String matching: Rabin Karp Approach

Session No.: 11
Course Name: Design and analysis of algorithm
Course Code: R1UC407B
Instructor Name: Mili Dhar
Duration: 50 Min.
Date of Conduction of Class:

Review of the key concepts

1. Review of previous session

Q: What was the reason that generated worst-case complexity by the naïve method of string matching?

Learning Outcome

Apply the algorithm to solve string matching problems

Analyse the time complexity of the algorithm.

Session Outline

1 Introduction to Rabin Karp approach

2 Step by Step demonstration of Naïve approach

3 Apply Rabin Karp approach to find a pattern in a given text

4 Analyse its complexity

Rabin Karp Approach

- The **Rabin-Karp algorithm** checks the given pattern within a text by moving window one by one, but without checking all characters for all cases.
- It finds the hash value of text and pattern.
- If the hash values match, then there is a possibility that the pattern and the substring are equal, and we can verify it by comparing them character by character.
- If the hash values do not match, then we can skip the substring and move on to the next one.

In the next section, we will understand how to calculate hash values.

Demonstration

Step 1: Suppose that the given text string is:

and the pattern to be searched in the above text string is:

Step 2: Let us start by assigning a numerical value (v)/weight for the characters we will use in the problem. Here, we have selected the first ten alphabets only (i.e., A to J).

Demonstration

Step 3: Let x be the length of the pattern, and y be the length of the text string. Here, we have y = 10 and x = 3. Moreover, let n be the number of characters present in the input set. Since we have taken input set as {A, B, C, D, E, F, G, H, I, J}. Therefore, n will be equal to 10. But, We can assume any preferable value for n.

Step 4: Let us now start calculating the hash value of the pattern:

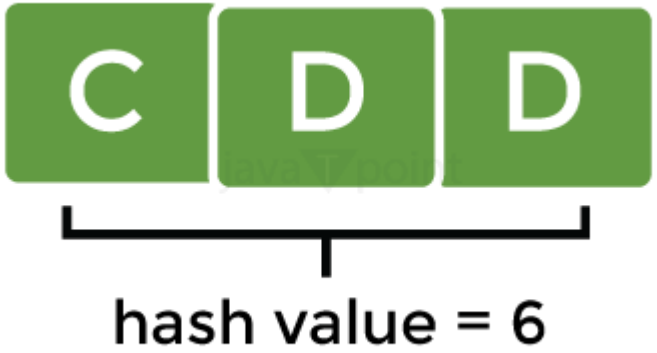
Hash value = $\sum(v * d^{x-1}) mod\ q$ ***where,***

v = weight of the characters

d = number of character in the input set

x = length of pattern

q = a prime number



$= ((3 * 10^2) + (4 * 101) + (4 * 10^0))\ mod\ 13$

$= 344\ mod\ 13$

$= 6$

Demonstration

Step 5: We will now calculate the hash value for the text window of size y.

$$= ((1 * 10^2) + (2 * 101) + (3 * 10^0)) \bmod 13$$

$$= 123 \bmod 13$$

$$= 6$$

Step 6: We will now compare the hash value of the pattern with the hash value of the text string. If they match, we will perform the character-matching operation. Therefore, we will start matching the characters between ABC and CDD. Since the pattern does not match the first window, we will move to the next window.

Demonstration

Step 7: We will calculate the hash value of the next window by subtracting the first term and adding the next term, as shown below:

$$s = ((1 * 10^2) + ((2 * 10^1) + (3 * 10^0)) * 10 + (3 * 10^0)) \bmod 13$$

$$= 233 \bmod 13$$

$$= 12$$

We will use the former hash value in the following way in order to optimize this process:

$$s = (d * (t - v[\text{character to be eliminated}] * h) + v[\text{character to be included}]) \bmod 13$$

$$= ((10 * (6 - 1 * 9) + 3) \bmod 13$$

$$= 12$$

$$\text{Where, } h = n^{y-1} = 10^{3-1} = 100$$

Pseudocode : Rabin Karp

```
n = t.length
m = p.length
h = d^(x-1) mod q
hash_p = 0
hash_t = 0
for i = 1 to m
    hash_p = (d*hash_p + p[i]) mod q
    hash_t = (d*hash_t + t[i]) mod q
for s = 0 to n - m
    if hash_p = hash_t
        if p[1.....m] = t[s + 1..... s + m]
            print "pattern found at position" s
    If s < n-m
        hash_t  = (d* (hash_t - T[s]*h) + T[s + m ]) mod q
```

Activity 2: Apply Rabin Karp method to the given scenarios

Input:

Main String: ABAAABCDBBABCDDEBCABC

Pattern: ABC

Analyse the Complexity of the followings:

Example:1
txt = "BBACCAADDEE";
pat = "HBB";

Note: The best case time complexity O(n+m).

Example 2:
txt[] = "DDDDDDDDDDDD";
pat[] = "DDDDD";

The worst case time complexity O(nm)

Example 3:
txt[] = "VVVVVVVVVVVVK";
pat[] = "VVVK";

Post class assessment:

```
txt[] = "VVVVVVVVVVVVVK";
pat[] = "VVVK";
```

In the next class we will going through devide and conquer algorithmic approach

Thank You