Assignment 3 Search Analysis

COMP 205 Data Structures and Algorithms

Mert Doğan (041701041)
Date: 26.12.2018

1- My Main Code's Algorithm

1 – In the main method, I import java.util.ArrayList, HashMap, LinkedList for using ArrayList, HashMap, LinkedList for search operation and java.util.Random assign random integers. I also import some io and jxl libraries to write an excel file.

2 – I declare and 4 arrays for searching and assign random integers, i, j, k, l, t for indexing, double data types for calculating and print elapsed time.

3 – I use a nested for loops, first one for create new integer arrays and increase their size for data structures, the second one for filling the array between 1 to M. I shuffle the filled array and copy the array to other data structure's array in the first loop. In the third for loop, I insert the shuffled random integers to each data structures.

4 – In addition, I create a for loop which inside the first for loop to print each elapsed time for each data structures and print out calculated elapsed time.

5 – In the shuffleArray method, I create a random index and swap the current index and random index in a for a loop.

6 – The last piece of codes in the main method for write the search values to excel file.

2- The BST Class Algorithm

1 – The class has already given me from my instructor. – The class contains Binary Search Tree features which are data fields, BST' s operations, methods and some conditions.

2 – I used the class to create a BST object, fill into the object with shuffled array and use it in the main method for searching.

3- The Plots

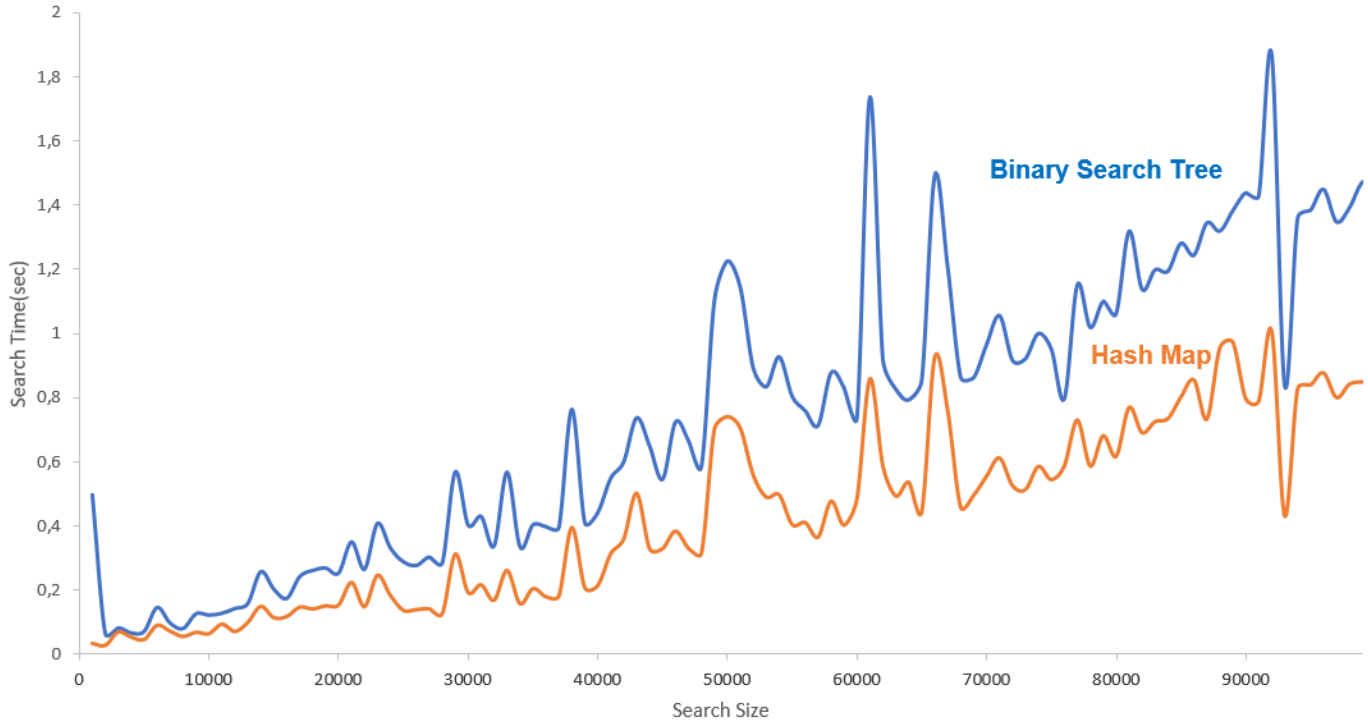

Figure 1. Search size vs search time plot for Linked List and Array List.

Figure 2. Search size vs search time plot for Hash Map and BST.

4- Sample Output

The outputs are given M= M+100 (1000,1100,1200,…M) because of running time.

```
Sample Program Output
Sequence Size; Array List Search Time; Hash Map Search Time; Linked List Search Time; BST Search Time
1000;8,85;0,41;14,26;0,90
2000;1,42;0,28;3,86;0,58
3000;1,66;0,29;4,60;0,39
4000;1,78;0,27;5,43;0,45
5000;1,46;0,31;5,84;0,33
6000;1,32;0,18;6,66;0,36
7000;1,47;0,21;7,54;0,36
8000;1,66;0,21;8,47;0,38
9000;1,89;0,22;9,50;0,44
10000;2,02;0,25;10,19;0,41
11000;2,25;0,24;11,97;0,45
12000;2,36;0,26;12,52;0,49
13000;3,03;0,32;14,90;0,49
14000;3,36;0,17;15,48;0,38
15000;3,43;0,16;16,22;0,41
16000;3,65;0,19;18,82;0,41
17000;3,84;0,19;18,81;0,43
18000;4,17;0,24;21,23;0,46
19000;4,49;0,21;22,53;0,47
20000;4,81;0,23;23,97;0,50
21000;5,00;0,23;25,66;0,55
22000;5,39;0,25;27,57;0,69
23000;5,72;0,26;29,58;0,63
24000;6,32;0,40;31,72;0,75
25000;6,55;0,29;33,83;0,72
26000;7,82;0,30;35,23;0,67
27000;9,30;0,24;38,96;0,81
28000;9,46;0,22;40,67;0,94
29000;14,06;0,31;50,21;1,24
30000;15,10;0,39;51,59;1,16
31000;14,04;0,34;55,06;1,02
32000;11,02;0,22;49,21;0,92
33000;11,40;0,24;52,08;1,11
34000;11,97;0,26;54,86;1,16
```

35000;12,50;0,26;57,39;1,09
36000;13,37;0,28;60,24;1,12
37000;13,73;0,27;63,41;1,16
38000;14,35;0,30;65,89;1,20
39000;16,84;0,53;78,83;1,37
40000;23,07;0,45;101,94;1,89
41000;16,67;0,51;76,10;1,34
42000;21,08;0,47;96,11;1,75
43000;18,94;0,38;85,11;1,49
44000;18,71;0,40;86,45;1,47
45000;19,82;0,38;89,19;1,65
46000;20,44;0,59;92,84;1,66
47000;40,84;0,74;92,43;1,81
48000;25,20;0,46;112,77;2,02
49000;23,07;0,42;102,50;1,71
50000;33,93;0,59;152,75;2,57
51000;24,06;0,40;109,29;1,78
52000;25,69;0,43;115,95;1,92
53000;33,68;0,62;156,22;2,41
54000;28,65;0,51;128,26;1,94
55000;29,35;0,49;130,13;2,23
56000;30,77;0,50;135,21;2,10
57000;36,81;0,63;164,63;2,61
58000;30,39;0,50;135,46;2,01
59000;30,87;0,51;139,15;2,20
60000;32,82;0,51;143,97;2,13
61000;51,82;1,01;232,42;3,65
62000;34,12;0,51;153,64;2,20
63000;42,49;0,65;192,80;2,87
64000;53,41;0,82;240,72;3,50
65000;66,48;1,04;293,11;4,37
66000;37,84;0,58;170,20;2,41
67000;38,98;0,56;175,65;2,43
68000;40,14;0,59;179,50;2,55
69000;41,49;0,60;182,38;2,61
70000;57,79;0,71;168,19;2,23

```
70000;57,79;0,71;168,19;2,23
71000;43,27;0,60;193,72;2,67
72000;44,34;0,61;201,35;2,78
73000;45,21;0,59;206,78;2,78
74000;46,41;0,65;210,61;2,80
75000;47,76;0,63;215,84;2,89
76000;49,64;0,66;221,49;2,84
77000;64,55;1,04;285,04;3,75
78000;54,03;0,75;243,57;3,12
79000;61,82;0,83;275,17;3,63
80000;54,53;0,73;243,40;3,15
81000;60,27;0,75;272,36;3,43
82000;56,39;0,71;252,05;3,28
83000;58,22;0,70;258,96;3,25
84000;58,78;0,72;260,39;3,30
85000;60,24;0,72;266,22;3,27
86000;65,32;0,77;287,10;3,61
87000;65,09;0,80;284,02;3,52
88000;90,83;1,06;401,16;5,06
89000;66,03;0,79;294,45;3,47
90000;80,69;0,93;357,48;4,37
91000;85,58;0,93;377,99;4,23
92000;83,42;0,87;363,52;4,17
93000;73,33;0,81;331,41;3,73
94000;73,44;0,84;323,84;3,74
95000;74,46;0,85;333,86;3,79
96000;84,13;0,93;379,37;4,40
97000;95,49;1,09;416,63;4,63
98000;86,61;0,87;386,00;4,52
99000;91,22;0,99;409,96;4,32
100000;88,03;1,02;408,63;4,15
```

4- The Conclusion

As we have seen the given outputs and plots, If we want to make a search comparison with the data structures it will be like that:

HashMap>Binary Search Tree>ArrayList>LinkedList