**프로그래밍 실습 #8**

2023년 11월 3주차

□ 다음 스트링 탐색 알고리즘을 파이썬으로 구현해 보라.

(1) 직선적 알고리즘

--------------------------------------------------------------------

bruteForce(p[], t[])

M ← 패턴의 길이; N ← 텍스트의 길이;

for (i ← 0, j ← 0; j < M and i < N; i ← i + 1, j ← j + 1) do

if (t[i] ≠ p[j]) then {

i ← i - j;

j ← -1;

}

if (j = M) then return i - M;

else return i;

end bruteForce()

--------------------------------------------------------------------

- 텍스트 ababababcababababcaabbabababca에 대하여 패턴 abababca를 탐색한다고 할 때, 직선적 알고리즘을 파이썬으로 구현하라.

- 실행 결과로 패턴이 나타난 위치를 출력한다.

- 마지막 패턴을 인식하지 못하면 텍스트의 마지막에 널 문자('\0')를 추가해 보라.

- 텍스트와 패턴에 있는 문자에 대한 총 비교횟수를 구하라.

[프로그램 소스 코드]

--------------------------------------------------------------------

def bruteForce(p, t, k):

pass

text = 'ababababcababababcaabbabababca' + '\0'

pattern = 'abababca'

M = len(pattern)

N = len(text)

K = 0

while True:

pos = bruteForce(pattern, text, K)

K = pos + M

if K < N: print('패턴이 나타난 위치 : ', pos)

else: break

print('스트링 탐색 종료')

--------------------------------------------------------------------

[실행 결과]

--------------------------------------------------------------------

패턴이 나타난 위치 : 2

패턴이 나타난 위치 : 11

패턴이 나타난 위치 : 22

스트링 탐색 종료

--------------------------------------------------------------------

(2) 재시작 위치를 구하는 알고리즘

--------------------------------------------------------------------

initNext(p[])

M ← 패턴의 길이;

next[0] ← -1;

for (i ← 1, j ← 0; i < M; i ← i + 1, j ← j + 1) do {

next[i] ← j;

while (j ≥ 0 and p[i] ≠ p[j]) do

j ← next[j];

}

end initNext()

--------------------------------------------------------------------

- initNext() 함수를 파이썬으로 구현하여 다음 문자열에 대한 재시작 위치를 구하라.

(1) aaaaaaaaa

(2) 00000001

(3) 10100111

(4) ababca

(5) abababca

(6) abcabcabc

(7) abcabcacab

(8) abracadabra

(3) initNext() 함수 개선

* 개선된 재시작 위치를 구하도록 initNext() 함수를 수정하라.

--------------------------------------------------------------------

initNext(p[])

M ← 패턴의 길이;

next[0] ← -1;

for (i ← 1, j ← 0; i < M; i ← i + 1, j ← j + 1) do {

if (p[i] = p[j]) then next[i] ← next[j];

else next[i] ← j;

while (j ≥ 0 and p[i] ≠ p[j]) do

j ← next[j];

}

end initNext()

--------------------------------------------------------------------

- 수정된 initNext() 함수를 사용하여 다음 문자열에 대한 재시작 위치를 구하라.

(1) aaaaaaaaa

(2) 00000001

(3) 10100111

(4) ababca

(5) abababca

(6) abcabcabc

(7) abcabcacab

(8) abracadabra

(4) KMP 알고리즘

--------------------------------------------------------------------

KMP(p[], t[])

M ← 패턴의 길이; N ← 텍스트의 길이;

initNext(p);

for (i ← 0, j ← 0; j < M and i < N; i ← i + 1, j ← j + 1) do

while ((j ≥ 0) and (t[i] ≠ p[j])) do

j ← next[j];

if (j = M) then return i - M;

else return i;

end KMP()

--------------------------------------------------------------------

- 텍스트 ababababcababababcaabbabababca에 대하여 패턴 abababca를 탐색한다고 할 때, KMP 알고리즘을 파이썬으로 구현하라.

- 실행 결과로 패턴이 나타난 위치를 출력한다.

- 마지막 패턴을 인식하지 못하면 텍스트의 마지막에 널 문자('\0')를 추가해 보라.

- 텍스트와 패턴에 있는 문자에 대한 총 비교횟수를 구하라.

[프로그램 소스 코드]

--------------------------------------------------------------------

def initNext(p):

pass

def KMP(p, t, k):

pass

next = [0] \* 50

text = 'ababababcababababcaabbabababca' + '\0'

pattern = 'abababca'

M = len(pattern)

N = len(text)

K = 0

while True:

pos = KMP(pattern, text, K)

K = pos + 1

if K <= N - M: print('패턴이 나타난 위치 : ', pos)

else: break

print('스트링 탐색 종료')

--------------------------------------------------------------------

(5) 보이어-무어 알고리즘

--------------------------------------------------------------------

BM(p[], t[])

M ← 패턴의 길이; N ← 텍스트의 길이 - 1;

initSkip(p);

for (i ← M-1, j ← M-1; j ≥ 0; i ← i - 1, j ← j - 1) do

while (t[i] ≠ p[j]) do {

if (i ≥ N) then return N;

s ← skip[index(t[i])];

if (M-j > s) then i ← i + M - j;

else i ← i + s;

j ← M - 1;

}

return i+1;

end BM()

--------------------------------------------------------------------

- 텍스트 STRING STARTING CONSISTING에 대하여 패턴 STING을 탐색한다고 할 때, 보이어-무어 알고리즘을 파이썬으로 구현하라.

- 실행 결과로 패턴이 나타난 위치를 출력한다.

- 텍스트와 패턴에 있는 문자에 대한 총 비교횟수를 구하라.

- initSkip() 함수는 패턴에 대한 skip 배열을 만들어주는 함수이다.

- index() 함수는 문자가 띄어쓰기(space)이면 0을 반환하고, 나머지 경우에는 영문 대문자의 i번째 문자에 대해 정수 i를 반환하는 함수이다. 예를 들어, index(A)를 호출하면 1을 반환하고, index(E)는 5를 반환한다.

- initSkip() 함수에서 NUM은 띄어쓰기와 영문대문자 개수를 더한 값인 27을 가지고 있다.

--------------------------------------------------------------------

def index(c):

if ord(c) == 32: return 0

else: return ord(c) - 64

def initSkip(p):

M = len(p)

for i in range(NUM):

skip[i] = M

for i in range(M):

skip[index(p[i])] = M - i – 1

--------------------------------------------------------------------

[프로그램 소스 코드]

--------------------------------------------------------------------

def index(c):

if ord(c) == 32: return 0

else: return ord(c)-64

def initSkip(p):

M = len(p)

for i in range(NUM):

skip[i] = M

for i in range(M):

skip[index(p[i])] = M - i - 1

def BM(p, t, k):

pass

NUM = 27

skip = [0] \* NUM

text = 'VISION QUESTION ONION CAPTION GRADUATION EDUCATION' + '\0'

pattern = 'ATION'

M = len(pattern)

N = len(text)

K = 0

while True:

pos = BM(pattern, text, K)

K = pos + 1

if K <= N - M: print('패턴이 나타난 위치 : ', pos)

else: break

print('스트링 탐색 종료')

--------------------------------------------------------------------

(6) 라빈-카프 알고리즘

□ 첫 번째 패턴을 찾는 경우

--------------------------------------------------------------------

RK(p[], t[])

dM ← 1; h1 ← 0; h2 ← 0;

M ← 패턴의 길이; N ← 텍스트의 길이;

for (i ← 1; i < M; i ← i + 1) do

dM ← (d\*dM) mod q;

for (i ← 0; i < M; i ← i + 1) do {

h1 ← (h1 × d + index(p[i])) mod q;

h2 ← (h2 × d + index(t[i])) mod q;

}

for (i ← 0; h1 ≠ h2; i ← i + 1) do {

if (i + M ≥ N) then return N;

h2 ← (h2 + d × q - index(t[i]) × dM) mod q;

h2 ← (h2 × d + index(t[i+M])) mod q;

if (i > N-M) then return N;

}

return i;

end RK()

--------------------------------------------------------------------

□ n번째 패턴을 찾는 경우

--------------------------------------------------------------------

RK(p[], t[], k)

dM ← 1; h1 ← 0; h2 ← 0;

M ← 패턴의 길이; N ← 텍스트의 길이;

for (i ← 1; i < M; i ← i + 1) do

dM ← (d × dM) mod q;

for (i ← 0; i < M; i ← i + 1) do

h1 ← (h1 × d + index(p[i])) mod q;

for (i ← k, j ← 0; i < N and j < M; i ← i + 1, j ← j + 1) do

h2 ← (h2 × d + index(t[i])) mod q;

for (i ← k; h1 ≠ h2; i ← i + 1) do {

if (i + M ≥ N) then return N;

h2 ← (h2 + d × q - index(t[i]) × dM) mod q;

h2 ← (h2 × d + index(t[i+M])) mod q;

if (i > N-M) then return N;

}

return i;

end RK()

--------------------------------------------------------------------

[프로그램 소스 코드]

--------------------------------------------------------------------

def index(c):

if ord(c) == 32: return 0

else: return ord(c)-64

def RK(p, t, k):

pass

q = 33554393

d = 32

text = 'VISION QUESTION ONION CAPTION GRADUATION EDUCATION' + '\0'

pattern = 'ATION'

M = len(pattern)

N = len(text)

K = 0

while True:

pos = RK(pattern, text, K)

K = pos + 1

if K <= N - M: print('패턴이 나타난 위치 : ', pos)

else: break

print('스트링 탐색 종료')

--------------------------------------------------------------------