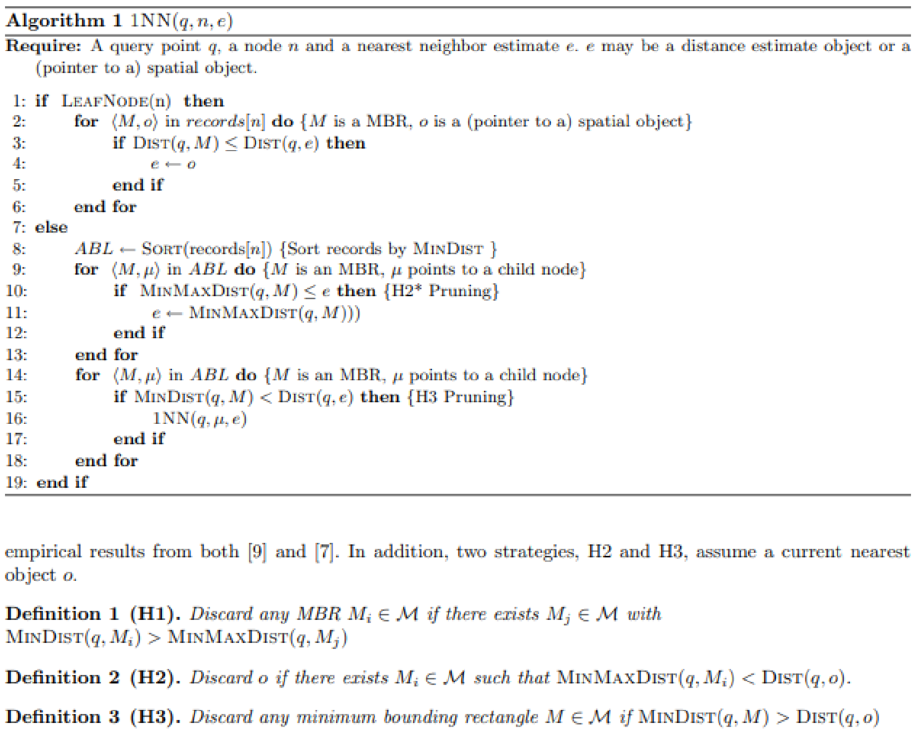
```
=========== R-Tree Found Info ==============
Setting - Max Bucket Size: 256   Max Sub Tree: 5
Tree Depth: 7
Leaf Count: {'leaf': 1081, 'non_leaf': 442}
Bucket Utilization: {'<20%': 0, '20-80%': 887, '>80%': 194}
Number of non-leaf overlapping: 1081
============================================
```

There is a tree depth (height) 7, 1081 leaf node and 442 non-leaf nodes. For each bucket in leaf node containing points within n, there is no bucket under 20%, 887 within 20%-80%, and 194 over 80%. And there are 1081 pairs of non-leaf node have an overlapping region.

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

Task 2 (2) – Implementation of nearest neighbor search

For nearest neighbor search, I follow the implementation from Search Space Reductions for Nearest-Neighbor Queries (TAMC, 2008) [3]. The code follows exactly what written in paper with the three pruning rules. The screenshot from this section is from the papers.



Algorithm 1 1NN(q, n, e)

Require: A query point q, a node n and a nearest neighbor estimate e. e may be a distance estimate object or a (pointer to a) spatial object.

```
if LeafNode(n) then
    for ⟨$M, o$⟩ in $records[n]$ do {$M$ is a MBR, $o$ is a (pointer to a) spatial object}
        if Dist$(q, M) \le$ Dist$(q, e)$ then
            $e \leftarrow o$
        end if
    end for
else
    $ABL \leftarrow$ Sort(records[$n$]) {Sort records by MinDist }
    for ⟨$M, \mu$⟩ in $ABL$ do {$M$ is an MBR, $\mu$ points to a child node}
        if MinMaxDist$(q, M) \le e$ then {H2* Pruning}
            $e \leftarrow$ MinMaxDist$(q, M)$))
        end if
    end for
    for ⟨$M, \mu$⟩ in $ABL$ do {$M$ is an MBR, $\mu$ points to a child node}
        if MinDist$(q, M) <$ Dist$(q, e)$ then {H3 Pruning}
            1NN$(q, \mu, e)$
        end if
    end for
end if
```

empirical results from both [9] and [7]. In addition, two strategies, H2 and H3, assume a current nearest object o.

Definition 1 (H1). *Discard any MBR $M_i \in \mathcal{M}$ if there exists $M_j \in \mathcal{M}$ with* MinDist$(q, M_i) >$ MinMaxDist(q, M_j)

Definition 2 (H2). *Discard o if there exists $M_i \in \mathcal{M}$ such that* MinMaxDist$(q, M_i) <$ Dist(q, o).

Definition 3 (H3). *Discard any minimum bounding rectangle $M \in \mathcal{M}$ if* MinDist$(q, M) >$ Dist(q, o)

<u>Clarify in pseudocode</u>

e issue - In actual implementation, for e (nearest neighbor instance) can be either a number, or an instance of Point. Therefore, the e in the actual code is defined as dictionary, with key ‘pt’ and ‘dist’. ‘pt’ contains the nn point, and ‘dist’ contain the distance for pruning and the actual nn distance for final result.

ABL Issue – ABL should have sorted with records n. It is constructed by getting all instance in child node and compute its related MBR minimum distance, and put the data into ABL list, which contain tuple (node, min_dist). Then for the rest of looping, the code loops the tuple (node, min_dist) in the ABL list.

Minimax distance issue – It computed as follows, retrieved from Nearest Neighbor Queries by Nick Roussopoulos[4].

$$MINMAXDIST(P, R) =$$

$$\min_{1 \le k \le n} \left(|p_k - rm_k|^2 + \sum_{\substack{i \ne k \\ 1 \le i \le n}} |p_i - rM_i|^2 \right)$$

where:

$$rm_k = \begin{cases} s_k & if\ p_k \le \frac{(s_k + t_k)}{2}; \\ t_k & otherwise. \end{cases} \quad and$$

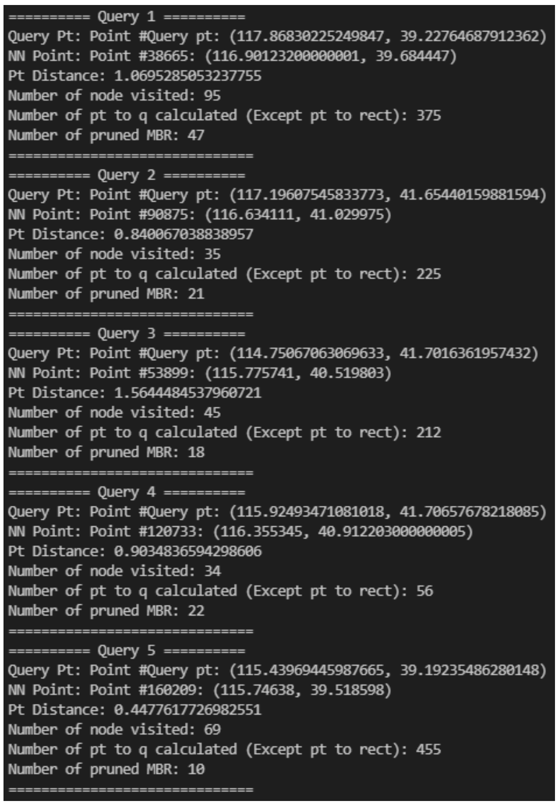
$$rM_i = \begin{cases} s_i & if\ p_i \ge \frac{(s_i + t_i)}{2}; \\ t_i & otherwise. \end{cases}$$

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

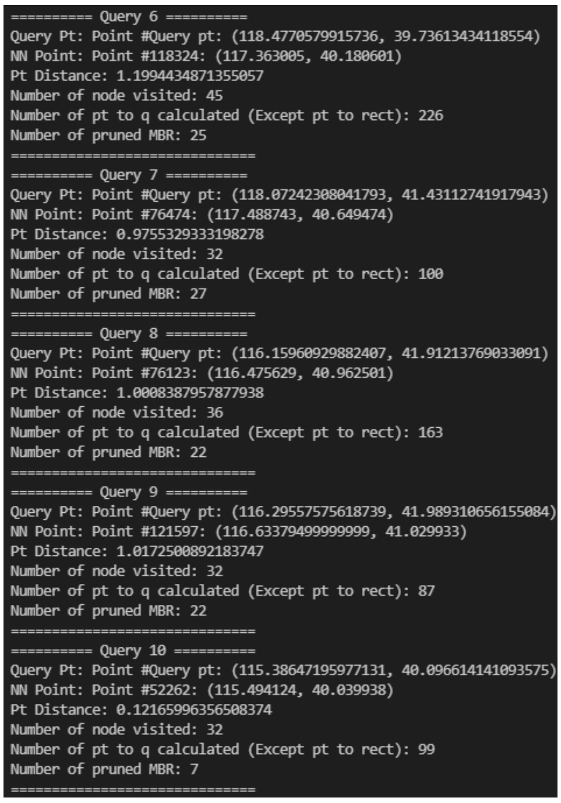
Task 2 (3) –Nearest Neighbor Search Result

The query points are randomly generated the minimum and maximum value in dataset from min -1 to max + 1

R Tree setting (n=128, d=2)

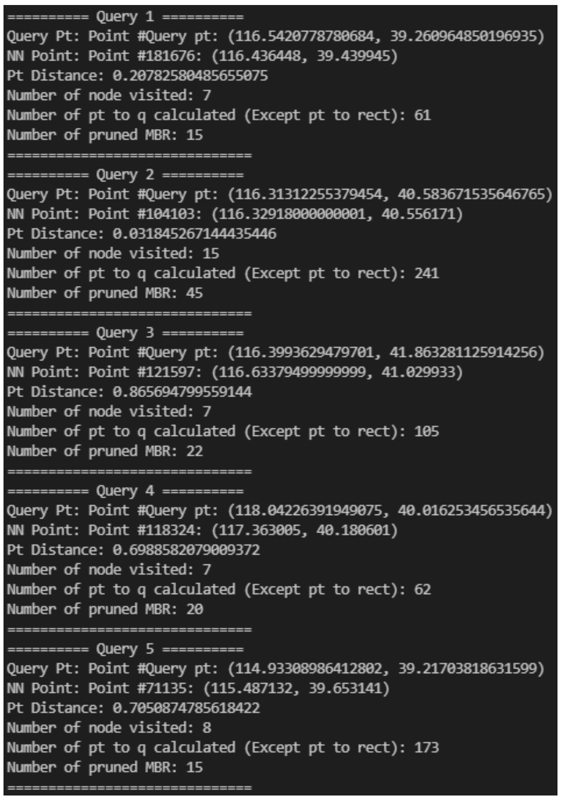


```
========== Query 1 ==========
Query Pt: Point #Query pt: (117.86830225249847, 39.22764687912362)
NN Point: Point #38665: (116.90123200000001, 39.684447)
Pt Distance: 1.0695285053237755
Number of node visited: 95
Number of pt to q calculated (Except pt to rect): 375
Number of pruned MBR: 47
==============================
========== Query 2 ==========
Query Pt: Point #Query pt: (117.19607545833773, 41.65440159881594)
NN Point: Point #90875: (116.634111, 41.029975)
Pt Distance: 0.840067038838957
Number of node visited: 35
Number of pt to q calculated (Except pt to rect): 225
Number of pruned MBR: 21
==============================
========== Query 3 ==========
Query Pt: Point #Query pt: (114.75067063069633, 41.7016361957432)
NN Point: Point #53899: (115.775741, 40.519803)
Pt Distance: 1.5644484537960721
Number of node visited: 45
Number of pt to q calculated (Except pt to rect): 212
Number of pruned MBR: 18
==============================
========== Query 4 ==========
Query Pt: Point #Query pt: (115.92493471081018, 41.70657678218085)
NN Point: Point #120733: (116.355345, 40.912203000000005)
Pt Distance: 0.9034836594298606
Number of node visited: 34
Number of pt to q calculated (Except pt to rect): 56
Number of pruned MBR: 22
==============================
========== Query 5 ==========
Query Pt: Point #Query pt: (115.43969445987665, 39.19235486280148)
NN Point: Point #160209: (115.74638, 39.518598)
Pt Distance: 0.4477617726982551
Number of node visited: 69
Number of pt to q calculated (Except pt to rect): 455
Number of pruned MBR: 10
==============================
```

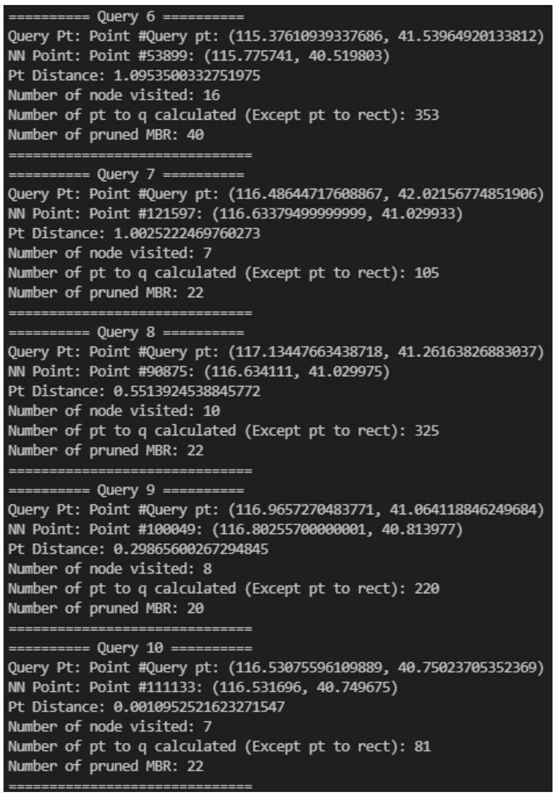


```
========== Query 6 ==========
Query Pt: Point #Query pt: (118.4770579915736, 39.73613434118554)
NN Point: Point #118324: (117.363005, 40.180601)
Pt Distance: 1.1994434871355057
Number of node visited: 45
Number of pt to q calculated (Except pt to rect): 226
Number of pruned MBR: 25
==============================
========== Query 7 ==========
Query Pt: Point #Query pt: (118.07242308041793, 41.43112741917943)
NN Point: Point #76474: (117.488743, 40.649474)
Pt Distance: 0.9755329333198278
Number of node visited: 32
Number of pt to q calculated (Except pt to rect): 100
Number of pruned MBR: 27
==============================
========== Query 8 ==========
Query Pt: Point #Query pt: (116.15960929882407, 41.91213769033091)
NN Point: Point #76123: (116.475629, 40.962501)
Pt Distance: 1.0008387957877938
Number of node visited: 36
Number of pt to q calculated (Except pt to rect): 163
Number of pruned MBR: 22
==============================
========== Query 9 ==========
Query Pt: Point #Query pt: (116.29557575618739, 41.989310656155084)
NN Point: Point #121597: (116.63379499999999, 41.029933)
Pt Distance: 1.0172500892183747
Number of node visited: 32
Number of pt to q calculated (Except pt to rect): 87
Number of pruned MBR: 22
==============================
========== Query 10 ==========
Query Pt: Point #Query pt: (115.38647195977131, 40.096614141093575)
NN Point: Point #52262: (115.494124, 40.039938)
Pt Distance: 0.12165996356508374
Number of node visited: 32
Number of pt to q calculated (Except pt to rect): 99
Number of pruned MBR: 7
==============================
```

R Tree setting (n=128, d=5)

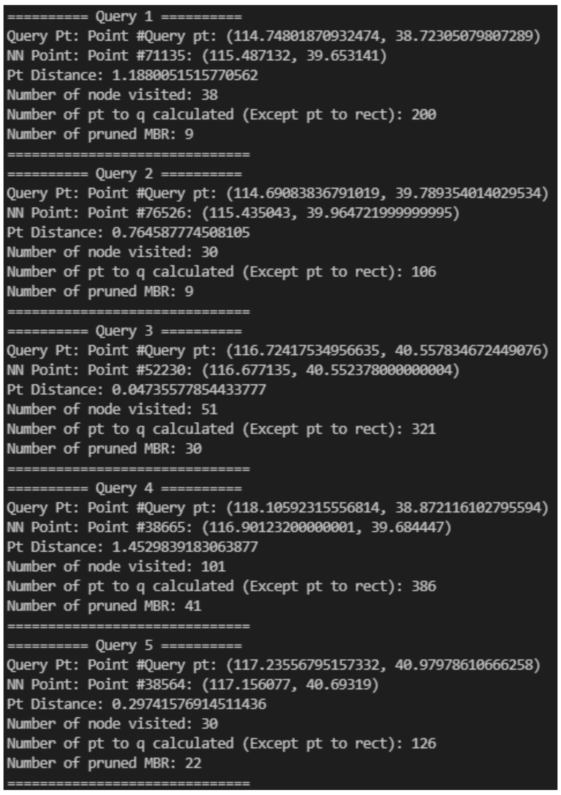


```
========== Query 1 ==========
Query Pt: Point #Query pt: (116.5420778780684, 39.260964850196935)
NN Point: Point #181676: (116.436448, 39.439945)
Pt Distance: 0.20782580485655075
Number of node visited: 7
Number of pt to q calculated (Except pt to rect): 61
Number of pruned MBR: 15
==============================
========== Query 2 ==========
Query Pt: Point #Query pt: (116.31312255379454, 40.583671535646765)
NN Point: Point #104103: (116.32918000000001, 40.556171)
Pt Distance: 0.031845267144435446
Number of node visited: 15
Number of pt to q calculated (Except pt to rect): 241
Number of pruned MBR: 45
==============================
========== Query 3 ==========
Query Pt: Point #Query pt: (116.3993629479701, 41.863281125914256)
NN Point: Point #121597: (116.63379499999999, 41.029933)
Pt Distance: 0.865694799559144
Number of node visited: 7
Number of pt to q calculated (Except pt to rect): 105
Number of pruned MBR: 22
==============================
========== Query 4 ==========
Query Pt: Point #Query pt: (118.04226391949075, 40.016253456535644)
NN Point: Point #118324: (117.363005, 40.180601)
Pt Distance: 0.6988582079009372
Number of node visited: 7
Number of pt to q calculated (Except pt to rect): 62
Number of pruned MBR: 20
==============================
========== Query 5 ==========
Query Pt: Point #Query pt: (114.93308986412802, 39.21703818631599)
NN Point: Point #71135: (115.487132, 39.653141)
Pt Distance: 0.7050874785618422
Number of node visited: 8
Number of pt to q calculated (Except pt to rect): 173
Number of pruned MBR: 15
==============================
```

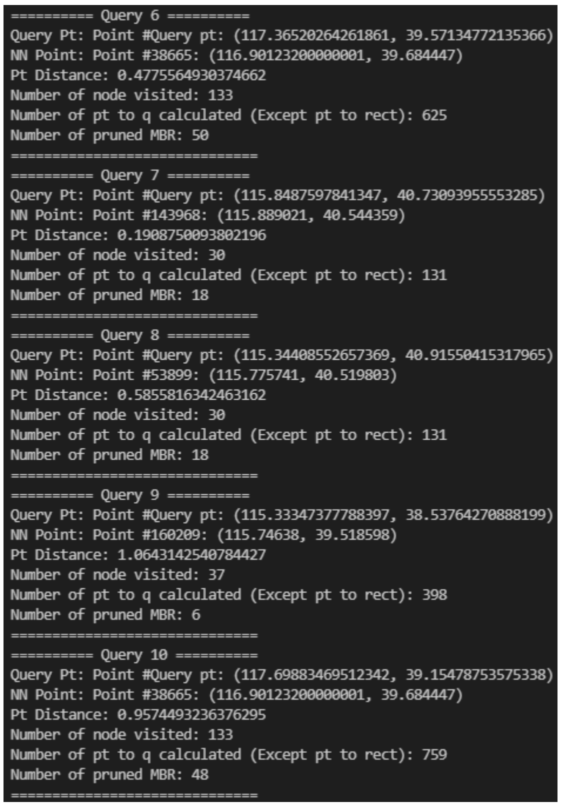


```
========== Query 6 ==========
Query Pt: Point #Query pt: (115.37610939337686, 41.53964920133812)
NN Point: Point #53899: (115.775741, 40.519803)
Pt Distance: 1.0953500332751975
Number of node visited: 16
Number of pt to q calculated (Except pt to rect): 353
Number of pruned MBR: 40
==============================
========== Query 7 ==========
Query Pt: Point #Query pt: (116.48644717608867, 42.02156774851906)
NN Point: Point #121597: (116.63379499999999, 41.029933)
Pt Distance: 1.0025222469760273
Number of node visited: 7
Number of pt to q calculated (Except pt to rect): 105
Number of pruned MBR: 22
==============================
========== Query 8 ==========
Query Pt: Point #Query pt: (117.13447663438718, 41.26163826883037)
NN Point: Point #90875: (116.634111, 41.029975)
Pt Distance: 0.5513924538845772
Number of node visited: 10
Number of pt to q calculated (Except pt to rect): 325
Number of pruned MBR: 22
==============================
========== Query 9 ==========
Query Pt: Point #Query pt: (116.9657270483771, 41.064118846249684)
NN Point: Point #100049: (116.80255700000001, 40.813977)
Pt Distance: 0.29865600267294845
Number of node visited: 8
Number of pt to q calculated (Except pt to rect): 220
Number of pruned MBR: 20
==============================
========== Query 10 ==========
Query Pt: Point #Query pt: (116.53075596109889, 40.75023705352369)
NN Point: Point #111133: (116.531696, 40.749675)
Pt Distance: 0.0010952521623271547
Number of node visited: 7
Number of pt to q calculated (Except pt to rect): 81
Number of pruned MBR: 22
==============================
```

R Tree setting (n=156, d=2)

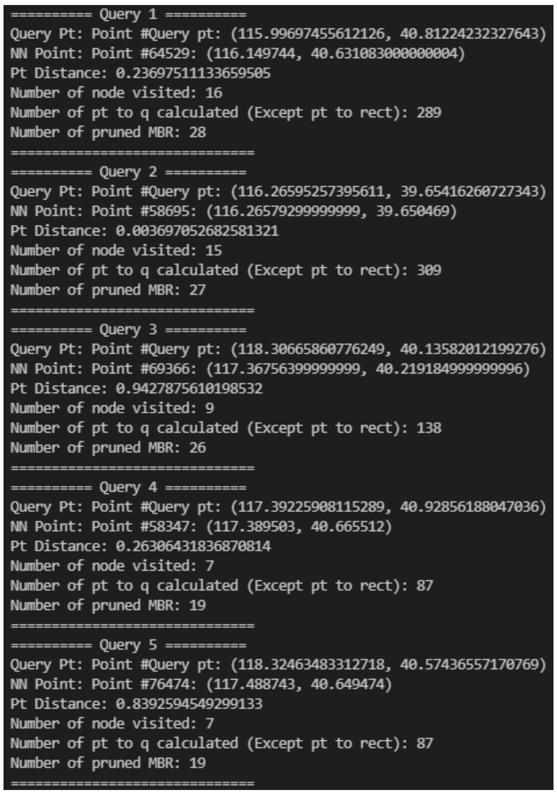


```
========== Query 1 ==========
Query Pt: Point #Query pt: (114.74801870932474, 38.72305079807289)
NN Point: Point #71135: (115.487132, 39.653141)
Pt Distance: 1.1880051515770562
Number of node visited: 38
Number of pt to q calculated (Except pt to rect): 200
Number of pruned MBR: 9
==============================
========== Query 2 ==========
Query Pt: Point #Query pt: (114.69083836791019, 39.789354014029534)
NN Point: Point #76526: (115.435043, 39.964721999999995)
Pt Distance: 0.764587774508105
Number of node visited: 30
Number of pt to q calculated (Except pt to rect): 106
Number of pruned MBR: 9
==============================
========== Query 3 ==========
Query Pt: Point #Query pt: (116.72417534956635, 40.557834672449076)
NN Point: Point #52230: (116.677135, 40.552378000000004)
Pt Distance: 0.04735577854433777
Number of node visited: 51
Number of pt to q calculated (Except pt to rect): 321
Number of pruned MBR: 30
==============================
========== Query 4 ==========
Query Pt: Point #Query pt: (118.10592315556814, 38.872116102795594)
NN Point: Point #38665: (116.90123200000001, 39.684447)
Pt Distance: 1.4529839183063877
Number of node visited: 101
Number of pt to q calculated (Except pt to rect): 386
Number of pruned MBR: 41
==============================
========== Query 5 ==========
Query Pt: Point #Query pt: (117.23556795157332, 40.97978610666258)
NN Point: Point #38564: (117.156077, 40.69319)
Pt Distance: 0.29741576914511436
Number of node visited: 30
Number of pt to q calculated (Except pt to rect): 126
Number of pruned MBR: 22
==============================
```

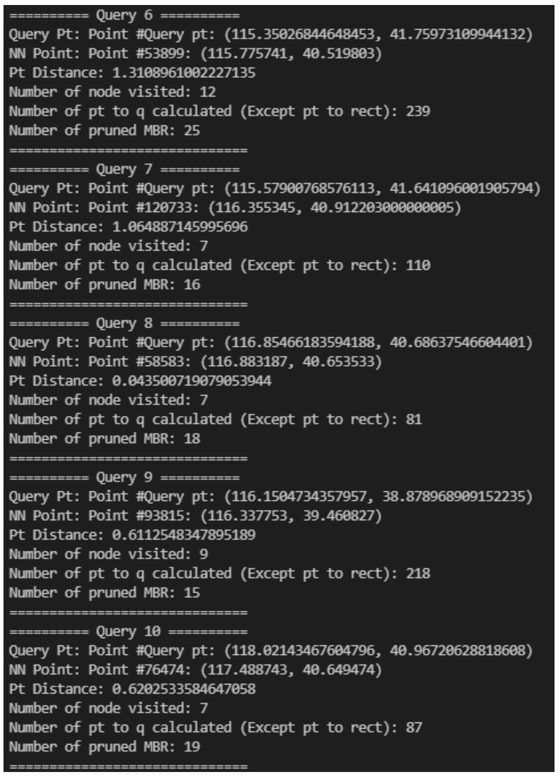


```
========== Query 6 ==========
Query Pt: Point #Query pt: (117.36520264261861, 39.57134772135366)
NN Point: Point #38665: (116.90123200000001, 39.684447)
Pt Distance: 0.4775564930374662
Number of node visited: 133
Number of pt to q calculated (Except pt to rect): 625
Number of pruned MBR: 50
==============================
========== Query 7 ==========
Query Pt: Point #Query pt: (115.8487597841347, 40.73093955553285)
NN Point: Point #143968: (115.889021, 40.544359)
Pt Distance: 0.1908750093802196
Number of node visited: 30
Number of pt to q calculated (Except pt to rect): 131
Number of pruned MBR: 18
==============================
========== Query 8 ==========
Query Pt: Point #Query pt: (115.34408552657369, 40.91550415317965)
NN Point: Point #53899: (115.775741, 40.519803)
Pt Distance: 0.5855816342463162
Number of node visited: 30
Number of pt to q calculated (Except pt to rect): 131
Number of pruned MBR: 18
==============================
========== Query 9 ==========
Query Pt: Point #Query pt: (115.33347377788397, 38.53764270888199)
NN Point: Point #160209: (115.74638, 39.518598)
Pt Distance: 1.0643142540784427
Number of node visited: 37
Number of pt to q calculated (Except pt to rect): 398
Number of pruned MBR: 6
==============================
========== Query 10 ==========
Query Pt: Point #Query pt: (117.69883469512342, 39.15478753575338)
NN Point: Point #38665: (116.90123200000001, 39.684447)
Pt Distance: 0.9574493236376295
Number of node visited: 133
Number of pt to q calculated (Except pt to rect): 759
Number of pruned MBR: 48
==============================
```

R Tree setting (n=156, d=5)



```
========== Query 1 ==========
Query Pt: Point #Query pt: (115.99697455612126, 40.81224232327643)
NN Point: Point #64529: (116.149744, 40.631083000000004)
Pt Distance: 0.23697511133659505
Number of node visited: 16
Number of pt to q calculated (Except pt to rect): 289
Number of pruned MBR: 28
==============================
========== Query 2 ==========
Query Pt: Point #Query pt: (116.26595257395611, 39.65416260727343)
NN Point: Point #58695: (116.26579299999999, 39.650469)
Pt Distance: 0.003697052682581321
Number of node visited: 15
Number of pt to q calculated (Except pt to rect): 309
Number of pruned MBR: 27
==============================
========== Query 3 ==========
Query Pt: Point #Query pt: (118.30665860776249, 40.13582012199276)
NN Point: Point #69366: (117.36756399999999, 40.219184999999996)
Pt Distance: 0.9427875610198532
Number of node visited: 9
Number of pt to q calculated (Except pt to rect): 138
Number of pruned MBR: 26
==============================
========== Query 4 ==========
Query Pt: Point #Query pt: (117.39225908115289, 40.92856188047036)
NN Point: Point #58347: (117.389503, 40.665512)
Pt Distance: 0.26306431836870814
Number of node visited: 7
Number of pt to q calculated (Except pt to rect): 87
Number of pruned MBR: 19
==============================
========== Query 5 ==========
Query Pt: Point #Query pt: (118.32463483312718, 40.57436557170769)
NN Point: Point #76474: (117.488743, 40.649474)
Pt Distance: 0.8392594549299133
Number of node visited: 7
Number of pt to q calculated (Except pt to rect): 87
Number of pruned MBR: 19
==============================
```

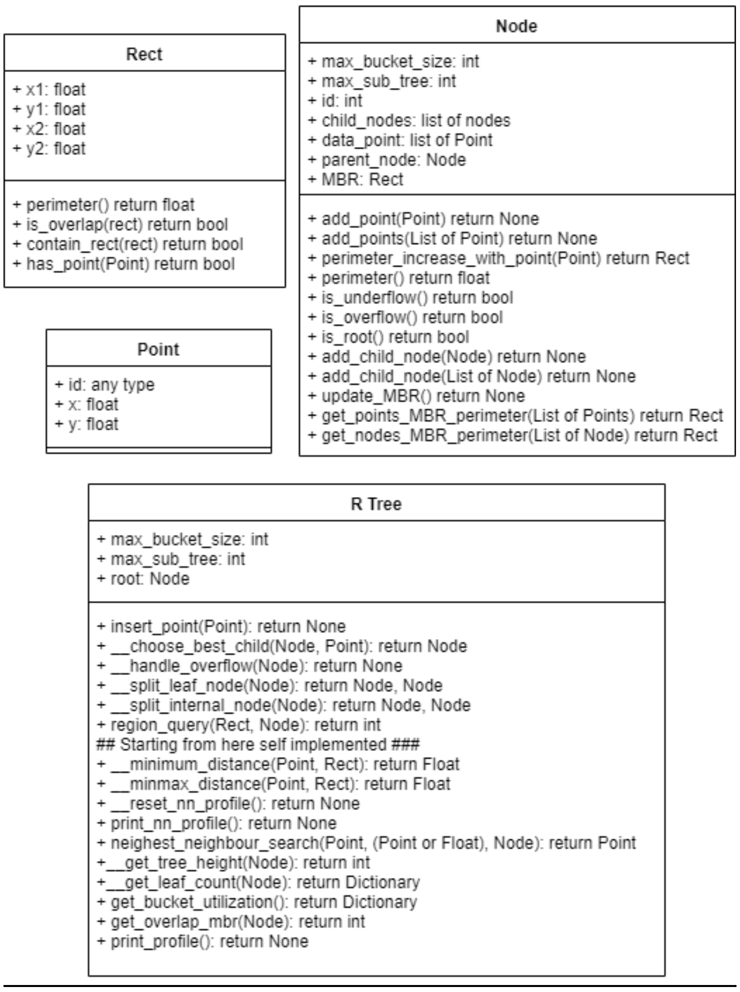


```
========== Query 6 ==========
Query Pt: Point #Query pt: (115.35026844648453, 41.75973109944132)
NN Point: Point #53899: (115.775741, 40.519803)
Pt Distance: 1.3108961002227135
Number of node visited: 12
Number of pt to q calculated (Except pt to rect): 239
Number of pruned MBR: 25
==============================
========== Query 7 ==========
Query Pt: Point #Query pt: (115.57900768576113, 41.641096001905794)
NN Point: Point #120733: (116.355345, 40.912203000000005)
Pt Distance: 1.064887145995696
Number of node visited: 7
Number of pt to q calculated (Except pt to rect): 110
Number of pruned MBR: 16
==============================
========== Query 8 ==========
Query Pt: Point #Query pt: (116.85466183594188, 40.68637546604401)
NN Point: Point #58583: (116.883187, 40.653533)
Pt Distance: 0.043500719079053944
Number of node visited: 7
Number of pt to q calculated (Except pt to rect): 81
Number of pruned MBR: 18
==============================
========== Query 9 ==========
Query Pt: Point #Query pt: (116.1504734357957, 38.878968909152235)
NN Point: Point #93815: (116.337753, 39.460827)
Pt Distance: 0.6112548347895189
Number of node visited: 9
Number of pt to q calculated (Except pt to rect): 218
Number of pruned MBR: 15
==============================
========== Query 10 ==========
Query Pt: Point #Query pt: (118.02143467604796, 40.96720628818608)
NN Point: Point #76474: (117.488743, 40.649474)
Pt Distance: 0.6202533584647058
Number of node visited: 7
Number of pt to q calculated (Except pt to rect): 87
Number of pruned MBR: 19
==============================
```

For number of MBR pruned, it only counts the pruned in same node. It does not count the pruned sub-tree, or children MBR.

Appendix

UML Class Diagram for the program structure as reference.


Rect
+ x1: float
+ y1: float
+ x2: float
+ y2: float
+ perimeter() return float
+ is_overlap(rect) return bool
+ contain_rect(rect) return bool
+ has_point(Point) return bool
Point
+ id: any type
+ x: float
+ y: float
Node
+ max_bucket_size: int
+ max_sub_tree: int
+ id: int
+ child_nodes: list of nodes
+ data_point: list of Point
+ parent_node: Node
+ MBR: Rect
+ add_point(Point) return None
+ add_points(List of Point) return None
+ perimeter_increase_with_point(Point) return Rect
+ perimeter() return float
+ is_underflow() return bool
+ is_overflow() return bool
+ is_root() return bool
+ add_child_node(Node) return None
+ add_child_node(List of Node) return None
+ update_MBR() return None
+ get_points_MBR_perimeter(List of Points) return Rect
+ get_nodes_MBR_perimeter(List of Node) return Rect
R Tree
+ max_bucket_size: int
+ max_sub_tree: int
+ root: Node
+ insert_point(Point): return None
+ __choose_best_child(Node, Point): return Node
+ __handle_overflow(Node): return None
+ __split_leaf_node(Node): return Node, Node
+ __split_internal_node(Node): return Node, Node
+ region_query(Rect, Node): return int
## Starting from here self implemented ###
+ __minimum_distance(Point, Rect): return Float
+ __minmax_distance(Point, Rect): return Float
+ __reset_nn_profile(): return None
+ print_nn_profile(): return None
+ neighest_neighbour_search(Point, (Point or Float), Node): return Point
+__get_tree_height(Node): return int
+__get_leaf_count(Node): return Dictionary
+ get_bucket_utilization(): return Dictionary
+ get_overlap_mbr(Node): return int
+ print_profile(): return None

References

[1] LBDM2707. (n.d.). *Python_r-tree/region_tree.py at master · LBDM2707/python_r-tree*. GitHub. Retrieved November 17, 2021, from https://github.com/LBDM2707/Python_R-Tree/blob/master/Region_tree.py.

[2] Tao, Y. F. (n.d.). *The R-tree - CUHK CSE*. Retrieved November 17, 2021, from https://www.cse.cuhk.edu.hk/~taoyf/course/infs4205/lec/rtree.pdf.

[3] Adler, M., & Heeringa, B. (n.d.). *Search space reductions for nearest-neighbor queries*. Retrieved November 17, 2021, from http://www.cs.williams.edu/~heeringa/publications/heeringa-tamc.pdf.

[4] Roussopoulos, N., Kelley, S., & Vincent, F. (n.d.). *Nearest neigh B or queries - UMD*. Retrieved November 17, 2021, from http://www.cs.umd.edu/~nick/papers/nnpaper.pdf.