2D Range Tree

A JavaScript Implementation & Visualization

Kaixuan Zhou, 05/03/2021

Range Tree Recap

Range Tree Complexity Overview

- Construction: O(nlog^(d-1)n)
- Space: O(nlog^(d-1)n)
- Search Time Complexity: O(log^d(n) + k)
- Adding fractional cascading
  - Search time Complexity: O(log^(d-1)n + k)
- KD Tree Size: O(n), Time Complexity: O(n^(1-1/d) + k)

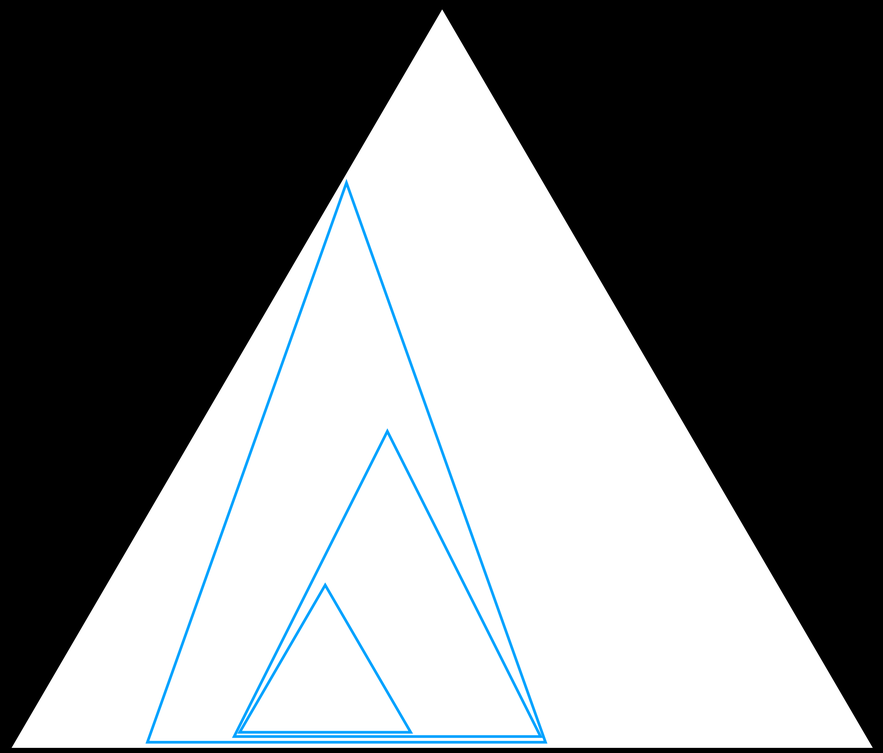
Range Tree Construction - Time Complexity

- Construct a OneD Range tree, and construct another OneD Range tree for each of the node. Do this recursively for k-dimensions.
- Uses the median value of that dimension as the node -> A balanced tree
- $$T_d(n) = O(n \log n) + O(\log n) \cdot T_{d-1}(n).$$
- With Optimization:

  $T_2(n) = O(n \log n)$, this recurrence solves to $O(n \log^{d-1} n)$

Range Tree Storage Complexity

- Each point would appear in [log^(d-1)n] trees
- The number of nodes in the tree in linear in respects to the number of leafs
- O(nlog^(d-1)n)



Range Tree Search Review

- Searching in a first-level tree, takes O(logn), and query logn number of (d - 1)-dimensional range trees.
- $$Q_d(n) = O(\log n) + O(\log n) \cdot Q_{d-1}(n)$$
  where $Q_2(n) = O(\log^2 n)$
- $$Q_d(n) = O(\log^d n)$$
- Report points, +k

My Project Demo

- Takes in points & query from user
- Draw the 2D range tree
- Visualize the search