PENERAPAN ALGORITMA STRING MATCHING DALAM PENCOCOKAN DATA STRING

Rizkyria Angelina Pandapotan Hutabarat[1], Juwita Stefany Hutapea[2] dan Marlina Magdalena Lubis[3]

[1,2,3)]Program Studi D4 Teknik Informatika,
Universitas Logistik dan Bisnis Internasional
Jl. Sariasih No. 54, Bandung, Indonesia

email: [1)] rizkyriahutabarat18@gmail.com,[2)] juwitastefany13@gmail.com,[3)] lubismarlina5@gmail.com

Abstrak

Algoritma string matching merupakan metode pemrosesan teks yang digunakan untuk mencari pola yang cocok dalam suatu rangkaian karakter. Dalam studi ini, kami menyajikan sebuah tinjauan mengenai *systematic literature review* (SLR) yang bertujuan untuk mengumpulkan, mengevaluasi, dan menganalisis penelitian yang telah dilakukan sebelumnya mengenai algoritma pencocokan string. Tujuan dari penelitian ini adalah untuk menganalisis dan menerapkan algoritma pencocokan string dalam proses pencocokan data teks. Beberapa algoritma pencocokan string yang akan dibahas dan diterapkan meliputi algoritma Brute-Force, algoritma Knuth-Morris-Pratt (KMP). Diharapkan hasil dari penelitian ini dapat memberikan pemahaman yang lebih baik tentang algoritma pencocokan string menggunakan metode SLR dan membantu pengembang dalam memilih algoritma yang paling sesuai dengan kebutuhan aplikasi yang dimiliki.

Kata Kunci: Algoritma String Matching, SLR, String, Algoritma Brute-Force, Algoritma Knuth-Morris-Pratt (KMP)

Abstract

The string matching algorithm is a text processing method used to search for matching patterns within a sequence of characters. In this study, we present a review on systematic literature review (SLR) aimed at gathering, evaluating, and analyzing previous research on string matching algorithms. The objective of this research is to analyze and implement string matching algorithms in the process of matching text data. Several string matching algorithms that will be discussed and implemented include the Brute-Force algorithm and the Knuth-Morris-Pratt (KMP) algorithm. The results of this study are expected to provide a better understanding of string matching algorithms using the SLR method and assist developers in selecting the most suitable algorithm for their application needs.

Keywords: *Algorithm String Matching, SLR, String, Brute-Force Algorithm, Knuth-Morris-Pratt (KMP) Algorithm*

1. PENDAHULUAN

1.1 Latar Belakang

Algoritma String Matching merupakan teknik yang digunakan dalam pemrosesan teks untuk mencari kemunculan suatu pola string di dalam teks yang lebih besar. Penerapan algoritma string matching ini memiliki berbagai aplikasi, seperti dalam analisis teks, pengenalan pola, pemrosesan bahasa alami, dan bioinformatika. Pencocokan string sangat penting dalam aplikasi ini karena seringkali melibatkan pemrosesan teks yang sangat besar.

Salah satu metode yang digunakan untuk meningkatkan efisiensi algoritma string matching adalah Metode String Length Reduction (SLR). Metode ini berfokus pada reduksi panjang string yang diproses saat pencocokan string dilakukan. SLR mengurangi jumlah operasi yang harus dilakukan pada setiap langkah pencocokan, sehingga

meningkatkan kinerja algoritma secara keseluruhan.

1.2 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk mengimplementasikan sebuah algoritma String Matching dan melakukan sebuah analisis dengan menggunakan metode SLR. Penelitian ini bertujuan untuk memahami prinsip- prinsip dasar dari metode SLR dan melaukan pengujian cara kerja dalam membandingkan dengan metode pencocokan string tradisional seperti algoritma Brute Force dan algoritma Knuth-Morris-Pratt (KMP).

2. LANDASAN TEORI

2.1. *Systematic Literatur Review* (SLR)

Systematic Review merupakan sebuah istilah yang digunakan untuk metodologi atau penelitian tertentu yang diciptakan untuk mengumpulkan dan mengevaluasi penelitian yang berkaitan dengan topik tertentu. Beberapa peneliti yang telah melakukan penelitian menggunakan SLR mendefinisikan SLR sebagai berikut [9]:

a. SLR digunakan untuk mengevaluasi dan mengindentifikasi sebuah penelitian yang tersedia dengan menggunakan teknologi yang tersedia untuk mengidentifikasi dan mengevaluasi.
b. SLR digunakan untuk mengidentifikasi semua penelitian yang sudah ada dengan topik, pertanyaan tertentu dan suatu fenomena yang menarik.
c. SLR merupakan teknik yang digunakan untuk meninjau atau menemukan isu dalam sebuah *Software Engineering.*

2.2. String Matching

Pada dasarnya, istilah pencocokan *string* biasanya mencakup antara teks dan *pattern* atau pola. Teks adalah string dengan panjang n. Patter disebut juga pola yaitu string dengan panjang m yang dimana karakter yang akan dicari dalam teks yaitu (m < n) [4].

2.3 *Algoritma Brute Force*

Algoritma Brute Force merupakan suatu metode pendekatan langsung yang digunakan untuk memecahkan permasalahan dengan mengikuti langkah-langkah yang ada dan yang terlibat dalam suatu permasalahan tersebut[4]. Algoritma ini memiliki kemampuan untuk menyelesaikan berbagai macam masalah yang mudah, kompleks, sederhana, dan jelas. Dalam konteks pencocokan string, istilah yang digunakan adalah teks dan pola. Teks merujuk pada kata yang ingin dicocokkan dengan pola tertentu[16].

Pencocokan *string brute force* merupakan pendekatan yang cukup sederhana. Algoritma akan mencoba mencocokkan pola P dengan sub-string teks T pada posisi berurutan dari kiri ke kanan. Kemudian, setiap ketidaksesuaian algoritma akan menggeser pola tepat satu posisi ke kanan. Maka dari itu, algoritma memeriksa semua posisi dalam teks T dari tanggal 1 sampai (n-m), apakah kemunculan pola tersebut dimulai dari sana atau tidak (Azizah Abd Manaf et al., 2011).

2.4 *Algoritma Knuth Morris Pratt*

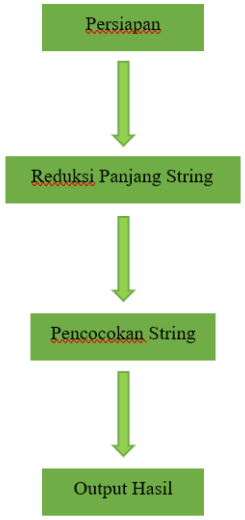
Algoritma Knuth Morris Pratt atau disebut dengan (KMP) adalah hasil pengembangan dari D.E. Knuth, Bersama dengan J.H. Morris dan V.R.Pratt. Algoritma ini digunakan untuk mencari pola pada string menggunakan pendekatan Brute Force, namun dengan setiap ketidakcocokan yang ditemukan pada teks, pola akan digeser satu karakter ke kanan. Perbedaannya dengan algoritma Brute Force terletak pada pengolahan informasi yang digunakan untuk menentukan jumlah pergeseran yang dilakukan.

3. METODE PENELITIAN

Metode penelitian yang digunakan dalam penerapan Algoritma String Matching dalam pencocokan data String adalah sebagai berikut :

1. Deskripsi Algoritma String Matching dengan Metode SLR

Algoritma String Matching dengan metode String Length Reduction (SLR) merupakan metode SLR pendekatan yang digunakan untuk meningkatkan kemudahan dalam pencocokan string. Metode SLR berfokus pada reduksi panjang string yang diproses selama pencocokan string dilakukan. Tujuannya adalah untuk mengurangi jumlah operasi yang harus dilakukan dalam setiap langkah pencocokan, sehingga meningkatkan kinerja algoritma secara keseluruhan.



Gambar 2.1 Deskripsi Algoritma String Matching dengan Metode SLR

2. Pseudo Code

Setelah mengdeskripsikan algoritma string matching, tahap berikutnya adalah menyusun pseudo code atau kode pseudo yang menggambarkan langkah-langkah yang diperlukan dalam implementasi algoritma String Matching dengan metode SLR. Pseudo code akan memberikan suatu panduan yang jelas tentang cara atau langkah-langkah yang harus diambil dalam implementasi algoritma.

4. HASIL DAN PEMBAHASAN

1. Menggunakan Algoritma Knutt Morris Pratt

```
def computeLPS(pattern):
    m = len(pattern)
    lps = [0] * m
    length = 0
    i = 1
    while i < m:
        if pattern[i] == pattern[length]:
            length += 1
            lps[i] = length
            i += 1
        else:
            if length != 0:
                length = lps[length - 1]
            else:
                lps[i] = 0
                i += 1
    return lps

def SLRStringMatch(text, pattern):
    n = len(text)
    m = len(pattern)
    lps = computeLPS(pattern)
    i = 0
    j = 0
    while i < n:
        if pattern[j] == text[i]:
            i += 1
            j += 1
        if j == m:
            print("Pattern found at index", i - j)
            j = lps[j - 1]
        elif i < n and pattern[j] != text[i]:
            if j != 0:
                j = lps[j - 1]
            else:
                i += 1

# Example usage
text = " ABCABBACCBB"
pattern = "ABB"
SLRStringMatch(text, pattern)
```

```
def computeLPS(pattern):
    m = len(pattern)
    lps = [0] * m
    length = 0
    i = 1
    while i < m:
        if pattern[i] == pattern[length]:
            length += 1
            lps[i] = length
            i += 1
        else:
            if length != 0:
                length = lps[length - 1]
            else:
                lps[i] = 0
                i += 1
    return lps
```

```
def SLRStringMatch(text, pattern):
    n = len(text)
    m = len(pattern)
    i = 0
    while i <= n - m:
        j = 0
        while j < m and text[i+j] == pattern[j]:
            j = j + 1
        if j == m:
            print("Pattern found at index", i)
        i = i + 1

# Example usage
text = "ABCABBACCBB"
pattern = "ABB"
SLRStringMatch(text, pattern)
```

```
Pattern found at index 3
```

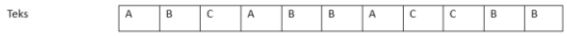
Gambar 4.1 Algoritma Knutt Morris Pratt

Kode yang diberikan adalah implementasi fungsi computeLPS dan SLRStringMatch dalam bahasa Python untuk melakukan pencocokan string menggunakan metode String Length Reduction (SLR) dengan menggunakan algoritma Knuth-Morris-Pratt (KMP).

Pembahasan

Berikut ini merupakan proses algoritma Knutt Morris Pratt untuk mencari kata dalam rangkain kalimat, Adapun caranya sebagai berikut :

1. Diberikan sebuah variabel string T dengan susunan huruf sebagai berikut ini:

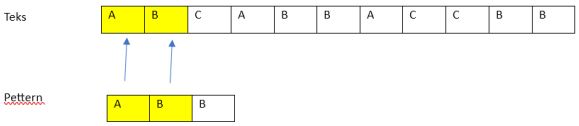


Gambar 4.2 Variabel String T

2. Selanjutnya diberikan variabel H pola dengan kata yang akan dicari dalam variabel T

Gambar 4.3 Variabel Pola H

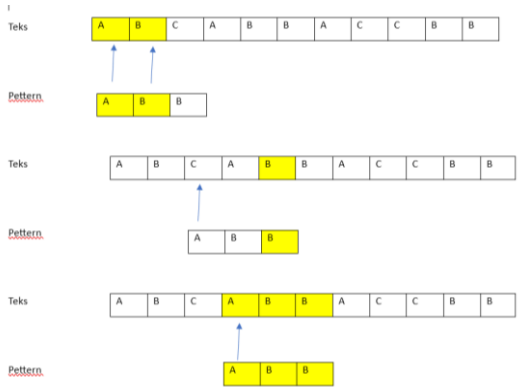
3. Langkah Pertama yang dilakukan bandingkan pola H[1] dengan string T [1] berikut ini hasilnya



Gambar 4.3 Langkah Pertama Algoritma Knutt Morris Pratt

Pola [1, 2] cocok dengan string [1, 2]. Dikarenakan ada kecocokan dengan pola maka algoritma Knutt Morris Pratt akan menyimpan informasi tersebut, kemudian pola akan bergeser satu posisi ke kanan yaitu B.

4. Algoritma Knutt Morris Pratt tidak selalu melakukan pengeseran pola sebanyak satu string ke kanan, Algoritma Knutt Morris Pratt membutuhkan sebuah table (array) yang berisi nilai dari masing-masing pinggiran karakter pada string. Table tersebut dibuat oleh sebuah prosedur pencarian string dengan prosedur yang terpisah.



Gambar 4.3 Langkah Kedua Algoritma Knutt Morris Pratt

2. **Menggunakan Algoritma Brute Force**

```
def SLRStringMatch(text, pattern):
    n = len(text)
    m = len(pattern)
    i = 0
    while i <= n - m:
        j = 0
        while j < m and text[i+j] == pattern[j]:
            j = j + 1
        if j == m:
            print("Pattern found at index", i)
        i = i + 1

# Example usage
text = " ABCABBACCBB"
pattern = "ABB"
SLRStringMatch(text, pattern)
```

```
: def SLRStringMatch(text, pattern):
      n = len(text)
      m = len(pattern)
      i = 0
      while i <= n - m:
          j = 0
          while j < m and text[i+j] == pattern[j]:
              j = j + 1
          if j == m:
              print("Pattern found at index", i)
          i = i + 1

  # Example usage
  text = "ABCABBACCBB"
  pattern = "ABB"
  SLRStringMatch(text, pattern)

  Pattern found at index 3
```

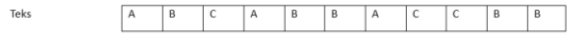
Gambar 4.4 Algoritma Brute Force

Kode tersebut adalah implementasi fungsi SLRStringMatch dalam bahasa Python untuk melakukan pencocokan string menggunakan metode Algoritma Brute Force.

Pembahasan

Berikut ini merupakan proses algoritma Brute Force untuk mencari kata dalam rangkain kalimat, Adapun caranya sebagai berikut :

5. Diberikan sebuah variabel string T dengan susunan huruf sebagai berikut ini:

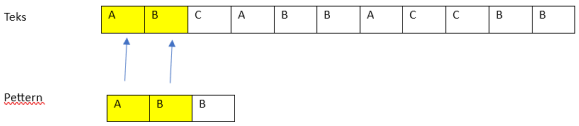


Gambar 4.5 Variabel String T

6. Selanjutnya diberikan variabel H pola dengan kata yang akan dicari dalam variabel T

Gambar 4.6 Variabel Pola H

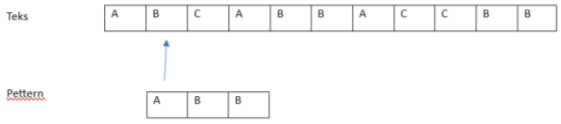
7. Langkah Pertama yang dilakukan bandingkan pola H[1] dengan string T [1] berikut ini hasilnya



Gambar 4.7 Langkah Pertama Algoritma Brute Force

Pola [1, 2] memiliki kesamaan tau kecocokan dengan string [1, 2]. Karena memiliki kecocokan maka algoritma Brute Force akan menyimpan informasi tersebut, kemudian pola tersebut akan bergeser satu posisi ke kanan yaitu B.

8. Pola T[1] dengan string H[2] inilah hasilnya



Gambar 4.8 Langkah kedua Algoritma Brute Force

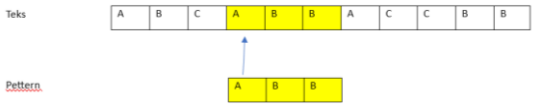
Pola T [1]tidak sesuai dengan string H [2] maka pola akan berpindah satu posisi ke kanan yaitu C.

9. Pola T[1] dengan string H[3] inilah hasilnya

Gambar 4.9 Langkah ketiga Algoritma Brute Force

Pola T [1, 2] tidak sesuai dengan string H [1, 2] tetapi terdapat kecocokan pada pola T [3] dengan string H [5]. Karena ada kecocokan maka algoritma Brute Force akan menyimpan informasi tersebut, kemudian pola akan bergeser satu posisi ke kanan yaitu B.

10. Pola T [1] dengan string H [5] inilah hasilnya



Gambar 4.10 Langkah keempat Algoritma Brute Force

Dengan menggunakan algoritma Brute Force, Pola T [1, 2, 3] akan dicocokan dengan string H [4, 5, 6]. Karena memiliki kecocokan maka algoritma Brute Force akan menyimpan informasi tersebut. Dan pola tidak akan bergeser dan melanjutkan ke pencocokan pola T [4] namun karena jumlah pola hanya 3 huruf maka pencarian akan berhenti dan akan menghasilkan pola T cocok dengan string H sebesar 100 persen.

Dalam contoh ini, fungsi SLRStringMatch akan mencari pola string "ABB"dalam teks yang diberikan dan mencetak pesan jika pola ditemukan.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dalam penelitian ini, algoritma pencocokan string dianalisis dan diterapkan menggunakan metode pengurangan panjang string (SLR) untuk mencocokkan data string. Beberapa algoritma yang digunakan dalam metode ini termasuk Brute Force, Knuth-Morris-Pratt (KMP), dan Boyer-Moore. Dengan metode SLR, algoritma pencocokan string dapat ditingkatkan efisiensi dengan mengurangi jumlah operasi yang dilakukan dalam setiap langkah pencocokan.

Hasil penelitian menunjukkan bahwa algoritma KMP dengan metode SLR memiliki kecepatan pencocokan yang lebih cepat daripada pendekatan Brute Force. Ini terutama berlaku untuk teks dengan ukuran yang besar atau pola string yang panjang. Pergeseran yang cerdas berdasarkan informasi LPS

(Longest Proper Prefix which is also Suffix) memungkinkan untuk melewati banyak karakter dalam teks tanpa melakukan perbandingan yang tidak perlu.

5.2 Saran

Berdasarkan penelitian ini, berikut beberapa saran untuk pengembangan lebih lanjut:

1. Disarankan untuk melakukan penelitian lebih lanjut terhadap algoritma seperti Rabin-Karp, Aho-Corasick, dan Shift-Or.
2. Untuk menyelesaikan masalah tertentu di bidang tertentu, algoritma penggabungan string dapat diubah dan disesuaikan.
3. Dapat dilakukan analisis performa yang mendalam untuk membandingkan kecepatan dan efisiensi antara berbagai algoritma penggabungan string dengan metode SLR.

6. DAFTAR PUSTAKA

[1]Akhtar Rasool, Nilay Khare. 2012. "Parallelization of KMP String Matching Algorithm on Different SIMD architectures: Multi-Core and GPGPU's". International Journal of Computer Applications., Vol 49-No.11.

[2] Azizah Abd Manaf, Akram Zeki, Mazdak Zamani, Suriayati Chuprat, Eyas El-Qawa-smeh. 2011. "Informatics Engineering and Information Science". Part II, Springer Heidelberg Dordrecht, New York.

[3] Charles Barutu, Naufal Abdi. 2020."Brute Force Algorithm Implememtation Of Dictionary Search". Informatika dan Sains, Volume 10.

[4] D. Danuri, "Pencarian File Teks Berbasis Content dengan Pencocokan String Menggunakan Algoritma Brute force," Sci. J. Informatics, vol. 3, no. 1, pp. 68–75, 2016, doi: 10.15294/sji.v3i1.6515.

[5] Dana Shapira, Ajay Daptardar. 2006. "Adapting the Knuth-Morris-Pratt Algorithm for patterb matching in Huffman encoded texts". Information Processing and Management.

[6] Garima Pandey, Mamta Martolia, dkk. 2017. "A Novel String Matching Algorithm and Comparison with KMP Algorithm". International Journal of Computer Applications., Vol 179 - No.3.

[7] H. Fernando and A. Boyer-, "Perbandingan dan Pengujian Beberapa Algoritma Pencocokan String," no. 10, 2009.

[8] Koloud Al-Khamaiseh, Shadi ALShagarin. 2014. "A Survey of String Matching Algorithms". Journal of Engineering Research and Applications., Vol.4, Issue 7.

[9] Lusiana dan Melva Suryani, "Metode SLR untuk Mengidentifikasi Isu-Isu dalam Software Engineering," J. SATIN -Sains dan Teknol. Inf., vol. 3, no. 1, hal. 1–11, 2014.

[10] Mohamed F.Ababneh, Saleh Oqeli, dkk. 2006. "Occurrences Algorithm for string Searching Based on Brute-force Algorithm". Journal of Computer Science.

[11] Nimisha Singla, Deepak Garg. 2012. "String Matching Algorithms and their Applicability in various Applications". Internasional Journal of Soft Computing and Engineering(IJSCE), Vol-I, Issue-6.

[12] Pooja Manisha Rahate, M. B. Chandak. 2018. "Comparative Study of String Matching Algorithms for DNA dataset". International Journal of Computer Sciences and Engineering, vol-6, issue-5.

[13] P. P. Borah, G. Talukdar. 2013. "A Comparison of String Matching Algorithms-Boyer-Moore Algorithm and Brute-Force Algorithm".

[14] Preeti Narooka. 2018. "String Matching Algorithms". International Journal Of Engineering And Computer Science, Vol 7 Issue 3

[15] Robbi Rahim. 2017. "A review: search visualization with Knuth Morris Pratt algorithm". IOP Conference Series : Materials Science and Engineering.

[15] R Yu Tsarev, A S Chernigovskiy, dkk. 2016. "Combined string searching algorithm

based on knuth-morris-pratt and boyer-more algorithms". XIX International Scientific Conference Reshetnev Readings

[16] S. Ahn, H. Hong, H. Kim, J. H. Ahn, D. Baek, and S. Kang, "A hardware-efficent multi-character string matching architecture using brute-force algorithm," 2009 Int. SoC Des. Conf. ISOCC 2009, no. December 2009, pp. 464–467, 2009, doi: 10.1109/SOCDC.2009.5423922.

[17] Shivendra Kumar Pandey, Neeraj Kumar, dkk. 2014."A Study on String Matching Methodogies". International Journal of Computer Science and Information Technology, Vol 5(3).

[18] Y. Daniel Liang, Weitian Tong. 2020. "Usung Animations to Tech String Matching Effectively". The Journal of Computing Sciences in Colleges., Vol 35, Number 10.