



# 컴퓨터 알고리즘과 실습



---

학번 : 2016110056

학과 : 불교학부

이름 : 박승원

날짜 : 2017년 5월 5일

---

# 문제 1. 스트링 매칭 알고리즘 구현

- Text String과 Pattern String을 입력받아 1차원 배열에 저장하고, Text String에서 Pattern String을 모두 찾는 프로그램을 Brute-Force, KMP, Rabin-Karp 알고리즘을 이용하여 작성하시오.
  - 각 알고리즘의 코드를 제출하시오.
  - 아래의 Text String과 Pattern String을 입력한 후 각 알고리즘의 결과와 수행시간(단위는 적절하게 알아서)을 출력하시오.

Text String: A STRING SEARCHING EXAMPLE CONSISTING OF A GIVEN PATTERN STRING

Pattern String: STRING

```
// 2016110056 박승원
#include<cmath>
#include <time.h>
#include<string.h>
#include<iostream>
#include<chrono>
using namespace std;
using namespace chrono;

char text [] = "abababacabababcacaabbc";//A STRING SEARCHING EXAMPLE CONSISTING OF A GIVEN
PATTERN STRING";
char pattern [] = "abababca";
const int t_sz = sizeof(text)-1;
const int p_sz = sizeof(pattern)-1;

void brute()
{
    for(int i=0; i<t_sz; i++) {
        bool match = true;
        for(int j=0; j<p_sz; j++) if( text[i+j] != pattern[j]) match = false ;
        if( match ) cout << "Match found at index " << i << endl;
    }
}
```

```

        if(match) cout << pattern << " is found at index " << i << endl;
    }
}

unsigned long get_num_from_string(char* p, int n)
{ // 포인터 p로부터 n개의 문자를 하나의 숫자로 리턴한다 .
    unsigned long r = 0;
    for(long i = n-1, m = 1; i >= 0; i--, m *= 26) r += (*(p+i) - 'a') * m;
    return r;
}

void rk()
{
    unsigned long pi = get_num_from_string(pattern, p_sz);
    for(int i=1; i<t_sz-p_sz+1; i++)
        if(get_num_from_string(text+i, p_sz) == pi)
            cout << pattern << " is found at index " << i << endl;
}

int prefixTable [p_sz];
int generate_table (char* p, int n)
{ // 포인터로부터 n개의 문자에서 최대 접두부의 인덱스를 반환한다 ..
    for(int i=n-1; i>0; i--) //i = string length
        bool match = true;
        for(int j=0; j<i; j++) if(p[j] != p[n-i+j]) match = false;
        if(match) return i-1;
    }
    return -1;
}

void kmp()
{
    prefixTable [0] = -1;
    for(int i=0; i<p_sz; i++) prefixTable [i+1] = generate_table (pattern, i+1);
    for(char *pp = pattern, *pt = text; pt != text + t_sz;) {
        if(*pt != *pp) {
            if(pp == pattern) pt++;
            pp = pattern + prefixTable [pp - pattern] + 1;
        } else pt++, pp++;
        if(pp == pattern + p_sz)
            cout << pattern << " is found at index " << pt - text - p_sz << endl;
    }
}

```

```
}
```

```
int main()
{
    auto from = system_clock::now();
    brute();
    auto to = system_clock::now();
    cout << "brute " << (to - from).count() / 1000 << " miliseconds" << endl;

    from = system_clock::now();
    rk();
    to = system_clock::now();
    cout << "rk " << (to - from).count() / 1000 << " miliseconds" << endl;

    from = system_clock::now();
    kmp();
    to = system_clock::now();
    cout << "kmp " << (to - from).count() / 1000 << " miliseconds" << endl;
}
```

```
----- 문제 2번 실행을 종료합니다. -----
//2016110056 박승원
#include<cmath>
#include <time.h>
#include<string.h>
#include<iostream>
#include<chrono>
using namespace std;
using namespace chrono;

char text[] = "abababacabababcacaabbc";//A STRING SEARCHING EXAMPLE CONSISTING OF A
                                         GIVEN PATTERN STRING";
----- 문제 1번 실행을 시작합니다. -----
./1.x
abababca is found at index 8
brute 39 miliseconds
abababca is found at index 8
rk 3 miliseconds
abababca is found at index 8
kmp 4 miliseconds
----- 문제 1번 실행을 종료합니다. -----
```

## 참고 코드

- 아래 코드들은 참고를 위해 첨부하였지만 원하는 언어로 코드도 각자 구현하면 됩니다. 강의자료 코드 역시 참고해도 됩니다. 알고리즘은 같으나 구현방법에 따라 인덱스등은 조금씩 다를 수 있으니 알고리즘을 잘 이해했으면 알고리즘대로 결과 패턴을 잘 찾도록 구현은 각자 편한 방식으로 하면 됩니다.

```
void brutesearch(char *p, char *a) { //p : Pattern String, a : Text String
    int i, j, m = strlen(p), n = strlen(a);
    for (i = 0; i<=n-m; i++) {
        for (j = 0; j<m; j++) {
            if (a[i+j] != p[j]) break;
        }
        if (j == m) { //결과값 출력 }
    }
}
```

```

void kmpsearch(char *p, char *a) {
    int i, j, m = strlen(p), n = strlen(a);
    initSP(p);
    for (i = 0, j = -1; i<=n-1; i++) {
        while ((j >= 0) && (p[j+1] != a[i])) j = SP[j];
        if(p[j+1] == a[i] ) j++;
        if (j == m-1) {
            // 결과값 출력
            j=SP[j];
        }
    }
}

initSP(char *p) {
    int i, j, m = strlen(p);
    SP[0] = -1;
    for (i = 1, j = -1; i <= m-1; i++) {
        while ((j >= 0) && (p[j+1] != p[i])) j = SP[j];
        if(p[j+1]==p[i]) j++;
        SP[i]=j;
    }
}
int *SP; // SP에 대한 메모리는 p의 크기만큼 동적 할당 해야한다.
const int q = 33554393;
const int d = 26;

```

```

M <- 11
c <- [5, 3, 1]
ar <- [1,2,1,2,1, 0,0,0,0,0, 0]
FUNCTION fill_table(idx) :
    m <- min(ar[idx-1], ar[idx-3])
    n <- min(m, ar[idx-5])
    RETURN n+1
ENDFUNCTION

for i in range(5, 11):
    ar[i] <- fill_table (i)
ENDFOR
OUTPUT ar

```

함수에서 2번의 비교와 2번의 대입이 이루어지고 각 요소에 한번씩 함수가 호출되므로,  $T(4n) = O(n)$ 이다.

1	2	1	2	1							
1	2	1	2	1	2						
1	2	1	2	1	2	3					
1	2	1	2	1	2	3	2				
1	2	1	2	1	2	3	2	3			
1	2	1	2	1	2	3	2	3	2		
1	2	1	2	1	2	3	2	3	2	3	

```
M = 11
c = [5, 3, 1]
ar = [1,2,1,2,1, 0,0,0,0,0, 0]
```

```
def fill_table (idx) :
    m = min(ar[idx-1], ar[idx-3])
    n = min(m, ar[idx-5])
    return n+1

for i in range(5, 11):
    ar[i] = fill_table (i)

print ar
```

```
zezeon@ubuntuZ:~/Programming/python$ python coin.py
[1, 2, 1, 2, 1, 2, 3, 2, 3, 2, 3]
```

```

void rksearch(char *p, char *a) {
    int i,j, dM = 1, h1 = 0, h2 = 0, flag;
    int m = strlen(p), n = strlen(a);
    for (i = 1; i < m; i++) dM = (d*dM) % q;
    for (i = 0; i < m; i++) {
        h1 = (h1*d+index(p[i])) % q;
        h2 = (h2*d+index(a[i])) % q;
    }
    for (i = 0; i< n-m+1; i++) {
        if (h1 == h2) {
            flag=1;
            for(j=i; j<m; j++) if(a[j]!=p[j-i]) { flag=0; break;}
        }
        if(flag) // 결과값 출력
        if(s<n-m) {
            h2 = (h2-index(a[i])*dM) % q;
            h2 = (h2*d+index(a[i+m])) % q;
        }
    }
}

```

## 참고코드

**NAIVE-STRING-MATCHER ( $T, P$ )**

```

1   $n = T.length$ 
2   $m = P.length$ 
3  for  $s = 0$  to  $n - m$ 
4      if  $P[1..m] == T[s + 1..s + m]$ 
5          print “Pattern occurs with shift”  $s$ 

```

누가 먼저 시작하는지에 따라 승패가 갈린다.

테이블 상에서 좌측 혹은 위쪽 혹은 좌상에 L이 있으면 승리할 수 있다. 좌에서 하나를 빼거나 우에서 하나를 빼거나 양쪽에서 하나를 빼는 경우가 상대의 패가 되는 경우이므로.

	0	1	2	3	4	5
0	L	W	L	W	L	W
1	W	W	W	W	W	W
2	L	W	L	W	L	W
3	W	W	W	W	W	W
4	L	W	L	W	L	W
5	W	W	W	W	W	W

```
m <- [['L']*11 for i in range(11)]  
ENDFOR  
for i in range(5):  
    m[2*i+1][0] <- 'W'  
    m[0][2*i+1]='W'  
ENDFOR  
for x in range (1,11) :  
    for y in range (1,11) :  
        IF m[x][y-1] is 'L' OR m[x-1][y] is 'L' OR m[x-1][y-1] is 'L':  
            m[x][y] <- 'W'  
        ENDIF  
    ENDFOR  
ENDFOR  
for a in m:  
    OUTPUT a;
```

```
m = [['L']*11 for i in range(11)]  
for i in range(5):  
    m[2*i+1][0] = 'W'  
    m[0][2*i+1]='W'  
for x in range(1,11) :  
    for y in range(1,11) :  
        if m[x][y-1] is 'L' or m[x-1][y] is 'L' or m[x-1][y-1] is 'L':  
            m[x][y] = 'W'  
  
for a in m:  
    print a;
```

```

zezeon@ubuntuZ:~/Programming/basicProgramming$ python game.py
['L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L']
['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W']
['L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L']
['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W']
['L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L']
['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W']
['L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L']
['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W']
['L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L']
['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W']
['L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L']
['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W']
['L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L']
['W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W', 'W']
['L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L', 'W', 'L']

```

위의 도표 상에서 보면 승리한다.

## 참고코드

RABIN-KARP-MATCHER( $T, P, d, q$ )

```

1   $n = T.length$ 
2   $m = P.length$ 
3   $h = d^{m-1} \bmod q$ 
4   $p = 0$ 
5   $t_0 = 0$ 
6  for  $i = 1$  to  $m$            // preprocessing
7       $p = (dp + P[i]) \bmod q$ 
8       $t_0 = (dt_0 + T[i]) \bmod q$ 
9  for  $s = 0$  to  $n - m$        // matching
10     if  $p == t_s$ 
11         if  $P[1..m] == T[s + 1..s + m]$ 
12             print "Pattern occurs with shift"  $s$ 
13         if  $s < n - m$ 
14              $t_{s+1} = (d(t_s - T[s + 1]h) + T[s + m + 1]) \bmod q$ 

```

# 참고코드

KMP-MATCHER( $T, P$ )

```
1   $n = T.length$ 
2   $m = P.length$ 
3   $\pi = \text{COMPUTE-PREFIX-FUNCTION}(P)$ 
4   $q = 0$                                 // number of characters matched
5  for  $i = 1$  to  $n$                   // scan the text from left to right
6    while  $q > 0$  and  $P[q + 1] \neq T[i]$ 
7       $q = \pi[q]$                       // next character does not match
8      if  $P[q + 1] == T[i]$ 
9         $q = q + 1$                     // next character matches
10     if  $q == m$                       // is all of  $P$  matched?
11       print "Pattern occurs with shift"  $i - m$ 
12        $q = \pi[q]$                   // look for the next match
```

V는 바이러스의 수, m은 분, B는 박테리아의 수.

$$V_m = 2^m$$

$$B_{m+1} = (B_m - 2^m) \times 2$$

$$B_{m+1} = 2B_m - 2^{m+1}$$

$$\frac{B_{m+1}}{2^{m+1}} = \frac{B_m}{2^m} - 1$$

$$b_m = \frac{B_m}{2^m}$$

$$b_0 = n, b_m = n - m$$

$\therefore B_m = 2^m(n - m) = 0$   $\Rightarrow$ 므로 n분 이후에 박테리아의 수는 0이 된다.