**Question 1**

The asymptotic (big O) runtime complexity of the setTraining() method for the BaseMarkov implementation is expected to be O(T). This is because the only algorithm used in the method is .split(), which splits the training text into separate words and adds each one to the array myWords. As T is the number of words in the training text, this addition would happen T times, making the asymptotic runtime complexity O(T).

The asymptotic (big O) runtime complexity of the getRandomText() method for the BaseMarkov implementation is expected to be O(NT). As we are given that nextInt() is a constant time operation, the first four rows of getRandomText() is constant time as they are generations of new array lists, integers, etc. The for loop, then, iterates for N-2 times, as our order is 2 here. This means the Big O of the loop is O(N-2), which is equivalent to O(N) times because we can drop the constant. Inside the loop, the runtime for shiftAdd() is O(T) since the for loop used for this method in WordGram runs for T times, with other codes running in constant time. Hence, with O(T) being iterated for N times, they are multiplied, so the final asymptotic runtime complexity is O(NT).

Experiment:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data file** | **T** | **N** | **Training Time (s)** | **Generating time (s)** |
| biden-2021.txt | 6,129 | 100 | 0.010 | 0.053 |
| biden-2021.txt | 6,129 | 1,000 | 0.014 | 0.248 |
| biden-2021.txt | 6,129 | 10,000 | 0.013 | 1.592 |
| alice.txt | 28,196 | 100 | 0.024 | 0.135 |
| alice.txt | 28,196 | 1,000 | 0.029 | 0.840 |
| alice.txt | 28,196 | 10,000 | 0.026 | 6.712 |
| kjv10.txt | 823,135 | 100 | 0.169 | 2.168 |
| kjv10.txt | 823,135 | 1,000 | 0.158 | 20.165 |
| kjv10.txt | 823,135 | 10,000 | 0.148 | 205.645 |
| shakespeare.txt | 901,325 | 100 | 0.175 | 2.236 |
| shakespeare.txt | 901,325 | 1,000 | 0.165 | 23.086 |
| shakespeare.txt | 901,325 | 10,000 | 0.173 | 214.276 |

Experimenting with different values of T and N, it can be found that the empirical data conforms to my expectations for the runtime complexity of getRandomText(). When N is held constant, there is a linear relationship between T and generating time. For example, when N = 10,000, as T increases by a factor 4.6 (6,129 🡪 28,196), generating time increases by a factor of 4.2 (1.592 🡪 6.712), and as T increases by a factor of 29 (28,196 🡪 823,135), generating time increases by a factor of 30 (6.712 🡪 205.645). When T is held constant, there is also a linear relationship between N and generating time. For example, when T = 901,325, as N increases by a factor of 10 (100 🡪 1,000), generating time increases by a factor of 10 (2.23 🡪 23.09), and as N increases by a factor of 100 (1,000 🡪 10,000), generating time increases by a factor of 100 (23.09 🡪 214.28). Combining these effects, when T increases by a factor of t and N increases by a factor of n, the generating time is shown to increase by a factor of nt.

**Question 2**

**Question 3**