**Performance Test Report**

**Project name: Nearest Neighbor Search**

**Executive summary**

In this program, I use scipy cKTree implementation for nearest neighbor search since it is the fastest nearst neighbor implement for both tree construction and queries. The detail of KDTree data structure can be found in (Maneewongvatana, S., & Mount, D.M. 1999). The implementation is based on scipy.spatial.cKDTree, pytorch tensor.

**Performance Testing**

1. System Configuration

|  |  |
| --- | --- |
| OS | macOS |
| CPU | Intel core i7 |
| #CPU | 2 |
| #core | 2 |
| RAM | 4GB |

1. Assumptions:

* For scipy KDTree.query function, I use default values except clearly defined.
* I also use default values for percentile to query data points within given percentage. numpy.percentile(a, q, axis=None, out=None, overwrite\_input=False, method='linear', keepdims=False, \*, interpolation=None)

1. Performance results

In theory, building kdtree is O(N) time and query is O[NlogN].

To prove it, I did performance test to my program. The results are described in the following Table I and Table II.

**Table I: Time and Memory Consumption for Building Tree**

|  |  |  |
| --- | --- | --- |
|  | coord | cen |
| time | 1208.308 | 6.207 |
| mem c | 24146833 | 30425 |
| mem p | 32161595 | 322845 |

**Table II: Time and Memory Consumption for Tree Query**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | radius = 5 | radius=10 | 80% | c limit = 1000 |
| time | 2973 | 3740 | 1047 | 223.377 |
| mem c | 6407 | 6407 | 16008440 | 27273 |
| mem p | 18990816 | 26185704 | 16009788 | 28541 |

1. Conclusion

From table 1, we can say that the more data points, the more time and memory usage increase. Accordingly, time and memory consumption increased if search range becomes large. However, performing rigouriously testing will be future step for this program.

References:

Maneewongvatana, S., & Mount, D.M. (1999). Analysis of approximate nearest neighbor searching with clustered point sets. *Data Structures, Near Neighbor Searches, and Methodology*.