MARMARA UNIVERSITY
FACULTY OF ENGINEERING

ANALYSIS OF ALGORITHMS (CSE2046)
HOMEWORK 1 REPORT

Submitted to: *Dr. Ömer Korçak*

Due Date: *17.05.2023 - 23.59*

	Student Id	Name Surname
1	*150120031*	*Şükrü Can Mayda*
2	*150120063*	*Hamza Ali İslamoğlu*
3	*150120528*	*Samet Can*

Brute Force

Type 1				Type 2			
Has Match		Has No Match		Has Match		Has No Match	
Input		Input		Input		Input	
the	72(ms)	abc	64(ms)	010	138(ms)	There is no pattern which length is 3 and has no matching.	
	1482106		1365216		2210532		
algorithm	50(ms)	abcdefgh k	40(ms)	1001001 00	118(ms)	1100110 10	66(ms)
	1373990		1340723		2636406		2673846
they immerse themselv es	66(ms)	abcdefgh jklmnopr stuwxyz	27(ms)	0100010 0100100 0010100 10	72(ms)	1100110 1101010 0101010 01	45(ms)
	1358286		1336090		2957752		2372752
ele	33(ms)	prstuwxy z	74(ms)			11111111 0	94(ms)
	1464548		1336104				2290086

Horspool

Type 1				Type 2			
Has Match		Has No Match		Has Match		Has No Match	
Input		Input		Input		Input	
the	34(ms) 538864	abc	31(ms) 469082	010	123(ms) 1434530	There is no pattern which length is 3 and has no matching.	
algorithm	14(ms) 193157	abcdefghk	8(ms) 174726	10010010 0	70(ms) 1792768	11001101 0	59(ms) 1856954
they immerse themselves	29(ms) 381267	abcdefghjklmnoprstuwxyz	3(ms) 84534	01000100 01001000 01001010	48(ms) 2074640	11001101 01010100 1	33(ms) 1379589
ele	23(ms) 514837	aaa	32(ms) 489452			11111111 0	16(ms) 394958
		prstuwxyz	26(ms) 160389				

Boyer Moore

Type 1				Type 2			
Has Match		Has No Match		Has Match		Has No Match	
Input		Input		Input		Input	
the	39(ms) 538864	abc	36(ms) 469082	010	102(ms) 1434530	There is no pattern which length is 3 and has no matching.	
algorithm	10(ms) 193157	abcdefghk	10(ms) 174726	100100100	67(ms) 1323237	110011010	16(ms) 656260
they immerse themselves	33(ms) 381267	abcdefghjklmnoprstuwxyz	3(ms) 84534	010001001001000010010	19(ms) 849240	110011011010100101010101	23(ms) 798569
ele	25(ms) 514837	aaa	31(ms) 475530			111111110	16(ms) 394958
		prstuwxyz	33(ms) 160389				

1-) Given Example Results

The input and pattern given in the assignment file have been converted to strings. Queries were made for 3 algorithms. 70 comparisons were made for Brute Force, 20 for Horspool and 18 for BoyerMoore. This query is made automatically at the beginning of the code and the result is printed. These queries were also solved manually and the results were correct. The accuracy of our algorithm codes were tested with other inputs, and at the same time, they were manually solved and checked. Experiments began after this validation.

2-) For Type 1 Experiments and Results

a. Same Length, Both Match, Different patterns

- First pattern is the,

```
Please give pattern=
the
Comparison time is 72(ms) comparison number is 1482106 matching number is 19404 algorithm is Brute Force
Comparison time is 34(ms) comparison number is 538864 matching number is 19404 algorithm is Horspool
Comparison time is 39(ms) comparison number is 538864 matching number is 19404 algorithm is BoyerMoore
```

- Second pattern is ele,

```
Please give pattern=
ele
Comparison time is 33(ms) comparison number is 1464548 matching number is 1848 algorithm is Brute Force
Comparison time is 23(ms) comparison number is 514837 matching number is 1848 algorithm is Horspool
Comparison time is 25(ms) comparison number is 514837 matching number is 1848 algorithm is BoyerMoore
```

Since our pattern is recursive, we expected it to make more comparisons, but it made fewer text-dependent comparisons.From this we can draw the following conclusion: ele was recursive and worked faster, expecting it to work slower. That means that the 'h' before the e character is found more than the

'l'.It worked the same for boyer moore and horspool ‘ele’ because the shift amounts are almost the same

b. Same Length, Both No Matches, Different patterns

- First pattern is abcdefghk,

```
Please give pattern=
abcdefghk
Comparison time is 40(ms) comparison number is 1340723 matching number is 0 algorithm is Brute Force
Comparison time is 8(ms) comparison number is 174726 matching number is 0 algorithm is Horspool
Comparison time is 10(ms) comparison number is 174726 matching number is 0 algorithm is BoyerMoore
```

- Second pattern is prstuwxyz,

```
Please give pattern=
prstuwxyz
Comparison time is 34(ms) comparison number is 1336104 matching number is 0 algorithm is Brute Force
Comparison time is 16(ms) comparison number is 160389 matching number is 0 algorithm is Horspool
Comparison time is 16(ms) comparison number is 160389 matching number is 0 algorithm is BoyerMoore
```

In the first pattern comparison times of three algorithms are short because there is no match. Brute Force is the slowest algorithm because it checks letter by letter. Horspool algorithm and Boyer Moore algorithm have the same comparison number. Because Booyer Moore and Horspool algorithm behaves the same when there is no any match. However their comparison time is a little bit different. In the Brute Force algorithm of the second pattern, the comparison number is almost the same, and because of that the first and second pattern’s comparison times are very similar. In Horspool and Boyer Moore algorithms, comparison number and comparison time are exactly the same; In the second pattern there is less comparison number but there is a bit much longer comparison times. It is probably because of the computer’s execution.

c. Same Length, Has Match vs Has No match

- First pattern is , 110011010

```
Please give pattern=
110011010
Comparison time is 66(ms) comparison number is 2673846 matching number is 0 algorithm is Brute Force
Comparison time is 59(ms) comparison number is 1856954 matching number is 0 algorithm is Horspool
Comparison time is 16(ms) comparison number is 656260 matching number is 0 algorithm is BoyerMoore
```

- Second is 100100100

```
Please give pattern=
100100100
Comparison time is 118(ms) comparison number is 2636406 matching number is 10920 algorithm is Brute Force
Comparison time is 70(ms) comparison number is 1792768 matching number is 10920 algorithm is Horspool
Comparison time is 67(ms) comparison number is 1323237 matching number is 10920 algorithm is BoyerMoore
```

For all three algorithms, we can observe that the time is longer in the pattern in which the search is successful. While the Boyer-moore algorithm and the Horspool algorithm for the matching pattern work almost the same time, there is a difference for the algorithms in the case that they do not match.This is because Boyer moore makes less comparison while jumping more due to its structure, which is far from recursive, while horspool makes more comparison while jumping less according to the table.Conversely, because we see a recursive structure in the matching pattern, Boyer-Moore will start working more slowly and approach the Horspool working principle now that it repeats itself more often.This is the reason for the decrement of the number of comparisons between Horspool and Boyer-Moore.

d. Different Lengths, Both Match

- First pattern is

```
Please give pattern=
the
Comparison time is 72(ms) comparison number is 1482106 matching number is 19404 algorithm is Brute Force
Comparison time is 34(ms) comparison number is 538864 matching number is 19404 algorithm is Horspool
Comparison time is 39(ms) comparison number is 538864 matching number is 19404 algorithm is BoyerMoore
```

- Second pattern is

```
Please give pattern=
algorithm
Comparison time is 50(ms) comparison number is 1373990 matching number is 1386 algorithm is Brute Force
Comparison time is 14(ms) comparison number is 193157 matching number is 1386 algorithm is Horspool
Comparison time is 10(ms) comparison number is 193157 matching number is 1386 algorithm is BoyerMoore
```

- Third pattern is

```
Please give pattern=
they immerse themselves
Comparison time is 66(ms) comparison number is 1358286 matching number is 1848 algorithm is Brute Force
Comparison time is 29(ms) comparison number is 381267 matching number is 1848 algorithm is Horspool
Comparison time is 33(ms) comparison number is 381267 matching number is 1848 algorithm is BoyerMoore
```

For the Brute Force algorithm when the length of the pattern is increased, there are no such bigger differences between their comparison numbers; so that their comparison times also are similar. For Horspool algorithm and Booyer Moore algorithm their comparison numbers and comparison times are exactly the same for each pattern. Comparison times of them depend on comparison numbers, when the pattern length is increased, firstly comparison numbers and times decrease very fastly (about 1/3) and then pattern is increased much more this time, later comparison number is increased 2 times. Their comparison times have changed according to these comparison numbers.

e. Different Lengths, Both No Match

- Pattern is abc

```
Please give pattern=
abc
Comparison time is 64(ms) comparison number is 1365216 matching number is 0 algorithm is Brute Force
Comparison time is 31(ms) comparison number is 469082 matching number is 0 algorithm is Horspool
Comparison time is 36(ms) comparison number is 469082 matching number is 0 algorithm is BoyerMoore
```

- Pattern is abcdefghk

```
Please give pattern=
abcdefghk
Comparison time is 40(ms) comparison number is 1340723 matching number is 0 algorithm is Brute Force
Comparison time is 8(ms) comparison number is 174726 matching number is 0 algorithm is Horspool
Comparison time is 10(ms) comparison number is 174726 matching number is 0 algorithm is BoyerMoore
```

- Pattern is abcdefghjklmnoprstuwxyz

```
Please give pattern=
abcdefghjklmnoprstuwxyz
Comparison time is 27(ms) comparison number is 1336090 matching number is 0 algorithm is Brute Force
Comparison time is 3(ms) comparison number is 84534 matching number is 0 algorithm is Horspool
Comparison time is 3(ms) comparison number is 84534 matching number is 0 algorithm is BoyerMoore
```

We can clearly observe that the algorithm speed increases as the pattern length increases.The brute force algorithm of the pattern works faster than horspool and boyer moore algorithms according to these patterns.For these pattern types, we can observe that boyer moore and horspool algorithms work at almost the same speed and with the same comparison.As the pattern size increases, the speed of the algorithms increases, and the reason is that as the size increases, the matching possibilities of the entire pattern are more difficult, so the word skips without checking too many characters.Because it is more difficult for long characters to match one by one, respectively, the moment there is no match, the jump is made without looking at any of the other characters. Therefore, as can be understood from the images above, as

the pattern increases in size, the algorithm comparisons decrease and the speed increases.

3-)For Type 2 Experiements and Results

f. Same Length, Both Match, Different patterns

- First pattern is 100100100,

```
Please give pattern=
100100100
Comparison time is 126(ms) comparison number is 2636406 matching number is 10920 algorithm is Brute Force
Comparison time is 72(ms) comparison number is 1792768 matching number is 10920 algorithm is Horspool
Comparison time is 79(ms) comparison number is 1323237 matching number is 10920 algorithm is BoyerMoore
```

- Second pattern is 001001001,

```
Please give pattern=
001001001
Comparison time is 109(ms) comparison number is 2343126 matching number is 15600 algorithm is Brute Force
Comparison time is 87(ms) comparison number is 1415439 matching number is 15600 algorithm is Horspool
Comparison time is 80(ms) comparison number is 1409275 matching number is 15600 algorithm is BoyerMoore
```

Both inputs have similar comparison times and comparison numbers. Their matching numbers are a little different. For the first pattern, the Brute Force algorithm is slower than the second. The Horspool algorithm is slower in the first pattern too. However, Horspool's comparison time is very similar to Boyer Moore algorithm. In the first pattern Horspool is faster, in the second pattern Boyer Moore is faster. Also in the Boyer Moore algorithm there are no such changes (just 1 ms) in comparison times. Comparison numbers are similar too.

g. Same Length, Both No Matches

- First pattern is 111111110,

```
Please give pattern=
111111110
Comparison time is 94(ms) comparison number is 2290086 matching number is 0 algorithm is Brute Force
Comparison time is 16(ms) comparison number is 394958 matching number is 0 algorithm is Horspool
Comparison time is 16(ms) comparison number is 394958 matching number is 0 algorithm is BoyerMoore
```

- Second pattern is 110011010,

```
Please give pattern=
110011010
Comparison time is 66(ms) comparison number is 2673846 matching number is 0 algorithm is Brute Force
Comparison time is 59(ms) comparison number is 1856954 matching number is 0 algorithm is Horspool
Comparison time is 16(ms) comparison number is 656260 matching number is 0 algorithm is BoyerMoore
```

First pattern is not recursive, so that the Boyer Moore algorithm behaves as the Horspool algorithm and their comparison number is equal. According to this, their comparison time is equal. Second pattern is recursive, so comparison numbers of all three algorithms increased. Comparison time of Brute Force algorithm is increased 1.5 times when Horspool algorithm is increased about 4 times and so on, in Boyer Moore algorithm comparison time has not changed because comparison number is increased a bit. However it may be changed a bit too but not changed. The reason why the Horspool algorithm's comparison time has increased so much, the comparison number of it has increased as well.

h. Same Length, Has Match vs Has No match

- First pattern is 100100100,

```
Please give pattern=
100100100
Comparison time is 126(ms) comparison number is 2636406 matching number is 10920 algorithm is Brute Force
Comparison time is 72(ms) comparison number is 1792768 matching number is 10920 algorithm is Horspool
Comparison time is 79(ms) comparison number is 1323237 matching number is 10920 algorithm is BoyerMoore
```

- Second pattern is 110011010,

```
Please give pattern=
110011010
Comparison time is 66(ms) comparison number is 2673846 matching number is 0 algorithm is Brute Force
Comparison time is 59(ms) comparison number is 1856954 matching number is 0 algorithm is Horspool
Comparison time is 16(ms) comparison number is 656260 matching number is 0 algorithm is BoyerMoore
```

For the Brute Force algorithm when there are matches the comparison times increases 2 times. Also in the Boyeer Moore algorithm, its comparison time increases so fast when there are matches (about 5 times). When there are so many matches, Boyer Moore is slower than the Horspool algorithm, but when there is no match Boyer Moore has even less comparison numbers and comparison times. So it is much faster than the others.

i. Different Length, Has Match

- First pattern is 010,

```
Please give pattern=
010
Comparison time is 138(ms) comparison number is 2210532 matching number is 282360 algorithm is Brute Force
Comparison time is 123(ms) comparison number is 1434530 matching number is 282360 algorithm is Horspool
Comparison time is 102(ms) comparison number is 1434530 matching number is 282360 algorithm is BoyerMoore
```

- Second pattern is 100100100,

```
Please give pattern=
100100100
Comparison time is 126(ms) comparison number is 2636406 matching number is 10920 algorithm is Brute Force
Comparison time is 72(ms) comparison number is 1792768 matching number is 10920 algorithm is Horspool
Comparison time is 79(ms) comparison number is 1323237 matching number is 10920 algorithm is BoyerMoore
```

- Third pattern is 010001001001000010100 10,

```
Please give pattern=
01000100100100001010010
Comparison time is 72(ms) comparison number is 2957752 matching number is 9360 algorithm is Brute Force
Comparison time is 48(ms) comparison number is 2074648 matching number is 9360 algorithm is Horspool
Comparison time is 19(ms) comparison number is 849240 matching number is 9360 algorithm is BoyerMoore
```

For short and medium patterns, brute force has not changed much; when pattern is larger, comparison number is increased a bit but comparison time is decreased almost half of first two ones. Horspool and Boyer Moore algorithms are also slower for short patterns, because their matching numbers are huge compared to others. When the pattern becomes larger, the Horspool algorithm becomes a little faster, and the Boyer Moore algorithm becomes much faster. Also their matching numbers decrease as well. For three of the algorithms , comparison numbers are very similar to each other for a short pattern. When a medium pattern is entered, Brute Force and Horspool algorithms decrease a bit, Boyer Moore increases a bit. For larger patterns, Brute Force and Horspool algorithm's comparison number increases a bit again, However Boyer Moore algorithm's comparison number decreases well.

j. Different Length, Has No Match

- Pattern is 110011010

```
Please give pattern=
110011010
Comparison time is 66(ms) comparison number is 2673846 matching number is 0 algorithm is Brute Force
Comparison time is 59(ms) comparison number is 1856954 matching number is 0 algorithm is Horspool
Comparison time is 16(ms) comparison number is 656260 matching number is 0 algorithm is BoyerMoore
```

- Pattern is 11001101101010010101001

```
Please give pattern=
11001101101010010101001
Comparison time is 45(ms) comparison number is 2372752 matching number is 0 algorithm is Brute Force
Comparison time is 33(ms) comparison number is 1379589 matching number is 0 algorithm is Horspool
Comparison time is 23(ms) comparison number is 798569 matching number is 0 algorithm is BoyerMoore
```

The general rule for algorithms as we have seen in English pattern matching applies here as well, where the algorithm speeds up and reduces the number of comparisons as the pattern size increases. However, there is one algorithm that breaks this general rule, and that is the Boyer-Moore algorithm.Unlike the others, Boyer-Moore algorithm struggles to perform excessive shifts due to patterns consisting only of 0s and 1s.The reason Boyer-Moore performs faster and better with longer patterns is its ability to make larger jumps more efficiently. However, due to the excessive recursion in this case, it will perform more searches with smaller shifts.

4-) Type 1 vs Type 2

k. Short patterns, Both Matches

- Pattern is the

```
Please give pattern=
the
Comparison time is 72(ms) comparison number is 1482106 matching number is 19404 algorithm is Brute Force
Comparison time is 34(ms) comparison number is 538864 matching number is 19404 algorithm is Horspool
Comparison time is 39(ms) comparison number is 538864 matching number is 19404 algorithm is BoyerMoore
```

- Pattern is 010

```
Please give pattern=
010
Comparison time is 138(ms) comparison number is 2210532 matching number is 282360 algorithm is Brute Force
Comparison time is 123(ms) comparison number is 1434530 matching number is 282360 algorithm is Horspool
Comparison time is 102(ms) comparison number is 1434530 matching number is 282360 algorithm is BoyerMoore
```

Since there is similarity between the pattern characters in Type 2, BoyerMoore and Horspool algorithms are not much different from BruteForce. This situation is evident in Type 1. Since the characters of the entered pattern are completely different, longer jumps were experienced and the difference increased with BruteForce.

l. Long patterns, Both Matches

- they immerse themselves

```
Please give pattern=
they immerse themselves
Comparison time is 66(ms) comparison number is 1358286 matching number is 1848 algorithm is Brute Force
Comparison time is 29(ms) comparison number is 381267 matching number is 1848 algorithm is Horspool
Comparison time is 33(ms) comparison number is 381267 matching number is 1848 algorithm is BoyerMoore
```

- 01000100100100001010010

```
Please give pattern=
01000100100100001010010
Comparison time is 72(ms) comparison number is 2957752 matching number is 9360 algorithm is Brute Force
Comparison time is 48(ms) comparison number is 2074648 matching number is 9360 algorithm is Horspool
Comparison time is 19(ms) comparison number is 849240 matching number is 9360 algorithm is BoyerMoore
```

In this comparison, we can clearly see the difference between Horspool and BoyerMoore algorithms. We can say that the Suffix table greatly increases the jump amounts for the 2nd type, and thus BoyerMoor works much faster than Horspool. In this way, we have seen an example of a non-recurring pattern and a repeating pattern. We understood the contribution of the Suffix table very well.

m. Long patterns, Both No Matches

- abcdefghjklmnoprstuwxyz

```
Please give pattern=
abcdefghjklmnoprstuwxyz
Comparison time is 27(ms) comparison number is 1336090 matching number is 0 algorithm is Brute Force
Comparison time is 3(ms) comparison number is 84534 matching number is 0 algorithm is Horspool
Comparison time is 3(ms) comparison number is 84534 matching number is 0 algorithm is BoyerMoore
```

- 01000100100100001010010

```
Please give pattern=
01000100100100001010010
Comparison time is 72(ms) comparison number is 2957752 matching number is 9360 algorithm is Brute Force
Comparison time is 48(ms) comparison number is 2074648 matching number is 9360 algorithm is Horspool
Comparison time is 19(ms) comparison number is 849240 matching number is 9360 algorithm is BoyerMoore
```

For a non-recursive pattern where there is no match, we can say that there is almost no difference for boyermoore and horspool. These two experiments have shown very well how the patterns affect the suffix tables and thus the amount of jumps.

5-)Conclusion

a. Coding

While writing the algorithms, we wrote the BruteForce, Horspool and BoyerMoore algorithms, respectively. The only thing that could cause problems in the BruteForce algorithm was the loop boundaries, but we didn't have much difficulty here. First we wrote this algorithm and made checks, when we were sure it worked, we switched to the Horspool algorithm. Here we clearly understood that the most important difference between these three algorithms is the amount of jumps. One table was enough for the Horspool algorithm, and obviously it wasn't too difficult to create this table. We drew all the characters before the file we received as input, and then we wrote the jump amount for each character on the map. I should also point out that printing this table properly took more effort than creating the table. The Horspool algorithm didn't bother us in a big way, either.

When we came to the BoyerMoore algorithm, it was not difficult to create the suffix table, but then we had a lot of difficulty. We spent more than half of the time we spent on this project trying to correctly calculate the jump amount in the BoyerMoore algorithm. First of all, we realized that we need to give a

condition for the case where the match is achieved. Then we realized that we need to set a condition for the case where no match is found.

After writing the algorithms, we tested a lot, and it was time to read and create HTML files. It wasn't hard to read, we used StringBuilder and got the whole file into a string. While printing, we realized that we needed an index list. In fact, we later took the number of matches from the size of this list. We located the <mark> tags according to the index list and created our new file. This also worked flawlessly, except for one case. Where there was nesting in the results, these labels were nested. I will be specifying an example of this with the image.

Finally, we wrote the code for the menu and put the example given in the assignment at the beginning. We are getting some errors in the return to the top as we do not understand in the menu, but other than that, our algorithms and all our printing processes are working fine.

b. Experiements Results

After all the coding work was done, we started the experiments. Here we encountered some expected results and some unexpected results. First of all, we thought that the BruteForce algorithm was more primitive than the other two algorithms, and it was definitely not better than the other two in terms of comparison.

The second thing I should mention is actually the most important one; Horspool and BoyerMoore difference. First of all I have to say that BoyerMoore is better than Horspool but I can't say that for every situation. Especially in cases where the input consists of completely separate characters, BoyerMoore does not provide any advantage in the amount of skips, and it is more inefficient in terms of time since it prolongs the work in calculating the amount of skips.

On the other hand, in the expected recursive patterns, the result is truly magnificent. Detecting these results is much easier on the type2 input.

We have explained our interpretations in different situations in detail in the previous sections.

c. Inferences

What remained to us from this assignment was to increase our analytical ability. Of course, using these 3 algorithms in detail allowed us to learn these

3 algorithms better. On the other hand, these 3 algorithms were already existing algorithms. For this reason, the biggest thing we contributed to ourselves was to improve our ability to analyze algorithms. We also discovered that the net time the algorithms will use depends on the current state of the computer.

- 00111100000100000000010111110101001010rk>10rk>10rk>10rk>10rk>10rk>100000010010010100100101001100101001010001001001000010100101rk>101
- 00000011000000101111101010010101rk>10rk>10rk>10rk>10rk>10rk>1000000100101010010100110001010010001001001000010100101rk>101
- 00000000000010000000001011111010000000010101101001110101rk>10rk>10rk>10010110010100110001010010001001001000010100101rk>101
- 00111100000100011010101rk>10rk>10rk>10010010000101111101010010101rk>10rk>10rk>10rk>10rk>10rk>10000001001001010010100110001010010001001
- 0001010rk>10000101011110100001011111010100101rk>10rk>10rk>10rk>10rk>10rk>10000001001001010010100110001010010001001001000010100101
- 000001110101rk>10rk>1000100000000010111110100000000101011010011101rk>10rk>10rk>10010110010100110001010010001001001000010100101rk
- 00111100000100000000010111110101001010rk>10rk>10rk>10rk>10rk>10rk>1000000100100101001001100010100100010010010000101001010rk>101
- 00000011000000101111101010010101rk>10rk>10rk>10rk>10rk>10rk>10000001001001010010100110001010010001001001000010100101rk>101

Output problem in recursive pattern

6-)Tasks Sharing

Count methods, create table, menu and ReadHTML class written by Samet,
Count methods, printing table written by Can
Create Suffix written by Hamza

Experiements mostly maded by Hamza and results shared us
Our transactions are correct checked by Hamza

Report written together