# Penerapan Algoritma Knuth-Morris-Pratt dalam *Music Identification (Musipedia)*

Adi Nugraha Setiadi 13508062
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*If18062@students.itb.ac.id*

***Saat ini telah terdapat jutaan judul music dengan berbagai macam genrenya di seluruh dunia. Tentu saja tidak seorangpun mampu mengenali setiap musik yang didengarnya. Untuk itu diperlukan sebuah aplikasi yang mendukung untuk pengidentifikasian musik yang dicari berdasarkan konten/isi dari musik tersebut. Sehingga pencarian musik kini tidak hanya mengandalkan pencarian dengan menggunakan keyword berupa artis, pencipta lagu, judul album, bahkan potongan liriknya. Meskipun sudah terdapat aplikasi yang menggunakan pencarian berdasarkan konten lagu, diharapkan aplikasi seperti ini lebih dikenal masyatakat. Penulis hanya akan membahas cara pencarian dan pengambilan informasi musik yang telah ada dalam database.***

**Kata kunci: Kontribusi, Knuth-Morris-Pratt, Musipeda, Matching, String, Substring, Pola, Pencocokan string, *database.***

## I. PENDAHULUAN

Aplikasi *track identification* merupakan aplikasi pencari nama artis, judul lagu, serta judul album dari sebuah sampel musik yang telah didengarkan dan ingin dicari metadatanya oleh pengguna aplikasi ini. Sebelum membahas lebih jauh tentang cara kerja *Musipedia*, kita teliti terlebih dahulu tentang komponen pencari yang digunakan.

*Musipedia* merupakan salah satu aplikasi yang menggunakan algoritma Knuth-Morris-Pratt (KMP). Karena pada aplikasi ini dilakukan pencarian suatu *pattern* atau pola yang telah tersimpan pada database. Algoritma KMP dapat secara baik digunakan untuk mencari pola musik yang sama dengan sampel yang ada. Algoritma ini akan selalu menghasilkan sebuah solusi.

Pencarian ini tidak hanya terbatas pada sebuah lagu, namun dapat berupa instrument-instrumen yang dinyanyikan, fitur ini melebihi kemampuan dari *Music Information Retrieval* (MIR) biasa. Pada aplikasi berbasis MIR yang telah banyak ada, contohnya Shazam, Shazam menggunakan *fingerprinting Audio* untuk melakukan pencarian, sebuah teknik yang memungkinkan untuk mengidentifikasi rekaman. Di sisi lain Musipedia mengidentifikasi potongan musik yang berisi melodi yang diberikan. Shazam menemukan persis rekaman yang berisi potongan yang diberikan, namun tidak ada tidak terdapat rekaman lain dari bagian yang sama.

## II. KONSEP DASAR ALGORITMA KNUTH-MORRIS-PRAT (KMP)

Algoritma KMP merupakan algoritma yang digunakan untuk melakukan proses pencocokan string. Algoritma ini merupakan jenis *Exact String Matching Algoritm* yang merupakan pencocokan *string* secara tepat dengan susunan karakter dalam *string* yang dicocokkan memiliki jumlah maupun urutan dalam *string* yang sama. [1]

Contoh: kata *algoritma* akan menunjukkan kecocokan hanya dengan kata *algoritma*. Pada persoalan pencocokan string umumnya diberikan dua buah hal utama:

- Teks, string yang relative panjang yang akan menjadi bahan yang akan dicari kecocokannya (dimisalkan panjang dari teks adalah n)
- Pattern, string dengan panjang relatif lebih pendek dari teks (misalkan panjang pattern adalah m, maka m<n). Pattern akan digunakan sebagai string yang dicocokkan ke dalam teks

Pada algoritma KMP, kita simpan informasi yang digunakan untuk melakukan pergeseran lebih jauh, tidak hanya satu karakter seperti algoritma *Brute Force*. Algoritma ini melakukan pencocokan dari kiri ke kanan. [3]

Terdapat beberapa difinisi pada algoritma KMP:

1. Misalkan *A* adalah alphabet dan $x = x1x2...xk$ adalah *string* yang panjangnya k yang dibentuk dari karakter-karakter di dalam alphabet A.
   - Awalan (*prefix*) dari *x* adalah upa-string (*substring*) *u* dengan $u = xk - b\ xk - b + 1...xk$, $k \in \{1, 2, 3, ..., k - 1\}$ dengan kata lain, k diawali dengan u.
   - Akhiran (*suffix*) dari *x* adalah upa-string (*substring*) *u* dengan $u = xk - b\ xk - b + 1...xk$, $k \in \{1, 2, ..., k - 1\}$ dengan kata lain *x* diakhiri dengan *v*.
   - Pinggiran (*border*) dari *x* adalah upa-string (*substring*) *r* sedemikian sehingga $r = x1\ x2...xk - 1$ dan $u = xk - b\ xk - b + 1\ ...xk$, $k \in$

*{1, 2, ..., k = 1},* dengan kata lain, pinggiran dari *x* adalah upa-string yang keduanaya awalan dan juga akhiran sebenarnya dari *x*.

2. Fungsi pinggiran b(j) didefinisikan sebagai ukuran awalan terpanjang dari P yang merupakan akhiran dari P[1..j]

Yang perlu diketahui oleh pemakai kode adalah bahwa algoritma KMP melakukan proses awal(preprocessing) terhadap pattern P dengan menghitung fungsi pinggiran yang mengindikasikan pergeseran s terbesar yang mungkin dengan menggunakan perbandingan yang dibentuk sebelum pencarian string. Dalam literatur lain fungsi ini sering disebut dengan overlap function, failure function, fungsi awalan dan lainnya. Fungsi preprosesing pinggiran tersebut dapat dipahami sebagai berikut:

```
procedure FungsiHitungPinggiran
(
   input m: integer,
   P: array [1..m] of char,
   output b : array[1..m] of integer
)

{ Menghitung nilai b[1..m] untuk
pattern P[1..m] }

Deklarasi
k,q : interger

Algoritma:
b[1←0
q ←2
k ← 0
for q→2 to m do
   while((k>0)and(P[q]!=P[k+1]))do
      k ←b[ k]
   endwhile
   if (P[q]=P[k+1]) then
      k ←k+1
   endif
   b[q]←k
endfor
```

Fungsi tersebut akan menghasilkan output berupa array integer yang merupakan angka-angka pinggiran untuk setiap posisi iterasi pada pattern. Barulah kemudian dapat diproses pencocokkan antara pattern dan teks yang diberikan. Algoritma Knuth-Morris-Pratt selengkapnya adalah sebagai berikut:

```
procedure Algoritma KMP
(
   input m, n: integer,
   P: array [1..m] of char,
   T: array [1..n] of char
   output idx : integer
)

{ Menghitung nilai b[1..m] untuk
pattern P[1..m] }

Deklarasi
i,j  : interger
ketemu : Boolean
b : array [1..m] of integer
```

```
procedure FungsiHitungPinggiran
(
   input m: integer,
   P: array [1..m] of char,
   output b : array[1..m] of integer
)

Algoritma:
FungsiHitungPinggiran(m, P, b)
j ←0
i ←1
ketemu ←false

while (i <= n and not ketemu) do
   while((j>0) and (P[j+1]!=T[i]))do
      j ←b[j]
   endwhile

   if (P[j+1]=T[i]) then
      j ← j+1
   endif

   if j = m then
      ketemu ← true
   else
      i ← i+1
   endif
endwhile

if ketemu then
   idx ← i-m+1
else
   idx ← -1
endif
```

Kompleksitas algoritma pencocokan string dengan KMP ini dihitung dari kompleksitas untuk menghitung fungsi pinggiran dan pencarian string. Untuk menghitung fungsi pinggiran dibutuhkan waktu O(m), sementara untuk pencarian string dibutuhkan O(n), sehingga kompleksitas waktu algoritma KMP adalah O(m + n)[2].

Keuntungan dari algoritma Knuth-Morris-Pratt selain cepat juga sangat baik digunakan pada file berukuran besar karena pencarian kecocokkan tidak perlu kembali ke belakang pada input teks. Namun memiliki kekurangan yakni efektifitas dari algoritma ini akan berkurang seiring dengan bertambahnya jumlah alphabet (jenis karakter) dari teks.

## III. A MELODY SEARCH ENGINE

Penerimaan musik dari sebuah database yang besar membutuhkan kemampuan komputasi yang besar juga. Sebagai alternatif dari pendekatan pencarian dengan *brute force* di database, Musipedia lebih memilih untuk melakukan pra-pemrosesan tiap item di database dengan menghapus material yang tidak menarik. Penghapusan tersebut dilakukan dengan cara yang cerdas supaya menghasilkan pencarian yang baik dan cepat.

Musipedia menggunakan mekanisme pengurangan dan evaluasi dari musik-musik yang ada pada databasenya dengan membandingkan kebutuhan dari penyimpanan dan pembetulan dari hasil penerimaan untuk musik *query* sebelum dan sesudah dilakukannya pengurangan material musik.

Yang membedakan dari pencarian musik lainnya adalah Musipedia mampu melakukan akses terhadap file musik tidak hanya berdasarkan metadata tekstualnya saja, seperti judul lagu, artis penyanyi, composer, judul album, dan lain sebagainya. Yang membuat Musipedia ini lebih cerdas adalah kemampuannya untuk mengidentifikasi musik yang mengandung melodi pada sampelnya.

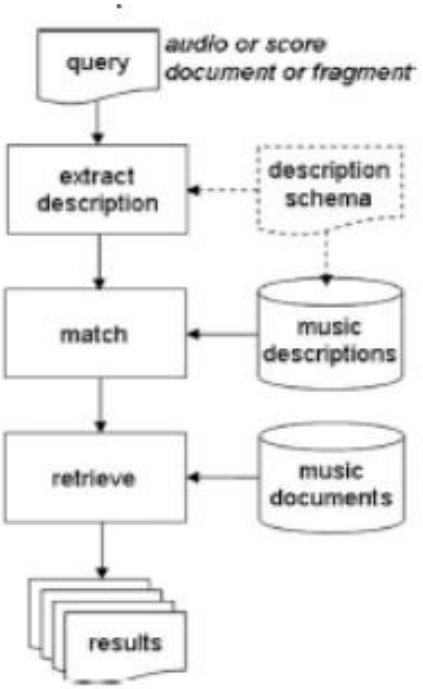
## IV. KONSEP PENGAMBILAN FILE MUSIK

Pada flowchart di bawah menunjukkan secara kasar tahapan dari konsep pengambilan file musik berdasarkan konten/isi lagunya, dan juga melodi-melodi yang ada.

File-file musik yang jumlahnya jutaan tersebut telah disimpan di dalam database berdasarkan kriteria yang telah ditentukan oleh perancang sistemnya. Telah dijelaskan sebelumnya bahwa pada database Musipedia telah dilakukan pengurangan-pengurangan material pada musik yang dirasa tidak berguna saat pencarian, sehingga menyebabkan pencarian pada database tersebut lebih cepat. Setelah file-file tersebut terkumpul, barulah pencarian file musik dapat dilakukan berdasarkan kontennya.

Pada umumnya pencarian file musik diawali dengan pengambilan query sanpel audio. Metode perekaman sampel dapat bermacam-macam, contoh yang tersedia adalah dengan cara merekam tuts *keyboard* visual pada layar, *contour search*, flash piano, rhythm search, serta perekaman dengan mikropon. Sampel-sampel audio tersebut kemudian akan diproses, dianalisis, serta dikonversi menjadi deretan string. Kemudian algoritma Knuth-Morris-Pratt baru bekerja pada tahap ini. Adanya kecocokan sampel audio dengan sumber musik itu yang akan menjadi hasil pencarian tersebut,

Dalam persoalan ini, teks algoritma Knuth-Morros-Pratt akan menganggap file-file sumber musik yang berada di database, entah itu karakteristiknya ataupun langsung bit per but dari file tersebut.




**Gambar 1 Skema alur pencarian musik**

## V. PENYIMPANAN FILE MUSIK DI DATABASE

Beberapa sistem penyimpanan file musik yang telah ada di dunia maya saat ini telah menyediakan content-based MIR. Setiap sistem merancang penyimpanan database tersebut secara berbeda-beda.

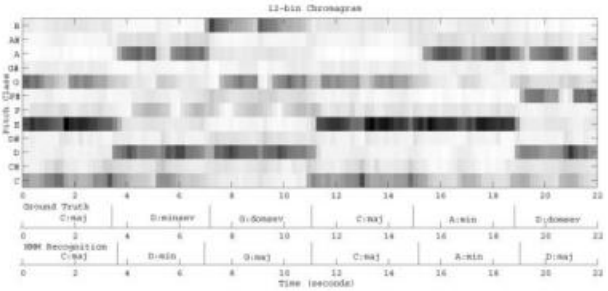
Penyimpanan file musik pada Musipedia tidak dilakukan secara utuh, database tidak menyimpan konten dari lagu secara keseluruhan. Sistem cerdas yang ada pada penyaringan material musik telah membuat kinerja dari pencarian musik semakin cepat. Sehingga tidak mengalami kesulitan pada saat pengaksesan database walaupun dilakukan dengan pencarian bit per bit file musik.

Musipedia melakukan tes pada metode *reduction* dengan bantuan mesin pencari melodi. Mesin pencari MIDI dari Musipedia.org diaplikasiakan untuk 1000 file MIDI yang juga digunakan untuk *symbolic melody similarity* pada MIREX 2006 [6]. Untuk set ini, beberapa kebenaran dasar telah diketahui, dan ini merupakan harapan bahwa mesin pencara akan dapat menerima file yang sama ketika melakukan query pada *unprocessed* dan *reduced file*. Musipedia berbasis web mencari sekitar 100.000 polyphonic MIDI file untuk melodic query dari user.

| High-level Description | Data Source | Task Description |
|---|---|---|
| Timbre | Audio | Instrument Recognition<br>Percussive, Pitched, Ensemble Recognition |
| Melody / Bass | Audio / Symbolic | Melody-line extraction<br>Bass-line extraction |
| Rhythm | Audio | Onset detection<br>Meter identification<br>Meter alignment (bars)<br>Beat (tactus) tracking<br>Tempo tracking<br>Average tempo |
| Pitch | Audio | Single fundamental freq.<br>Multiple fundamental freq. |
| Harmony | Audio / Symbolic | Chord label extraction<br>Bass-line extraction |
| Key | Audio / Symbolic | Modulation tracking<br>Pitch spelling |
| Structure | Audio / Symbolic | Verse / chorus extraction<br>Repeat extraction |
| Lyrics | Audio | Singing detection, lyrics-identification, word recognition |
| Non-Western music | Audio | Micro-tonal tuning systems<br>Non-Western canon of concepts |

**Tabel 1 contoh pemisahan karakter pada database**

Database file musik biasanya memisahkan karakteristik-karakteristik tertentu dari sebuah musik. Pada tabel di atas telah ditunjukkan contoh beberapa ciri yang ada pada musik. Karakteristik sebuah musik tersebut akan disimpan menjadi bagian-bagian string atau deretan karakter yang mewakili setiap cirri dari musik.



**Gambar 2 contoh ekstraksi dari karakteristik sebuah musik yang pada akhirnya akan berbentuk sebuah string**

Agar lebih efektif dan cepat dalam melakukan pancarian maka pelu dilakukan reduksi pada file musik tersebut, hal tersebut juga membuat file yang disimpan menjadi lebih ringan. File yang disimpan berbentuk MIDI. Reduksi dari file MIDI dapat menjadi sebuah keuntungan jika database dapat diminimalkan dengan tidak mengurangi hal positif dari query tertentu. Berikut merupakan contoh dari *reduction*:

| Query | Before reducing | Reduction 1 | Reduction 2 | Reduction 3 | Reduction 4 |
|---|---|---|---|---|---|
| q000001 | *k000258*, *k000061*, *k001051* | *k000258*, *k000061*, *k001051* | *k000061*, *k000258*, *k001051* | *k000258*, *k000061*, *k001051* | *k000258*, *k000061*, *k001051* |
| q000002 | *k004517*, *k000589*, k000076, k002413, k000034, k092493, *k092387*, k092386, k000843, *k096238* | *k004517*, *k000589*, k000836, k092948, k000076, k000034, k000861, k092493, *k092387*, k092386, k000843, *k096238* | k000836, k092948, k000076, k000916, k001012, k000034, k000861, *k092387*, k092386, k000843, *k096238*, ... | *k004517*, *k000589*, k000836, k092948, k000076, k000034, k000861, k092493, *k092387*, k092386, k000843, *k096238* | *k004517*, *k000589*, k000836, k092948, k000076, k000034, k000861, k092493, *k092387*, k092386, k000843, *k096238* |
| q000003 | *k004517*, *k000589*, k000780, ... | k000780, k000899, k096248, k000116, k000050, k092960, k092959, k000699, k000881, k000270, k092948, k000092, k000392, k000234, k092442, ... | k000780, k000899, k096248, k000116, k000050, k092960, k092959, k000699, k000881, k000270, k092948, k000092, k000392, *k000589*, *k004517*, ... | k000780, k096248, k000116, k000050, k092960, k092959, k000699, k000881, k000270, k092948, k000092, k000392, *k000589*, *k004517*, ... | k000780, k000899, k096248, k000116, k000050, k092960, k092959, k000699, k000881, k000270, k092948, k000092, k000392, k000234, k092442, ... |
| q000004 | *k095188*, *k095520* | *k095188*, *k095520* | k095520, k000811, ... | *k095188*, *k095520* | *k095188*, *k095520* |
| q000005 | *k094777*, *k096523* | *k094777*, *k096523* | *k094777*, *k096523* | *k094777*, *k096523* | *k094777*, *k096523* |
| Reduction | | 27.15 % | 36.71 % | 15.50 % | 36.34 % |
| Measure | | $H_{C,D}$ | $\frac{4}{4}(H_{C,D}) + \frac{4}{4}H_D$ | $H_C$ & $H_D$ & $H_{C,D}$ | $\frac{4}{4}(H_{C,D}) + \frac{4}{4}H_D$ |
| Window size | | 4 sec | 4 sec | 4 sec | 10 sec |
| Hop size | | 1 sec | 1 sec | 1 sec | 0.8 sec |
| Threshold | | 0.5 | 0.6 | 0.5 | 0.5 |

**Tabel 2 contoh *reduction* yang dilakukan untuk mempercepat kinerja pencarian**

## VI. PENCOCOKAN STRING DARI SAMPEL AUDIO DENGAN FILE MUSIK SUMBER

Sampel musik yang akan dicari file nya pada sumber akan dianalisis sesuai dengan karakteristik-karakteristik yang ditentukan oleh sistem pencarian tersebut. Hasil ekstraksi tersebut yang nantinya akan menjadi pattern-pattern untuk dicocokkan dengan file sumber.

Kemudian barulah pada tahap ini algoritma pencocokan string Knuth-Morris-Pratt. Penggunaan algoritma pencocokan ini tidak baku untuk penerapan pencarian file musik, namun pemilihan algoritma ini tergantung dari perancang sistem itu sendiri. Dari analisis yang telah banyak dilakukan, algoritma KMP memang relatif lebih cepat daripada algoritma lain dalam menemukan pola dalam teks.

Setelah pencarian string dalam pola musik MIDI dilakukan, baru kemudian hasil pencarian tersebut dikirimkan kepada peminta query. Hasil pencarian ini biasanya berupa metadata dari sebuah musik, misalnya judul musik, artis, serta album.

## VI. KESIMPULAN

Jika diperhatikan secara seksama peran dari algoritma pencocokkan string yang dipakai dalam content-based music information retrieval sangatlah kecil.

Faktor paling menonjol yang menentukan efektivitas dari akses ke file musik lebih cenderung kepada desain

database, penentuan karakteristik file musik, serta manajemen file musik yang baik.

Walaupun peran dari algoritma pencocokkan string ataupun pola tersebut sangat kecil, namun sedikit saja perbedaan kompleksitas antara satu algoritma dengan algoritma lainnyayang dipakai akan berdampak besar pada persoalan database musik.

Database musik relatif besar. Sekecil apapun perbedaan kecepatan untuk melakukan pencocokkan pola string itu akan sangat membantu. Terbukti ada saja sistem pencarian musik ada yang menggunakan algoritma Knuth-Morris-Pratt sebagai ujung tombak pencocokkan pola string[3].

## REFERENSI


[1] Munir, Rinaldi, "Diktat Kuliah IF3051 Strategi Algoritma", Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, 2009.

[2] http://rainer.typke.org/uploads/media/amr08.pdf, diakses pada tanggal 6 Desember 2010 pukul 21.00

[3] ]http://www.ics.uci.edu/~eppstein/161/960227.html, diakses pada tanggal 7 Desember 2010 Pukul 22.00

[4] ]http://www.cs.cmu.edu/afs/andrew.cmu.edu/course/15/354/www/postscript/kmp.pdf, diakses pada tanggal 6 Desember 2010 Pukul 21.00

[5] http://www.music-ir.org/?q=node/1 , diakses pada tanggal 5 Desember 2010 pukul 22.00

[6] http://www.music-ir.org/mirex/2006/index.php/Main Page, diakses pada 8 Desember 2010 pukul 22.28


## 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, 8 Desember 2010

Ttd

Adi Nugraha Setiadi 13508062