Tugas ke :

Mata Kuliah : Analisis Algoritma

**Tugas Pattern Matching**

****

### Disusun Oleh :

### NAMA : Moch Achmar

**STB : 222362**

**KELAS : K**

**UNIVERS1TAS DIPA MAKASSAR**

**MAKASSAR**

**2023**

Hal : \_\_\_

Hal : \_\_\_

1. Lakukan analisis algoritma pattern matching untuk string sebagai berikut:  
   Text = SAYA SEDANG BELAJAR ALGORITMA GENETIKA

Pattern = ALGORITMA

Selesaikan:  
a. Brute force

b. Knuth-Morris-Pratt

c. Booyer Moore

d. Rabin-Karp

1. Penyelesaian yang sama seperti No. 1

Text = MOCH ACHMAR

Pattern = ACHMAR

1. Selesaikan analisis pola string berikut:

Text = ABACADABABCABA

Pattern = ABA

Selesaikan:

1. Brute force
2. Knuth-Morris-Pratt

Jawab:

1. a. Brute force



Kompleksitas waktu pada algoritma pencocokan pola brute force tergantung pada panjang teks (n) dan panjang pattern (m). Pada kasus terburuk, kompleksitas waktu adalah O((n - m + 1) \* m), di mana n adalah panjang teks dan m adalah panjang pattern (m < n).

Jumlah pergeseran yang dilakukan dalam algoritma ini adalah (n - m + 1). Setiap pergeseran memerlukan perbandingan karakter antara pola dan teks pada posisi yang bersesuaian. Jumlah perbandingan dalam setiap pergeseran adalah sebanyak m kali perbandingan.

Kompleksitas waktu pada kasus ini adalah O((35 - 8 + 1) \* 8) = O(28 \* 8) = O(224).

Jumlah pergeseran = (35 - 8 + 1) = 28.

Setiap pergeseran memerlukan perbandingan sebanyak 8 kali pergeseran.

Algoritma pencocokan pola brute force ini memiliki kompleksitas waktu yang tinggi, terutama jika panjang teks dan panjang patternnya besar.

1. Knuth-Morris-Pratt



Menghitung fungsi pinggiran : O(m)

Pencarian String : O(n)

Kompleksitas total : O(m+n)

Keuntungan : Tidak pernah bergerak “mundur” (mengulang pemeriksaan) seperti Brute Force dan cocok untuk memproses file yang ukuran lebih besar atau streaming

Kerugian : Ketika variasi karakter teks nya beragam, akan lebih sering mismatch seperti Brute Force dan Algoritma ini tidak memperhitungkan apakah karakter yang mismatch

1. Booyer Moore



Menghitung fungsi L(x) : O(A)

Pencarian String kasus terburuk : O(nm)

Kompleksitas total untuk kasus terburuk: O(nm+A)

Keuntungan : Lebih cepat dibandingkan Brute Force untuk teks dengan variasi karakter yang sangat beragam (A besar) atau karakter dalam Bahasa Inggris

Kerugian : Lambat untuk teks dengan variasi karakter tidak beragam (karakter binary)

1. Rabin-Karp



Kasus waktu yang terjadi saat menggunakan algoritma Rabin-Karp bervariasi tergantung pada faktor-faktor seperti panjang teks, panjang pola, dan distribusi karakter dalam teks. Secara umum, algoritma Rabin-Karp memiliki kompleksitas waktu rata-rata O(n+m), di mana n adalah panjang teks dan m adalah panjang pola.

1. a. Brute force



Kompleksitas waktu pada algoritma pencocokan pola brute force tergantung pada panjang teks (n) dan panjang pattern (m). Pada kasus terburuk, kompleksitas waktu adalah O((n - m + 1) \* m), di mana n adalah panjang teks dan m adalah panjang pattern (m < n).

Jumlah pergeseran yang dilakukan dalam algoritma ini adalah (n - m + 1). Setiap pergeseran memerlukan perbandingan karakter antara pola dan teks pada posisi yang bersesuaian. Jumlah perbandingan dalam setiap pergeseran adalah sebanyak m kali perbandingan.

Panjang teks (n) = 11

Panjang pattern (m) = 6

Sehingga, kompleksitas waktu pada kasus ini adalah O((11 - 6 + 1) \* 6) = O(6 \* 6) = O(36).

Jumlah pergeseran = (11 - 6 + 1) = 6.

Setiap pergeseran memerlukan perbandingan sebanyak 6 kali.

Algoritma pencocokan pola brute force ini memiliki kompleksitas waktu yang tinggi, terutama jika panjang teks dan panjang patternnya besar.

b. Knuth-Morris-Pratt



Menghitung fungsi pinggiran : O(m)

Pencarian String : O(n)

Kompleksitas total : O(m+n)

1. Booyer Moore



Menghitung fungsi L(x) : O(A)

Pencarian String kasus terburuk : O(nm)

Kompleksitas waktu untuk kasus rata-rata: O(n/m)

Keuntungan : Lebih cepat dibandingkan Brute Force untuk teks dengan variasi karakter yang sangat beragam (A besar) atau karakter dalam Bahasa Inggris

Kerugian : Lambat untuk teks dengan variasi karakter tidak beragam (karakter binary)

1. Rabin-Karp



Jumlah pergeseran yang terjadi pada algoritma Rabin-Karp tergantung pada posisi kecocokan atau ketidakkocokan antara jendela dan pola. Jika tidak ada kecocokan, jendela akan digeser satu karakter ke kanan, sehingga jumlah pergeseran adalah 1. Jika terdapat kecocokan parsial, kita masih harus memindahkan jendela ke posisi berikutnya untuk mencari kemungkinan kecocokan yang lebih lanjut.

Dalam kasus terburuk, ketika tidak ada kecocokan antara jendela dan pola, algoritma Rabin-Karp akan memindahkan jendela sebanyak (n - m + 1) kali, di mana n adalah panjang teks dan m adalah panjang pola. Jumlah pergeseran ini mencakup semua kemungkinan posisi jendela yang memungkinkan di atas teks.

Namun, perlu diperhatikan bahwa algoritma Rabin-Karp memiliki mekanisme deteksi dini (early detection) yang memungkinkan kita untuk menghentikan pencarian jika hash dari jendela tidak cocok dengan hash dari pola. Hal ini mengurangi jumlah pergeseran yang sebenarnya diperlukan dalam beberapa kasus, tergantung pada distribusi karakter dalam teks.

Jadi, dalam kasus Anda, jumlah pergeseran pada algoritma Rabin-Karp dapat bervariasi tergantung pada posisi kecocokan atau ketidakkocokan antara jendela dan pola, dengan jumlah maksimum pergeseran adalah (n - m + 1) di kasus terburuk.

1. a. Brute force



Kompleksitas waktu pada algoritma pencocokan pola brute force di atas adalah kasus terburuk, yaitu O((n - m + 1) \* m), di mana n adalah panjang teks dan m adalah panjang pattern (m < n).

Jumlah pergeseran yang dilakukan dalam algoritma ini adalah (n - m + 1). Algoritma brute force mencoba memindahkan pattern secara bergeser satu karakter ke kanan pada setiap langkah untuk mencari kemungkinan kecocokan dengan teks.

Setiap pergeseran memerlukan perbandingan karakter antara pola dan teks pada posisi yang bersesuaian. Jumlah perbandingan dalam setiap pergeseran adalah sebanyak m kali perbandingan. Dalam kasus ini, panjang pattern adalah m = 3, sehingga setiap pergeseran memerlukan 3 perbandingan karakter.

Jumlah pergeseran yang dilakukan adalah (n - m + 1) = (14 - 3 + 1) = 12. Setiap pergeseran memerlukan 3 perbandingan karakter antara pola dan teks.

Algoritma pencocokan pola brute force ini, kompleksitas waktu adalah O((n - m + 1) \* m) dan jumlah pergeseran yang dilakukan adalah 12. Setiap pergeseran memerlukan 3 perbandingan karakter.

b. Knuth-Morris-Pratt



Menghitung fungsi pinggiran : O(m)

Pencarian String : O(n)

Kompleksitas total : O(m+n)