# Report of A.D.S. HW3

Abdullah AYDEĞER

040090533

This report is prepared for Advanced Data Structure 3.Homework

Istanbul Technical University

Advanced Data Structures

Zehra ÇATALTEPE

10.01.2012

# B-Trees

In this homework, I wrote codes for creating B-Tree and implementing its functions. Order of the B-Tree is wanted from command prompt. According to this given value, B-Tree is created by using given text file. After this, B-Tree search algorithm is called with randomized words which are in the given text for 17-19-…..-29 times. In this calling, I calculate the total time of the searcher implementation. Furthermore, I drew histogram by using the total time of all searcher implementations. All histograms are given below.

Prior to the drawing histograms, I run searcher implementation for 10 times with same order of tree and with same number of searching words.

## Histograms

### *For 17 times searching;*

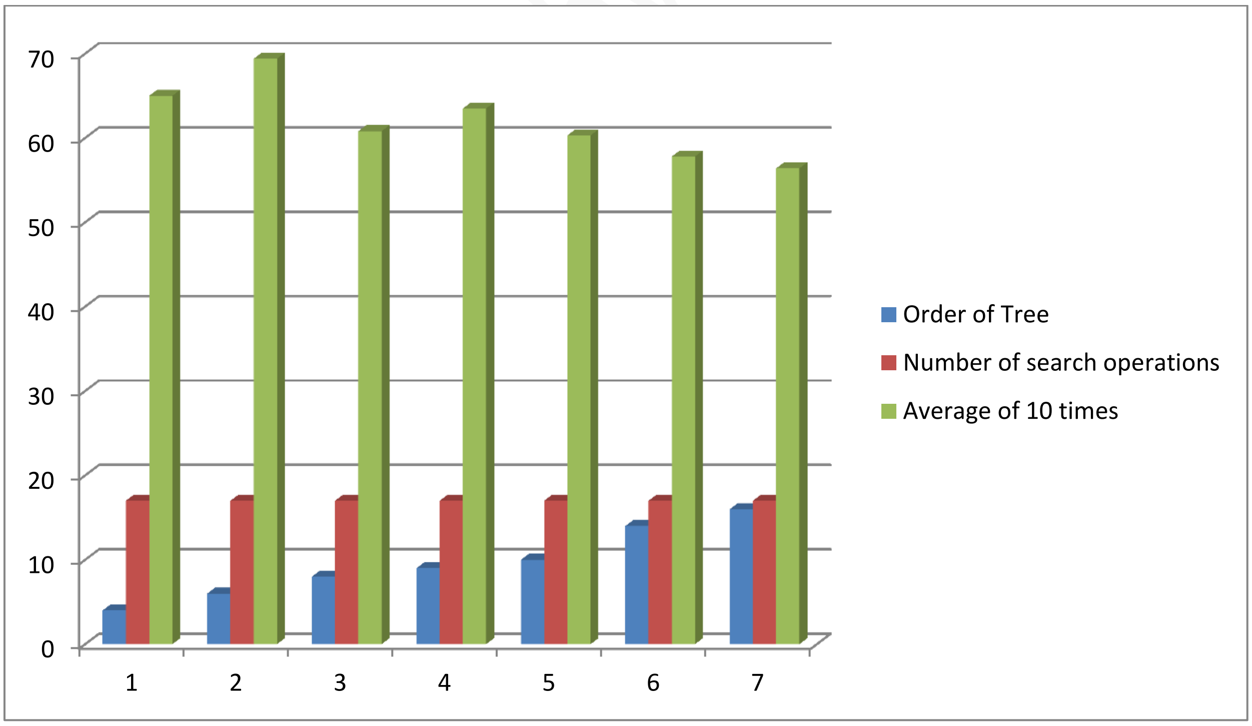


**Figure 1. # of Searching : 17**

| Order of Tree | Number of search operations | Runnig Times (ms): | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | Average of 10 times |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 17 | | 70 | 61 | 50 | 81 | 68 | 49 | 68 | 78 | 70 | 55 | 65 |
| 6 | 17 | | 82 | 72 | 75 | 53 | 79 | 60 | 74 | 65 | 73 | 61 | 69,4 |
| 8 | 17 | | 63 | 48 | 85 | 64 | 54 | 54 | 60 | 71 | 56 | 53 | 60,8 |
| 9 | 17 | | 54 | 67 | 66 | 53 | 64 | 61 | 57 | 69 | 88 | 56 | 63,5 |
| 10 | 17 | | 40 | 59 | 69 | 56 | 72 | 58 | 59 | 63 | 72 | 55 | 60,3 |
| 14 | 17 | | 62 | 55 | 65 | 55 | 56 | 73 | 55 | 39 | 56 | 62 | 57,8 |
| 16 | 17 | | 64 | 55 | 59 | 58 | 62 | 65 | 41 | 67 | 34 | 59 | 56,4 |

**Figure 2. Table of 10 times calling with 17 words**

*For 19 times searching;*

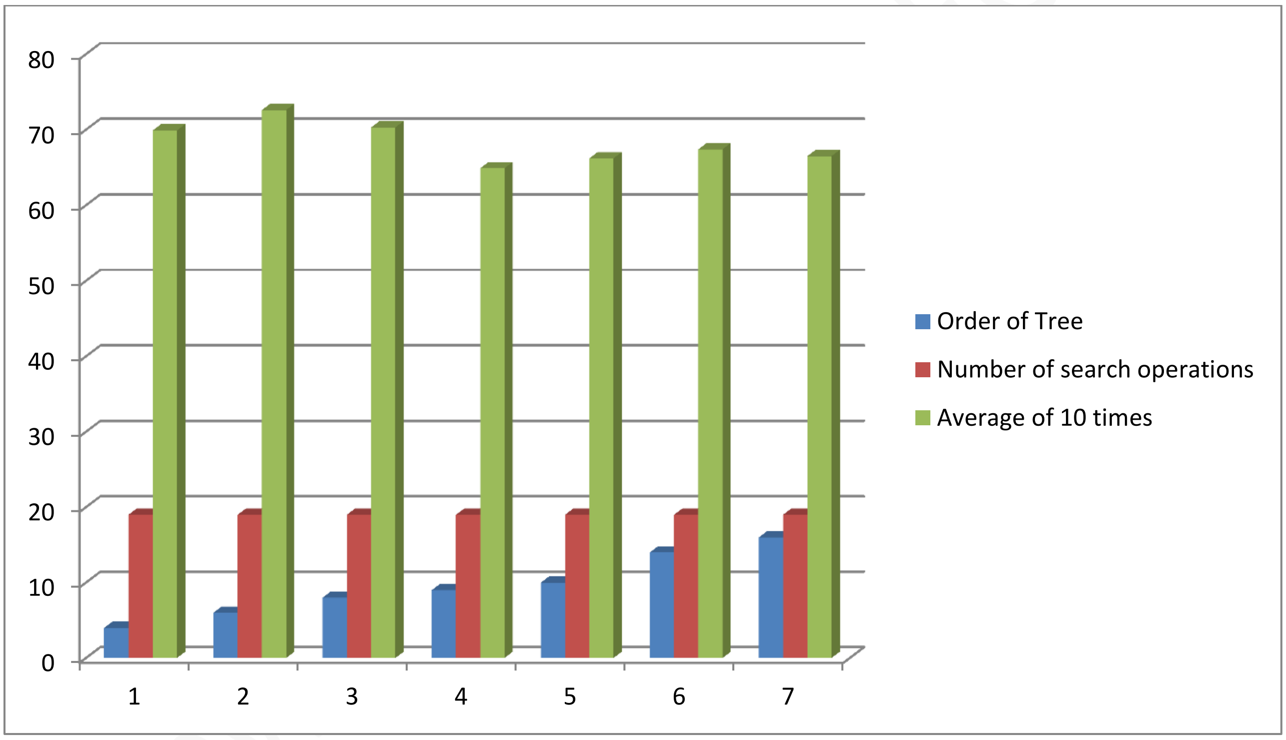


**Figure 3. # of Searching : 19**

| Order of Tree | Number of search operations | Runnig Times (ms): | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | Average of 10 times |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 19 | | 59 | 62 | 85 | 77 | 76 | 71 | 65 | 82 | 58 | 64 | 69,9 |
| 6 | 19 | | 83 | 70 | 66 | 60 | 73 | 94 | 69 | 76 | 66 | 69 | 72,6 |
| 8 | 19 | | 82 | 67 | 79 | 56 | 69 | 59 | 75 | 66 | 84 | 66 | 70,3 |
| 9 | 19 | | 59 | 67 | 64 | 65 | 69 | 57 | 58 | 84 | 61 | 65 | 64,9 |
| 10 | 19 | | 59 | 63 | 76 | 82 | 61 | 61 | 56 | 65 | 55 | 84 | 66,2 |
| 14 | 19 | | 69 | 96 | 60 | 71 | 60 | 63 | 73 | 59 | 69 | 54 | 67,4 |
| 16 | 19 | | 65 | 59 | 65 | 55 | 67 | 67 | 76 | 71 | 77 | 63 | 66,5 |

**Figure 4. Table of 10 times calling with 19 words**

*For 21 times searching;*

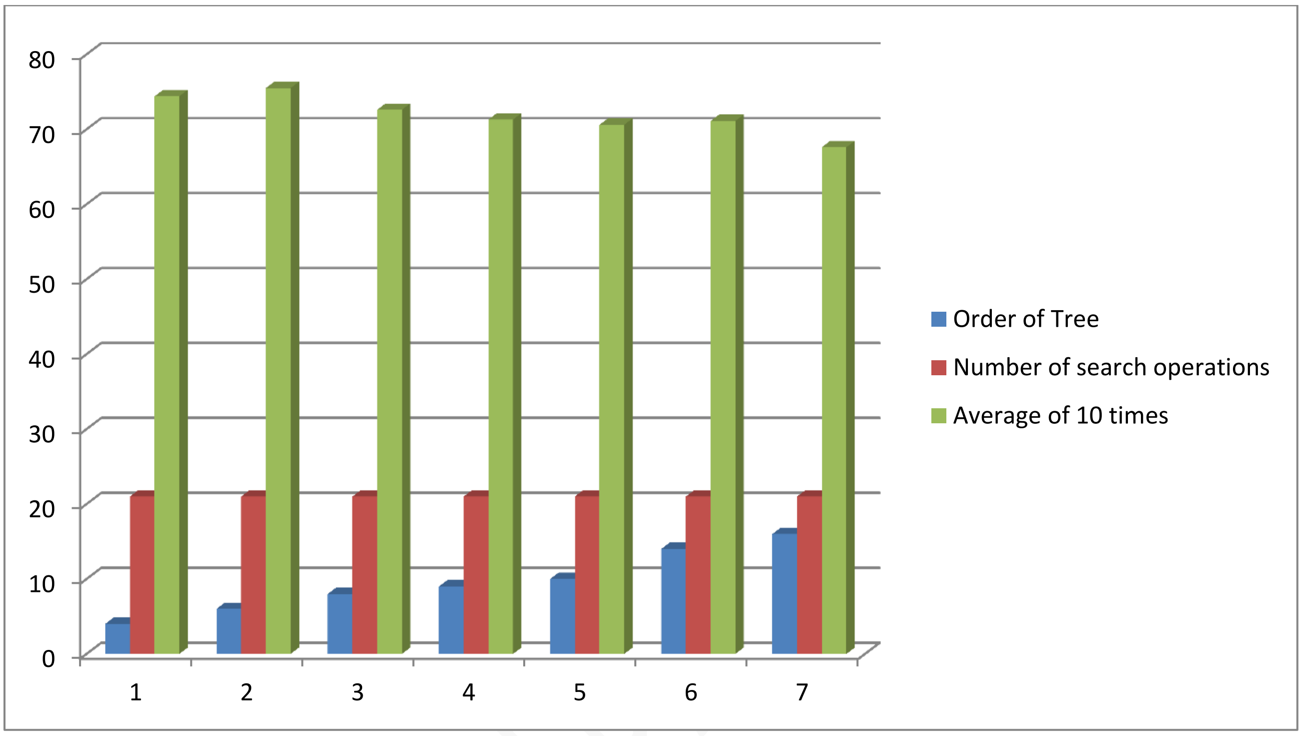


**Figure 5. # of Searching : 21**

| Order of Tree | Number of search operations | Runnig Times (ms): | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | Average of 10 times |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 21 |  | 85 | 76 | 75 | 76 | 68 | 81 | 80 | 63 | 71 | 69 | 74,4 |
| 6 | 21 |  | 70 | 66 | 83 | 89 | 78 | 68 | 79 | 70 | 71 | 81 | 75,5 |
| 8 | 21 |  | 63 | 58 | 75 | 76 | 99 | 68 | 73 | 75 | 70 | 69 | 72,6 |
| 9 | 21 |  | 71 | 64 | 75 | 78 | 80 | 71 | 68 | 72 | 72 | 62 | 71,3 |
| 10 | 21 |  | 62 | 78 | 71 | 70 | 58 | 69 | 77 | 67 | 74 | 80 | 70,6 |
| 14 | 21 |  | 63 | 84 | 61 | 66 | 69 | 73 | 64 | 79 | 85 | 67 | 71,1 |
| 16 | 21 |  | 64 | 76 | 74 | 70 | 61 | 69 | 60 | 63 | 63 | 76 | 67,6 |

**Figure 6. Table of 10 times calling with 21 words**

*For 23 times searching;*

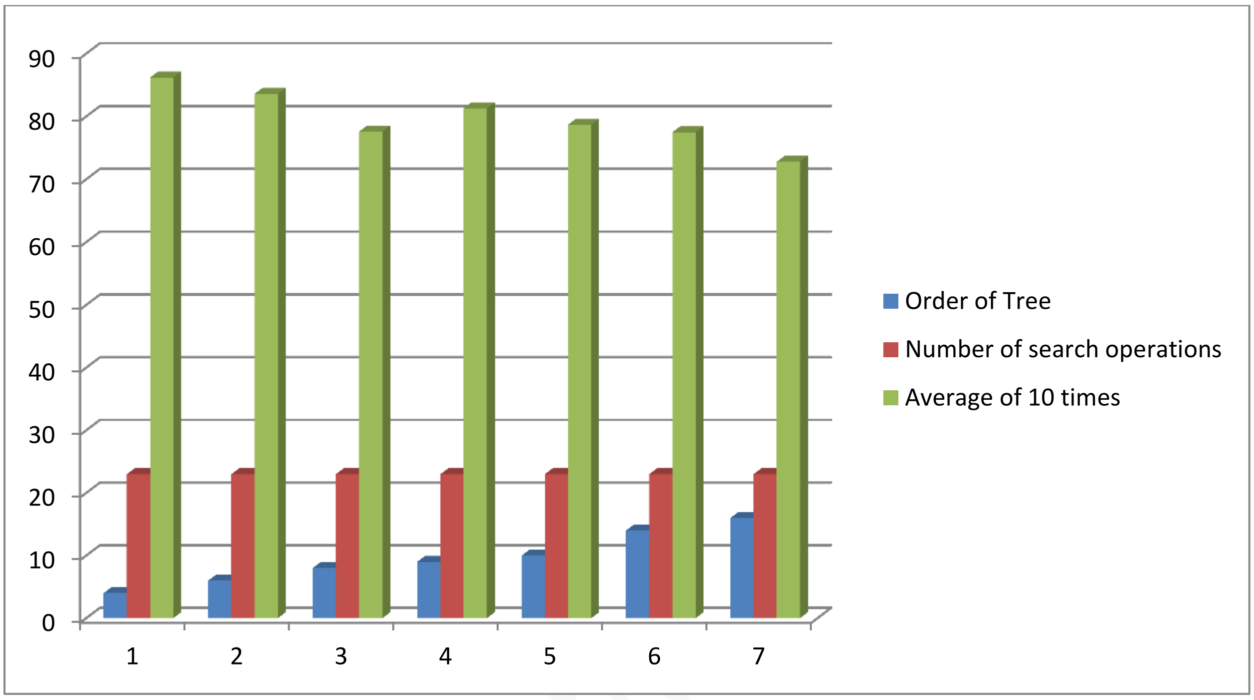


**Figure 7. # of Searching : 23**

| Order of Tree | Number of search operations | Runnig Times (ms): | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | Average of 10 times |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 23 | | 95 | 80 | 80 | 89 | 71 | 85 | 95 | 86 | 100 | 81 | 86,2 |
| 6 | 23 | | 89 | 86 | 87 | 74 | 78 | 87 | 92 | 84 | 87 | 72 | 83,6 |
| 8 | 23 | | 76 | 71 | 61 | 88 | 80 | 82 | 80 | 87 | 80 | 71 | 77,6 |
| 9 | 23 | | 81 | 78 | 85 | 84 | 68 | 81 | 76 | 76 | 86 | 98 | 81,3 |
| 10 | 23 | | 71 | 78 | 84 | 78 | 93 | 81 | 84 | 76 | 72 | 70 | 78,7 |
| 14 | 23 | | 77 | 74 | 73 | 96 | 71 | 81 | 69 | 83 | 76 | 75 | 77,5 |
| 16 | 23 | | 69 | 65 | 69 | 78 | 73 | 72 | 79 | 71 | 75 | 77 | 72,8 |

**Figure 8. Table of 10 times calling with 23 words**

*For 25 times searching;*

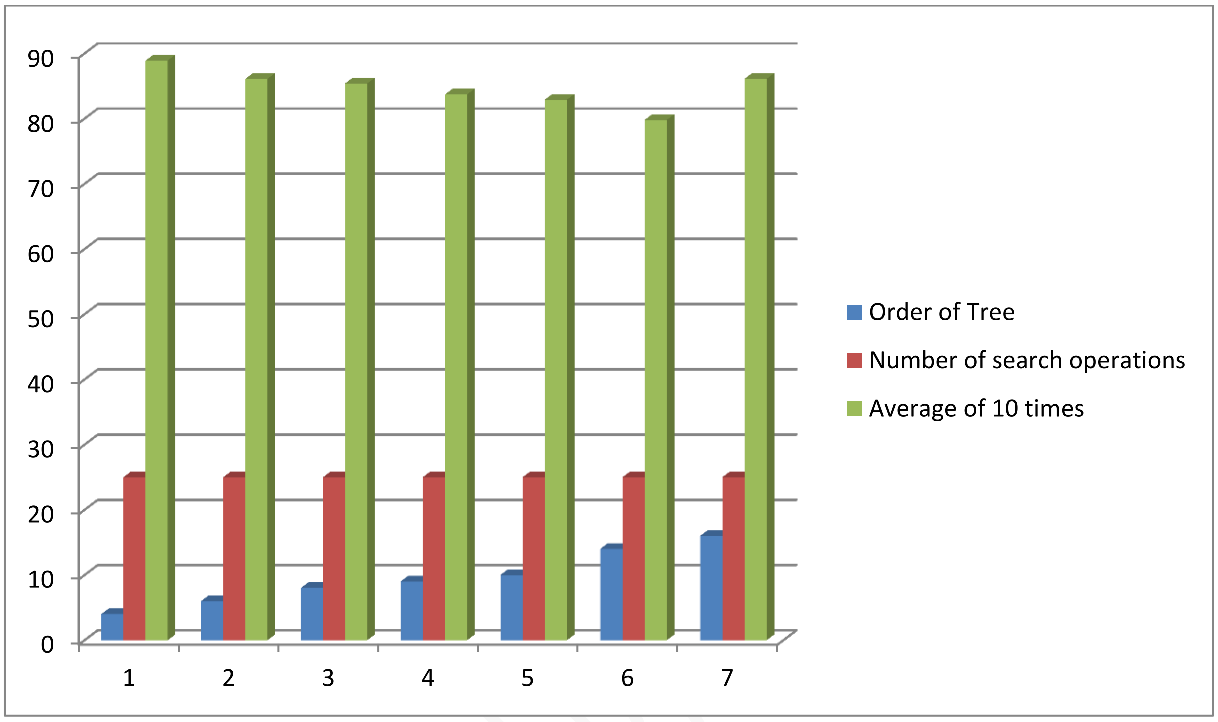


**Figure 9. # of Searching : 25**

| Order of Tree | Number of search operations | Runnig Times (ms): | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | Average of 10 times |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 25 | | 80 | 92 | 80 | 96 | 79 | 109 | 85 | 77 | 98 | 93 | 88,9 |
| 6 | 25 | | 86 | 88 | 87 | 85 | 81 | 86 | 93 | 78 | 83 | 94 | 86,1 |
| 8 | 25 | | 75 | 84 | 79 | 95 | 79 | 84 | 98 | 93 | 81 | 86 | 85,4 |
| 9 | 25 | | 83 | 82 | 74 | 86 | 95 | 87 | 84 | 82 | 82 | 82 | 83,7 |
| 10 | 25 | | 82 | 75 | 88 | 88 | 83 | 82 | 80 | 84 | 85 | 82 | 82,9 |
| 14 | 25 | | 74 | 83 | 70 | 60 | 139 | 50 | 70 | 83 | 99 | 70 | 79,8 |
| 16 | 25 | | 100 | 90 | 83 | 96 | 88 | 78 | 80 | 83 | 84 | 79 | 86,1 |

**Figure 10. Table of 10 times calling with 25 words**

*For 27 times searching;*

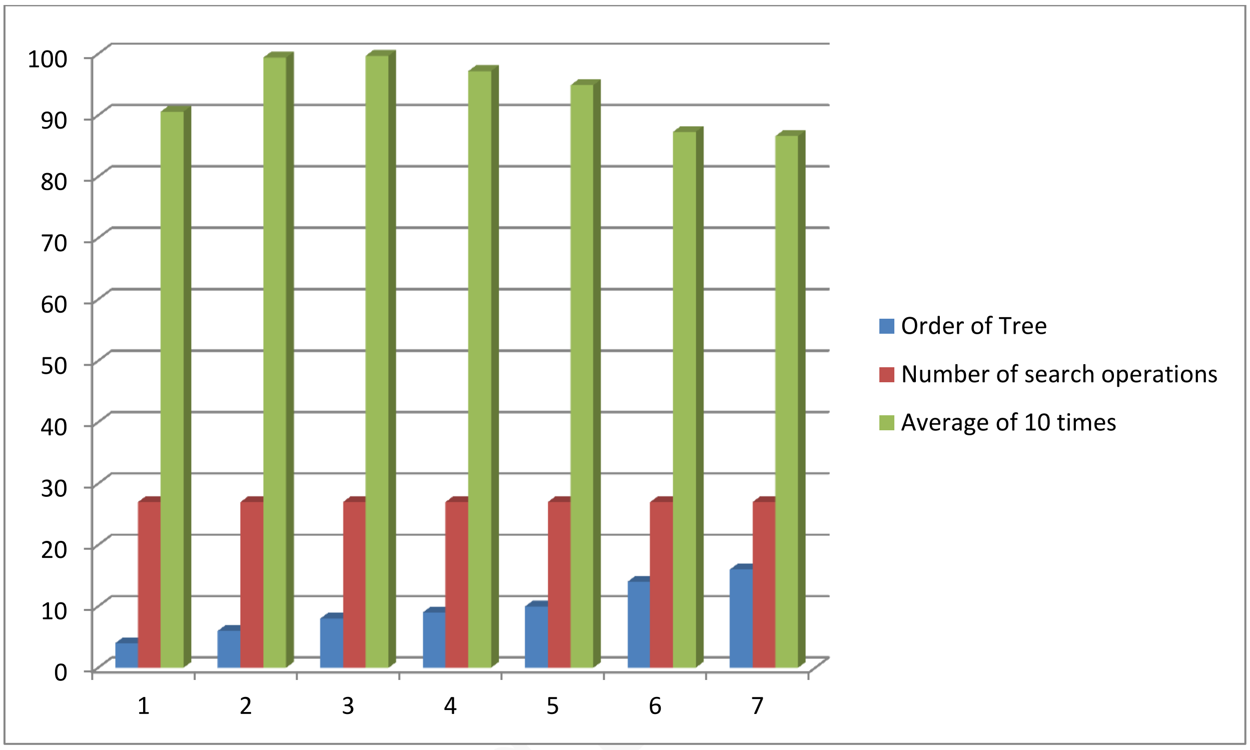


**Figure 11. # of Searching : 27**

| Order of Tree | Number of search operations | Runnig Times (ms): | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | Average of 10 times |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 27 | | 87 | 94 | 84 | 95 | 89 | 107 | 82 | 86 | 103 | 79 | 90,6 |
| 6 | 27 | | 120 | 90 | 96 | 103 | 93 | 101 | 105 | 86 | 102 | 98 | 99,4 |
| 8 | 27 | | 100 | 90 | 115 | 93 | 107 | 99 | 92 | 99 | 95 | 107 | 99,7 |
| 9 | 27 | | 102 | 90 | 91 | 111 | 96 | 90 | 105 | 103 | 100 | 84 | 97,2 |
| 10 | 27 | | 91 | 90 | 96 | 97 | 88 | 87 | 102 | 107 | 92 | 99 | 94,9 |
| 14 | 27 | | 90 | 93 | 83 | 90 | 90 | 90 | 83 | 87 | 82 | 85 | 87,3 |
| 16 | 27 | | 83 | 88 | 95 | 88 | 82 | 84 | 75 | 88 | 94 | 89 | 86,6 |

**Figure 12. Table of 10 times calling with 27 words**

*For 29 times searching;*

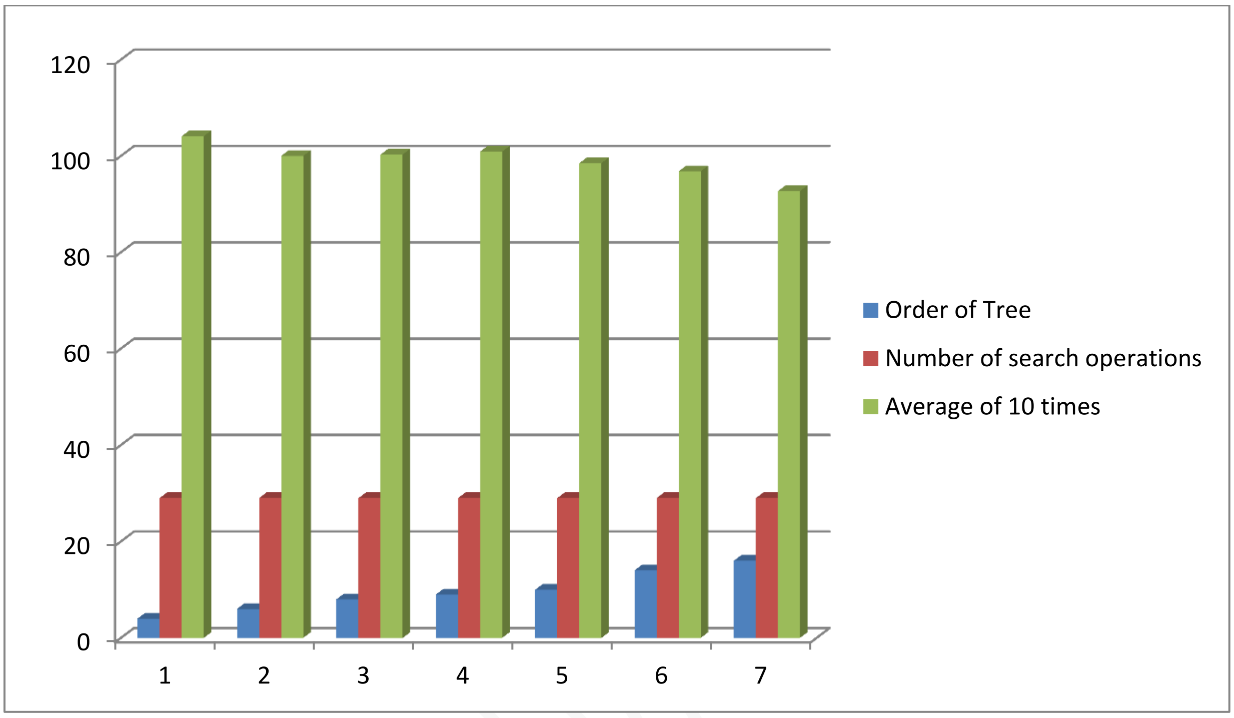
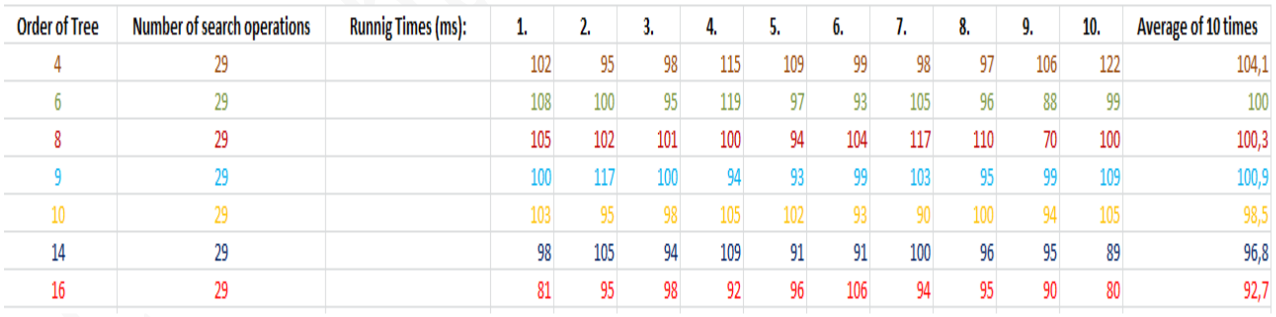


Figure 13. # of Searching : 29



| Order of Tree | Number of search operations | Runnig Times (ms): | 1. | 2. | 3. | 4. | 5. | 6. | 7. | 8. | 9. | 10. | Average of 10 times |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 29 | | 102 | 95 | 98 | 115 | 109 | 99 | 98 | 97 | 106 | 122 | 104,1 |
| 6 | 29 | | 108 | 100 | 95 | 119 | 97 | 93 | 105 | 96 | 88 | 99 | 100 |
| 8 | 29 | | 105 | 102 | 101 | 100 | 94 | 104 | 117 | 110 | 70 | 100 | 100,3 |
| 9 | 29 | | 100 | 117 | 100 | 94 | 93 | 99 | 103 | 95 | 99 | 109 | 100,9 |
| 10 | 29 | | 103 | 95 | 98 | 105 | 102 | 93 | 90 | 100 | 94 | 105 | 98,5 |
| 14 | 29 | | 98 | 105 | 94 | 109 | 91 | 91 | 100 | 96 | 95 | 89 | 96,8 |
| 16 | 29 | | 81 | 95 | 98 | 92 | 96 | 106 | 94 | 95 | 90 | 80 | 92,7 |

Figure 14. Table of 10 times calling with 29 words

Histograms are seen as expected. If order value “m” is greater, then the running time of searching is smaller. Because of the fact that I used greater “m” value, I allocated more memory space for implementing B-Tree. But sometimes, the running time is not decrementing while m is increasing. I think this is caused by randomized search. Randomized search means any of the word in the sonnets can be chosen. Most important difference between words are  some words are in the leaves and can be searched slower.

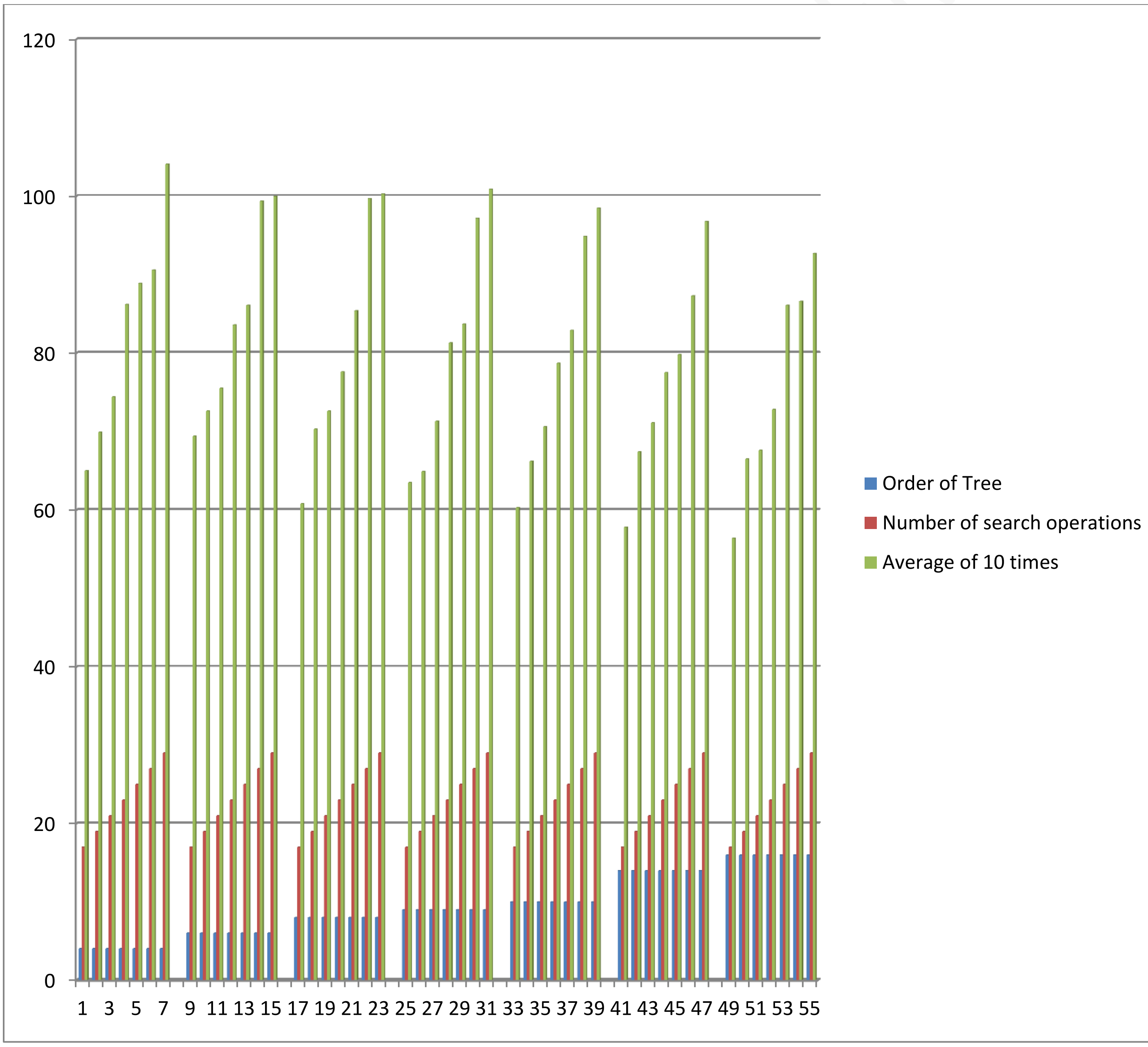


Figure 15. All of searching implementations

Calling searcher implementations with more words causes more running time as seen in the Figure 15. This is expected information and does not require any explanation.

## Random Words

I used randomized words in the searcher implementation. I use some functions for getting randomized words. I'll explain this by showing necessary codes.

```
i = rand()%8 +1;
j = rand()%(sentence-1) +1;

t1 = clock();
NodeData nd = BT->B_TREE_SEARCH(BT->root, BT->readWord(fileName, j, i));
t2 = clock();
```

Firstly, I used rand function for creating random numbers. After this, modulo i value which represent column value by "8" since I assume that column numbers are between 1-8. Furthermore, modulo  j value by number of sentence -1 for getting valid sentence number. And add they "1" for do not getting "0". Then these integers are sent to readWord function as parameters.

```
string BTree::readWord(string fileName, int s, int c){
    /*
    * Takes fileName to read, and two integers.
    * According to these two integers, finding the word in the place s. sentence & c.column
    * and returning it back
    */
```

This readWord function find the requested word in the sonnet and return it back.

After all these operations B-Tree-Search implementation works with randomized words.

## Open Issue

I have a lot of problems before the writing codes. But with necessary explorations I overcome most of these. Now I have a problem for calculating the running time of searcher implementation. When I want to calculate running time of searcher implementation, I calculate both searcher and my readWord function running time and add these two values totalTime variable. Therefore I did not calculate and report only the running time of searcher implementation, but also readWord function too. This causes more running time value in the screen which is not real running time. In contravention of these, I think that issue is not important as comparing to other searcher implementation with different order value. This problem can be solved very easily. But if I change this, I need to calculate all running times again and draw all histograms again too. Therefore, I can not spend more time fort his.