**REPORT**

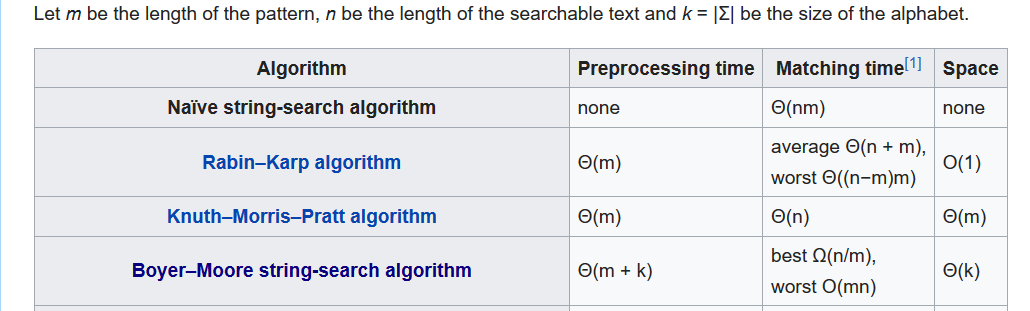
**Introduction:**

Searching for a string in a given text is one of the core elements in programs like plagiarism detection, genetic analysis and malware analysis etc. As these applications gets enormous amount of data i.e in billions of bytes it is important to consider the resources such as time and space. The following algorithms are proposed to find whether a given string is a substring or not.

* Brute force Algorithm
* Knuth - Morris - Pratt algorithm
* Boyer – Moore Algorithm
* Rabin Karp Algorithm

We will analyze the performance of these algorithms by taking a sample input “it is far, far better thing that i do than i have ever done” in the text of Two Cities (tale.txt).

**Theoretical Analysis:**



**Predictions using Theory:**

According to the theoretical analysis we think KMP gives the Best performance among others in the worst case as it got a good Matching time O(N) irrespective of the preprocessing and space issue.

The least performer could be Brute Force as it takes O(NM) and if there is no uniform hash function Rabin Karp would as results in sub optimal performance based on the hash value generation.

**Procedure:**

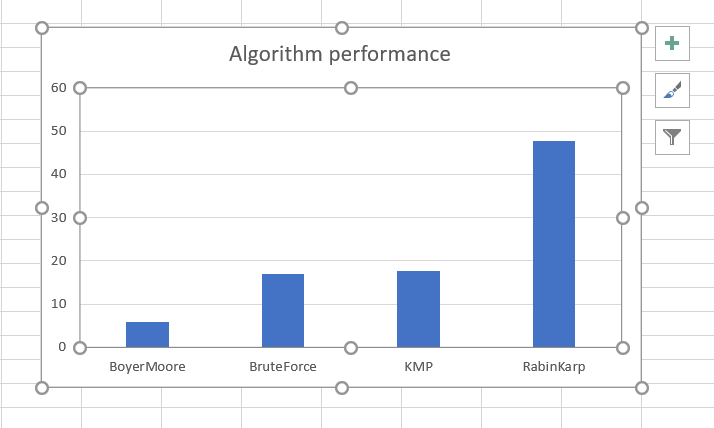
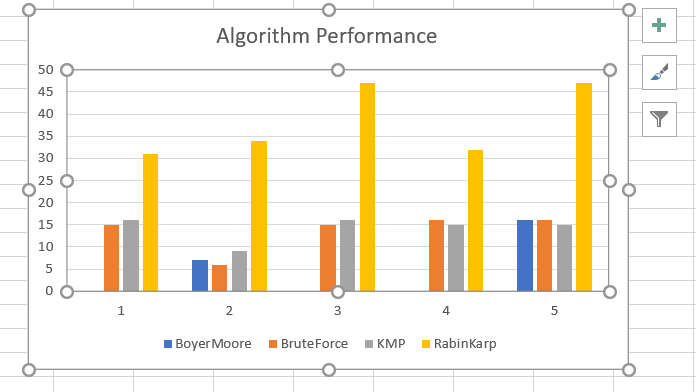
* Making all the four algorithms as a child instances of Parent SearchAlgo which is an abstract class with abstract method search.
* In Solution.java using the concept of Factory Method Design pattern at run time a particular class object is invoked and is run according to text and pattern search, wrapped under system time method.
* By running number of trails and collecting the run time we plot the Bar graph for algorithm vs runtime.
* By using Statistical concepts like mean and mode we present the observations.

**Observations:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | | Trail1(ms) | Trail2(ms) | Trail3(ms) | Trail4(ms) | Trail5(ms) | Average(ms)­­­­­­­­­­­­ |
| BoyerMoore | 0 | 7 | 0 | 0 | 16 | 5.75 |
| BruteForce | 15 | 6 | 15 | 16 | 16 | 17 |
| KMP | 16 | 9 | 16 | 15 | 15 | 17.75 |
| RabinKarp | 31 | 34 | 47 | 32 | 47 | 47.75 |

**Statistical Analysis:**

* Boyer Moore algorithm takes lesser time than any other algorithm for substring search.The reason for that could be it slides the pattern over the text one by one and also uses shift factor and starts matching from the last character of the pattern.
* KMP algorithm does pre-processing over the pattern so that the pattern can be shifted by more than one. Unlike the previous pattern searching algorithms. It gave constant amount of performance over the trails mode of the sample analysis data is of the KMP’s run result.
* In KMP algorithm, more clearly we focus on sub-strings of patterns that are either prefix and suffix. It uses DFA simulation and takes average time to find the substring.
* According to theory Brute force algorithm is the slowest algorithm for substring search but it matched KMP performance very near **(due to the input given)**.
* Rabin Karp Algorithm using non uniform hash function and also randomizing the divisor is causing the run time variance too drastic.



**Conclusion:**

* According to the given testcase the algorithm outperformed is Boyer Moore.
* Predictions were not met but KMP shown its consistence performance over its trails.
* For any other test cases if space is not an issue KMP can produce optimum level performance in worst case.
* If an uniform Hash Function is derived Rabin Karp can excel in all scenarios.
* Finally, according resource constraints and probable test case scenarios we can conclude that Boyer Moore algorithm can be used in most of the scenarios as is fastest.
* Brute force is the slowest and in some cases Rabin Karp can also outperform.

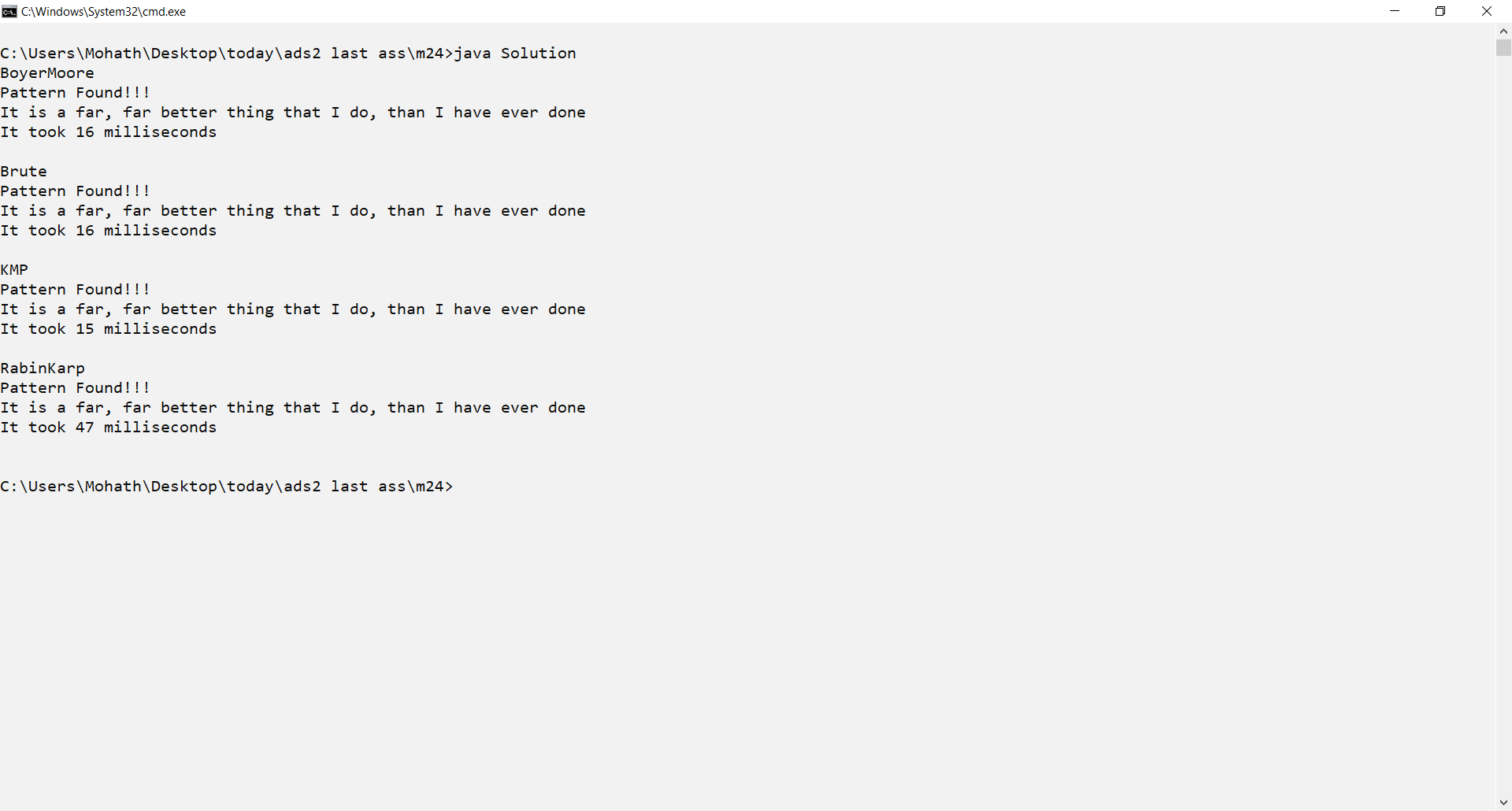
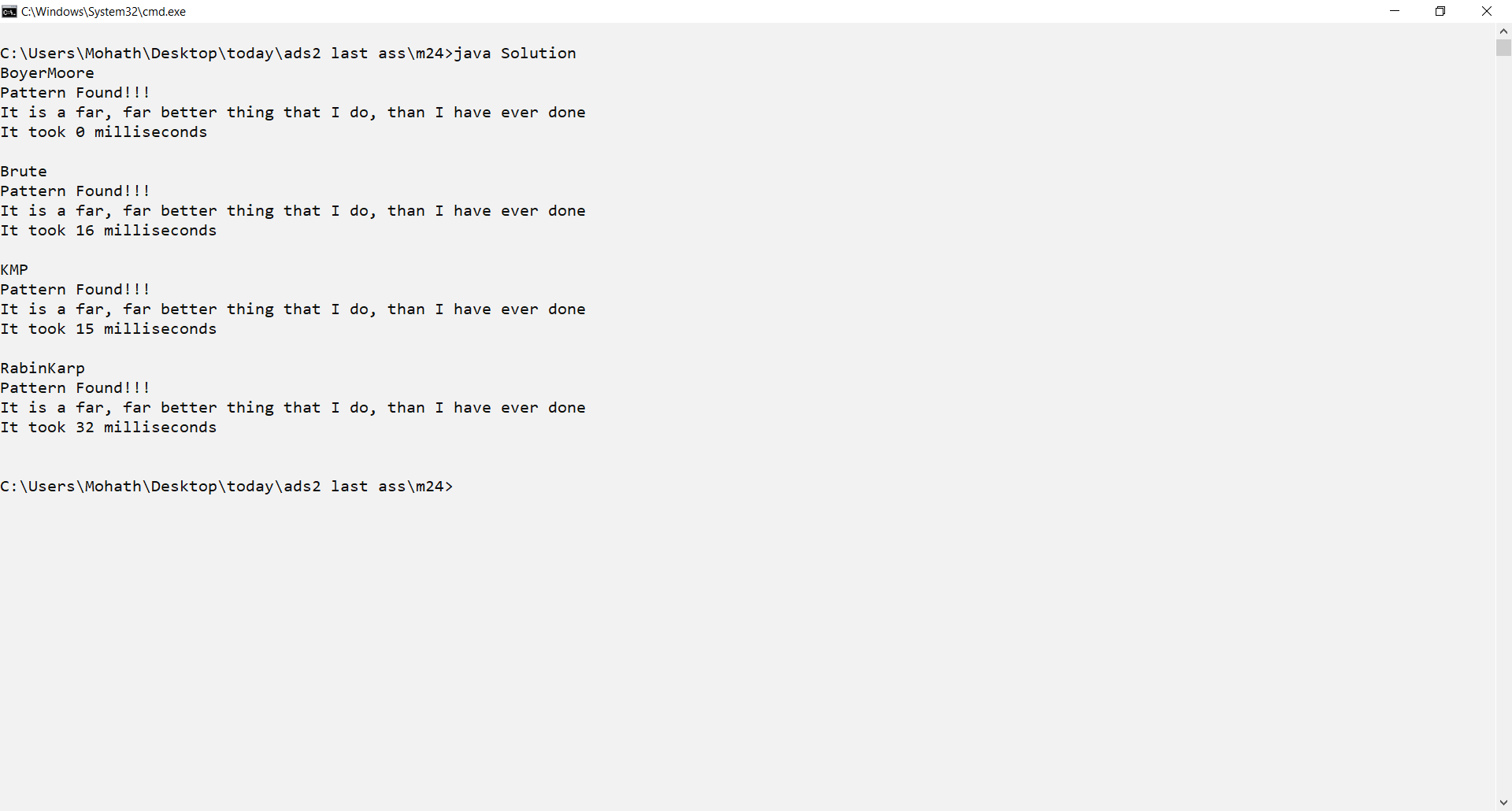
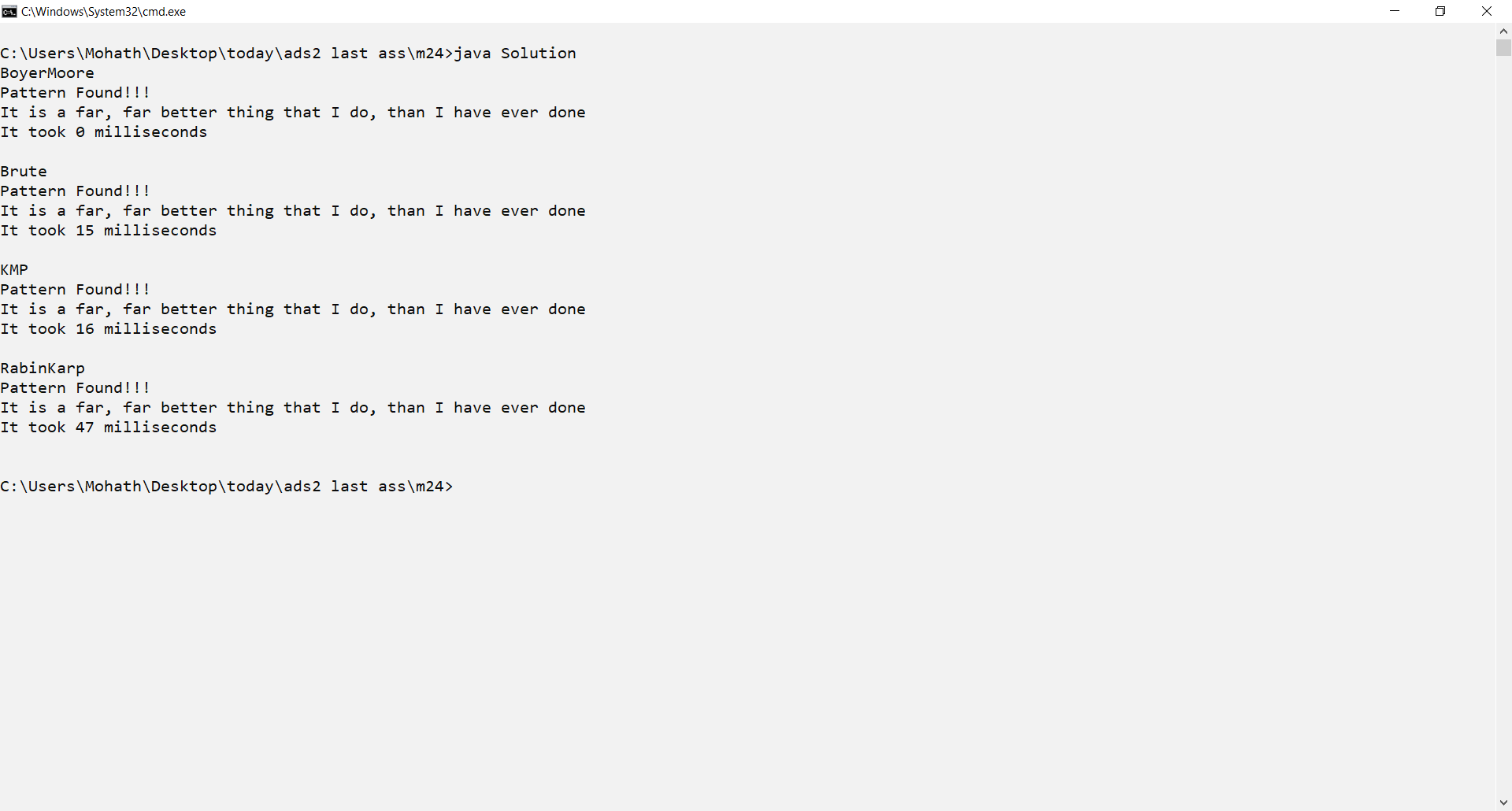
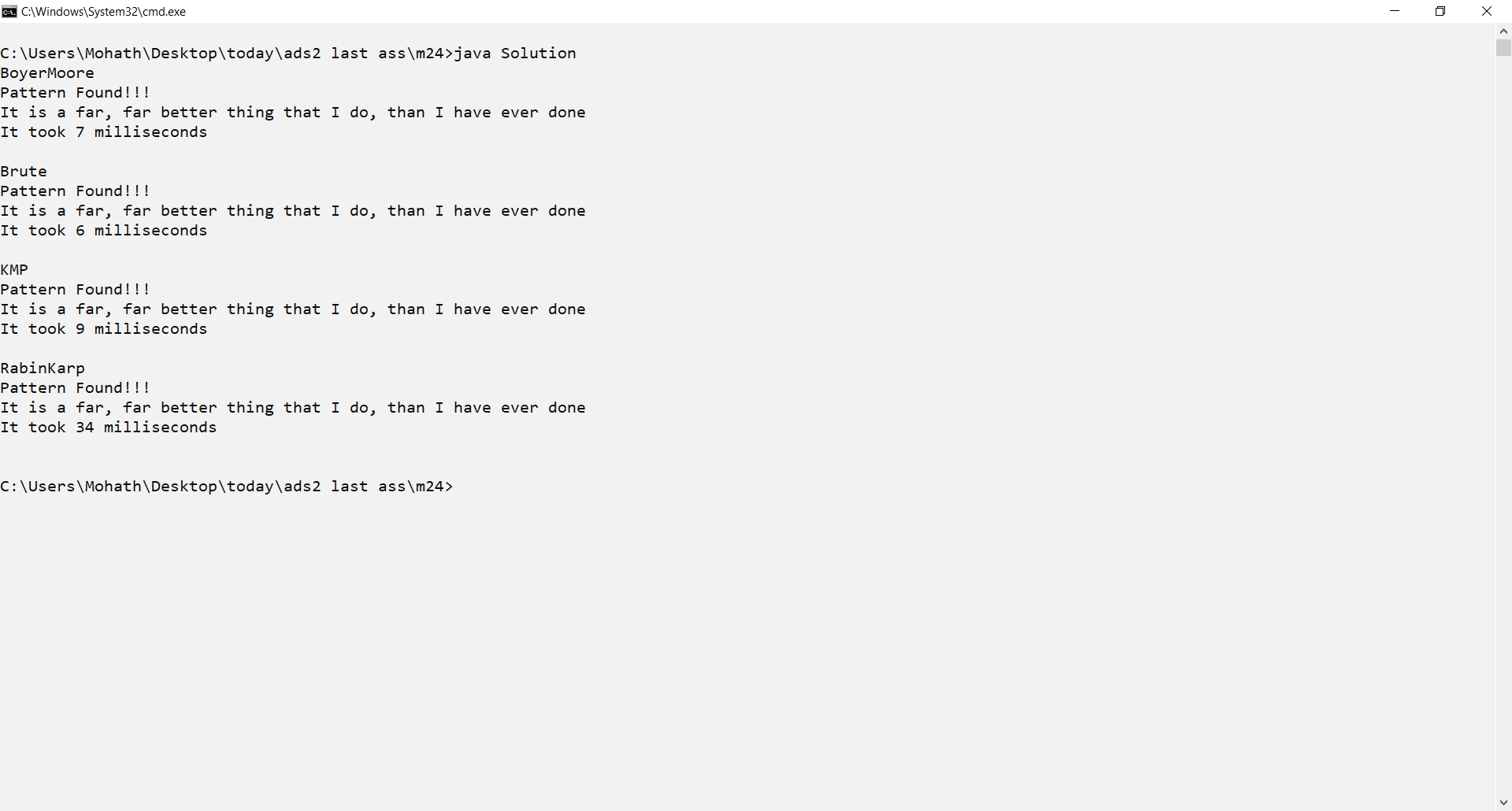
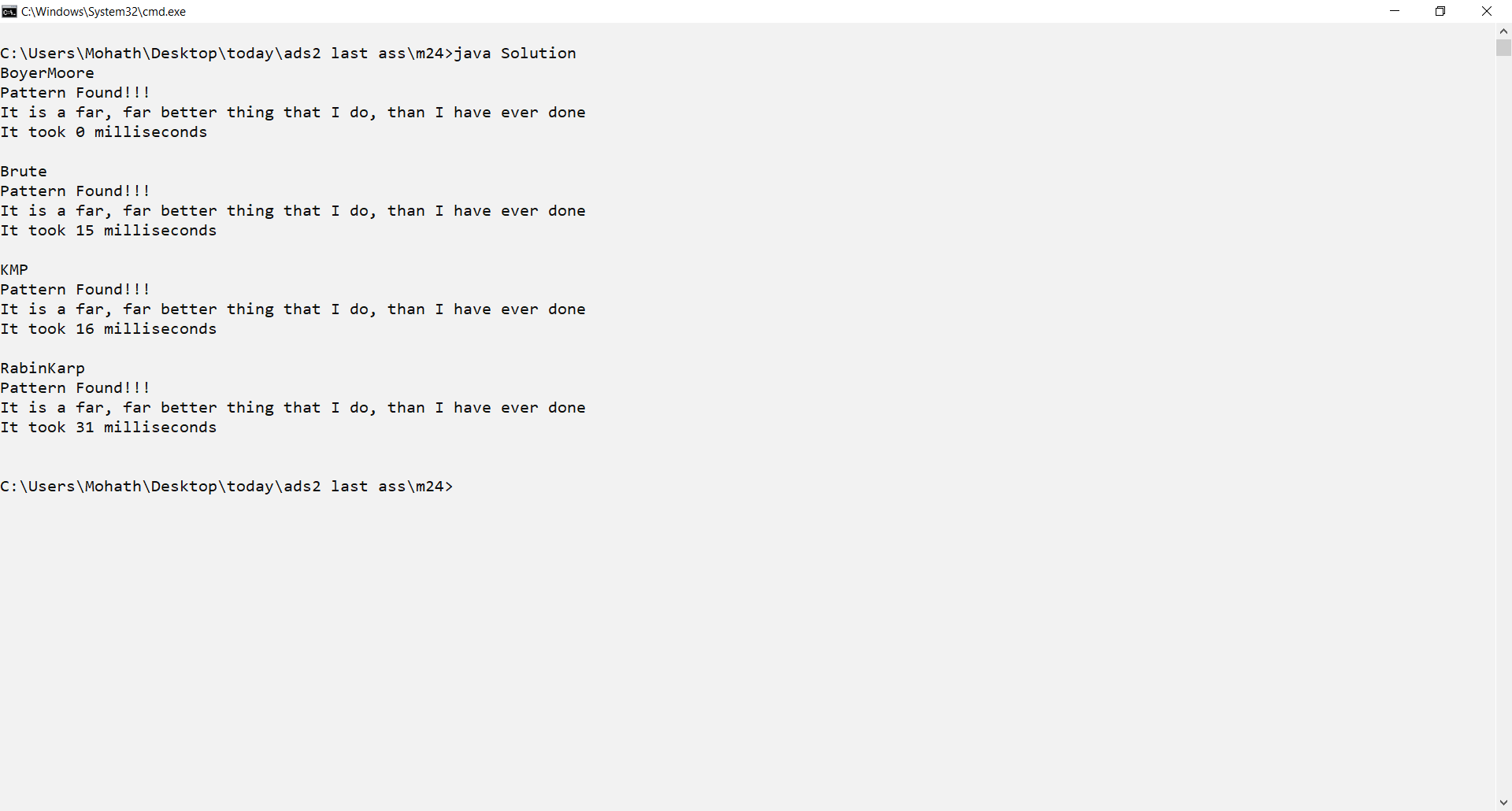
**References:**

<https://algs4.cs.princeton.edu/code/>

<https://algs4.cs.princeton.edu>

<https://en.wikipedia.org/wiki/String-searching_algorithm>

**Screen Shots:**

****