문자열

NEXTERS

문자열 검색

주어진 문자열 H 에서 문자열 N을 부분 문자열로 포함하는지 찾는 문제

H = “Hogwarts”, N = “gwart” 라고 하면 부분 문자열이 되는 시작 위치는 2

N의 가능한 모든 시작 위치를 다 시도해 본다.

문자열 검색 브루트-포스

```
int BruteForceMatch(int T[], int n, int P[], int m) {
    for (int i = 0; i <= n - m; i++) {
        int j = 0;
        while (j < m && P[j] == T[i + j])
                j = j + 1;
        if (j == m)
                return i;
    }
return -1;
}
```

KMP 알고리즘

H 와 N이 주어졌을 때 미리 N의 패턴을 분석하여 다음으로 검색할 위치를 정한다.

예) N = “aabaabac” 일 경우 c에서 불일치가 발생하였을 때 H의 i+1번째 위치부터 재 검색 할 필요 없이 i + 3의 위치부터 재 검색 할 수 있다.

KMP 알고리즘

답이 될 수 있는 다음위치 검색을 위해 N의 접두사도 되고 접미사도 되는 문자열의 최대 길이를 계산해 둔다.

Pi[i] = N[...i] 의 접두사도 되고 접미사도 되는 문자열의 최대 길이

H와 N의 글자를 비교하면서 matched글자가 일치한 후 불일치가 발생했다고 할 때 다음으로 pi[matched-1] 부터 시도 한다.

KMP 알고리즘

"aabaabac" 일 때 부분 일치 테이블 계산

i	N[…i]	접두사 접미사 공통 최대 문자열	Pi[i]
0	a	(없음)	0
1	aa	a	1
2	aab	(없음)	0
3	aaba	a	1
4	aabaa	aa	2
5	aabaab	aab	3
6	aabaaba	aaba	4
7	aabaabac	(없음)	0

부분 문자열 검색 구현

```
vector<int> getPartialMatch(const string& N) {
int m = N.size(); vector<int> pi(m, 0); int begin = 1, matched = 0;
while (begin + matched < m) {
    if (N[begin + matched] == N[matched]) {
        ++matched;
        pi[begin + matched - 1] = matched;
    }
    else {
        if (matched == 0) ++begin;
        else {begin += matched - pi[matched - 1]; matched = pi[matched - 1];}
    }
}
return pi;
}
```

KMP 검색 구현

```
vector<int> kmpSearch(const string& H, const string& N) {
int n = H.size(), m = N.size(); vector<int> ret;
vector<int> pi = getPartialMatch(N); int begin = 0, matched = 0;
while (begin <= n - m) {
            if (matched < m && H[begin + matched] == N[matched])  {
              ++matched; if (matched == m) ret.push_back(begin);
            }
            else {
                        if (matched == 0) {++begin;}
                        else {begin += matched - pi[matched - 1];matched = pi[matched - 1];}
            }
}
return ret;
}
```

접미사 배열

어떤 문자열 S의 모든 접미사를 사전순으로 정렬 해 둔 것

일반적인 정렬 알고리즘을 사용하여 접미사 배열을 만들 때 n^2lgn의 시간 복잡도가 걸림

특정 문자열에 대하여 많은 시간이 걸리기 때문에 맨버-마이어스 알고리즘 사용

맨버 – 마이어스 알고리즘

접미사들의 목록을 정렬 할 때 매번 그 기준을 바꾸어 정렬한다.

처음에는 접미사의 첫 한 글자 만을 기준으로 정렬하고 그 다음에는 접미사의 첫 두 글자를 기준으로, 그 다음에는 접미사의 첫 네 글자를 기준으로 정렬한다.

위와 같은 방법으로 lgn 번의 정렬을 하고 나면 접미사 배열을 얻을 수 있음

문제: Jaeha’s safe

문자열을 시계방향 또는 시계 반대 방향으로 돌렸다고 가정했을 때 몇 번을 돌려야 현재 상태가 나오는지 풀어내는 문제

오리지널 문자열을 두 배로 만들어서 부분 문자열을 찾아 낸다.

✓문제

https://algospot.com/judge/problem/read/JAEHASAFE

✓풀이코드

https://github.com/Nexters/algorithmStudy/blob/master/seokjoong/Chapter20/Jaehasafe.cpp

문제: 말버릇

대본 중 k 번 이상 나오는 부분 문자열중 가장 긴 문자열을 찾아내는 문제

인접한 모든 접미사의 쌍에 대하여 최장 공통 접두사를 계산한다.

✓문제

https://algospot.com/judge/problem/read/HABIT

✓풀이코드

https://github.com/Nexters/algorithmStudy/blob/master/seokjoong/Chapter20/Habit.cpp