MSBD6000J Spatial and Multimedia Databases

Assignment 2 Report

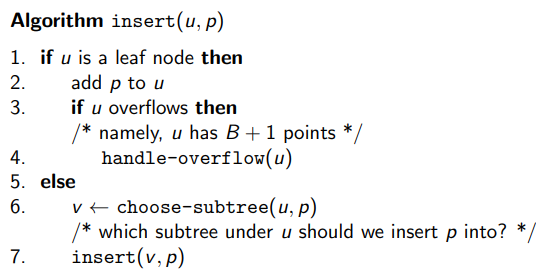
Task

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.

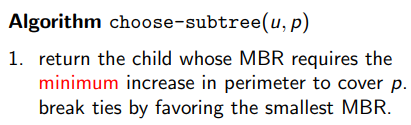
Basically, the construction of R-Tree starts from insert function.

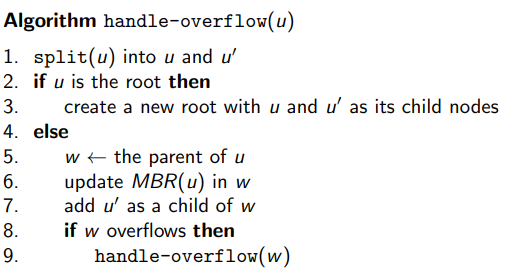


There are two overflow conditions of the node

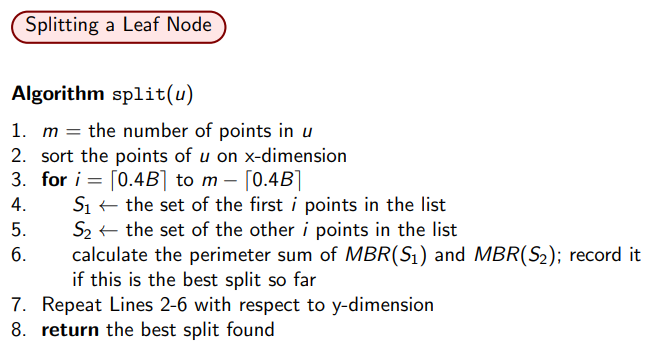
1. Data points > max bucket size (b) (Leaf node)
2. Child nodes > Sub Tree (d) (Non-Leaf node)

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

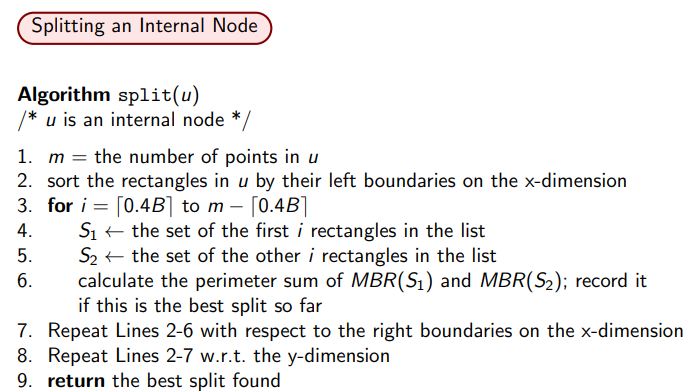


The way of overflow handling.

Splitting a leaf node when overflow happen



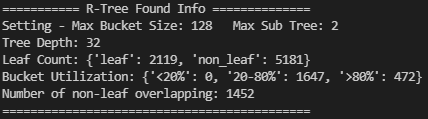
Splitting a non-node when overflow happen



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.

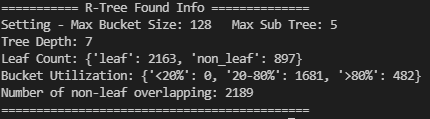
Task 1 (3) – Result of the constructed R-Tree

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



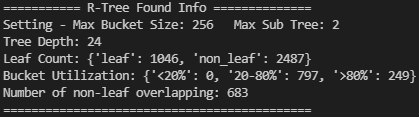
There is a tree depth (height) 32, 2119 leaf node and 5181 non-leaf nodes. For each bucket in leaf node containing points within n, there is no bucket under 20%, 1647 within 20%-80%, and 472 over 80%. And there are 1452 pairs of non-leaf node have an overlapping region.

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



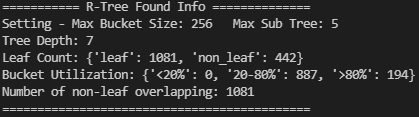
There is a tree depth (height) 7, 2163 leaf node and 897 non-leaf nodes. For each bucket in leaf node containing points within n, there is no bucket under 20%, 1681 within 20%-80%, and 482 over 80%. And there are 2189 pairs of non-leaf node have an overlapping region.

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



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.

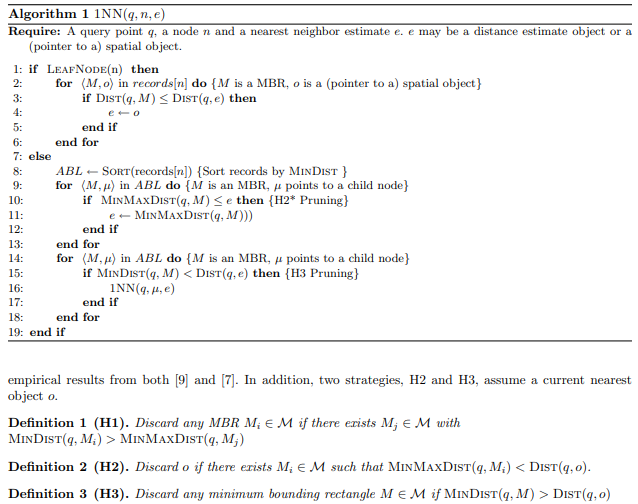
R Tree setting (n=256, d=5)



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.

Task 2 (2) – Implementation of nearest neighbor search

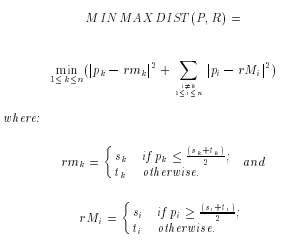
For nearest neighbor search, I follow the implementation from Search Space Reductions for Nearest-Neighbor Queries (TAMC, 2008) [3]. The code follows exactly what written in paper with the three pruning rules. The screenshot from this section is from the papers.



Clarify in pseudocode

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.

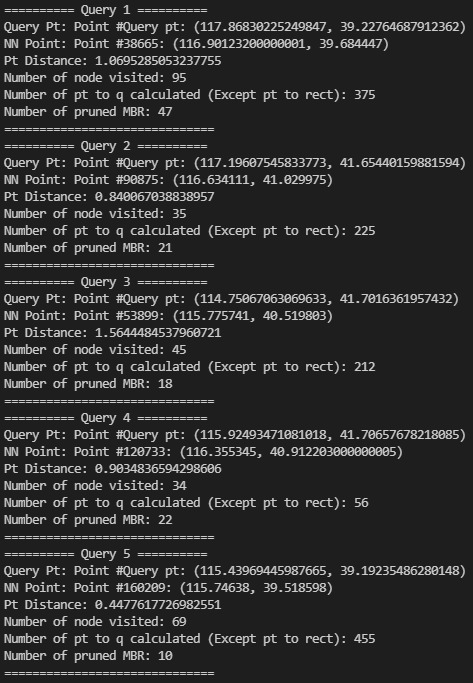
ABL Issue – ABL should have sorted with records n. It is constructed by getting all instance in child node and compute its related MBR minimum distance, and put the data into ABL list, which contain tuple (node, min\_dist). Then for the rest of looping, the code loops the tuple (node, min\_dist) in the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

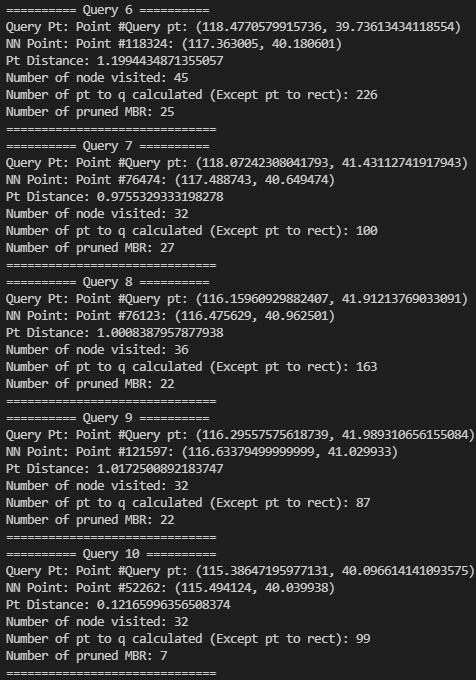
Minimax distance issue – It computed as follows, retrieved from Nearest Neighbor Queries by Nick Roussopoulos[4].

Task 2 (3) –Nearest Neighbor Search Result

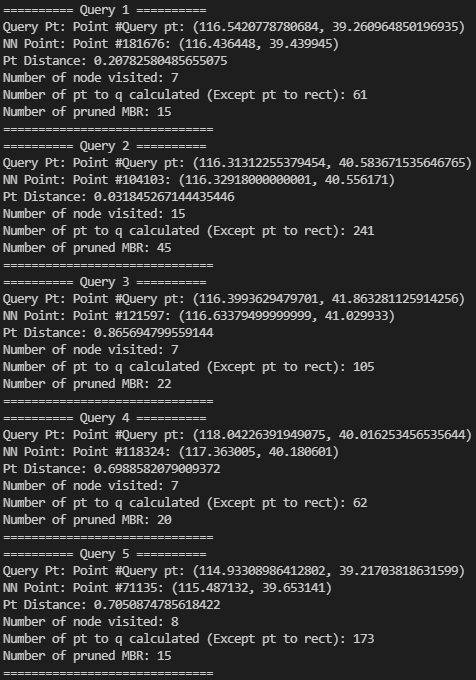
The query points are randomly generated the minimum and maximum value in dataset from min -1 to max + 1

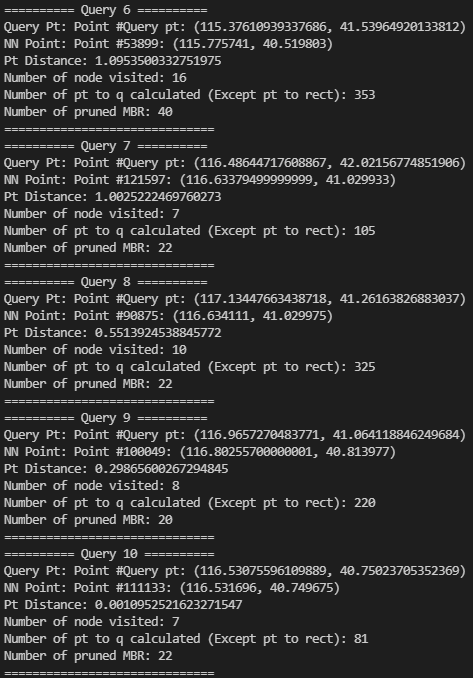
R Tree setting (n=128, d=2)



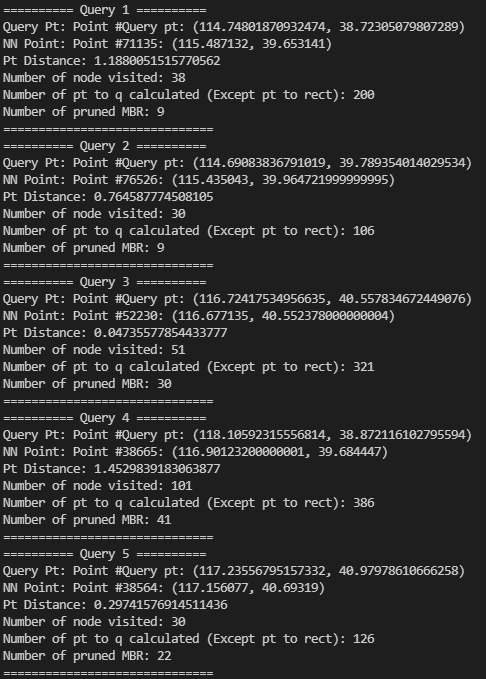


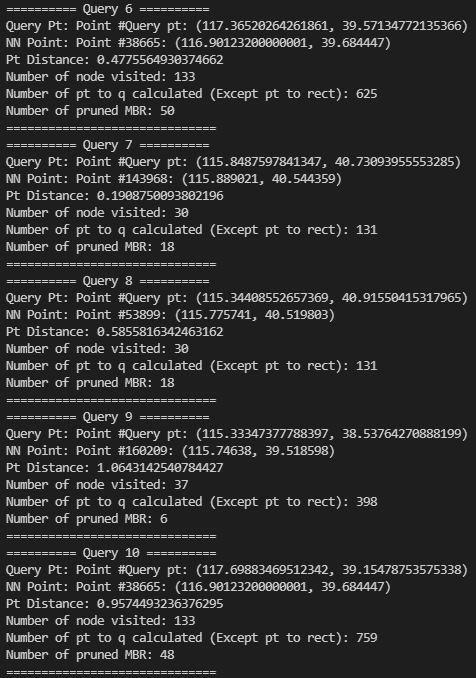
R Tree setting (n=128, d=5)



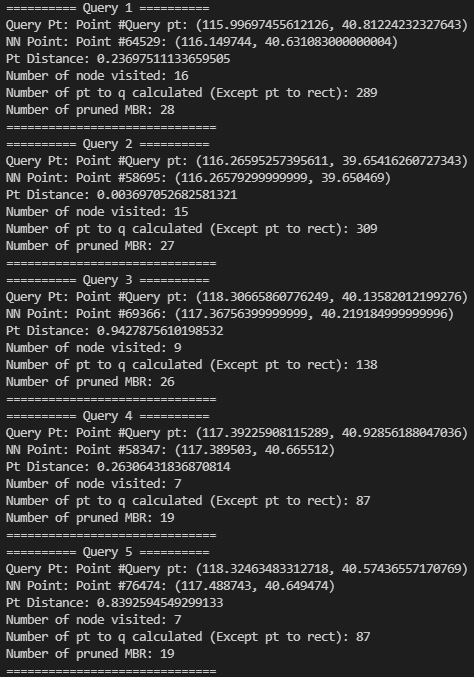


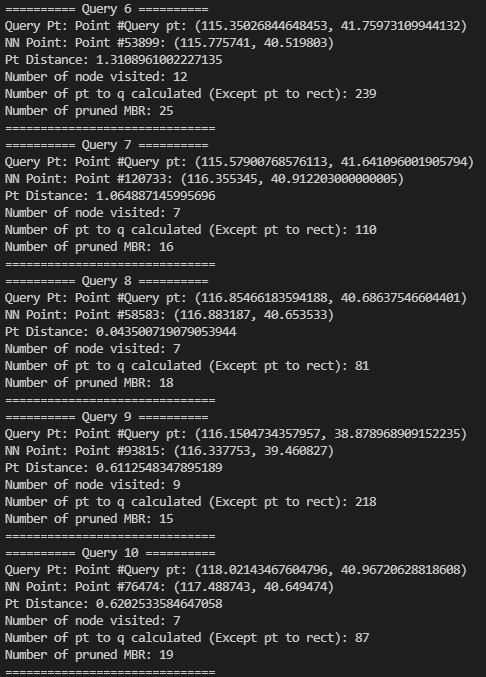
R Tree setting (n=156, d=2)





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

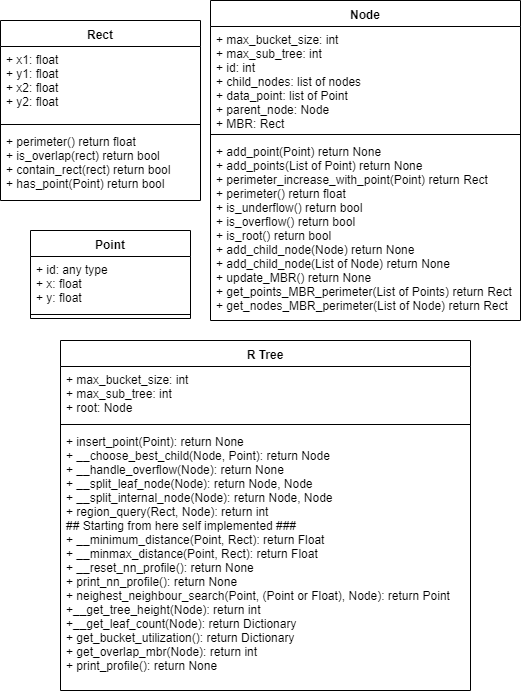




For number of MBR pruned, it only counts the pruned in same node. It does not count the pruned sub-tree, or children MBR.

Appendix

UML Class Diagram for the program structure as reference.



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.