8NN search

the first run:

tree max depth: 8
40392 - 0.02
22068 - 0.03
43343 - 0.04
50976 - 0.04
41217 - 0.05
18088 - 0.05
59128 - 0.05
9813 - 0.06
In total 15 comparison operations.
[40392 22068 43343 50976 41217 18088 59128 9813]
[0.02091988 0.02517117 0.04114122 0.04275282 0.05267616 0.05394018
 0.05437505 0.05530288]
Radius search normal:
Search takes 18457.615ms(Note: I change the number of iteration from 100 to 20 for both cases)

Radius search fast:
Search takes 18630.866ms

the second run:

tree max depth: 8
29542 - 0.03
41052 - 0.03
3883 - 0.03
2955 - 0.05
56861 - 0.05
14669 - 0.05
50890 - 0.06
6232 - 0.06
In total 19 comparison operations.
[29542 41052 3883 2955 56861 14669 50890 6232]
[0.02954144 0.03301189 0.03342762 0.04565899 0.04709097 0.05047586
 0.05654842 0.05932954]
Radius search normal:
Search takes 20788.574ms

Radius search fast:
Search takes 16836.644ms

Question:
Why is the radius search slower than the normal one?

octree ----------
Octree: build 7.940, knn 2.276, radius 1.901, brute 2.257
kdtree ----------
Kdtree: build 0.093, knn 0.840, radius 0.501, brute 1.648
scipy spatial kdtree ----------
Kdtree: build 6.007, knn 10.804, radius 12.121, brute 1.861

Question:
Why is the time of the last two brute methods faster than the first brute method?

```
def knn_search(root: Node, db: np.ndarray, result_set: KNNResultSet, query: np.ndarray):
    if root is None:
        return False

    if root.is_leaf():
        # compare the contents of a leaf
        leaf_points = db[root.point_indices, :]
        diff = np.linalg.norm(np.expand_dims(query, 0) - leaf_points, axis=1)
        for i in range(diff.shape[0]):
            result_set.add_point(diff[i], root.point_indices[i])
        return False

    if query[root.axis] <= root.value:
        knn_search(root.left, db, result_set, query)
        if math.fabs(query[root.axis] - root.value) < result_set.worstDist():
            knn_search(root.right, db, result_set, query)
    else:
        knn_search(root.right, db, result_set, query)
        if math.fabs(query[root.axis] - root.value) < result_set.worstDist():
            knn_search(root.left, db, result_set, query)

    return False
```

Compare query to every point inside the leaf, put into the result set

q[axis] inside the partition

|q[axis] – splitting_value| < w

Question:
Is the above algorithm correct? All cases would return False…….