Meningkatkan Akurasi Pengenalan Audio Shazam dengan Algoritma String Matching

Muhhamad Syauqi Jannatan - 13521014[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521014@std.stei.itb.ac.id

***Abstrak*—Akhir-akhir ini, industri musik berkembang dengan pesat. Setidaknya satu lagu baru dirilis setiap harinya. Hal ini kadang membuat kita bingung untuk mengidentifikasi lagu yang terdengar asing bagi kita. Namun, seiring dengan perkembangan teknologi, masalah tersebut sudah dapat dipecahkan. Banyak dari kita cenderung menggunakan layanan aplikasi dengan fitur *music recognition* setiap kali kita menemukan lagu asing. Lagi pula, sangat mudah untuk mengeluarkan ponsel kita, membuka aplikasi, dan mengetahui segalanya tentang lagu misteri dalam hitungan detik. Shazam merupakan salah satu aplikasi paling terkenal yang menggunakan sistem tersebut. Namun, bagaimana Shazam memberi kita semua informasi ini dengan begitu cepat?**

Makalah ini membahas pengembangan metode untuk meningkatkan akurasi sistem pengenalan lagu Shazam menggunakan algoritma string matching. Dalam pengembangan metode ini, dilakukan pemilihan fitur-fitur kunci yang paling mungkin terdapat pada sebuah lagu dengan menggunakan algoritma string matching. Selanjutnya, fitur-fitur ini digunakan untuk menghasilkan sidik jari audio yang lebih baik dan akurat. Metode ini diuji pada dataset audio yang beragam dan hasilnya menunjukkan peningkatan akurasi sistem pengenalan lagu Shazam yang signifikan. Oleh karena itu, metode ini dapat dijadikan sebagai alternatif yang lebih baik untuk pengenalan lagu dalam aplikasi musik digital.

***Kata Kunci*—*Shazam*, *fingerprinting*, KMP, *Boyer-Moore*.**

I. PENDAHULUAN

Teknologi audio fingerprinting telah berkembang pesat dalam beberapa tahun terakhir dan telah menjadi solusi yang umum digunakan untuk mengenali musik dan audio dalam berbagai aplikasi seperti Shazam, Spotify, dan lainnya. Aplikasi Shazam, yang pertama kali diluncurkan pada tahun 1999, telah menjadi salah satu aplikasi audio fingerprinting terkemuka dan paling dikenal di dunia. Namun, teknologi ini masih memiliki beberapa kelemahan dalam hal akurasi pengenalan audio.

Gambar 1.1 Logo Aplikasi Shazam

Shazam adalah mesin pencarian musik yang mengidentifikasi file audio berdasarkan grafik frekuensi waktu yang disebut spektogram. Ini menggunakan mikrofon internal smartphone atau komputer untuk mengumpulkan sampel singkat audio yang sedang diputar. Shazam menyimpan katalog sidik jari audio dalam database. Pengguna menandai lagu selama 10 detik dan aplikasi membuat audio. Shazam bekerja dengan menganalisis suara yang ditangkap dan mencari rekaman yang relevan berdasarkan sidik jari akustik di database yang mereka miliki. Jika menemukan kecocokan, informasi seperti artis, judul lagu, dan album akan dikirim kembali ke pengguna. Salah satu alasan popularitas Shazam adalah kemampuannya untuk memberikan hasil identifikasi musik yang singkat. Waktu respons yang singkat ini dimungkinkan karena algoritma yang memanfaatkan sistem kerja yang dibut spektogram.

Dalam penggunaan aplikasi Shazam, masih terjadi kesalahan dalam mengenali audio, terutama ketika audio yang direkam memiliki kualitas rendah atau terdapat banyak kebisingan. Oleh karena itu, diperlukan suatu metode untuk meningkatkan akurasi pengenalan audio Shazam. Disini, penulis tidak akan membahas lebih lanjut tentang algoritma yagn dipakai perusahaan Shazam secara mendalam. Jadi, penulis akan membahas tentang sistem kerja unik yang digunakan aplikasi Shazam pada pembacaan identitas musik yang memanfaatkan prinsip kerja dari pencocokan string.

Tujuan dari penelitian ini adalah untuk meningkatkan akurasi pengenalan audio Shazam dengan menggunakan algoritma pencocokan string. Selain itu, penelitian ini juga dapat menjadi referensi bagi peneliti lain yang tertarik untuk mengembangkan aplikasi pengenalan audio menggunakan algoritma pencocokan string.

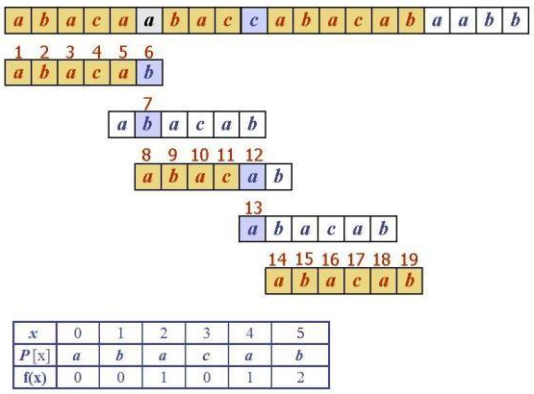
II. LANDASAN TEORI

A. *Algoritma Knuth-Morris-Pratt (KMP)*

Algoritma KMP adalah salah satu algoritma pencarian string, dikembangkan secara terpisah oleh Donald E. Knuth pada tahun 1967 dan James H. Morris bersama Vaughan R. Pratt pada tahun 1966, namun keduanya mempublikasikannya secara bersamaan pada tahun 1977 (Astuti, 2017). Jika kita melihat algoritma brute force lebih mendalam, kita mengetahui bahwa dengan mengingat beberapa perbandingan

yang dilakukan sebelumnya kita dapat meningkatkan besar pergeseran yang dilakukan. Hal ini menghemat perbandingan, yang selanjutnya akan meningkatkan kecepatan pencarian. Secara Sistematis, langkah-langkah yang dilakukan algoritma KMP pada saat pencocokan string adalah sebagai berikut (Nursobah & Pahrudin. P, 2019) :

1. Algoritma KMP mulai mencocokan pattern pada awal teks.
2. Dari kini ke kanan, algoritma ini akan mencocokan karakter per karakter pattern, dengan karakter di teks yang bersesuaian sampai salah satu kondisi berikut terpenuhi:
   a. Karakter di pattern dan diteks yang dibandingkan tidak cocok (mismatch).
   b. Semua Karakter di pattern cocok. Kemudian algoritma akan memberitahu penemuan posisi ini.
3. Algoritma kemudian menggeser pattern berdasarkan table next, lalu menghitung langkah 2 sampai pattern berada diujung teks.



x	0	1	2	3	4	5
P[x]	a	b	a	c	a	b
f(x)	0	0	1	0	1	2

Gambar 2.1 Pencocokan string dengan Algoritma KMP

Sumber: https://koding4fun.wordpress.com/2010/05/29/knuth-morris- pratt-algorithm/

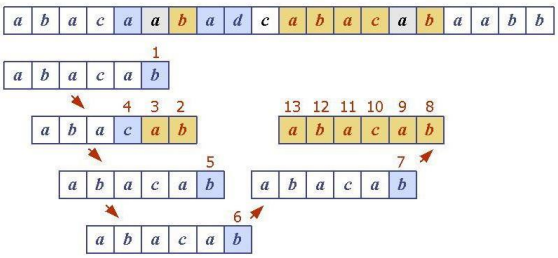
Algoritma Knuth-Morris-Pratt (KMP) merupakan salah satu algoritma pencocokan pola. Metode pencarian KMP bekerja dengan melewatkan perbandingan-perbandingan yang tidak diperlukan untuk menghindari besarnya jumlah perbandingan, dengan demikian mencapai waktu berjalan O(n+m) yang optimal dalam kasus terburuk (worst case) pencocokan pola algoritma harus memeriksa semua karakter teks dan semua karakter dari pola setidaknya sekali. Ide utama algoritma KMP adalah untuk preprocess string pola P sehingga untuk menghitung fungsi kegagalan f yang menunjukan pergeseran P yang tepat sehingga kita dapat menggunakan kembali perbandingan yang dilakukan sebelumnya (Goodrich, Tamassia, & Mount, 2011).

B. Algoritma Boyer-Moore

Algoritma Boyer Moore adalah algoritma pencarian string yang paling efektif saat ini. Algoritma yang ditemukan oleh Bob Boyer dan J. Strother Moore ini telah menjadi standar untuk berbagai literatur pencarian string. Algoritma Boyer Moore akan menyimpan informasi pergeseran untuk melakukan pencarian string. Karakteristik utama dari algoritma Boyer Moore adalah algoritma ini melakukan pencocokan string mulai dari kanan ke kiri.

Algoritma *Boyer-Moore* memanfaatkan *looking glass technique*. Jika karakter belakang *pattern* sudah tidak sesuai dengan teks, maka karakter bagian depan sudah tidak perlu dicek lagi karena akan tetap salah. Algoritma *Boyer-Moore* mengecek mulai dari karakter bagian belakang, lalu jika ditemukan ketidakcocokkan, maka indeks pencarian bisa berpindah sesuai dengan kondisi. Ada tiga kondisi yang mungkin:

1. Karakter yang tidak cocok tidak terdapat pada pattern, maka pattern akan dipindahkan melewati karakter itu.
2. Karakter yang tidak cocok terdapat pada pattern, dan di sebelah kanan indeks pattern yang salah. Pattern digeser menyesuaikan karakter itu.
3. Karakter yang tidak cocok terdapat pada pattern, tetapi tidak memungkinkan untuk menyesuaikan dengan pattern karena indeks karakter ada di sebelah kiri pattern. Maka pattern digeser ke kanan sekali.



Gambar 2.2 Pencocokan string dengan Algoritma BM

Sumber: https://koding4fun.wordpress.com/2010/05/29/boyer-moore-algorithm/

Algoritma Boyer-Moore memiliki kompleksitas yang lebih baik dari algoritma *brute force*. Akan tetapi tidak selalu lebih baik daripada algoritma KMP. Ada suatu saat dimana algoritma Boyer-Moore tidak lebih baik daripada algoritma KMP. Algoritma Boyer-Moore memiliki kompleksitas O(*m*+x) untuk melakukan inisialisasi array kemunculan karakter terakhir dari *pattern*. Langkah pencarian dalam algoritma Boyer-Moore memiliki kompleksitas terbaik yaitu O(*n*/*m*) dan kompleksitas terburuk yaitu O(*mn*), sama dengan kompleksitas *worst case brute force*. Salah satu contoh kasus yang menyebabkan algoritma Boyer-Moore berada dalam *worst case* adalah contoh kasus di bawah ini.

```
T: "aaaaaaaaaaaaaaaaaaapaaaa"
P: "paaaa"
```

Contoh kasus di atas menyebabkan berkali-kali pencocokan gagal di karakter pertama pada *pattern*. Selain itu,huruf pada teks yang menyebabkan kegagalan berada di sebelah kanan pada *pattern*, menyebabkan banyak pergeseran yang dilakukan menjadi hanya 1 karakter, sehingga *pattern* akan selalu digeser sebanyak 1 kali hingga ditemukan huruf `p` pada teks yang kemudian menyebabkan *alignment* antara teks dan *pattern* pada karakter `p`. Oleh karena itu, dapat disimpulkan dari sini bahwa

kompleksitas algoritma untuk *worst case* pada algoritma Boyer-Moore adalah O(*mn*).

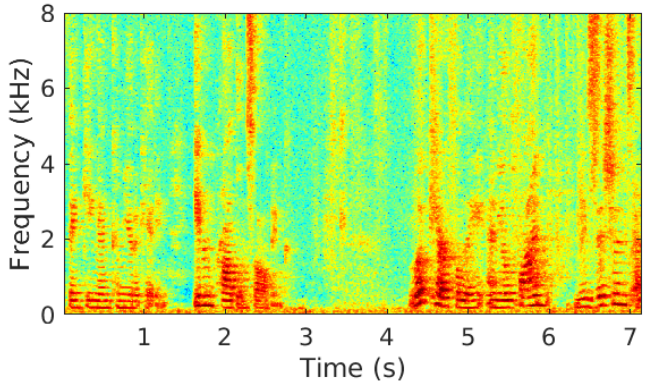
C. Pengenalan Audio Fingerprinting

Sidik jari audio adalah tanda tangan ringkas berbasis konten yang meringkas rekaman audio. Sidik jari audio telah menarik banyak perhatian karena kemampuan pemantauan audionya. Audio sidik jari atau teknologi identifikasi berbasis konten (CBID). mengekstrak karakteristik akustik yang relevan dari konten audio dan menyimpannya dalam database. Saat disajikan dengan bagian konten audio yang tidak dikenal, karakteristik bagian tersebut dihitung dan dicocokkan dengan yang disimpan dalam database. Menggunakan sidik jari dan algoritme pencocokan, versi berbeda dari satu rekaman dapat diidentifikasi sebagai judul musik yang sama.

Untuk mendapatkan hasil identifikasi lagu yang akurat, sidik jari audio yang digunakan haruslah unik dan robust. Salah satu fitur audio dari sebuah lagu yang dapat dijadikan sebagai sidik jari audio yang unik dan robust adalah peak frequency yaitu frekuensi yang memiliki magnitudo tertinggi di antara frekuensi-frekuensi lainnya dalam rentang frekuensi tertentu. Untuk mendapatkan peak frequency, sinyal audio lagu ditransformasikan dari domain waktu ke domain waktu-frekuensi menggunakan teknik STFT (Short Time Fourier Transform).

D. Spektogram dan Short Time Fourier Transform

Spektrogram adalah cara visual untuk merepresentasikan kekuatan sinyal, atau "kenyaringan" sinyal dari waktu ke waktu dari berbagai frekuensi yang ada dalam bentuk gelombang tertentu. Dengan kata lain, ini adalah grafik tiga dimensi. Jika kita melihat contoh di bawah ini, kita dapat melihat bahwa sumbu mewakili frekuensi dan waktu, sedangkan nilai ketiga (amplitudo) diwakili oleh intensitas warna setiap titik pada gambar.



Gambar 2.3 Contoh Spektografi
Sumber:https://www.researchgate.net/figure/Example-of-spectrogram-for-a-signal-from-the-Dynamic-dataset-a-close-talk-clean_fig4_329841571

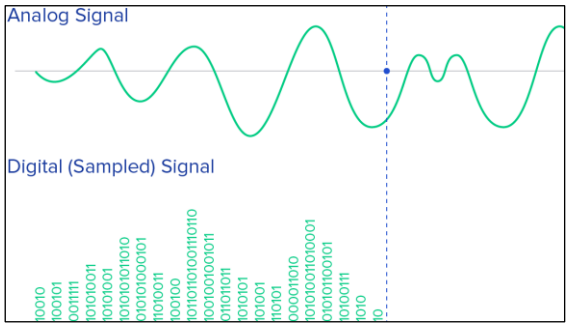
Spektogram sering digunakan untuk menganalisis suara dan musik serta menentukan karakteristik suara yang dihasilkan oleh suatu benda atau hewan, atau untuk mengidentifikasi pola dalam sinyal audio yang kompleks. Spektogram dapat dibentuk melalui perhitungansignal waktu dengan menggunakan *Short-Time Fourier transform (STFT).*

Short-Time Fourier Transform (STFT) adalah sebuah teknik yang digunakan untuk menganalisis suatu sinyal dalam domain frekuensi dengan memecah sinyal menjadi beberapa bagian yang lebih kecil (biasanya disebut sebagai "time-frames") dan melakukan transformasi Fourier pada masing-masing bagian tersebut. Ini berguna untuk menganalisis sinyal yang berubah-ubah atau memiliki struktur frekuensi yang kompleks dalam waktu yang singkat. STFT sering digunakan dalam bidang audio dan musik untuk menganalisis dan memodifikasi sinyal suara.

III. METODOLOGI

A. Proses Pengubahan Audio Menjadi String

Seperti yang sudah dibahas sebelumnya, Shazam mengidentifikasi suara dengan melakukan beberapa Langkah. Langkah pertama adalah melakukan sampling, yaitu proses mengubah suara analog menjadi digital. Mikrofon mengubah suara disekitar menjadi sinyal diskrit, tetapi bentuknya tetap kontinu agar bisa diproses. Transformasi dari analog ke digital ini dilakukan menggunakan Transformasi Fourier. Transformasi dilakukan dalam batasan waktu tertentu, missal 0,1 second, karena batasan dalam mengubah dari kontinu menjadi diskrit. Misalnya, kita dapat mengambil sampel pada setiap detik lagu yang direkam dan mengubah frekuensi pada detik tersebut menjadi representasi biner. Sebagai contoh, jika pada detik pertama frekuensi lagu yang direkam adalah 10 Hz, maka kita dapat mewakilinya dengan angka biner 1010. Proses ini diulangi untuk setiap 0,1 second selama durasi lagu, dan kita akan mendapatkan serangkaian angka biner yang terbentuk dari setiap konversi tersebut. Hasil konversi ini akan membentuk sebuah string panjang yang merupakan hasil gabungan dari konversi pada setiap interval waktu, yang terdiri dari angka biner 0 dan 1. String ini kemudian digunakan sebagai pola yang akan dicari dalam teks lagu yang tersedia dalam media penyimpanan atau katalog lagu yang ada.



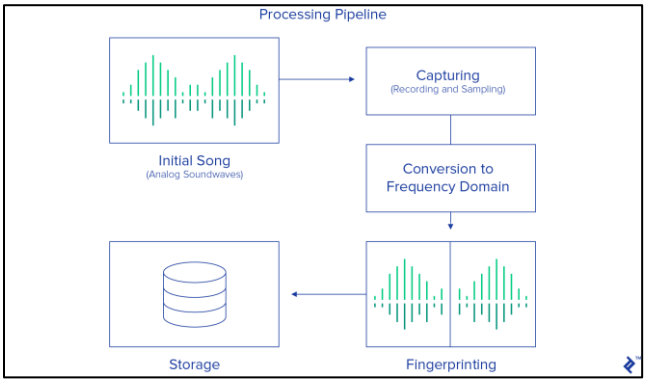
Gambar 3.1 Konversi sinyal analog ke digital.
Sumber: https://www.toptal.com/algorithms/shazam-it-music-processing-fingerprinting-and-recognition

Dengan kata lain, dalam proses ini, sinyal analog lagu direpresentasikan dalam bentuk biner berdasarkan frekuensi pada setiap satuan waktu. Kemudian, string biner yang dihasilkan tersebut digunakan sebagai pola untuk pencocokan dalam teks lagu yang ada dalam basis data atau katalog lagu yang tersedia.

Setelah string biner lagu dihasilkan, langkah selanjutnya adalah menghasilkan fingerprint atau sidik jari lagu tersebut. Fingerprinting melibatkan proses ekstraksi fitur atau karakteristik unik dari string biner lagu. Fitur-fitur ini dapat berupa pola frekuensi, energi, atau informasi lainnya yang mencirikan lagu secara unik. Fingerprint yang dihasilkan adalah representasi singkat dan unik dari lagu yang digunakan untuk tujuan identifikasi.

Setelah fingerprint lagu dihasilkan, langkah selanjutnya adalah mencocokkan fingerprint tersebut dengan fingerprint lagu dalam basis data atau katalog lagu yang ada. Ini melibatkan penggunaan algoritma string matching, di mana fingerprint lagu yang diidentifikasi digunakan sebagai pola yang dicari dalam fingerprint lagu dalam basis data. Setelah proses identifikasi dilakukan, informasi tentang lagu yang diidentifikasi dapat disimpan dalam media penyimpanan. Ini dapat berupa penyimpanan dalam basis data yang mengandung informasi seperti judul lagu, artis, album, dan metadata lainnya terkait lagu tersebut. Data ini dapat digunakan untuk mengenali lagu di masa mendatang saat lagu tersebut diputar atau dicocokkan lagi.

Jika digambarkan dalam bentuk skema, proses sebuah penentuan sebuah lagu dapat digambarkan dalam skema di bawah ini.



Gambar 3.2 Overview Proses Pencocokan Lagu

Sumber: https://www.toptal.com/algorithms/shazam-it-music-processing-fingerprinting-and-recognition

B. Efektivitas Algoritma KMP dan Boyer-Moore

Setelah berhasil mendapatkan string yang berisi karakter biner, yaitu 0 dan 1 dari konversi sinyal analog ke sinyal digital, string tersebut akan dijadikan sebuah pattern. Pattern itulah yang akan dicocokkan dengan teks yang ada dalam kamus lagu menggunakan string matching. Pada bagian ini, akan dilakukan eksperimen untuk membandingkan efektivitas kedua algoritma tersebut, yaitu algoritma KMP dan algoritma Boyer-Moore.

Berikut merupakan program algoritma KMP untuk membandingkan antara teks dan pattern:

```
def kmp_match(text, pattern):
    n = len(text)
    m = len(pattern)

    # Membuat tabel lompatan
    lps = [0] * m
    j = 0
    for i in range(1, m):
        while j > 0 and pattern[j] != pattern[i]:
            j = lps[j-1]
        if pattern[j] == pattern[i]:
            j += 1
        lps[i] = j

    # Pencocokan string menggunakan tabel lompatan
    j = 0
    comparisons = 0
    for i in range(n):
        while j > 0 and pattern[j] != text[i]:
            j = lps[j-1]
        if pattern[j] == text[i]:
            j += 1
        if j == m:
            return True, comparisons
        comparisons += 1

    return False, comparisons
```

Fungsi diatas menerima dua parameter, yaitu text (teks yang akan dicocokkan) dan pattern (pola yang dicari). Fungsi ini akan mengembalikan sebuah tuple yang berisi boolean (True jika pola ditemukan dalam teks, False jika tidak) dan jumlah perbandingan yang dilakukan. Pertama, membuat tabel lompatan: Variabel lps (Longest Proper Prefix which is also Suffix) merupakan tabel lompatan yang digunakan dalam algoritma KMP. Pada bagian ini, tabel lompatan diinisialisasi dengan nilai 0. Kemudian dilakukan iterasi untuk mengisi nilai-nilai pada tabel lompatan dengan menggunakan metode prefix-suffix matching. Selanjutnya, dalam pencocokan string menggunakan tabel lompatan, variabel j digunakan sebagai indeks pola yang sedang dicocokkan. Kemudian dilakukan iterasi pada teks dan dilakukan pencocokan karakter antara pola dan teks. Jika karakter pada pola dan teks cocok, maka indeks j akan bertambah. Jika j sama dengan panjang pola (m), berarti pola telah ditemukan dalam teks dan fungsi akan mengembalikan True berserta jumlah perbandingan yang dilakukan. Jika karakter pada pola dan teks tidak cocok, maka dilakukan lompatan berdasarkan nilai pada tabel lompatan. Variabel comparisons digunakan untuk menghitung jumlah perbandingan yang dilakukan. Jika pada akhir pencocokan tidak ditemukan pola dalam teks, maka fungsi akan mengembalikan False berserta jumlah perbandingan yang dilakukan.

Berikut merupakan program algoritma Boyer-Moore menggunakan bahasa python

```
class last_occurrence(object):
    def __init__(self, pattern, alphabet):
        self.occurrences = dict()
        for letter in alphabet:
            self.occurrences[letter] = pattern.rfind(letter)
```

```
    def __call__(self, letter):
        return self.occurrences[letter]

def boyer_moore_match(text, pattern):
    alphabet = set(text)
    last = last_occurrence(pattern, alphabet)
    m = len(pattern)
    n = len(text)
    i = m - 1
    j = m - 1
    comparisons = 0
    while i < n:
        comparisons += 1
        if text[i] == pattern[j]:
            if j == 0:
                return True, comparisons
            else:
                i -= 1
                j -= 1
        else:
            l = last(text[i])
            i = i + m - min(j, 1+l)
            j = m - 1
    return False, comparisons
```

Class last_occurrence diatas berfungsi sebagai fungsi pemetaan yang menghasilkan informasi posisi kemunculan terakhir dari setiap huruf dalam pola. Kemudian fungsi boyer_moore_match diatas menerima dua parameter, yaitu text (teks yang akan dicocokkan) dan pattern (pola yang dicari). Kurang lebih sama seperti kode algoritma KMP, tetapi ada tambahan variabel, yaitu variabel alphabet digunakan untuk menyimpan himpunan karakter unik dalam teks dan variabel last menggunakan objek last_occurrence yang telah dibuat sebelumnya untuk mendapatkan informasi posisi kemunculan terakhir dari setiap huruf dalam pola. Pencocokan dilakukan dengan menggunakan aturan pada algoritma Boyer-Moore, dengan melakukan perbandingan karakter antara teks dan pola. Jika pola ditemukan dalam teks, fungsi mengembalikan True berserta jumlah perbandingan yang dilakukan. Jika pola tidak ditemukan dalam teks, fungsi mengembalikan False berserta jumlah perbandingan yang dilakukan.

Hasil eksekusi program yang dihasilkan memberikan hasilsebagai berikut:

1. Test Case 1

```
Masukkan text disini: 10100101010101010101011101000101010
Masukkan pattern disini: 1110
Jumlah perbandingan menggunakan Algoritma KMP: 24
Jumlah perbandingan menggunakan Algoritma BM: 44
```

Gambar 3.3 Test case 1 Pencocokan String

Sumber: Dokumen Pribadi

2. Test Case 2

```
Masukkan text disini: 11111111010101010101011111110010
Masukkan pattern disini: 0010
Jumlah perbandingan menggunakan Algoritma KMP: 32
Jumlah perbandingan menggunakan Algoritma BM: 56
```

Gambar 3.4 Test case 2 Pencocokan String

Sumber: Dokumen Pribadi

3. Test Case 3

```
Masukkan text disini: 00100010010101010101010110100
Masukkan pattern disini: 11010
Jumlah perbandingan menggunakan Algoritma KMP: 27
Jumlah perbandingan menggunakan Algoritma BM: 63
```

Gambar 3.5 Test case 3 Pencocokan String

Sumber: Dokumen Pribadi

Dari hasil eksperimen, dapat disimpulkan bahwa algoritma KMP memiliki performa yang lebih baik dalam hal efektivitas dan efisiensi dibandingkan dengan algoritma Boyer-Moore. Algoritma KMP menggunakan tabel lompatan (failure function) yang membantu menghindari perbandingan ulang yang tidak perlu, sehingga jumlah perbandingan karakter yang dilakukan lebih sedikit. Algoritma KMP cocok digunakan dalam kasus-kasus di mana variasi karakter dalam pola cukup kecil dan tidak ada karakter bersebelahan yang berbeda di akhir pola.

Di sisi lain, algoritma Boyer-Moore menggunakan pendekatan last occurrence (kemunculan terakhir) untuk mengurangi perbandingan ulang. Namun, algoritma ini memiliki keterbatasan dalam menangani variasi karakter yang sedikit dan karakter bersebelahan yang berbeda di akhir pola. Hal ini menyebabkan pergeseran yang terbatas dan jumlah perbandingan karakter yang lebih banyak.

IV. KESIMPULAN

Dalam penelitian ini, dilakukan pengembangan metode string matching untuk meningkatkan akurasi sistem pengenalan audio Shazam. Metode ini didasarkan pada konversi sinyal analog menjadi sinyal digital yang diwakili oleh string biner, dan kemudian dilakukan pencocokan string antara pola (pattern) dengan teks lagu yang ada dalam kamus lagu.

Dua algoritma string matching yang dievaluasi dalam penelitian ini adalah algoritma Knuth-Morris-Pratt (KMP) dan algoritma Boyer-Moore (BM). Kedua algoritma ini memiliki pendekatan yang berbeda dalam mencocokkan pola dengan teks. Melalui eksperimen yang dilakukan, dapat diperoleh pemahaman tentang efektivitas dan efisiensi kedua algoritma ini dalam mengenali lagu. Dalam konteks pengenalan lagu pada sistem Shazam, efektivitas dan efisiensi algoritma KMP menjadi faktor penting untuk meningkatkan akurasi dan kecepatan dalam mengidentifikasi lagu berdasarkan pola yang dihasilkan dari sinyal audio yang direkam. Keunggulan algoritma KMP dalam hal jumlah perbandingan karakter yang lebih sedikit dapat membantu dalam mempercepat proses pencocokan dan meningkatkan akurasi pengenalan lagu.

Dalam penelitian ini, kontribusi utama adalah implementasi dan evaluasi efektivitas algoritma KMP dan Boyer-Moore dalam konteks pengenalan audio pada sistem Shazam. Hasil eksperimen memberikan pemahaman yang lebih baik tentang perbandingan kinerja kedua algoritma ini, dan memberikan dasar untuk memilih algoritma yang paling sesuai untuk meningkatkan akurasi dan efisiensi sistem Shazam.

VI. Ucapan Terima Kasih

Puji syukur penulis panjatkan kepada Allah SWT atas berkat, rahmat, dan hidayat-Nya, dapat terselesaikannya tugas makalah ini dengan baik. Pada kesempatan ini, penulis ingin mengucapkan terima kasih kepada Ir. Rila Mandala, M.Eng., Ph.D. selaku Dosen Mata Kuliah IF2211 Strategi Algoritma Kelas 03 atas bimbingan dan ilmu yang telah diberikan. Penulis juga ingin menyampaikan terima kasih kepada kedua orang tua atas dukungan yang selalu diberikan kepada penulis. Penulis juga mengucapkan terima kasih kepada teman-teman yang senantiasa memberikan bantuan dalam penyusunan makalah ini.

Referensi

[1] Munir, Rinaldi. Pencocokan string (String matching/pattern matching), URL: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf >
Diakses: 12 Mei 2023

[2] Ana Haris. *How does Shazam work?, URL:* <*https://medium.com/@anaharris/how-does-shazam-work-d38f74e41359*>
Diakses: 12 Mei 2023

[3] Columbia. *Paper An Industrial-Strength Audio Search Algorithm* < http://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf >
Diakses: 21 Mei 2023

[4] Siddharth. *The Boyer Moore String Search Algorithm*. < https://ccrma.stanford.edu/~jos/mdft/Spectrograms.html >
Diakses: 21 Mei 2023

[5] Jovanic, Jovan. < https://www.toptal.com/algorithms/shazam-it-music-processing-fingerprinting-and-recognition>
Diakses: 21 Mei 2023

[6] MacLeod, Cameron. abracadabra: How does Shazam work? < https://www.toptal.com/algorithms/shazam-it-music-processing-fingerprinting-and-recognition>
Diakses: 12 Mei 2023

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, 22 Mei 2023

Muhhamad Syauqi Jannatan
NIM : 13521014