ALGORITHM
알고리즘

Lecture 13

스트링 알고리즘 (2)

컴퓨터과학과 | 김진욱 교수

한국방송통신대학교

학습목차

01. 보이어-무어 알고리즘

보이어-무어 알고리즘

Boyer-Moore 알고리즘

- 패턴 내의 문자들의 관계를 이용하여 매칭 시 중복된 비교를 줄임
- 텍스트의 첫 위치에서 패턴의 뒷부분부터 문자 비교
- 불일치 또는 매치 발생 → 패턴 이동
  - ✓ 불일치 문자 방법, 일치 접미부 방법 중 더 많이 이동시킬 수 있는 값 선택

위치 0 1 2 3 4 5 6 7 8 9 …

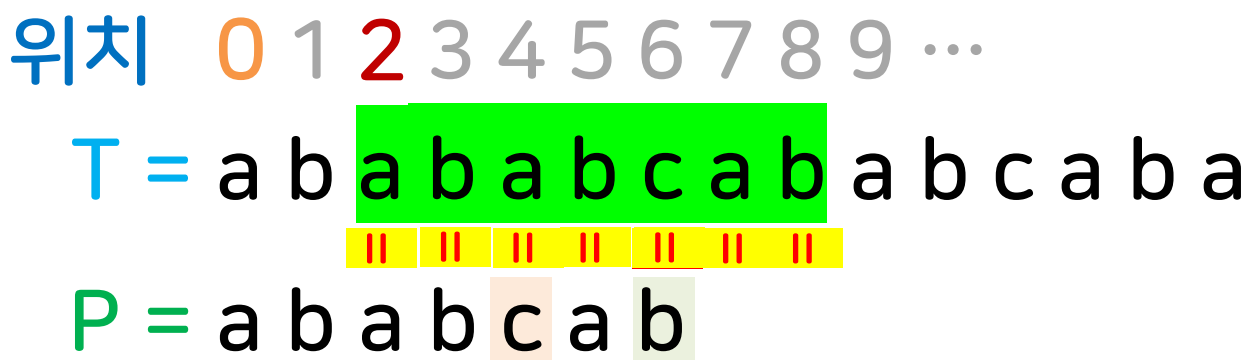
T = a b a b a b c a b a b c a b a

P = a b a b c a b

보이어-무어 알고리즘

불일치 문자 bad character 방법

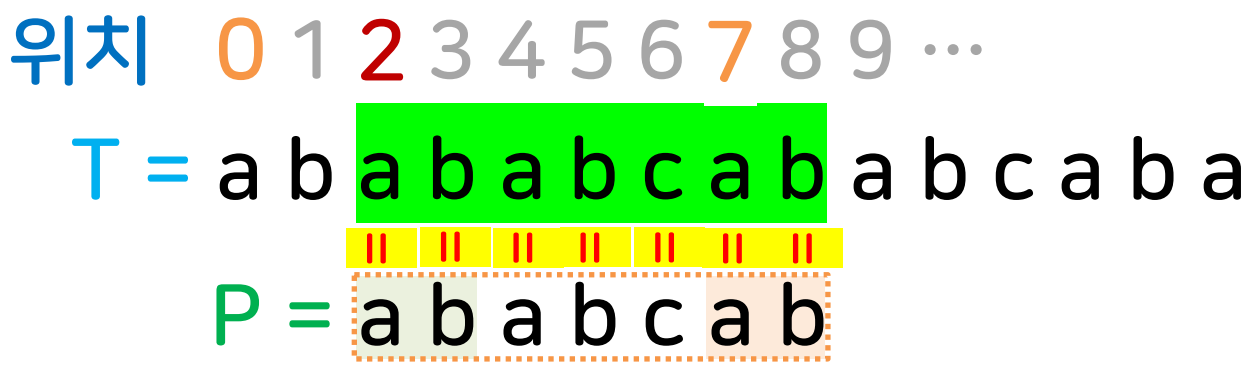
- 불일치가 발생한 텍스트의 문자가 패턴에서 가장 마지막에 나타나는 위치가 불일치가 발생한 곳으로 오도록 패턴을 이동



보이어-무어 알고리즘

일치 접미부 good suffix 방법

- 일치한 서브스트링(접미부)에 대한 접두부와 접미부의 최대 일치 정보 활용
- KMP 알고리즘과 유사하면서도 다름



보이어-무어 알고리즘_불일치 문자


```
BadChar (m, P[ ], Σ)
{
  for (알파벳의 각 문자 c)
     δ1[c] = -1;          // -1로 배열 초기화(패턴에 없는 경우 대비)
  for (i=0; i < m; i++)
     δ1[P[i]] = i;        // 패턴에서 가장 마지막에 나타나는 위치만 남김
  return (δ1);
}
```

Σ={a, b, c, d}

	0	1	2	3	4	5	6
P =	a	b	a	b	c	a	b

	a	b	c	d
δ_1	5	6	4	-1

보이어-무어 알고리즘_일치 접미부


```
GoodSuf (m, P[ ])
{
  revP = P를 뒤집은 스트링;
  revF[0..m-1] = PreKMP (m, revP); // 뒤집은 패턴의 최대 일치 정보 찾기
  for (i=-1; I < m; i++)
     δ2[i] = m-1 – revF[m-1];          // 전체 일치인 경우의 이동값으로 배열 초기화
  for (k=m-1; k >= 0; k--)
     δ2[m-1-revF[k]-1] = k – revF[k];
  return (δ2);
}
```

	-1	0	1	2	3	4	5	6
P =		a	b	a	b	c	a	b
δ_2	5	5	5	5	5	3	3	1

	0	1	2	3	4	5	6
revP =	b	a	c	b	a	b	a
revF	-1	-1	-1	0	1	0	1

보이어-무어 알고리즘


```
BM (n, T[ ], m, P[ ], Σ)
{
   δ1 = BadChar (m, P, Σ);
   δ2 = GoodSuf (m, P);
  i = m-1;
  while (i < n) {
     j = m-1;                  // 패턴의 마지막 문자부터 시작
     while (j >= 0 && P[j] == T[i]) { i--; j--; }
     if (j == -1) {
        위치 i+1 출력;         // 매치 발견
        i += δ2[-1] + m;       // 전체 일치인 경우로 이동
     } else                    // 불일치 문자 정보와 일치 접미부 정보 중
        i += max(j - δ1[T[i]], δ2[j]) + m-1-j;     // 큰 값만큼 이동
  }
}
```

보이어-무어 알고리즘_예


```
                                  i
위치   0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
T =    a b a b a b c a b a b c a b a
       || || || || || || || || || || || ||
P =    a b a b c a b
       0 1 2 3 4 5 6
                   j
```

```
i = m-1;
while (i < n) {
    j = m-1;
    while (j >= 0 && P[j] == T[i]) { i--; j--; }
    if (j == -1) {
        위치 i+1 출력;
        i += δ2[-1] + m;
    } else
        i += max(j - δ1[T[i]], δ2[j]) + m-1-j;
}
```

	a	b	c	d
δ_1	5	6	4	-1

	-1	0	1	2	3	4	5	6
δ_2	5	5	5	5	5	3	3	1

성능과 특징

전처리 → O(m)

- 불일치 문자 방법 → O(m+|Σ|)
  - ✓ 알파벳 크기(|Σ|)와 패턴 길이(m)만큼 반복
- 일치 접미부 방법
  - ✓ KMP 알고리즘의 전처리 → O(m)
  - ✓ 패턴의 길이(m)만큼 두 번 반복

```
for (알파벳의 각 문자 c)
    δ1[c] = -1;
for (i=0; i < m; i++)
    δ1[P[i]] = i;
```

```
revF[0..m-1] = PreKMP (m, revP);
for (i=-1; I < m; i++)
    δ2[i] = m-1 – revF[m-1];
for (k=m-1; k >= 0; k--)
    δ2[m-1-revF[k]-1] = k – revF[k];
```

성능과 특징

최악의 시간 복잡도 → O(nm)

```
while (i < n) {                          O(n)
    j = m-1;                                  O(m)
    while (j >= 0 && P[j] == T[i]) { i--; j--; }
    if (j == -1) {
        위치 i+1 출력;
        i += δ2[-1] + m;                  1 증가
    }
    else
        i += max(j - δ1[T[i]], δ2[j]) + m-1-j;
}
```

T = aaaaaaa…aaa

P = aaaaa

성능과 특징

최선의 시간 복잡도 → O(n/m)

```
while (i < n) {
    j = m-1;
    while (j >= 0 && P[j] == T[i]) { i--; j--; }
    if (j == -1) {
        위치 i+1 출력;
        i += δ2[-1] + m;
    }
    else
        i += max(j - δ1[T[i]], δ2[j]) + m-1-j;
}
```

O(1)

m 증가

T = aaaaaaa…aaa

P = bbbbb

02. 데이터 압축 기본 개념

데이터 압축 data compression?

주어진 데이터를 보다 적은 공간을 사용하여 표현하는 것

- 데이터 → 1차원(스트링), 2차원(이미지), 3차원(동영상) 등

```
uteo:~/gzip Gnugi$
uteo:~/gzip Gnugi$ ls -l

ugi  Gnugi  2609111 Aug 14 16:03 above_cys1_Se1,20000.txt
uteo:~/gzip Gnugi$
uteo:~/gzip Gnugi$
uteo:~/gzip Gnugi$
uteo:~/gzip Gnugi$ gzip above_cys1_Se1,20000.txt
uteo:~/gzip Gnugi$
uteo:~/gzip Gnugi$
uteo:~/gzip Gnugi$
uteo:~/gzip Gnugi$ ls -l

ugi  Gnugi  778760 Aug 14 16:03 above_cys1_Se1,20000.txt.gz
uteo:~/gzip Gnugi$
```

데이터 압축 구분

무손실 압축

- 압축된 데이터로부터 원래의 데이터를 완벽하게 복원할 수 있는 압축 방법
- 데이터의 내용 하나하나가 모두 중요한 경우에 사용
- 종류 → RLE, 허프만 코딩, LZ77 등

손실 압축

- 압축된 데이터로부터 원래의 데이터를 완벽하게 복원할 수 없는 압축 방법
- 데이터의 내용이 약간 변형되어도 무방한 경우에 사용
- 종류 → JPEG 표준, MPEG 표준 등

데이터 압축 기본 용어

인코딩 encoding

- 원래의 데이터를 **압축된 데이터로 변환**하는 것

디코딩 decoding

- 압축된 데이터를 **압축되지 않은 데이터로 변환**하는 것
- 손실 압축의 경우 디코딩 결과가 인코딩 이전 데이터와 동일하지 않을 수 있음

03. RLE

RLE?

Run Length Encoding

- 스트링에서 동일 문자가 연속해서 나타나는 것을
  그 문자와 반복 횟수로 압축하는 방법

bbbbb

(b,5)

RLE_인코딩

주어진 스트링을 차례로 보며 문자가 달라질 때까지 횟수 셈

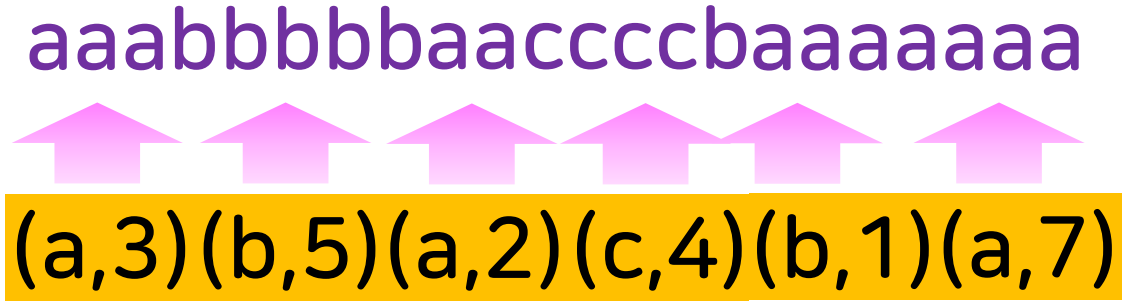
```
RLE_enc (n, S[])
{
  int idx=0;
  for (i=0; i < n; i++) {
     count = 1;
     while (i+1 < n && S[i] == S[i+1]) {
        count++;  i++;
     }
     C[idx++] = (S[i], count);
  }
  return (C[0..idx-1]);
}
```

aaabbbbbaaccccbaaaaaaa

(a,3)(b,5)(a,2)(c,4)(b,1)(a,7)

RLE_디코딩

압축된 데이터를 차례로 보며 각 문자를 횟수만큼 반복




```
RLE_dec (m, C[])
{
  int i=0;
  for (idx=0; idx < m; idx++)
     for (j=0; j < C[idx].count; j++, i++)
        S[i] = C[idx].ch;
  return (S[0..i-1]);
}
```

성능과 특징

성능 → O(n)

- 인코딩과 디코딩 모두 이중 루프로 보이지만
  실제로는 스트링의 길이(n)만큼 수행

```
for (i=0; i < n; i++) {
    count = 1;
    while (i+1 < n && S[i] == S[i+1]) {
        count++;  i++;
    }
    C[idx++] = (S[i], count);
}
```

i 증가

i 증가

```
for (idx=0; idx < m; idx++)
    for (j=0; j < C[idx].count; j++, i++)
        S[i] = C[idx].ch;
```

i 증가

정리하기

1. 보이어-무어 알고리즘

- 패턴 내의 문자들의 관계를 이용하여 매칭 시 중복된 비교를 줄임
- 텍스트의 첫 위치에서 패턴의 뒷부분부터 문자 비교
- 불일치 문자 방법과 일치 접미부 방법 이용
- 전처리 O(m), 매칭-최악 (nm), 최선 (n/m)

2. 데이터 압축 기본 개념

- 주어진 데이터를 보다 적은 공간을 사용하여 표현하는 것
- 무손실 압축, 손실 압축
- 인코딩, 디코딩

3. RLE

- 스트링에서 동일 문자가 연속해서 나타나는 것을 그 문자와 반복 횟수로 압축
- 성능 O(n)

ALGORITHM
알고리즘

다음시간에는

Lecture 14

스트링 알고리즘 (3)

컴퓨터과학과 | 김진욱 교수

한국방송통신대학교