Searching for Patterns | Set 3 (Rabin-Karp Algorithm)

Given a text *txt[0..n-1]* and a pattern *pat[0..m-1]*, write a function *search(char pat[], char txt[])* that prints all occurrences of *pat[]* in *txt[]*. You may assume that n > m.

Examples:
1) Input:

```
txt[] =  "THIS IS A TEST TEXT"
pat[] = "TEST"
```

Output:

Pattern found at index 10

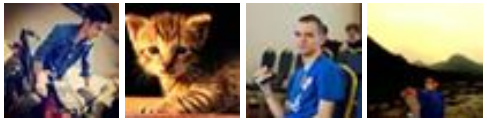
2) Input:

```
txt[] =  "AABAACAADAABAAABAA"
pat[] = "AABA"
```

Output:

```
Pattern found at index 0
Pattern found at index 9
Pattern found at index 13
```

The Naive String Matching algorithm slides the pattern one by one. After each slide, it one by one checks characters at the current shift and if all characters match then prints the match.
Like the Naive Algorithm, Rabin-Karp algorithm also slides the pattern one by one. But unlike the Naive algorithm, Rabin Karp algorithm matches the hash value of the pattern with the hash value



of current substring of text, and if the hash values match then only it starts matching individual characters. So Rabin Karp algorithm needs to calculate hash values for following strings.

1) Pattern itself.
2) All the substrings of text of length m.

Since we need to efficiently calculate hash values for all the substrings of size m of text, we must have a hash function which has following property.
Hash at the next shift must be efficiently computable from the current hash value and next character in text or we can say *hash(txt[s+1 .. s+m])* must be efficiently computable from *hash(txt[s .. s+m-1])* and *txt[s+m]* i.e., *hash(txt[s+1 .. s+m])= rehash(txt[s+m], hash(txt[s .. s+m-1])* and rehash must be O(1) operation.

The hash function suggested by Rabin and Karp calculates an integer value. The integer value for a string is numeric value of a string. For example, if all possible characters are from 1 to 10, the numeric value of "122" will be 122. The number of possible characters is higher than 10 (256 in general) and pattern length can be large. So the numeric values cannot be practically stored as an integer. Therefore, the numeric value is calculated using modular arithmetic to make sure that the hash values can be stored in an integer variable (can fit in memory words). To do rehashing, we need to take off the most significant digit and add the new least significant digit for in hash value. Rehashing is done using the following formula.

*hash(txt[s+1 .. s+m]) = d (hash(txt[s .. s+m-1]) – txt[s]*h) + txt[s + m]) mod q*

hash(txt[s .. s+m-1]) : Hash value at shift *s*.
hash(txt[s+1 .. s+m]) : Hash value at next shift (or shift *s*+1)
d: Number of characters in the alphabet
q: A prime number
h: d^(m-1)

```
/* Following program is a C implementation of the Rabin Karp Algorithm
   given in the CLRS book */

#include<stdio.h>
#include<string.h>

// d is the number of characters in input alphabet
#define d 256
```

Geometric Algorithms

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding "extern" keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
/*  pat  -> pattern
    txt  -> text
    q    -> A prime number
*/
void search(char *pat, char *txt, int q)
{
    int M = strlen(pat);
    int N = strlen(txt);
    int i, j;
    int p = 0;  // hash value for pattern
    int t = 0; // hash value for txt
    int h = 1;

    // The value of h would be "pow(d, M-1)%q"
    for (i = 0; i < M-1; i++)
        h = (h*d)%q;

    // Calculate the hash value of pattern and first window of text
    for (i = 0; i < M; i++)
    {
        p = (d*p + pat[i])%q;
        t = (d*t + txt[i])%q;
    }

    // Slide the pattern over text one by one
    for (i = 0; i <= N - M; i++)
    {

        // Chaeck the hash values of current window of text and patter
        // If the hash values match then only check for characters on
        if ( p == t )
        {
            /* Check for characters one by one */
            for (j = 0; j < M; j++)
            {
                if (txt[i+j] != pat[j])
                    break;
            }
            if (j == M)  // if p == t and pat[0...M-1] = txt[i, i+1, .
            {
                printf("Pattern found at index %d \n", i);
            }
        }

        // Calulate hash value for next window of text: Remove leading
        // add trailing digit
        if ( i < N-M )
```

```
        {
            t = (d*(t - txt[i]*h) + txt[i+M])%q;

            // We might get negative value of t, converting it to posi
            if(t < 0)
              t = (t + q);
        }
    }
}

/* Driver program to test above function */
int main()
{
    char *txt = "GEEKS FOR GEEKS";
    char *pat = "GEEK";
    int q = 101;  // A prime number
    search(pat, txt, q);
    getchar();
    return 0;
}
```

The average and best case running time of the Rabin-Karp algorithm is O(n+m), but its worst-case time is O(nm). Worst case of Rabin-Karp algorithm occurs when all characters of pattern and text are same as the hash values of all the substrings of txt[] match with hash value of pat[]. For example pat[] = “AAA” and txt[] = “AAAAAAA”.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

References:

http://net.pku.edu.cn/~course/cs101/2007/resource/Intro2Algorithm/book6/chap34.htm

http://www.cs.princeton.edu/courses/archive/fall04/cos226/lectures/string.4up.pdf

http://en.wikipedia.org/wiki/Rabin-Karp_string_search_algorithm

Related Posts:
Searching for Patterns | Set 1 (Naive Pattern Searching)
Searching for Patterns | Set 2 (KMP Algorithm)

Recent Comments

Related Tpoics:

- Print all possible words from phone digits
- Printing Longest Common Subsequence
- Suffix Array | Set 2 (nLogn Algorithm)
- Rearrange a string so that all same characters become d distance away
- Recursively remove all adjacent duplicates
- Find the first non-repeating character from a stream of characters
- Dynamic Programming | Set 33 (Find if a string is interleaved of two other strings)
- Remove “b” and “ac” from a given string

Writing code in comment? Please use **ideone.com** and share the link here.

27 Comments GeeksforGeeks

Sort by Newest

Prateek Surana • 9 days ago

Nice explanation

```
a.out: main.c search.c
gcc -o a.out main.c main.h search.c
clean:
rm -vf a.out *~
```

Reply • Share ›

MM • 5 months ago

In the code, I don't see the variable "h" being used anywhere. What's the use c

// The value of h would be "pow(d, M-1)%q"

for (i = 0; i < M-1; i++)

h = (h*d)%q;

Reply • Share ›

RAHUL KUMAR ➔ MM • 4 months ago

chk this line of code....
t = (d*(t - txt[i]*h) + txt[i+M])%q;

Reply • Share ›

renu • 6 months ago

why are we choosing 'q' to be a prime number?and also in hashing we always is so?can someone please reply?

Reply • Share ›

RAHUL KUMAR ➔ renu • 4 months ago

to calculate the mod value we need prime number .. go through ur bas prime... good luck :)

lizard • 10 months ago

This part in the code is wrong...

```
// Calculate the hash value of pattern and first window of text
   for (i = 0; i < M; i++)
   {
       p = (d*p + pat[i])%q;
       t = (d*t + txt[i])%q;
   }
```

This should be....

```
// Calculate the hash value of pattern and first window of text
   for (i = M-1; i >=0; i--)
   {
       p = (d*p + pat[i])%q;
       t = (d*t + txt[i])%q;
   }
```

Please go through this for my say,
http://en.wikipedia.org/wiki/R...

kartik ➦ lizard • 10 months ago

It looks correct to me.

Consider the strings as strings of digits. When you want to calculate it
from from first digit.
p = 2
p = 2*10 + 3
p = 23*10 + 1
p = 231*10 + 4
p = 2314

Reply • Share ›

lizard kartik • 10 months ago
Thanks....got it..

Reply • Share ›

abhishek08aug • a year ago
Intelligent :D

Reply • Share ›

Vibhu Tiwari • a year ago
This is the source code for pattern searching in much less effort with the time
for various strings by passing the lengths of the two strings to be matched.The
the number of times that substring occurs in the string.

```
#include<stdio.h>
#include<conio.h>
void patternsearch(char *a,char *b,int n,int m)
{ int k,count=0,j=0,i=0,c=0;
while(i!=n)
{ if(j==m)
{j=0;
c=c+1;count=0;
i=c;}
k=a[i]-b[j];
if(k==0){
```

```
count++;}
if(count==m)
{printf("Pattern Match found\n");}
i=i+1;
```

see more

1 ▲ | ▼ • Reply • Share ›

Castle Age ➦ Vibhu Tiwari • 2 months ago

How? if the a pattern not found, i starts from the C+1. The worst case i

complexity is not n in the following case:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB.
AAAAAAAB

▲ | ▼ • Reply • Share ›

Atri • a year ago

We can save the string matching by using multiple hashing.For e.g. Use one h value,and pass the first hash value as input to a second(different) hash value.I function are the same for two strings,we can safely assume that the two string introducing more hash functions.

▲ | ▼ • Reply • Share ›

NitHish Divakar ➦ Atri • 7 months ago

This can only give probablistic matching no matter howmany hash func hashing=> probablity is high...which is acceptable for most practical ap

▲ | ▼ • Reply • Share ›

sparco • 2 years ago

@sandeep

Can u explain the modular arithmetic in calculation of next range's hash value

```
// The value of h would be "pow(d, M-1)%q"
    for (i = 0; i < M-1; i++)
        h = (h*d)%q;

t = (d*(t - txt[i]*h) + txt[i+M])%q;
```

Ishara • 2 years ago

Great Article! Simple & easy to understand. Thx.

Abhinav Kumar • 3 years ago

In Rabin Karp Algo we can modify it with the initial and last character checking
index i and then if both are same then we check the chars in between otherwis
This reduces the computation involves in hashing.

ricky • 3 years ago

@geeksforgeek....Most of The program posted by you is not running when i pa
program is saying so please try to take care of formatting & re-post it again ..h

prog.c:3: error: stray ‘\302’ in program
prog.c:3: error: stray ‘\240’ in program
prog.c:6: error: stray ‘\302’ in program
prog.c:6: error: stray ‘\240’ in program
prog.c:9: error: stray ‘\302’ in program
prog.c:9: error: stray ‘\240’ in program
& so on

shubh • 3 years ago

Another good link about Rabin-Karp

http://courses.csail.mit.edu/6.006/spring11/rec/rec06.pdf

shek8034 ➦ shubh • 11 months ago

Nice Explanation.. Thanks for the link

```
/* Paste your code here (You may delete these lines if not writ
```


Anuj Jindal ➦ shubh • a year ago

in the below given link:
http://courses.csail.mit.edu/6...

h(Si+1) = [(h(Si) (10^5
?rst digit of Si)) 10 + next digit after Si
] mod m

SHOULD BE
h(Si+1) = [(h(Si) (10^4
?rst digit of Si)) 10 + next digit after Si
] mod m

correct me if I am wrong..

kp101090 ➦ shubh • 3 years ago

Its Awesome !! Thanks for the Link .

 • Reply • Share ›

hari6988 • 3 years ago

The integer value for a string is just multiplication of the ASCII values of all cha

But, in the code, this is not done...can u explain how u calculate the integer ha d=256 stand for ?

 • Reply • Share ›

Sandeep hari6988 • 3 years ago

@hari6988: As mentioned in the post, the multiplication is done using r

The following loop calculates hash values (or modular multiplication) fo of size m of text.

```
// Calculate the hash value of pattern and first window of text
    for (i = 0; i < M; i++)
    {
        p = (d*p + pat[i])%q;
        t = (d*t + txt[i])%q;
    }
```

And the following line of code calculates hash values (or modular multi of text

```
t = (d*(t - txt[i]*h) + txt[i+M])%q;
```

d is the number of possible characters in pattern and text. If we consid

value is 256.

Reply • Share ›

martin ➚ Sandeep • 3 years ago

@sandeep ..also can you please post the comparison betweer posted ..???

Reply • Share ›

sparco ➚ martin • 2 years ago

Example)

String bacbababababacaab
Pattern abababaca

LPS Array
0123456
abababaca
0012301 lps[4]=3=>pattern[4]=String[9]
By using same criteria to form lps array , the last match avoiding backtracking to the first element.

i=10 j=4
bacbababababa|caab
ababa|ca

Pattern found at i - len(pattern)

Reply • Share ›

sparco ➚ sparco • 2 years ago

Apologies for the wrong reply

Reply • Share ›

martin ➚ Sandeep • 3 years ago

@sandeep..can you also please post the The Boyer-Moore stri

• Reply • Share ›

Subscribe Add Disqus to your site