Implementasi Algoritma Pattern Matching Sebagai Alat Deteksi Mutasi Gen CFTR pada Penyakit Cystic Fibrosis

Debrina Veisha Rashika Wijayanto - 13522025
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : debrina.veisha@gmail.com

***Abstrak*— Cystic Fibrosis (CF) adalah penyakit genetik yang disebabkan oleh mutasi pada gen Cystic Fibrosis Transmembrane Conductance Regulator (CFTR). Deteksi dini mutasi pada gen CFTR penting untuk pengembangan terapi yang efektif. Penelitian ini mengimplementasikan tiga algoritma pattern matching, yaitu Brute Force, Knuth-Morris-Pratt (KMP), dan Boyer-Moore, untuk mendeteksi mutasi pada sekuens CFTR. Studi ini mengevaluasi efisiensi algoritma berdasarkan jumlah perbandingan karakter dan waktu eksekusi. Hasil menunjukkan bahwa algoritma Boyer-Moore paling efisien dalam mendeteksi mutasi dengan jumlah perbandingan karakter dan waktu eksekusi yang lebih rendah dibandingkan Brute Force dan KMP. Implementasi ini memudahkan peneliti dalam menganalisis mutasi CFTR, membantu dalam diagnosis dan pengembangan terapi yang tepat.**

Kata Kunci—Cystic Fibrosis, CFTR, pattern matching, Brute Force, Knuth-Morris-Pratt, Boyer-Moore.

I. PENDAHULUAN

Cystic Fibrosis (CF) adalah penyakit genetik yang serius dan dapat mengancam nyawa, mempengaruhi sistem pernapasan dan pencernaan. Penyebab utama dari CF adalah mutasi pada gen Cystic Fibrosis Transmembrane Conductance Regulator (CFTR). Gen CFTR bertanggung jawab untuk memproduksi protein yang mengatur transportasi ion klorida dan natrium melintasi membran sel epitel. Ketika fungsi CFTR terganggu, terjadi akumulasi lendir kental di organ-organ seperti paru-paru dan pankreas, yang menyebabkan infeksi berulang, kerusakan organ, dan berbagai komplikasi lainnya.

Deteksi dini mutasi CFTR sangat penting untuk diagnosis dan manajemen CF. Mengidentifikasi tipe mutasi spesifik dapat membantu dokter menentukan terapi yang paling efektif. Misalnya, obat-obatan tertentu seperti Kalydeco® dirancang untuk mengatasi mutasi tertentu pada gen CFTR. Oleh karena itu, teknik yang cepat dan akurat untuk mendeteksi mutasi sangat diperlukan dalam praktik klinis dan penelitian genetik.

Pattern matching adalah salah satu teknik yang dapat digunakan untuk mendeteksi mutasi dalam sekuens DNA. Teknik ini membandingkan urutan genetik yang diketahui dengan urutan target untuk menemukan kesesuaian atau perbedaan. Dalam penelitian ini, kami mengimplementasikan tiga algoritma pattern matching yang populer: Brute Force, Knuth-Morris-Pratt (KMP), dan Boyer-Moore. Ketiga algoritma ini dipilih karena memiliki karakteristik yang berbeda dalam hal kecepatan dan efisiensi. Penelitian ini bertujuan untuk mengevaluasi dan membandingkan kinerja masing-masing algoritma dalam mendeteksi lima mutasi umum pada gen CFTR: ΔF508, G542X, G551D, N1303K, dan W1282X.

II. TEORI DASAR

A. Pattern Matching

Pattern Matching adalah mekanisme untuk memeriksa apakah terdapat suatu nilai dari suatu pola dan jika nilai ada pada pola tersebut, maka informasi yang ada dapat diesktrak [1]. Dalam pemrograman fungsional, pencocokan pola memberikan alternatif yang ringkas dan deklaratif terhadap pernyataan kondisional, memungkinkan untuk menentukan serangkaian pola dan menentukan tindakan untuk setiap pola. sehingga menghasilkan kode yang tidak hanya lebih mudah dibaca tetapi juga lebih mudah dipelihara dan tidak terlalu rawan kesalahan.

Pada *pattern matching* dilakukan proses mencari kemunculan atau kesesuaian dari suatu pola dalam teks atau data untuk mengidentifikasi kecocokan dengan bagian-bagian dari teks atau data yang sedang dianalisis. Dalam proses ini, dilakukan pembandingan antara pola yang ditentukan dengan bagian-bagian dari teks secara sistematis. Pola ini dapat berupa urutan karakter, pola grafis, pola numerik, atau kombinasi dari semuanya. Teknik pattern matching yang tepat memungkinkan untuk menemukan kejadian atau entitas yang sesuai dengan pola yang dicari, bahkan jika pola tersebut tersembunyi dalam teks panjang atau data kompleks.

*Pattern matchin*g memiliki berbagai aplikasi dalam berbagai bidang seperti ilmu komputer, bioinformatika, pengolahan gambar, analisis data, dan lain-lain. Contoh penggunaannya meliputi pencarian kata kunci dalam teks, identifikasi pola genetik dalam bioinformatika, pencocokan objek pada gambar, analisis pola dalam data statistik untuk

mengidentifikasi tren atau anomali, serta pencarian pola dalam aliran data untuk deteksi dini kejadian penting.

Untuk mencapai pencocokan pola yang efektif, digunakan berbagai algoritma dan metode seperti algoritma Brute-Force, Boyer-Moore, Knuth-Morris-Pratt, Rabin-Karp, Aho-Corasick, dan lainnya. Pemilihan algoritma yang tepat bergantung pada sifat pola yang dicari dan kompleksitas teks atau data yang dianalisis. Secara praktis, pattern matching menjadi dasar untuk berbagai aplikasi seperti pencarian teks, pemrosesan bahasa alami, pengenalan pola, dan sistem rekomendasi.

B. Algoritma Brute Force

Algoritma *brute force* adalah teknik yang memeriksa semua kemungkinan solusi untuk mencapai tujuan tertentu. Algoritma *brute force* menggunakan pendekatan yang lempang (*straightforward*) untuk memecahkan suatu persoalan. Algoritma ini akan mencoba setiap langkah atau kombinasi yang mungkin untuk menemukan solusi yang diinginkan.

Dalam *pattern matching*, algoritma brute force adalah metode dasar yang digunakan untuk menemukan pola tertentu dalam teks atau string. Cara kerjanya adalah dengan membandingkan pola tersebut dengan setiap kemungkinan substring dalam teks secara berurutan.

Algoritma ini bekerja dengan mencoba setiap kemungkinan substring yang sesuai dengan panjang pola. Kompleksitas waktu algoritma ini adalah O(m * n), di mana m adalah panjang pola dan n adalah panjang teks. Dalam scenario terbaik algoritma ini memiliki kompleksitas waktu O(n) dan terjadi apabila karakter pertama pada pola tidak pernah sama dengan karakter pada teks yang dicocokkan. Dalam skenario terburuk, algoritma ini harus membandingkan pola dengan setiap karakter dalam teks dengan kompleksitas waktu O(mn) [2].

C. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) adalah metode pencarian pola yang dikembangkan oleh Donald Knuth, James H. Morris, dan Vaughan Pratt pada tahun 1977, algoritma ini melakukan pencarian pada teks dari kiri ke kanan dengan memanfaatkan informasi yang sudah diketahui tentang pola untuk menghindari perbandingan yang tidak perlu saat mencocokkan pola dengan teks. Pendekatan ini menggunakan tabel fungsi pinggiran yang didapat dengan mencari jumlah kemunculan pola sufiks pada pola yang sesuai dengan prefiks dari pola tersebut. Hal ini memungkinkan pencocokan pola dilakukan dengan lebih cepat. Berikut ilustrasi dari fungsi pinggiran:

TABLE I. INDEKS POLA

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

TABLE II. FUNGSI PINGGIRAN

k	0	1	2	3	4
b(k)	0	0	1	1	2

Kelebihan utama dari algoritma KMP adalah kemampuannya untuk menghindari perbandingan ulang yang tidak perlu. Dengan menggunakan tabel fungsi pinggiran yang telah dibuat dengan cermat, algoritma ini dapat mencapai kompleksitas waktu O(n + m), di mana n adalah panjang teks dan m adalah panjang pola.

Algoritma KMP paling efisien ketika pola yang dicari memiliki sedikit atau tidak ada karakter yang berulang. Dalam situasi ini, pencocokan pola bisa dilakukan dalam satu kali lintasan teks, dan algoritma KMP mencapai kompleksitas waktu terbaiknya, yaitu O(n). Keunggulan utama KMP adalah efisiensinya dalam melakukan pencocokan tanpa memeriksa ulang karakter yang sudah diperiksa. Namun, KMP kurang efisien saat ukuran pola sangat panjang dan bervariasi.

Oleh karena itu, algoritma KMP sangat berguna dalam berbagai aplikasi yang membutuhkan pencarian pola dalam teks, seperti pemrosesan string, pengenalan pola, kompresi data, dan lain sebagainya.

D. Algoritma Boyer-Moore

Algoritma Boyer-Moore (BM) adalah algoritma untuk mencocokkan pola yang dikembangkan oleh Robert S. Boyer dan J Strother Moore pada tahun 1977. Algoritma ini bekerja dengan mencocokkan pola dari akhir ke awal dalam teks dan menggeser pola berdasarkan informasi dari karakter yang tidak cocok. Algoritma Boyer-Moore menggunakan dua teknik utama:

1. *The looking glass technique*, yaitu mencocokkan karakter pada pola dari belakang ke depan
2. *The character-jump technique*, saat terjadi ketidakcocokan antara karakter x di teks T[i] dengan karakter y di pola P[j], terdapat tiga langkah yang dilakukan secara berurutan:
   a. Jika karakter x ada di sebelah kiri posisi pola P, geser pola sehingga karakter x terakhir dalam pola sejajar dengan karakter x di teks.
   b. Jika tidak memungkinkan untuk menggeser ke karakter x terakhir dan ada karakter x di sebelah kanan posisi ketidakcocokan dalam pola P, geser pola ke kanan satu karakter.
   c. Jika karakter x tidak ada dalam pola P, sejajarkan karakter pertama pola P dengan karakter berikutnya di teks, yaitu T[i+1].

Pada Algoritma Boyer-Moore digunakan fungsi *Last Occurrence* untuk menandakan lokasi indeks terakhir pada pola yang nantinya digunakan pada proses pergeseran. Contoh dari *Last Occurrence* pada pola ‘abacab’ sebagai berikut:

TABLE III. LAST OCCURENCE

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

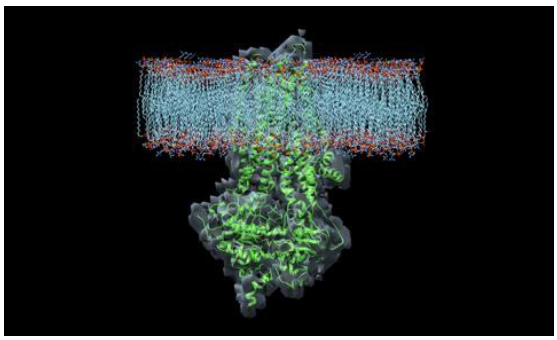
Algoritma BM lebih efisien untuk pola yang panjang. Ketika pola memiliki banyak karakter yang tidak cocok dengan teks, algoritma ini bisa melakukan pergeseran besar dan mengurangi jumlah perbandingan yang perlu dilakukan. Ini membuatnya sangat cocok untuk mencari pola panjang dalam teks yang besar. Kompleksitas waktu terbaik dari algoritma Boyer-Moore adalah O(n/m), di mana n adalah panjang teks dan m adalah panjang pola, terjadi ketika pola tidak cocok dengan teks. Namun, algoritma ini kurang cocok digunakan jika pola yang dicari pendek atau kurang bervariasi seperti bilangan biner.

E. Cystic Fibrosis Transmembrane Conductance Regulator (CFTR) Protein

Cystic Fibrosis Transmembrane Conductance Regulator (CFTR) adalah protein yang membantu menjaga keseimbangan garam dan air di berbagai permukaan tubuh, seperti permukaan paru-paru. Ketika protein ini tidak berfungsi dengan benar, klorida (komponen dari garam) akan terjebak di dalam sel. Tanpa pergerakan klorida yang benar, air tidak bisa menghidrasi permukaan sel. Hal ini membuat lendir yang menutupi sel menjadi kental dan lengket, menyebabkan banyak gejala yang terkait dengan fibrosis kistik [3].

Protein adalah mesin kecil yang melakukan tugas-tugas spesifik dalam sel. Instruksi untuk membangun setiap protein tertulis dalam DNA. Protein dirakit dari blok bangunan yang disebut asam amino. Ada 20 jenis asam amino yang berbeda. Semua protein terdiri dari rantai asam amino ini yang terhubung dalam urutan berbeda, seperti kata-kata berbeda yang ditulis menggunakan 26 huruf alfabet. Instruksi DNA memberi tahu sel asam amino mana yang digunakan di setiap posisi dalam rantai untuk membuat protein tertentu.

Protein CFTR terdiri dari 1.480 asam amino. Setelah rantai protein CFTR dibuat, ia dilipat menjadi bentuk 3-D yang spesifik. Protein CFTR berbentuk seperti tabung yang melewati membran sel.



Gambar 2.1. Struktur CFTR

(Sumber: www.cff.org)

Protein CFTR adalah jenis protein khusus yang disebut saluran ion. Saluran ion memindahkan atom atau molekul yang memiliki muatan listrik dari dalam sel ke luar, atau dari luar sel ke dalam. Di paru-paru, saluran ion CFTR memindahkan ion klorida dari dalam sel ke luar sel. Untuk keluar dari sel, ion klorida bergerak melalui tengah tabung yang dibentuk oleh protein CFTR.

Pada orang dengan fibrosis kistik, mutasi pada gen CFTR dapat menyebabkan masalah berikut dengan protein CFTR, seperti CFTR tidak berfungsi dengan baik, CFTR tidak diproduksi dalam jumlah yang cukup, dan CFTR tidak diproduksi sama sekali.

Ketika salah satu dari masalah ini terjadi, ion klorida terjebak di dalam sel, dan air tidak lagi tertarik ke ruang di luar sel. Ketika ada lebih sedikit air di luar sel, lendir di saluran napas menjadi kering dan mengental, menyebabkan silia menjadi tertekan. Silia tidak bisa menyapu dengan benar ketika lendir yang kental dan lengket menekan mereka. Hal ini dapat menyebabkan berbagai penyakit bagi tubuh manusia.

F. Tipe Mutasi dari CFTR

Fibrosis kistik disebabkan oleh mutasi, atau kesalahan, pada gen *cystic fibrosis transmembrane conductance regulator* (CFTR). Mutasi ini menyebabkan protein CFTR tidak terbentuk sama sekali atau terbentuk dalam bentuk yang salah sehingga tidak dapat menjalankan fungsinya dengan baik di dalam sel. Tipe mutasi dikeleompokkan berdasarkan masalah yang ditimbulkan dalam produksi protein CFTR, sebagai berikut [4]:

1. Mutasi Produksi Protein (Kelas 1)

   Mutasi ini, termasuk mutasi *nonsense* dan *splice*, mengganggu produksi protein CFTR. Mutasi *nonsense* menambahkan sinyal "berhenti" dini dalam instruksi gen, menyebabkan produksi protein berhenti terlalu cepat sehingga tidak menghasilkan protein CFTR yang fungsional. Mutasi *splice* mengganggu kemampuan sel untuk membaca instruksi dengan benar, menyebabkan protein yang dihasilkan tidak sempurna.

2. Mutasi Pengolahan Protein (Kelas 2)

   Mutasi ini menyebabkan protein CFTR tidak bisa membentuk bentuk 3-D yang benar, yang diperlukan agar berfungsi dengan baik. Contohnya, mutasi F508del menghilangkan satu asam amino penting, menyebabkan protein tidak stabil dan dibuang oleh sel.

3. Mutasi Pengaturan Gerbang (Kelas 3)

   Mutasi ini mengunci gerbang pada protein CFTR dalam posisi tertutup, mencegah klorida untuk lewat. Obat seperti Kalydeco® dapat membantu membuka gerbang ini, memungkinkan klorida mengalir melalui saluran dan mengurangi gejala fibrosis kistik.

4. Mutasi Konduksi (Kelas 4)

   Mutasi ini mengubah bagian dalam saluran protein CFTR, sehingga klorida tidak bisa bergerak dengan lancar melalui saluran tersebut. Meskipun protein terbentuk dengan bentuk 3-D yang benar, fungsinya dalam transportasi klorida terganggu, yang mengurangi efisiensi saluran.

5. Mutasi Protein Tidak Memadai (Kelas 5)

   Mutasi ini menyebabkan jumlah protein CFTR yang berfungsi di permukaan sel berkurang. Penyebabnya bisa berupa produksi protein yang terbatas, protein yang tidak berfungsi dengan baik, atau protein yang cepat rusak. Beberapa mutasi splice termasuk dalam kategori ini, dan obat seperti Kalydeco dapat memperpanjang waktu terbukanya gerbang protein CFTR untuk meningkatkan aliran klorida.

III. Aplikasi

Pada bagian ini akan dijelaskan mengenai kode program yang digunakan untuk *pattern matching*. Program ini menggunakan Bahasa pemrogaman python.

A. *Input/Output*

Saat memulai program, program akan meminta input dari pengguna berupa sekuens CFTR. Pengguna dapat memasukkan input secara manual melalui terminal atau melalui file txt. Setelah memasukkan input, sekuens CFTR akan diproses untuk dicari lokasi terjadinya mutasi menggunakan algoritma *pattern matching*. Pada program akan diperiksa lima mutase pada sekuens, yakni, ΔF508, G542X, G551D, N1303K, dan W1282X.

Keluaran dari program ini berupa lokasi ditemukannya pola mutasi tersebut pada sekuens. Jika tidak ditemukan pada pola, program akan menuliskan -1. Selain itu, ditampilkan juga jumlah perbandingan dan waktu yang dibutuhkan untuk masing-masing algoritma.

B. *Algoritma Brute Force*

Pada Algoritma *Brute Force* akan dilakukan pencarian pola dalam teks dengan membandingkan setiap kemungkinan posisi dalam teks satu per satu. Pada setiap posisi, algoritma ini memeriksa apakah semua karakter dalam pola cocok dengan karakter yang bersesuaian dalam teks. Jika cocok, maka posisi tersebut dikembalikan sebagai lokasi ditemukannya pola. Jika tidak, algoritma bergerak satu posisi ke kanan dan terus mengulangi proses hingga ditemukan atau akhir dari teks. Berikut adalah implementasi kode dari algoritma *brute force*:

```
def brute_force_search(pattern, text):
    m = len(pattern)
    n = len(text)
    comparisons = 0
    for i in range(n - m + 1):
        j = 0
        while j < m and text[i + j] == pattern[j]:
            comparisons += 1
            j += 1
        if j == m:
            return i, comparisons  # Pattern found
        comparisons += 1  # For the final comparison that fails
    return -1, comparisons  # Pattern not found
```

C. *Algoritma Knuth-Morris-Pratt*

Dalam algoritma KMP, dilakukan dua tahap yaitu tahap *preprocessing* dan pencarian. Pada tahap preprocessing, tabel border dibuat untuk menentukan panjang prefiks yang juga merupakan sufiks. Hasil dari fungsi akan disimpan pada sebuah list yang akan digunakan saat melakukan pencarian nantinya. Pada tahap pencarian, dilakukan proses perbandingan dari depan ke belakang. Jika saat melakukan perbandingan terdapat perbedaaan karakter pada pola dan teks, maka akan digunkan tabel border untuk menggeser pola secara efisien saat terjadi ketidakcocokan, sehingga menghindari perbandingan yang berulang. Berikut adalah implementasi kode dari algoritma Knuth-Morris-Pratt:

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

def kmp_search(pattern, text):
    m = len(pattern)
    n = len(text)
    border = kmp_preprocess(pattern)
    i = 0  # index for text
    j = 0  # index for pattern
```

```
    comparisons = 0
    while i < n:
        if pattern[j] == text[i]:
            comparisons += 1
            i += 1
            j += 1
        if j == m:
            return i - j, comparisons  # Pattern found
        elif i < n and pattern[j] != text[i]:
            comparisons += 1
            if j != 0:
                j = border[j - 1]
            else:
                i += 1
    return -1, comparisons  # Pattern not found
```

D. Algoritma Boyer-Moore

Algoritma Boyer-Moore adalah algoritma pencarian pola yang sangat efisien, terutama untuk teks panjang. Algoritma ini menggunakan teknik *The looking glass technique* yang mana dilakukan pencocokan dari belakang ke depan dan menggunakan teknik *character jump* untuk menggeser pola lebih jauh ketika terjadi ketidakcocokan. Tabel last_occurance dibuat selama tahap preprocessing untuk menentukan seberapa jauh pola dapat digeser berdasarkan karakter dalam teks yang tidak cocok. Selama pencarian, jika terjadi ketidakcocokan, pola digeser sesuai dengan nilai dalam tabel last_occurance. Hal ini memungkinkan lompatan besar yang mengurangi jumlah perbandingan yang diperlukan.

```
def boyer_moore_search(pattern, text):
    m = len(pattern)
    n = len(text)
    last_occurance = [m] * 256
    comparisons = 0

    # Preprocess the pattern
    for k in range(m - 1):
        last_occurance[ord(pattern[k])] = m - k - 1

    k = m - 1
    while k < n:
        j = m - 1
        i = k
        while j >= 0 and text[i] == pattern[j]:
            comparisons += 1
            j -= 1
            i -= 1
        if j == -1:
            return i + 1, comparisons  # Pattern found
        comparisons += 1  # For the final comparison that fails
        k += last_occurance[ord(text[k])]
    return -1, comparisons  # Pattern not found
```

IV. Eksperimen Dan Analisis

A. Pengujian

Pada makalah ini, penulis melakukan proses pencocokan pola protein mutasi pada sekuens CFTR protein dengan mengimplementasikan algoritma Brute Force, Knuth-Morris-Pratt, dan Boyer-Moore. Terdapat lima Sekuens CFTR yang digunakan untuk proses pengujian. Sekuens dapat dilihat pada folder test yang terdapat pada github atau bisa diakses melalui test case.

Untuk setiap algoritma pattern matching yang dilakukan, akan dihitung banyaknya perbandingan yang dilakukan serta waktu yang dibutuhkan untuk mencari pola mutasi dengan potongan CFTR. Berikut adalah contoh *output* dari program.

```
Results using Brute Force:
ΔF508:
  Normal pattern found at position: 273 with 339 comparisons in 0.083400 ms
  Mutated pattern found at position: -1 with 3444 comparisons in 0.606900 ms
G542X:
  Normal pattern found at position: 444 with 491 comparisons in 0.076900 ms
  Mutated pattern found at position: -1 with 3368 comparisons in 0.506100 ms
G551D:
  Normal pattern found at position: -1 with 3376 comparisons in 0.464200 ms
  Mutated pattern found at position: 390 with 425 comparisons in 0.052300 ms
N1303K:
  Normal pattern found at position: -1 with 3368 comparisons in 0.459100 ms
  Mutated pattern found at position: 643 with 694 comparisons in 0.088500 ms
W1282X:
  Normal pattern found at position: -1 with 3971 comparisons in 0.530300 ms
  Mutated pattern found at position: 566 with 612 comparisons in 0.076400 ms

Results using KMP:
ΔF508:
  Normal pattern found at position: 273 with 339 comparisons in 0.137900 ms
  Mutated pattern found at position: -1 with 3443 comparisons in 1.440000 ms
G542X:
  Normal pattern found at position: 444 with 489 comparisons in 0.147900 ms
  Mutated pattern found at position: -1 with 3373 comparisons in 1.007000 ms
G551D:
  Normal pattern found at position: -1 with 3372 comparisons in 0.932600 ms
  Mutated pattern found at position: 390 with 421 comparisons in 0.113300 ms
N1303K:
  Normal pattern found at position: -1 with 3374 comparisons in 0.931200 ms
  Mutated pattern found at position: 643 with 694 comparisons in 0.139400 ms
W1282X:
  Normal pattern found at position: -1 with 3976 comparisons in 0.649100 ms
  Mutated pattern found at position: 566 with 611 comparisons in 0.107300 ms

Results using Boyer-Moore:
ΔF508:
  Normal pattern found at position: 273 with 44 comparisons in 0.026700 ms
  Mutated pattern found at position: -1 with 355 comparisons in 0.085700 ms
G542X:
  Normal pattern found at position: 444 with 79 comparisons in 0.028300 ms
  Mutated pattern found at position: -1 with 471 comparisons in 0.094900 ms
G551D:
  Normal pattern found at position: -1 with 384 comparisons in 0.077300 ms
  Mutated pattern found at position: 390 with 65 comparisons in 0.015400 ms
N1303K:
  Normal pattern found at position: -1 with 403 comparisons in 0.083500 ms
  Mutated pattern found at position: 643 with 100 comparisons in 0.021600 ms
W1282X:
  Normal pattern found at position: -1 with 384 comparisons in 0.071500 ms
  Mutated pattern found at position: 566 with 89 comparisons in 0.018100 ms
```

Gambar 4.1. Output Test Case 1

Hasil keluaran program menunjukkan posisi ditemukannya pola mutasi. Jika tidak ditemukan dalam teks akan tertulis posisi ditemukannya berada di -1. Hal ini tentu akan mempermudah peneliti dan mengetahui pengobatan yang tepat dengan mengetahui mutasi yang terdapat pada CFTR manusia beserta lokasi ditemukannya. Selain itu juga diberikan hasil perbandingan menggunakan tiga algoritma *string matching*.

TABLE IV. BANYAK PERBANDINGAN PENCARIAN POLA W128X

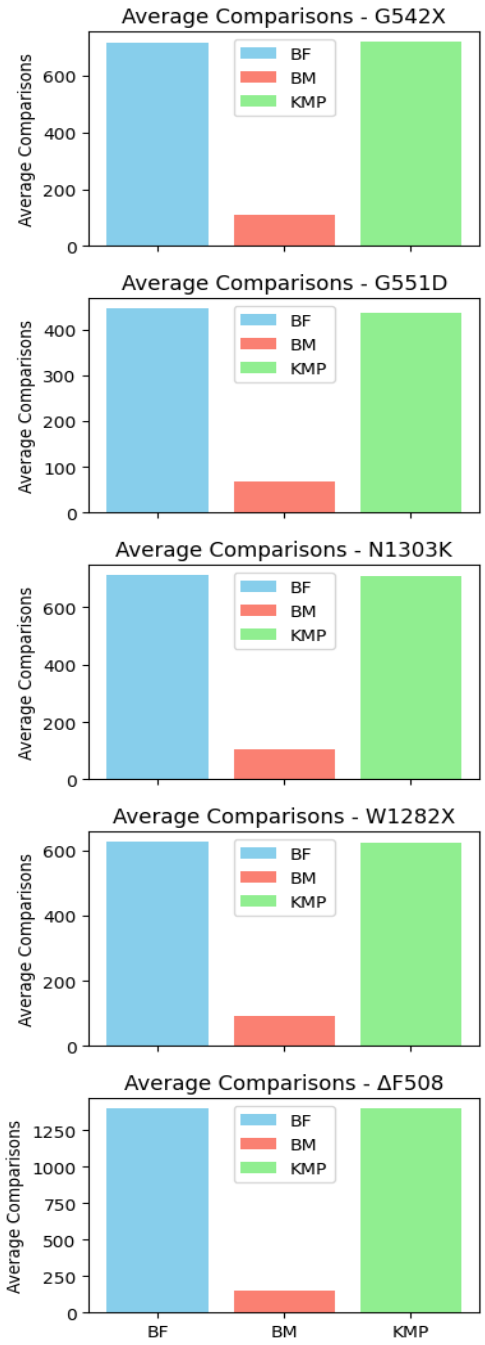
No Uji	Banyak Perbandingan		
	Brute Force	KMP	BM
1	612	611	89
2	635	629	69
3	635	629	96
4	635	629	96
5	612	611	90

TABLE V. WAKTU PENCARIAN POLA W128X

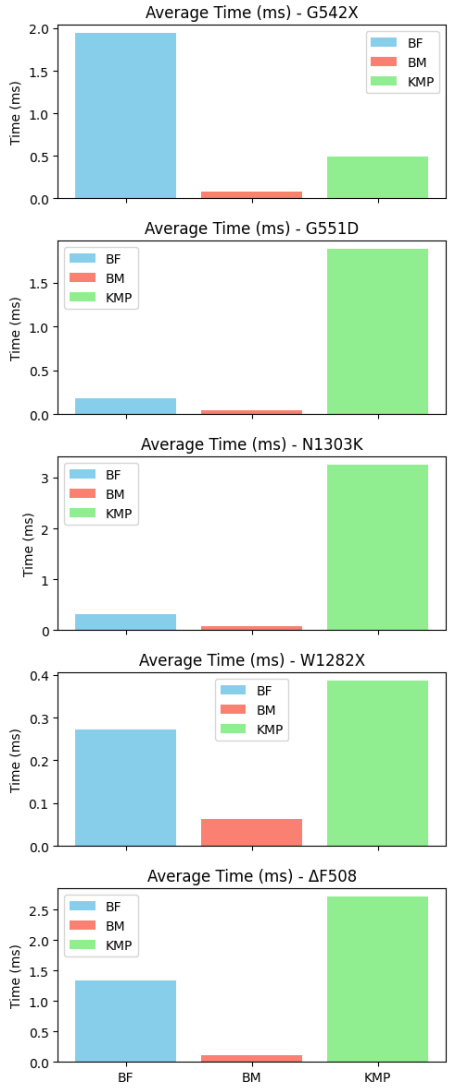
No Uji	Waktu (ms)		
	Brute Force	KMP	BM
1	0.0793	0.1152	0.02000
2	0.0544	0.0761	0.01350
3	0.0970	0.1333	0.02440
4	0.1329	0.1756	0.02870
5	0.0975	0.1332	0.02200

B. Analisis Perbadingan Algoritma

Dari hasil pengujian didapat bahwa algoritma Boyer-Moore merupakan algoritma yang paling sangkil dan efisien dalam melakukan pencarian motif protein pada sekuens CFTR. Hal ini terlihat pada jumlah perbandingan karakter yang sedikit dibandingkan dengan dua algoritma lainnya dan dari segi waktu yang dibutuhkan algoritma ini adalah algoritma tercepat. Hal tersebut dapat dilihat dalam visualisasi rata-rata waktu dan perbandingan pada seluruh motif dan kasus uji, sebagai berikut:



Gambar 4.2. Grafik Batang Jumlah Perbandingan



Gambar 4.3. Grafik Batang Waktu Eksekusi

Dari visualisasi kedua grafik di atas dapat dikehui dari segi perbandingan karakter yang dilakukan, Boyer-Moore melakukan perbandingan karakter paling sedikit. sedangkan Knuth-Morris-Pratt dan Brute Force memiliki jumlah perbandingan yang tidak berbeda jauh. Hal ini disebabkan karena algoritma BM memiliki teknik *jump forward* yang mana jika tidak ditemukan kesamaan antara karakter pada pola dengan yang ada pada teks akan langsung dilakukan lompatan jauh sehingga mengurangi perbandingan jumlah karakter yang ada. Berbeda dengan KMP dan *brute force* yang melakukan perbandingan karakter satu per satu dari depan ke belakang. Pada kasus ini KMP memiliki kasus terburuk karena pola dan teks yang ada sangat variatif dari segi jumlah karakter. Oleh karena itu, KMP menjadi mirip dengan algoritma *brute force* yang terpaksa harus membandingkan karakter satu demi satu.

Dari segi waktu yang dibutuhkan Boyer-Moore juga memiliki waktu tercepat dalam melakukan proses pencarian. Hal ini, juga didukung dengan jumlah perbandingan yang sedikit. Meski begitu, KMP dan *brute force* memiliki jumlah perbandingan yang mirip, tetapi dari segi waktu yang dibutuhkan *brute force* memiliki waktu yang lebih cepat dibanding KMP. Hal ini terjadi karena pada KMP setiap ditemukan karakter yang sama dengan teks akan dilakukan perpindahan berdasarkan *border function*. Dalam kasus ini, hal tersebut dapat menyebabkan redudansi dan dibutuhkan waktu yang lebih lama dalam melakukan pencarian. Berbeda dengan *brute force* yang secara iteratif hanya memeriksa tiap karakter satu per satu.

Oleh karena itu, didapat algoritma Boyer-Moore merupakan algoritma paling sangkil dalam menyelesaikan kasus analisis motif protein dari sekuens CFTR. Hal ini dibuktikan dengan waktu pencarian yang cepat dan jumlah perbandingan karakter yang paling sedikit dibanding KMP dan *brute force.*

C. Manfaat Sebagai Target Terapi

Dengan mengetahui motif mutasi pada CFTR melalui *pattern* matching dapat dilakukan identifikasi mutasi pada gen CFTR. Berikut adalah beberapa manfaat utama dari deteksi mutasi CFTR sebagai target terapi:

1. Pengobatan yang Dipersonalisasi: Dengan mengetahui mutasi spesifik yang ada pada gen CFTR pasien, dokter dapat memilih terapi yang paling tepat.
2. Pengembangan Obat Baru: Penelitian mengenai mutasi spesifik pada gen CFTR dapat memfasilitasi pengembangan obat-obatan baru yang ditargetkan.
3. Pemantauan Efektivitas Terapi: Deteksi mutasi juga penting untuk memantau respons pasien terhadap terapi. Dengan memeriksa apakah mutasi tertentu tetap ada atau apakah ada perubahan dalam pola mutasi, dokter dapat menilai efektivitas pengobatan dan melakukan penyesuaian yang diperlukan.
4. Peningkatan Kualitas Hidup: Dengan pengobatan yang lebih efektif dan ditargetkan, pasien dengan CF dapat mengalami peningkatan kualitas hidup yang signifikan. Terapi yang tepat dapat mengurangi frekuensi dan keparahan infeksi paru-paru, meningkatkan fungsi pencernaan, dan memperpanjang harapan hidup.
5. Pengurangan Biaya Kesehatan: Terapi yang dipersonalisasi dapat mengurangi biaya kesehatan jangka panjang dengan mengurangi kebutuhan rawat inap dan intervensi medis yang seringkali mahal. Dengan fokus pada pengobatan yang efektif sejak awal, beban ekonomi bagi pasien dan sistem kesehatan dapat dikurangi.

V. KESIMPULAN

Penelitian ini membandingkan efisiensi tiga algoritma *pattern matching* dalam mendeteksi mutasi pada sekuens CFTR: Brute Force, Knuth-Morris-Pratt (KMP), dan Boyer-Moore. Hasil penelitian menunjukkan bahwa algoritma Boyer-Moore paling efisien dalam hal jumlah perbandingan karakter dan waktu eksekusi. Boyer-Moore mengungguli dua algoritma lainnya karena kemampuannya untuk melakukan pergeseran besar saat terjadi ketidakcocokan karakter, yang mengurangi jumlah perbandingan yang diperlukan.

Implementasi algoritma *pattern matching* ini mempermudah peneliti dalam menganalisis mutasi CFTR dan membantu dalam pengembangan terapi yang tepat. Penelitian ini menegaskan pentingnya pemilihan algoritma yang tepat dalam analisis sekuens genetik, terutama untuk penyakit genetik seperti Cystic Fibrosis. Dengan menggunakan algoritma yang efisien, deteksi dan diagnosis mutasi genetik dapat dilakukan dengan lebih cepat dan akurat, mendukung pengembangan terapi yang lebih efektif dan personal. Hasil ini membuka peluang lebih besar untuk pengobatan yang lebih baik dan kualitas hidup yang lebih tinggi bagi pasien dengan CF.

LINK VIDEO YOUTUBE DAN REPOSITORY

A. *Youtube Link*

https://youtu.be/jwRYVv8d7_s

B. *Repository*

https://github.com/debrinashika/Pattern-Matching-Algorithm-as-a-Tool-for-Detecting-CFTR-Gene

UCAPAN TERIMA KASIH

Puji Syukur terhadap Allah SWT karena berkat Rahmat dan karunia-Nya, makalah berjudul “Implementasi Algoritma Pattern Matching Sebagai Alat Deteksi Mutasi Gen CFTR pada Penyakit Cystic Fibrosis” dapat terselesaikan dengan baik dan tepat waktu. Terima kasih kepada orang tua dan teman-teman seperjuangan yang selalu memberi semangat dalam proses pengerjaan. Terima kasih juga kepada Bapak Dr. Ir. Rinaldi Munir, MT. selaku dosen mata kuliah Strategi Algoritma atas bimbingannya dan sudah menyediakan laman untuk berbagi proses pembelajaran selama ini.

REFERENSI

[1] deepchecks, "deepchecks glosarry," [Online]. Available: https://deepchecks.com/glossary/pattern-matching/#:~:text=The%20process%20of%20algorithmically%20searching,and%20cannot%20create%20new%20patterns.. [Accessed 11 June 2024].

[2] R. Munir. "Pencocokan String," 2021. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf. [Accessed 11 June 2024].

[3] Cystic Fibrosis Foundation, "Basics of the CFTR Protein," [Online]. Available: https://www.cff.org/research-clinical-trials/basics-cftr-protein. [Accessed 11 June 2024].

[4] Cystic Fibrosis Foundation, "Types of CFTR Mutation," [Online]. Available: https://www.cff.org/research-clinical-trials/types-cftr-mutations. [Accessed 11 June 2024].

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, 12 Juni 2024

Debrina Veisha Rashika W
13522035