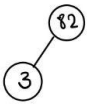
CS 202, Fall 2023
Homework 1 - Binary Search Trees
Deniz Polat
22103590
Section 001

Question 1:

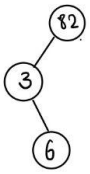
a)

* 1: Insert 82

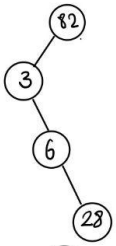
* 2: Insert 3



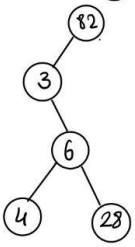
* 3: Insert 6:



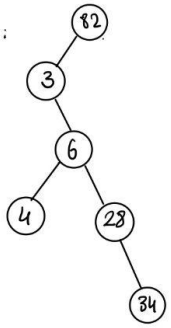
* 4: Insert 28



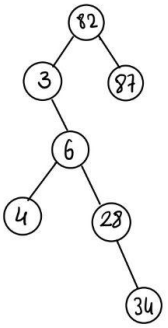
* 5: Insert 4



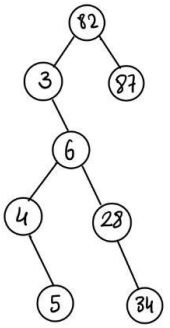
*6: Insert 34:



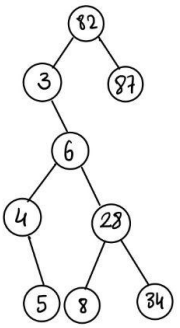
* 7: Insert 89



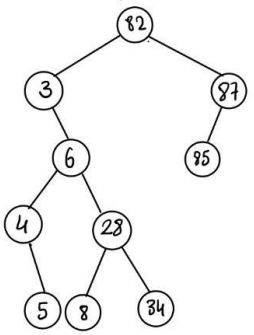
* 8: Insert 5



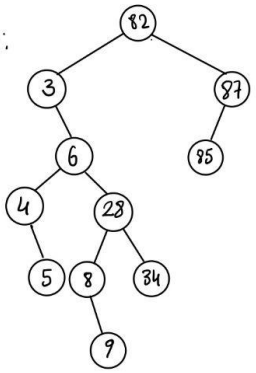
* 9: Insert 8



* 10: Insert 85:



* 11: Insert 9:



b) *Preorder Traversal: (Root - Left - Right)

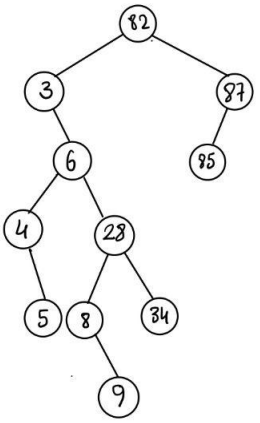
82, 3, 6, 4, 5, 28, 8, 9, 34, 87, 85

*Inorder Traversal (Left - Root - Right)

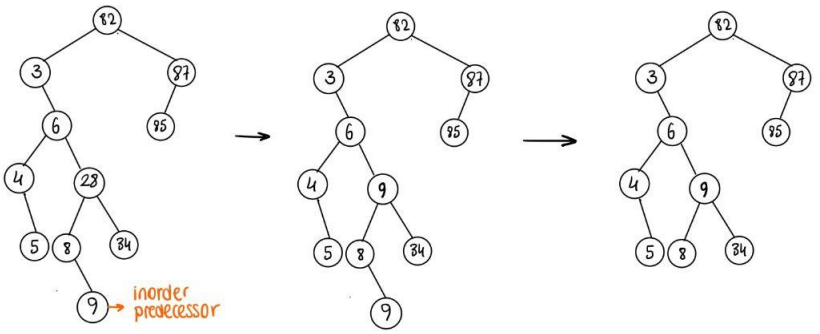
4, 5, 6, 8, 9, 28, 34, 82, 85, 87

* PostOrder Traversal: (Left - Right - Root)

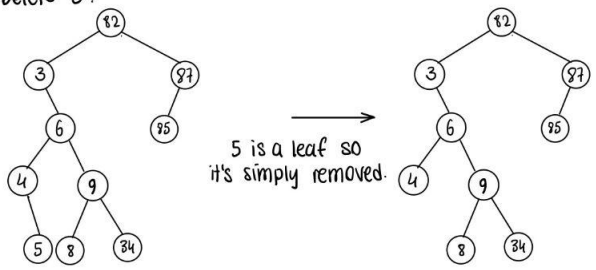
5, 4, 9, 8, 34, 28, 6, 3, 85, 87, 82



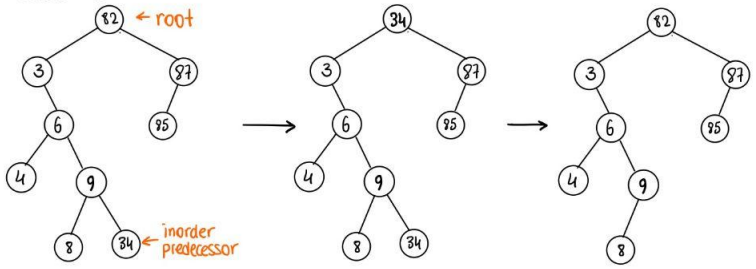
c) 1- Delete 28:



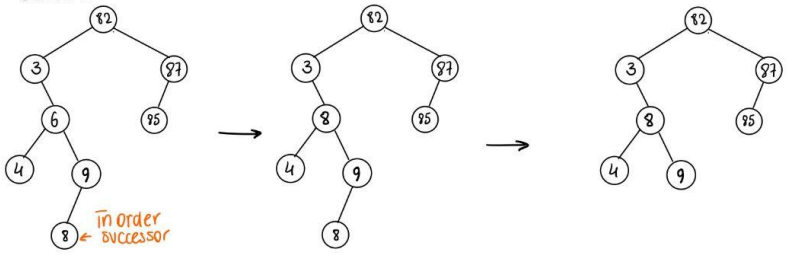
2- Delete 5:



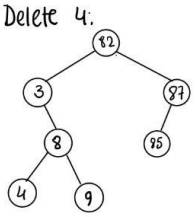
3: Delete 82:



4: Delete 6



5. Delete 4:



4 is now a leaf, simply removed.

d) Recursive pseudocode implementation for finding the minimum element in a BST:

```
FindMinimum(node)
  IF (node.left=NULL) THEN
     RETURN node.data
  ENDIF
  RETURN FindMinimum(node.left)
```

e) maximum height of a BST of n items: n.
minimum height of a BST of n items: $\lceil \log_2(n+1) \rceil$.

Question 3:

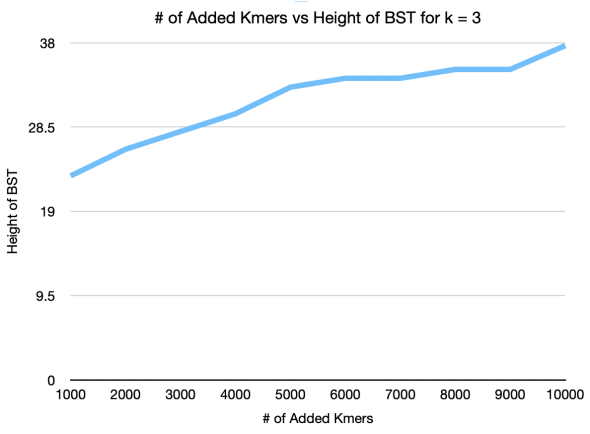
After implementing a function for time analysis, I generated two Kmer Trees for the values of k = 3, and 5. I used a randomly generated .txt file input with 45000 characters. Here are the terminal outputs for the values of k = 3, and k = 5.

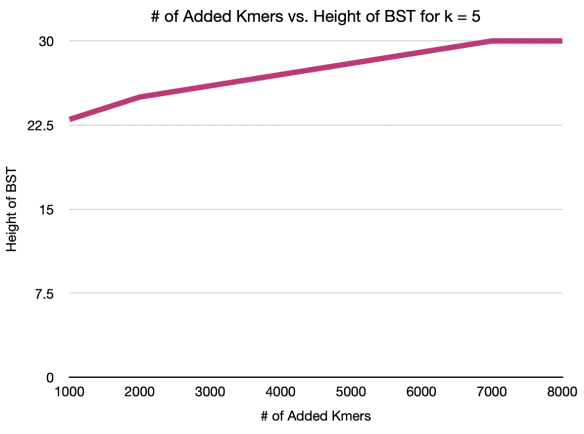
```
-------------------- k = 3 --------------------
----------------Added 1000 kmers----------------
The number of unique kmers: 985
The height of the BST: 23
Time taken for the period: 0.003484
----------------Added 2000 kmers----------------
The number of unique kmers: 1895
The height of the BST: 26
Time taken for the period: 0.007318
----------------Added 3000 kmers----------------
The number of unique kmers: 2764
The height of the BST: 28
Time taken for the period: 0.011394
----------------Added 4000 kmers----------------
The number of unique kmers: 3602
The height of the BST: 30
Time taken for the period: 0.015465
----------------Added 5000 kmers----------------
The number of unique kmers: 4402
The height of the BST: 33
Time taken for the period: 0.019232
----------------Added 6000 kmers----------------
The number of unique kmers: 5125
The height of the BST: 34
Time taken for the period: 0.023121
----------------Added 7000 kmers----------------
The number of unique kmers: 5803
The height of the BST: 34
Time taken for the period: 0.026943
----------------Added 8000 kmers----------------
The number of unique kmers: 6452
The height of the BST: 35
Time taken for the period: 0.030428
----------------Added 9000 kmers----------------
The number of unique kmers: 7073
The height of the BST: 35
Time taken for the period: 0.03415
----------------Added 10000 kmers----------------
The number of unique kmers: 7647
The height of the BST: 35
Time taken for the period: 0.037683
```

```
-------------------- k = 5 --------------------
----------------Added 1000 kmers----------------
The number of unique kmers: 1001
The height of the BST: 23
Time taken for the period: 0.001566
----------------Added 2000 kmers----------------
The number of unique kmers: 2001
The height of the BST: 25
Time taken for the period: 0.003565
----------------Added 3000 kmers----------------
The number of unique kmers: 3001
The height of the BST: 26
Time taken for the period: 0.005798
----------------Added 4000 kmers----------------
The number of unique kmers: 4000
The height of the BST: 27
Time taken for the period: 0.008071
----------------Added 5000 kmers----------------
The number of unique kmers: 4999
The height of the BST: 28
Time taken for the period: 0.010404
----------------Added 6000 kmers----------------
The number of unique kmers: 5999
The height of the BST: 29
Time taken for the period: 0.012851
----------------Added 7000 kmers----------------
The number of unique kmers: 6999
The height of the BST: 30
Time taken for the period: 0.015469
----------------Added 8000 kmers----------------
The number of unique kmers: 7999
The height of the BST: 30
Time taken for the period: 0.018052
```

Then I plotted six graphs in total, showing the behavior of the height, the number of nodes, and elapsed time to generate the tree of a binary search tree, i.e., a Kmer Tree.

The graphs of the height of the binary search trees and the number of added k-mers are shown below, for k = 3 and k = 5 respectively.

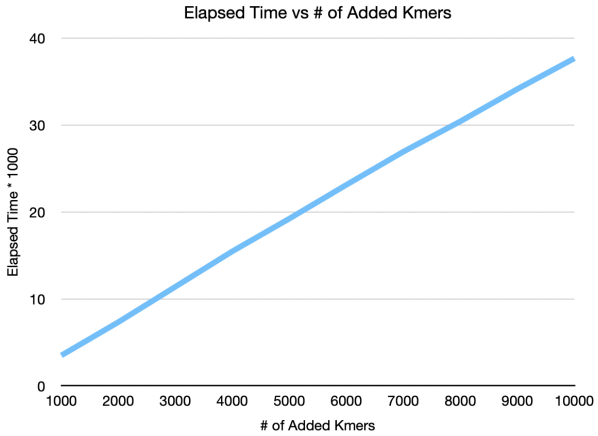


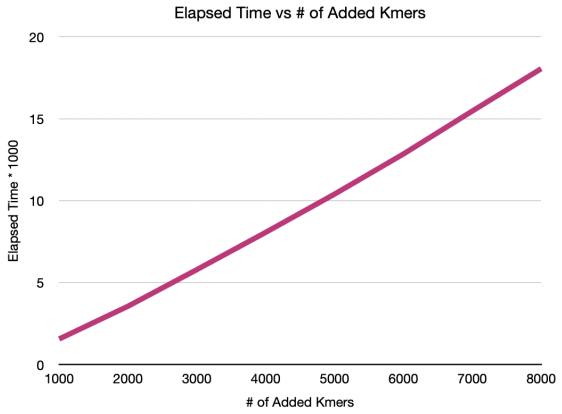


As it can be seen, a logarithmic behavior is observed in the change of height of BST as 1000, 2000, 3000, …, 10000 k-mers are inserted.

The function that gives the minimum height of a binary search tree is $log_2(n + 1)$, and the maximum is (n-1). So, we know that the height of the BST that is generated will be between these values. So the logarithmic relation is observed as expected since all the values are random and the implementation inserts new elements to empty nodes.

The graphs of the elapsed time while generating the BST and the number of added k-mers are shown below, for k = 3 and k = 5 respectively.





As it can be interpreted from the graphs, the behavior is linear, hence resulting in a time complexity of O(n). We know that the worst-case scenario of insertion in a BST has a time complexity of O(n), and the average is O(n*log(n)). The input file that the k-mers are generated consists of random characters, which has resulted in giving the worst-case results. We can say that the results are also as they were expected.

However, if sorted k-mers were being added to the tree, it would give a best-case scenario which is O(1), since the tree is initially empty and the k-mers would be inserted one by one. This situation does not depend on the height of the BST, and no search would be done.