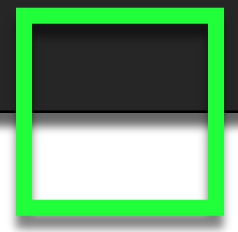
#고급반
#4주차

KMP

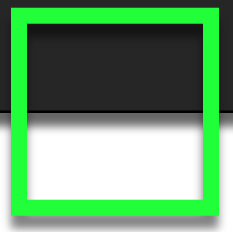
T. 성창호



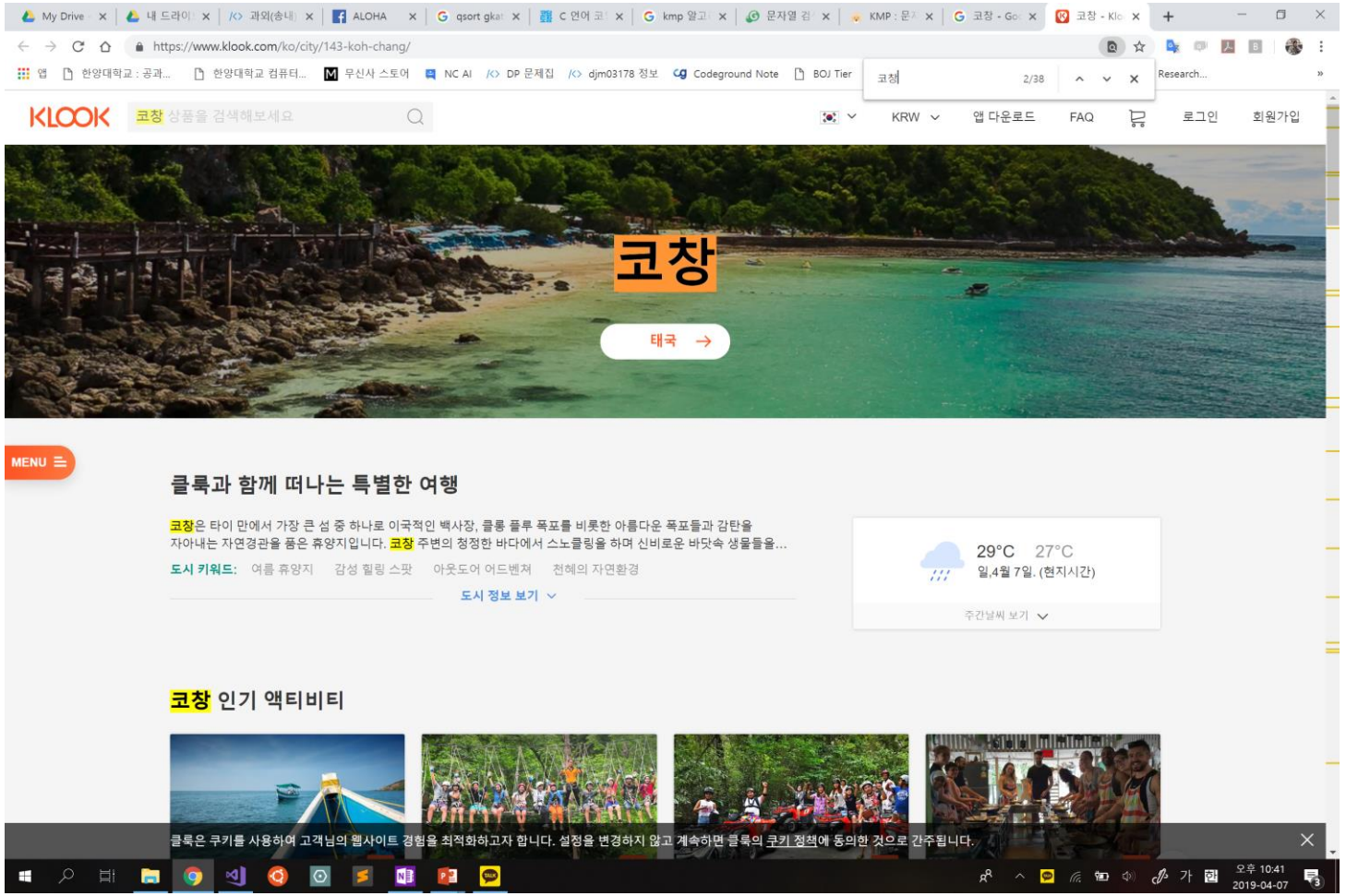
ALOHA
The algorithm club.

#1

문자열 검색



문자열 검색 문제


https://www.klook.com/ko/city/143-koh-chang/
코창
2/38
KLOOK
코창 상품을 검색해보세요
KRW
앱 다운로드
FAQ
로그인
회원가입
코창
태국
MENU
클룩과 함께 떠나는 특별한 여행
코창은 타이 만에서 가장 큰 섬 중 하나로 이국적인 백사장, 클룽 플루 폭포를 비롯한 아름다운 폭포들과 감탄을
자아내는 자연경관을 품은 휴양지입니다. 코창 주변의 청정한 바다에서 스노클링을 하며 신비로운 바닷속 생물들을...
도시 키워드: 여름 휴양지 감성 힐링 스팟 아웃도어 어드벤쳐 천혜의 자연환경
도시 정보 보기
29°C 27°C
일,4월 7일. (현지시간)
주간날씨 보기
코창 인기 액티비티
클룩은 쿠키를 사용하여 고객님의 웹사이트 경험을 최적화하고자 합니다. 설정을 변경하지 않고 계속하면 클룩의 쿠키 정책에 동의한 것으로 간주됩니다.
오후 10:41
2019-04-07

문자열 검색 문제

텍스트 내 특정 문자열(패턴)을 찾는 것

Brute Force

텍스트 “ABCABABCDE”에서 패턴 “ABC“가 몇 번 등장하는지 알아보자!

A	B	C	A	B	A	B	C	D	E
A	B	C							

Brute Force

텍스트 “ABCABABCDE”에서 패턴 “ABC”가 몇 번 등장하는지 알아보자!

A	B	C	A	B	A	B	C	D	E
A	B	C							

Brute Force

텍스트 “ABCABABCDE”에서 패턴 “ABC”가 몇 번 등장하는지 알아보자!

A	B	C	A	B	A	B	C	D	E
	A	B	C						

Brute Force

텍스트 “ABCABABCDE”에서 패턴 “ABC“가 몇 번 등장하는지 알아보자!

A	B	C	A	B	A	B	C	D	E
		A	B	C					

Brute Force

텍스트 “ABCABABCDE”에서 패턴 “ABC”가 몇 번 등장하는지 알아보자!

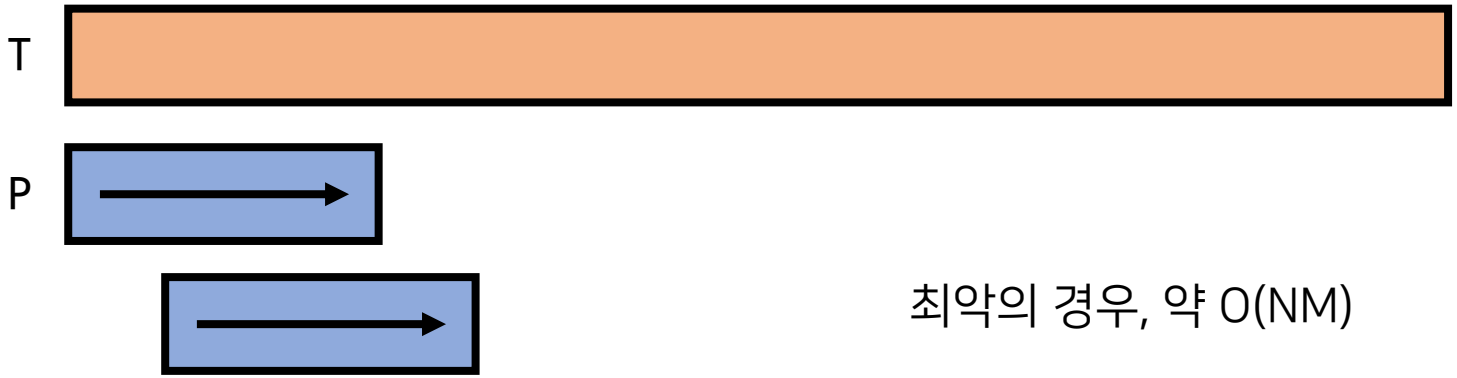
A	B	C	A	B	A	B	C	D	E
		A	B	C	…				

Brute Force

텍스트 T에서 패턴 P가 몇 번 등장하는지 알아보자!

Using Brute Force

-> T[i] 부터 시작하는 문자열이 P와 같은지 검사한다. 검사하다가 P랑 다르면 T[i + 1] 부터 다시 검사



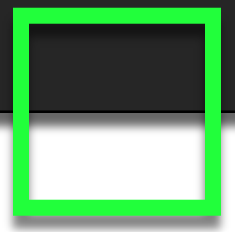
최악의 경우, 약 O(NM)

Brute Force

```
1  // Brute Force
2  string T, P;
3
4  void find_pattern()
5  {
6      int N = T.size(), M = P.size();
7      for(int i = 0; i <= N-M; i++){
8          bool fail = false;
9          for(int j = 0; j < M; j++){
10             if(T[i] != P[i+j]){
11                 fail = true;
12                 break;
13             }
14         }
15         if(!fail) printf("%d\n", i);
16     }
17 }
```

#2

KMP



KMP

Knuth-Morris-Pratt Algorithm

어떤 문자열이 다른 문자열의 부분 문자열인지를
판단하는 알고리즘

Donald Knuth, James H. Morris, Vaughan Pratt
세 사람이 공동으로 1977년에 발표

KMP

텍스트 "ABCDABCDABEE"에서 패턴 "ABCDABE"가 몇 번 등장하는지 알아보자!

idx	0	1	2	3	4	5	6	7	8	9	10	11
T	A	B	C	D	A	B	C	D	A	B	E	E
P	A	B	C	D	A	B	E					

KMP

텍스트 "ABCDABCDABEE"에서 패턴 "ABCDABE"가 몇 번 등장하는지 알아보자!

idx	0	1	2	3	4	5	6	7	8	9	10	11
T	A	B	C	D	A	B	C	D	A	B	E	E
P		A	B	C	D	A	B	E				

KMP

텍스트 "ABCDABCDABEE”에서 패턴 “ABCDABE“가 몇 번 등장하는지 알아보자!

idx	0	1	2	3	4	5	6	7	8	9	10	11
T	A	B	C	D	A	B	C	D	A	B	E	E
P	A	B	C	D	A	B	E					

KMP

텍스트 "ABCDABCDABEE"에서 패턴 "ABCDABE"가 몇 번 등장하는지 알아보자!

idx	0	1	2	3	4	5	6	7	8	9	10	11
T	A	B	C	D	A	B	C	D	A	B	E	E
P	A	B	C	D	A	B	E					

KMP

패턴 P에서 접두사(prefix)와 접미사(suffix)를
이용하여 중복된 계산을 줄여보자!

KMP

텍스트 "ABCDABCDABEE"에서 패턴 "ABCDABE"가 몇 번 등장하는지 알아보자!

A	B	C	D	A	B	E				
				A	B	C	D	A	B	E

KMP

텍스트 "ABCDABCDABEE"에서 패턴 "ABCDABE"가 몇 번 등장하는지 알아보자!

idx	0	1	2	3	4	5	6	7	8	9	10	11
T	A	B	C	D	A	B	C	D	A	B	E	E
P	A	B	C	D	A	B	E					

KMP

텍스트 "ABCDABCDABEE"에서 패턴 "ABCDABE"가 몇 번 등장하는지 알아보자!

JUMP

idx	0	1	2	3	4	5	6	7	8	9	10	11
T	A	B	C	D	A	B	[illegible]	[illegible]	A	B	E	E
P	A	B	C	D	A	B	E					

KMP

텍스트 "ABCDABCDABEE"에서 패턴 "ABCDABE"가 몇 번 등장하는지 알아보자!

idx	0	1	2	3	4	5	6	7	8	9	10	11
T	A	B	C	D	A	B	C	D	A	B	E	E
P					A	B	C	D	A	B	E	

KMP

텍스트 "ABCDABCDABEE"에서 패턴 "ABCDABE"가 몇 번 등장하는지 알아보자!

idx	0	1	2	3	4	5	6	7	8	9	10	11
T	A	B	C	D	A	B	C	D	A	B	E	E
P					A	B	C	D	A	B	E	

KMP

```cpp
// KMP
string T, P;
vector pi;

void kmp() {
    int begin = 0, matched = 0;

    while (begin + matched < n) {
        if (T[begin + matched] == P[matched]) {
            matched++;
        if (matched == m) ans.push_back(begin + 1);
    }
    else {
        if (matched == 0) begin++;
        else {
            begin += (matched - pi[matched - 1]);
             matched = pi[matched - 1];
        }
    }
}
```

$$O(N + M)$$

KMP

```
1  // KMP
2  string T, P;
3  vector pi;
4
5  void kmp() {
6      int begin = 0, matched = 0;
7
8      while (begin + matched < n) {
9          if (T[begin + matched] == P[matched]) {
10             matched++;
11         if (matched == m) ans.push_back(begin + 1);
12     }
13     else {
14         if (matched == 0) begin++;
15         else {
16             begin += (matched - pi[matched - 1]);
17              matched = pi[matched - 1];
18         }
19     }
20 }
```

KMP

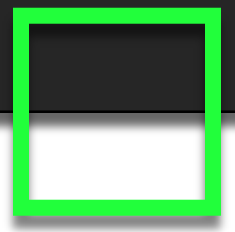
패턴 P에서 접두사(prefix)와 접미사(suffix)를
이용하여 중복된 계산을 줄여보자!

KMP

?

#3

Prefix & Suffix, pi Array



Prefix & Suffix

접두사(prefix), 접미사(suffix)

banana

Prefix & Suffix

접두사(prefix), 접미사(suffix)

banana 

b
ba
ban
bana
banan
banana

Prefix & Suffix

접두사(prefix), 접미사(suffix)

banana 

a
na
ana
nana
anana
banana

pi Array

pi[i] :

주어진 문자열의 0~i 까지의 부분 문자열 중에서
prefix == suffix가 될 수 있는 부분 문자열 중에서 가장 긴 것의 길이

pi Array

pi[i] :

주어진 문자열의 0~i 까지의 부분 문자열 중에서
prefix == suffix가 될 수 있는 부분 문자열 중에서 가장 긴 것의 길이

ABAABAB →

i	부분 문자열	pi[i]
0	A	0
1	AB	0
2	ABA	1
3	ABAA	1
4	ABAAB	2
5	ABAABA	3
6	ABAABAB	2

pi Array

```cpp
// pi Array
string P;
vector pi;

void getpi() {
    pi = vector<int>(m, 0);

    int begin = 1, matched = 0;

    while (begin + matched < m) {
        if (P[begin + matched] == P[matched]) {
            matched++;
             pi[begin + matched - 1] = matched;
        }
        else {
             if (matched == 0) begin++;
             else {
                 begin += (matched - pi[matched - 1]);
                 matched = pi[matched - 1];
             }
        }
    }
}
```

$O(M)$

과제

1786 : 찾기
1305 : 광고
1787 : 문자열의 주기 예측
1097 : 마법의 단어
10290 : 단어를 찾아서
4354 : 문자열 제곱

Reference

https://bowbowbow.tistory.com/6
https://mygumi.tistory.com/61

술!