Page No.:
Date:

Assignment -3

Group Member -

1. Riya Bhatnagar (46)
2. Shasvi Negi (53)

Hash function

$$H(s) = (C_0 + C_1 * p + C_2 + p^2 + \dots C_{(n-1)} * p^{n-1}) \% m$$

Where

- $H(s)$ → Hash value of strings
- C_i → ASCII value of i^{th} character in the string.
- p → chosen prime no.
- m → chosen large prime no. for modulo operations
- n → length of the string.

Why more Efficient

1. Horner's Method Utilization -
   Employing Horner's Method reduces the no. of multiplication and modulus operations, streamlining the hash calculation process.

2. Fewer Operations -
   With fewer multiplication and

modulus operation the algorithm computation overhead is reduced leading to faster execution.

3. Mitigation Risk of Overflow -
By minimizing the no. of operations, the risk of overflow is mitigated, ensuring the resulting hash value remains within a manageable range.

4. Enhanced Performance -
The streaming hash calculation process enhances the algorithms overall performance, particularly in pattern matching tasks, where efficiency is crucial.

Solved Example

Rabin-Karp Algorithm by Substring Search

1. Initialize Parameters:
- Large String: "The quick brown fox jumps over the lazy dog"
- Substring: "brown"
- Polynomial Hash functions:
$HCS = (c_0 + c_1 * p + c_2 * p^2 + \ldots . c^{n+1} * p^{n+1}) \% m$
S: Substring we're considering.

Page No.:
Date:

$c_i \rightarrow$ ASCII value of i^{th} character of a substring

$p \rightarrow 31$

$m \Rightarrow 1000$

2. Calculate hash value of substring "brown"

- ASCII values:
- 'b' :98, 'r': 114, 'o' : 111, 'w' :119, 'n' :110
- Using the polynomial hash function:

$H(\text{substring}) = (98 + 114 \times 31 + 111 \times 31 \times 31 + 119 \times 31 \times 31 \times 31 + 110 \times 31^4) \mod 1000$

$H(\text{substring}) = 149393157 \mod 1000$

$= 157$

3. Search Process:

- Start with first substring of length 5:
- Substring "thequ"
- calculate hash value using the polynomial hash function
- Compare hash value with the target hash value (157)
- Continue with Next substring:
- Shift one character to the right and recalculate the hash value.
- compare hash value with the target hash value (157)

- Repeat Untill match found or All substring checked

4. Efficiency:

- The polynomial hash function ensures constant time compulation of hash value
- Compare hash value reduces time complexity compared to character-by-character comparision.
- This approach is especially efficient for large strings or repeated substring searches.