컴퓨터알고리즘과실습 실습8

2020112046 이재원

8-1

```
#include <string.h>
#include <iostream>
using namespace std;
int BFSearchCnt = 0;
int KMPSearchCnt = 0;
int RKSearchCnt = 0;
// String search algorithm - Brute Force
int bruteSearch(char *p, char *a) {
   int i, j;
   int M = strlen(p);
   int N = strlen(a);
   for (i = 0, j = 0; i < N && j < M; i++, j++) {
       BFSearchCnt++;
       if (a[i] != p[j]) {
           i -= j;  // reset i
           j = -1;  // reset j (pattern index = 0)
       }
   }
   if (j == M)  // pattern found
       return i - M;
   else  // pattern not found
       return i;
}
// String search algorithm - KMP
int kmpSearch(char *p, char *a) {
   int i = 0, j = 0;
   int N = strlen(a);
   int M = strlen(p);
   // get pi table
```

```
    int pi[M];
    for (i = 0, j = -1; i < M; i++, j++) {
        pi[i] = j;
        while (j >= 0 && p[i] != p[j]) {
            j = pi[j];
        }
    }
    cout << "pi table: ";
    for (i = 0; i < M; i++) {
        cout << pi[i] << " ";
    }
    cout << endl;
    // search
    for (i = 0, j = 0; i < N && j < M; i++, j++, ++KMPSearchCnt) {
        while (j >= 0 && a[i] != p[j]) {
            j = pi[j];
            KMPSearchCnt;
        }
    }
    if (j == M)  // pattern found
        return i - M;
    else  // pattern not found
        return i;
}
const int q = 33554393;
const int d = 32;
// String search algorithm - Rabin-Karp
int rabinKarpSearch(char *p, char *a) {
    int i;
    int dM = 1, h1 = 0, h2 = 0;
    int N = strlen(a);
    int M = strlen(p);
    for (i = 0; i < M - 1; i++) {
        dM = (dM * d) % q;
    }
    for (i = 0; i < M; i++) {
```

```
        h1 = (d * h1 + p[i]) % q;
        h2 = (d * h2 + a[i]) % q;
    }
    for (i = 0; h1 != h2; i++, ++RKSearchCnt) {
        h2 = (h2 + d * q - a[i] * dM) % q;
        h2 = (h2 * d + a[i + M]) % q;
        if (i > N - M)
            return N;
    }
    return i;
}
int main() {
    char a[] = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";
    char b[] = "AAAAAAAAAAAAAAB";
    // Brute Force
    int pos = bruteSearch(b, a);
    cout << "Brute Force: " << pos << endl;
    // KMP
    pos = kmpSearch(b, a);
    cout << "KMP: " << pos << endl;
    // Rabin-Karp
    pos = rabinKarpSearch(b, a);
    cout << "Rabin-Karp: " << pos << endl;
    // Search Count of each algorithm
    cout << "BFSearchCnt: " << BFSearchCnt << endl;
    cout << "KMPSearchCnt: " << KMPSearchCnt << endl;
    cout << "RKSearchCnt: " << RKSearchCnt << endl;
}
```

실습 결과

```
jw101@DESKTOP-OD6E1AA MINGW64 ~/Desktop/Code/C++
$  /usr/bin/env c:\\Users\\jw101\\.vscode\\extensions\\ms-vs
-MIEngine-In-spgk3gu4.vqs --stdout=Microsoft-MIEngine-Out-w2
p.foj --dbgExe=C:\\mingw64\\bin\\gdb.exe --interpreter=mi
Brute Force: 59
pi table: -1 0 1 2 3 4 5 6 7 8 9 10 11 12 13
KMP: 59
Rabin-Karp: 59
BFSearchCnt: 689
KMPSearchCnt: 59
RKSearchCnt: 45
```

세 알고리즘 모두 정확히 위치를 찾았고,

비교 회수는 라빈카프-KMP-Brute Force 순으로 많은 것을 볼 수 있었다.