Penggunaan Algoritma Pattern Matching pada Music Score

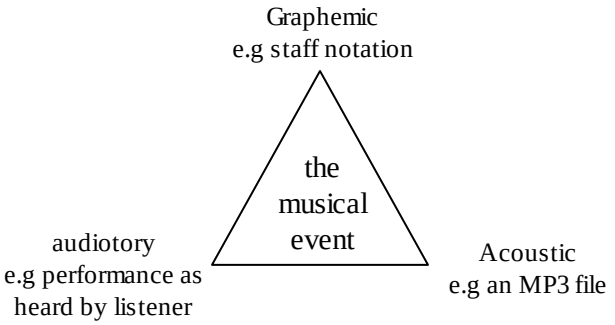
Marvello Oni (13508031)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Email : Marvello_oni@itb.ac.id, Marvello_Oni@yahoo.com

***Abstract* — Dewasa ini music sudah menjadi bagian dari kehidupan keseharian kita. Salah satu representasi music adalah *music score*. Dari berbagai macam jenis musik yang kita dengar, sering kali kita mendapati adanya bagian dari satu lagu dengan lagu lain. Hal seperti ini adalah salah satu tindakan yang tergolong dalam pembajakan yang melanggar hukum. Dalam mengatasi hal ini kita harus dapat menemukan kesamaan dari sebuah potongan lagu dengan lagu lainnya. Untuk melakukan pencarian tersebut dapat dilakukan beberapa macam pendekatan. Makalah ini dibuat untuk mengetahui apakah pendekatan menggunakan algoratima pattern matching secara spesifik algoritma Boyer-Moore dapat dimanfaatkan dalam mencari sebuah bagian dari lagu (pattern) yang menjadi potongan dari lagu lainnya.**

***Index Terms* : music score, pattern matching, pembajakan, pattern.**

I. PENDAHULUAN

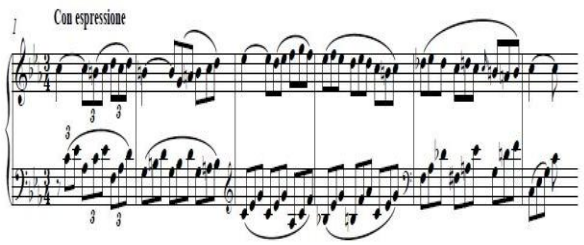
Musik dapat dibagi menjadi 3 buah bagian besar yaitu *graphemic*, *acoustic*, dan audiotary. Acoustic adalah bentuk musik yang dapat didengar, misalnya adalah MP3. Audiotaty merujuk pada music yang ditampilkan, contohnya sebuah konser musil oleh okestra. Graphemic merujuk pada music dalam bentuk tulisan atau pun gambar, contohnya adalah not balok.



Gambar 1. Diagram ini di presentasikan oleh Geraint Wiggins dalam presentasi berjudul 'Computational modeling of music cognition'

Pada Sebagian besar makalah ini istilah musik merujuk pada satu atau lebih '*music scores*' dalam bentuk tangga nada atau not balok. Not Balok adalah sebuah penulisan music, sehingga dapat dimainkan. Pada gambar 2 dapat dilihat contoh dari sebuah not balok. Pada contoh tersebut adalah not balok yang digunakan untuk alat music piano. Pada umumnya sebuah not balok terdiri atas 5 garis. 5 garis pertama pada gambar ditujukan untuk tangan kanan ditunjukan oleh gambar kunci 'G' di awal dan 5 garis kedua pada gambar ditujukan untuk tangan kiri ditunjukan oleh gambar kunci 'F' diawal. Tiga informasi awal yang ditunjukan setiap not balok adalah *clef, key-signature, time-signature*. Di sebelah kanan informasi ini adalah sekumpulan titik dengan ekor yang disebut sebagai not. Setiap not mempunyai :

1. Sebuah *ontime* (keterangan kapan sebuah not harus dimainkan bertambah dari kiri ke kanan)
2. Tinggi nada (tinggi relatih dari sebuah tangga nada
3. Durasi (diindikasikan oleh kombinasi dari isi atau kosongnya not dan tipe ekor)



Gambar 2. Bar 1-6 dari variasi ke-5 dari Sven Variations lagu 'God save the King' dalam C major, oleh Ludwig van Bethoven

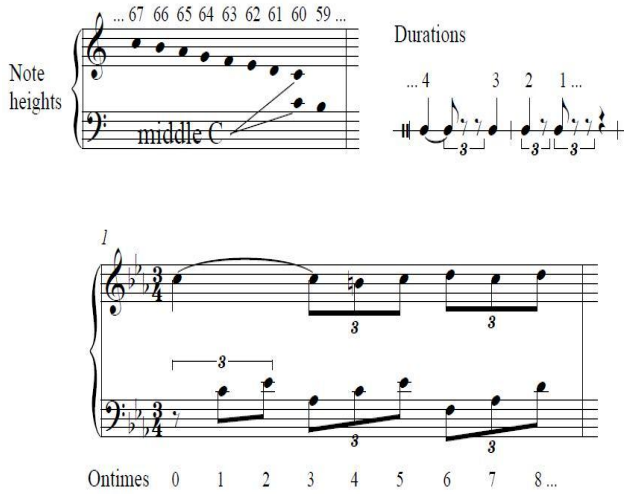
Pada makalah ini akan digunakan algoritma yang telah dikenal dan dianggap mangkus dalam menyelesaikan masalah pattern matching yaitu algoritma Booyer-Moore. Analisi yang dilakukan berupa '*query-based analysis*'. Dalam *query-based analysis* sebuah potongan atau bagian dari musik dalam hal ini adalah *pattern* dimasukan oleh *user*, kemudian program computer akan mencari potongan musik untuk *pattern* dalam *database* yang telah ada untuk sebuah potongan yang sama persis (exact matching) dengan pattern dan menampilkan hasilnya dalam urutan relevansi hasil dengan *pattern* yang dimasukan.

Dalam makalah ini lebih di fokuskan menggunakan contoh musik yang berupa monophonic (hanya melodi). Alasannya adalah karena pada monophony, not dapat diperlakukan seperti huruf dalam teks sehingga algoritma untuk pattern matching dalam teks dapat diadaptasi relative lebih mudah. Kedua pada lingkungan istilah teori-musik, monophony lebih spontan daripada polyphony, karena itu pada penelitian monophony banyak dijadikan rujuakan dalam meneliti polyphony.

II. METODE PENELITIAN

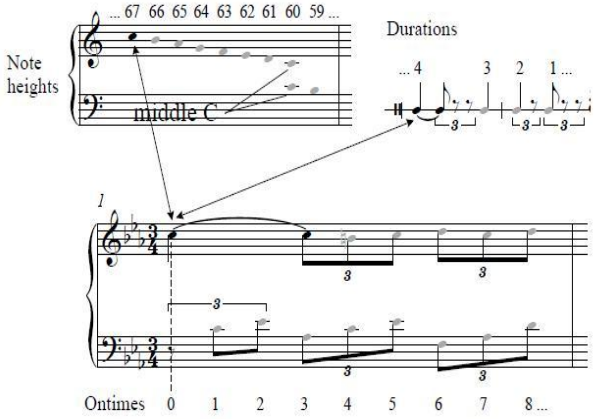
II.1 KONVERSI NOT BALOK (STAFF NOTATION)

Salah satu masalah dalam melakukan pattern matching dalam musik adalah bagi kita merepresentasikan gambar dari *staff notation* menjadi sebuah bentuk data yang dikenali oleh program. Salah satunya adalah merepresentasikan not balok menjadi sebuah *integer*. Dengan menggunakan nada dasar pertama sebagai tolak ukur maka kita dapat menentukan tabel konvensi dari not balok menjadi integer.

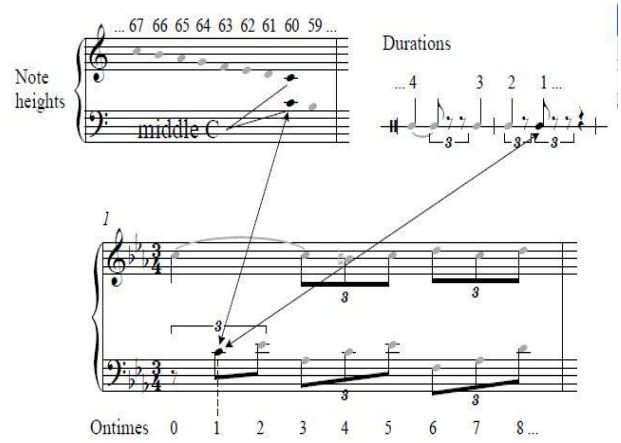


Gambar 3. Bar pertama dari gambar 2 dengan *ontime*, tinggi nada dan panjang nada yang mungkin

Pada gambar 3 dapat dilihat *ontime* direpresentasikan oleh integer dengan urutan dari 0 dan bertambah 1 untuk sebuah not yang ada. Tinggi nada direpresentasikan dengan integer dimana nada do (1 dalam not angka) direpresentasikan dengan angka 60. Sehingga nada do tinggi direpresentasikan menjadi angka 67. Kemudian panjang nada direpresentasikan sebagai integer untuk setiap ketukan.



Gambar 4. Konversi fokus pada Kunci G



Gambar 5. Konversi fokus pada Kunci F

Dengan cara demikian kita dapat mengkonversikan keseluruhan dari *music score* kedalam suatu struktur data. Konversi dilakukan terurut sesuai dengan *ontime* yang ada karena adanya kemungkinan beberapa not dimainkan pada waktu yang tepat sama. Apabila semua not pada gambar 2 dikonversikan pada menjadi sebuah point pada ruang multidimensi maka akan diperoleh sebuah *dataset*.

$$D = \{ (0,67,4), (1,60,1), (2,62,1), (3,58,1), (4,60,1), (4,66,1), \ldots , (48,60,1)\} \quad \textbf{(1)}$$

Pada representasi ini masih terdapat beberapa keterangan yang hilang, seperti lirik, dll. yang tidak dapat

direpresentasikan sebagai angka real ataupun integer. Baik gambar 2 maupun dataset yang didefinisikan pada (1) merupakan representasi simbolik dari musik. Untuk mendengarkan music yang ditunjukan oleh mereka, bentuk simbolik ini harus mencari bentuk rekaman dari lagunya atau mengubah bentuk simbolik tersebut menjadi rekaman. Dalam hal ini rekaman adalah bentuk *acoustic* dari musik.

Dengan menggunakan bentuk dataset seperti (1) maka kita akan memperoleh struktur data sebagai berikut

```
typedef struct Not {
    int Ontime;
    int KeySig;
    int TimeSig;
    } Not;

typedef struct SfaffNot{
                Not[] AllNotes;
                int Neff;
                String Title;
                }StaffNot;

Not[30] Pattern;
```

II.2 ALGORITMA BOYER-MOORE

Algoritma Boyer-Moore adalah algoritma pattern-matching dengan kompleksitas waktu sebesar O(mn) dan tabel untuk penggeseran dapat dihitung dengan kompleksitas waktu dan ruang sebesar O (n + q) dengan q adalah besar ruang alphabet. Cara kerja algoritma ini dapat dibagi menjadi 2 bagian pencarian dan bagian pergeseran. Perhitungan pergeseran dilakukan sebagai berikut, misalkan ada sebuah usaha pencocokan yang terjadi pada teks[i ... i + n – 1], dan anggap ketidakcocokan pertama terjadi diantara teks [i + j] dan pattern[j], dengan 0 < j <n. Berarti, teks[i + j + 1 ... i + n – 1] = pattern [j + 1 ... n – 1] dan a = teks[i + j] tidak sama dengan b = pattern[j]. Jika u adalah akhiran dari pattern sebelumb dan v adalah sebuah awalan dari pattern, maka penggeseran-penggeseran yang mungkin adalah :

1. Penggeseran good-suffix yang terdiri dari mensejajarkan potongan teks[i + j + 1 ... i + n + 1] = pattern[j + 1 ... n – 1] dengan kemunculan paling kanan di pattern yang didahului oleh karakter yang berbeda dengan pattern[j]. Jika tidak ada potongan seperti itu, maka algoritma akan mensejajarkan akhiran v dari teks[i + j + 1... i + n - 1] dengan awalan dari pattern yang sama.
2. Penggeseran bad-character yang terdiri dari mensejajarkan teks[i + j] dengan kemunculan paling kanan karakter tersebut di patter. Bila karakter tersebut tidak ada di pattern, maka pattern akan disejajakan dengan teks[i + n + 1]

Secara sistemasi, langkah-langkah yang dilakukan algortima Boyer-Moore pada saat mencocokan string adalah :

1. Algortima Boyer-Moore mulai mencocokan pattern pada awal teks
2. Dari kanan ke kiri, algoritma ini akan mencocokan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah sati kondisi berikut terpenuhi :
    a. Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch)
    b. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan diposisi ini.
3. Algortima kemudian menggeser pattern dengan memaksimalkan nilai penggeseran good-suffix dan penggeseran bad-character, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

Pseudo code dari Algoritma diatas adalah :

```
public static int[] buildLast(Not[] pattern)
/* Return array storing index of last
occurrence of each ASCII char in pattern. */
{
        int last[] = new int[128]; // ASCII char set
        for(int i=0; i < 128; i++)
                last[i] = -1; // initialize array
        for (int i = 0; i < pattern.length(); i++)
                last[pattern[i].KeySig] = i;
        return last;
} // end of buildLast()

public static int bmMatch(StaffNot text, Not[] pattern)
{
   int last[] = buildLast(pattern);
   int n = text.Neff;
   int m = pattern.size();
   int i = m-1;
        if (i > n-1)
        return -1; // no match if pattern is
        // longer than text       :

        int j = m-1;
        do {
          if (pattern[j].KeySig == text.AllNotes[i].KeySig)
                if (j == 0)
                        return i; // match
                else { // looking-glass technique
                        i--;
                        j--;
                }
          else { // character jump technique
                int lo = last[text.AllNotes[i].KeySig];
//last occ
```

```
                    i = i + m - Math.min(j, 1+lo);
                    j = m - 1;
                    }
          } while (i <= n-1);
          return -1; // no match
  } // end of bmMatch()
```

III. PEMBAHASAN

Dari pseduo code diatas dapat diperoleh bahwa penggunaan pattern matching terutama algoritma Boyer-Moore dapat digunakan dalam mencari pola pada musik monophonic. Hal ini dapat dilakukan dengan setelah merepresentasikan *music score* sebagai kumpulan point pada ruang multidimensi.

Banyaknya kemungkinan not yang ada (1 sampai dengan hampir 150) membuat alphabet dalam pencarian menjadi besar dan membuat kecepatan dalam pencarian menjadi lebih cepat karena besarnya pergeseran yang terjadi. Dengan cepatnya pencarian yang di lakukan algoritma ini tetap dapat diterapkan pada sebuah database yang cukup besar.

Pada pencarian perlu dibatasi panjangnya pattern. Karena apabila pattern terlalu pendek maka hasil yang muncul menjadi sangat banyak dan apabila terlalu panjang, perbedaan sebuah not saja akan membuat hasil tidak muncul. Keduanya menyebabkan hasil yang muncul menjadi tidak sesuai dengan yang diharapkan. Kedua masalah diatas merupakan akibat dari *exact matching* yang diterapkan pada algoritma.

Hanya saja dengan menggunakan konversi sederhana tersebut terdapat beberapa batasan, seperti hilangnya beberapa keterangan dari tangga nada sehingga pengkonversian kembali dari representasi struktur data ke *music score* akan menyebabkan *music score* menjadi tidak sempurna. Pada file polyphinic, konversi yang dilakukan tidak dapat diterapkan karena pada file polyphonic, musik tidak dapat dibedakan secara spesifik menjadi integer tetapi menggunakan bilangan real untuk setiap perubahan desible (dB) yang ada sehingga alphabet akan menjadi tak terhingga. Selain itu file-file polphonic mempunyai noise yang terjadi pada saat pembuatan rekaman sehingga *exact matching* hampir tidak mungkin dilakukan.

Selain itu banyaknya not dalam sebuah pratitur musik yang lengkap akan menyebabkan besarnya memori yang digunakan untuk pencarian. Apabila partitur musik yang disimpan pada database untuk pencarian merupakan partitur orkestra maka akan terjadi penambahan yang sangat besar pada penggunaan memori. Hal ini akan membuat pencarian menjadi lambat.

Selain itu batasan terbesar adalah tidak semua musik mempunyai partitur yang sempurna terutama lagu-lagu daerah dan lagu-lagu yang menggunakan alat musik yang hanya terdiri dari kunci. Dengan kata lain pemaanfaatan pattern matching menjadi terbatas hanya pada musik yang dapat dimainkan dengan alat-alat musik klasik seperti piano, biola, dll.

IV. KESIMPULAN

Algoritma pattern matching yaitu Boyer-Moore yang digunakan untuk mencari kecocokan pada suatu teks juga dapat digunakan untuk mencari pattern pada suatu lagu dengan melakukan modifikasi-modifikasi terhadap representasi lagu dan juga algortima. Penggunaan algoritma SIA, NFA, MIR, dll. dapat dilakukan dengan mengubah struktur data yang ada.

Walau masih terdapat beberapa keterbatasan. Algortima dapat menghasilkan hasil yang baik apabila pattern yang dimasukan sesuai. Pattern yang dimasukan sebaiknya adalah *theme* (bagian dari sebuah lagu yang merupakan dasar dari pembuatan, biasanya sering diulang sepanjang lagu tersebut) sehingga hasil dari pencarian akan muncul dengan akurat.

Konversi yang dilakukan mempunyai batasan tersendiri sehingga hanya dapat digunakan pada sebagian besar partitur musik klasik. Apabila ingin dimanfaatkan untuk musik lain harus dilakukan usaha untuk mengubah musik tersebut ke dalam bentuk partitur sehingga bisa dikonversi.

REFERENCES

[1] Babbitt, Milton, 'The use of computers in musicological research', in *Perspectives of New Music* 3(2) (1965), 74-83.

[2] Leman, Marc, 'Symbolic and subsymbolic description of music', in *Music processing*, ed. Goffredo Haus (Oxford: Oxford University Press, 1993), 119-164.

[3] Kennedy, Michael, *The concise Oxford dictionary of music*, 4th ed. (Oxford: Oxford University Press, 1996).

[4] McNab, R. J., L. A. Smith, I. H. Witten and C. L. Henderson, "Tune retrieval in the multimedia library," Multimedia Tools and Applications 10, 113-132, 2000.

[5] Mongeau, M. and D. Sankoff. "Comparison of musical sequences," Computers and the Humanities 24, 161-175, 1990..

[6] Meredith, D., Lemstr¨om, K., Wiggins, G.: Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. Journal of New Music Research 31(4) (2002) 321–245

[7] Cook, Nicholas, 'Perception: a perspective from music theory', in *Musical Perceptions*,eds. Rita Aiello and John A. Sloboda (Oxford: Oxford University Press, 1994), 64-95.

[8] Cope, David, *Computational models of musical creativity* (Cambridge, Massachusetts: MIT Press, 2005).

[9] Crochemore, Max, 'An optimal algorithm for computing the repetitions in a word', in *Information Processing Letters* 12(5) (1981), 244-250.
[10] Lincoln, Harry B. (ed.), *The computer and music* (Ithaca, New York: Cornell University Press, 1970).
[11] Loy, Gareth, *Musimathics: the mathematical foundations of music*, vol. 1 (Cambridge,Massachusetts: MIT Press, 2005).
[12] Downie, J. Stephen, 'Music information retrieval', in *Annual Review of Information Science and Technology* 37 (2003), 295-340.

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, 10 Desember 2010
Marvello Oni (13508031)