BAB I

PENDAHULUAN

1.1 Latar Belakang

Plagiarisme atau plagiat adalah penjiplakan atau pengambilan karangan, pendapat, dan sebagainya dari orang lain dan menjadikannya seolah karangan dan pendapat sendiri (Kamus Besar Bahasa Indonesia, 1997). Sebagai tindakan yang dapat dikategorikan "mencuri", praktik plagiarisme tentu harus dihindarkan.

Dalam mengurangi praktik plagiarisme terdapat dua metode yang dapat dilakukan, yaitu dengan mencegah plagiarisme atau mendeteksi plagarisme. Mencegah plagiarisme berarti melakukan tindakan pencegahan agar plagiarisme tidak terjadi, misalnya dengan menetapkan sebuah kebijakan tentang plagiarisme atau sistem hukuman bila terbukti melakukan plagiarisme. Sedangkan mendeteksi plagiarisme berarti menemukan tindakan plagiarisme yang telah terjadi.

Mendeteksi plagiarisme hanya dapat mengurangi plagiarisme yang telah terjadi, tetapi mencegah plagiarisme tentu dapat menghilangkan atau paling tidak mengurangi sebagian besar plagiarisme. Namun dalam kenyataan, implementasi pencegahan plagarisme merupakan masalah moral masyarakat secara luas yang tidak dapat diselesaikan hanya dengan upaya Universitas atau Departemen. (Lukashenko, Graudina, & Grundspenkis, 2007). Oleh karena itu, yang mungkin dilakukan adalah mendeteksi plagiarisme.

Pendeteksian plagiarisme dilakukan secara manual atau dengan bantuan komputer. Menurut Fakhri (Mahathir, 2011), Saat ini pendeteksian secara manual merupakan cara yang paling akurat dalam mendeteksi plagiat. Kelemahan dari

cara ini adalah sangat menghabiskan tenaga, waktu, serta tidak konsisten karena dipengaruhi factor emosional manusia. Oleh karena itu, sampai saat ini para akademisi berusaha mengembangkan sebuah sistem komputer untuk mendeteksi plagiat dengan tingkat akurasi yang mendekati sistem manual.

Pada masa sekarang ini plagiat sudah sering terjadi dimana-mana, dalam lingkup akademispun tindak plagiat tergolong cukup banyak. Akademisi sering melakukan tindak penjiplakan atau plagiarisme dalam mengerjakan tugas-tugas yang seharusnya dikerjakan sendiri. Praktik plagiarisme yang dilakukan tidak terbatas hanya pada hasil karangan atau pendapat seseorang, namun terjadi juga dalam pemrograman komputer, dimana kode program dengan mudah dapat digandakan dan dimodifikasi baik sebagian atau keseluruhan kode program.

Joy dan Luck (Joy & Luck, 1999) menggolongkan plagiarisme kode program menjadi dua, perubahan non-struktural (leksikal) yaitu perubahan tanpa mempengaruhi jalannya program dan perubahan struktural yang mengubah jalannya program. Contoh perubahan non-struktural meliputi merubah komentar dan nama variabel, sedangkan contoh perubahan struktural meliputi merubah urutan dan mengganti *statement* (Cosma, 2008).

Sebelumnya, di Universitas Multimedia Nusantara telah dilakukan penilitian yang berjudul “Rancang Bangun Aplikasi Pendeteksi Plagiarisme Kode Program Bahasa C Menggunakan Algoritma Levenshtein Distance dan Brute Force” oleh Andreas Arifianto (Arifianto, 2011). Yang kemudian ditingkatkan oleh Regina Natalia (Natalia, 2011) dalam penelitiannya yang berjudul

“Peningkatan Kemampuan Aplikasi Pendeteksi Plagiarisme pada Kode Program Berbahasa Pemprograman C dengan Algoritma Boyer Moore dan Brute Force”.

Pada penelitian kedua dilakukan peningkatan pada aplikasi tersebut dengan mengganti Algoritma Levenshtein Distance dengan Algoritma Boyer Moore. Peningkatan tersebut terbukti berhasil dengan peningkatan pada pendeteksian plagiarisme non-struktural. Namun dalam plagiarisme struktural, algoritma Brute Force yang digunakan dalam kedua penelitian tersebut tidak menjadi fokus penelitian. Karena itu dibutuhkan algoritma yang dapat meningkatkan kemampuan aplikasi tersebut pada sisi plagiarisme struktural.

Algoritma Smith-Waterman memiliki keakuratan yang lebih baik pada saat membandingkan dokumen yang mengandung pengubahan struktur (Novanta, 2009). Dengan mengacu pada penelitian-penelitian sebelumnya, dilakukan upaya pengoptimalisasian aplikasi pendeteksi plagiarisme pada kode program dalam hal plagiarisme struktural, yaitu dengan mengganti Algoritma Brute Force pada penelitian kedua dengan Algortima Smith-Waterman. Upaya optimalisasi divalidasi dengan membandingkan hasil persentase Algoritma Smith-Waterman dengan hasil persentase Algoritma Brute Force dari penelitian sebelumnya.

Walaupun difokuskan pada plagiarisme struktural, pendeteksian plagiarisme pada sisi non-struktural tetap diperlukan untuk mengoptimalkan aplikasi pendeteksi plagiarisme. Untuk itu digunakan algoritma Boyer-Moore yang telah terbukti lebih optimal dibandingkan algoritma Levenshtein Distance dalam plagiarisme non-struktural. Sehingga, berdasarkan hal-hal tersebut penulis

melakukan penelitian dengan judul “Implementasi Algoritma Boyer-Moore dan Smith-Waterman pada Aplikasi Pendeteksi Plagiarisme Kode Program”.

1.2 Rumusan Masalah

1. Bagaimana membuat aplikasi pendeteksi plagiarisme pada kode program?
2. Bagaimana mengimplementasi algoritma Boyer Moore pada sisi plagiarisme non-struktural?
3. Bagaimana mengoptimalisasi aplikasi pendeteksi plagiarisme dalam mendeteksi plagiarisme struktural dengan Algoritma Smith-Waterman?
4. Bagaimana performa dari Algoritma Smith-Waterman dalam mendeteksi plagiarisme struktural pada kode program dibandingkan dengan Algoritma Brute Force?

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah

1. Aplikasi hanya dapat mendeteksi dokumen kode program berbahasa C dengan format .c atau .cpp.
2. Aplikasi bekerja secara offline.
3. Materi kode program yang dibandingkan meliputi: *variable, conditional, looping, function, array, pointer, struct* dan *file.*
4. Pendeteksian dapat dilakukan dengan memilih dua buah kode program atau sebuah *folder* dengan hasil persentase kemiripan setiap dua kode program berdasarkan plagiarisme non-struktural dan struktural.

1.4 Tujuan Penelitian

1. Membangun aplikasi pendeteksi plagiarisme pada kode program.
2. Mengimplementasikan algoritma Boyer Moore pada sisi plagiarisme non-struktural.
3. Mengoptimalisasi kemampuan aplikasi pendeteksi plagiarisme dalam mendeteksi plagiarisme struktural dengan algoritma Smith-Waterman.
4. Mengukur performa dari Algoritma Smith-Waterman dalam mendeteksi plagiarisme struktural pada kode program dibandingkan dengan Algoritma Brute Force.

1.5 Manfaat Penelitian

Penelitian ini bermanfaat untuk membantu mendeteksi praktik plagiarisme dalam mata kuliah pemrograman. Penelitian ini diharapkan dapat digunakan oleh para pengajar dalam mendeteksi plagiarisme kode program.

1.6 Metode Penelitian

Penelitian aplikasi pendeteksi plagiarisme yang dilakukan melewati beberapa langkah sebagai berikut

1. Studi Literatur

   Penelitian diawali dengan membaca jurnal-jurnal dan artikel-artikel ilmiah untuk mempelajari teori-teori yang berkaitan dengan plagiarisme, plagiarisme dalam kode program, dan berbagai algoritma pencocokan *string.*

2. Pengumpulan Data dan Desain

Pada tahap ini dikumpulkan data-data yang berhubungan dengan algoritma-algoritma yang biasa digunakan dalam membandingkan kode program dalam pendeteksian plagiarisme, kemudian dimulai perancangan aplikasi yang terdiri dari perancangan cara kerja aplikasi berupa *flowchart* dan sketsa desain tampilan program.

3. Pembuatan Aplikasi

   Pada tahap ini aplikasi dibuat dengan mengimplementasikan *flowchart* dan rancangan desain yang telah dibuat sebelumnya dengan menggunakan IDE Microsoft Visual Studio C# 2008.

4. Uji Coba dan Evaluasi

   Melakukan uji coba terhadap aplikasi agar sesuai dengan tujuan penelitian dan juga evaluasi untuk menemukan masalah-masalah yang mungkin muncul.

5. Penulisan Skripsi

   Tahap ini dilakukan seiring penelitian dilakukan untuk membuat dokumentasi mengenai permbuatan aplikasi, mulai dari tahap studi literatur, analisis dan perancangan aplikasi, implementasi hingga uji coba dan evaluasi.

1.7 Sistematika Penulisan

Sistematika penulisan laporan skripsi ini dijelaskan sebagai berikut.

Bab I Pendahuluan

Berisi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

Bab II Tinjauan Pustaka

Berisi landasan teori mengenai Plagiarisme dan Deteksi Plagiarisme, Metode Perancangan, Bahasa Pemrograman yang terdiri dari Bahasa Pemrograman C dan Bahasa Pemrograman c#, Algoritma yaitu algoritma Boyer Moore dan algoritma Smith-Waterman, *preprocessing,* dan Microsoft Visual Studio

BabIII Analisa dan Perancangan Sistem

Berisi spesifikasi umum kebutuhan sistem dan rancangan desain antarmuka aplikasi

Bab IV Implementasi dan Uji Coba

Berisi penjelasan mengenai implementasi dari perancangan aplikasi dan hasil uji coba aplikasi berdasarkan rancangan desain di bab sebelumnya

Bab V Simpulan dan Saran

Berisi kesimpulan dari penelitian yang dilakukan dan saran untuk penelitian selanjutnya