Design And Analysis Of Algorithms
Bonus Homework Report

Enes ELDEŞ - 22050111017

1 Boyer–Moore Implementation

The Boyer–Moore algorithm was implemented using the *bad character heuristic*. The general logic is familiar with the GeeksForGeeks implementation.

1.1 Unicode-Aware Modification

Standard Boyer–Moore implementations typically assume ASCII characters with a fixed-size table (usually 256). However, during testing, cases involving Unicode characters were encountered.

To solve this, the implementation dynamically determines the maximum character value appearing in both the text and the pattern, and constructs the bad character table accordingly.

This ensures algorithm is reliable, but causes overhead as seen in the results table. **Some test cases that are best case for Boyer Moore are not faster with this implementation.**

1.2 Design Summary

- Uses bad character heuristic only
- Dynamically sized bad character table
- Correct handling of empty patterns and edge cases

2 Pre-Analysis Strategy

The selection strategy is based on the following features:

1. **Pattern Length:** Very short patterns use the Naive algorithm.
2. **Repetition Ratio:** Detected via the LPS array; high repetition favors KMP.
3. **Character Diversity:** High diversity favors Boyer–Moore due to better skipping behavior.
4. **Text Entropy:** Shannon entropy is used to estimate randomness. High entropy favors Rabin–Karp, while low entropy favors KMP.

3 Experimental Results

See the output of the ./test.sh command in the Figures 1 and 2.

Test Case	RabinKarp (µs)	GoCrazy (µs)	BoyerMoore (µs)	KMP (µs)	Naive (µs)	Winner
Simple Match	3.473	N/A	4.438	3.134	3.786	🏆 KMP
No Match	1.157	N/A	1.742	1.431	0.741	🏆 Naive
Single Character	6.863	N/A	6.777	6.186	6.758	🏆 KMP
Pattern at End	5.808	N/A	3.077	1.543	1.002	🏆 Naive
Pattern at Beginning	2.421	N/A	2.140	1.423	1.074	🏆 Naive
Overlapping Patterns	2.802	N/A	3.055	1.825	2.102	🏆 KMP
Long Text Multiple Matches	5.738	N/A	6.229	7.279	4.736	🏆 Naive
Pattern Longer Than Text	0.160	N/A	0.154	0.677	0.190	🏆 BoyerMoore
Entire Text Match	1.270	N/A	2.054	1.374	1.063	🏆 Naive
Repeating Pattern	3.483	N/A	3.794	2.307	2.045	🏆 Naive
Case Sensitive	2.561	N/A	2.878	2.400	1.768	🏆 Naive
Numbers and Special Characters	3.810	N/A	3.565	3.540	2.287	🏆 Naive
Unicode Characters	7.004	N/A	130.285	2.668	1.651	🏆 Naive
Very Long Text	39.241	N/A	30.738	41.983	28.691	🏆 Naive
Pattern with Spaces	4.026	N/A	3.796	3.570	2.625	🏆 Naive
All Same Character	14.056	N/A	14.122	6.874	13.418	🏆 KMP
Alternating Pattern	21.103	N/A	20.233	12.492	17.430	🏆 KMP
Long Pattern	11.901	N/A	10.021	12.272	7.852	🏆 Naive
Pattern at Boundaries	2.535	N/A	3.058	3.374	1.934	🏆 Naive
Near Matches	1.778	N/A	3.664	1.978	1.652	🏆 Naive
Empty Pattern	0.868	N/A	0.822	1.074	1.131	🏆 BoyerMoore
Empty Text	0.160	N/A	0.163	0.403	0.179	🏆 RabinKarp
Both Empty	0.527	N/A	0.316	0.542	0.326	🏆 BoyerMoore
Single Character Pattern	7.551	N/A	7.363	6.598	6.191	🏆 Naive
Complex Overlap	3.156	N/A	3.855	2.357	2.713	🏆 KMP
DNA Sequence	10.977	N/A	9.100	8.131	9.933	🏆 KMP
Palindrome Pattern	6.389	N/A	5.141	6.409	4.171	🏆 Naive
Worst Case for Naive	6.859	N/A	12.809	8.849	6.190	🏆 Naive
Best Case for Boyer-Moore	3.119	N/A	5.957	6.594	1.477	🏆 Naive
KMP Advantage Case	1.535	N/A	2.500	8.588	4.395	🏆 RabinKarp

Figure 1: Execution time comparison of string matching algorithms across test cases

Test Case	Choice	PreA+Choice (µs)	vs Naive	vs RabinKarp	vs BoyerMoore	vs KMP
Simple Match	Naive	1.78	N/A	+0.72 µs	-1.76 µs	+0.53 µs
No Match	Naive	0.41	N/A	+0.04 µs	-0.93 µs	-0.28 µs
Single Character	Naive	1.59	N/A	-0.02 µs	-1.59 µs	+0.15 µs
Pattern at End	BoyerMoore	3.62	+2.98 µs	+2.86 µs	N/A	+2.80 µs
Pattern at Beginning	BoyerMoore	2.79	+2.45 µs	+2.32 µs	N/A	+1.73 µs
Overlapping Patterns	Naive	1.03	N/A	+0.02 µs	-0.67 µs	-0.11 µs
Long Text Multiple Matches	Naive	1.47	N/A	-0.35 µs	-1.60 µs	-1.15 µs
Pattern Longer Than Text	BoyerMoore	1.40	+1.28 µs	+1.29 µs	N/A	+0.89 µs
Entire Text Match	BoyerMoore	2.89	+2.59 µs	+2.45 µs	N/A	+1.89 µs
Repeating Pattern	Naive	1.08	N/A	-0.12 µs	-0.75 µs	-0.07 µs
Case Sensitive	Naive	0.75	N/A	-0.25 µs	-0.97 µs	-0.45 µs
Numbers and Special Characters	BoyerMoore	3.53	+2.79 µs	+2.17 µs	N/A	+2.26 µs
Unicode Characters	Naive	1.10	N/A	+0.33 µs	-49.09 µs	-0.04 µs
Very Long Text	Naive	9.42	N/A	-1.27 µs	-2.26 µs	-4.19 µs
Pattern with Spaces	BoyerMoore	3.65	+2.54 µs	+2.64 µs	N/A	+2.27 µs
All Same Character	KMP	4.77	-0.22 µs	-0.42 µs	-1.96 µs	N/A
Alternating Pattern	Naive	9.64	N/A	-1.78 µs	-0.58 µs	+1.97 µs
Long Pattern	BoyerMoore	7.99	+4.34 µs	+4.49 µs	N/A	+3.25 µs
Pattern at Boundaries	Naive	1.13	N/A	-0.46 µs	-1.38 µs	-0.65 µs
Near Matches	Naive	1.08	N/A	+0.25 µs	-0.62 µs	-1.75 µs
Empty Pattern	Naive	1.21	N/A	+0.50 µs	+0.53 µs	+0.51 µs
Empty Text	Naive	0.44	N/A	+0.26 µs	+0.27 µs	-0.05 µs
Both Empty	Naive	0.51	N/A	-0.22 µs	+0.19 µs	-0.18 µs
Single Character Pattern	Naive	4.44	N/A	-0.10 µs	-1.05 µs	-0.47 µs
Complex Overlap	KMP	2.70	+0.95 µs	+0.92 µs	-0.06 µs	N/A
DNA Sequence	Naive	4.87	N/A	-0.38 µs	-1.34 µs	-0.56 µs
Palindrome Pattern	RabinKarp	17.25	+15.78 µs	N/A	+14.34 µs	+14.17 µs
Worst Case for Naive	KMP	17.64	+8.22 µs	+15.30 µs	+12.63 µs	N/A
Best Case for Boyer-Moore	Naive	2.45	N/A	+0.11 µs	-1.01 µs	-0.58 µs
KMP Advantage Case	KMP	19.69	+13.63 µs	+17.31 µs	+16.25 µs	N/A

Figure 2: Execution time comparison with pre-analysis strategy (green: pre-analysis was faster, red: pre-analysis was slower).

3.1 Key Observations

- KMP dominates in highly repetitive patterns and structured data (e.g., DNA sequences).
- Boyer–Moore excels when character diversity is high and mismatches occur early.
- Rabin–Karp performs best in high-entropy texts but is sensitive to hashing overhead.

- Pre-analysis algorithm is not perfectly designed since overhead (red labels) is seen in most of the test cases in the Figure 2. It sometimes selects the inappropriate algorithm, too.

References

- GeeksForGeeks, *Boyer Moore Algorithm for Pattern Searching*. https://www.geeksforgeeks.org/dsa/boyer-moore-algorithm-for-pattern-searching/