

|  |  |
| --- | --- |
| **Name** | Vedant Deshmukh |
| **UID No.** | 2021300025 |
| **Class** | COMPS A (B batch) |
| **Experiment No.** | 9 |

**Aim:** To implement string matching algorithm using the naïve approach and Rabin Karp approach

# Theory:

* **String Matching**

String matching is a process of finding all occurrences of a pattern (a sequence of characters) within a text (a larger sequence of characters).

String matching is a fundamental problem in computer science, which involves finding a substring within a larger string. It is an important problem in various applications, including text processing, image processing, and bioinformatics.

# Rabin Karp

Rabin Karp is a string matching algorithm that uses hashing to find the occurrence of a pattern string within a text string. It was developed by Michael O. Rabin and Richard M. Karp in 1987. The algorithm works by comparing the hash values of the pattern string and all possible substrings of the text string.



# Algorithm:

NaiveStringMatch(T, P)

1. n ← length[T]
2. m ← length[P]
3. for i ← 0 to n - m

4. j ← 0

5. while j < m and P[j] = T[i+j]

6. j ← j + 1

1. if j = m
2. print "Pattern occurs with shift" i
3. RabinKarpMatch(text, pattern):
4. n = length(text)
5. m = length(pattern)
6. pattern\_hash = hash(pattern) 5.
7. for i from 0 to n-m:
8. text\_hash = hash(text[i:i+m]) 8.
9. if pattern\_hash == text\_hash:
10. if pattern == text[i:i+m]:
11. return i
12. return -1

# Code:

















Naive Vs Rabin Karp

250

200

150

100

50

0

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49

size of string

Naive

Rabin-Karp

no of comparision



**Output:**

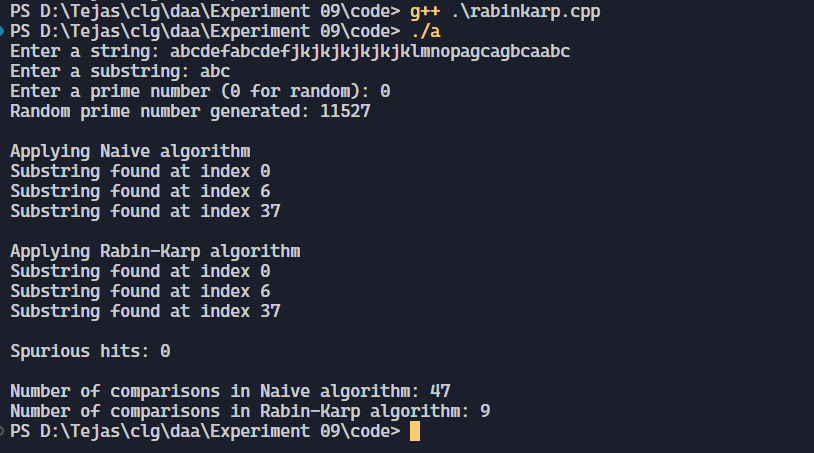


Chart:



# Conclusion:

Rabin-Karp is a simple and efficient string matching algorithm that can find all occurrences of a pattern within a text in linear time. It uses hashing to compare substrings, and can handle any alphabet size. Spurious hits can be minimized by using a strong hash function.