Graphical user interface, text, application

Description automatically generated

This is the question.

<https://www.youtube.com/watch?v=hXqRLILcC1k>

Logic:

Diagram, timeline

Description automatically generated

Graphical user interface, text

Description automatically generated

The value(T)=4 is the length of word to be found 4 and current index of actual character is 0 and -1.

Timeline

Description automatically generated

See here for last t. this t is repetitive as t at start is already there. Hence just update t in table with latest value of t.

Diagram, timeline

Description automatically generated

See we have updated it with 1.bad match table is made from pattern given. This star represents in table any character not present in the pattern.it initial value is length of the pattern that is 4. Diagram, timeline

Description automatically generated

See keep comparing the rightmost character in the input.

Table, timeline

Description automatically generated

Timeline

Description automatically generated

See here t does not match with the s in the input. Hence the value of s is 1 in table and hence shift by 1 to the right.

Diagram, timeline

Description automatically generated

See here t is matched with empty string and hence this space is not any character in the pattern and hence use \* and shift by 4.

Timeline

Description automatically generated

Again, repeat the steps and see here t matched with a and a is not present in any pattern hence shift by 4.

Timeline

Description automatically generated

T matched with s and not matched. s says shift by 1 hence shift by 1.see shifting is done according to the character in the input.

Diagram, timeline

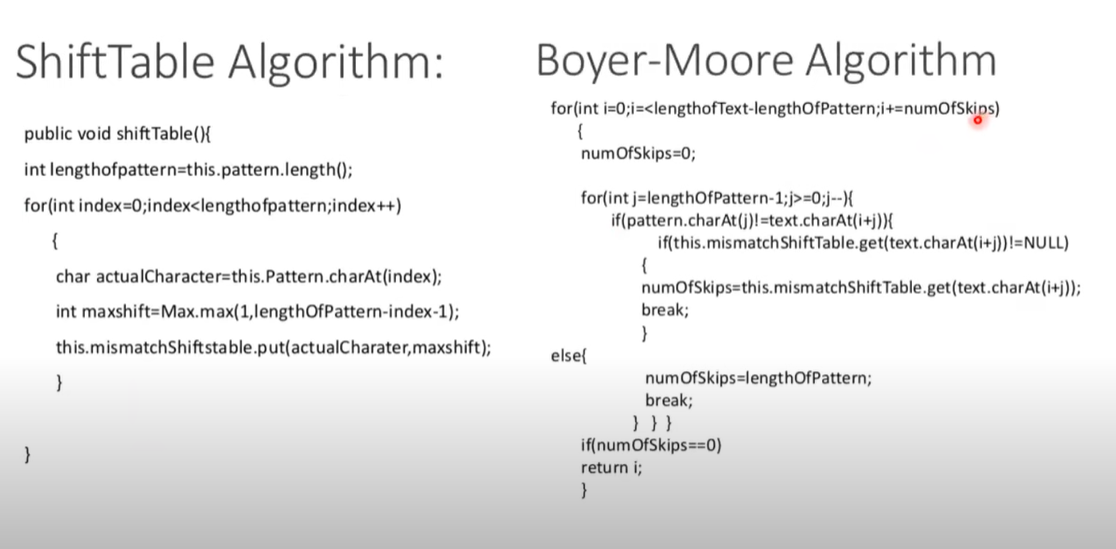
Description automatically generated

Now compare from the right and all are matching.

Text

Description automatically generated

Code by video:



Text, letter

Description automatically generated

<https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/>

this gfg explanation important.

Code:

/\* C++ Program for Bad Character Heuristic of Boyer

Moore String Matching Algorithm \*/

#include <bits/stdc++.h>

using namespace std;

# define NO\_OF\_CHARS 256

// The preprocessing function for Boyer Moore's

// bad character heuristic

void badCharHeuristic( string str, int size,

int badchar[NO\_OF\_CHARS])

{

int i;

// Initialize all occurrences as -1

for (i = 0; i < NO\_OF\_CHARS; i++)

badchar[i] = -1;

// Fill the actual value of last occurrence

// of a character

for (i = 0; i < size; i++)

badchar[(int) str[i]] = i;

}

/\* A pattern searching function that uses Bad

Character Heuristic of Boyer Moore Algorithm \*/

void search( string txt, string pat)

{

int m = pat.size();

int n = txt.size();

int badchar[NO\_OF\_CHARS];

/\* Fill the bad character array by calling

the preprocessing function badCharHeuristic()

for given pattern \*/

badCharHeuristic(pat, m, badchar);

int s = 0; // s is shift of the pattern with

// respect to text

while(s <= (n - m))

{

int j = m - 1;

/\* Keep reducing index j of pattern while

characters of pattern and text are

matching at this shift s \*/

while(j >= 0 && pat[j] == txt[s + j])

j--;

/\* If the pattern is present at current

shift, then index j will become -1 after

the above loop \*/

if (j < 0)

{

cout << "pattern occurs at shift = " << s << endl;

/\* Shift the pattern so that the next

character in text aligns with the last

occurrence of it in pattern.

The condition s+m < n is necessary for

the case when pattern occurs at the end

of text \*/

s += (s + m < n)? m-badchar[txt[s + m]] : 1;

}

else

/\* Shift the pattern so that the bad character

in text aligns with the last occurrence of

it in pattern. The max function is used to

make sure that we get a positive shift.

We may get a negative shift if the last

occurrence of bad character in pattern

is on the right side of the current

character. \*/

s += max(1, j - badchar[txt[s + j]]);

}

}

/\* Driver code \*/

int main()

{

string txt= "ABAAABCD";

string pat = "ABC";

search(txt, pat);

return 0;

}

// This code is contributed by rathbhupendra