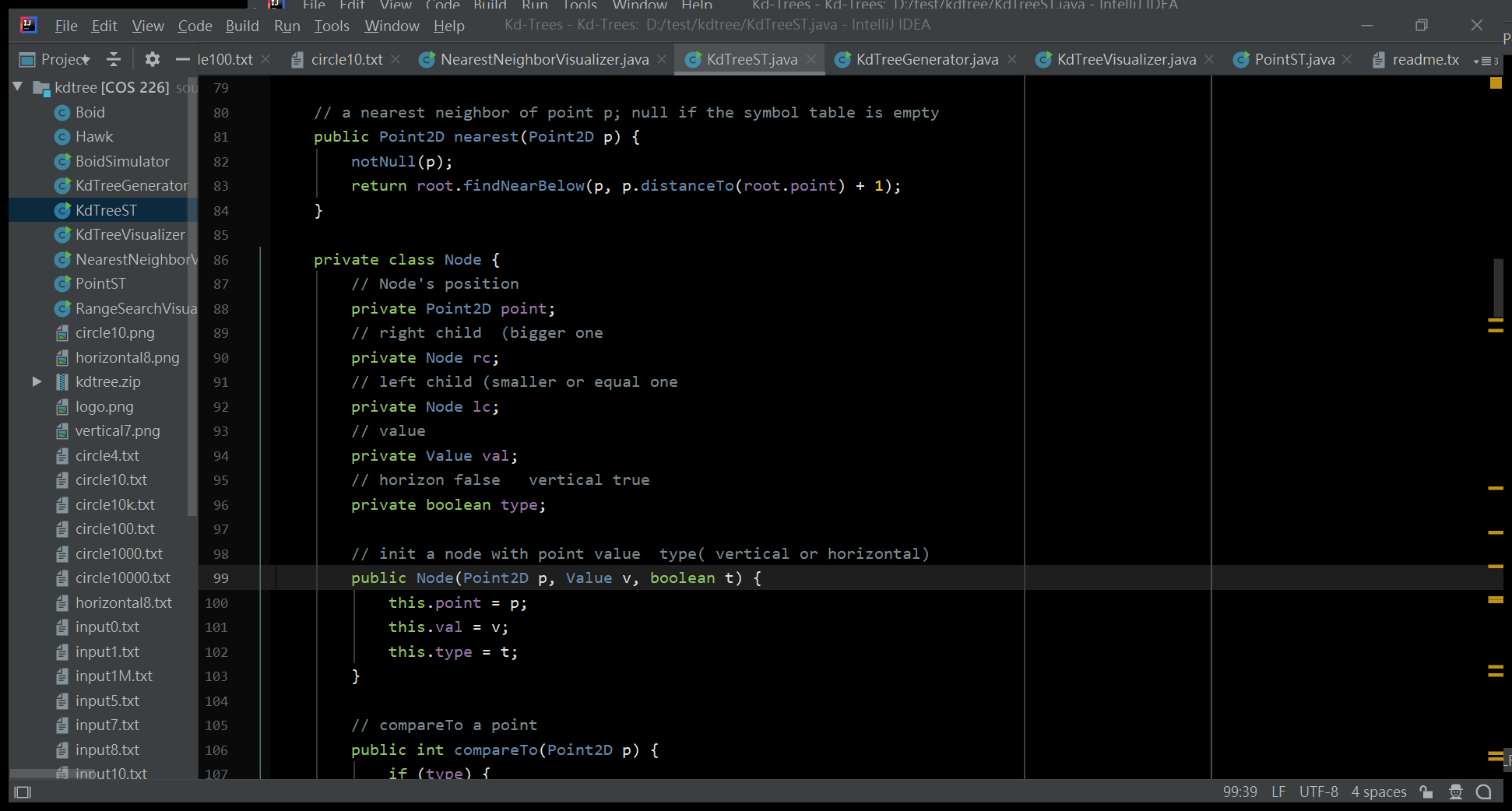
KD\_tree B06505004 莊博翰

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Describe the Node data type you used to implement the

\* 2d-tree data structure.



如同一般的binary search tree 會有right child ,left child

同時要記錄這個node 的實際位置 Point,還有儲存的值val

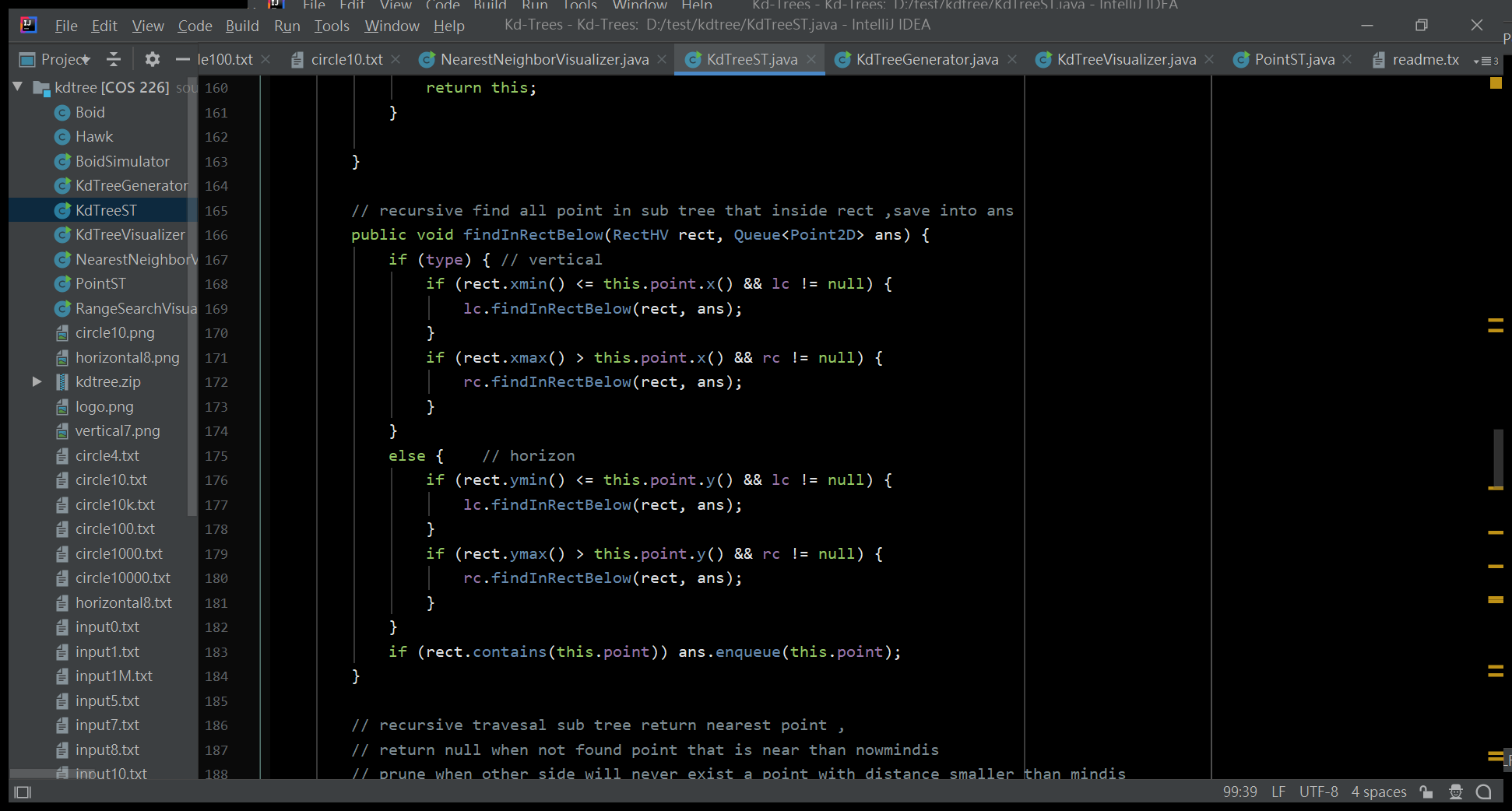
最後，用一個booean 的type 來表示這個node 是水平切割還是垂直切割

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Describe your method for range search in a kd-tree.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

對於range search 我先implement 了一個Node 的method



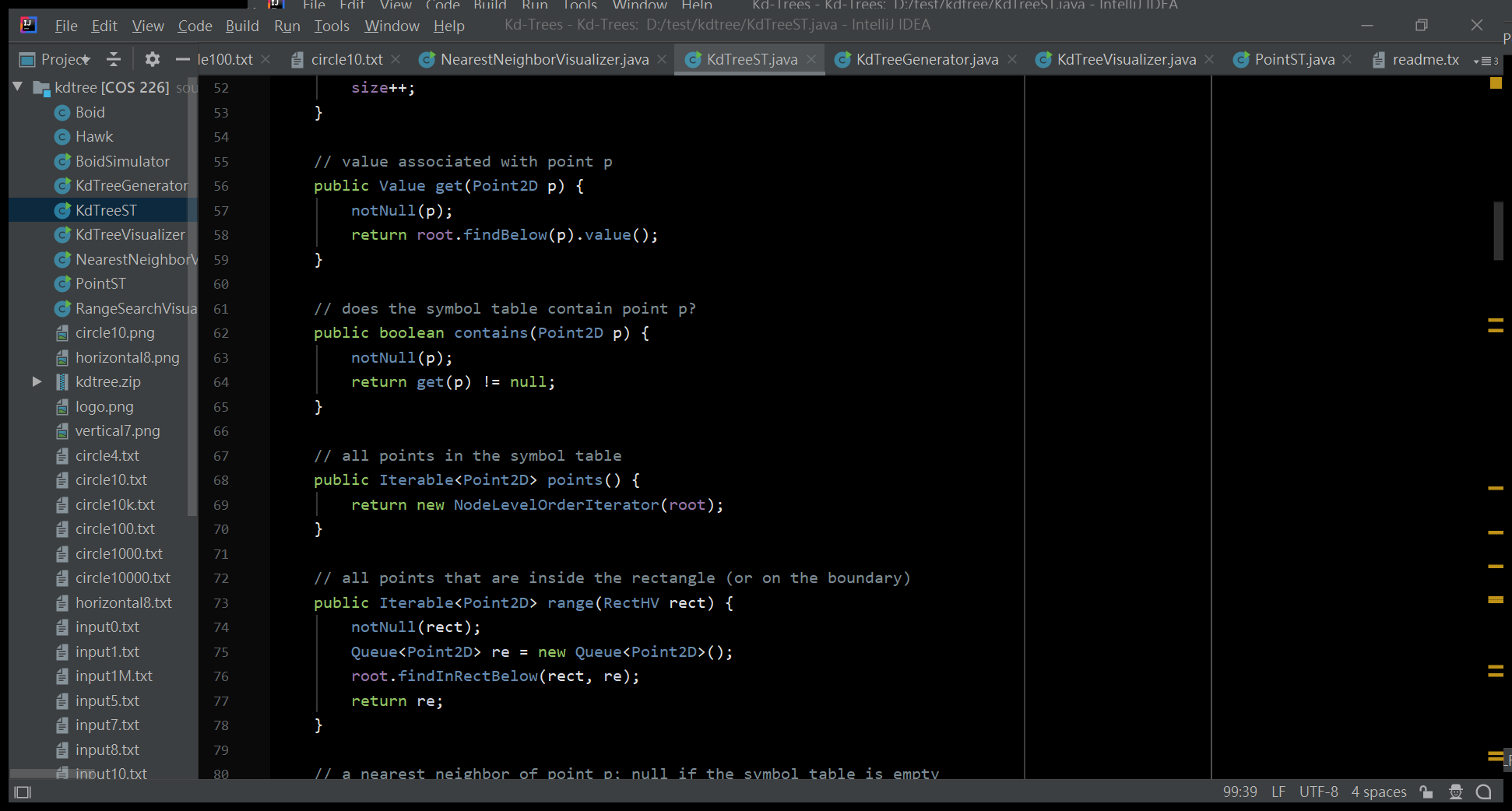
這個method 會recursive去尋找這個node 以下的sub tree 有沒有點

在range 中，找到了就把它放入Queue 中，直到right child, left child == null

當然會有一些cut off 的條件，比如rect 完全在point 的左側

此時就不必traverse Node 的right child 了，以此類推

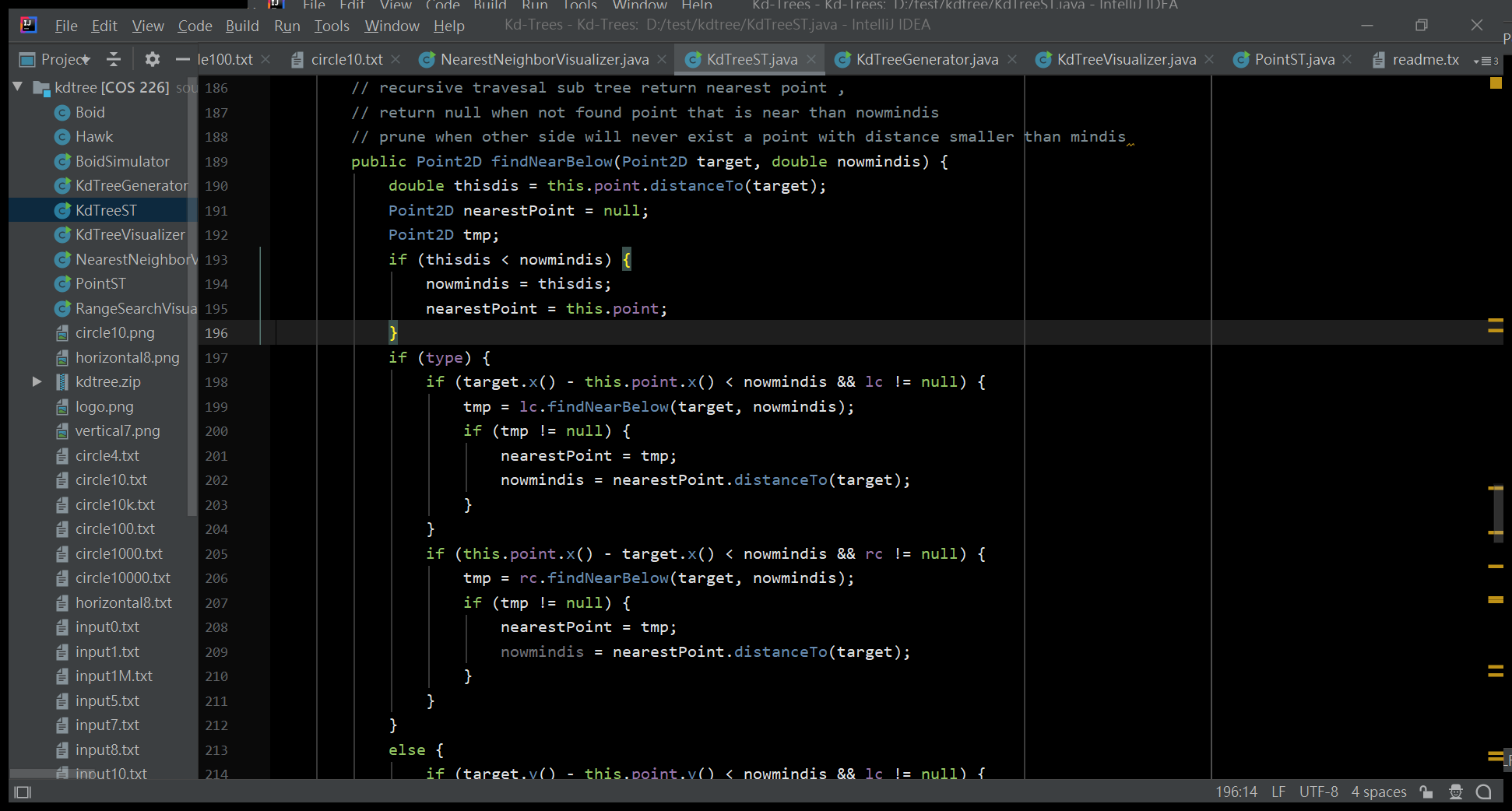
然後時實際在 range()中 我們只需要call root.findRectBelow()



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Describe your method for nearest neighbor search in a kd-tree.

同樣的，先implement 一個 Node 的method



首先會按照type 分成兩種case( vertical , horizon)，然後

recursive去尋找這個node 以下的sub tree 有沒有點比現在的mindis還小

找不到的話就return null，還有以些cut off 的條件如:

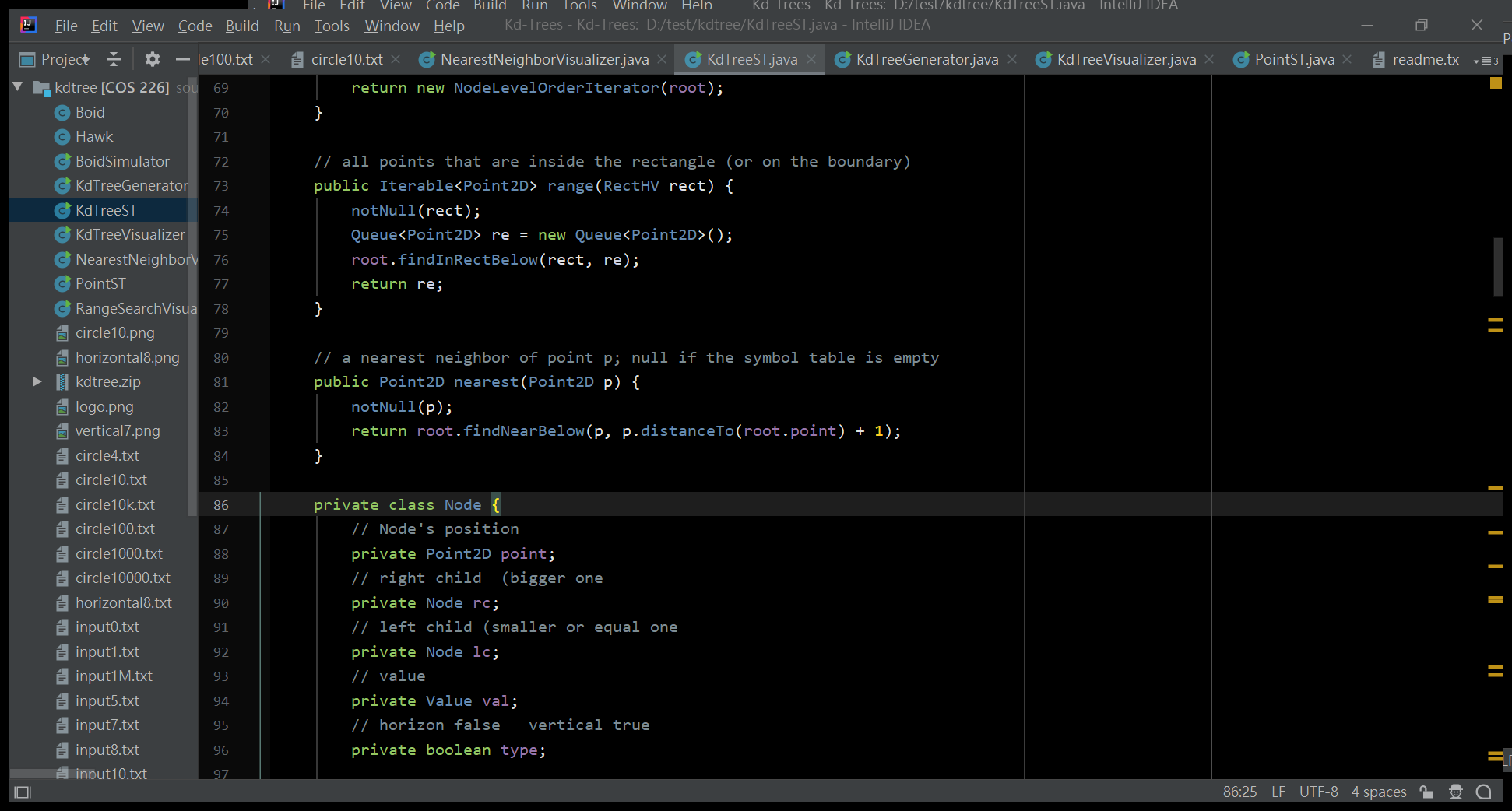
目前mindis 目前走到的點

目標 與分隔線距離

如果現在走到的node 的分隔線離目標的距離大於現在的min distance

沒有必要traverse right child，其他方向以此類推。

有了這個method 後 nearest()就簡單了



從root 開始findNearBelow 即可。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* How many nearest-neighbor calculations can your PointST implementation

\* perform per second for input1M.txt (1 million points), where the query

\* points are random points in the unit square?

\*

\* Fill in the table below, using one digit after the decimal point

\* for each entry. Use at least 1 second of CPU time.

\* (Do not count the time to read the points or to build the 2d-tree.)

\*

\* Repeat the same question but with your KdTreeST implementation.

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

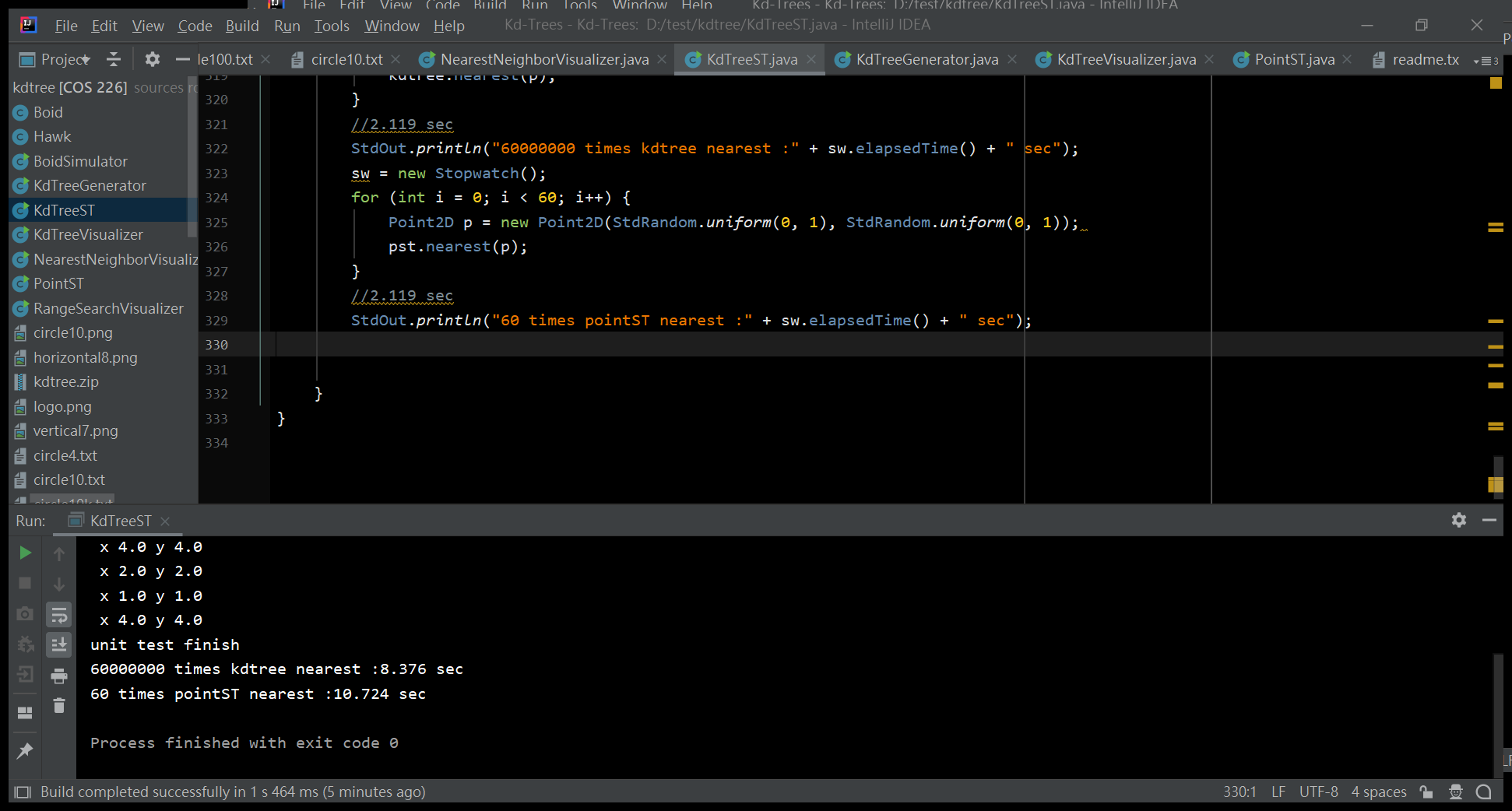
# calls to / CPU time = # calls to nearest()

client nearest() (seconds) per second

------------------------------------------------------

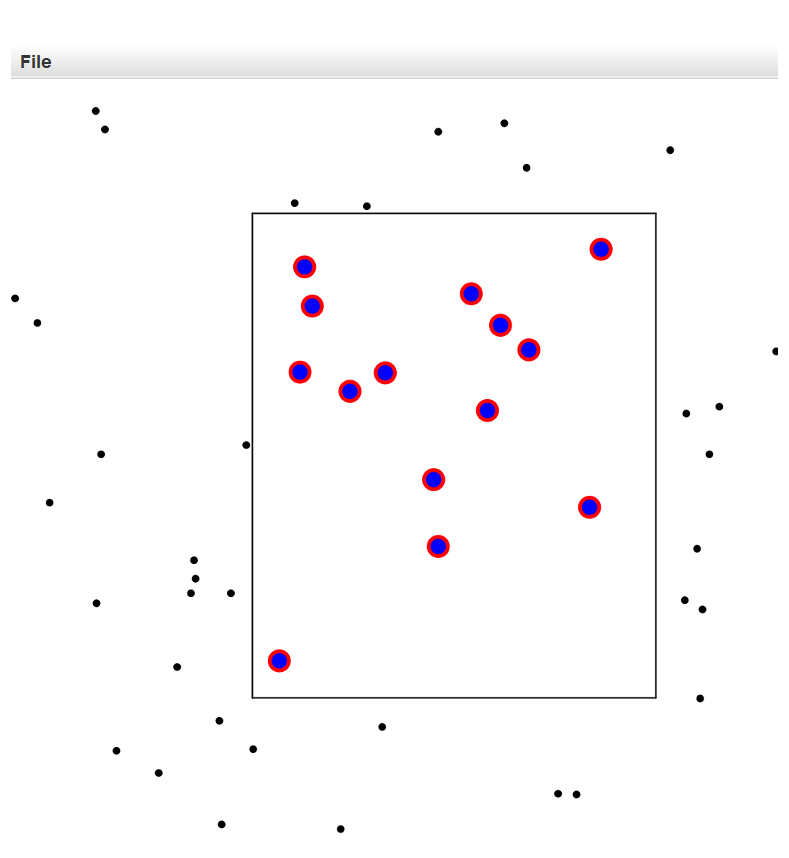
PointST: 60 / 10.72 = 5.5/sec

KdTreeST: 60000000 / 8.37= 7.1\*10^6 /sec



Appendix

執行結果截圖

input50.txt