University of Washington
Department of Computer Science and Engineering
CSEP 521, Winter 2021

February 16, 2021

Homework 7, Due Thursday, February 25, 2021

Problem 1 (10 points):

Argue that it is impossible to construct a 20-bit binary code with 5000 code words that can correct two errors.

Problem 2 (15 points):

Consider the following set of points with the given coordinates: $S = \{(0,2),(1,0),(1,4),(3,3),(4,2)\}$. Draw the Voronoi diagrams for the set of points using the L_2 norm, the L_1 norm, and the L_∞ norms. You should draw the diagrams on paper, by hand. While these will not be graded on perfect accuracy, you should aim to have relatively neat diagrams, instead of free hand sketches. (Graph paper and a straight edge is recommended.) The key to the L_1 and L_∞ cases is to understand what the set of points that are equidistant from points p_1 and p_2 looks like under these two metrics. Start out by building the the Voronoi diagrams for $\{(0,2),(1,0)\}$.

a) Draw the Voronoi diagram for the point set $S = \{(0,2),(1,0),(1,4),(3,3),(4,2)\}$ under the L_2 norm, Euclidean distance. For points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$,

$$d_2(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

b) Draw the Voronoi diagram for the point set $S = \{(0,2),(1,0),(1,4),(3,3),(4,2)\}$ under the L_1 norm, Manhattan distance. For points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$,

$$d_1(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|.$$

c) Draw the Voronoi diagram for the point set $S = \{(0,2),(1,0),(1,4),(3,3),(4,2)\}$ under the L_∞ norm. For points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$,

$$d_\infty(p_1, p_2) = \max(|x_1 - x_2|, |y_1 - y_2|).$$

Programming Problem 3 (10 points):

Implement a quad tree data structure. The version of the quad tree we will consider has a square associated with each node, and the four children of a node correspond to a split of the square into quadrants defined by the central point. The quad tree should subdivide the point set until there is at most one point in a region. The depth of the leaves of a tree will vary, based on the distribution of points.

For your implementation, you should generate test data of n random points in the unit square, so the region at the top level should be the square with corners $\{(0.0, 0.0), (1.0, 0.0), (0.0, 1.0), (1.0, 1.0)\}$. You should use floating point numbers for coordinates.

For results, the most convenient format would be to draw the quad tree partition and point set for a few sample input sizes, maybe one quite small ($n = 10$), and a larger one, $n = 200$. For submitting results, there is no need for large plots. If you decide not to draw this - generate some statistics of the trees, such as the number of nodes at each level.

There is obviously lots of quad-tree code in the wild - the expectation is that you use references to understand the algorithm, and then write the algorithmic code yourself. You can, of course, use libraries and other code for things like the graphics component.

Programming Problem 4 (10 points):

Implement the nearest neighbor search using the quad tree structure from the previous problem. You should generate a random set of points, and then determine which of these points is closest to the query point. Your search algorithm should be a tree based search that runs in time proportional to the depth of the tree.

For submitting results, you can either generate program traces to show which comparisons are done to determine the nearest neighbor, or augment the visualization from problem 3, to show the query point and its nearest neighbor. For problem 4, you can just give the code for your search method separately.