Nearest State/County Finder

Problem Description

We are given a large file containing US reference points with their respective state, county, latitude and longitude. The user should be able to input coordinates to a point as well as the number of nearest neighbors he would like. We are to return the K ($1 \leq K \leq 10$) nearest points to the input point, as well as perform a majority vote among the 5 nearest points to find the state and county of the input point. The use of an efficient data structure for the problem is important to return the result as fast as possible. We can compute the distance between two points using the "equirectangular approximation" where Φ is the latitude, λ is the longitude, and R is earth's radius (6371km).

$$x = (\lambda 2 - \lambda 1) * \cos((\phi 1 + \phi 2)/2);$$
$$y = (\phi 2 - \phi 1);$$
$$\text{Distance} = \sqrt{x*x + y*y} * R;$$

Background

A k-d tree is a k-dimensional tree (not to be confused with k the number of nearest points we want). It is similar to a binary search tree in the sense that the left child has a key smaller than the parent's and the right child has a key larger than the parent's, except since the keys in this case are multidimensional, we need to specify the dimension of comparison. Let cd the current dimension of comparison where cd = depth%k, for example in a 2d case, the root will compare the first dimension, root's children will compare the second dimension, and their children will compare the first dimension…

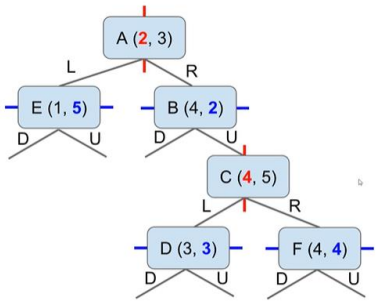

Fig 1: Example of 2d k-d tree

As seen if figure 1, A is the root and compares the x dimension, so E is on the left because it has x=1 and B is on the right because it has x=4. E and B then use y as the current dimension of comparison.

Since each node splits the hyperplane, when looking for a point, you can first look into the side of the hyperplane closest to the point, and then only look at the other side if it is possible that a point closer than the best point lives there.

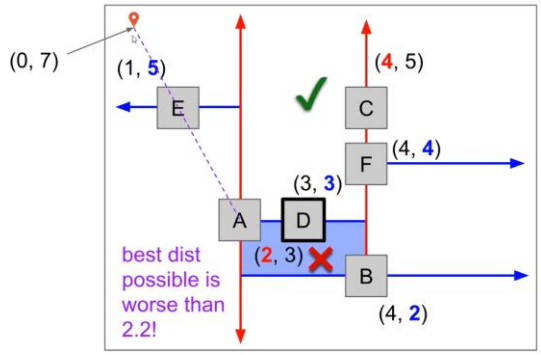

Fig 2: Example of pruning

An example of pruning is shown in figure 2. D splits the hyperplane horizontally since it compares y, the two sides are therefore “up” above D and “down” below D. Given the input (0,7), we can see that up is the closer side. After traversing all of “up”, we consider what the best possible point in “down” could be, and calculate the distance to it from (0,7). Since the distance is not better than our current best, we don’t go down and avoid calculating distances of all points D could have in its subtree.

Implementation

The input file is read line by line, each being put in a Point structure that contains the State, Name, Latitude and Longitude. The first implementation is a naïve array of Points implementation where the search for one nearest neighbor requires traversing the entire array and computing the distance for each point, keeping track of the point with the smallest distance; that implementation would have a search complexity of O(n).

The second and better implementation uses a 2 dimensional (latitude and longitude) k-d tree. A Node structure contains the Point structure, as well as two pointers to Nodes for left and right children. The tree is built by adding the Nodes one by one, each level of depth has a different dimension of comparison (latitude or longitude). The search for one nearest neighbor requires traversing down the tree starting from the root node, looking at the “good” child node, and then only going down the “bad” child node if the best possible point that could be there would be better than largest best distance. This allows to prune subtrees and skipping the distance calculation for all the points inside. This leads to a search complexity of O(log n) (provided the tree is well balanced). To handle cases where k is larger than 1, we keep an array of size k that has the best distances yet, replacing the largest one if we find a better point.

Once we have the k nearest points, we perform a required vote of five if k is larger or equal than 5, otherwise we use case by case to determine (if possible) the state and county of the input point.

Features

- User can use command line interface to query the k nearest neighbors of an input point
- Speed of search

Results

The k-d tree implementation is faster than the array implementation by a factor of around 10 for this input size (2.2 million points). On average looking for the 10 nearest points takes 236ms for the array and 21ms for the k-d tree.

```
Welcome to "Nearest State County Finder"
Loading input file
How many nearest neighbors do you want? Enter 1<K<10: 10
Please Enter the Latitude of your point: 12
Please Enter the Longitude of your point: 14

You have chosen to find 10 nearest neighbors to the point 12, 14.

Search by array took 222 milliseconds
Closest point #1 is 8467.536853004 km away, at 17.7535846, -64.5662484 at St.Croix, VI.
Closest point #2 is 8468.039831231 km away, at 17.7538623, -64.5709709 at St.Croix, VI.
Closest point #3 is 8467.79970267 km away, at 17.7513623, -64.5684707 at St.Croix, VI.
Closest point #4 is 8468.29650467 km away, at 17.7521957, -64.5731932 at St.Croix, VI.
Closest point #5 is 8466.03300707 km away, at 17.7549734, -64.5523588 at St.Croix, VI.
Closest point #6 is 8467.458790276 km away, at 17.7552512, -64.5656928 at St.Croix, VI.
Closest point #7 is 8468.569477275 km away, at 17.7544179, -64.5759711 at St.Croix, VI.
Closest point #8 is 8468.178546837 km away, at 17.749418, -64.5718042 at St.Croix, VI.
Closest point #9 is 8468.163101031 km away, at 17.7508069, -64.5718042 at St.Croix, VI.
Closest point #10 is 8468.326032345 km away, at 17.7602512, -64.5743044 at St.Croix, VI.

Search by Kd-tree took 19 milliseconds
Closest point #1 is 8468.659071664 km away, at 17.7463624, -64.5759711 at St.Croix, VI.
Closest point #2 is 8468.039831231 km away, at 17.7538623, -64.5709709 at St.Croix, VI.
Closest point #3 is 8468.326032345 km away, at 17.7602512, -64.5743044 at St.Croix, VI.
Closest point #4 is 8467.79970267 km away, at 17.7513623, -64.5684707 at St.Croix, VI.
Closest point #5 is 8468.178546837 km away, at 17.749418, -64.5718042 at St.Croix, VI.
Closest point #6 is 8467.536853004 km away, at 17.7535846, -64.5662484 at St.Croix, VI.
Closest point #7 is 8468.569477275 km away, at 17.7544179, -64.5759711 at St.Croix, VI.
Closest point #8 is 8468.739693735 km away, at 17.760529, -64.5781935 at St.Croix, VI.
Closest point #9 is 8467.458790276 km away, at 17.7552512, -64.5656928 at St.Croix, VI.
Closest point #10 is 8466.03300707 km away, at 17.7549734, -64.5523588 at St.Croix, VI.


By consensus of 5 closest points, State is VI and county is St.Croix
```

Files

- **Inputfile.txt** (Loaded by Nearest.cpp, contains LatLong coordinates of 2.190.113 points
- **Nearest.cpp** (Main file)

- **makefile**
- **Report.pdf** (This file)

Instructions

1. Run the makefile

   ```
   $ Make
   ```

2. Run the executable

   ```
   $ ./Nearest
   ```

3. Input k the number of nearest neighbors you want
4. Enter Latitude
5. Enter Longitude