Penerapan Algoritma Branch & Bound Pada Melodi Akord

Adrian Hartarto Pramudita, 13515091
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, JL. Ganesha 10 Bandung, 40132, Indonesia
13515091@std.stei.itb.ac.id

***Abstrak*—Musik dapat dimainkan oleh individu. Salah satu teknik memainkanya adalah dengan menerapkan melodi akord pada sebuah lagu. Melodi akord adalah permainan sebuah lagu denga memainkan melodinya dalam bentuk akord. Terdapat banyak cara untuk memainkan melodi akord ini. Hal tersebut dapat dilihat sebagai permasalahan pencarian jalur atau pola untuk mendapatkan suatu hasil. Algoritma *branch & bound* dapat diterapkan dalam masalah ini untuk mencari bagaimana jalur yang dapat digunakan untuk memainkan melodi akord tersebut.**

Kata Kunci—akord; branch&bound; melodi; strategi algoritma; musik; melodi akord

I. PENDAHULUAN

Musik merupakan bagian dari setiap manusia. Musik sudah ada dan menjadi bagian dari manusia sejak manusia belum dilahirkan. Musik adalah bahasa yang universal karena musik mampu dimengerti oleh berbagai orang diseluruh dunia. Sebagai bahasa yang universal, tentunya semua orang dapat menikmati musik.

Terdapat banyak sekali jenis musik di dunia ini. Mulai dari *jazz, rock, pop, classic,* dllnya. Setiap orang tentunya memiliki selera dalam musik yang disukainya. Walaupun dengan berbagai macam jenis, cara memainkannya tetap sama yaitu dengan menggunakan alat musik. Bermain musik dapat dilakukan dengan 2 cara, sebagai individu dan sebagai sebuah kelompok.

Ketika memainkan musik sebagai sebuah kelompok, haromonisasi dapat dibentuk dari bermacam-macam alat musik yang dipakai dalam kelompok tersebut. Tetapi bagaimana cara orang sebagai individu dapat membuat harmonisasi musik yang baik ketika memainkan musik? Sebagai individu, ada beberapa cara bermain musik dengan membuat harmonisasi sendiri. Tidak semua alat musik dapat membuat harmonisasi ketika dimainkan sendiri, karena tidak semua alat musik dapat berperan sebagai pengiring.

Salah satu teknik dalam bermain musik ketika dilakukan individu adalah dengan memainkan melodi akord. Melodi akord ini adalah permainan *solo* seseorang dalam memainkan semua lagu. Mengkombinasikan melodi dan akord untuk mendapatkan harmonisasi. Teknik ini merupakan teknik tingkat lanjut yang tidak mudah untuk dilakukan.

Dengan teknologi yang sudah sangat berkembang di jaman ini, penulis ingin mencoba menerapkan teknologi tersebut pada musik. Penulis melihat bahwa banyak orang yang ingin mempelajari musik tetapi kesusahan dalam belajar dan mencari contoh. Dalam musik sendiri terdapat berbagai macam pola atau jalur yang dapat dipilih dalam bermain musik. Penulis melihat permasalahan ini dapat diselesaikan dengan sebuah algoritma pencarian. Oleh karena itu makalah ini dibuat untuk menunjukkan bahwa algoritma pencarian seperti *branch & bound* dapat diterapkan pada bidang musik.

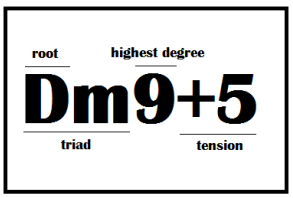
II. LANDASAN TEORI

A. Akord

Akord adalah tiga nada atau lebih yang dimainkan secara bersamaan untuk membentuk suatu harmoni. Setiap akord memiliki kualitas nada dan karakteristiknya masing-masing. Keluarga akord atau akord relatif adalah pengelompokan akord dimana setiap akord yang dimainkan pada sebuah lagu cocok untuk kunci dasar pada lagu tersebut.

[1]Akord dilambangkan dengan symbol yang memiliki 4 informasi berurutan, yaitu:

1. Akar dari akord (ditulis dengan huruf kapital)
2. Kualitas dari *triad* akord
3. Derajat tertinggi pada akord
4. Penambahan *pitches* pada akord (*tension*): +5, -5, +9, dllnya [1](bagian ini tidak selalu ada pada akord)



Gambar 1 : Akord

[1] Nettles, Barrie dan Richard Graf. 1997. *The Chord Scale Theory & Jazz Harmony*

[2]Akar dari akord merupakan dasar nada-nada yang akan dipakai untuk akord tersbut dibentuk. *Triad* akord merupakan struktur dasar akord yang paling minimum, yaitu dibentuk dari tiga nada. Interval dari nada-nada dalam akord triad disebut *third* Ada 2 tipe dari *thirds* yang digunakan dalam pembentukan triad. Yang pertama adalah *major third* (4 *half*-step) dan yang kedua adalah *minor third* (3 *half*-step). Ada 5 macam *triad* akord yaitu :

1. *Major*

   Major triad dibentuk dengan *major third* yang dihitung dari akar setelah itu dilanjutkan dengan *minor third* (1-3-5)

2. *Minor*

   Minor triad dibentuk dengan *minor third* yang dihitung dari akar setelah itu dilanjutkan dengan *major third* (1-b3-5)

3. *Suspended*

   Suspended triad dibentuk dengan *major third* yang ditambahkan atau dikurangi 1 *half-step* dihitung dari akar setelah itu dilanjutkan dengan *minor third* (1-2-5 atau 1-4-5)

4. *Diminished*

   Diminished triad dibentuk dengan *minor third* yang dihitung dari akar setelah itu dilanjutkan dengan *minor third* (1-b3-b5)

5. *Augmented*

   Augmented triad dibentuk dengan *major third* yang dihitung dari akar setelah itu dilanjutkan dengan *major third* (1-3-#5)

Derajat tertinggi merupakan penambahan nada dari *triad* akord. Dengan menambahkan *triad* akord dengan interval *major third* atau *minor third*, akan didapatkan *seventh* akord yaitu akord *natural seventh* dan akord dominan *seventh*. Kedua akord ini diperlukan untuk mendapatkan derajat tertinggi dari sebuah akord. Akord dengan derajat tertinggi lebih dari 7^{th} disebut dengan akord *extended*. Akord ini didapat dengan menambahkan nada-nada dari akar dengan memperluas penggunaan oktaf menjadi 2 oktaf. Contoh penggunaannya pada akord Cm11. Akord ini akan dibentuk dengan struktur 1-b3-5 –b7-9-11. Derajat tertinggi akord hanya ada 4 yaitu 7^{th}, 9^{th}, 11^{th}, dan 13^{th}.

Penambahan *pitches* pada akord disebut dengan *altered* akord. Penambahan ini dimaksudkan untuk memberikan penekanan pada akord. Contoh, Cm7 kettika dilakukan penambahan nada #5 akan menjadi Cm7+5 dimana struktur akordnya menjadi 1-b3-#5-b7.

Satu akord memiliki berbagai macam cara untuk dimainkan. Susunan nada-nada pada akord dapat dirubah urutannya. Hal tersebut dinamakan dengan inversi akord. Inversi akord dilakukan dengan cara memindahkan susunan nada pertama dan menjadikannya menjadi yang terakhir. Contohnya akord CMaj7 (1-3-5-7). Jika kita menginversi akord tersebut sehingga susunan nadanya menjadi (3-5-7-1). Inversi akord tersebut adalah CMaj7 *first inversion*. Akord dapat diinversi beberapa kali.

Sebuah akord juga dapat digantikan dengan akord lainnya dengan beberapa aturan. Cara mengganti akord yang paling sederhana adalah dengan mencari akord yang memiliki nada-nada pembentuk yang sama dengan akord yang ingin digantikan. Salah satu kegunaan penggantian akord ini yaitu untuk membuat sebuah lagu menjadi tidak monoton.

B. Melodi

Melodi merupakan kalimat pada lagu yang disusun dengan dari nada-nada yang dibuat teratur. Melodi merupakan salah satu unsur utama dalam sebuah sebuah lagu.

C. Melodi Akord

Dalam sebuah lagu, akord-akord digunakan untuk membuat sebuah harmoni. Cara bagaimana suatu akord digunakan dalam suatu lagu itu disebut dengan *voicing*. Setiap *voicing* selalu memperhatikan keseimbangan dan keterhubungan suatu akord dengan akord lainnya pada suatu lagu. Interaksi antar akord dan perkembangan dari akord ke akord digambarkan dengan *voice leading*. Ada 4 dasar yang perlu diketahui dalam menentukan *voice leading* dalam voicing, yaitu:

1. Tetap mempertahankan *common tone*. Jika ada nada yang sama pada akord berikutnya, sebisa mungkin pertahankan nada tersebut dalam pemakaian akord berikutnya.
2. Berpindah ke nada terdekat. Untuk berpindah ke akord selanjutnya, diusahakan berpindah dengan jarak paling pendek (nada terdekat). Jika memang harus berbeda oktaf, pilihlah interval yang paling minimum.
3. Hindari pararel oktaf. Pararel oktaf adalah keadaan dimana nada yang sama dimainkan secara bersamaan pada oktaf yang berbeda. Pararel oktaf harus digunakan jika punya tujuan dan pertimbangan tertentu seperti ingin memberikan penekanan atau menebalkan suara dari akord, karena jika tidak dengan pertimbangan, akan terjadi redundansi yang membuat akord menjadi tidak efektif.
4. Mempertimbangkan pemilihan *bass line* sebagai *countermelody*. *Bass line* harus ditentukan agar dapat sejalan dengan lagu dan karakter pembawaannya. Penentuan *bass line* sebisa mungkin harus dapat mengimbangi dari melody dan alur dari sebuah lagu.

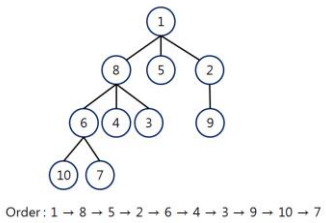
Akord-akord dalam voicing dapat diterapkan pada setiap melodi dalam sebuah lagu. Ketika sebuah lagu melodinya dimainkan dengan cara membentuk akord, hal tersebut dinamakn dengan memainkan melodi akord.

D. Breadth-First Search

[2]*Breadth-First Search* (BFS) atau metode pencarian melebar merupakan alogritma pencarian yang

[2] Munir, Rinaldi. 2007. *Diktat Kuliah IF2211 Strategi Algoritma*
[3] http://www.wimerguitar.com/Assets/WholeChordThing.pdf

direpresentasikan dengan mengorganisasikan semua kemungkinan solusi dari persoalan ke dalam suatu struktur untuk memudahkan mendapatkan solusi. Kebanyakan struktur yang dipakai dalam algoritma ini yaitu pohon berakar. Pencarian dilakukan dengan membentuk pohon dinamis. Algoritma ini bertujuan mencari solusi terpendek yang berawal dari simpul sumber ke semua simpul yang akan diperiksa untuk mendapatkan solusi.



Gambar 2 : BFS
http://stackoverflow.com/questions/14208080/breadth-first-search-query-in-mysql

Simpul-simpul dalam pohon dinamis menyatakan status persoalan. Operator yang mentransformasikan sebuah simpul ke simpul yang lain dinyatakan dengan sisi. Semua simpul-simpul yang berada pada pohon dinamis disebut ruang status. Status solusi merupakan kumpulan status yang menyatakan solusi dari persoalan. Himpunan status solusi tersebut disebut ruang solusi. Akar pohon dinamis merupakan status awal.

Pengerjaan BFS dilakukan dengan menggunakan *queue* yang menerapkan prinsip FIFO (*First In First Out*) untuk menampung simpul-simpul yang akan diekspansi. Simpul-simpul yang dimasukkan ke dalam *queue* merupakan simpul tetangga dari simpul sebelumnya yang telah dieksekusi. Ada 3 tahapan umum yang perlu dilakukan dalam memasukkan simpul-simpul ke dalam *queue*, yaitu

1. Kunjungi simpul yang belum dikunjungi (dalam hal ini adalah *head* dalam *queue*). Tandai simpul menjadi sudah dikunjungi dan hapus dari *queue*. Masukkan semua simpl-simpul yang bertetangga yang telah dikunjungi ke dalam *queue.*
2. Jika sebuah simpul pada *head* tidak memiliki simpul tetangga, hilangkan langsung dari *queue.*
3. Telusuri terus semua simpul sampai *queue* kosong atau sudah menemukan simpul tujuan.

E. Branch & Bound

[4]Branch & Bound digunakan untuk optimisasi dimana algoritma ini meminimalkan atau memaksimalkan suatu fungsi objektif yang tidak melanggar batasan yang telah dibuat. Algoritma ini menggunakan metode *Breadth-First Search* yang dikombinasikan dengan memilih *least cost search.* Pada algoritma ini dikenal fungsi pembatas. Fungsi ini digunakan untuk membatasi pencarian jalur. Ketika sebuah jalur dianggap tidak akan mengarah pada sebuah solusi, maka jalur tersebut akan dipangkas.

Fungsi pembatas ini direpresentasikan dengan sebuah nilai yang memiliki kriteria-kriteria sebagai berikut:

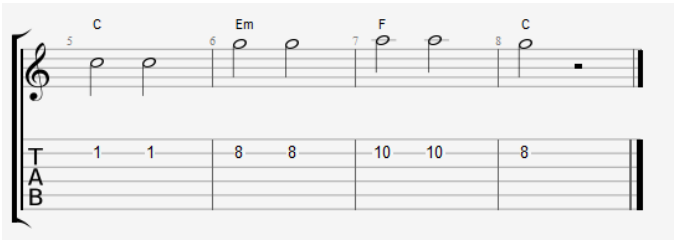
1. Nilai simpul tidak lebih baik dari nilai terbaik yang dimiliki saat ini
2. Simpul tidak merepresentasikan solusi yang *feasible* karena ada batasan yang dilanggar
3. Solusi hanya terdiri dari satu titik

F. Brute Force

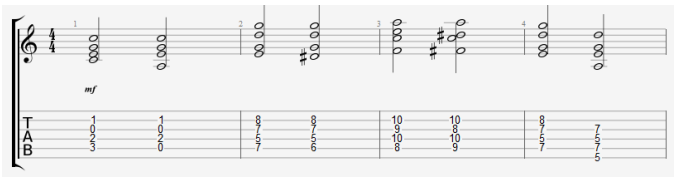
Brute Force adalah cara untuk menyelesaikan persoalan yang paling sederhana. Cara membuat algoritmanya pun sangat sederhana. Algoritma *Brute Force* didasarkan pada pernyataan pada persoalan dan konsep yang dipakai dalam suatu masalah. Hampir semua permasalahan dapat diselesaikan dengan algoritma *brute force*. Tetapi terkadang algoritma ini kurang efektik shingga memakan waktu dan ruang yang cukup banyak.

III. Penerapan Algoritma

Branch & Bound digunakan untuk mencari akord apa saja yang harus dipakai dalam membentuk suatu melodi akord. Sebuah lagu memilik melodi dan *voicing* dimana *voicing* tersebut dapat diterapkan untuk setiap melodi yang berada pada lagu tersebut. Contoh pada lagu *twinkle-twinkle*



Gambar 3 melodi dan akord dari lagu *twinkle-twinkle*



Gambar 4 akord melodi pada lagu *twinkle-twinkle*

A. Penerapan Algoritma Brute Force

Algoritma *Brute Force* digunakan untuk mencari semua akord yang mungkin dari suatu melodi. Akord-akord tersebut disimpan dalam sebuah array. Ada beberapa fungsi yang digunakan, yaitu:

1. Procedure searchChord(melodi : Integer, chordLib : Array of String, var arrayChord : Array of String)

   Prosedur ini digunakan untuk mengisi *array* yang isinya merupakan semua akord yang dapat digunakan untuk memainkan melodi yang diinginkan.

   chordLib merupakan *array* yang menampung semua akord yang dapat dibentuk dari 12 nada yang ada. Pada prosedur ini, dilakukan iterasi sebanyak jumlah elemen

yang berada pada chordLib. Disetiap iterasi dilakukan pemanggilan fungsi isChordMatch, jika hasil dari isChordMatch adalah *true*, maka elemen chordLib yang dibandingkan akan dimasukkan kedalam arrayChord.

Contoh Pseudocode:

```
Procedure searchChord(melodi : Integer, chordLib : Array of String, var arrayChord : Array of String)
Var
    idx, idxArr : Integer
Algoritma
    idx ← 0
    idxArr←0
    while (idxArr < ARRAYMAKS AND idx < chorLib.length) do
        if ((IsChord(chordLib[idx],melodi)) then
                arrayChord[idxArr] ← GetName(chordLib[idx])
                idxArr++
        idxArr++
```

2. Function isChordMatch(chordPattern : String, melodi : Integer) : Boolean

   Fungsi ini mengembalikan nilai *boolean* dimana menyatakan apakah suatu melodi merupakan *leading tone* dari suatu akord atau bukan.

3. Function getChordName(chordPattern : String) : String

   Fungsi ini mengembalikan sebuah *string* yang merupakan nama akord dari nada-nada yang diketahui. Masukkan nada-nada yang diketahui berupa string dengan nama *chordPattern*.

Pada tahap selanjutnya arrayChord yang didapat dari algoritma ini akan digunakan sebagai sumber simpul dari algoritma *branch & bound.*

B. Penerapan Algoritma Branch & Bound

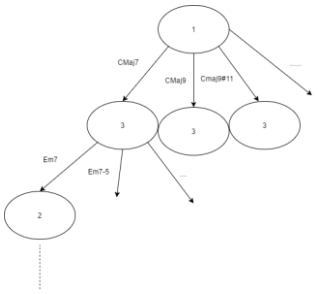
Fungsi pembatas yang dipakai algoritma ini dalam persoalan melodi akord menggunakan aturan *voice leading*. Aturan tersebut dibuat menjadi penilaian untuk batas apakah akord tersebut cocok dipakai atau tidak dalam melodi yang menjadi masukkan. Fungsi Pembatas dapat dihitung sebagai berikut:

$$F_p = S(i) + C(i) + R(i)$$

1. Sebuah akord jika memiliki *commont tone* dengan akord sebelumnya ditambahkan dengan nol.
2. S(i) merupakan nilai bobot simpul sebelumnya. Untuk simpul awal nilai S(i) = 0.
3. C(i) merupakan setiap nada yang berubah dari akord sebelumnya ditambahkan dengan nilai satu per *step*. Dalam hal ini step adalah banyaknya perpindahan setengah nada dari nada awal.
4. R(i) merupakan nilai jika terdapat 2 nada yang sama pada sebuah akord tetapi dalam oktaf yang berbeda. Nilai pembatasnya ditambahkan dengan nilai 12 (sebanyak nada yang telah dilewati).

Fungsi pembatas tersebt digunakan untuk membentuk pohon melodi akord. Pembentukanpohon melodi akord dibuat dengan memanfaatkan algoritma *brute force* sebelumnya. Semua simpul yang memungkinkan dalam akord akan menjadi simpul tetangga yang dapat dikunjungi. Berikut langkah-langkah algoritma *branch & bound* yang digunakan.

1. Petama dilakukan pencarian simpul untuk nada pertama. Simpul ini dicari dengan menggunakan prosedur searchChord pada algoritma *brute force.* Setelah mendapatkan semua kemungkinan, dipilih satu dari semua kemungkinan untuk dijadikan akar dari pohon yang ingin dibuat. Bobot dari akar dihitung dengan fungsi pembatas yang dimiliki.
2. Setelah mendapatkan akar, dicari simpul-simpul yang dapat dituju dengan memanfaatkan prosedur searchChord pada algoritma *brute force.* arrayChord yang didapat adalah semua simpul yang dapat dituju dari simpul awal.
3. Pilih simpul dengan bobot terkecil. Jika ada simpul dengan nilai bobot yang sama, pilih simpul secara acak.
4. Lakukan langkah 3 dan 4 sampai tingkatan pohon sejumlah dengan banyaknya melodi dalam dari masukkan yang didapat.
5. Ketika mencapai tingkat akhir, cari semua simpul yang memungkinkan dengan menggunakan prosedur searchChord pada algoritma *brute force.* Hal ini ditujukan untuk mendapat akord apa yang terakhir dapat dipakai
6. Setelah mendapatkan solusi, simpan semua solusi dalam *array list* yang menyimpan perjalanan apa saja yang memungkinkan dapat digunakan dalam memainkan melodi tersebut. Untuk penyederhanaan diplih semua simpul dengan bobot terkecil.



Gambar 5 contoh pohon yang dibentuk

Setiap simpul melambangkan melodi yang dimainkan. Sisi melambangkan akord apa yang dimainkan pada simpul sebelumnya.

IV. Analisis Algortima

A. Analisis

Pada *brute force* yang diterapkan, waktu untuk mencari akord bisa sangat lama. Kompleksitasnya yaitu O(n). Jumlah iterasi yang dilakukan itu sebanyak kombinasi akord yang dimungkinkan.

3 nada = ${}_{12}C_3 = 220$

4 nada = ${}_{12}C_4 = 495$

5 nada = ${}_{12}C_5 = 792$

6 nada = ${}_{12}C_6 = 924$

7 nada = ${}_{12}C_7 = 792$

8 nada = ${}_{12}C_8 = 495$

9 nada = ${}_{12}C_9 = 220$

10 nada = ${}_{12}C_{10} = 66$

11 nada = ${}_{12}C_{11} = 12$

12 nada = ${}_{12}C_{12} = 1$

Dengan kombinasi nada tersebut, total akord yang dapat dibentuk adalah 4017 buah akord. Jumlah akord tersebut masih bisa disederhanakan karena terdapat akord yang cara memainkannya sama tetapi memiliki nama yang berbeda.

Pada algoritma *branch & bound* dilakukan pencarian semua solusi yang dimungkinkan untuk membuat melodi akord dari masukkan yang didapat. Fungsi pembatas yang didapat digunakan untuk membatasi akord yang sudah terlalu jauh dari tangga nada yang dipakai pada melodi akord. Namun fungsi pembatas yang dipakai masih belum optimal. Hal ini dikarenakan fungsi pembatas yang dipakai adalah aturan dalam musik yaitu *voice leading* dimana aturan tersebut hanya mengatur tentang *voicing* pemakaian akord yang seharusnya dalam sebuah lagu. Dengan aturan tersebut, masih dimungkinkan ada akord yang sama tetai berbeda nama. Hal itu bisa didapatkan karena setiap akord memiliki bentuk inversi dan akord pengganti.

Pada pencarian semua kemungkinan solusi juga memakan waktu dan ruang yang sangat banyak. Algoritma yang dibuat tidak menggunakan pembatas untuk jumlah *path* yang dapat dipilih. Dengan kombinasi 4017 akord dan banyaknya melodi yang ingin dicari. Kompleksitasnya menjadi $O(h^n)$ dimana h adalah tingkatan (banyaknya melodi) daroi pohon yang dibentuk, dan n adalah jumlah akord yang memungkinkan yang didapat dari algoritma *brute force*.

V. Kesimpulan

Algoritma *branch & bound* digunakan untuk mencari *path* optimum pada suatu persoalan. Algoritma ini juga bisa digunakan dalam bidang musik. Penggunaannya dalam bidang musik juga tetap pada mencari *path* yang tepat dalam memilih keputusan akord yang ingin dipakai. Salah satunya adalah melodi akord.

Melodi akord dapat dicari dengan menggunakan algoritma *branch & bound*. Banyak kombinasi dan *path* yang dapat dipilih dari hasil penyelesaian dengan algoritma *branch & bound*. Tetapi dengan algoritma dan cara penyelesaian diatas masih belum optimum hasil yang didapatkan karena pada hasil yang didapat masih ada *path* yang secara teori merupakan hal (akord inversi/*substitusi*) yang sama. Masih harus dilakukan pembenaran pada fungsi pembatas yang digunakan pada algoritma *branch & bound*.

Solusi yang didapat juga memiliki banyak variasi karena jumlah bobot minimum yang dihitung oleh fungsi pembatas terdapat banyak yang sama. Ini mengakibatkan ruang yang dipakai dalam penyimpanan juga sangat besar. Untuk mengatasi hal ini bisa dilakukan dengan memilih salah satu solusi saja.

VI. Ucapan Terimakasih

Pertama-tama penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa karena oleh-Nya penulis dapat menyelesaikan tulisan ini. Penulis berterima kasih kepada dosen Strategi Algoritma IF2211 yaitu Pak Rinaldi Munir, Bu Masayu Leylia Khodra, dan Bu Nur Ulfa Maulidevi yang telah mengajarkan dasar-dasar strategi algoritma yang penulis gunakan dalam tulisan ini. Penulis mengucapkan terima kasih juga kepada unit kegiatan mahasiswa ITBJazz yang telah memberikan masukan-masukan bagi penulis.

Referensi

[1] Saskatchewan. *Jazz Theory*. Ministry of Education

[2] Fuentes, David. 2010. *Figuring Out Melody*

[3] O'Rourke, Greg. 2016. *Jazz Guitar Chord Melody Basics*

[4] Nettles, Barrie dan Richard Graf. 1997. *The Chord Scale Theory & Jazz Harmony*

[5] Smith, Stuart. 2008. *Jazz Theory 4th Revised edition*

[6] Prong, Bob. 2011. *A music theory reference guide for guitar*

[7] Munir, Rinaldi. 2007. *Diktat Kuliah IF2211 Strategi Algoritma*

[8] Levitin, Anany. 2011. *Introduction to the Design and Analysis of Algorithms (3rd Edition)*. United State: Pearson.

[9] https://www.khanacademy.org/computing/computer-science/algorithms/breadth-first-search/a/breadth-first-search-and-its-uses (diakses pada 16 Mei 2017)

[10] https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/ (diakses pada 16 Mei 2017)

[11] https://www.tutorialspoint.com/data_structures_algorithms/breadth_first_traversal.htm (diakses pada 16 Mei 2017)

[12] https://web.stanford.edu/class/ee364b/lectures/bb_slides.pdf (diakses pada 16 Mei 2017)

[13] https://www.vocabulary.com/dictionary/chord (diakses pada 17 Mei 2017)

[14] http://www.musiceducatorsinstitute.com/course/guitar/course3/M02S01_relative_chords.html (diakses pada 17 Mei 2017)

[15] http://www.wimerguitar.com/Assets/WholeChordThing.pdf (diakses pada 17 Mei 2017)

[16] http://www.cyberflotsam.com/Music_ExtendedChords1.htm (diakses pada 17 Mei 2017)

[17] http://www.gmajormusictheory.org/Fundamentals/Ch12.pdf (diakses pada 17 Mei 2017)

[18] http://www.worshipteamcoach.com/upload/documents/download_content/inversions-workshop_opt.pdf (diakses pada 17 Mei 2017)

[19] https://www.premierguitar.com/articles/19696-digging-deeper-how-many-chords-are-there (diakses pada 18 Mei 2017)

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, 16 Mei 2017

Adrian Hartarto P
13515091