Pembuatan Motif Musik Sederhana Berpola dengan Algoritma *Backtrack*

Ferdiant Joshua Muis - 13516047
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516047@itb.ac.id

***Abstract*—Dalam dunia musik, pembuatan musik yang tersusun atas ritme dan nada tertentu tidak pernah berhenti. Musik sampai sejauh ini tidak dapat dipetakan dengan suatu teori atau algoritma nyata untuk membuat suatu lagu yang dipastikan indah. Namun sesungguhnya setiap lagu yang nyaman bagi manusia untuk didengar memiliki pola tertentu yang disebut dengan motif. Pada makalah ini penulis akan membuat suatu program yang dapat membuat pola/motif dengan melanjutkan pola/motif yang ada dari suatu potongan lagu dengan algoritma *exhaustive search* dan *backtrack*.**

Kata kunci---musik, ritme, motif, exhaustive search, backtrack.

I. PENDAHULUAN

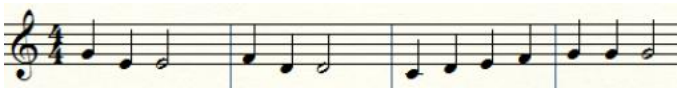
Dalam kehidupan manusia sejauh ini, musik sudah ada, hidup, dan digunakan dari sejak waktu yang lama. Musik yang terus berkembang memiliki berbagai fungsi untuk membantu kehidupan manusia. Fungsi tersebut dimulai dari rekreasi, ritual, atau bahkan sebagai penyemangat dalam bekerja. Musik kemudian terus dikembangkan oleh manusia, hingga mulai pemetaan dan pembelajaran strukturnya oleh orang-orang yang mendalami musik ,yang kemudian menjadi ahli musik, mulai dari abad 14.

Oleh masyarakat dunia, musik yang berada dalam bidang seni seringkali dipisahkan dari keilmuan eksak seperti matematika, informatika, dan lain-lain. Padahal sesungguhnya seni yang adalah musik itu sendiri dapat dipetakan jika didalami dengan dan diteliti dengan lebih baik. Komposisi musik yang terlihat dan terdengar selalu berbeda, sebenarnya memiliki pola-pola tertentu. Inilah yang membedakan suatu musik terdengar nyaman atau tidak di telinga manusia.

Tidak heran, sejak pertengahan abad 20 ketika komputer mulai ditemukan dan ilmu informatika mulai berkembang, semakin banyak orang yang berusaha membuat algoritma untuk mengomposisi musik. Sebab ketika diteliti lebih dalam, musik-musik yang melegenda hingga saat ini (seperti karya Mozart, Beethoven, dan komposer-komposer lainnya) memiliki pola yang teratur dan digunakan kembali hampir di setiap komposisinya.

Pola yang terdapat dalam musik secara luas dan garis besar dapat dibedakan menjadi 2 bagian besar, yaitu melodi dan ritmis. Pola melodis adalah pola yang berhubungan dengan *pitch, interval*, dan tangga nada. Sedangkan pola ritmis adalah pola yang berhubungan dengan durasi setiap not, penyusunannya dan metode-metode penggunaan ulangnya.

Pada makalah ini, penulis akan membuat program untuk melakukan pengembangan pola/motif ritmis suatu potongan lagu. Yaitu melanjutkan suatu pola/motif yang diberikan (pada makalah ini kasus dibatasi hanya menangani lagu dengan 4 *beat)*. Pola yang dilanjutkan ini memang masih sangat sederhana dan mungkin belum nyaman bagi telinga manusia untuk didengar, tetapi dari pembentukan pola ini, pengembangan dan pembentukan inspirasi dapat mulai dipicu untuk membuat pola/motif lagu yang nantinya akan dan dapat dijadikan suatu kesatuan komposisi lagu yang utuh. Pembentukan pola ini juga dapat menjadi awal dari komposisi otomatis (komposisi dengan *Artifiial Intelligence*).



Gambar 1. Motif lagu Si Semut (4 bar)

II. DASAR TEORI

A. Algoritma Backtrack

Algoritma *backtrack* adalah algoritma untuk menemukan satu atau banyak solusi dari suatu masalah komputasi. Algoritma *backtrack* bekerja dengan prinsip DFS (*Depth-First Search*) dan merupakan perbagian dari algoritma *brute force*. Algoritma ini berusaha mencari setiap kemungkinan yang dapat terbentuk, tetapi segera mundur (melakukan *backtrack*) dan meninggalkan suatu kandidat solusi ketika kandidat solusi tersebut sudah dipastikan tidak mendekati solusi

Contoh kasus penerapan algoritma *backtrack* yang paling umum adalah kasus *N-Queen Problem.Backtrack* merupakan salah satu alat penting untuk menyelesaikan berbagai masalah komputasi seperti pemenuhan solusi berbatas (*constraint satisfaction problem*).

Algoritma *backtrack* secara dasar memiliki properti umum sebagai berikut :

1) Solusi Persoalan

Solusi persoalan *backtrack* dinyatakan sebagai vektor dengan tuple sebanyak n, dengan n adalah jumlah persoalan.

$$X= (x_1, x_2, \ldots, x_n), x_i \in Si.$$

2) Fungsi Pembangkit

Fungsi pembangkit dinyatakan sebagai predikat T(k). Fungsi pembangkit ini berguna untuk membangkitkan nilai untuk x_k yang adalah bagian dari vektor solusi.

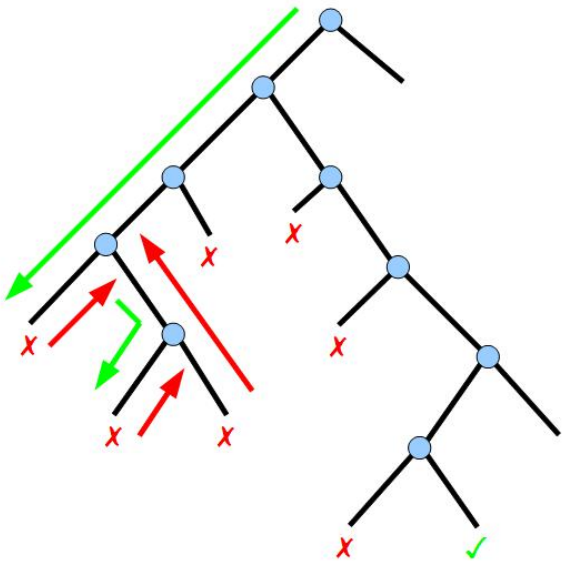
3) Fungsi Pembatas

Fungsi pembatas dinyatakan sebagai predikat

$$B(x_1, x_2, \ldots, x_k)$$

B menghasilkan nilai *true* jika setiap x dalam B mengarah ke solusi, dan menghasilkan nilai *false* jika ada nilai x yang tidak mengarah ke solusi.

Jika hasil B adalah *true* maka nilai x_{k+1} akan dibangkitkan, tetapi jika hasilnya adalah *false* maka nilai tersebut akan dibuang.



Gambar 2. Ilustrasi algoritma backtrack
(sumber slide Algoritma Runut-balik Rinaldi Munir)

B. Komponen Dasar Ritmis Musik

1) Ritme (Rhythm)

Ritme adalah elemen waktu dalam musik. Ritme disebut juga peletakan suara dalam komponen waktu. Ritme secara umum terdiri atas :

a) Durasi

Durasi menentukan seberapa lama bunyi suatu not bertahan sebelum hilang.

b) Beat

Satuan unit waktu terkecil dalam musik. Dalam makalah ini hanya dibahas pola/motif dengan 4 *beat.*

c) Tempo

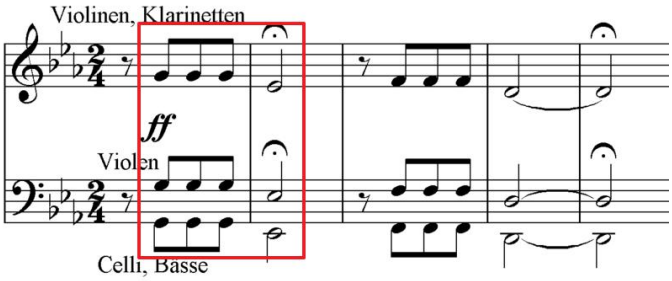
Kecepatan dari *beat,* yang dapat didefinisikan sebagai jumlah *beat* per detik.

d) Meter

Durasi menentukan seberapa lama bunyi suatu not bertahan sebelum hilang.

2) Figur

Figur adalah ide tersingkat dan paling sederhana dalam musik, merupakan kumpulan rentetan pendek dari beberapa not dan umumnya berulang. Figur tersusun oleh nada, dan/atau ritme, dan melakukan modifikasi (penulis istilahkan "bermain-main") dengan kedua komponen tersebut untuk membentuk pola dasar dalam suatu komposisi.



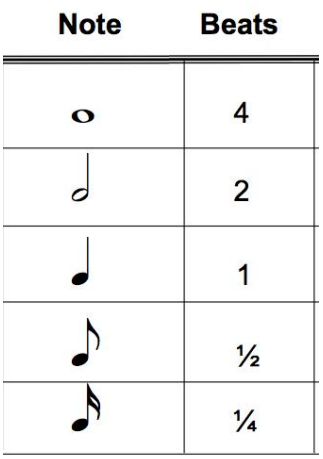
Gambar 3. Figur yang sangat terkenal dari komposisi Simfoni Kelima dari Beethoven

3) Motif

Motif adalah komponen ide singkat dalam musik, yang adalah pengulangan dari figur. Figur yang dipakai kembali umumnya diberikan sedikit dimodifikasi untuk membangun struktur musik yang berbeda tetapi tetap berpola. Motif adalah bentuk lebih luas dari figur, dan merupakan komponen yang berinteraksi langsung (membuat manusia merasa nyaman atau tidak) dengan manusia. Makalah ini membahas pembuatan motif.

4) Nilai Not

Dalam musik setiap not memiliki nilai ritmisnya sendiri. Berikut adalah representasi not balok dengan nilai ritmis notnya :



Gambar 4. Representasi not balok dengan nilainya dalam satuan beat

Sumber (https://id.pinterest.com/pin/77968637270713288/?lp=true)

Sebenarnya terdapat lebih dari 5 jenis not, tetapi makalah dan program yang dibuat hanya mencakup 5 not di atas untuk menyederhanakan persoalan.

III. Implementasi Algoritma Backtrack dalam Membuat Motif

Penyelesaian persoalan pembuatan (pelanjutan) pola/motif musik dilakukan dengan algoritma *backtrack.* Pengguna memasukkan sejumlah *n* nilai not (harus bernilai 0.25, 0.5, 1, 2, atau 4) yang adalah representasi sebuah motif. Jumlah setiap nilai not ini harus sama dengan 4 (karena cakupan masalah yang dibahas hanya pola/motif dengan 4 *beat*). Kemudian algoritma *backtrack* akan mencari bentukan motif lain yang mirip dengan motif yang diberikan oleh pengguna.

Jumlah solusi yang akan dihasilkan pada program ini sejumlah jenis nilai not. Yaitu setiap pola dimulai dengan masing-masing jenis nilai not.

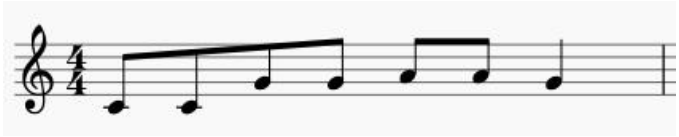
Pada program ini didefinisikan sebuah pola mirip jika pola yang baru :

1) Jumlah semua nilai not adalah sama dengan pola asli (dalam kasus ini selalu 4)
2) Memiliki setiap kategori not dengan jumlah minimal setengah dari jumlah setiap not pada pola asli
3) Memiliki setiap kategori not dengan jumlah maksimal sama dengan jumlah setiap not pada pada pola asli

Tiga poin di atas sekaligus menjadi fungsi pembatas bagi algoritma *backtrack.*

B. Representasi Motif dalam Program

Pengguna memasukkan sejumlah bebas nilai not dalam larik (disimpan dalam file eksternal) yang kemudian akan dibaca dan disimpan oleh program. Berikut adalah contoh memasukkan 4 *beat* pertama lagu Twinkle-twinkle Little Star :



Gambar 5. empat beat pertama dari lagu Twinkle-twinkle Little Star (Sumber : milik pribadi)

Index	0	1	2	3	4	5	6
Nilai not	0.5	0.5	0.5	0.5	0.5	0.5	1

Gambar 6. Representasi dalam larik (Sumber : milik pribadi)

Terdapat 6 not bernilai 0.5 dan 1 not bernilai 1 (lihat gambar 4) dengan urutan seperti yang tergambar. Maka representasi ketujuh not tersebut dalam larik terbentuk seperti pada gambar 6. Perhatikan bahwa jumlah seluruh nilai not akan selalu sama dengan 4.

C. Algoritma Backtrack

Solusi yang diperoleh berupa tuple yaitu kumpulan-kumpulan not dengan nilai not-nya masing-masing.

Fungsi pembangkit dalam program ini dibuat dengan fungsi rekursif, yaitu program akan mencoba untuk membangkitkan semua nilai not yang memungkinkan (pada kasus ini 0.25, 0.5, 1, 2. 4)

Fungsi pembatas adalah yang didefinisikan pada awal bagian III. Berikut adalah rincian dari setiap fungsi pembatas :

1) Jika hasil pembangkitan simpul menghasilkan total nilai not yang lebih dari 4, maka *backtrack* dilakukan dan pembangkitan simpul dihentikan.
2) Jika hasil pembangkitan simpul menghasilkan salah satu kategori not menjadi tidak mungkin terpenuhi jumlah maksimalnya, maka *backtrack* dilakukan dan pembangkitan simpul tersebut dihentikan.
3) Jika hasil pembangkitan simpul menghasilkan salah satu kategori not memiliki jumlah lebih dari jumlah maksimum yang diizinkan, maka *backtrack* dilakukan dan pembangkitan simpul tersebut dihentikan.

D. Implementasi Algoritma Backtrack

1) Struktur Data

Berikut adalah pseudocode dari keseluruhan program pembuatan pola/motif :

```
STRUKTUR DATA

input : array of integer
output : array of array of integer
rhythm_type_count_min : array of integer
```

```
rhythm_type_count_max : array of integer
```

Variabel input menyimpan menyimpan data masukkan pengguna yang diperoleh melalui file eksternal bertipe .json. Berikut adalah contoh input file eksternal :

```
{
  "Rhythm":[0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1]
}
```

Variabel output menyimpan semua hasil *backtrack* (yang adalah larik), oleh karena itu variabel output bertipe larik 2 dimensi.

Variabel rhythm_type_count_min menyimpan jumlah minimal not yang harus dibangkitkan berdasarkan masukkan pengguna dalam bentuk larik, yang nilainya berdasarkan fungsi pembatas yang sudah dijelaskan pada bagian awal bab III

Variabel rhythm_type_count_max menyimpan jumlah max not yang harus dibangkitkan berdasarkan masukkan pengguna dalam bentuk larik, yang nilainya berdasarkan fungsi pembatas yang sudah dijelaskan pada bagian awal bab III

2) Fungsi Utama

Berikut adalah fungsi Validate, yang berfungsi sebagai fungsi pembatas, yaitu memvalidasi apakah simpul yang dibangkitkan mendekati solusi atau tidak.

Pemeriksaan dilakukan berdasarkan pembandingan jumlah jenis not yang sudah dibangkitkan dengan variabel rhythm_type_count_max dan rhythm_type_count_min, dan total nilai not harus sama dengan 4. Semua nilai pembandingan didapatkan dari parameter fungsi.

```
Function Validate(output,
rhythm_type_count_min, rhythm_type_count_max)
-> boolean

KAMUS

   remainder : integer
   reserved_value : integer

ALGORITMA

   remainder = 4 - output.sum

   Kurangi setiap elemen pada
   rhythm_type_count_max dan
   rhythm_type_count_min sejumlah jenis not
   yang sudah dibangkitkan

   reserved_value = jumlah beat total yang
   belum terpenuhi (iterasi setiap elemen
   rhythm_type_count_min dan kalikan dengan
   nilai notnya)

   if (remainder < 0 or remainder <
     reserved_value) then
         return False

    else

      if (ada elemen rhythm_type_count_max
      yang bernilai < 0 OR remainder + setiap
      kemungkinan rhythm_count_min != 4) then

         return False

      else

         return True
```

Berikut adalah prosedur rekursif CalculateRec yang mengalkulasi setiap kemungkinan secara *brute force* tetapi setiap kalkulasi hasilnya diperiksa dahulu dengan fungsi validate. Agar segera dilakukan *backtrack* jika pembangkitan simpul tersebut sudah dipastikan gagal dan tidak mendekati solusi.

Fungsi ini memasukkan jenis not satu demi satu dimulai dari parameter rhythm_value (yang menyimpan nilai not yang sedang dicoba), dan beriterasi ke nilai not yang selanjutnya ketika semua kemungkinan telah dicoba (jika rhythm_value dimulai 0.25, iterasi selanjutnya adalah mencoba 0.5, 1, 2, 4, dan kembali ke 0.125). Hasil disimpan pada variabel global, oleh karena itu perhitungan ini dilakukan oleh prosedur.

```
Procedure CalculateRec(result, rhythm_value,
rhythm_type_count_min, rhythm_type_count_max)

KAMUS

     next_rhythm_value : integer
     total : integer


ALGORITMA

    total = total nilai not dari result

     if(total != 4) then
       append rhythm_value ke dalam result
       if(Validate(…)) then
            CalculateRec(result,…)
       else
           pop elemen terakhir result (backtrack)
            CalculateRec(result,…)

     else
       append result ke dalam variabel global
       output
```

Berikut adalah prosedur Calculate yang memanggil CalculateRec sejumlah jumlah jenis not dengan parameter rhythm_value yang berbeda-beda. Prosedur ini bertujuan agar keluaran program menjadi beberapa jenis. Yaitu sejumlah jumlah jenis not (hasil yang not pertamanya adalah not dengan nilai 0.125, hasil yang not pertamanya adalah not dengan nilai 0.25, dst)

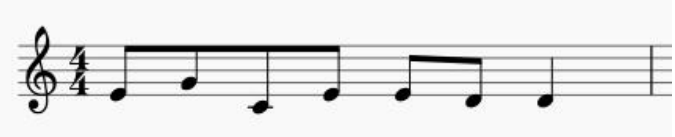
```
Procedure Calculate(rhythm_type_count,
rhythm_type_count_min, rhythm_type_count_max)

ALGORITMA

     for i = 0 to length(rhythm_type_count) do
       rhythm_value = 2 ^ (i-2)
       CalculateRec([],          rhythm_value,
```

```
        rhythm_type_count_min,
        rhythm_type_count_max)
```

IV. Hasil Percobaan dan Analisis

Berdasarkan program yang sudah dijelaskan dan dirancang pada bab III, penulis mengeksekusi program, meneliti dan menganalisis hasil yang terbentuk.

Berikut adalah contoh masukkan yang diberikan :



Gambar 7. empat beat pertama dengan 7 not sebagai sampel masukan (Sumber : milik pribadi)

Index	0	1	2	3	4	5	6
Nilai not	0.5	0.5	0.5	0.5	0.5	0.5	1

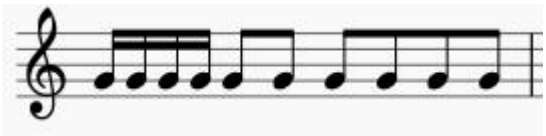
Gambar 8. Representasi masukkan dalam larik (Sumber : milik pribadi)

A. Hasil Keluaran Program

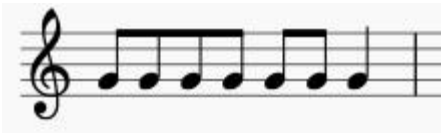
Setelah progam dieksekusi, maka hasil yang ditampilkan program adalah :

```
{'Rhythm': [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1]}


Execution time : 0.22342 seconds.

Result :

Kemungkinan 1 : [0.25, 0.25, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5]
Kemungkinan 2 : [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1]
Kemungkinan 3 : [1, 2, 0.25, 0.25, 0.25, 0.25]
Kemungkinan 4 : [2, 0.25, 0.25, 0.25, 0.25, 0.5, 0.5]
```

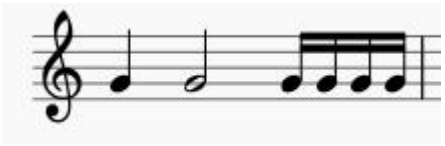
Berdasarkan hasil di atas, maka 4 kemungkinan yang terbentuk adalah :



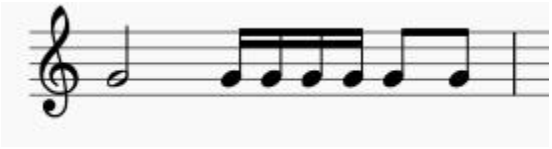
Gambar 9. Hasil Kemungkinan 1



Gambar 10. Hasil Kemungkinan 2



Gambar 11. Hasil Kemungkinan 3



Gambar 12. Hasil Kemungkinan 4

Program ini mengabaikan melodi, sehingga hasil yang diperoleh hanya ritmenya saja (penulis memilih suatu not acak sebagai representasi not) Dari hasil keluaran program, terlihat bahwa hasil keluaran yang terbentuk sudah cukup terstruktur.

B. Analisis

Dari hasil keluaran program, terlihat bahwa hasil *brute force* dan *backtrack* berhasil membuat pola/motif baru berdasarkan pola/motif yang diberikan. Pola yang terbentuk juga tidak begitu acak dengan digunakannya batasan yang dibahas pada bab III yaitu :

1) Jumlah semua nilai not adalah sama dengan pola asli (dalam kasus ini selalu 4)

2) Memiliki setiap kategori not dengan jumlah minimal setengah dari jumlah setiap not pada pola asli

3) Memiliki setiap kategori not dengan jumlah maksimal sama dengan jumlah setiap not pada pada pola asli

Program ini jika tidak menggunakan optimisasi *backtrack* akan memiliki kompleksitas :

$$O(n) = 4 * (n+m+p+l+q)^4$$

Dengan :
n = jumlah not dengan nilai 0.25
m = jumlah not dengan nilai 0.50
p = jumlah not dengan nilai 1.00
l = jumlah not dengan nilai 2.00
q = jumlah not dengan nilai 4.00

Algoritma *backtrack* telah sangat mengurangi jumlah kemungkinan solusi. Yaitu jumlah tiap jenis not yang dicoba tidak akan lebih dari jenis not * jumlah kemunculan.

Hasil ini dapat dikatakan termasuk terstruktur sebab setiap jenis not terkelompokan dalam satu daerah yang sama. Setiap hasil dari pola/motif baru ini dapat dipakai ulang

dengan perubahan melodi atau ritme, atau *chord* yang bersesuaian sehingga musik yang dikomposisikan dapat menjadi lebih variatif dan indah untuk didengar manusia. Hasil ini juga dapat menjadi inspirasi dalam pembuatan komposisi musik. Sebab pembuatan sebagian kecil dapat menjadi patokan untuk membuat pola/motif yang lebih besar lagi dalam suatu komposisi.

Jika algoritma batasan dikembangkan dan batasan kasus diperluas,

V. KESIMPULAN

Dari hasil percobaan dalam makalah ini, terlihat bahwa musik dapat dipolakan dengan suatu algoritma tertentu. Walaupun hasil yang penulis buat dalam makalah ini masih sangat sederhana dengan batasan-batasan masalah serta algoritma yang digunakan juga belum terkategorikan sebagai algoritma yang mangkus untuk membuat pola/motif pada suatu komposisi. Sudah ada penelitian yang dilakukan mengenai komposisi musik otomatis, dan dari percobaan kecil yang penulis lakukan, penulis menyimpulkan bahwa dengan pengembangan dan penelitian yang lebih mendalam maka komposisi otomatis suatu saat akan dapat dilakukan.

VI. UCAPAN TERIMAKASIH

Untuk pembuatan makalah ini, penulis mengucapakan terima kasih kepada Tuhan Yang Maha Esa karena penulis dapat menyelesaikan makalah ini dengan baik dan tepat waktu. Penulis juga mengucapkan terima kasih kepada Dr. Nur Ulfa Maulidevi ST,M.Sc. selaku dosen pembimbing dalam proses pembuatan makalah ini.

REFERENSI

[1] https://web.archive.org/web/20070317015632/http://www.cse.ohio-state.edu/~gurari/course/cis680/cis680Ch19.html#QQ1-51-128. Diakses tanggal 12 Mei 2018 pukul 14:14.

[2] http://wmich.edu/mus-gened/mus170/RockElements.pdf. Diakses tanggal 12 Mei 2018 pukul 14:43.

[3] soundsofnewmexico.com/B1_glossary_of_terms-N06.html. Diakses tanggal 12 Mei 2018 pukul 16:20.

[4] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Runut-balik-(2018).pdf. Diakses tanggal 12 Mei 2017 pukul 16:35

[5] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2016-2017/Makalah2017/Makalah-IF2211-2017-028.pdf. Diakses tanggal 12 Mei 2017 pukul 12:02.

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 Mei 2018

Ferdiant Joshua Muis, 13516047