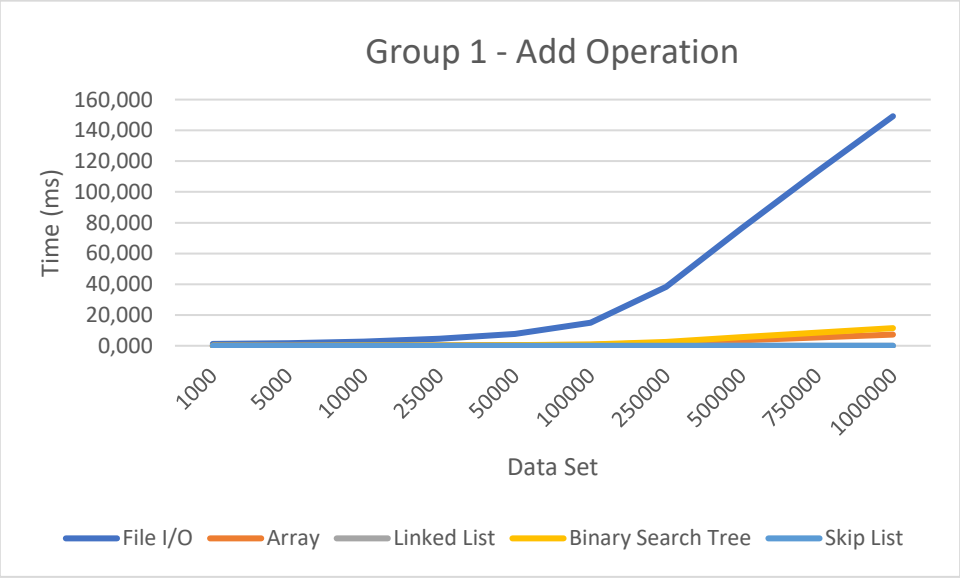
Harun Yahya Demirpençe – 150210022

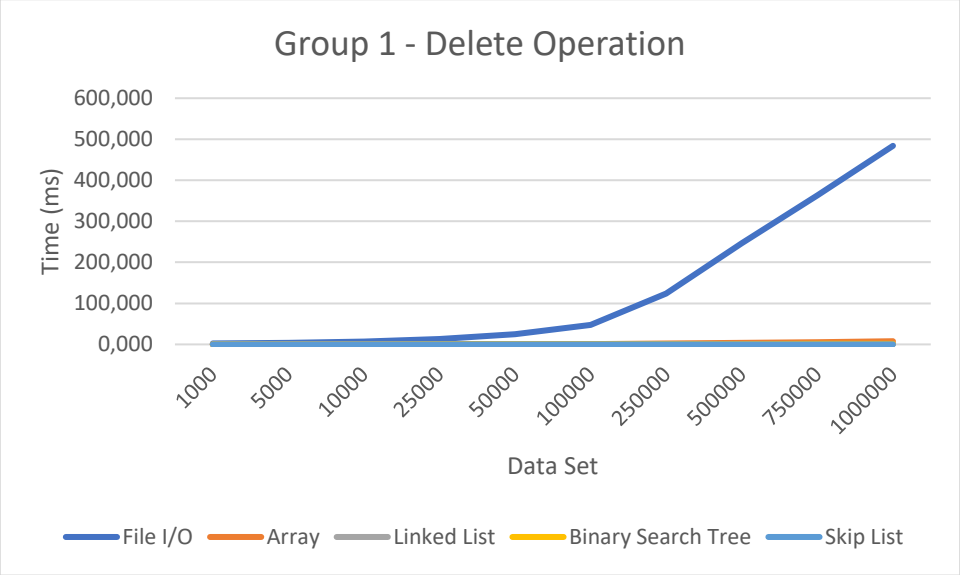
Homework 5 – Comprehensive Comparison

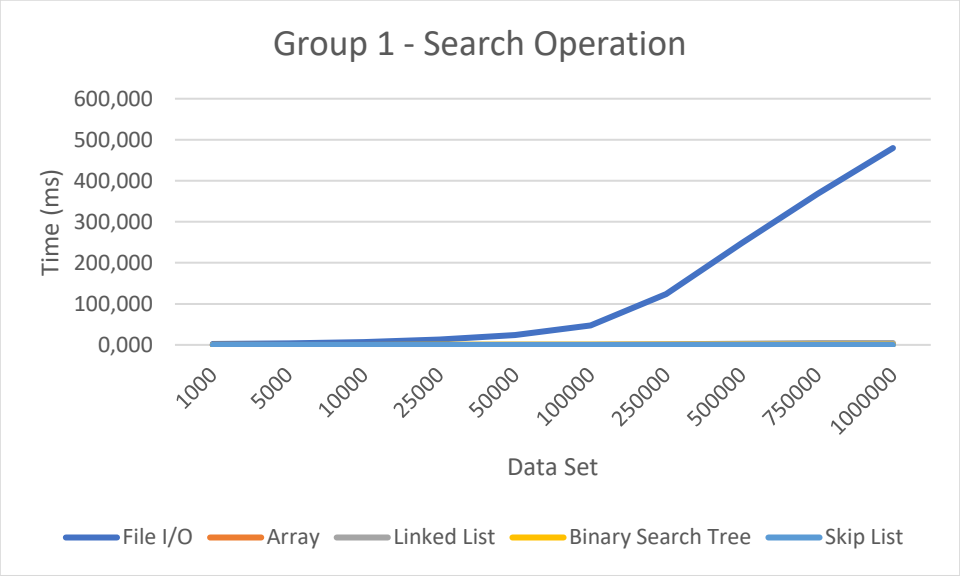
In this homework, we are expected to measure the addition, removal, and search operation times for File I/O, Array, Linked List, STL Vector, STL List, Binary Search Tree, STL Map, Skip List solutions, then compare the operations and data structures. The data sets provided in time measuring changes between 1000 to 1000000 employees. The report about these comparisons written in below:

Group 1

As seen in the below group 1 graphics; the file solution consistently exhibits higher time complexity than all other data structures for each operation. This increased time is attributed to the sequential process of reading data from the file, storing it in a temporary structure, and subsequently copying it back to the original file, resulting in considerable time overhead. The binary search tree structure behaves similarly to a linked list due to the sorted dataset, causing their performance to degrade O(logn) to O(n) (worst case). In contrast, the skip list solution stands out by demonstrating the least time consumption among the various data structures. Skip lists provide efficient search operations by maintaining multiple layers of pointers, allowing for O(logn) complexity. The remaining data structures operate as anticipated.

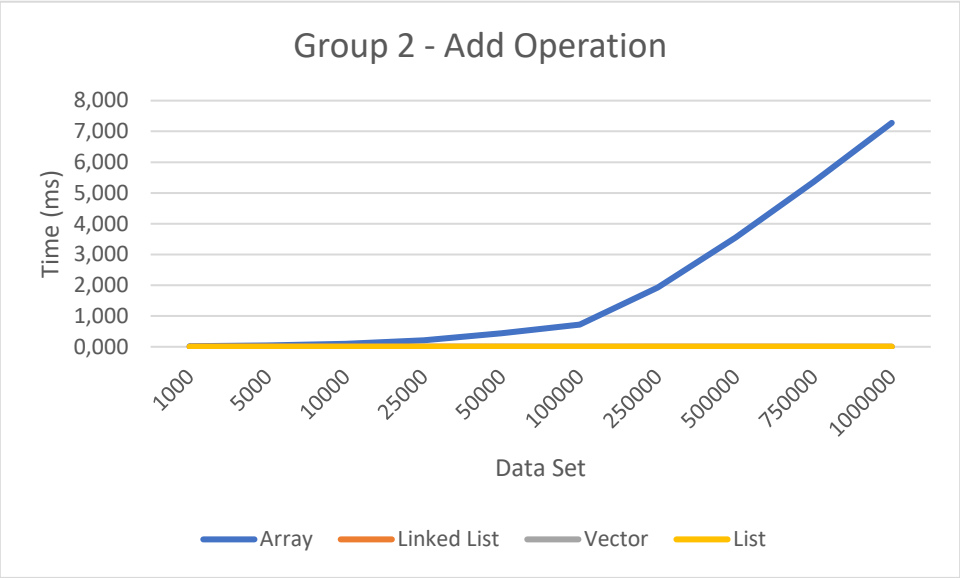


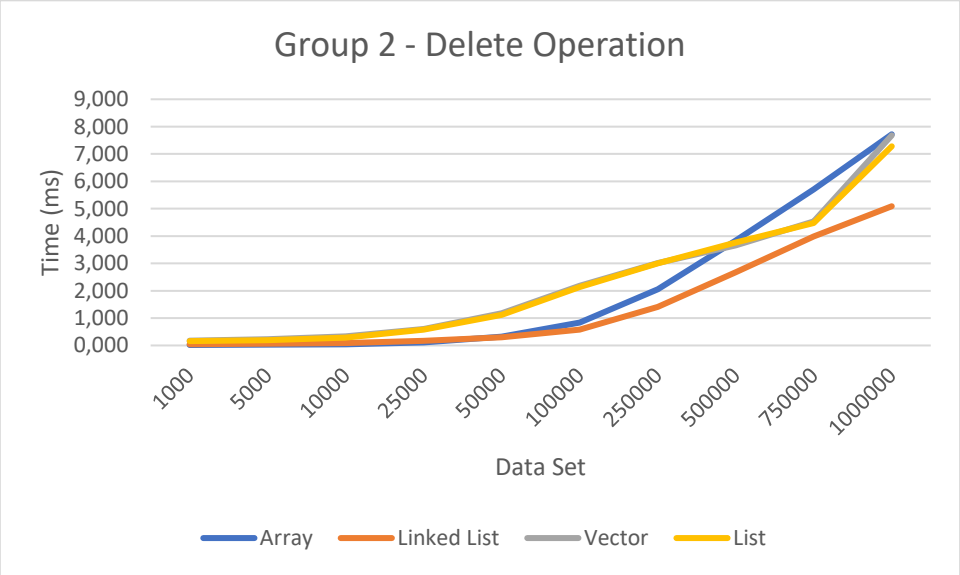


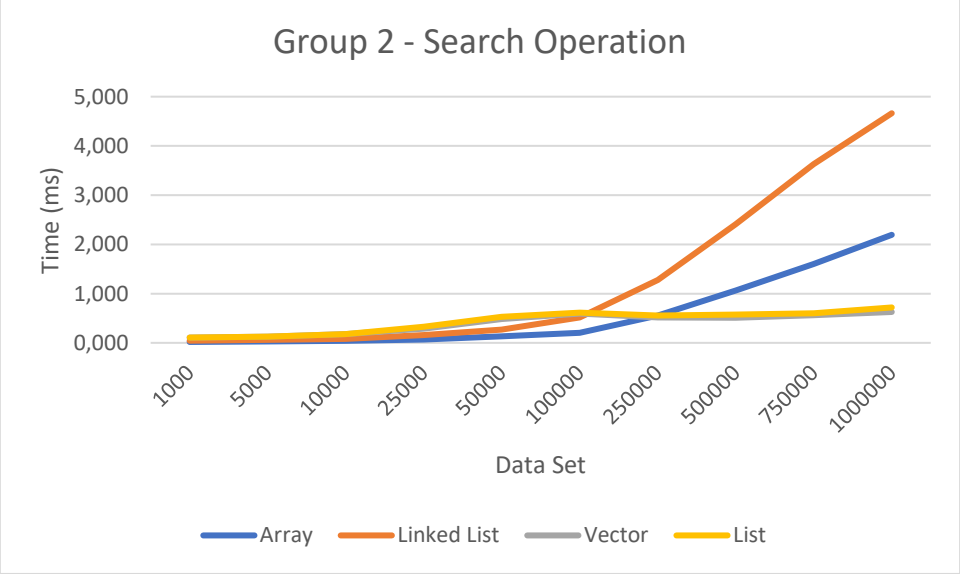


Group 2

As seen in the below Group 2 graphics, the linear structure of the array solution shows it has O(n) time complexity for the addition, removal, and search operations. The addition operation for other data structures, such as list, vector, and linked list, exhibits O(1) time complexity because they add elements to the end of the list, and the process involves a constant time operation. For the removal and searching operations, the time complexities are similar among the data structures. The vector, list and linked list can have O(n) time complexity for removal and searching, as they may require traversing the entire list in the worst case. Therefore, while addition tends to be more efficient for these structures, the efficiency of removal and searching operations depends on the specific characteristics of the data structure used.


Group 2 - Add Operation
Time (ms)
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0,000
1000
5000
10000
25000
50000
100000
250000
500000
750000
1000000
Data Set
Array
Linked List
Vector
List


Group 2 - Delete Operation
Time (ms)
9,000
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0,000
1000
5000
10000
25000
50000
100000
250000
500000
750000
1000000
Data Set
Array
Linked List
Vector
List

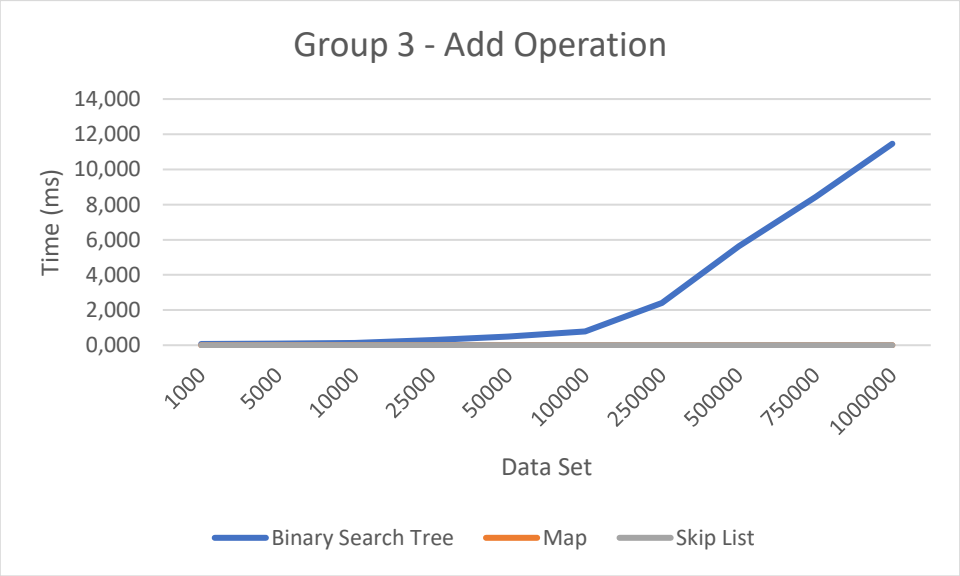

Group 2 - Search Operation
Time (ms)
5,000
4,000
3,000
2,000
1,000
0,000
1000
5000
10000
25000
50000
100000
250000
500000
750000
1000000
Data Set
Array
Linked List
Vector
List

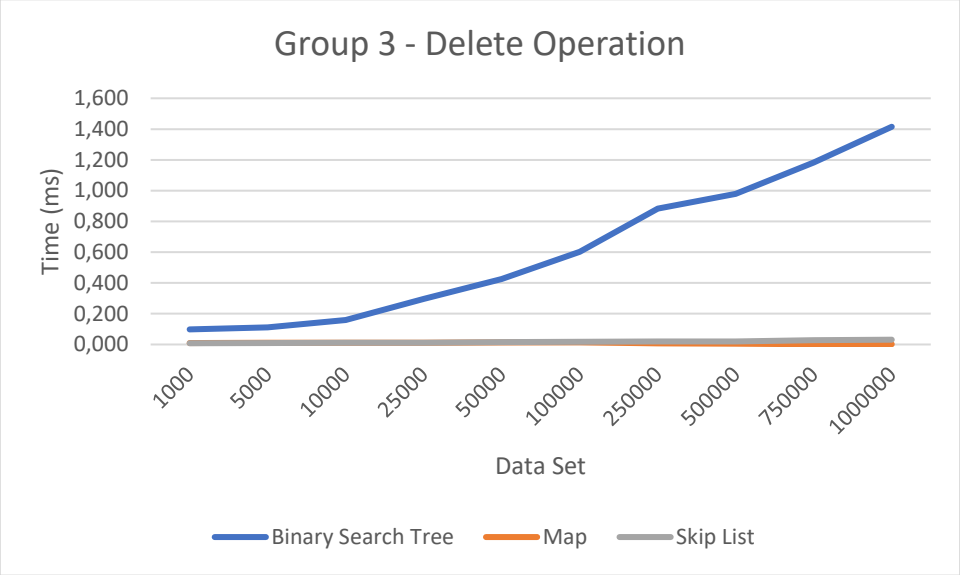
Group 3

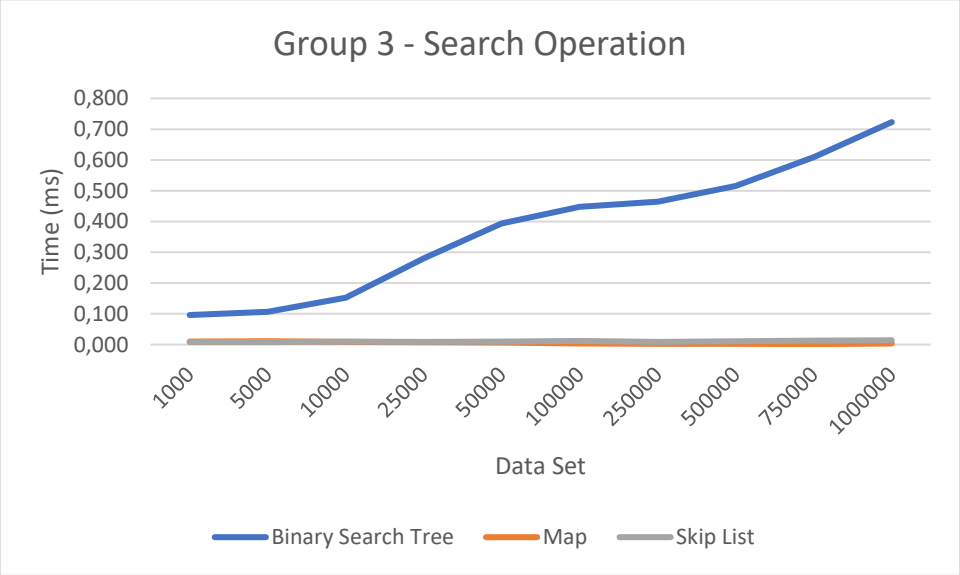
As seen in below group 3 graphics, the binary search tree addition, removal, and search algorithms take most of the time since the provided data set is sorted. The binary search tree's inherent structure relies on maintaining a balanced arrangement of elements for optimal performance. In the case of a sorted dataset, each new element often gets added to one side of the tree, leading to an unbalanced state and causing search operations to approximate linear time complexity. Removals and additions also incur additional overhead due to the need for maintaining balance. On the other hand, the skip list, with its layered structure and multiple levels of pointers, provides efficient search operations. Skip lists maintain O(logn) time complexity for search operations. The skip list's adaptability to varying datasets contributes to its consistent performance. Additionally, maps, often implemented as red-black trees or balanced search tree variants, offer efficient search, insertion, and deletion operations. Their self-balancing properties ensure stable logarithmic time complexity for search operations, making them well-suited for dynamic datasets. In the case of a sorted dataset, maps can still exhibit reliable performance due to their ability to maintain balance dynamically.

Result

In conclusion, the comprehensive comparison of data structures and operations across three distinct groups reveals valuable insights into their respective strengths and weaknesses.




Group 3 - Delete Operation
Time (ms)
1,600
1,400
1,200
1,000
0,800
0,600
0,400
0,200
0,000
1000
5000
10000
25000
50000
100000
250000
500000
750000
1000000
Data Set
Binary Search Tree
Map
Skip List


Group 3 - Search Operation
Time (ms)
0,800
0,700
0,600
0,500
0,400
0,300
0,200
0,100
0,000
1000
5000
10000
25000
50000
100000
250000
500000
750000
1000000
Data Set
Binary Search Tree
Map
Skip List