Algoritma Pencarian *String* (*String Matching*)

Bahan Kuliah IF2251 Strategi Algoritmik

Oleh: Rinaldi Munir

Persoalan pencarian *string*

Diberikan:

1. teks (*text*), yaitu (*long*) *string* yang panjangnya *n* karakter
2. *pattern*, yaitu *string* dengan panjang *m* karakter (*m* < *n*) yang akan dicari di dalam teks.

Carilah (*find* atau *locate*) lokasi pertama di dalam teks yang bersesuaian dengan *pattern*.

Contoh 1:

- *Pattern*: hari
- Teks: kami pulang **hari** kamis
  ⇑ target

Contoh 2:

- *Pattern*: not
- Teks: nobody **not**iced him
  ⇑ target

Contoh 3:

- *Pattern*: apa
- Teks: Si**apa** yang menjemput P**apa**
  dari kota Balikp**apa**n?

Algoritma *Brute Force*

Contoh 4:

Teks: nobody noticed him

Pattern: not

```
      nobody noticed him
s=0   not
s=1    not
s=2     not
s=3      not
s=4       not
s=5        not
s=6         not
s=7          not
```

Contoh 10.4:

Teks: 1001010**001011**10101010001

Pattern: 001011

```
    1001010001011110101010001
s=0 001011
s=1  001011
s=2   001011
s=3    001011
s=4     001011
s=5      001011
s=6       001011
s=7        001011
s=8         001011
```

Kompleksitas algoritma *brute-force*:

- Kompleksitas kasus terbaik adalah $O(n)$.
- Kasus terbaik terjadi jika yaitu bila karakter pertama *pattern P* tidak pernah sama dengan karakter teks *T* yang dicocokkan
- Pada kasus ini, jumlah perbandingan yang dilakukan paling banyak *n* kali misalnya:

- Teks: `String ini berakhir dengan zz`
- *Pattern*: `zz`

- Kasus terburuk: $m(n - m + 1) = O(mn)$

- Teks: aaaaaaaaaaaaaaaaaaaaaaaaaaaaab
- *Pattern*: aaaab

Algoritma Knuth-Morris-Pratt (KMP)

- Dikembangkan oleh D. E. Knuth, bersama-sama dengan J. H. Morris dan V. R. Pratt.

- Pada algoritma *brute force*, setiap kali ditemukan ketidakcocokan *pattern* dengan teks, maka *pattern* digeser satu karakter ke kanan.

- Sedangkan pada algoritma KMP, kita memelihara informasi yang digunakan untuk melakukan jumlah pergeseran.

- Algoritma KMP menggunakan informasi tersebut untuk membuat pergeseran yang lebih jauh, tidak hanya satu karakter seperti pada algoritma *brute force*.

```
            123456789…
Teks:       bimbingan belajar atau bimbel
Pattern:    bimbel
                 ↑
              j = 5

            123456789…
Teks:       bimbingan belajar atau bimbel
Pattern:       bimbel
                ↑
              j = 2
```

Definisi:

- Misalkan *A* adalah alfabet dan $x = x_1 x_2 \ldots x_k$ adalah *string* yang panjangnya *k* yang dibentuk dari karakter-karakter di dalam alfabet *A*.

- Awalan (*prefix*) dari *x* adalah upa-string (*substring*) *u* dengan

- $u = x_1 x_2 \ldots x_{j-1}$, $j \in \{1, 2, \ldots, k\}$
  dengan kata lain, *x* diawali dengan *u*.

- Akhiran (*suffix*) dari *x* adalah upa-string (*substring*) *u* dengan

- $u = x_{j-b}\, x_{j-b+1} \ldots x_k$, $j \in \{1, 2, \ldots, k\}$

dengan kata lain, *x* di akhiri dengan *v*.

- Pinggiran (*border*) dari *x* adalah upa-string *r* sedemikian sehingga

$$r = x_1 x_2 \ldots x_{j-1} \text{ dan}$$

$$u = x_{j-b}\, x_{j-b+1} \ldots x_j \,,$$

$$j \in \{1, 2, \ldots, k\}$$

- dengan kata lain, pinggiran dari *x* adalah upa-string yang keduanya awalan dan juga akhiran sebenarnya dari *x*.

Contoh 6. Misalkan x = abacab.

Awalan dari x adalah

, a, ab, aba, abac, abaca

Akhiran dari x adalah

, b, ab, cab, acab, bacab

Pinggiran dari x adalah

, ab

Pinggiran mempunyai panjang 0, pinggiran ab mempunyai panjang 2.

Fungsi Pinggiran (*Border Function)*

Fungsi *pinggiran b*(*j*) didefinisikan sebagai ukuran awalan terpanjang dari *P* yang merupakan akhiran dari *P*[1..*j*].

Sebagai contoh, tinjau *pattern P* = a b a b a a. Nilai *F* untuk setiap karakter di dalam *P* adalah sebagai berikut:

j	1	2	3	4	5	6
P[*j*]	a	b	a	b	a	a
b(*j*)	0	0	1	2	3	1

```
procedure HitungPinggiran(input m : integer, P : array[1..m] of char,
                          output b : array[1..m] of integer)
{ Menghitung nilai b[1..m] untuk pattern P[1..m] }

Deklarasi
  k, q : integer

Algoritma:
  b[1]←0
  q←2
  k←0
  for q←2 to m do
    while ((k > 0) and (P[q] ≠ P[k+1])) do
      k←b[k]
    endwhile
    if P[q]=P[k+1] then
      k←k+1
    endif
    b[q]=k
  endfor
```

Contoh 7:

Teks: a b c a b c a b d

Pattern: a b c a b d

Mula-mula kita hitung fungsi pinggiran untuk *pattern* tersebut:

j	1	2	3	4	5	6
$P[j]$	a	b	c	a	b	d
$b(j)$	0	0	0	1	2	0

Teks: a b c a b c a b d

Pattern: a b c a b d

↑

$j = 3$

Kompleksitas Waktu Algoritma KMP

- Menghitung fungsi pinggiran : $O(m)$,
- Pencarian *string* : $O(n)$
- Kompleksitas waktu algoritma KMP adalah $O(m+n)$.