Task: Provide a README.docx file that includes the complexity analysis of your two methods. Include large input text/files in order to justify your analysis. Include plots and tables for test results.

Complexity analysis for Brute Force:

The best case occurs when the first character of the pattern is not present in the text at all, meaning every character is passed over. The number of comparisons in this case is O(n).
The worst case of Brute Force (Naïve Pattern Searching) occurs in the following scenarios:

1. When all characters of the text and pattern are the same.
2. Worst case also occurs when only the last character is different.

The number of comparisons in the worst case is O(m*(n-m+1)). Only in some applications are where repeated characters are likely to occur.

n is the length of the text, while n is the length of the pattern.

Complexity analysis for Rabin-Karp:

The Naïve String Matching Algorithm slides the pattern one by one. After each slide, it one by one checks characters at the current shift and if all characters match then it increases the count of occurrences.

Like the Naïve Algorithm, Rabin-Karp algorithm also slides the pattern one by one. But unlike the Naïve algorithm, Rabin-Karp algorithm matches the hash value of the pattern with the hash value of the current substring of text, and if the hash values match then, and only then does it start matching individual characters thereafter.

The average and best case running time of the Rabin-Karp algorithm is O(m+n), but its worst-case time is O(mn). Worst case of Rabin-Karp algorithm occurs when all characters of a pattern and text are the same as the hash values of all the substrings of a text array, for example txt[] = AAAAAAA, match with hash value of a pattern array, for example pat[] = AAA.

n is the length of the text, while n is the length of the pattern.

Results from the Data:
NOTE: It was a bit difficult to find long common *phrases* in the Bible and the Amendments Document. It was easier to find simple *words* that existed in the two text files.

NOTE 2: I ran 10 tests (back-to-back) for each string or file and corresponding pattern and then took the average of these 10 tests when recording the actual values (you see in the table below). It seemed that each test produced differing values, but not by much when compared to the mean.

In all cases, using different string lengths, different file lengths, and different pattern lengths, brute force (naïve method) was consistently faster than the Rabin-Karp method. Indicated by the table and graph below, using twelve examples, the Rabin-Karp method was consistently slower than the brute force method. The slow behavior of the Rabin-Karp method seemed to take place in the “sliding” part of the algorithm…when it moves up one character to do a comparison…that loop was the slowest part.

Table: Substring Search

	Substring Search			Brute Force Time (nanoseconds)	Rabin-Karp Time (nanoseconds)
	String or File	word	occurences		
0	the cat in the hat	the	2	3307	3218
1	tonight is the night to win	night	2	3818	5224
2	this is the name of the game	the	2	1577	2799
3	where is the blue cat, where is the black cat?	cat	2	7680	9895
4	If you're visiting this page, you're likely here because you're searching for a random sentence.	sentence	1	13441	17445
5	Nory was a Catholic because her mother was a Catholic, and Nory's mother was a Cahtolic because her father was a Catholic, and her father was a Catholic because his mother was a Cahtolic, or had been.	Catholic	6	19575	25487
6	File: amendments.txt	people	7	1916564	1989194
7	File: amendments.txt	law	17	1895515	2063450
8	File: amendments.txt	Amendment	26	462970	2189760
9	File: bible.txt	Jesus	977	13569348	48957246
10	File: bible.txt	good	848	14998009	49344602
11	File: bible.txt	land	1800	15981252	48131249

Graph: Substring Search