CMPS 6130 Computational Geometry – Spring 20

4/7/20

9. Homework (grad)

Due **4/23/20** before class

Please justify all your answers. Often it helps to draw pictures.

1. **Fractional Cascading (10 points)**
   Read slides 19-23 of the Orthogonal Range Searching I slides. These slides describe fractional cascading, which is an array-based implementation of a 2D range tree ("layered range tree"), that enables $O(\log n + k)$ query time (as opposed to $O(\log^2 n + k)$ query time for a regular 2D range tree).

   Give pseudocode for a two-dimensional range query in a layered range tree that uses fractional cascading. You only need to consider the left side of the query by x-range after the split node, i.e., you only need to modify the code on slide 11 of the slides. Make sure to describe the attributes in the tree nodes that you are using in your code (such as $v.A$ for the array stored in node v).

2. **KD-trees (12 points)**
   (a) (5 points) Describe an algorithm to construct a d-dimensional kd-tree for a set P of n points in $\mathbb{R}^d$. Prove that the algorithm takes $O(n \log n)$ time and that the tree can be stored in $O(n)$ space. Assume d is constant.

   (b) (3 points) Describe a query algorithm for performing an orthogonal range query in a d-dimensional kd-tree.

   (c) (2 points) For $d = 3$, show that your query algorithm runs in time $O(n^{\frac{2}{3}} + k)$. For this, develop a recurrence for $Q(n)$ and solve it. *(Hint:* $\log_8 4 = \frac{2}{3}$*)*

   (d) (2 points) Generalize your query time analysis for general (constant) d, to show that the query algorithm runs in time $O(n^{\frac{d-1}{d}} + k)$.

3. **Nesting Trees (10 points)**
   In class we used a *segment-range* tree to solve the 2-dimensional windowing problem. This two-level tree consists of a segment tree as the primary tree, and each node of the primary tree stores a link to a secondary range tree.

   Now consider defining a *range-segment* tree which has a range tree as the primary tree and segment trees as the secondary trees. We can also define a *segment-segment* tree in a similar way, and *range-range* trees we have already studied in class.

   Compare all four data structures and argue what kinds of problems each can be used to solve. Analyze and compare the query times, construction times, and space complexities.