Algorithm Engineering

Md.Atiqur Rahman
Rabin-Karp Algorithm

Rabin Karp Algorithm

The main idea behind Rabin-Karp is to convert strings into numeric hashes so we can compare numbers instead of characters. This makes the process much faster.

We first compute the hash of the pattern, then compare it with the hashes of all substrings of the text. If the hashes match, we do a quick character-by-character check to confirm (to avoid errors due to hash collisions and this step will incur high time complexity if there are enough collisions).

Hash function is simple (for now). Let's say I have string "abc" its hash value is,

Hash("abc") = 31^{2}*1 + 31^{1}*2 + 3, here 'a' has the value 1 and similarly 'z' has the value 26, 31 is the base and 31^{2} == 31*31.

For the simplicity, we will ignore all the modular functions **[FOR NOW]**.

HOW THE ALGORITHM WORKS

Let’s say I have text as “abcd” and pattern as “bc”. Then, the hash values for the pattern is, Hash(“bc”) = 31*2 + 3.

Now, should we calculate a single hash value for “abcd”? Obviously, not. We prepare hash values for every available prefix of the “abcd”.

Hash[0] = Hash(“a”) = 1

We don’t need to compute Hash of every prefix since we can use previous computed hash.

Hash[1] = Hash(“ab”) = Hash[0]*31 + 2 = (1)*31 + 2 = 31*1 + 2

Hash[2] = Hash(“abc”) = Hash[1]*31 + 3 = (31*1 + 2)*31 + 3 = 31^{2}*1 + 31*2 + 3

Hash[3] = Hash(“abcd”) = Hash[2]*31 + 4 = (31^{2}*1 + 31*2 + 3)*31 + 4
= 31^{3}*1 + 31^{2}*2 +31*3 + 4

We now compare each substring of the text, equal in length to the pattern, with the pattern itself.

Continued

At first, we match text[0..1] with pattern since pattern is of size 2.

Hash[1] = Hash(text[0..1]) = Hash(“ab”) = 31*1 + 2 != Hash(pattern) = 31*2 + 3

So, no match. Now, we go for, text[1..2] and here we need to do some calculation,

Hash[2] = Hash(text[0..2]) = Hash(“abc”) = 31^{2}*1 + 31*2 + 3

Hash[1-1] = Hash[0] = Hash(text[0..0]) = Hash(“a”) = 1

Hash[1-1]*31^{2-1+1} = 31^{2}*1

Hash[1..2] = Hash[2] - Hash[1-1]*31^{2-1+1} = (31^{2}*1 + 31*2 + 3) - (31^{2}*1) = 31*2 + 3 = Hash(pattern)

It’s a match. Since, text[1..2] = “bc” and pattern is “bc” so it should match.

ANOTHER EXAMPLE

My student id is “01234567” and the pattern is always the last two digits of the student id, in my case it is “67”. Now, ‘0’ has the value 1 and ‘9’ has the value 10. Since, I already know there’s a match, hence let’s directly calculate the hash values and try to see if it’s a match.

Hash[L..R] = Hash[R] - Hash[L-1]*base^{R-L+1}, why do we need this equation ? because, we need to remove the hash value for the 0..L-1 prefix. But, if we just subtract the Hash[L-1] it will not work since, that prefix is already multiplied by base for R-L+1 times to reach Hash[R].

Hash[6..7] = Hash[7] - Hash[5]*31^{2}

= (31^{7}*1 + 31^{6}*2 + 31^{5}*3 + 31^{4}*4 + 31^{3}*5 + 31^{2}*6 + 31*7 + 8) -

(31^{5}*1 + 31^{4}*2 + 31^{3}*3 + 31^{2}*4 + 31*5 + 6) * 31^{2}

Since after 5 we have more 2 characters at position 6 and 7, all the values of the previous characters will be multiplied by 31 for 2 times.

= (31^{7}*1 + 31^{6}*2 + 31^{5}*3 + 31^{4}*4 + 31^{3}*5 + 31^{2}*6 + 31*7 + 8) -

(31^{7}*1 + 31^{6}*2 + 31^{5}*3 + 31^{4}*4 + 31^{3}*5 + 31^{2}*6) = 31*7 + 8, it’s a match.

Class Work with actual SID

My actual student id is “180041123”, and the pattern will be “23”. Now I need to calculate Hash[6..7] and Hash(“23”), to show both the values are same.

Write the same in your notebook by hand, scan the page(s), convert them to a PDF, and submit it in the classroom to receive today's attendance and classwork marks.