PENCOCOKAN STRING DENGAN *FINITE AUTOMATA*

Firdi Mulia [1)] - NIM 13507045

1) Jurusan Teknik Informatika ITB,
Jalan Ganesha 10 Bandung Indonesia 40132
e-mail: if17045@students.if.itb.ac.id

ABSTRAK

Permasalahan pencocokan string semakin bertambah seiring dengan berjalannya waktu, bahkan permasalahannya berasal dari bidang-bidang yang berbeda. Seperti misalnya dalam pencarian pola DNA. Untaian DNA tertentu menggambarkan sifat-sifat dari mahkluk hidup. Jadi dengan pencarian pola dan urutan kemunculan dari suatu DNA, bisa dilihat kemiripan sifat dari DNA yang satu dengan yang lain. Permasalahan lain pencocokan string ada dalam bidang musik. Analis musik tertarik dalam mencari kemunculan pola, pengulangan pola, dan mungkin variasi dalam not musik.

Berbagai macam algoritma pencocokan string yang kompleksitas waktunya linier telah ditemukan, salah satunya adalah dengan menggunakan *finite automata*. Algoritma ini menarik karena kecepatannya dalam mencari string, algoritmiknya yang jelas, serta berlandaskan dari algoritma inilah muncul algoritma-algoritma pencocokan string lain seperti KMP(*Knuth Morris Pratt*) dan *Boyer-Moore*. Jadi, pada *finite automata* tersebut akan dibagi dalam *state-state*. Ada *state* yang menunjukkan bahwa string diterima. Jadi bila dalam pembacaan string, bisa masuk ke *state* tersebut, maka pola string ditemukan.

Kata kunci: *finite automata*, KMP, *Boyer-Moore,state*.

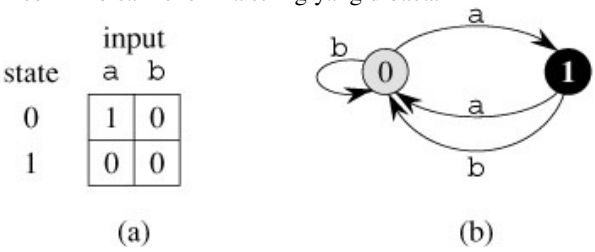
1. PENDAHULUAN

Pada bagian pendahuluan ini akan dibahas pengertian dari finite *automaton*.

Sebuah finite *automaton* M adalah sebuah 5-tuple $(Q,q_0,A,\sum,\delta)$ dimana

- Q adalah himpunan *state*
- q_0 adalah *start state*
- A adalah himpunan *state* yang diterima
- $\sum$ adalah himpunan alfabet masukan
- δ adalah fungsi transisi dari M

Finite automaton mulai dari *state* q_0 dan membaca karakter-karakter dari string masukan secara berurutan. Bila *automaton* pada *state* q dan membaca karakter masukan a, *automaton* tersebut akan berpindah dari *state* q ke *state* δ (q,a). Ketika *state* sekarang q adalah anggota A, mesin M bisa menerima string yang dibaca.

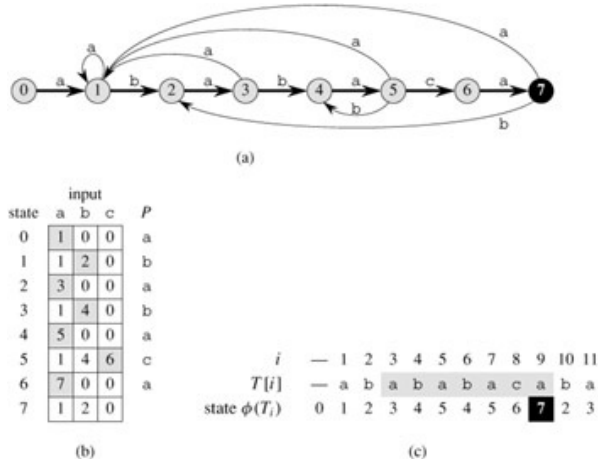


Gambar 1. *Finite automaton* sederhana dengan himpunan *state* Q = {0,1}, *state* awal q_0 = 0, dan input alphabet = {a,b}. (a) representasi tabel dari fungsi transisi δ. (b) Sebuah diagram transisi.

2. METODE

2.1 *Automaton* Pencocokan String

Ada sebuah *automaton* pencocokan string untuk setiap pola P; *automaton* ini harus dibangun dari pola dalam sebuah langkah pendahuluan sebelum digunakan untuk pencarian string teks. Misalnya contoh dibawah menggambarkan pembuatan *automaton* untuk pola P = ababaca.



Gambar 2 : (a) sebuah *state*-transition diagram untuk

***automaton* pencocokan string yang menerima semua string yang dengan akhiran string abababaca. (b) Transisi fungsi δ, dan pola string P = abababaca. (c) Operasi dari *automaton* dari teks T = abababacaba.**

Pada gambar 2(a) *state* 0 adalah *state* awal, dan *state* 7 (yang dihitamkan) adalah *state* yang diterima. Sebuah sisi berarah dari *state* i ke *state* j dilabelkan dengan a melambangkan δ(i,a) = j. Sisi kanan dari *automaton*, seperti yang ditunjukkan dengan garis tebal dalam gambar, berhubungan dengan pencocokan yang sesuai diantara pola dan karakter masukan. Sisi di sebelah kiri dilalui bila ditemukan pencocokan yang tidak sesuai. Beberapa sisi berhubungan dengan pencocokan yang tidak sesuai tidak ditunjukkan tapi dengan aturan, bila sebuah state i tidak mempunyai sisi keluar dengan label a, maka δ(i,a) = 0. Pada gambar 2(b) entri yang berhubungan dengan pencocokan yang sesuai antara pola dan karakter input dibuat berbayang.

Untuk menspesifikasikan *automaton* pencocokan string yang sesuai dengan pola P [1 ≤ m], kita perlu mendefinisikan fungsi σ yang memetakan $\sum^*$ ke {0,1, …, m} sehingga σ(x) adalah panjang dari awalan terpanjang dari P yang merupakan akhiran dari x:

Fungsi akhiran σ selalu terdefinisi karena string kosong $P_0 = \varepsilon$ adalah akhiran dari semua string. Sebagai contoh, untuk pola P = ab, terdapat σ(ε) = 0, σ(ccaca) = 1, dan σ (ccab) = 2. Untuk sebuah pola P dengan panjang m, terdapat σ(x) = m jika dan hanya jika P □ x. Dari sana bisa diturunkan definisi fungsi akhiran sebagai berikut jika x □ y, maka σ(x) ≤ σ(y).

Automaton pencocokan string didefinisikan sebagai :

- Himpunan state Q {0, 1, …, m}. State awal q_0 adalah state 0, dan state m adalah satu-satunya state yang diterima.
- Fungsi transisi δ didefinisikan dengan persamaan berikut, untuk setiap state q dan karakter a:

$$\delta(q,a) = \sigma(Pqa) \qquad (1)$$

Setelah membaca i karakter pertama dari string T, mesin ada pada state φ(Ti) = q, dimana q = σ (Ti) adalah panjang dari akhiran terpanjang dari Ti yang juga merupakan awalan dari pola P. Bila karakter berikutnya yang dibaca adalah T[i+1] =a, maka mesin akan membuat sebuah transisi ke state σ(Ti+1) = σ(Tia). Untuk menghitung panjang dari akhiran terpanjang dari Tia yang merupakan awalan dari P, akhiran terpanjang dapat dihitung Pqa yang merupakan awalan dari P. Pada setiap state, mesin hanya perlu mengetahui panjang awalan terpanjang dari P yang merupakan akhiran dari apa yang sudah dibaca sejauh ini. Pada *automaton* pencocokan string dari gambar 1, sebagai contoh, δ(5,b) = 4. Transisi ini dibuat karena bila *automaton* membaca a b pada state q =5, kemudian Pqb = ababab, dan awalan terpanjang dari P yang juga akhiran dari ababab adalah P4 = abab.

Untuk memperjelas operasi dari *automaton* pencocokan string, berikut contoh program sederhana, efisien untuk mensimulasikan *automaton* tersebut (transisi fungsinya dilambangkan dengan δ) dalam mencari kemunculan pola P pada string dengan panjang m dalam teks masukan T[1≤n]. Untuk semua *automaton* pencocokan string untuk pola dengan panjang m, himpunan state Q adalah {0, 1, …, m}. State awal adalah 0, dan state yang diterima adalah state m.

```
PENCOCOKAN-FINITE-AUTOMATON(T, δ, m)
1 n ← panjang[T]
2 q ← 0
3 for i ← 1 to n
4 do q ← δ(q, T[i])
5 if q = m
6  then print "Pola terjadi pada" i – m
```

Struktur pengulangan sederhana dari PENCOCOKAN-FINITE-AUTOMATON mengimplikasikan bahwa waktu pencocokan dari sebuah teks string dengan panjang n adalah Θ(n). Waktu pencocokan ini, tidak termasuk waktu preproses yang diperlukan untuk mendefinisikan fungsi transisi δ. Masalah ini akan dibahas pada poin berikutnya, setelah membuktikan PENCOCOKAN-FINITE-AUTOMATON berfungsi dengan benar.

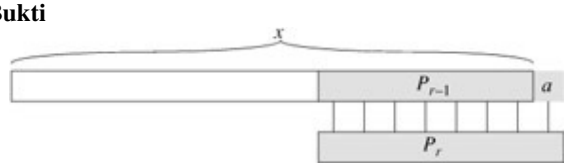
2.2. Pembuktian

Pada poin ini akan dibuktikan bahwa *automaton* pada state σ(Ti) setelah membaca karakter T[i]. Karena σ(Ti) = m jika dan hanya jika P ❐ Ti, mesin pada state m jika dan hanya jika pola P telah dibaca.Untuk membuktikan hasil ini, ada dua buah lemma untuk fungsi akhiran σ.

2.2.1 Lemma Ketidaksamaan Fungsi Akhiran

Untuk semua string x dan karakter a, terdapat persamaan σ(xa) ≤ σ(x) + 1.

Bukti



Gambar 3 : Ilustrasi yang menunjukkan r ≤ σ(x) + 1

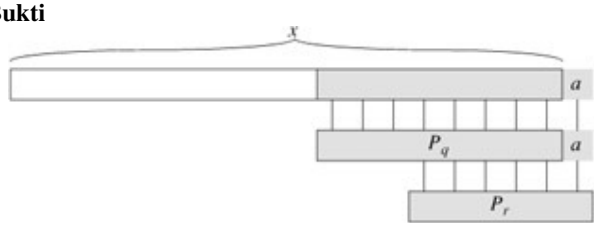
Misalnya pada kasus dengan contoh di atas, r = σ(xa). Jika r = 0, maka kesimpulan σ(xa) = r ≤ σ(x) + 1 terpenuhi, dengan nilai σ(x) tidak negatif. Jadi diasumsikan r > 0. Sekarang, Pr ❐ xa dengan definisi σ. Jadi, Pr-1 ❐ x, dengan menghilangkan a dari ujung Pr dan

dari ujung xa. Jadi, r-1 ≤ σ(x), karena σ(x) adalah k terbesar sehingga Pk ⊐ x, dan σ(xa) = r ≤ σ(x) + 1.

2.2.2 Lemma Rekursif Fungsi Akhiran

Untuk setiap string x dan karakter a, jika q = σ(x), jadi σ(xa) = σ(Pqa).

Bukti



Gambar 4. Ilustrasi pembuktian lemma 3.3. Gambar tersebut menunjukkan r = σ(Pqa), dimana q=σ(x) dan r = σ(xa).

Dari definisi σ, didapat Pq ⊐ x. Seperti ditunjukkan oleh gambar diatas, Pqa ⊐ xa. Jika r = σ(xa), kemudian r ≤ q + 1 dengan lemma 3.2. Karena Pqa ⊐ xa, Pr ⊐ xa, dan |Pr| ≤ |Pqa|, lemma 3.2 mengimplikasikan bahwa Pr ⊐ Pqa. Jadi, r ≤ σ(Pqa), yaitu σ(xa) ≤ σ(Pqa). Juga bisa disimpulkan σ(Pqa) ≤ σ(xa), karena Pqa ⊐ xa. Jadi, σ(xa) = σ(Pqa).

2.3 Pembuktian Teorema Pencocokan String

Dengan adanya dua lemma tersebut, pembuktian sebuah teorema yang mencirikan *automaton* pencocokan string dapat dilakukan. Seperti yang disebutkan di atas, pembuktian ini menunjukkan bahwa *automaton* menjaga, pada setiap langkah, awalan terpanjang dari pola yang merupakan akhiran dari apa yang sudah dibaca sejauh ini.

Jika φ adalah fungsi state-akhir dari *automaton* pencocokan string dari pola P dan T [1≤n] adalah teks masukan dari *automaton*, maka

$$\varphi(Ti) = \sigma(Ti) \text{ untuk } i = 0, 1, \ldots, n \qquad (2)$$

Bukti

Pembuktiannya adalah dengan cara induksi pada i. Untuk i=0, teoremanya sudah jelas bernilai benar, karena T0 = ε. Jadi, φ(T0) = 0 = σ(T0)

Sekarang, diasumsikan φ(Ti) = σ(Ti) dan membuktikan bahwa φ(Ti+1) = σ(Ti+1). Misalnya q melambangkan φ(Ti), dan a melambangkan T[i+1]. Maka

$$\begin{aligned} \varphi(Ti+1) &= \varphi(Tia) \\ &= \delta(\varphi(Ti),a) \\ &= \sigma(Pqa) \\ &= \sigma(Tia) \\ &= \sigma(Ti+1) \end{aligned}$$

Dengan teorema diatas, bila mesin memasuki state q pada baris 4, maka q adalah nilai terbesar yang menyebabkan Pq ⊐ Ti. Maka, didapat q = m pada baris 5 jika dan hanya jika sebuah kemunculan pola P telah dibaca. Dari sana bisa disimpulkan PENCOCOKAN-FINITE-AUTOMATON bekerja dengan benar.

2.4. Membuat Fungsi Transisi

Prosedur berikut menghitung fungsi transisi δ dari pola yang diberikan P[1 ≤ m].

```
COMPUTE-TRANSITION-FUNCTION(P, Σ)
1 m ← length[P]
2 for q ← 0 to m
3 do for each character a _ Σ
4 do k ← min(m + 1, q + 2)
5 repeat k ← k – 1
6 until Pk ⊐ Pqa
7 δ(q, a) ← k
8 return δ
```

Prosedur ini menghitung δ(q,a) dalam pendekatan yang langsung menurut definisinya. Kalang pengulangan mulai dari baris kedua dan ketiga dengan memperhitungkan semua state q dan karakter a, dan baris 4-7 mengeset nilai δ(q,a) menjadi nilai k terbesar dimana Pk · Pqa. Kode dimulai dengan nilai k terbesar yang mungkin, yaitu min(m,q+1), dan turunkan nilai k hingga Pk · Pqa.

2.5 Perbandingan dengan algoritma lain

Algoritma pencocokan string dengan *finite automaton* memerlukan waktu dengan kompleksitas $O(m\ |\sum|)$ untuk preproses membangun fungsi transisi *state* nya. Kemudian waktu pencocokan stringnya $\Theta(n)$.

Tabel 1 Perbandingan algoritma pencocokan string

Algoritma	Waktu Pencocokan
Brute Force	O(n-m+1)m)
Rabin-Karp	O((n-m+1)m)
Finite Automaton	$\Theta(n)$
Knuth-Morris-Pratt	$\Theta(n)$

3. KESIMPULAN

Algoritma pencocokan string dengan menggunakan finite *automaton* waktu pencocokannya cepat, bahkan setara dengan algoritma KMP. Hanya saja algoritma ini memerlukan waktu untuk preproses sehingga pada praktiknya lebih lambat dari KMP. Tetapi algoritma ini lebih jelas algoritmiknya dan dapat menjadi alternatif dalam memilih algoritma pencocokan string yang cepat.

REFERENSI

[1] English Wikipedia 2009
http://en.wikipedia.org/wiki/String_searching_algorithm
Tanggal akses : 28 Desember 2009 pukul : 22.00

[2] String Matching with Finite Automata
http://www.math.uic.edu/~leon/cs-mcs401-s08/handouts/finite-automata.pdf
Tanggal akses : 28 Desember 2009 pukul : 22.30

[3] J. E. Hopcroft, R. Motwani, and J. D. Ullman, “Introduction to Automata Theory, Languages, and Computation”, Addison-Wesley, Reading MA, 2001.

[4] T. Cormen, C. E. Leiserson, and R. L. Rivest, “Introduction to Algorithm”, The MIT Press, 2001.