PENERAPAN ALGORITMA *EDIT DISTANCE* PADA PENDETEKSIAN PRAKTIK PLAGIAT

Willy Goenawan[1], Ronald Augustinus[2], Krisantus Sembiring[3]

Laboratorium Ilmu danRekayasa Komputer
Departemen Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung

E-mail : if13048@students.if.itb.ac.id[1],
if13117@students.if.itb.ac.id[2], if13121@students.if.itb.ac.id[3]

Abstrak

Praktik plagiat dan kolusi merupakan hal tidak asing lagi di telinga kita dan seringkali terjadi dewasa ini, tak terkecuali di kalangan mahasiswa, seiring dengan kemajuan teknologi informasi, teknologi komputer misalnya, yang sangat pesat. Plagiat adalah proses meniru atau menyalin hasil karya milik orang lain yang diklaim sebagai hasil karya sendiri. Dalam ruang lingkup yang lebih kecil, di dunia informatika, praktik plagiat dilakukan dengan cara penyalinan kode program (*source code*) yang dilakukan ketika ada suatu tugas yang harus diselesaikan oleh mahasiswa. Di kalangan mahasiswa informatika, yang selalu berinteraksi dengan komputer yang mempermudah praktik plagiat mengingat adanya fasilitas untuk menyalin dan mengubah teks (*copy and paste*) dan fasilitas koneksi yang memungkinkan untuk mengakses hasil karya orang lain secara bebas melalui internet, praktik plagiat ini sering dilakukan. Untuk meminimalisasi praktik plagiat, diperlukan pendeteksian terhadap tugas yang dikumpulkan oleh mahasiswa. Namun, proses pendeteksian secara manual sulit untuk dilakukan mengingat jumlah mahasiswa yang banyak dan kemahiran mahasiswa memanipulasi teks sumber, misalnya mengubah komentar atau nama variabel pada kode program (*source code*). Oleh karena itu, proses pendeteksian secara cepat dan mangkus sangatlah dibutuhkan. Praktik plagiat secara sederhana dapat dideteksi berdasarkan tingkat kemiripan antara hasil plagiat dengan sumber. Salah satu algoritma yang mangkus yang dapat digunakan untuk pendeteksian berdasarkan kemiripan penulisan adalah pemrograman dinamis berdasarkan perbandingan string atau pencocokan string (*string matching*) yang akan dibahas lebih lanjut dalam makalah ini.

Kata kunci*: plagiat, pemrograman dinamis, string matching*

1. Pendahuluan

Praktik plagiat tidaklah menjadi hal asing lagi, apalagi di kalangan mahasiswa yang hampir setiap hari mengerjakan tugas yang diberikan oleh dosen. Tak terkecuali pula, mahasiswa informatika sering mendapat tugas untuk membuat program aplikasi dengan menggunakan bahasa pemrograman tertentu. Dalam pengerjaan tugas tersebut, praktik plagiat tak terelakkan lagi untuk dilakukan mengingat waktu pengerjaan tugas yang terbatas dan tidak adanya motivasi untuk berusaha menyelesaikan tugas dengan kemampuan sebdiri. Praktik plagiat dilakukan dengan cara tukar-menukar kode program (*source code*) yang telah berhasil. Mahasiswa yang memplagiat dapat dengan mudah menyalin atau mengganti kode program yang telah didapatkan secara cepat dengan menggunakan fitur-fitur yang disediakan oleh komputer. Untuk mengatasi praktik plagiat, tidaklah cukup hanya mengingatkan kepada mahasiswa bahwa tindakan plagiat tidak baik dilakukan. Pendeteksian praktik plagiat merupakan solusi yang sebaiknya dilakukan sehingga tindakan curang tersebut dapat diminimalisasi.

2. Teknik Plagiat

Plagiat adalah teknik penyalinan atau meniru karya orang lain yang diklaim menjadi hasil karya sendiri. Tidak adanya motivasi ataupun kemudahan dalam proses penyalinan dengan harapan tidak diketahui orang lain menjadi alasan utama terjadinya praktik plagiat. Beberapa jenis plagiat yang dikenal selama ini, yaitu:

a. *Word-for-word plagiarism* : menyalin setiap kata secara langsung tanpa diubah sedikitpun

b. *Plagiarism of the form of a source* : menyalin dan atau menulis ulang kode-kode program tanpa mengubah struktur dan jalannya program
c. *Plagiarism of authorship*: mengakui hasil karya orang lain sebagai hasil karya sendiri dengan cara mencantumkan nama sendiri menggantikan nama pengarang sebenarnya

Beberapa contoh praktik plagiat pada sebuah program, yaitu:
a. Leksikal: perubahan pada kode (*source code*) program, misalnya:
   1. Komentar diubah (ditambah, dikurangi, atau diganti)
   2. Format penulisan diubah
   3. Nama variabel diubah
b. Struktural : perubahan struktur program
   1. Perubahan urutan algoritma yang tidak mengubah jalannya program
   2. Prosedur diubah menjadi fungsi atau sebaliknya
   3. Pemanggilan prosedur diganti dengan isi prosedur itu sendiri

3. Algoritma Pemrograman Dinamis

Algoritma ini digunakan untuk mencari kecocokan antara dua string. Dalam proses perbandingannya , string kedua dimanipulasi sehingga pada akhirnya serupa dengan string pertama. Dalam proses perubahan string tersebut dibuat sebuah tabel dua dimensi dengan baris sesuai dengan panjang string terpanjang dan jumlah kolom sebanyak panjang string terpendek. Berikut ini contoh dua string yang dibandingkan:

KORUPSI
KOLUSI

Tabel dua dimensi yang terbentuk (latar belakang biru menunjukkan indeks tabel) adalah:

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1						
O	2						
R	3						
U	4						
P	5						
S	6						
I	7						

Baris dan kolom pada tabel dua dimensi (matriks) di atas diisi dengan langkah-langkah sebagai berikut:
1. Elemen matriks [0,0] akan diisi dengan nilai 0
2. Elemen matriks [x,0] akan diisi dengan nilai matriks [x-1,0]+1.
3. Elemen matriks [0,x] akan diisikan nilai matriks[0,x-1]+1
4. Elemen lainnya (matriks[X,Y])diisi dengan urutan langkah di bawah ini:
   a. Jika karakter ke-X pada string ke-1 memiliki kesamaan dengan karakter ke-Y pada string ke-2 maka nilai matriks[X-1,Y-1] akan dianggap ditambahkan 1 dari nilai sebelumnya.
   b. Bandingkan 3 elemen matriks pada posisi matriks[X-1,Y-1],matriks[X,Y-1], dan matriks[X,Y] untuk pencarian nilai minimum di antara ketiganya. Elemen dengan nilai terkecil akan dimasukkan nilainya ke dalam matriks[X,Y]
   c. ulangi langkah 1 dan 2 sampai semua elemen tabel terisi.

Berikut ini adalah algoritma *Edit Distance* dalam notasi bahasa Pascal:

```
function EditDistance (s1, s2):integer
var
m : Array of Array [1..N] of integer[1..N]
i, j,temp: integer ;

begin
    m[0][0] = 0;
    for j:=1 to s2.length do
    begin
        m[0][j] = m[0][j-1]-0 + 1;
    end;
    for i:=1 to  s1.length do
        //pengulangan untuk kolom
    begin
     m[i][0] = m[i-1][0]-0 + 1;
     for j:=1 to s2.length do
     //pengulangan untuk baris
     begin
        temp= m[i-1][j-1];
        //penampungan nilai
         if   (s1.charAt(i-1)<>s2.charAt(j-1)   )
        //pengecekan kesamaan karakter
        temp := temp + 1;
        //perbandingan nilai
        m[i][j]=minimum(temp, minimum(
        m[i-1][j]-0 + 1,  m[i][j-1]-0 + 1 ) )
     end
    end
end
return  m[s1.length][s2.length];
//pengembalian nilai element terakhir
```

Nilai dari ketidaksamaan diperoleh dari nilai indeks terakhir yang tercatat dalam matriks, seperti tertera di bawah ini.

		K	O	L	U	S	I
	0	1	2	3	4	5	6
K	1	0	1	2	3	4	5
O	2	1	0	1	2	3	4
R	3	2	1	1	2	3	4
U	4	3	2	2	1	2	3
P	5	4	3	3	2	2	3
S	6	5	4	4	3	2	3
I	7	6	5	5	4	3	2

Dari tabel diatas terlihat jelas bahwa elemen matriks[7,6] bernilai 2 (dua) yang merupakan jumlah perbedaan antara kedua string.

Algoritma ini juga dapat melakukan penyisipan maupun penghapusan sehingga dua string yang berbeda panjangnya dapat terdeteksi. Contoh:

Aku anak sang gembala
Aku anak gembala

Algoritma ini akan menghasilkan nilai 5 (lima) yang merupakan jumlah perbedaan antara kedua string.

4. Penerapan Algoritma *Edit Distance*

Algoritma *Edit Distance* ini dipilih untuk menentukan tingkat kemiripan antara dua teks karena memberikan keuntungan yaitu dapat melihat perbedaan di antara dua string dengan cepat dan akurat. Untuk mendeteksi plagiat pada dua teks sumber dapat digunakan fungsi *EditDistance* diatas dengan cara memasukkan isi tiap *file* sumber ke dalam string. Berikut ini algoritmanya dalam notasi bahasa pascal :

```
function tingkatKemiripan (file1,file2:file):real;
var
sumber1,sumber : String;
jlhKar1,jlhKar2,jlhkar: integer;

begin
  {
  membaca file , memasukkannya ke dalam
  string dan menyimpan panjangnya dalam
  variabel masukkan
  }
   bacaFile(sumber1, file1, jlhKar1);
   bacaFile(sumber2, file2, jlhKar2);
        if ( jlhkar1 > jlhKar2 )
        then jlhKar := jlhkar1
        else  jlhKar := jlhkar2;
return ( EditDistance (sumber1,sumber2) /
jlhKar);
end;
```

Dalam prosedur **bacaFile** Isi *file* teks dimasukkan ke dalam string adalah teks program tanpa komentar. Untuk mendeteksi komentar digunakan algoritma *brute force* untuk mencari awalan dari sebuah komentar (contoh : ‘//’,’/*’) dan juga akhiran komentar (contoh : ‘*/’).

Jadi, ketika karakter yang dibaca dari teks sumber adalah penanda komentar maka teks dalam komentar tersebut tidak akan dimasukkan ke dalam string. Algoritma ini juga dapat digunakan untuk dua buah teks yang panjangnya tidak sama.

Untuk mengecek keabsahan suatu teks digunakan persentase kemiripan yang dapat dihitung dengan rumus berikut:

$$P = \frac{D}{T}$$

Keterangan:
P = persentase kemiripan
D = hasil keluaran algoritma *Edit Distance*
T = jumlah karakter terpanjang antara 2 masukan

Dari nilai persentase dapat diketahui besarnya tingkat kemiripan antara kedua teks masukan, apakah masih dalam batas toleransi (80%). Jika ternyata P lebih besar dari 80%, dapat diambil kesimpulan bahwa telah terjadi sebuah praktik plagiat. Namun, hal di atas tidaklah mutlak karena pengguna dapat menentukan sendiri nilai batas toleransi.

Kompleksitas waktu algoritma *Edit Distance* ini adalah O (|String1|*|String2|) atau kuadratik $O(n^2)$ jika panjang kedua string sama. Algoritma pemrograman dinamis ini lebih baik daripada algoritma *brute force* dengan kompleksitas waktu eksponensial $O(3^n)$. Jadi, dapat disimpulkan bahwa algoritma ini sangat baik untuk diterapkan dalam pendeteksian praktik plagiat

5. Kesimpulan

Teknologi yang berkembang terutama teknologi komputer yang memberikan fasilitas kemudahan untuk menyalin atau mengubah suatu teks (termasuk kode program) serta konektivitas (jaringan internet) dari sebuah komputer banyak disalahgunakan untuk melakukan praktik plagiat. Hal ini terjadi pula di kalangan mahasiswa dengan berbagai alasan di baliknya.. Untuk menghindari maraknya praktik plagiat, pendeteksian praktik plagiat sangatlah dibutuhkan. Proses pendeteksian dapat dilakukan dengan mudah berdasarkan kemiripan penulisan. Salah satu algoritma yang dapat digunakan untuk menmperoleh tingkat kemiripan adalah algoritma pemrograman dinamis. Adapun, komplesitas waktu dari algoritma pemrograman dinamis ini adalah kuadratik $O(n^2)$ yang jauh lebih baik daripada algoritma *brute force* dengan kompleksitas waktu eksponensial $O(3^n)$.

6. Referensi

[1] Munir, Rinaldi.2005. *"Strategi Algoritmik"*. Departemen Teknik Informatika, Institut Teknologi Bandung

[2] Paris, Maeve. *"Source Code And Text Plagiarism Detection Strategies "*. http://www.infm.ulst.ac.uk/~maeve . Diakes tanggal 17 Mei 2005 pukul 19.00 WIB.

[3] Sedgewick, Robert. 1990. *"Algorithm in C"*. Addison Wesley.

[4] Skiena, Steven & Revilla, Miguel. 2003. *"Programming Challenges"*. Springer.