Aplikasi Algoritma Pencocokan *String* pada Mesin Pencari Berita

Patrick Nugroho Hadiwinoto / 13515040
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13515040@std.stei.itb.ac.id

***Abstrak*—Berita merupakan kebutuhan yang mutlak bagi manusia modern zaman sekarang. Kebutuhan akan informasi menjadi mutlak untuk dipenuhi agar dapat menghadapi era teknologi informasi saat ini. Salah satu fasilitas untuk mendapatkan berita secara luas adalah dengan mesin pencari berita. Mesin ini mengumpulkan berita-berita dari berbagai situs berita nasional. Algoritma yang dipakai merupakan algoritma pencocokan *string* yang terdiri dari algoritma Knuth-Morris-Pratt (KMP), algoritma Boyer-Moore dan regex.**

Kata Kunci*—Knuth-Morris-Pratt (KMP), Boyer-Moore, regex, mesin pencari berita, pencocokan *string

I. PENDAHULUAN

Saat ini, kebutuhan manusia tidak hanya berkisar antara sandang, pangan dan papan. Era teknologi dan informasi yang sudah datang menuntut manusia untuk selalu mengikuti perkembangan berita yang ada. Berita-berita yang ada dapat berupa berita yang baik maupun buruk, fakta maupun *hoax*. Berita menjadi kebutuhan mutlak bagi manusia modern zaman sekarang.

Berita dapat kita peroeh dari mana saja, salah satunya dari media massa elektronik, seperti situs-situs berita di internet. Terkadang kita ingin mencari suatu topik berita tertentu saja, tidak semuanya. Pada zaman dahulu, kita harus mencarinya secara manual dengan membuka satu per satu berita yang ada di situs-situs berita tersebut.

Namun, kini sudah ada perkembangan teknologi yang memudahkan manusia untuk mencari berita yang diinginkan. Kita dapat menggunakan mesin pencari berita. Namun, apa sih, mesin pencari berita itu?

Mesin pencari berita merupakan fasilitas untuk mencari berita-berita tanpa perlu mengunjungi semua situs berita yang ada. Mesin pencari berita tersebut terhubung dengan situs-situs berita nasional. Kita hanya perlu memasukkan kata kunci pada mesin pencari berita tersebut dan mesin tersebut akan menampilkan semua berita yang mengandung kata kunci tersebut.

Mesin pencari berita dalam makalah ini bekerja dengan algoritma pencocokan *string*, di antaranya adalah algoritma Knuth-Morris-Pratt (KMP), algoritma Boyer-Moore dan regex. Ketiga algoritma ini memiliki kelebihan dan kekurangan masing-masing.

II. DASAR TEORI

A. Algoritma Pencocokan String

Pencarian *string* di dalam teks disebut juga pencocokan *string* (*string matching* atau *pattern matching*). Persoalan pencarian *string* dirumuskan sebagai berikut:

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*. Aplikasi dari masalah pencocokan *string* antara lain pencarian suatu kata di dalam dokumen (misalnya menu *Find* di dalam *Microsoft Word*).

Contoh:

Pattern: hari

Teks: kami pulang ke rumah ketika **hari** mulai malam

Pattern: not

Teks: nobody **not**iced him

Kita menggunakan teks berada di dalam memori (bila mencari *string* di dalam arsip, maka semua isi arsip – atau potongan besar data arsip – perlu dibaca terlebih dahulu dan menyimpannya di dalam memori). Jika *pattern* muncul lebih dari sekali di dalam teks, maka pencarianhanya memberikan keluaran berupa lokasi *pattern* ditemukan pertama kali. Selain itu, untuk membedakan antara *pattern* masukan dengan *pattern* di dalam teks, maka *pattern* yang berada di dalam teks kita namakan *target*.

Contoh:

Pattern: apa

Teks: Si**apa** yang menjemput Papa dari kota Balikpapan?

B. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma KMP ini mirip dengan algoritma *Brute Force*, hanya saja pada algoritma ini kita menyimpan informasi yang digunakan untuk melakukan jumlah pergeseran. Algoritma KMP menggunakan informasi tersebut untuk membuat pergeseran yang dilakukan menjadi lebih jauh (tidak hanya satu karakter ke kanan seperti pada algoritma *Brute Force*). Dengan ini, waktu pencarian dapat dikurangi secara signifikan.

Algoritma ini melakukan pencarian dari kiri ke kanan. Algoritma ini memanfaatkan fungsi pinggiran (*border function*) sebagai proses awal (*pre-processing*) terhadap *pattern* P. Beberapa literatur meyebutnya sebagai fungsi *overlap*, fungsi *failure*, fungsi awalan, dan lain-lain.

Fungsi pinggiran menyebabkan pergeseran s terbesar yang mungkin dengan menggunakan perbandingan yang dibentuk sebelum pencarian *string*. Hal ini bertujuan untuk menghindari pergeseran yang tidak berguna.

Fungsi pinggiran hanya bergantung pada karakter-karakter yang terdapat dalam *pattern*, sehingga kita dapat melakukan perhitungan fungsi pinggiran sebelum pencarian *string* dilakukan.

Fungsi pinggiran b(j) didefinisikan sebagai ukuran awalan (prefiks) terpanjang dari P[0..j-1] yang sama dengan akhiran (sufiks) dari P[1..j]. Mari kita ambil contoh yaitu *pattern* P = abaaba. Nilai b untuk tiap karakter di dalam P adalah sebagai berikut:

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

Tabel 1 Nilai b() untuk Tiap Karakter pada *pattern* P

Sumber: Diktat IF2211 Strategi Algoritma, Rinaldi Munir, 2009

Algoritma penghitungan fungsi pinggiran sebagai berikut:

```
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]}
Kamus Lokal
    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
```

Tabel 2 Algoritma Penghitungan Fungsi Pinggiran

Sumber: Diktat IF2211 Strategi Algoritma, Rinaldi Munir, 2009

Algoritma KMP selengkapnya adalah:

```
procedure KMPsearch (input m, n: integer, input P: array[1..m] of char, input T: array [1..n] of char, output idx : integer)
{Masukan: pattern P yang panjangnya m dan teks T yang panjangnya n}
{Keluaran: posisi awal kecocokan (idx). Jika P tidak ditemukan, idx = -1}
Kamus Lokal
    i, j : integer
    ketemu = boolean
    b : array[1..m] of integer
    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]}
Algoritma
    HitungPinggiran(m, P, b)
    j ← 0
    i ← 1
    ketemu ← false
    while (i ≤ n and not ketemu) do
      while ((j > 0) and (P[j+1] ≠ T[i])) do
            j ← b[j]
      endwhile
      if P[j+1] = T[i] then
            j ← j+1
      endif
      if j = m then
            ketemu ← true
      else
            i ← i+1
      endif
    endwhile
```

```
        if ketemu then
                idx ← i-m+1{ catatan : jika indeks array
                dimulai dari 0, smaka idx ← i-m }
        else
                idx ← -1
        endif
```

Tabel 3 Algoritma KMP

Sumber: Diktat IF2211 Strategi Algoritma, Rinaldi Munir, 2009

Kompleksitas waktu algoritma KMP adalah O(m+n) dengan rincian: untuk menghitung fungsi pinggiran dibutuhkan O(m), sedangkan pencarian *string* membutuhkan waktu O(n).

C. Algoritma Boyer-Moore

Algoritma Boyer-Moore melakukan perbandingan *pattern* P dari kanan ke kiri pada teks T. Algoritma ini bekerja berdasarkan pada dua teknik:

1. Teknik "*looking-glass*"

   Mencari P pada T dengan bergerak mundur sepanjang P, dimulai dari akhir P.

2. Teknik "*character-jump*"

   -Ketika terjadi ketidakcocokan pada T[i] = x

   -Karakter pada *pattern* P[j] tidak sama dengan T[i]

Ada 3 kasus yang mungkin terjadi saat terjadi ketidakcocokkan pada T[i] dengan P[j]:

1. T[i] terletak di kiri P[j]

   Geser P ke kanan, sehingga last occurrence dari x pada P sejajar dengan T[i]

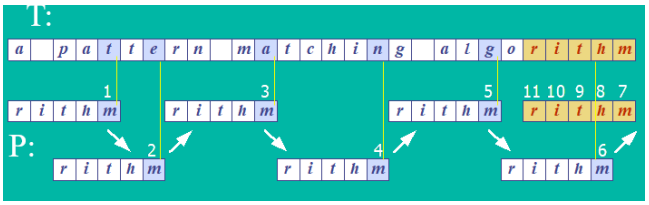
2. T[i] terletak di kanan P[j]

   Geser P ke kanan sebanyak 1 karakter

3. T[i] tidak terletak di kanan maupun di kiri P[j]

   Geser P ke kanan sehingga P[0] berada setelah T[i] (digeser sebesar P itu sendiri).

Contoh ilustrasi algoritma Boyer-Moore sebagai berikut (T adalah teks dan P adalah *pattern*):



Gambar 1 Ilustrasi Algoritma Boyer-Moore (angka menunjukkan pencocokan ke-sekian)

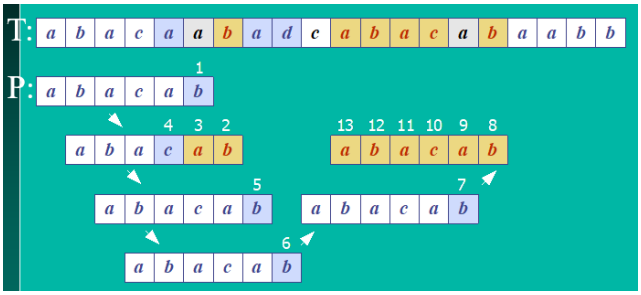
Sumber: Slide Pencocokan String, Kuliah Strategi Algoritma, 2017

Algoritma Boyer-Moore melakukan *pre-processing* terhadap *pattern* P dan alfabet A untuk membuat fungsi LastOccurence L(). L() memetakan semua huruf pada A menjadi bilangan bulat (*integer*).

L(x) didefinisikan sebagai: (x adalah huruf pada A)

- indeks i terbesar sehingga P[i]=x atau
- -1 jika indeks tersebut tidak terdapat pada *pattern*

Contoh L() dengan P dan T diberikan sebagai berikut:



Gambar 2 Ilustrasi Algoritma Boyer-Moore dengan penghitungan fungsi LastOccurence (L())

Sumber: Slide Pencocokan String, Kuliah Strategi Algoritma, 2017

x	a	b	c	d
L(x)	4	5	3	-1

Tabel 4 Tabel Penghitungan Fungsi LastOccurence

Sumber: Slide Pencocokan String, Kuliah Strategi Algoritma, 2017

Algoritma buildLast sebagai berikut:

```
function buildLast (pattern: array[1..m] of char) → array of integer

{mengembaikan array sorting index dari last occurrence dari setiap nilai ASCII dari karakter pada pattern}

  Kamus Lokal

        i, j : integer

        last : array[0..127] of integer {himpunan karakter ASCII}

  Algoritma

        for i ← 0 to 127 do
```

```
            last[i] ← -1  {menginisialisasi array}
        endfor
        for j ← 0 to m-1 do
            last[pattern[j]] ← j
        endfor
        → last
```

Tabel 5 Algoritma Penghitungan Fungsi buildLast

Sumber: Slide Pencocokan String, Kuliah Strategi Algoritma, 2017

Algoritma Boyer-Moore sebagai berikut:

```
function bmMatch (text: array[1..a] of char, pattern: array [1..b]of char ) → integer
{mengembaikan indeks pertama ditemukannya pattern pada teks, mengembalikan -1 jika tidak ketemu}
Kamus Lokal
    i, j,  n, m, lo, idx : integer
    last : array of integer
Algoritma
    last ← buildLast(pattern)
    n ← a
    m ← b
    if i > n-1 then {tidak cocok jika pattern lebih panjang dari teks}
        idx ← -1
    else
        j ← m-1
        while i ≤ n-1
        if pattern[j] = text[i] then
            if j=0 then
                idx ← i {match}
            else  {teknik looking glass}
                i ← i-1
                j ← j-1
            endif
        else {teknik character jump}
            lo ← last[text[i]] {last occurence}
            i ← i +m-min(j, 1+lo)
            j ← m-1
            idx ← -1
        endif
        endwhile
    endif
    → idx {mengembalikan indeks pertama ditemukannya pattern pada teks, -1 jika tidak ketemu}
```

Tabel 6 Algoritma Boyer-Moore

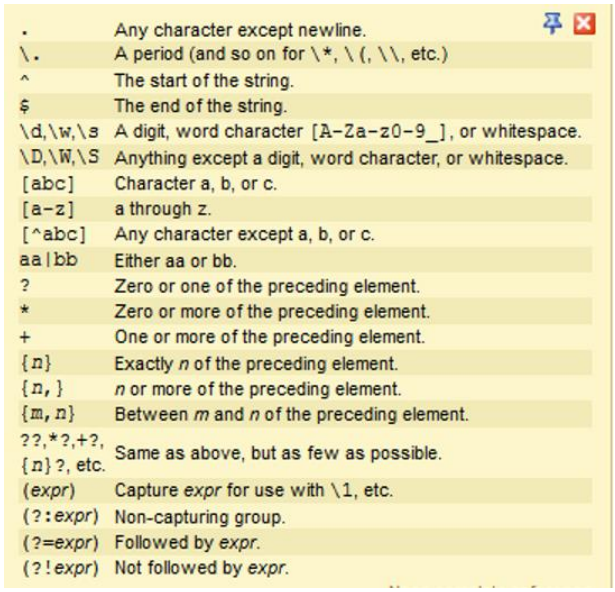
Sumber: Slide Pencocokan String Kuliah, Strategi Algoritma, 2017

Kompleksitas algoritma Boyer-Moore untuk kasus terburuk (*worst case*) sebesar O(nm+A).

D. Regex

Algoritma ini bekerja dengan prinsip *regular expression* (regex). Regex adalah sekumpulan karakter yang mendefinisikan sebuah *pattern* yang dicari (*search pattern*). Biasanya *pattern* ini digunakan untuk algoritma pencarian *string*.

Notasi regex di antaranya adalah ".", "\.", "\d", "\w" dan sebagainya. Algoritma pencocokan *string* dengan menggunakan regex tidak melakukan pencocokan secara *exact matching*, melainkan dengan pola tertentu saja. Hal ini berakibat pada waktu pencarian yang lebih cepat, walaupun kadang tidak terlalu teliti, karena pengecekan tidak dilakukan secara satu per satu.



.	Any character except newline.
\.	A period (and so on for *, \(, \\, etc.)
^	The start of the string.
$	The end of the string.
\d,\w,\s	A digit, word character [A-Za-z0-9_], or whitespace.
\D,\W,\S	Anything except a digit, word character, or whitespace.
[abc]	Character a, b, or c.
[a-z]	a through z.
[^abc]	Any character except a, b, or c.
aa|bb	Either aa or bb.
?	Zero or one of the preceding element.
*	Zero or more of the preceding element.
+	One or more of the preceding element.
{n}	Exactly *n* of the preceding element.
{n,}	*n* or more of the preceding element.
{m,n}	Between *m* and *n* of the preceding element.
??,*?,+?, {n}?, etc.	Same as above, but as few as possible.
(expr)	Capture *expr* for use with \1, etc.
(?:expr)	Non-capturing group.
(?=expr)	Followed by *expr*.
(?!expr)	Not followed by *expr*.

Gambar 3 Notasi Umum Regex
Sumber: Slide String Matching dengan Regex, Kuliah Strategi Algoritma, 2017

III. PEMBAHASAN

Mesin pencari berita bekerja dengan cara mengumpulkan berita-berita dari situs-situs berita nasional menggunakan

crawler berbasis RSS (*rich site summary* atau *really simple syndication*). Informasi yang dibutuhkan berupa judul, tanggal berita dan URL berita.

Langkah-langkah yang dilakukan adalah:

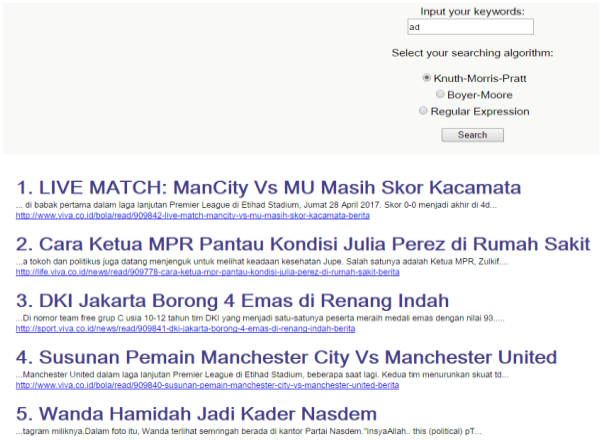
1. Melakukan pembacaan XML dari *link url* RSS tiap sumber berita. Dari pembacaan ini, didapatkan judul dan *link* berita
2. Kemudian dari *link* tersebut, didapatkan berita dalam bentuk HTML, dan dilakukan *parsing* terhadap HTML tersebut untuk mendapatkan judul dan isi dari berita
3. Dalam pembuatan Web, digunakan komponen asp::TextBox, untuk mendapatkan *keyword input* pengguna, dan digunakan komponen asp::RadioButton untuk mendapatkan pilihan metode algoritma pencocokan *string*
4. Pada setiap berita, dilakukan pencocokan *string* berdasarkan *keyword* dan jenis algoritma yang dipilih pengguna. Jika *keywords* berhasil ditemukan pada berita, tampilkan judul berita, potongan isi berita, dan *link* dari berita pada halaman Web.

Program ini dibuat dengan menggunakan bahasa pemrograman C#.

Pada proses *parsing* berita dari XML, dibuat suatu kelas Parsing yang memiliki *data member* sebagai antara lain:

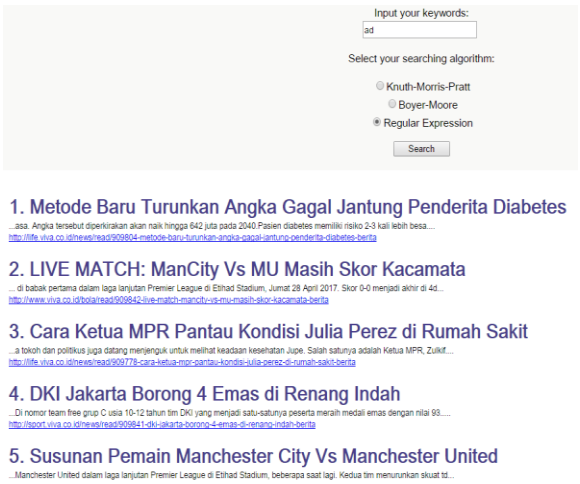
1. String[ ] link, *array of String* yang menyimpan daftar *link* dari berita
2. String[ ] berita, *array of String* yang menyimpan data daftar isi berita
3. String[ ] subject, *array of String* yang menyimpan data daftar judul berita
4. int indeks, menyimpan indeks berita yang sedang dilakukan parsing
5. int length, menyimpan banyaknya berita

Contoh eksekusi program:



Gambar 4 Hasil Pencarian Kata Kunci dengan Algoritma KMP
Sumber: Dokumen pribadi

Gambar 5 Hasil Pencarian Kata Kunci dengan Algoritma Boyer-Moore
Sumber: Dokumen pribadi



Gambar 6 Hasil Pencarian Kata Kunci dengan Regex
Sumber: Dokumen pribadi

Pada Gambar 4, 5 dan 6 kata kunci yang dimasukkan sama, yaitu “ad”. “ad” ini bertindak sebagai *pattern*, sedangkan yang bertindak sebagai teks adalah kumpulan berita-berita dalam RSS situs-situs berita nasional yang di- “*parsing*”. Semua berita tersebut diambil karakter-karakter yang membentuk berita dan tanggal berita. Dari teks yang begitu besar, dilakukanlah pencarian *pattern* (pencocokan *string*) dengan menggunakan *pattern* yang dimasukkan pengguna. Algoritma yang dipakai berupa algoritma KMP, Boyer-Moore dan regex. Mesin pencari berita akan menampilkan semua *link* berita yang mengandung kata kunci (*keywords*) tersebut dan tidak menampilkan apa-apa jika tidak menemukan kata kunci tersebut pada berita apapun.

Algoritma KMP melakukan perbandingan pada *pattern* dari kiri ke kanan dan memanfaatkan fungsi pinggiran (*boundary function*). Hal ini menyebabkan pencarian yang dilakukan sangat cepat jika dibandingkan dengan algoritma *Brute Force*, karena pergeseran yang dilakukan lebih efektif. Algoritma ini cepat dalam melakukan pencocokan pada teks dengan ragam karakter yang sedikit, tapi kurang baik pada teks dengan ragam karakter yang banyak.

Algoritma Boyer-Moore melakukan perbandingan pada *pattern* dari kanan ke kiri dan memanfaatkan fungsi LastOccurence. Algoritma ini akan lebih cepat jika ragam karakter yang dicocokkan banyak, tetapi lambat jika ragamnya sedikit. Algoritma ini sangat baik untuk melakukan pencarian pada teks berbahasa Inggris (karena ragamnya yang banyak), tetapi kurang baik pada teks biner (karena ragam karakternya sedikit). Algoritma ini merupakan salah satu algoritma *exact matching* terbaik yang pernah dikenal dalam dunia *computer science*.

Regex melakukan pencocokan *string* dengan mencocokkan teks dengan notasi umum pada regex, sehingga pencocokan yang dilakukan bukan merupakan *exact matching*. Hal ini memang terkadang memberikan hasil yang lebih cepat, tetapi terkadang hasil yang didapat tidak optimal (ada *substring* yang terlewat).

IV. KESIMPULAN

Dari pembahasan di atas dapat kita ambil kesimpulan bahwa berita merupakan elemen penting dalam kehidupan manusia. Salah satu fasilitas untuk mengumpulkan berita yang paling efektif adalah dengan menggunakan mesin pencari berita. Mesin ini bekerja dengan memanfaatkan algoritma pencocokan *string*.

Algoritma pencocokan *string* ini terdiri dari algoritma Knuth-Morris-Pratt (KMP), algoritma Boyer-Moore dan regex. Ketiga algoritma ini membantu kita dalam mencari berita berdasarkan pada kata kunci tertentu yang kita masukkan. Ketiga algoritma ini memiliki kelebihan dan kekurangannya masing-masing.

V. UCAPAN TERIMA KASIH

Puji dan syukur ke hadirat Tuhan Yang Maha Esa karena hanya dengan berkat dan rahmat-Nya lah penulis dapat menyelesaikan makalah ini dengan baik. Tidak lupa penulis juga berterima kasih kepada Ibu Dr. Masayu Leylia Khodra, Ibu Dr. Nur Ulfa Maulidevi dan Bapak Dr. Rinaldi Munir selaku dosen mata kuliah Strategi Algoritma yang telah membimbing penulis dalam menyelesaikan makalah ini. Serta kepada teman-teman yang memberikan koreksi dan motivasi dalam penulisan.

REFERENSI

[1] Slide Pencocokan String, Kuliah Strategi Algoritma, 2017

[2] Slide String Matching dengan Regex, Kuliah Strategi Algoritma, 2017

[3] Munir, Rinaldi. *Diktat Kuliah IF2211 Strategi Algoritma* Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2009.

[4] http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/StringMatch/boyerMoore.htm

[5] http://regexpal.com.s3-website-us-east-1.amazonaws.com/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 18 Mei 2017

Patrick Nugroho Hadiwinoto - 13515040