Penerapan Algoritma String Matching untuk Mendeteksi Musik Plagiat

Muhammad Zen - 13511060[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13511060@std.stei.itb.ac.id

Sudah ribuan tahun manusia mengenal music, mungkin sudah milyaran music tercipta. Musik yang merupakan hasil kerja seorang pencipta sudah sepatutnya kita hargai. Namun, mirisnya banyak orang yang tidak menghargai seorang pencipta music dengan melakukan penjiplakan baik disengaja maupun tidak. Dalam makalah ini akan dipaparkan penerapan algoritma *string matching* untuk mendeteksi music plagiat.

***Kata kunci* – musik, plagiat, *string matching*.**

I. PENDAHULUAN

Musik adalah suara yang disusun demikian rupa sehingga mengandung irama, lagu, dan keharmonisan terutama suara yang dihasilkan dari alat-alat yang dapat menghasilkan irama.

Musik dikenal sejak kehadiran manusia modern Homo sapien yakni sekitar 180.000 hingga 100.000 tahun yang lalu. Tiada siapa tahu bila manusia mula mengenal seni dan musik. Dari penemuan arkeologi pada lokasi-lokasi seperti pada benua Afrika sekitar 180.000 tahun hingga 100.000 tahun dahulu telah menunjukkan perubahan evolusi dari pemikiran otak manusia. Dengan otak manusia yang lebih pintar dari hewan, mereka membuat pemburuan yang lebih terancang sehingga bisa memburu hewan yang besar. Dengan kemampuan otak ini, mereka bisa berpikir lebih jauh hingga di luar nalar dan mencapai imajinasi dan spiritual. Bahasa untuk berkomunikasi telah terbentuk di antara mereka. Dari bahasa dan ucapan sederhana untuk tanda bahaya dan memberikan nama-nama hewan, perlahan-lahan beberapa kosa kata muncul untuk menamakan benda dan nama panggilan untuk sesorang.

Dalam makalah ini, penulis akan memaparkan penerapan algoritma *string matching* untuk mendeteksi apakah suatu music merupakan hasil plagiat atau tidak.

.

II. DASAR TEORI

A. Plagiat

Plagiarisme adalah penjiplakan atau pengambilan karangan, pendapat, dan sebagainya dari orang lain dan menjadikannya seolah karangan dan pendapat sendiri. Plagiat dapat dianggap sebagai tindak pidana karena mencuri hak cipta orang lain. Di dunia pendidikan, pelaku plagiarisme dapat mendapat hukuman berat seperti dikeluarkan dari sekolah/universitas. Pelaku plagiat disebut sebagai plagiator. Hal-hal yang digolongkan sebagai plagiarisme:

- menggunakan tulisan orang lain secara mentah, tanpa memberikan tanda jelas (misalnya dengan menggunakan tanda kutip atau blok alinea yang berbeda) bahwa teks tersebut diambil persis dari tulisan lain
- mengambil gagasan orang lain tanpa memberikan anotasi yang cukup tentang sumbernya

Dalam buku Bahasa Indonesia: Sebuah Pengantar Penulisan Ilmiah, Felicia Utorodewo dkk. menggolongkan hal-hal berikut sebagai tindakan plagiarisme:

- Mengakui tulisan orang lain sebagai tulisan sendiri,
- Mengakui gagasan orang lain sebagai pemikiran sendiri
- Mengakui temuan orang lain sebagai kepunyaan sendiri
- Mengakui karya kelompok sebagai kepunyaan atau hasil sendiri,
- Menyajikan tulisan yang sama dalam kesempatan yang berbeda tanpa menyebutkan asal-usulnya
- Meringkas dan memparafrasekan (mengutip tak langsung) tanpa menyebutkan sumbernya, dan
- Meringkas dan memparafrasekan dengan menyebut sumbernya, tetapi rangkaian kalimat dan pilihan katanya masih terlalu sama dengan sumbernya.

Hal-hal yang tidak tergolong plagiarisme:

- menggunakan informasi yang berupa fakta umum.
- menuliskan kembali (dengan mengubah kalimat atau parafrase) opini orang lain dengan memberikan sumber jelas.
- mengutip secukupnya tulisan orang lain dengan memberikan tanda batas jelas bagian kutipan dan menuliskan sumbernya.

Undang-undang yang mengatur hak cipta di Indonesia adalah UU nomor 19 tahun 2002 tentang hak cipta :

Pasal 2 ayat (1)

Pasal 3 ayat (1),(2)
Pasal 12
Pasal 15
Pasal 26 ayat (1)

Sanksi Plagiat (UU No. 20/2003)

Lulusan PT yang karya ilmiahnya digunakan untuk memperoleh gelar akademik, profesi, atau vokasi terbukti merupakan jiplakan:

- Pencabutan gelar (Pasal 25 ayat 2).
- dipidana penjara paling lama dua tahun dan/atau pidana denda paling banyak 200 juta rupiah (Pasal 70).

Hak Cipta

- Memberi perlindungan terhadap karya cipta di bidang seni, sastra, dan ilmu pengetahuan.
- Timbul secara otomatis setelah suatu ciptaan dilahirkan.
- Dianggap sebagai benda bergerak.
- Dapat beralih atau dialihkan.
- Ciptaan yang tidak diketahui penciptanya, hak ciptanya adalah pada Negara.

Sanksi Pidana Pelanggaran Hak Cipta

- Pidana penjara paling singkat satu bulan dan/atau denda paling sedikit 1 juta rupiah, atau pidana penjara paling lama 7 tahun dan/atau denda paling banyak 5 milyar rupiah.

Bentuk Ciptaan yang Dilindungi

- Buku, program komputer, pamflet, perwajahan (layout) terbitan, dan karya tulis lain;
- Ceramah, kuliah, pidato dan sejenis;
- Alat peraga pendidikan dan ilmu pengetahuan;
- Lagu atau musik dgn atau tanpa teks;
- Drama atau drama musikal, tari, koregografi, pewayangan dan pantomim;
- Seni rupa (seni lukis, gambar, ukir, kaligrafi, pahat, patung, kolase, dan seni terapan;
- Arsitektur;
- Peta;
- Seni batik;
- Fotografi;
- Sinematografi; dan
- Terjemahan, tafsir, saduran, bunga rampai, database, dan karya lain hasil pegalihwujudan.

Ruang Lingkup HAKI

- Hak cipta dan hak-hak berkaitan dgn hak cipta;
- Merek;
- Indikasi geografis;
- Rancangan industri;
- Paten;
- Desain layout dari rangkaian elektronik terpadu;
- Perlindungan thd rahasia dagang; dan
- Pengendalian praktek-praktek persaingan tdk sehat dalam perjanjian lisensi.

Cara Menghindari Plagiat

- Cantumkan dua tanda petik (“) pada pernyataan yang berasal langsung dari naskah asli dan cantumkan sumbernya dengan benar;
- Tulis ulang (paraphrase); dan
- Cantumkan sumbernya dengan benar.

Cara Melakukan Paraphrase

- Baca ulang secara cermat, singkirkan naskah aslinya;
- Gunakan kata-kata dan ide anda sendiri dalam merangkai kalimat;
- Urutkan pemikiran anda dan utarakan ide tersebut; dan
- Periksa ulang parapharase anda, bandingkan dengan naskah asli, pastikan tidak menggunakan kata/istilah yang sama, dan informasi yang akan disampaikan tepat.

Yang Termasuk Kutipan

- Ringkasan (summary);
- Ungkapan dengan kata-kata sendiri (paraphrase);
- Kutipan (quotation); dan
- Statistik dan grafik.

Yang Bukan Kutipan

- Pengetahuan umum (common sense);
- Kesimpulan anda sendiri;
- Fakta-fakta yang dapat ditemukan pada berbagai sumber; dan
- Istilah standar (standard term)

B. Algoritma *String Matching*

Algoritma string matching dalam bahasa Indonesia dikenal dengan istilah algoritma pencocokan string. [1]

Persoalan pencarian string dirumuskan sebagai berikut diberikan :

1. Sebuah teks (text), yaitu sebuah (long) string yang panjangnya n karakter
2. Pattern, yaitu sebuah string dengan panjang m karakter (m < n) yang akan dicari dalam teks

Carilah lokasi pertama di dalam teks yang bersesuaian dengan pattern.

Aplikasi permasalahan pencocokan string biasa ditemukan dalam pencarian sebuah kata dalam dokumen

(misalnya menu Find dalam Microsoft Word).

Contoh 1

Pattern : not

Teks : nobody noticed him

|target

Dalam algoritma pencocokan string, teks diasumsikan berada di dalam memori, sehingga bila kita mencari string di dalam sebuah arsip, maka semua isi arsip perlu dibaca terlebih dahulu kemudia disimpan di dalam memori. Jika pattern muncul lebih dari sekali di dalam teks, maka pencarian hanya akan memberikan keluaran berupa lokasi pattern ditemukan pertama kali. Ada tiga buah algoritma yang umum digunakan dalam melakukan pencocokan string, algoritma Brute Force, algoritma Knuth-Morris-Pratt, serta algoritma Boyer Moore.

B.1. Algoritma Brute Force

Algoritma brute force untuk pencocokan string adalah sebagai berikut :

(diasumsikan teks berada dalam array T [1..n] dan pattern berada dalam array P[1..m])

1. Mula-mula pattern P dicocokan pada awal teks T
2. Dengan bergerak dari kiri ke kanan, bandingkan setiap karakter di dalam pattern P dengan karakter yang berkesesuaian di dalam teks T sampai semua karakter yang dibandingkan cocok atau sama (pencarian berhasil), atau dijumpai sebuah ketidakcocokan karakter (pencarian belum berhasil)
3. Bila pattern P belum ditemukan kecocokannya dan teks T belum habis, geser pattern P satu karakter ke kanan dan ulangi langkah 2. [1]

Contoh 2

Teks : nobody noticed him

Pattern : not

```
    nobody noticed him
s=0 not
s=1  not
s=2   not
s=3    not
s=4     not
s=5      not
s=6       not
s=7        not
```

Pattern not ditemukan pada posisi indeks ke delapan dari awal teks. Kompleksitas algoritma pencocokan string yang dihitung dari jumlah perbandingan yang dilakukan untuk kasus terbaik adalah O(n). Kasus terbaik ini terjadi apabila karakter pertama pattern P tidak pernah sama dengan karakter pada teks T yang dicocokan. Pada kasus ini dilakukan paling banyak n buah operasi perbandingan. Untuk kasus terburuk, kompleksitas algoritma ini adalah O(mn) karena dibutuhkan m(n-m+1) perbandingan

B.2. Algoritma Knuth-Morris-Pratt

Pada algoritma Knuth-Morris-Pratt (KMP) informasi ketidakcocokan pattern dengan teks digunakan disimpan untuk menentukan jumlah pergeseran. Algoritma KMP melakukan pergeseran lebih jauh sesuai dengan informasi yang disimpan, tidak seperti pada algoritma Brute Force di mana pergeseran dilakukan setiap satu karakter, sehingga waktu pencarian dapat dikurangi secara signifikan. AlgoritmaKnuth-Morris-Pratt dikembangkan oleh D. E. Knuth, bersama-sama dengan J.H Morris dan V. R. Pratt[1].

Dalam algoritma KMP, ada beberapa definisi yang digunakan :

Misalkan A adalah alfabet dan x = x1x2…xk , k Î N, 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 = x1x2…xk – 1 , k Î {1, 2, …, k – 1} dengan kata lain, x diawali dengan u. Akhiran (suffix) dari x adalah upastring(substring) u dengan u = xk – b xk – b + 1 …xk , k Î {1, 2, …, k – 1} dengan kata lain, x diakhiri dengan v. Pinggiran (border) dari x adalah upa-string r sedemikian sehingga

r = x1x2…xk – 1 dan u = xk – b xk – b + 1 …xk ,
k Î {1, 2, …, k – 1}

dengan kata lain, pinggiran dari x adalah upastring yang keduanya awalan dan juga akhiran sebenarnya dari x. [1]

Algoritma KMP melakukan proses awal terhadap pattern P dengan menghitung fungsi pinggiran yang mengindikasikan pergeseran s terbesar yang mungkin dengan menggunakan perbandingan yang dibentuk sebelum pencarian string [1]. Hal ini dimaksudkan untuk mencegah pergeseran yang tidak berguna seperti pada algoritma Brute Force. Fungsi pinggiran hanya bergantung pada karakter-karakter di dalam pattern. Fungsi pinggiran b(j) didefinisikan sebagai ukuran awal terpanjang dari P yang merupakan akhiran dari P[1..j]. Algoritma untuk menghitung fungsi pinggiran adalah sbb

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

Untuk melakukan pencocokan string, mulamula kita harus menghitung fungsi pinggiran untuk pattern tersebut. selanjutnya dilakukan pencocokan pertama, samakan ujung kiri pattern dengan ujung kiri teks. Misalkan karakter cocok pada posisi[1..5] dan pada posisi 6 tidak ditemukan kecocokan, jumlah pergeseran selanjutnya ditentukan oleh pinggiran awalan P yang berkesesuaian. Kompleksitas algoritma KMP dihitung dengan menggabungkan kompleksitas waktu untuk menghitung fungsi pinggiran, yaitu O(m), dan kompleksitas waktu untuk melakukan pencarian string, O(n), sehingga kompleksitas waktu keseluruhan algoritma ini adalah O(m+n).

B.3. Algoritma Boyer-Moore

Algoritma Boyer-Moore adalah salah satu algoritma pencarian string, dipublikasikan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1977. Algoritma ini dianggap sebagai algoritma yang paling efisien pada aplikasi umum.Tidak seperti algoritma pencarian string yang ditemukan sebelumnya, algoritma Boyer-Moore mulai mencocokkan karakter dari sebelah kanan pattern. Ide dibalik algoritma ini adalah bahwa dengan memulai pencocokkan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat.

Misalnya ada sebuah usaha pencocokan yang terjadi pada teks[i..i+n-1], dan anggap ketidakcocokan pertama terjadi di antara 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 sebelum b dan v adalah sebuah awalan dari pattern, maka penggeseran-penggeseran yang mungkin adalah:

- Penggeseran good-suffix yang terdiri dari mensejajarkan potongan teks[i+j+1..i+n-1] = pattern[j+1..n-1] dengan kemunculannya 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.
- Penggeseran bad-character yang terdiri dari mensejajarkan teks[i+j] dengan kemunculan paling kanan karakter tersebut di pattern. Bila karakter tersebut tidak ada di pattern, maka pattern akan disejajarkan dengan teks[i+n+1].

Secara sistematis, langkah-langkah yang dilakukan algoritma Boyer-Moore pada saat mencocokkan string adalah:

- Algoritma Boyer-Moore mulai mencocokkan pattern pada awal teks.
- Dari kanan ke kiri, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
  - ✓ Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).
  - ✓ Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
- Algoritma kemudian menggeser pattern dengan memaksimalkan nilai penggeseran good-suffix dan penggeseran bad-character, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

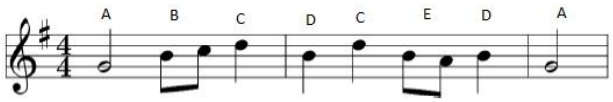
```
procedure preBmBc(
    input P : array[0..n-1] of
char,
    input n : integer,
    input/output bmBc :
array[0..n-1] of integer
)
Deklarasi:
  i: integer

Algoritma:
  for (i := 0 to ASIZE-1)
     bmBc[i] := m;
  endfor
  for (i := 0 to m - 2)
     bmBc[P[i]] := m - i - 1;
  endfor
procedure preSuffixes(
    input P : array[0..n-1] of
char,
    input n : integer,
    input/output suff :
array[0..n-1] of integer
)

Deklarasi:
  f, g, i: integer

Algoritma:
  suff[n - 1] := n;
  g := n - 1;
  for (i := n - 2 downto 0) {
     if (i > g and (suff[i + n -
1 - f] < i - g))
        suff[i] := suff[i + n - 1
- f];
     else
        if (i < g)
           g := i;
        endif
        f := i;
        while (g >= 0 and P[g] =
P[g + n - 1 - f])
           --g;
        endwhile
        suff[i] = f - g;
     endif
  endfor
procedure preBmGs(
    input P : array[0..n-1] of
char,
    input n : integer,
    input/output bmBc :
array[0..n-1] of integer
)
Deklarasi:
  i, j: integer
  suff: array [0..RuangAlpabet]
of integer
```

```
preSuffixes(x, n, suff);

  for (i := 0 to m-1)
     bmGs[i] := n
  endfor
  j := 0
  for (i := n - 1 downto 0)
     if (suff[i] = i + 1)
        for (j:=j to n - 2 - i)
           if (bmGs[j] = n)
              bmGs[j] := n - 1 - i
           endif
        endfor
     endif
  endfor
  for (i = 0 to n - 2)
     bmGs[n - 1 - suff[i]] := n -
1 - i;
  endfor
//Dan berikut adalah pseudocode
algoritma Boyer-Moore pada fase
pencarian:
procedure BoyerMooreSearch(
    input m, n : integer,
    input P : array[0..n-1] of
char,
    input T : array[0..m-1] of
char,
    output ketemu : array[0..m-1]
of boolean
)

Deklarasi:
i, j, shift, bmBcShift, bmGsShift:
integer
BmBc : array[0..255] of interger
BmGs : array[0..n-1] of interger

Algoritma:
preBmBc(n, P, BmBc)
preBmGs(n, P, BmGs)
i:=0
while (i<= m-n) do
    j:=n-1
    while (j >=0 n and T[i+j] =
P[j]) do
       j:=j-1
    endwhile
    if(j < 0) then
       ketemu[i]:=true;
    endif
    bmBcShift:=
BmBc[chartoint(T[i+j])]-n+j+1
    bmGsShift:= BmGs[j]
    shift:= max(bmBcShift,
bmGsShift)
    i:= i+shift
```

III. PENERAPAN

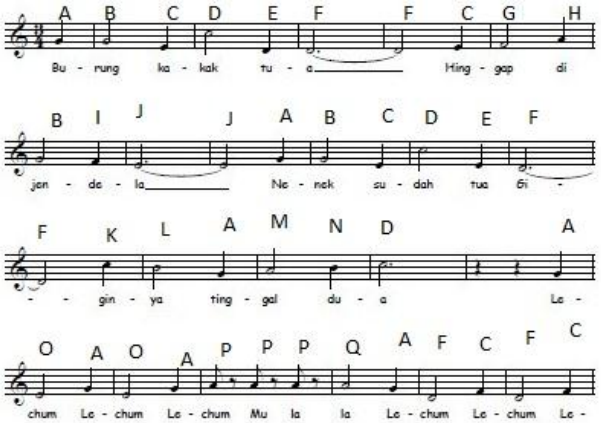
Untuk menentukan apakah suatu music merupakan hasil plagiat atau tidak maka dilakukan hal sebagai berikut.

1. Mengubah notasi dalam music menjadi suatu kode sehingga memudahkan dalam melakukan string matching. Contoh

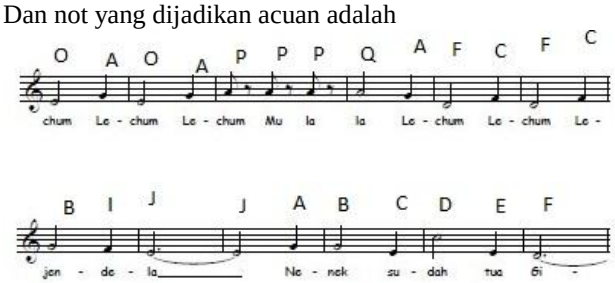


Sehingga dari notasi tersebut dikodekan menjadi ABCDCEDA

2. Melakukan string matching per baris.
Contoh
Terdapat suatu notasi lagu yang ingin dicek apakah hasil plagiat atau tidak, yaitu



Setelah dikodekan menjadi
ABCDEFFCGH
BIJJABCDEF
FKLAMNDA
OAOAPPPQAFCFC
Dan not yang dijadikan acuan adalah



Setelah dikodekan menjadi
OAOAPPPQAFCFC
BIJJABCDEF
Contohnya pada not yang dijadikan sebagai acuan pada baris pertama adalah "OAOAPPPQAFCFC" maka akan dilakuakan string matching pada not yang ingin dicek. Pencocokan dilakukan per baris jadi pertama dilakukan pencocokan terhadap baris pertama apabila ketemu dimasukkan ke indeks bar yang sama tersebut, bila tidak ketemu lanjut ke baris selanjutnya sampai ketemu atau baris tes habis.

Bila ketemu maka akan disimpan bar yang sama tersebut dan lanjut melakukan pencocokan terhadap baris acuan selanjutnya, bila bar tes habis lanjut ke bar acuan selanjutnya, hal tersebut diulangi sampai bar acuan habis.

Jadi yang digunakan sebagai pattern adalah baris not yang dijadikan acuan, sedangkan yang dijadikan teks adalah seluruh baris notasi yang ingin di tes. Suatu music dikategorikan sebagai hasil plagiat apabila terdapat minimal 8 baris yang sama.

Algoritma yang dipakai penulis dalam makalah ini adalah KMP. Berikut pseudo code nya

```
procedure KMPsearch(input m, n : integer,
input P: array[1..m] of char,input T :
array[1..n] of char, output idx : integer)
{Mencari kecocokan pattern P di dalam teks
T dengan algoritma Knuth-Morris-Pratt.
Jika
ditemukan P di dalam T, lokasi awal
kecocokan disimpan di dalam peubah idx.
Masukan: pattern P yang panjangnya m dan
teks T yang panjangnya n.
Teks T direpresentasika sebagai string
(array of character)
Keluaran: posisi awal kecocokan (idx).
Jika P tidak ditemukan, idx = -1.}

Deklarasi

 i, j, plag: integer
 //plag adalah jumlah bar yang sama
 ketemu, plagiat : boolean
 b : array[1..m] of integer
 procedure HitungPinggiran(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] }

Algoritma:

plag <-0
plagiat <- false
for ((setiap baris) and (not ketemu))
 HitungPinggiran(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
      plag <- plag+1
      idx <- i-m+1 { catatan: jika
      indeks     array
      dimulai dari 0, maka idx<-i-m }
  else
      idx <- -1
  endif
endfor
if plag >= 8 then
   plagiat <- true
```

IV. KESIMPULAN DAN SARAN

Algoritma *string matching* dapat dimanfaatkan untuk mendeteksi apakah suatu music merupakan hasil plagiasi karya orang lain atau tidak.

Untuk kedepanya penulis berharap algoritma *string matching* untuk mendeteksi music hasil plagiasi ini dapat dikembangkan lebih lanjut dan dapat dimanfaatkan oleh perusahaan rekaman sehingga sebelum suatu music disebarkan luaskan harus lolos uji. Dengan begitu tidak ada lagi plagiasi dalam music dan penghargaan hak kekayaan intelektual lebih dihargai.

REFERENSI

[1] Munir, Rinaldi.Diktat Kuliah IF2251 Strategi Algoritmik. Institut Teknologi Bandung. 2007.

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2013-2014/stmik13-14.html.

[3] http://www.artikata.com/arti-345419-plagiat.html

[4] http://syaifalmandiri.wordpress.com/2012/04/03/plagiat-dan-uu-tentang-plagiat/

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, 20 Desember 2013

Muhammad Zen - 13511060