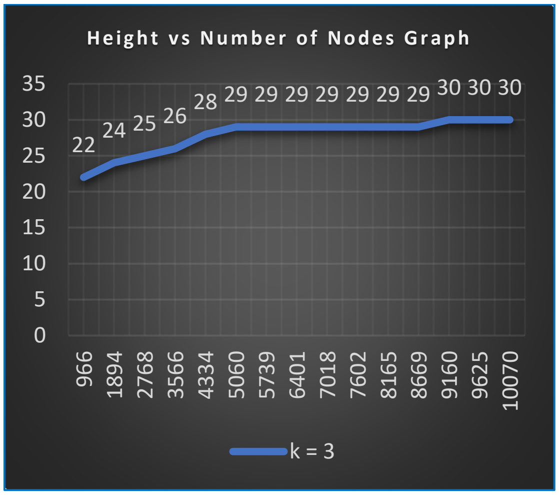
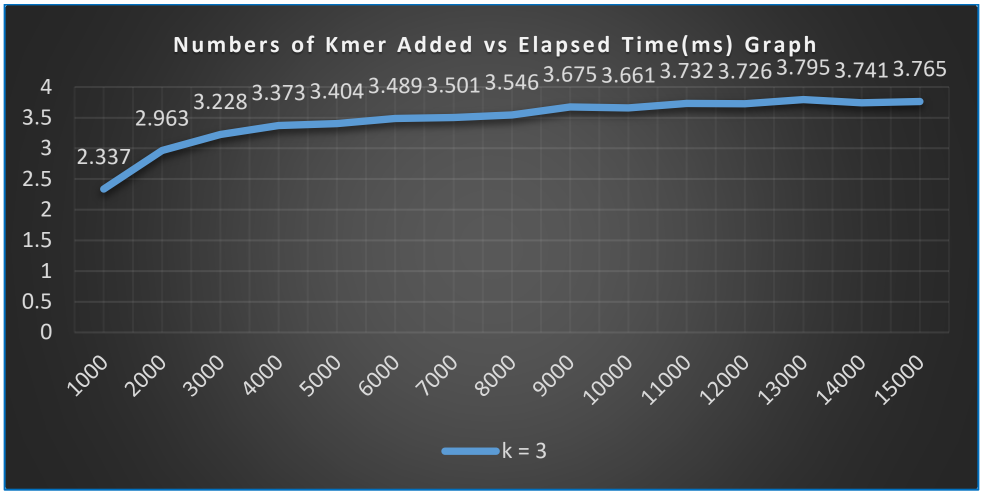
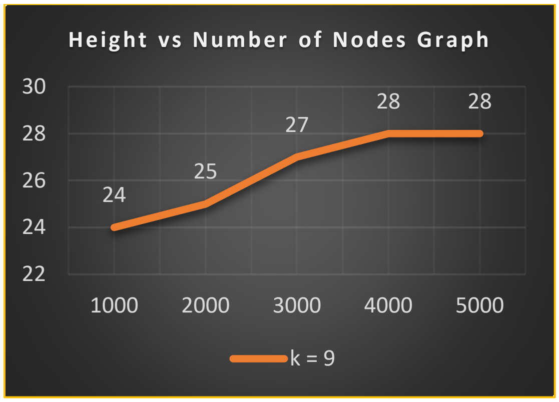
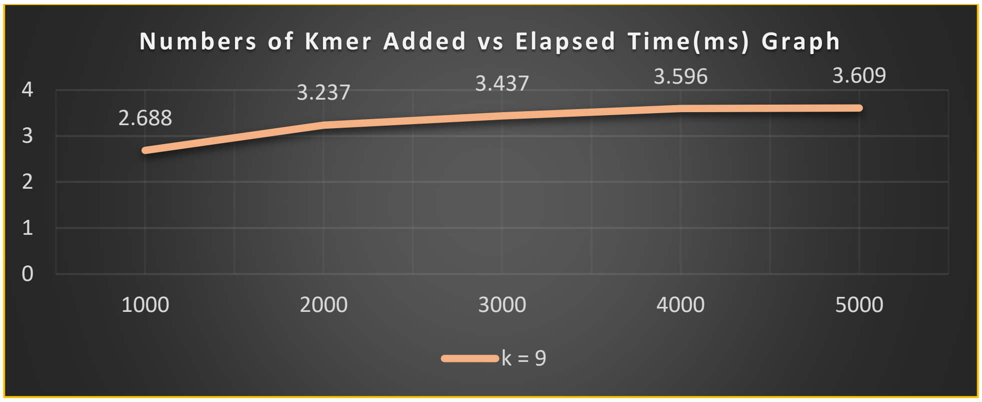
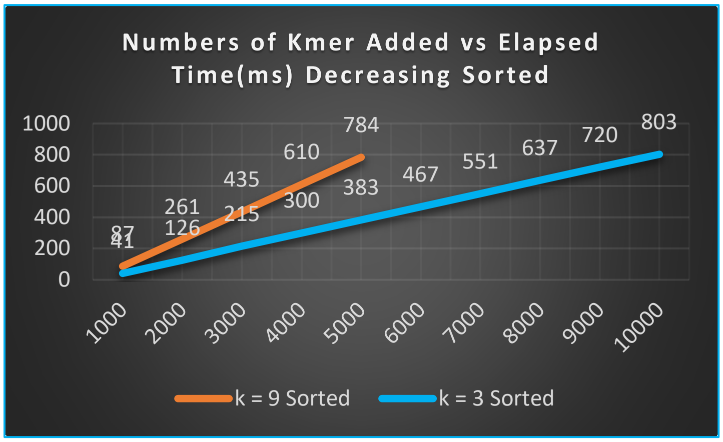
// Seçkin Alp Kargı – 22001942- CS202/1 – Hw2 – Time Analysis











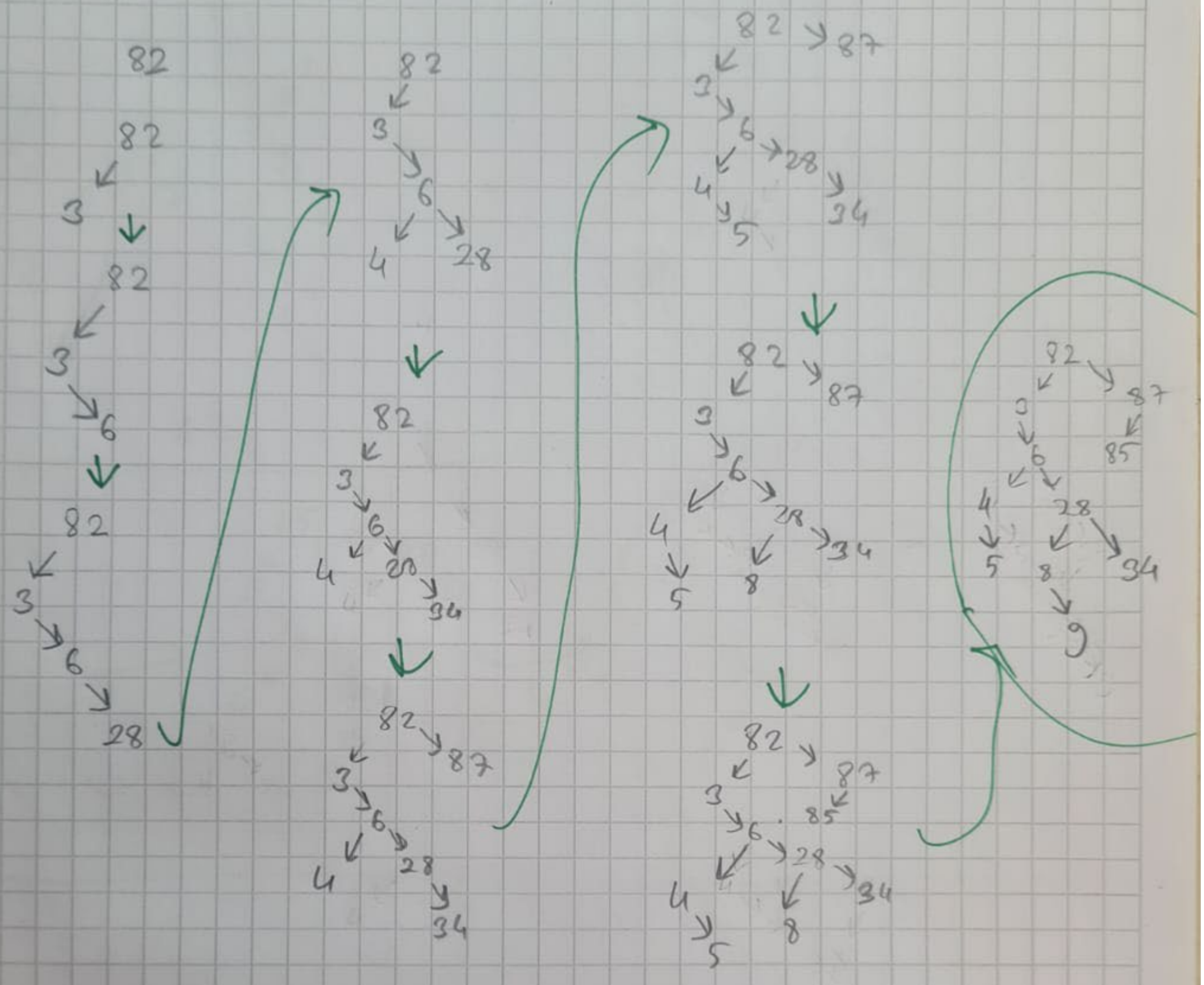
I used a randomly generated text file with a length of 45,000 characters for the analysis of k = 3 and k = 9. After that, I performed an inorder traversal of the trees and obtained decreasing sorted text files for 3-mers and 9-mers. During the inorder traversal, I added the kmers according to their counts. For example, if there are two occurrences of "aab" in the text file, the sorted analysis would have "aab" listed twice.

1) The relationship between the height of a BST and the number of nodes is generally logarithmic. As the number of nodes in the BST increases, the height of the tree tends to increase at a logarithmic rate. Specifically, a balanced BST is expected to have a height of O(log n), where n is the number of nodes in the tree. In my analysis using randomly generated text files, I expected a logarithmic relationship between the number of nodes and the height, and my results confirmed this expectation. However, when I used sorted text files, the height of the tree was equal to the number of nodes, resulting in a linear relationship. This is because a sorted text file can lead to a heavily skewed BST, where the tree is like a linked list with all nodes on one side. Thus, the expected linear relationship between the number of nodes and the height was observed in my analysis. Additionally, when comparing k = 3 and k = 9, I expected the height to be slightly lower for k = 3. This is because finding the same 3-mers is more likely than finding the same 9-mers due to the smaller search space (1/26^3 vs. 1/26^9). Consequently, the count of unique kmers is lower for k = 3, resulting in a generally lower height. The logarithmic shape of the graph in both cases is consistent with expectations.
2)I expected the time complexity of the randomly generated part to be O(log n), and my results confirmed this expectation. The time complexity of the BST operations is proportional to the height of the tree because the recursive functions traverse the tree that many times. Therefore, the time complexity is logarithmic, as expected. Furthermore, I anticipated that using sorted kmers into the BST would change the time complexity. When inserting sorted kmers, the BST can become heavily skewed, leading to a worst-case time complexity of O(n) for all operations. This happpens when the BST degenerates into a linked list with all nodes on one side. I tested this situation, and indeed, the elapsed time showed a linear relationship with the number of kmers inserted. Comparing k = 3 and k = 9 in the sorted case, k = 3 was faster because the number of uniquely added kmers was lower, resulting in an O(n) time complexity. This observation was consistent with my expectations. Similarly, in the randomly generated case, k = 3 was slightly faster due to the O(log n) time complexity and lower count of uniquely added kmers. I used decreasing sorted text, if I use increasing sorted text file, the result would still same and be O(n) time complexity.

Seçkin Alp Kargı -22001942

Part A.

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



Part B.

Preorder: 82 → 3 → 6 → 4 → 5 → 28 → 8 → 9 → 34 → 87 → 85

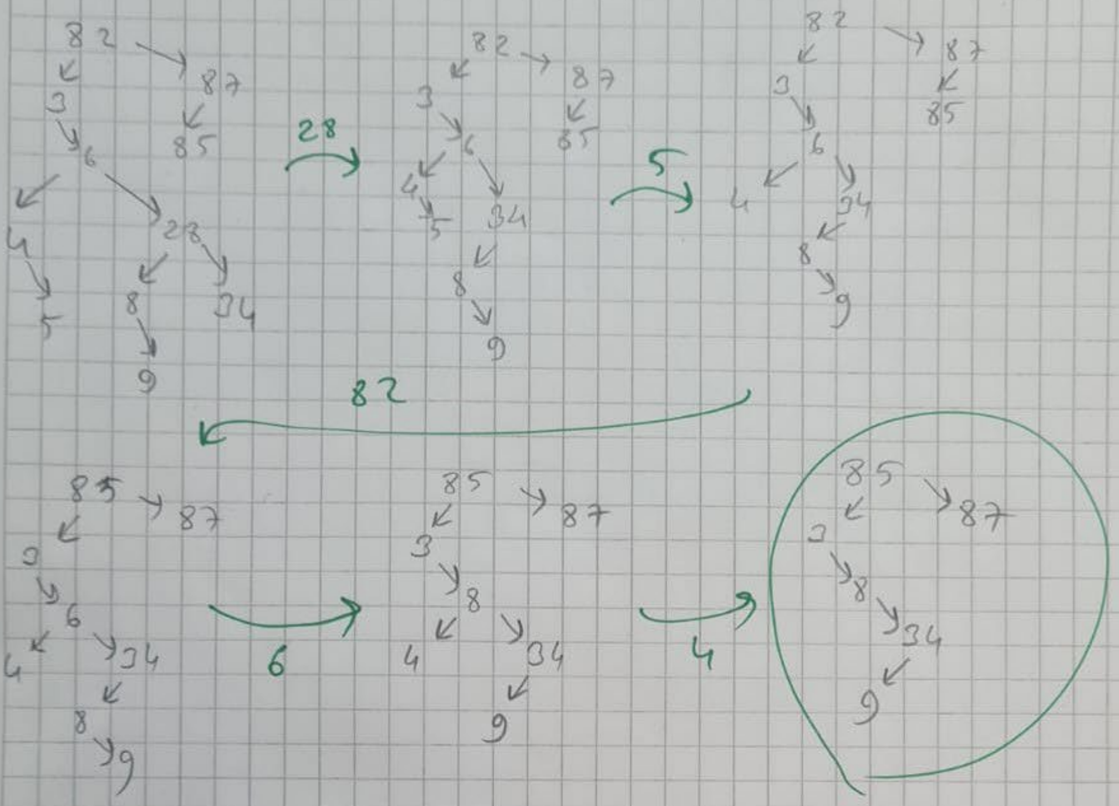
Inorder: 3 → 4 → 5 → 6 → 8 → 9 → 28 → 34 → 82 → 85 → 87

Post order: 5 → 4 → 9 → 8 → 34 → 28 → 6 → 3 → 85 → 87 → 82

Seçkin Alp Kağı -2200[illegible]

Part C

delete 28, 5, 82, 6, 4



Part D

```
int minimumfinder (BSTNode *node) {
    if (node->leftnode == nullptr) { return node->key; }
    return minimumfinder (node->leftnode); }
```

Part E

·n items

maximum: It happens when it is only left nodes or right nodes. Each node have only one child ~~all~~ except for leaf node.

So answer is $\boxed{n-1}$

minimum: maximum number of nodes at level n is $2^{n}-1$

so minimum height

$$n \leq 2^{h}-1$$

$$2^{h-1} < n \leq 2^{h}-1$$

$$2^{h-1} < n+1 \leq 2^{h}$$

$$h-1 < \log_2(n+1) \leq h$$

$$\boxed{h = \lceil \log_2(n+1) \rceil - 1} \rightarrow$$ minimum height of a binary tree with n nodes