Boyer Moore Algorithm

This algorithm involves constructing two tables. One is Bad symbol shift table and other one is good Suffix Shift table.

Bad Symbol Shift Table:
Given a character c, T(c) is computed as:
- the pattern length m, if c is not among the first m-1 characters of the pattern
- the distance from the rightmost c among the first m-1 characters of the pattern to its last occurrence, otherwise.

The size of the shift is computed by T(c) – k, where k is the number of matched characters. If this result turns out to be less than or equal to 0, then we shift by one position to right.

Good Suffix Shift Table:
The steps to be followed are:
- Check if there is another occurrence of matched pattern not preceded by same character as in its last occurrence. D2 is the distance between such rightmost occurrence of pattern and its rightmost occurrence.
- If not, find the longest prefix of size l, l<k, where k is matched pattern length, that matches the suffix of the same size l. If such a prefix exists, shift d2 is computed as the distance between the prefix and the corresponding suffix
- Otherwise d2 is set to pattern length m

The final distance D is computed by the formula:

$$D = \begin{cases} D_1 & \text{if } k = 0 \\ \max\{D_1, D_2\} & \text{if } k > 0 \end{cases}$$

where $D_1 = \max\{T(c)-k, 1\}$

Let us check out some examples.

1. Construct bad symbol shift table for the pattern:
BAD

c	B	A	D
T(c)	2	1	3

How did we arrive at this table? Here is the logic. We are only supposed to see the first m-1 characters provided that the length of the pattern is m.

Then, here is how we set the length for each of the character:

- As D is the mth character, set it with length of the pattern which is 3
- Letter A is 1 character away from last character

- Letter B is 2 characters away from the last character

2. Construct a bad symbol shift table for the pattern:
GOOD

c	G	O	O	D
T(c)	3		1	4

How did we arrive at those numbers?
- D, is the mth character and we put the length of the pattern
- O is one character away from D
- We have already covered O. Also meaning that we are only going to look for right most occurrence if there is a repetition.
- G is three characters away from B

3. Construct a bad symbol shift table for the pattern BARBER

c	B	A	R	B	E	R
T(c)		4	3	2	1	

4. Construct a good suffix shift table for the ABCBAB
For this table like explained in the beginning we are going to fill up the three part way.

k	pattern	D2
1	ABCBAB	2
2	ABCBAB	4
3	ABCBAB	4
4	ABCBAB	4
5	ABCBAB	4

When k = 1, which indicates the number of matched characters is 1, first look out for another occurrence of B not preceded by A. Look from right end. There is a B preceded by C. The distance between them is 2, so we write the length 2.

When k= 2, look for AB not preceded by B. There is one in the beginning of the string which is preceded by an empty string. The distance between is 4, hence we write 4.

When k = 3, there is no another occurrence of BAB. So we move to next rule. Find the longest prefix of size l, l<k, where k is matched pattern length, that matches the suffix of the same size l.

Let us write all the prefixes and suffixes of the string.

We have: A, AB, ABC, ABCB, ABCBA
And B, AB, BAB, CBAB, BCBAB

We are supposed to look at strings length less than 3. There is AB which satisfies the criteria. A prefix exists, so shift is computed as the distance between the prefix and the corresponding suffix which is 4.

Similar logic applies to remaining cases as well.

5. Construct a bad symbol table for the pattern CONSISTING

c	C	O	N	S	I	S	T	I	N	G
T(c)	9	8				4	3	2	1	10

6. Construct a bad symbol table for the pattern DISGUSTING

c	D	I	S	G	U	S	T	I	N	G
T(c)	9			6	5	4	3	2	1	

7. Apply Boyer-Moore algorithm on the given text and pattern.
Text: BESS_KNEW_ABOUT_BAOBABS
Pattern: BAOBAB

The length of the pattern string is 6.

Let us first create a bad symbol table:

c	A	B	C	D	…	O	…	Z	_
T(c)	1	2	6	6	6	3	6	6	6

We basically cover everything that is going to come up in the text.
_ is basically used for space for better representation.
… represents everything between the mentioned neighbours.

Let us now create the good suffix table.

k	pattern	D2
1	BAOBAB	2
2	BAOBAB	5
3	BAOBAB	5
4	BAOBAB	5
5	BAOBAB	5

Using both the tables, now let us apply the Boyer-Moore algorithm.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
B	E	S	S	_	K	N	E	W	_	A	B	O	U	T	_	B	A	O	B	A	B	S
					x																	
B	A	O	B	A	**B**																	
D1 = T(K) - k = 6 – 0 = 6 Shift by 6																						
									x	**√**	**√**											
						B	A	O	**B**	**A**	**B**											
						D1 = T(_) - k = 6-2 = 4 D2 = 5 D = max{ 4, 5 } = 5 Shift by 5																
															x	**√**						
											B	A	O	B	**A**	**B**						
											D1 = T(_) - k = 6-1 = 5 D2 = 2 D = max{ 5, 2 } = 5 Shift by 5											
																√	**√**	**√**	**√**	**√**	**√**	
																B	**A**	**O**	**B**	**A**	**B**	
																Match found at position 16						

8. Construct bad symbol table and good suffix table for the pattern: 00001
Length of the pattern is: 5

Bad symbol table:

c	0	1
T(c)	1	5

Good suffix table:

k	pattern	D2
1	00001	5
2	00001	5
3	00001	5
4	00001	5

9. Construct bad symbol table and good suffix table for the pattern: 10000
Length of the pattern is: 5

Bad symbol table:

c	0	1
T(c)	1	4

Good suffix table:

k	pattern	D2
1	10000	3
2	10000	2
3	10000	1
4	10000	5

Prefixes and Suffixes:
For a string school:
Prefixes are:
s
sc
sch
scho
schoo
school

and Suffixes are:
l
ol
ool
hool
chool
school

For the string s, sc, sch, scho, schoo, school are all proper prefixes. Similar explanation holds good for a proper suffixes too. A proper prefix or a proper suffix of a string is all the prefix or suffix other than the string itself.

Efficiency Analysis:
When searching for the first occurrence of the pattern in Boyer-Moore algorithm, the worst case efficiency is known to be linear.
If implemented as presented in the original paper, it has worst case running time of O(m+n) only if the pattern does not appear in text. When appears, the worst case in O(mn)