ASSIGNMENT 3

GROUP MEMBERS:

1. SHARVI NEGI
2. RIYA BHATNAGAR

HASH FUNCTION

$H(S) = (c_0 + c_1 * p + c_2 * p^2 + ... + c_{n-1} * p^{n-1}) \mod m$

Where:

- H(S)nis the hash value of string S.
- c_i represents the ASCII value of the ith character in the string.
- p is a chosen prime number.
- m is a chosen large prime number for modulo operation.
- n is the length of the string.

WHY MORE EFFICIENT

1. Horner's Method Utilization:

- Employing Horner's method reduces the number of multiplications and modulus operations, streamlining the hash calculation process.

2. Fewer Operations:

- With fewer multiplications and modulus operations, the algorithm's computational overhead is reduced, leading to faster execution.

3. Mitigated Risk of Overflow:

- By minimizing the number of operations, the risk of overflow is mitigated, ensuring the resulting hash values remain within a manageable range.

4. Enhanced Performance:

- The streamlined hash calculation process enhances the algorithm's overall performance, particularly in pattern matching tasks, where efficiency is crucial.

SOLVED EXAMPLE

Rabin-Karp Algorithm for Substring Search

1. Initialize Parameters:

- Large String: "thequickbrownfoxjumpsoverthelazydog"

- Substring: "brown"

- Polynomial Hash Function:

  H(S) = (c_0 + c_1 * p + c_2 * p^2 + ... + c_{n-1} * p^{n-1}) mod m

  S: Substring we're considering

  c_i: ASCII value of the i^{th} character of the substring

  p = 31, m = 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(Substring) = (98 + 114 * 31 + 111 * 31^2 + 119 * 31^3 + 110 * 31^4) mod 1000

  H(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 Until Match Found or All Substrings Checked.

4. Efficiency:

- The polynomial hash function ensures constant time computation of hash values.

- Comparing hash values reduces time complexity compared to character-by-character comparison.

- This approach is especially efficient for large strings or repeated substring searches.