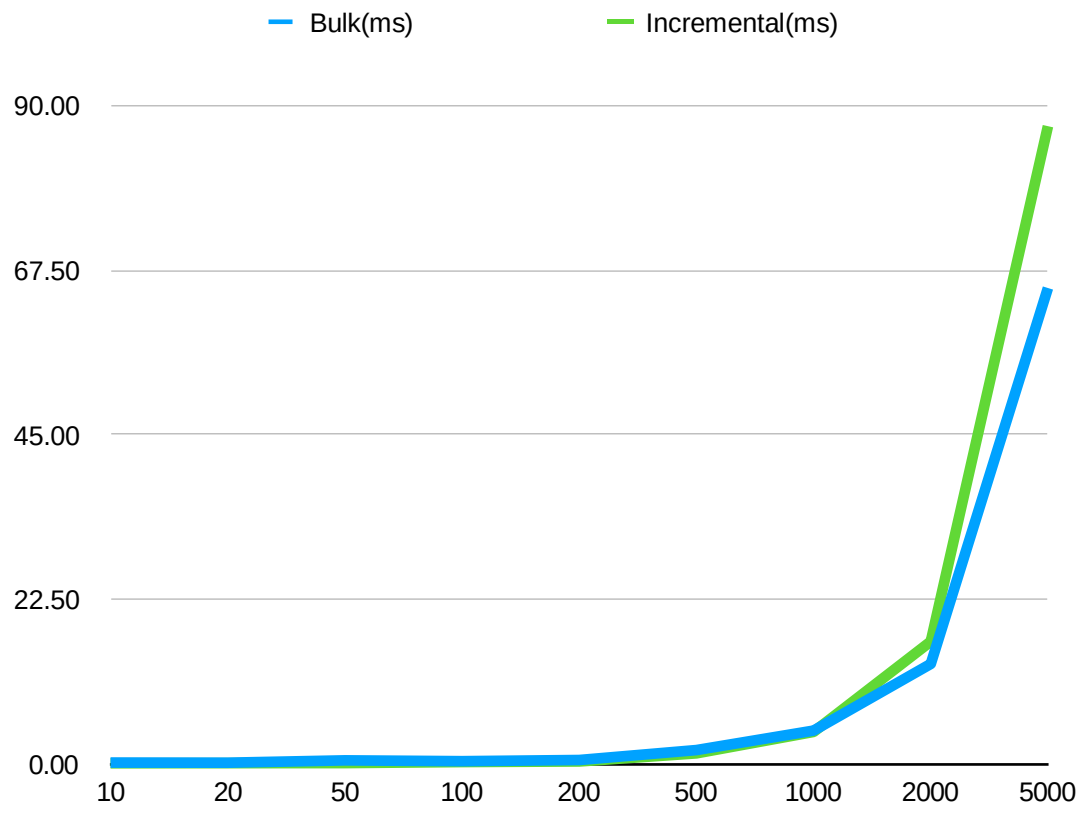
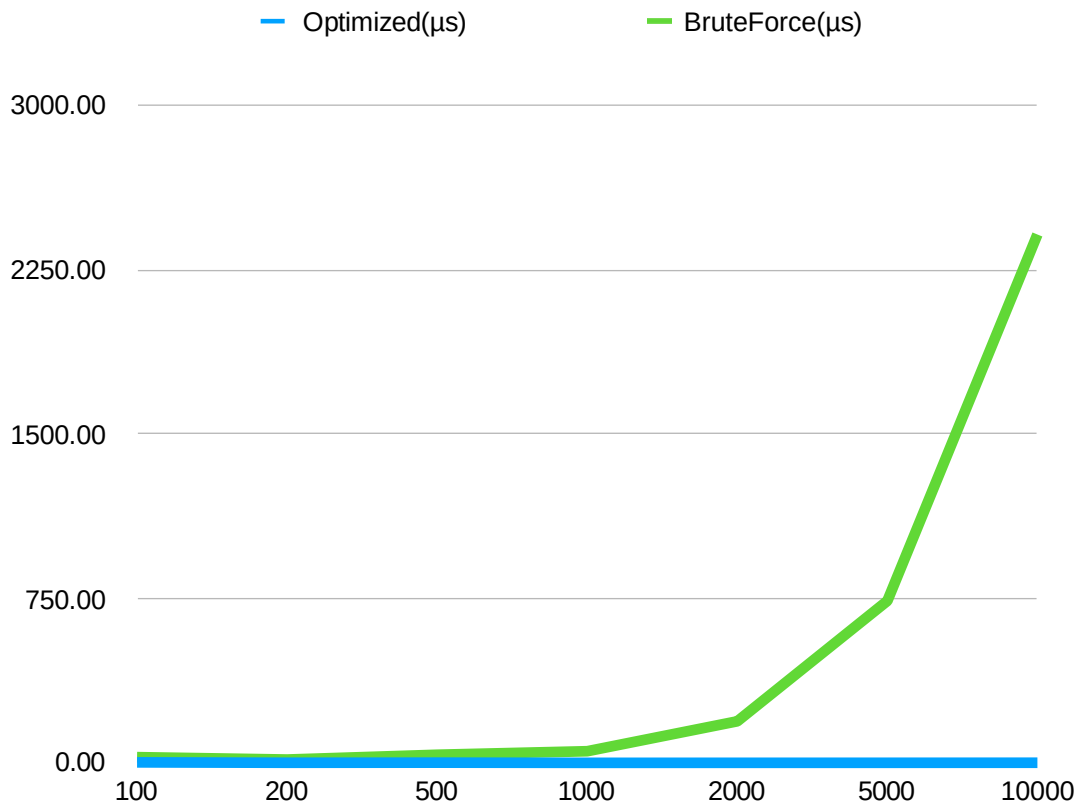
Perform an experiment to compare the time to construct a tree with N segments using a worst case input using the "bulk construction" constructor compared with inserting the segments one at a time into an initially empty tree. As an example, consider a bunch of vertical segments. What is the "worst case" order to insert them? Plot the runtimes of the 2 methods for building the tree. Do your experiments match the Big O growth rates you expected?



The worst-case input is vertical segments sorted from left to right, which creates a maximally unbalanced tree during incremental insertion, requiring traversal of the full depth for each new segment.
The Bulk construction expects O(nlogn) runtime and incremental expects O(N square). As shown in experiment, Incremental construction perform slower when size increases.

which will visit all nodes. Does our optimized collision detection routine run in the big O you expected? Be sure to describe the details of your experiment



Random segments were generated until they intersected with a query line, mimicking 2D rendering techniques. Performance was averaged over multiple trials with varying tree sizes. The brute force method's runtime increased rapidly with more segments. The optimized BSP tree maintained almost constant time, demonstrating the effectiveness of spatial partitioning for collision detection in large datasets.