horizontal line

**Assignment 04 | Advance Algorithms**

**CE-092**

Assignment submission for Advance Algorithms subject week 4.

nevilparmar24@gmail.com

**─**

**Task 1:**

To Implement RabinKarp’s algorithm for pattern searching.

Code:

/\*

*\* @Author: nevil*

*\* @Date: 2020-07-31 15:35:40*

*\* @Last Modified by: nevil*

*\* @Last Modified time: 2020-07-31 16:18:11*

\*/

#*include* <bits/stdc++.h>

using namespace std;

#*define* *d* 256

void *rabinKarp*(string pat, string txt, int q)

{

int M = pat.*length*();

int N = txt.*length*();

int i, j;

int p = 0;

int t = 0;

int h = 1;

bool found = 0;

int hits = 0;

*for* (i = 0; i < M - 1; i++)

h = (h \* *d*) % q;

*for* (i = 0; i < M; i++)

{

p = (*d* \* p + pat*[*i*]*) % q;

t = (*d* \* t + txt*[*i*]*) % q;

}

*for* (i = 0; i <= N - M; i++)

{

*if* ( p == t )

{

*for* (j = 0; j < M; j++)

{

*if* (txt*[*i+j*]* != pat*[*j*]*)

*break*;

}

*if* (j == M)

{

cout*<<*"Pattern found at index "*<<* i*<<endl*;

found = 1;

}

hits++;

}

*if* ( i < N-M )

{

t = (*d*\*(t - txt*[*i*]*\*h) + txt*[*i+M*]*)%q;

*if* (t < 0)

t = (t + q);

}

}

*if*(!found)

cout *<<* "Pattern not found" *<<* *endl*;

cout *<<* "Total number of hits when the hash values were same : " *<<* hits *<<* *endl*;

}

int *main*()

{

string text, pattern;

int q;// *this is the mode value we will be using in the rabinkarpt algo*

cout *<<* "Enter Your TEXT : ";

cin *>>* text;

cout *<<* "Enter pattern to search : ";

cin *>>* pattern;

cout *<<* "Enter the value of q : ";

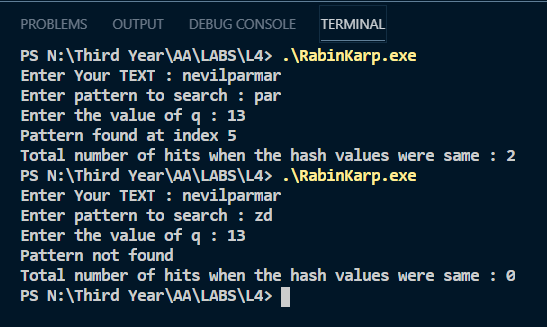
cin *>>* q;

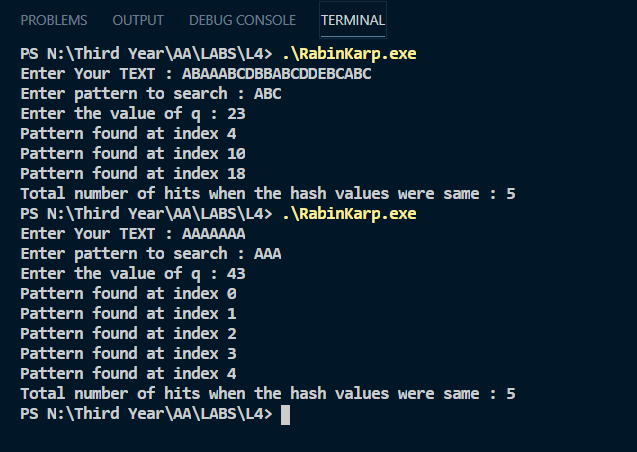
*rabinKarp*(pattern, text, q);

*return* 0;

}

Output:





**Complexity :**

**Spurious Hit :**

When the hash value of the pattern matches with the hash value of a window of the text but the window is not the actual pattern then it is called a spurious hit.

The average and best case running time of the Rabin-Karp algorithm is O(n+m), but its worst-case time is O((n-m+1)m). Worst case of Rabin-Karp algorithm occurs when all characters of pattern and text are the same as the hash values of all the substrings of txt[] match with hash value of pat[].

The example test case for worst scenario from the above inputs are :

TEXT : “AAAAAA”

PATTERN : “AAA”

We can observe the total number of hits when the hash values were the same for this input.

**Application of Rabin-Karp’s Algorithm :**

* Pattern Matching
* To search a string in bigger text
* To implement find , find & replace functionalities in text editors like word, notepad etc.

Nevil Parmar

CE-092

https://nevilparmar.me