Mission: KdTree

Balaji Cherukuri balusoft@gmail.com

Requirements

1. C++ APIs:
   a. Insert: Ability to insert given points into KdTree.
   b. findNearestNeighbor: Find nearest neighbor by using KdTree.
   c. Must allow customized partition.
2. Build_kdtree:
   a. Takes input points as CSV file path name and output tree file name.
   b. Generates KdTree in “output-tree-file”.
3. Query_kdtree:
   a. Takes input as “tree-file”, “query-points” and output “query-results” into file.
   b. Generates query results in CSV file as “nearest-point, distance”(for every point in query)

Non-requirements/Assumptions

1. No need to support dynamic insert. All input points are provided ahead.
2. No need to support remove point.
3. To make insert simplify if leaf-node has already point append to list. In my limited testing leaf node at most has two points.

Resources Used

1. http://en.cppreference.com/w/
2. Cxx Unit Test Guide: http://cxxtest.com/guide.html
3. Valgrind: http://valgrind.org/

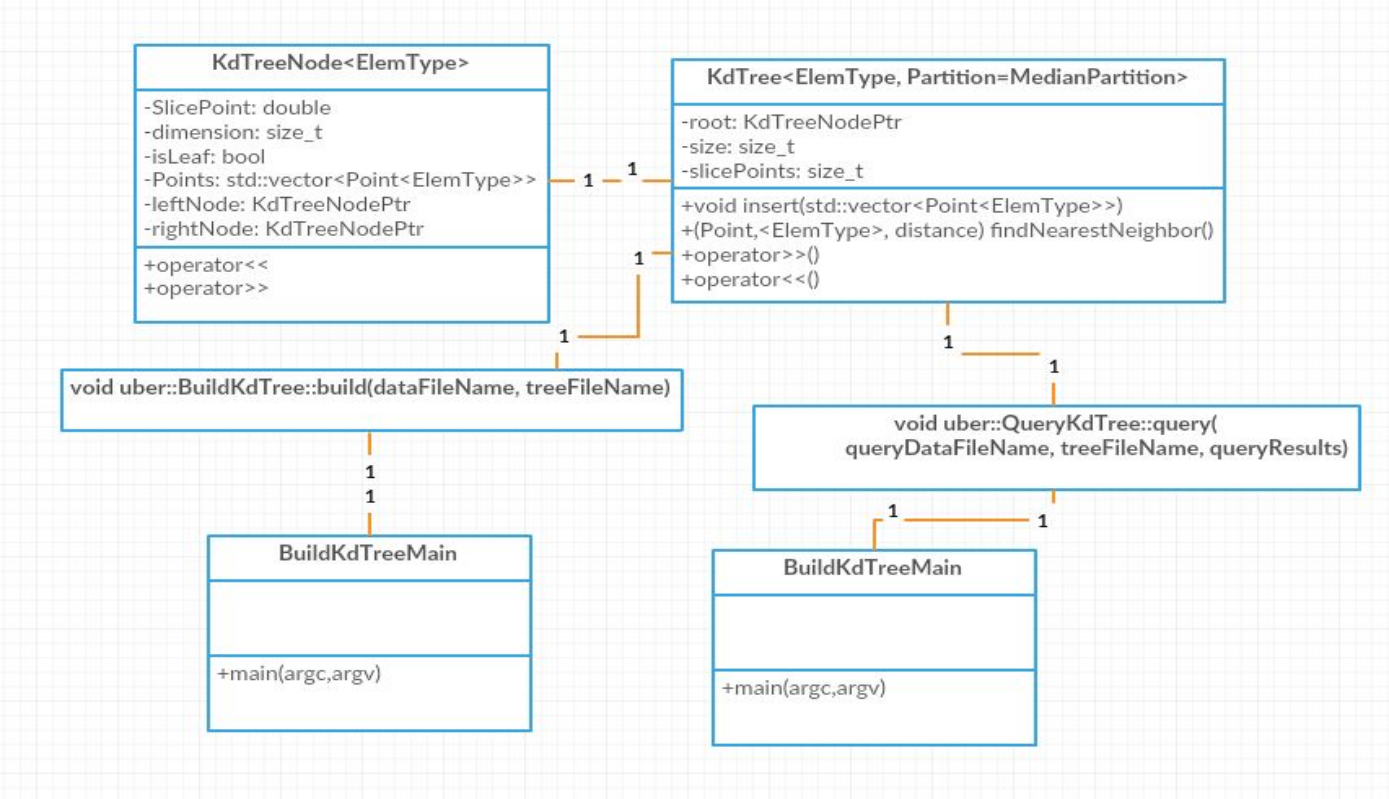
Design and implementation choices

1. Templated implementation to allow all numeric values.
2. Assume build_kdtree and query_kdtree need to work against “double” data.
3. Use STL for most of the algorithms like finding median.
4. Use operator >> and << for serialization of tree.
5. Use Make for build
6. Use g++ and clang-3.6 as compiler(Tests should pass with both binaries)

Quality

1. Unit Test all source code by using CxxTest automation.
2. Run all test binaries with valgrind and check for memory errors. (like leaks, invalid-reads etc)
3. Manually test applications

Class Diagram



How to build?

```
balajic:~balajic@balajic-VirtualBox:~/Documents/fun-run/Uber/TakeHomeTest/KDTree$ ls
cxxtest-4.3  data  Makefile  README  run.sh  src
balajic@balajic-VirtualBox:~/Documents/fun-run/Uber/TakeHomeTest/KDTree$ make
mkdir -p ./gen
./cxxtest-4.3/bin/cxxtestgen --error-printer -o gen/GenTestKdTree.cc src/tests/lib/TestKdTree.cc
mkdir -p ./bin
/usr/bin/g++ -I ./src -I ./cxxtest-4.3/ -g -Werror -std=c++11 -o bin/test_kdtree \
                    gen/GenTestKdTree.cc
...
mkdir -p ./bin
/usr/bin/g++ -I ./src -I ./cxxtest-4.3/ -g -Werror -std=c++11 -o bin/build_kdtree \
                     src/build_kdtree/BuildKdTree.cc src/build_kdtree/BuildKdTreeMain.cc
mkdir -p ./bin
/usr/bin/g++ -I ./src -I ./cxxtest-4.3/ -g -Werror -std=c++11 -o bin/query_kdtree \
                     src/query_kdtree/QueryKdTree.cc src/query_kdtree/QueryKdTreeMain.cc

balajic@balajic-VirtualBox:~/Documents/fun-run/Uber/TakeHomeTest/KDTree$ ls bin/
build_kdtree  query_kdtree  test_build_kdtree  test_kdtree  test_query_kdtree
```

Build_kdtree and query_kdtree

1) Generate tree

```
balajic@balajic-VirtualBox:~/Documents/fun-run/Uber/TakeHomeTest/KDTree$ ./bin/build_kdtree
usage: ./bin/build_kdtree <data-file-path> <tree-file-path>

balajic@balajic-VirtualBox:~/Documents/fun-run/Uber/TakeHomeTest/KDTree$ ./bin/build_kdtree ./data/kdtree_v5/sample_data.csv .
/gen/BalajiTree.tree
balajic@balajic-VirtualBox:~/Documents/fun-run/Uber/TakeHomeTest/KDTree$ wc ./gen/BalajiTree.tree
   0     1 60139 ./gen/BalajiTree.tree
```

2) query tree

```
balajic@balajic-VirtualBox:~/Documents/fun-run/Uber/TakeHomeTest/KDTree$ bin/query_kdtree
usage: bin/query_kdtree <query-data-file-path> <tree-file-path> <query-result-file-path>
balajic@balajic-VirtualBox:~/Documents/fun-run/Uber/TakeHomeTest/KDTree$ bin/query_kdtree data/kdtree_v5/query_data.csv
gen/BalajiTree.tree gen/BalajiQueryResults.csv
balajic@balajic-VirtualBox:~/Documents/fun-run/Uber/TakeHomeTest/KDTree$ tail -3 gen/BalajiQueryResults.csv
[0.8851,0.848954,0.941256],0.123173
[0.1849,0.649294,0.680832],0.0488941
[0.702284,0.806144,0.50311],0.0671074
```

What could be next?

1) Quality:
    a) Generate code coverage for all tests.
    b) Automated application testing.
    c) Generate performance numbers for tree generation and finding neighbors.
    d) This work is short of reviews(design, code, test), with reviews could have caught bugs residing in this code.
2) IO errors:Better handle IO errors for applications. As of now asserting on IO failures.
3) Insert enhancements:
    a) As of now, when points are not well distributed one leaf node could be having more than one point as a vector. Rather we could make insert algorithm to expand tree organically.
    b) Allow single point insert, this should change “partition” nodes distribution dynamically.
4) Support for remove:
    a) To keep KdTree up to date we could support remove operation. For example Uber drivers could go offline for variety of reasons in that case tree should be updated accordingly.
5) Build:
    a) Create libraries rather than recompiling .cc file for every target.
    b) Explore using CMakefile or any other modern build systems

Thank you