Deniz Karakay – 2443307 06.01.2023

EE441 – Programming Assignment 3

Part 3:

The plots of average time to compute determinant and verage memory usage of the computation for Binary Search Tree approach from the number of BST Nodes and from RAM usage can be seen in *Figure 1, Figure 2*, *and Figure 3* respectively. Similarly, the plots of average time to compute determinant and average memory usage of the computation for Hash Table from the number of HT Items, and from RAM usage approach can be seen in *Figure 3, Figure 4, and Figure 5,* respectively. The results from terminal for Hash Table can be seen in *Figure 7-11*, on the other hand the results from terminal for Binary Search Tree approach can be examined in *Figure 12-16*. The following results can be deduced from the figures:

1. As we increase the N, the dimension of square matrix, the average time and the average memory usage is increased exponentially. Therefore, the shapes of all figures are similar.

2. If we compare the results of *Figure 1* and *Figure 4*, we can observe that average time to compute determinant is much greater in every N for Binary Search Tree case. For N=13, the average time for Binary Search Tree is 4.52 seconds, whereas for Hash Table it is around 0.19 seconds. We can state that Hash Table approach is much faster than the Binary Search Tree approach for calculation of determinant.

   Hash Table is much faster because searching or inserting a new element into a Hash Table for average case is $O(1)$. The worst-case scenario for searching or inserting for Hash Table is $O(n)$. Due to probing we may be able to scan entire table to search for a value or insert a value. However, for a Binary Search Tree, searching or inserting a new element for average case is $O(\log n)$. The worst-case is $O(n)$ again.

3. If we compare the results of *Figure 2* and *Figure 5*, we can observe that average memory usage to compute determinant is exactly same as both cases. For N=13, both the average memory usage (average number of BST nodes) for Binary Search Tree and for Hash Table (average number of HT items) is 2345. If we compare the results of *Figure 3* and *Figure 6,* we can observe that RAM usages are also same. These values are same because we are using the same approach for both methods and consequently, we store same number of matrices regardless of the method that we utilized. The average and worst-case spaces for both Binary Search Tree and Hash Table is $O(n)$. Since their spaces are same, the result is expected.

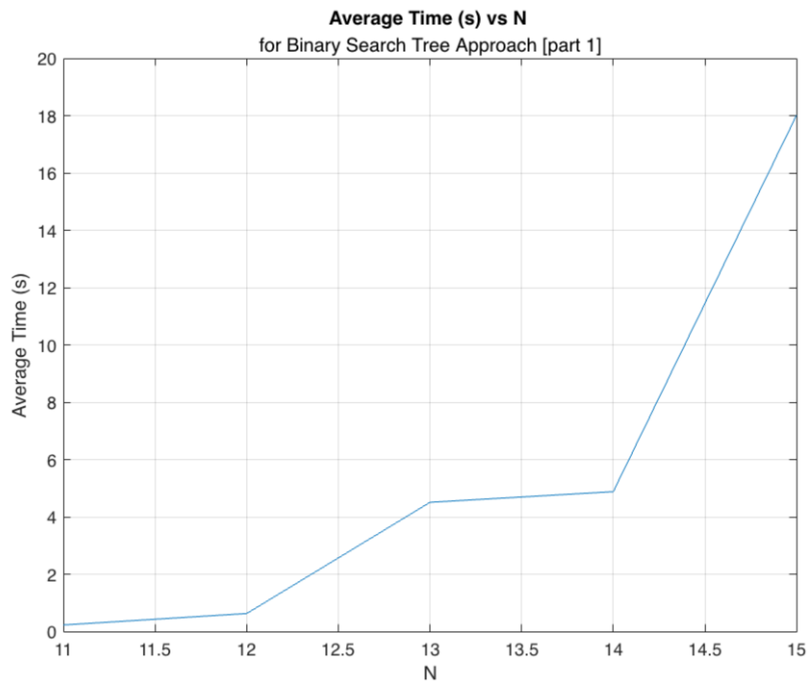

Figure 1. Average Time (s) vs N for Binary Search Approach [part 1]

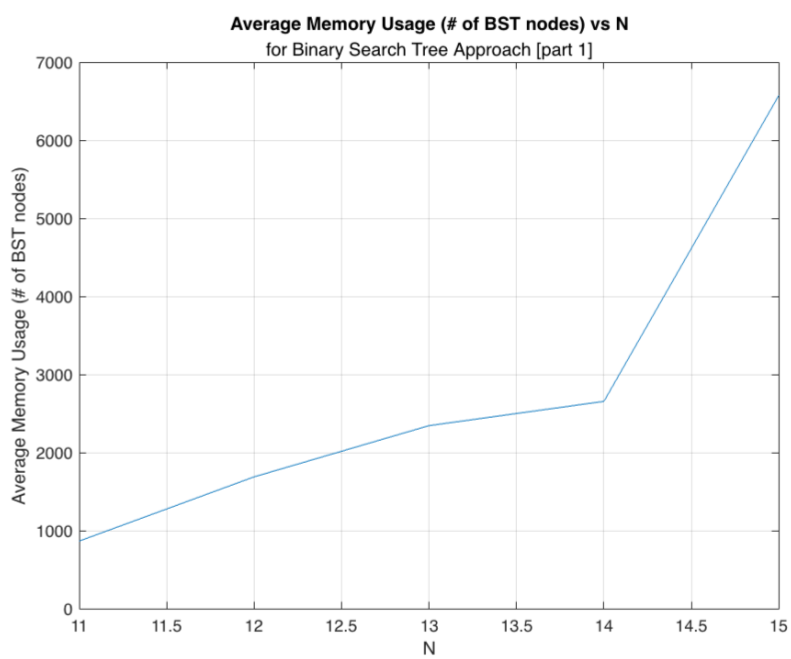

Figure 2. Average Memory Usage (# of BST nodes) vs N for Binary Search Approach [part 1]

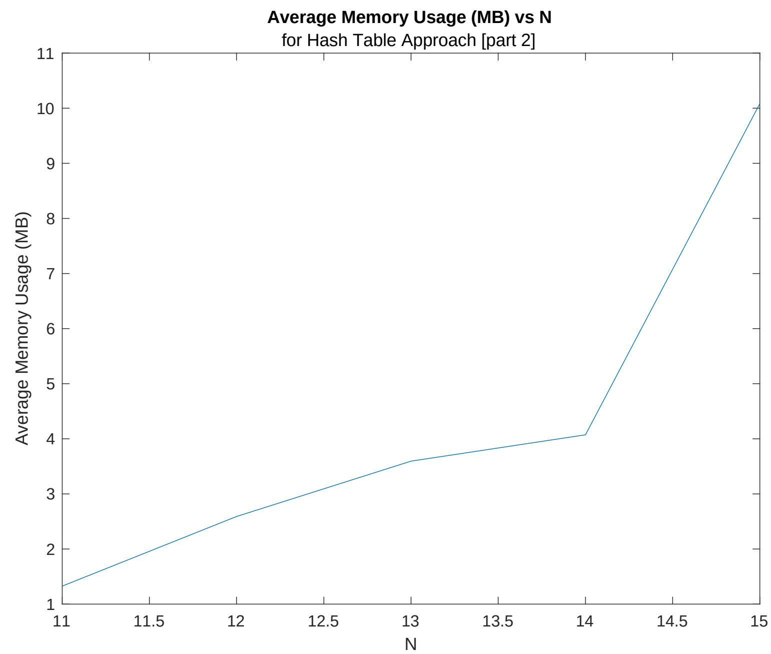

Figure 3. Average Memory Usage MB) vs N for Binary Search Approach [part 1]

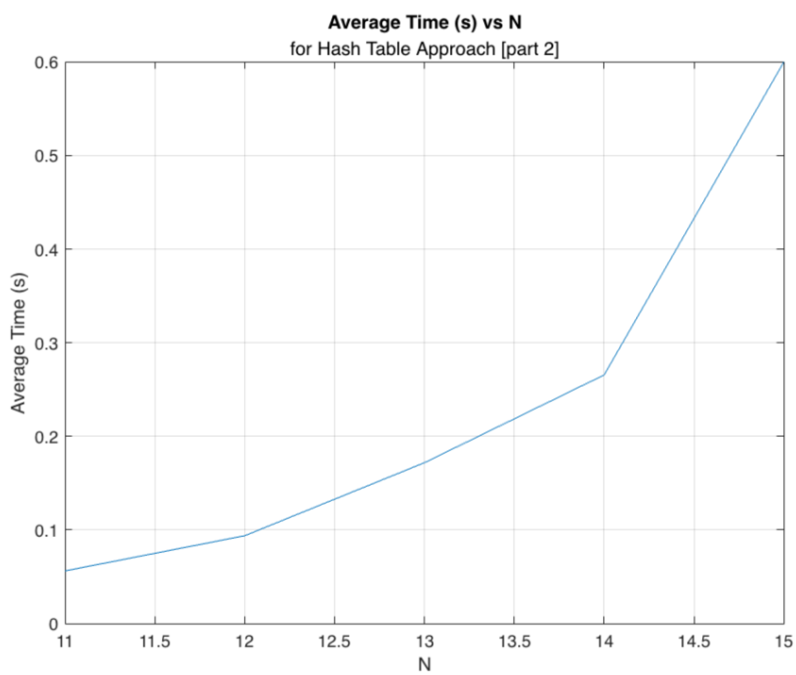

Figure 4. Average Time (s) vs N for Hash Table Approach [part 2]

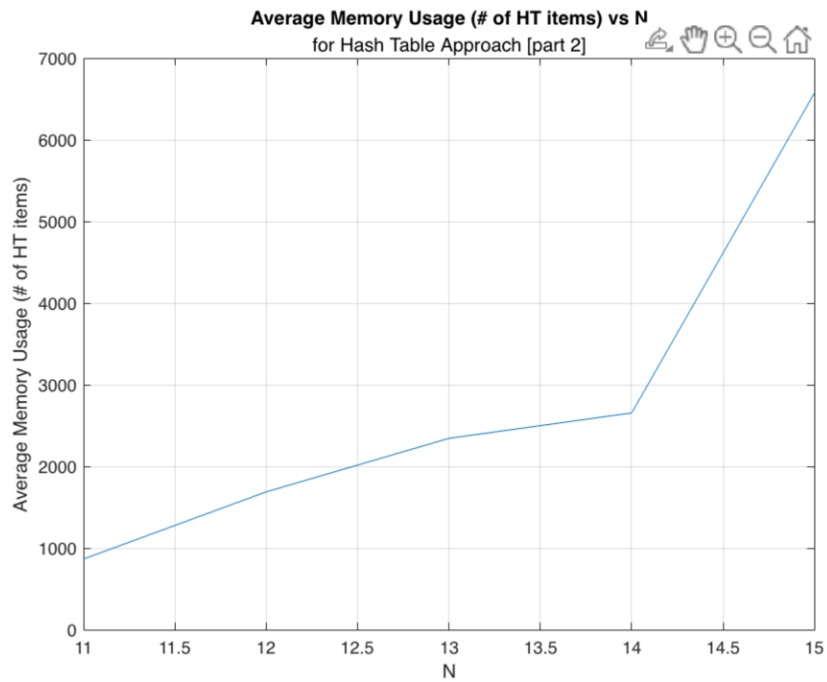

Figure 5. Average Memory Usage (# of HT items) vs N for Hash Table Approach [part 2]

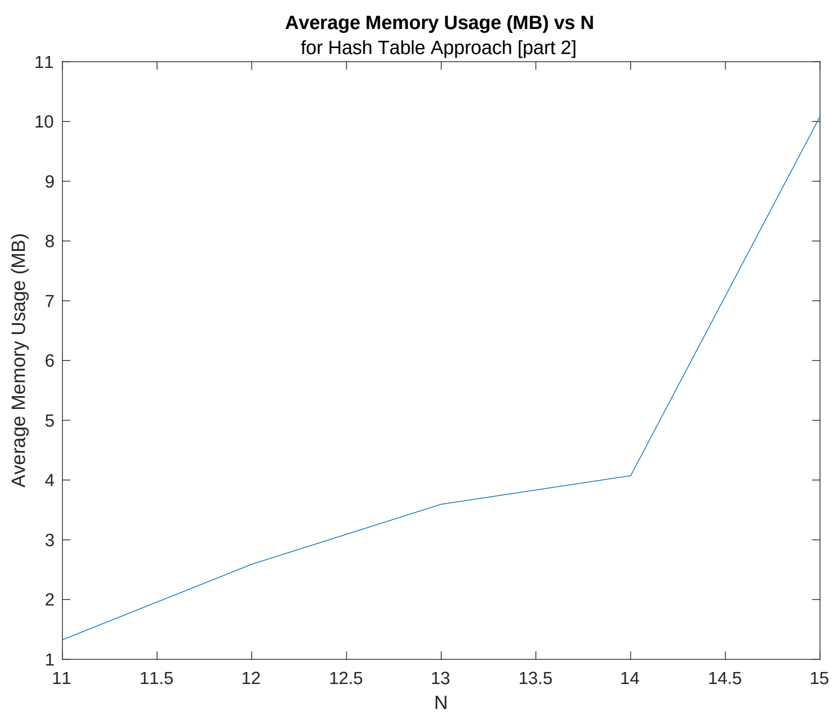

Figure 6. Average Memory Usage (MB) vs N for Hash Table Approach [part 2]

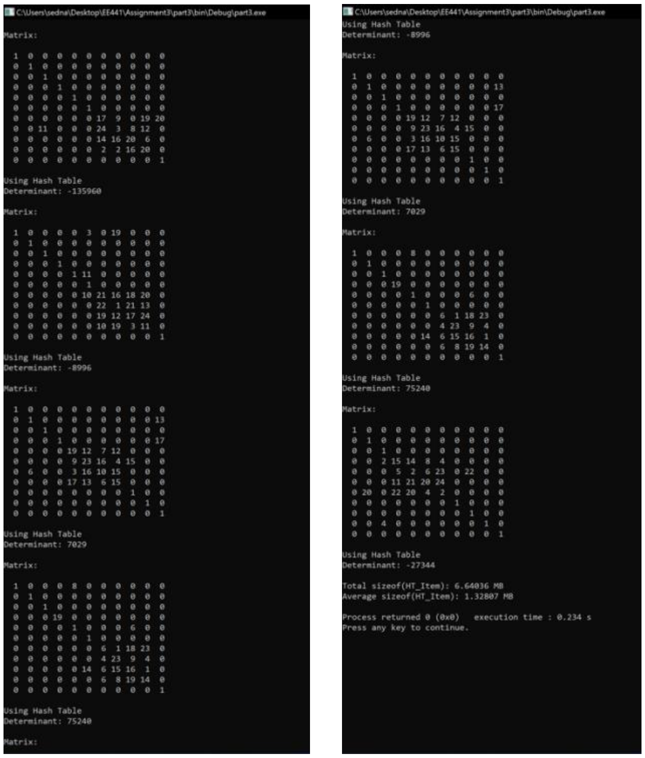

Figure 7. Results for Hash Table approach N=11

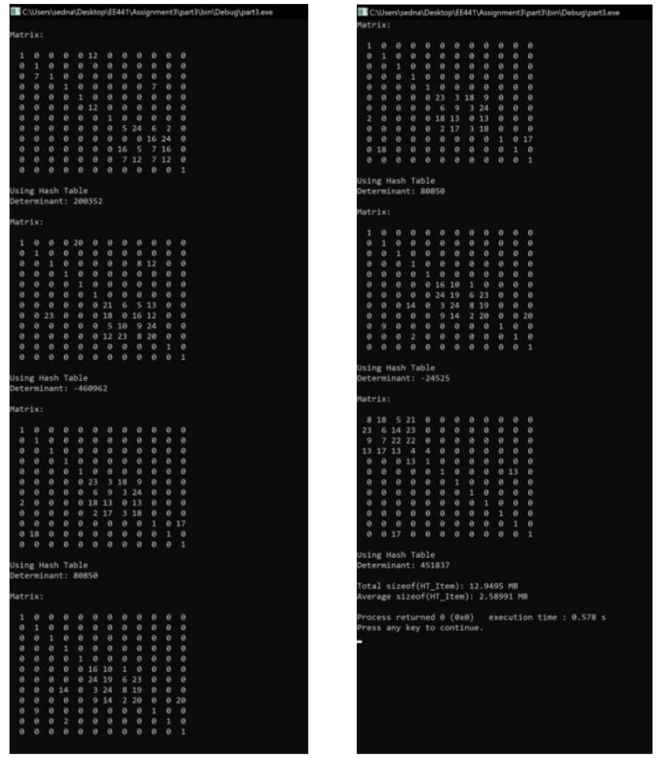

Figure 8. Results for Hash Table approach N=12

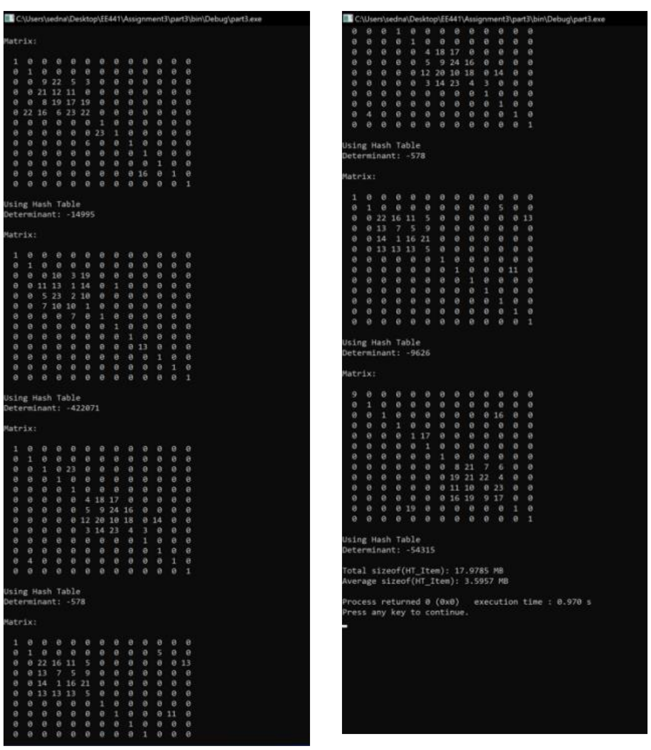

Figure 9. Results for Hash Table approach N=13

Figure 10. Results for Hash Table approach N=14

Figure 11. Results for Hash Table approach N=15

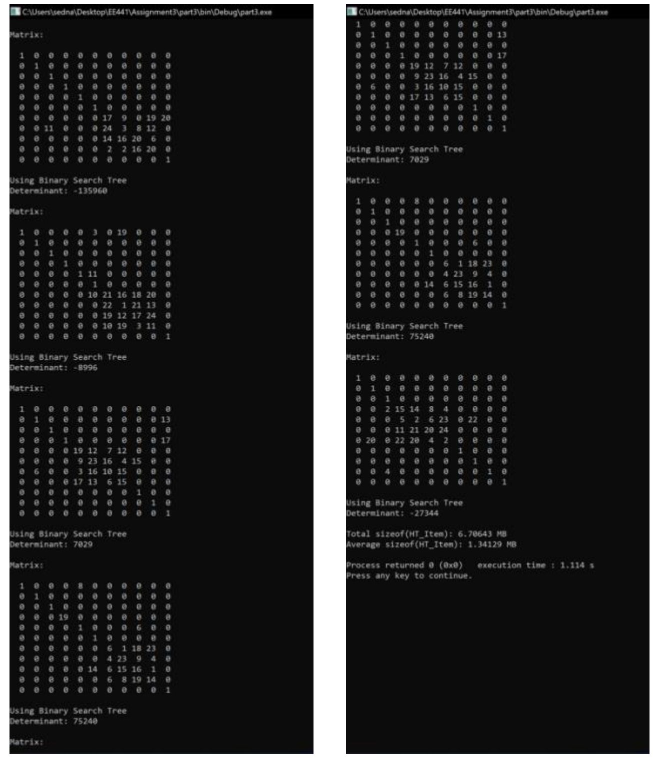

Figure 12. Results for Binary Search Tree approach N=11

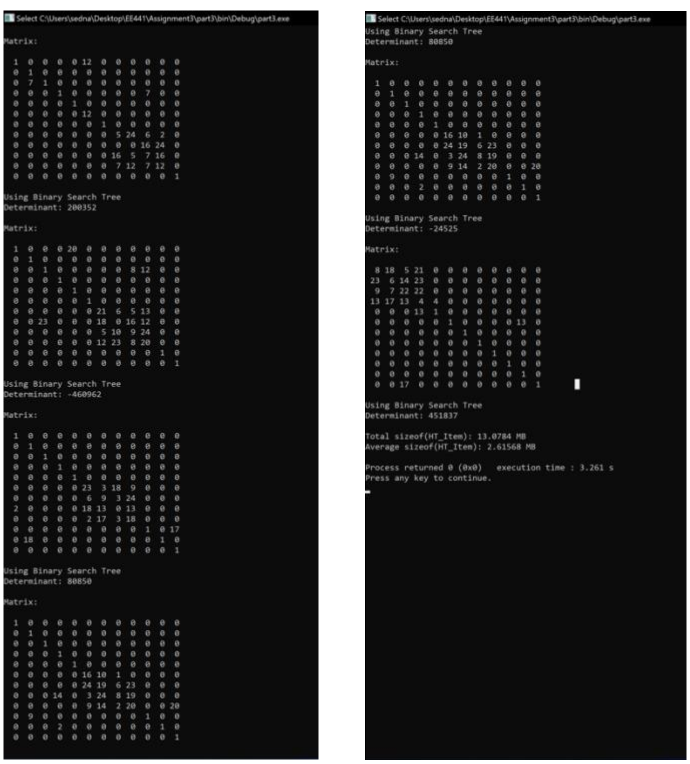

Figure 13. Results for Binary Search Tree approach N=12

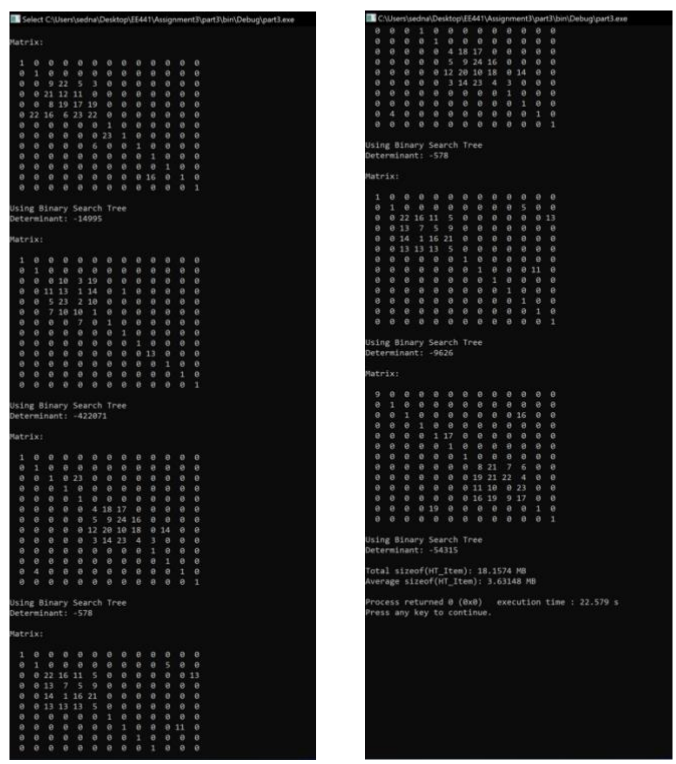

Figure 14. Results for Binary Search Tree approach N=13

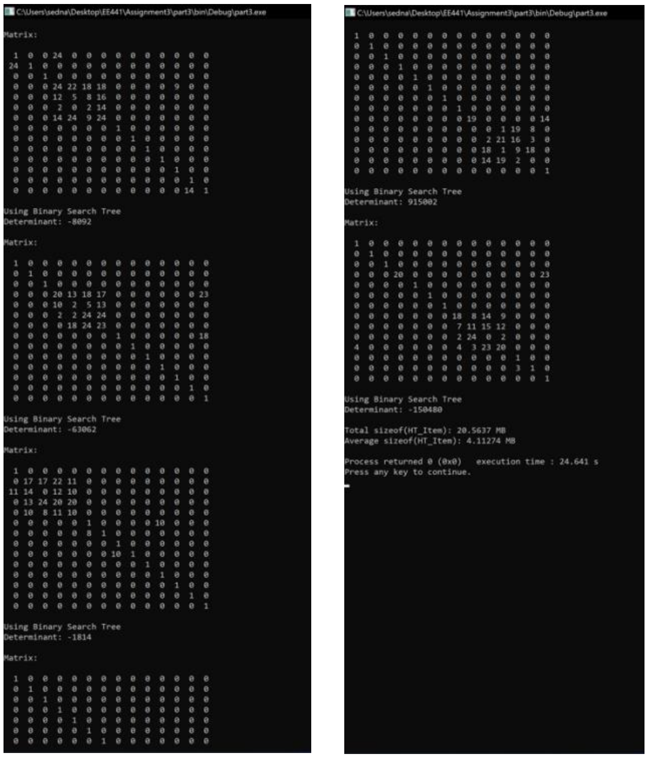

Figure 15. Results for Binary Search Tree approach N=14

Figure 16. Results for Binary Search Tree approach N=15