The expected time complexity for different data structures:

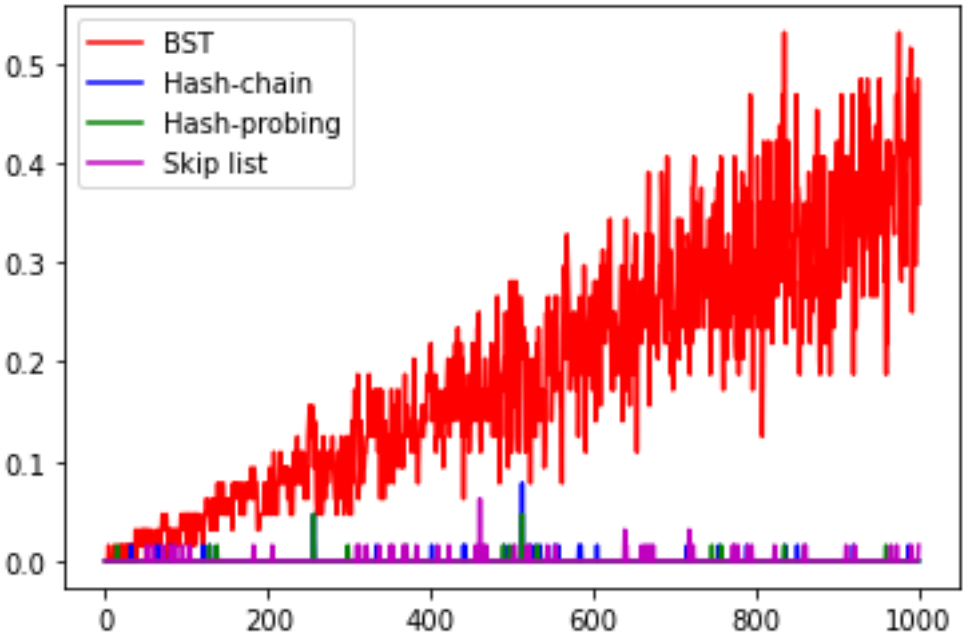
| Data structure | Access | Insertion | Deletion |
|---|---|---|---|
| **Binary Search Tree** | O(log N) | O(log N) | O(log N) |
| **Hash Table** | O(1) | O(1) | O(1) |
| **Skip List** | O(log N) | O(log N) | O(log N) |

The worst time complexity for different data structures:

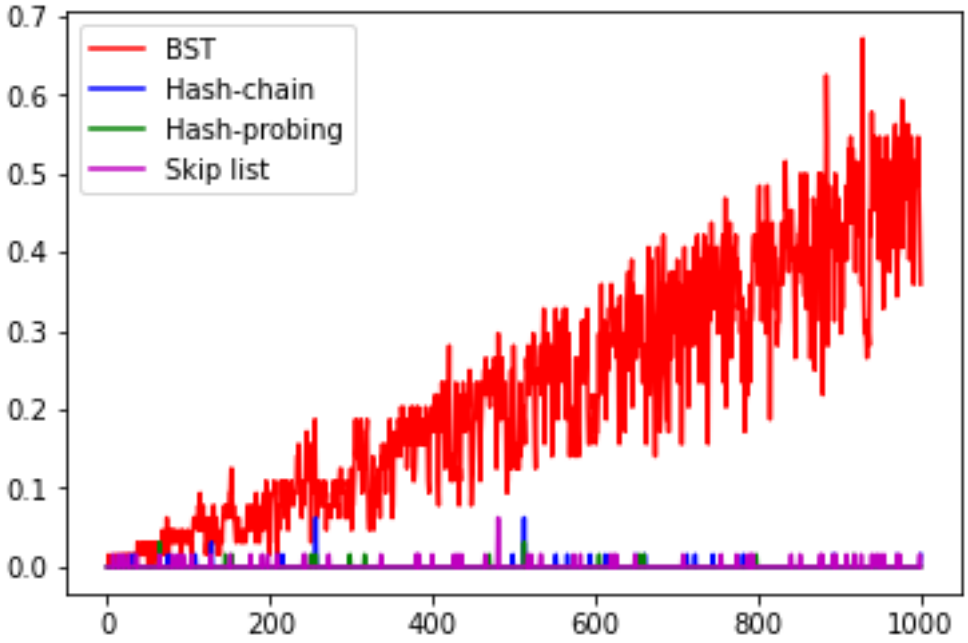
| Data structure | Access | Insertion | Deletion |
|---|---|---|---|
| **Binary Search Tree** | O(N) | O(N) | O(N) |
| **Hash Table** | O(N) | O(N) | O(N) |
| **Skip List** | O(log N) | O(log N) | O(log N) |

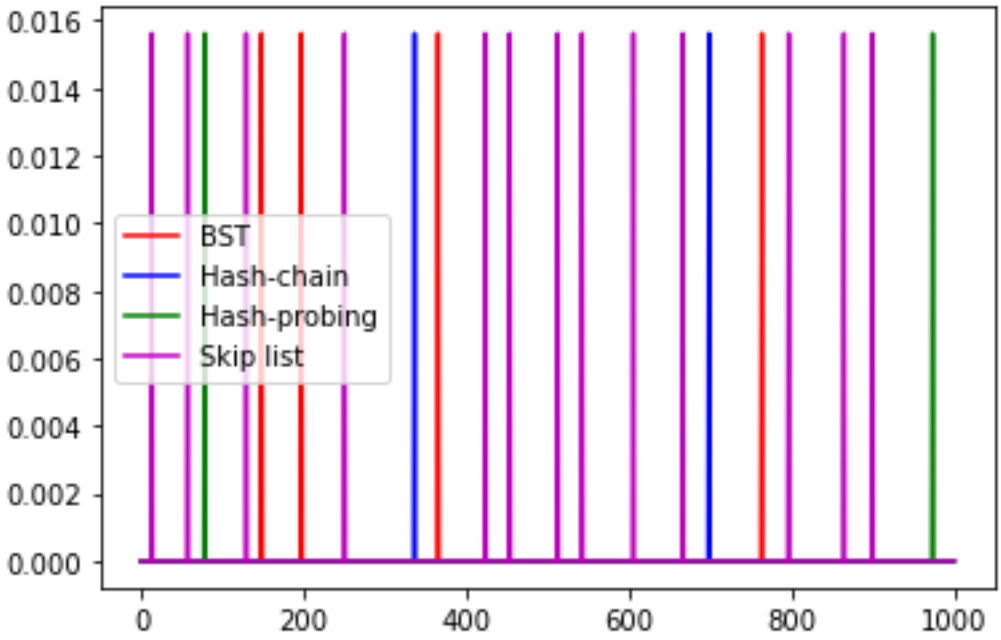
As single individual operation time varies a lot, the plot is done per every 10 operations per course instructor:

Insertion:




Insertion and access:






Deletion:

The insertion and insertion/visit plots show the expected O(log n) for the BST and O(1) for the skip list and hash tables. However, the deletion plot does not show the expected time complexity, it may be caused that each deletion operation is too fast to quantify in the plot.

Which hash table implementation (separate chaining or linear probing) is faster?
The two hash table implementations both have an expected O(1) time complexity. However, linear probing is faster on average. This is because the memory addresses used for the single list are closer together, while separate chaining can have each data structure in different locations far apart from each other.