BRUTE-FORCE STRING MATCHING

PERANCANGAN DAN ANALISIS ALGORITMA

disusun Untuk Memenuhi Tugas Mata Kuliah Perancangan Dan Analisis Algoritma

Dosen pengampu :

Randi Proska Sandra, S.Pd, M.Sc

Disusun Oleh :

Nama : Fatahul Fahmi

Nim : 23343005

Group : H

PRODI INFORMATIKA

DEPARTEMEN ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2025

1. PENJELASAN PROGRAM/ALGORITMA

1.1 Pengertian Brute-Force String Matching

Algoritma Brute Force adalah algoritma yang melakukan pencocokan sebuah pattern terhadap sebuah teks an- tara 0 dan n-m untuk menemukan keberadaan sebuah pattern pada sebuah teks. Dalam sebuah metode string match- ing terdapat dua istilah yaitu pattern dan teks. Pattern adalah sebuah kata yang dimasukan untuk dicocokan pada sebuah teks. (Azis et al., 2021)

Untuk melakukan pencocokan pattern pada sebuah teks yang dimulai dari karakter awal hingga karakter terakhir, ketika melakukan pengecekan pada karakter pertama dan tidak ditemukan kecocokan maka proses shift dilakukan yaitu proses pemindahan string ke karakter selanjutnya yang berada dikanan, proses ini dilakukan ke karakter selanjutnya sampai dengan ditemukan kecocokan pattern pada sebuah teks.

Algoritma brute force juga memiliki kelebihan dan kelemahan. Adapun kelebihan dari algoritma brute force yaitu :

a. Algoritma brute force dapat digunakan untuk memecahkan hampir sebagian besar masalah.
b. Algoritma brute force sederhana dan mudah dimengerti
c. Algoritma brute force menghasilkan algoritma yang layak untuk beberapa masalah penting seperti pencarian,
d. pengurutan, pencocokkan string, atau perkalian matriks.(Mukaromah et al., 2021)

Selain kelebihannya, algoritma brute force juga memiliki beberapa kelemahan. Salah satunya adalah efisiensi waktu yang kurang optimal karena algoritma ini melakukan pengecekan satu per satu terhadap setiap kemungkinan yang ada.

Hal ini menyebabkan waktu eksekusi menjadi lebih lama, terutama ketika bekerja dengan data yang besar. Selain itu, algoritma brute force tidak selalu menjadi pilihan terbaik dalam menyelesaikan permasalahan yang memiliki banyak kemungkinan solusi, karena pendekatannya yang tidak memanfaatkan strategi optimasi tertentu. Oleh karena itu, dalam beberapa kasus, metode yang lebih canggih seperti algoritma heuristik atau algoritma berbasis pencarian lebih disarankan untuk meningkatkan efisiensi pencarian solusi.

1.2 Penjelasan Langkah Pada Program

pola akan dibandingkan dengan karakter awal dalam teks. Jika pola tidak cocok dengan karakter yang sedang diperiksa, maka pola akan bergeser satu posisi ke kanan dan proses pencocokan diulang. Pergeseran ini terus dilakukan hingga seluruh teks telah diperiksa atau pola berhasil ditemukan dalam teks, proses akan ditampilkan secara visual baik dengan animasi berbentuk screenshot dan visual secara CLI

1. Pada langkah pertama pattern belum cocok dengan karakter pada text, berikut nya pattern akan bergeser 1 kearah kanan

```
Teks:    R a m d h a n B u l a n B e r k a h

Pola:    B u l a n
```

2. Selanjutnya pattern masih belum menemukan karakter yang cocok dengan pola maka bergeser 1 kearag kanan lihat perubahan pergeseran ditandai dengan petak bewarna hijau

```
Teks:    R a m d h a n B u l a n B e r k a h

Pola:      B u l a n
```

3. Pattern masih belum menemukan dan akan terus bergeser 1 ke kanan hingga menemukan pola yang cocok

```
Teks:    R a m d h a n B u l a n B e r k a h

Pola:        B u l a n
```

4. Dilangkah ke-4 pun pattern masih belum menemukan char pada text yang cocok dengan pola yang sudah di tentukan

```
Teks:    R a m d h a n B u l a n B e r k a h

Pola:          B u l a n
```

5. Pada langkah ke-5 kita percepat hingga pattern menemukan char yang cocok pada text, tetapi perlu kita lihat bahwa ketika pattern telah menemukan char yang sesuai dengan pola proses tidak berakhir dan terus berlanjut hingga akhir char pada text

```
Teks:    R a m d h a n B u l a n B e r k a h

Pola:                  B u l a n
```

```
Teks:    R a m d h a n B u l a n B e r k a h

Pola:                  B u l a n
```

6. Kita lihat bahwa proses tetap berlanjut hingga akhir dari karakter pada text, dapat kita lihat bahwa pattern berubah menjadi merah kembali yang menandakan pola tidak cocok dengan char pada text

```
Teks:    R a m d h a n B u l a n B e r k a h

Pola:                    B u l a n
```

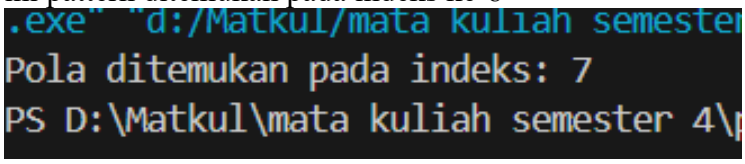
7. Proses terus berlanjut hingga akhir dari karakter pada text

```
Teks:    R a m d h a n B u l a n B e r k a h

Pola:                      B u l a n
```

8. Kita percepat sehingga program selesai pada akhir dari char pada text

```
Teks:    R a m d h a n B u l a n B e r k a h

Pola:                                B u l a n
```

9. Jika kita lihat visual nya secara CLI maka akan mengeluarkan output pada indeks keberapa pattern ditemukan, pada contoh kali ini pattern ditemukan pada indeks ke-8



2. PSEUDOCODE

Program algoritma brute-force string matching
Input: string teks dengan panjang panjang_teks dan string pola dengan panjang panjang_pola
Output: Indeks dari karakter pertama dalam teks yang menjadi awal substring yang cocok atau −1 jika pencocokan tidak ditemukan

for i ← 0 **to** panjang_teks – panjang_pola **do**
    j ← 0
    while j < panjang_pola **and** pola [j] = teks [i + j] **do**
        j ← j + 1
    if j = panjang_pola **return** i
return −1

3. SOURCE CODE

```
def pencocokan_bruteforce(teks, pola):
    panjang_teks = len(teks)
    panjang_pola = len(pola)

    for i in range(panjang_teks - panjang_pola + 1):
        j = 0
        while j < panjang_pola and pola[j] == teks[i + j]:
            j += 1

        if j == panjang_pola:
            return i

    return -1

teks = "RamdhanBulanBerkah"
pola = "Bulan"
hasil = pencocokan_bruteforce(teks, pola)
print("Pola ditemukan pada indeks:", hasil)
```

4. ANALISIS KEBUTUHAN WAKTU

4.1 Analisis menyeluruh

Operasi penugasan:

- i ← 0 → **1x**
- i ≤ (panjang_teks - panjang_pola) → **nx**
- i ← i + 1 → **nx**
- j ← 0 → **1x**
- teks[i + j] == pola[j] → **nx**
- j ← j + 1 → **nx**
- j == panjang_pola → **1x**
- Ketemu ← True → **1x**
- Ketemu ← False → **1x**

Jumlah seluruh operasi pengisian nilai:
T1 = 1 + n + n + 1 + n + n + 1 + 1 = 5 + 4n

Operasi Penjumlahan:

- i ← i + 1 → nx
- j ← j + 1 → nx
- teks[i + j] → nx

Jumlah seluruh operasi penjumlahan:
T2 = 3n

Operasi Pengurangan:

- (panjang_teks - panjang_pola) → nx

Jumlah seluruh operasi pengurangan:
T3 = n

Dengan demikian, kompleksitas waktu algoritma dihitung berdasarkan jumlah operasi aritmetika dan operasi pengisian nilai:
T(n) = T1 + T2 + T3 = 5 + 4n + 3n + n = 5 + 8n
Jadi, kompleksitas waktu dalam notasi Big-O adalah: O(nm)

4.2 Analisis Kompleksitas Berdasarkan Jumlah Operasi Abstrak

Kompleksitas waktu didasarkan pada jumlah iterasi perulangan i dan j, dengan maksimal perbandingan n * m pada kondisi terburuk. Jumlah operasi abstrak:

$$T(n) = O(nm)$$

4.3 Analisis Best-Case, Worst-Case, dan Average-Case

Tmax(n) - Worst Case (Kasus Terburuk)
Terjadi ketika tidak ada kecocokan sama sekali, atau setiap iterasi membandingkan seluruh karakter tetapi tidak pernah menemukan pola.
Kompleksitas waktu maksimal terjadi, yaitu:

$$O(nm)$$

Tmin(n) - Best Case (Kasus Terbaik)
Terjadi jika pola ditemukan pada indeks pertama dalam teks.
Hanya satu kali iterasi penuh diperlukan untuk mencocokkan seluruh karakter pola dengan teks.
Kompleksitas waktunya adalah:

$$O(m)$$

Tavg(n) - Average Case (Kasus Rata-Rata)
Terjadi ketika pola muncul di beberapa tempat dalam teks tetapi tidak selalu cocok secara penuh dalam satu perbandingan.
Secara rata-rata, pencocokan terjadi sekitar separuh panjang teks.
Kompleksitas waktu rata-rata:

$$O(nm / 2) = O(nm)$$

5. REFERENSI

Azis, M. R., Fitri, I., & Rahman, B. (2021). Penggunaan Algoritma Brute Force String Matching Dalam Pencarian Orang Hilang Pada Website Temukandia.Com. *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, *6*(2), 205–212. https://doi.org/10.29100/jipi.v6i2.1979

Greenfield, T., Goodman, S. E., & Hedetniemi, S. T. (1979). Introduction to the Design and Analysis of Algorithms. In *Applied Statistics* (Vol. 28, Issue 1). https://doi.org/10.2307/2346822

Mukaromah, I. A., Jamil, A., Saputro, M. W., Farikha, N., Studi, P., Informatika, T., Muhammadiyah, S., Brebes, P., & Informasi, S. (2021). Analisis Pencocokan String Menggunakan Algoritma Brute Force. *Jurnal Teknik Informatika Dan Sistem Informasi (JURTISI)*, *1*(1), 18–22.

6. LAMPIRAN LINK GITHUB

https://github.com/bangpami/Perancangan-AnalisisAlgoritma.git