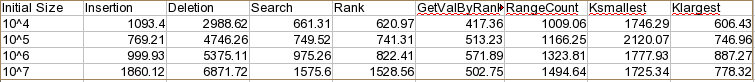
Gray Houston

ghousto

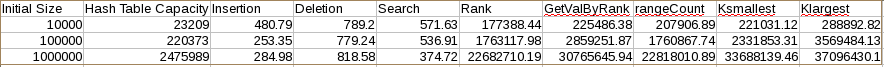
0026483532

Project 3 Analysis/Report

Report 1: RB Tree Performance

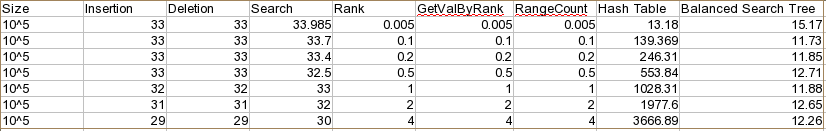


Report 2: Linear Probing Hash Table Performance



Report 3: Hashing vs. BST

* functions are represented in percentage of total calls
* data structure performance averaged where n = 10



The table above shows that when there are fewer calls to Rank, GetValByRank, and RangeCount, the Hash Table's performance will improve will the BST's will become worse. However, the Hash Table only actually surpasses the BST in speed when the rank-related functions are practically never called. As shown in the table, the smallest percentage of calls to non rank-related functions (e.g. Insertion, Deletion) where the Hash Table outperforms the BST is 99.985% . If the percentage becomes lower than this, the BST will outperform the Hash Table.

Therefore, we can infer that if there are no calls made to the rank-related functions, then the Hash Table will outperform the BST. However, if there are any calls to a rank-related function, the BST will perform better. The data structures' relative performance for the rank-related function shown in tables 1 and 2 provide evidence for this theory.

Furthermore, the data does not suggest that there is a noticeable correlation between the distribution of Insert, Delete, and Insert function calls and the performance of either the Hash Table or the BST. The only apparent determining factor is the total percentage of the first three functions to the total percentage of the rank-related functions (when the rank-related percentage is non-trivial). Both the Hash Table and the BST appear to perform worse as the percentage of rank-related functions increases. However, the effect is significantly stronger on the Hash Table than the BST