ANALISIS PERBANDINGAN ALGORITMA STRING MATCHING BRUTE FORCE DAN KNUTH-MORRIS-PRATT

Aditya Andar Rahim[1] Irfan Ahmad Asqolani[2] Muhammad Zalfa Thoriq[3]

NIM: 1301194029[1] 1301190323[2] 1301194443[3]
e-mail: adityaandar@student.telkomuniversity.ac.id[1] irfanahmad@student.telkomuniversity.ac.id[2]
zalfathoriq@student.telkomuniversity.ac.id[3]

ABSTRAK

Algoritma memiliki peran yang sangat penting dalam perangkat lunak ataupun pemrograman, sehingga diperlukan pemahaman dari konsep dasar suatu algoritma, dan juga setiap algoritma memiliki tingkat kompleksitas waktu yang berbeda. *String Matching* atau sering disebut algoritma pencocokan *string* yaitu algoritma untuk melakukan pencarian semua kemunculan *string* pendek dan panjang, untuk string pendek disebut *pattern* dan *string* yang lebih panjang disebut teks, dengan kata lain algoritma ini dikhususkan untuk mencocokkan *string pattern* pada suatu teks. Algoritma S*tring Matching* ini memiliki banyak jenis algoritma yang dapat digunakan beberapa diantaranya, yaitu algoritma *Brute Force*, *Knuth-Morris-Pratt*, dan *Boyer-Moore*. Namun pada makalah ini hanya berfokus membahas tentang perbandingan kompleksitas dan efektivitas algoritma *Brute Force* dan algoritma *Knuth-Morris- Pratt* yang merupakan algoritma *String Matching* yang umum digunakan. Tujuan dari makalah ini yaitu mencari algoritma yang lebih efektif untuk pencocokan *string* dalam kehidupan sehari-hari. Pengujian dilakukan dengan menguji lima *pattern* sampel data terhadap source string yang telah ditentukan, Kemudian akan dilihat kompleksitas dan *running time*nya antara algoritma *Brute Force* dan Algoritma *Knuth-Morris-Pratt*. Setelah pengujian, didapatkan bahwa algoritma *knuth-morris-pratt* lebih cepat dan efektif dalam menemukan *pattern* sampel data.

Kata kunci: *String Matching, Brute Force, Knuth-Morris- Pratt, Exact String Matching, Inexact String Matching.*

1. PENDAHULUAN

Di masa yang serba *digital* kini, perkembangan teknologi informasi dan komunikasi berkembang dengan sangat pesat, karena hal tersebut pencarian informasi menjadi semakin mudah dan cepat, penting bagi kita untuk mendapatkan informasi yang tepat dan sesuai dengan kebutuhan. Teks atau tulisan merupakan bentuk informasi yang paling banyak digunakan, tanpa kita sadari 95% informasi atau konten pada internet berbentuk tulisan atau teks. Dengan banyaknya informasi dalam bentuk teks atau tulisan, maka sangat diperlukan teknik yang dapat memperoleh informasi dengan isi yang sesuai dengan kebutuhan. Teknik tersebut yaitu pencocokan string (*string matching*), dengan algoritma *string matching* kita dapat mencari kata, frasa, atau kalimat untuk mempermudah pencarian informasi yang sesuai dengan kebutuhan.

String matching merupakan pencarian semua pemunculan suatu pola tertentu pada teks. Pola dan teks tersebut keduanya merupakan *string* yang terbentuk dari suatu *alphabet* tertentu (kumpulan karakter terbatas). Pada penjelasan di bagian bawah ini semua algoritma *string* matching menghasilkan semua pemunculan dari pola yang terdapat pada teks. Pola didenotasikan dengan x={0...m-1}, panjang pola sama dengan m. Teks didenotasikan sebagai y={0...n-1}(Sari, 2019). *String matching* secara garis besar dibedakan menjadi dua yaitu *exact string matching* merupakan pencocokan *string* secara tepat dengan susunan karakter dalam *string* yang dicocokkan, memiliki jumlah dan urutan karakter kata yang dalam *string* yang sama dan *inexact string matching* merupakan pencocokan *string* secara samar, yaitu pencocokan *string* dimana *string* yang dicocokkan memiliki kemiripan namun keduanya memiliki susunan karakter yang berbeda (baik jumlah atau urutannya) tetapi *string-string* tersebut memiliki kemiripan baik kemiripan tekstual (*approximate string matching*) atau kemiripan ucapan (*Phonetic string matching*) (Sagita & Prasetiyowati, 2012).

Makalah ini akan berfokus pada algoritma *exact string matching,* algoritma yang akan dibahas yaitu algoritma *string matching brute force* dan algoritma *string matching Knuth-Morris-Pratt*. Algoritma *Brute Force merupakan algoritma string matching* yang melakukan pengecekan dari kiri ke kanan, digunakan untuk mengecek pada setiap kedudukan *string* dalam teks mulai karakter awal hingga karakter akhir. Setelah melakukan pengecekan pada karakter pertama, maka proses *shift* dilakukan, yaitu dengan memindahkan *string* tepat satu posisi ke arah

kanan atau karakter akan berpindah menuju karakter kedua, ketiga dan seterusnya. Algoritma *knuth-morris-pratt* merupakan algoritma *string matching* yang melakukan pengecekan dari kiri ke kanan, sama seperti *brute force*, Tetapi algoritma knuth-morris-pratt melakukan pergeseran pengecekan string yang jauh lebih efisien dibandingkan dengan *brute force.* Makalah ini juga akan membahas tentang perbandingan penggunaan antara kedua algoritma *string matching* tersebut.

2. METODE STRING MATCHING

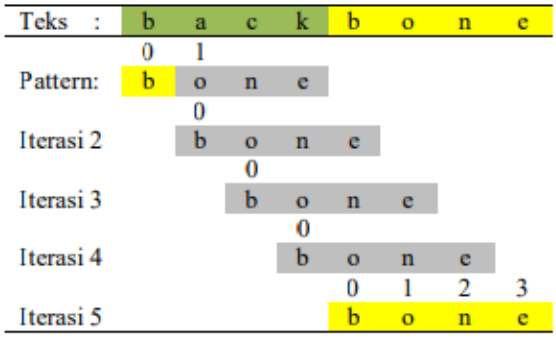
2.1 Algoritma String Matching #1: Brute Force

a. Konsep *String Matching Brute Force*

Algoritma *brute force* atau *naïve algorithm* merupakan algoritma *string matching* yang ditulis tanpa memikirkan peningkatan performa. Algoritma brute force memecahkan masalah dengan sangat sederhana, langsung dan dengan cara yang jelas (Sedgewick, 1984:243).

Untuk melakukan pencocokkan *string*, terdapat istilah teks dan *pattern*. *pattern* merupakan kata yang dicari yang akan dicocokkan dengan teks, sedangkan teks adalah kata atau *string* yang dimasukkan untuk diperiksa apakah terdapat pattern yang dicari dalam teks tersebut. Secara rinci, langkah-langkah yang dilakukan dalam algoritma ini saat mencocokkan *string* adalah (Munir, 2004:2) :

1. Algoritma Brute Force memulai pemeriksaan pattern dari awal teks.
2. Dari kiri ke kanan, algoritma ini memeriksa apakah karakter per karakter *pattern* cocok dengan karakter pada teks yang bersesuaian, sampai salah satu kondisi berikut terpenuhi:
   1) Karakter pada *pattern* dan teks yang dibandingkan tidak cocok.
   2) Semua karakter di *pattern* cocok, kemudian algoritma akan memberitahukan penemuan di posisi ini.
3. Algoritma akan terus bergeser *pattern* sebesar satu karakter ke kanan, dan mengulangi langkah ke-2 sampai *pattern* sudah berada di ujung teks.



Gambar 1. Ilustrasi *String Matching* Algoritma *Brute Force*

b. *Pseudocode* dan Kompleksitas *String Matching Brute Force*

```
procedure BruteForceSearch(input m, n : integer,
input P : array[1..m]of char, input T : array[1..n]
of char, output idx : integer)

{Mencari kecocokan pattern P di dalam teks T.
Jika ditemukan P didalam T, lokasi awal
kecocokan disimpan di dalam peubah idx.
Masukan: pattern P yang panjangnya m dan teks
T yang panjangnya n. Teks T direpresentasikan
sebagai string (array of character)
Keluaran: posisi awal kecocokan (idx). Jika P
tidak ditemukan, idx =-1.}

Deklarasi
  i : integer
  ketemu : boolean

Algoritma:
  i ← 0
  ketemu ← false
  while (i ≤ n – m) and (not ketemu) do
    j ← 1
    while (j ≤ m) and (Pj = Ti+j) do
      j ← j + 1
    endwhile
    { j > m or Pj ≠ Ti+j }

    if (j = m) then { kecocokan string ditemukan}
      ketemu ← true
    else
      i ← i + 1 {geser pattern satu karakter ke
      kanan teks }
    endif
  endwhile
  { i > n – m or ketemu }
  if ketemu then
```

```
    idx ← i + 1 { catatan: jika indeks array
    dimulai dari 0, idx  s }
  else
    idx ← –1
  endif
```

Diasumsikan bahwa m = panjang *pattern* dan n = panjang teks maka Kompleksitas waktu algoritma *string matching brute force* untuk kasus terburuk adalah m(n – m + 1) = O(mn). kasus terbaik terjadi jika karakter pertama pattern m tidak pernah sama dengan karakter teks n yang dicocokkan maka kompleksitasnya adalah O(n).

2.2 Algoritma String Matching #2: Knuth-Morris-Pratt

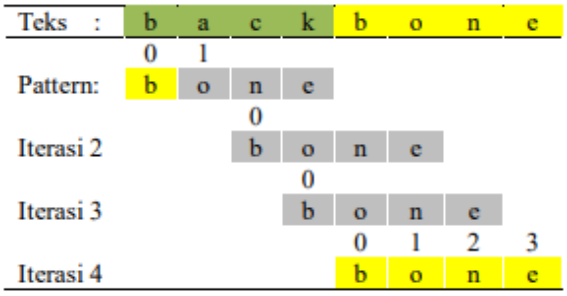
a. Konsep *String Matching Knuth-Morris-Pratt*

Algoritma *knuth-morris-pratt* merupakan salah satu algoritma pencarian *string* yang dikembangkan secara terpisah oleh Donald R. Knuth pada tahun 1967 dan James H. Morris bersama Vaughan R. Pratt pada tahun 1966, namun keduanya mempublikasikan secara bersamaan pada tahun 1977 (Sedgewick, 1984:242).

Algoritma ini merupakan algoritma hasil pengembangan *brute force* dengan membuat algoritma yang lebih efisien melalui dua metode yaitu mengingat dan melompat. Untuk menunjang hal tersebut, algoritma *knuth-morris-pratt* memakai fungsi yang diberi nama *Border Function* atau hitung pinggiran yang digunakan untuk melompati pengecekan *substring* agar tidak perlu terjadi pengecekan berulang yang tidak dibutuhkan. Dengan kata lain, penggunaan *border function* atau hitung pinggiran adalah untuk mengingat banyaknya kesamaan *substring* sebelum kesalahan pencocokan ditemukan pada suatu karakter, dan memakainya untuk melakukan lompatan pergeseran.

Berdasarkan informasi tersebut, waktu proses menjadi berkurang dan pencarian *pattern* menjadi lebih cepat.Algoritma *knuth-morris-pratt* melakukan perbandingan *pattern* mulai dari kiri ke kanan seperti cara kerja yang dilakukan oleh algoritma *brute force*. Secara sistematis, langkah-langkah yang dilakukan algoritma *knuth-morris-pratt* pada saat pencocokan string (Supardi, 2009:18) :

1. Algoritma *Knuth-Morris-Pratt* mulai mencocokkan *pattern* pada awal teks.
2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter *pattern* dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
   1) Karakter di *pattern* dan di teks yang dibandingkan tidak cocok (*mismatch*).
   2) Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
3. Algoritma kemudian menggeser pattern berdasarkan prosedur hitung pinggiran / tabel *kmpNext*, lalu mengulangi langkah 2 sampai *pattern* berada di ujung teks.



Gambar 2 . Ilustrasi *String Matching* Algoritma *Knuth-Morris-Pratt*

b. *Pseudocode* dan Kompleksitas *String Matching Knuth-Morris-Pratt*

```
procedure HitungPinggiran (
  Input P : array [0…n-1] of char,
  Input n : integer,
  output kmpNext : array [0…n-1] of integer
)

Deklarasi
  i, j : integer

Algoritma
  i ← 0
  j ← -1
  kmpNext[0] ← -1
  while (i < n - 1) do
    while ( j > -1 and (P[i] ≠ P[j])) do
      j ← kmpNext[j]
    i ← i + 1
    j ← j + 1
    if (P[i] = P[J]) then
      kmpNext[i] ← kmpNext[j]
    else
      kmpNext[i] ← j
    endif
  endwhile
```

```
procedure KMPsearch (
  Input  m, n: integer,
  Input P : array [0…n-1] of char,
  Input T : array [0…n-1] of char,
```

```
   output ketemu: array [0…n-1] of boolean
)

Deklarasi
   i, j, next : integer
   kmpNext : array [0…n-1] of integer

Algoritma
   HitungPinggiran(n, P, kmpNext)
   i ← 0
   next ← 1
   ketemu ← false
   while (i ≤ m-n ) do
      j  ← 0
      while ( j < n and (T[i+j] = P[j])) do
         j ← j + 1
      endwhile
      if (j ≥ n) then
         ketemu ← true
         next ← j
      endif
      next ← j – kmpNext[j]
      i ← i + next
   endwhile
```

Diasumsikan bahwa m = panjang *pattern* dan n = panjang teks maka kompleksitas waktu untuk menghitung fungsi pinggiran atau *border function* dibutuhkan waktu yaitu O(m), sedangkan saat pencarian *string* membutuhkan waktu O(n), dapat disimpulkan kompleksitas waktu algoritma *string matching knuth-morris-pratt* adalah O(m+n).

3. Hasil eksperimen/pengujian

3.1. Sampel data

Pengujian algoritma *brute force* dan algoritma *knuth-morris-pratt* dilakukan dengan mencocokan beberapa sampel data *(pattern)* ke dalam suatu teks dengan panjang 520 kata. berikut adalah sampel data yang kami gunakan :

1. deron
2. ayahnya
3. sementara
4. klinik
5. teman

3.2. Hasil pengujian algoritma *brute force* terhadap sampel data

Running time hasil pengujian sampel data 1-5 dengan algoritma *brute force* dapat dilihat pada tabel berikut

Tabel 1. *Running time* uji sampel data menggunakan algoritma *brute force*

Sampel Uji	Waktu(s)
deron	0.008995
ayahnya	0.003999
sementara	0.003996
klinik	0.004014
teman	0.004012

3.3. Hasil pengujian algoritma knuth-morris-pratt terhadap sampel data

Running time pengujian sampel data 1-5 dengan algoritma knuth-morris-pratt dapat dilihat pada tabel berikut

Tabel 2. Running time uji sampel data menggunakan algoritma knuth-morris-pratt

Sampel Uji	Waktu(s)
deron	0.001997
ayahnya	0.002999
sementara	0.002016
klinik	0.001994
teman	0.003016

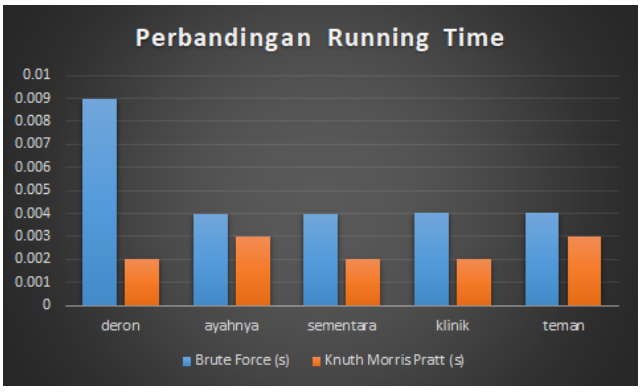
3.4. Running time kedua algoritma & analisis komparasi

Perbandingan *running time* pengujian sampel data 1-5 dengan menggunakan algoritma *brute force* dan *knuth-morris-prat*t secara keseluruhan dapat dilihat pada tabel berikut

Tabel 3 . Komparasi running time pengujian sampel data menggunakan algoritma *brute force* dan *knuth-morris-pratt*

Sampel Uji	Brute Force	Knuth-Morris-Pratt
deron	0.008995	0.001997
ayahnya	0.003999	0.002999
sementara	0.003996	0.002016
klinik	0.004014	0.001994
teman	0.004012	0.003016

Perbandingan *running time* pengujian sampel data 1-5 dengan menggunakan algoritma *brute force* dan *knuth-morris-pratt* secara keseluruhan dapat digambarkan seperti grafik berikut



Gambar 3 . Grafik perbandingan *running time* algoritma *Brute Force* dan *Knuth-Morris-Pratt*

Berdasarkan tabel dan grafik secara keseluruhan *running time* pencocokan string menggunakan algoritma *knuth-morris-pratt* selalu lebih cepat dibandingkan dengan algoritma *brute force*

3.5. Analisis kompleksitas kedua algoritma dikaitkan dengan running time pada 3.4

Secara keseluruhan *running time* pencocokan string menggunakan algoritma *knuth-morris-pratt* selalu lebih cepat dibandingkan dengan algoritma *brute force*, hal ini sesuai dengan kompleksitas dari masing-masing algoritma dimana kompleksitas algoritma *knuth-morris-pratt* yaitu O(m+n) lebih baik daripada kompleksitas algoritma *brute force* O(mn).

4. KESIMPULAN

Dari hasil percobaan yang kami lakukan, Algoritma *brute force* merupakan algoritma yang paling lempang atau sederhana dikarenakan pada algoritma *brute force* tidak terdapat *preprocessing,* Sedangkan pada algoritma *knuth-morris-pratt* terdapat *preprocessing yang* dapat mengurangi waktu proses pencarian. Saat melakukan pengujian pada beberapa sampel data *pattern*, algoritma *knuth-morris-pratt* selalu mendapatkan waktu proses yang lebih baik daripada algoritma *brute force*. Oleh karena itu, dalam melakukan pencocokan *string* lebih disarankan menggunakan algoritma *knuth-morris-pratt untuk* mendapatkan hasil waktu yang lebih baik.

LAMPIRAN

Source string, file excel tabel & grafik, program aplikasi beserta screenshot pengujian sampel data dapat dilihat pada link berikut:
https://drive.google.com/drive/folders/15ke62kDbRKE3QAalmz5EFctsKbA0zNYT?usp=sharing

REFERENSI

[1] Setiawan, C. B. *Penerapan dan Perbandingan Algoritma String Matching Pada Aplikasi UUD 1945 dan UU di Indonesia.* Makalah IF2211 Strategi Algoritma. 2019. 1-7

[2] Wibowo, K. *Perbandingan Algoritma Knuth-Morris-Pratt dan Algoritma Boyer-Moore dalam Pencarian Teks di Bahasa Indonesia dan Inggris.* Makalah IF3051 Strategi Algoritma. 2011. 1-7

[3] Sari, R. *Perancangan Aplikasi Pencocokan String Pada Dokumen Menggunakan Algoritma Not So Naive Pada Editor Teks*. Jurnal Pelita Informatika. Vol 8. No 1. 2019. 1-6

[4] Utomo,D. Harjo,E.W.& Handoko. *Perbandingan Algoritma String Searching Brute Force, Knuth-Morris-Pratt, Boyer-Moore dan Karp Rabin Pada Teks Bahasa Alkitab Bahasa Indonesia*. Jurnal Ilmiah Elektronika. Vol 7. No 1. 2008. 1-13

[5] Sagita, V. & Prasetiyowati. *Studi Perbandingan Implementasi Algoritma Boyer Moore, Turbo Boyer-Moore, dan Tuned Boyer-Moore dalam Pencarian String*. Jurnal Ultimatics,.Vol IV. No 1. 2021. 31-37.

[6] Rochmawati,Y. & Kusumaningrum, R. *Studi Perbandingan Algoritma Pencarian String dalam Metode Approximate String Matching Untuk Identifikasi Kesalahan Pengetikan Teks*.Jurnal Buana Informatika. Vol 7. No 2. 2016. 125-134

[7] Nababan, A.A. & Jannah,M. *Algoritma String Matching Brute Force dan Knuth-Morris-Pratt Sebagai Search Engine Berbasis Web Pada Kamus Istilah Jaringan Komputer*. Jurnal Mantik Penusa. Vol 3. No 2. 2019. 87-94

[8] Pratiwi,H. Arfyanti,I. & Kurniawan,D. *Implementasi Algoritma Brute Force Dalam Aplikasi Kamus Istilah Kesehatan*. Jurnal Ilmiah Teknologi Informasi Terapan. Vol II. No 2. 2016. 119-125

[9] Fernando, Harry. *Perbandingan dan Pengujian Beberapa Algoritma Pencocokan String.* Makalah IF2251 Strategi Algoritmik. 2009. 1-7

[10] Nugroho,Y.A. *Penggunaan Algoritma Brute Force dan KMP dalam Pencocokan Dua Gambar.* Makalah IF3051 Strategi Algoritma. 2012. 1-4

[11] Siregar,H.K. *Perbandingan Algoritma Knuth-Morris-Pratt Dan Apostolico-Crochemoe Pada Aplikasi Kamus Bahasa Indonesia - Belanda.* 2017. 7-10

[12] Waruwu, F.T. & Mandala, R. *Perbandingan Algoritma Knuth Morris Pratt Dan Boyer Moore Dalam Pencocokan String Pada Aplikasi Kamus Bahasa Nias*. Jurnal Ilmiah Infotek. Vol 1. No 1. 2016. 36-43

[13] Supardi. *Analisis Dan Penerapan Algoritma String Matching Pada Aplikasi Pencarian Berkas Di Komputer.* 2009. 14-20

[14] Sedgewick, R. Algorithms. USA: Addison-Wesley. 1984

[15] Munir, R. *Algoritma Pencarian String (String Matching)*. Bahan Kuliah IF2251 Strategi Algoritmik. 2004.