Vidyavardhini's College of Engineering & Technology

Department of Computer Science and Engineering (Data Science)

ACADEMIC YEAR: 2024-25

Course: Analysis of Algorithm Lab

Course code: CSL401

Year/Sem: SE/IV

Experiment No.: 10
Aim: To implement Rabin Karp string matching algorithm
Name: SOHAM HEMENDRA RAUT
Roll Number: 24
Date of Performance: 03/04/2025
Date of Submission: 10/04/2025

Evaluation

Performance Indicator	**Max. Marks**	**Marks Obtained**
Performance	5	
Understanding	5	
Journal work and timely submission.	10	
Total	**20**	

Performance Indicator	**Exceed Expectations (EE)**	**Meet Expectations (ME)**	**Below Expectations (BE)**
Performance	5	3	2
Understanding	5	3	2
Journal work and timely submission.	10	8	4

Checked by

Name of Faculty : Mrs. Komal Champanerkar

Signature :

Date :

❖ **Aim: To implement Rabin Karp string matching algorithm**

❖ **Theory:**

The Rabin-Karp string matching algorithm calculates a hash value for the pattern, as well as for each M-character subsequences of text to be compared. If the hash values are unequal, the algorithm will determine the hash value for next M-character sequence. If the hash values are equal, the algorithm will analyze the pattern and the M-character sequence. In this way, there is only one comparison per text subsequence, and character matching is only required when the hash values match.

Example: For string matching, working module q = 11, how many spurious hits does the Rabin-Karp matcher encounters in Text T = 31415926535.......

1. T = 31415926535.......
2. P = 26
3. Here T.Length =11 so Q = 11
4. And P mod Q = 26 mod 11 = 4
5. Now find the exact match of P mod Q...

❖ **Algorithm:**

Step 1: Initially calculate the hash value of the pattern.

Step 2: Start iterating from the starting of the string:-

- Calculate the hash value of the current substring having length m.
- If the hash value of the current substring and the pattern are same check if the substring is same as the pattern.
- If they are same, store the starting index as a valid answer. Otherwise, continue for the next substrings.

Step 3: Return the starting indices as the required answer.

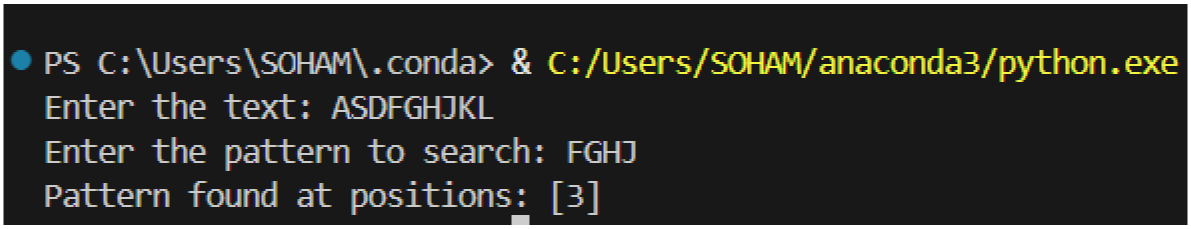
Step 4: Exit.

❖ **Program:**

```
def rabin_karp(text, pattern, d=256, q=101):
    n = len(text)
    m = len(pattern)
    h = pow(d, m-1) % q
    p_hash = 0
    t_hash = 0
    positions = []

    # Calculate initial hash values
    for i in range(m):
        p_hash = (d * p_hash + ord(pattern[i])) % q
        t_hash = (d * t_hash + ord(text[i])) % q

    for i in range(n - m + 1):
        if p_hash == t_hash:
            if text[i:i + m] == pattern:
                positions.append(i)

        # Rolling hash update
        if i < n - m:
            t_hash = (d * (t_hash - ord(text[i]) * h) + ord(text[i + m])) % q
            if t_hash < 0:
                t_hash += q

    return positions

text = input("Enter the text: ")
pattern = input("Enter the pattern to search: ")
```

```
matches = rabin_karp(text, pattern)   # Function Call


if matches:
    print("Pattern found at positions:", matches)
else:
    print("Pattern not found in the text.")
```

- **Output:**



```
PS C:\Users\SOHAM\.conda> & C:/Users/SOHAM/anaconda3/python.exe
Enter the text: ASDFGHJKL
Enter the pattern to search: FGHJ
Pattern found at positions: [3]
```

- **Conclusion:**

The Rabin-Karp algorithm efficiently identifies pattern matches in text using hashing. It's particularly beneficial when searching for multiple patterns due to its rolling hash technique, which minimizes unnecessary comparisons.