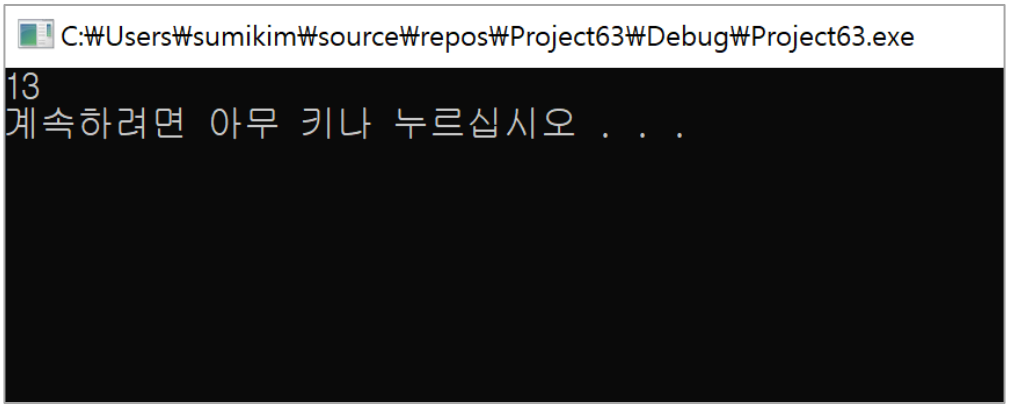
오픈랩 과제 #3 : KMP 알고리즘

2020. 10. 6 (화) / 20181202 김수미

1. 코드 실행 결과



2. 코드 및 알고리즘 설명

KMP 알고리즘은 두개의 스트링(대상스트링과 패턴스트링)을 입력 받아 pattern matching을 하는 알고리즘이다. 이를 통해 대상스트링에 패턴스트링이 포함 되어 있는지의 여부를 확인할 수 있다.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
 #define max_string_size 100
 #define max_pat_size 100

 int failure[max_pat_size];
 char string[max_string_size];
 char pat[max_pat_size];

 //KMP(Knuth, Morris, Pratt) 패턴 매칭 알고리즘
int pmatch(char *string, char *pat) {
    int i = 0, j = 0;
    int lens = strlen(string);
    int lenp = strlen(pat);
    while (i < lens && j < lenp) {
        if (string[i] == pat[j]) {
            i++;
            j++;
        }
        else if (j == 0) i++;
        else j = failure[j - 1] + 1;
    }
    return ((j == lenp) ? (i - lenp) : -1);
}
```

먼저 해당 코드 작성에 필요한 헤더파일들을 불러와 준다.

그리고 max_string_size와 max_pettern_size를 100으로 지정해 대상스트링과 패턴스트링의 길이가 100을

넘지 않도록 해준다. 정의할 함수 pmatch 와 fail 을 미리 선언해주고, 세가지 배열 failure, string, pat 의 자료형과 크기를 max_string_size와 max_pettern_size를 이용해 지정해준다.

첫번째로 함수 pmatch를 정의해준다. 이 함수는 패턴스트링과 대상스트링을 본격적으로 비교하는 역할을 담당한다. 두 스트링을 비교했을 때 일치하는 문자열이 있다면 'starting position of pat in string'을 반환해주고, 없다면 -1을 반환한다.

패턴스트링과 대상스트링의 문자형 변수를 하나씩 비교하면서, 두 문자가 동일하다면 패턴스트링과 대상스트링의 비교 문자가 모두 다음 대상으로 넘어가고, 대상스트링의 문자가 패턴스트링의 문자와 동일하지 않다면, 패턴스트링의 가장 첫번째 문자를 비교하고 있던 경우 : 패턴스트링의 문자는 그대로 두고 대상스트링의 비교 문자만 다음 대상으로 넘어가며, 패턴스트링의 가장 첫번째 문자를 비교하고 있던 것이 아닌 경우 : 비교대상을 pat[j]에서 j=failure[j-1]+1 로 넘어가준다.

```
27  //패턴의 실패 함수 계산
28  void fail(char*pat) {
29      int i, n = strlen(pat);
30      failure[0] = -1;
31
32      for (int j = 1; j < n; j++) {
33          i = failure[j - 1];
34          while ((pat[j] != pat[i + 1] && (i >= 0))) i = failure[i];
35          if (pat[j] == pat[i + 1]) failure[j] = i + 1;
36          else failure[j] = -1;
37      }
38  }
```

그 다음 fail 함수를 정의해준다. 이 함수는 패턴의 실패함수를 계산하기 위한 함수로, 실패함수란 패턴 p가 $p_0p_1\cdots p_{n-1}$일 때 'largest $i < j$ such that $p_0p_1\cdots p_i = p_{j-i}p_{j-i+2}\cdots p_j$ if such an $i \geq 0$ exists' 또는 -1 의 값을 가진다. 예를 들어 패턴스트링이 "abcabcacab"일 때, failure 배열의 값은 아래와 같다. (j=index)

j	0	1	2	3	4	5	6	7	8	9
pat	a	b	c	a	b	c	a	c	a	b
f	−1	−1	−1	0	1	2	3	−1	0	1

하지만 해당 코드에서 pat의 입력값은 'elli'였기 때문에, 해당 코드에서는 패턴스트링의 입력값을 다르게 하지 않는 이상 이 실패함수값이 크게 중요하게 작용하지는 않을 것이다.

```
40  int main() {
41      //kmp.txt 파일의 내용을 string, pat 배열에 입력받는다
42      FILE* fp;
43      fp = fopen("kmp.txt", "r");
44      if (fp == NULL) {
45          fprintf(stderr, "file open error ₩n");
46          exit(1); }
47      while (!feof(fp)) {
48          fscanf(fp, "%s", &string);
49          fscanf(fp, "%s", &pat); }
50      fclose(fp);
51
52      //KMP 알고리즘 실행 및 결과 출력
53      fail(pat);
54      printf("%d₩n", pmatch(string, pat));
55
56      system("pause");
57      return 0;
58  }
```

이제 메인함수에서 새롭게 정의한 함수 fail 과 pmatch 를 호출해 결과값을 계산할 차례이다. 먼저 kmp.txt 파일을 입력 받아 string과 pat에 저장해 대상스트링과 패턴스트링의 값을 입력해준다.

그리고 입력 받은 두 스트링을 함수 fail 과 pmatch 의 매개변수 값으로 넣어준다. 그 다음 입력 받은 값으로 실행된 함수 pmatch 의 반환값을 출력해주면 pattern matching 결과를 얻을 수 있다.

해당 코드의 실행결과, 13이라는 값이 출력되었으므로 “ArtificialIntelligence” 라는 대상스트링에서 “elli” 라는 패턴은 대상스트링의 index번호 13번을 시작으로 하는 자리에 포함되어 있음을 알 수 있다.