LINEAR SEARCH VS BK TREE TRAVERSAL IMPLEMENTATION ANALYSIS & COMPARISON

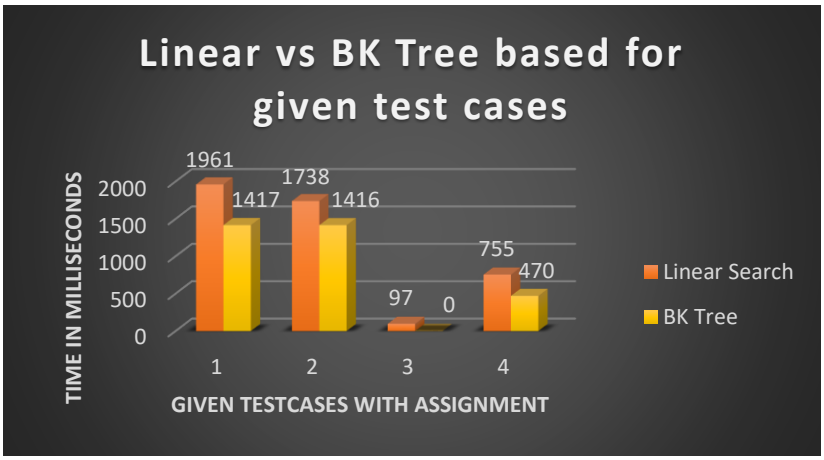
LINEAR SEARCH

```
//Check the possible correct words from the dictionary
dictHashMap.forEach((dictKey, dictValue) -> {
    currEditDistance[0] = getEditDistance(dictKey.toString(), wordToCheck);
    checkEditDist(writer,strList,dictKey.toString(),currEditDistance[0],maxDistValue);
});
```

BK TREE

```
//For all the childnodes in the range
for (Iterator<Node> currNode = childNode.childNodes.iterator(); currNode.hasNext();) {
    Node iterNode = currNode.next();
    //Check whether the given node is within the tol range
    //if true
    if (isWithinRange(iterNode.cost,higherLimit,lowerLimit)) {
        //Traverse for its child nodes
        traverseNodes(possibleCorrectWords,iterNode, wordToCheck, maxEditDist);
    }
}
```

Linear search is implemented using the normal search technique, where as the BK tree traversal is a recursive implementation and analysis is done using recurrence inclusive of edit distance computation.



Below is a general comparison of Linear Search and BK tree implementation based on below 3 parameters for various input sample data and dictionary, in a machine with configuration 8GB and i5 core processor.

- The size and type of dictionary.
- The length of search words.
- The maximum distance value.

The size and type of dictionary

Samples used : 3 test cases :

- Randomly generated words + The words in given test case.
- Word mixer of various geographic places + The words in given test case.
- Randomly generated mixer words + The words in given test case.

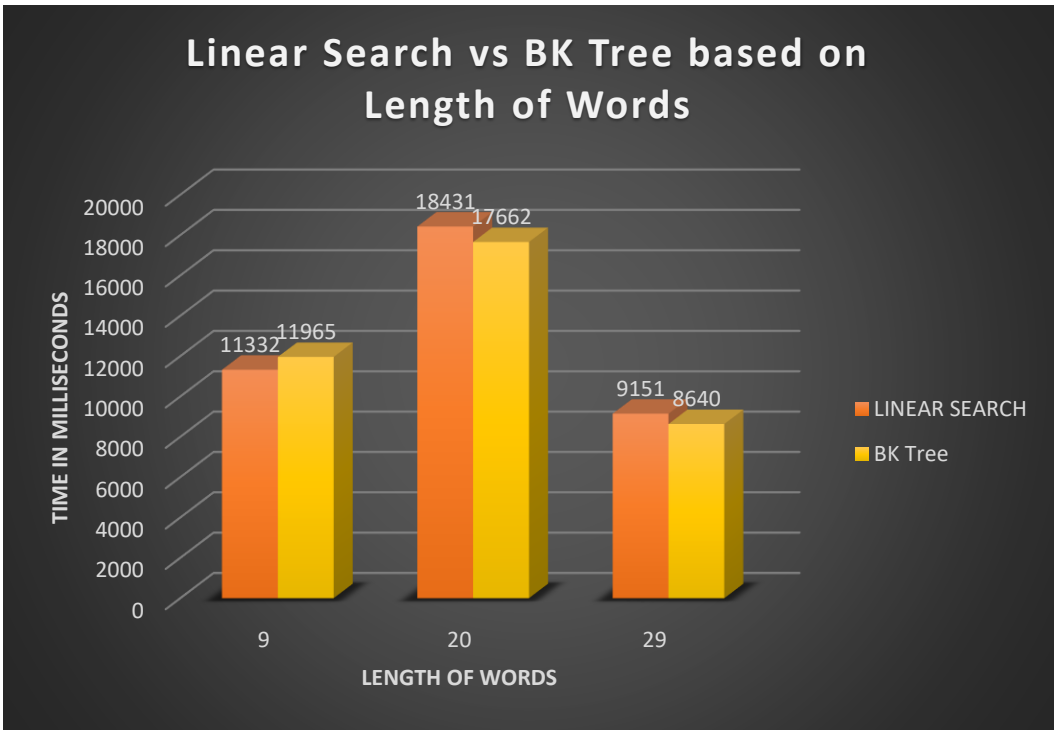
The BK tree implementation seems to perform fairly well when compared to Linear search when the dictionary size is large, for example : the third dataset consisting of 120000 words.

The Length of Words

Samples used : 3 test cases :

- Contains word of maximum length 9.
- Contains word of maximum length 20.
- Contains word of maximum length 29.

As the length of words increases, BK tree performs a little better than the Linear search.

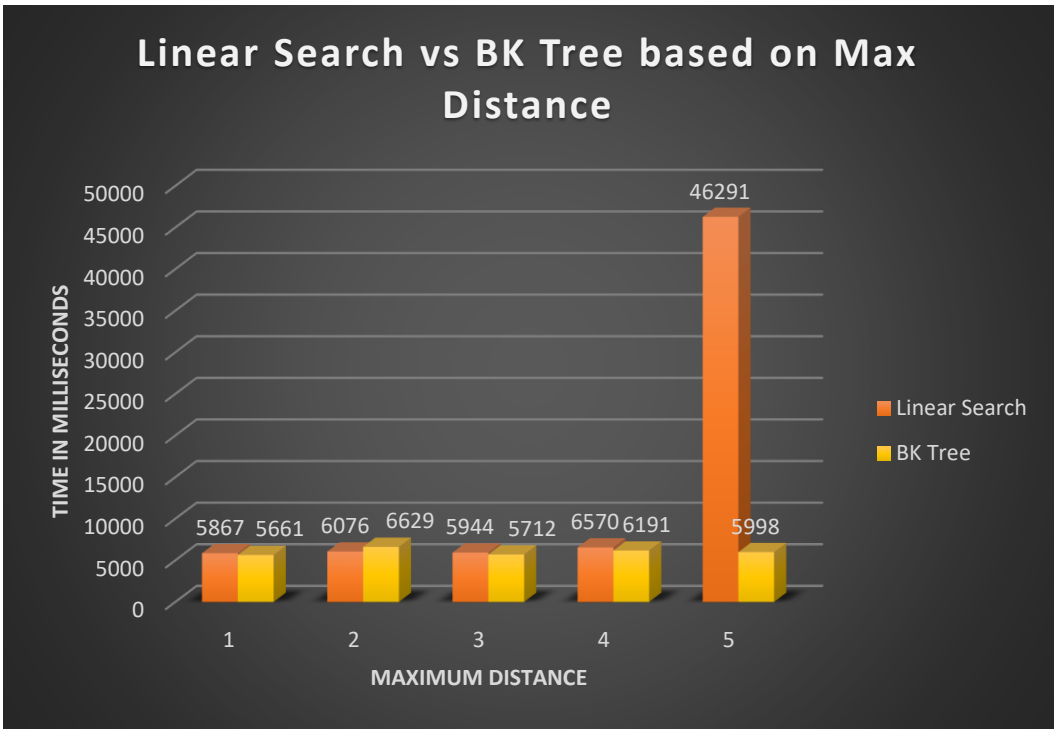


MAX DISTANCE

Samples used : 3 test cases :

- With maximum distance value 1.
- With maximum distance value 2.
- With maximum distance value 3.
- With maximum distance value 4.
- With maximum distance value 5.

BK tree complexity depends mostly on the maximum distance value only.



BK Tree Time Complexity:

Time complexity totally depends on the maximum distance value, as the maximum distance value is high, Linear search takes almost 8 times the time taken by the BK tree. So this shows, it depends mostly on the tolerance value or the maximum distance value.

Depth of the tree : log n, where n is the number of nodes(or the number of words in dictionary)

Edit distance complexity : length of current node * length of the misspelled word.

Complexity of BK tree : O(length of the current node * length of the misspelled word * logn)

Linear Search Complexity:

Complexity of Linear Search = O(number of misspelled words * number of words in the dictionary)

EDIT DISTANCE OBSERVATIONS :

		M	E
	0	1	2
G	1	1	2
R	2	2	2

Computation of edit distance of ME and GR words, insert, remove and replace are the key operations. Considering any two of the operations would result in a greater edit distance value.

- If only insert and remove are considered, then the final cost of the edit distance would be 4.
- If only remove and replace are considered, taking GR as string1 and ME as string2, we would be getting a cost of 3.
- If only insert and replace are considered, taking ME as string1 and GR as string2 we would be getting a cost of 3.
- If all 3 operations insert, remove and replace are considered, we get the best edit distance of 2.