BAB III

ANALISIS DAN PERANCANGAN

3.1 Analisis Masalah

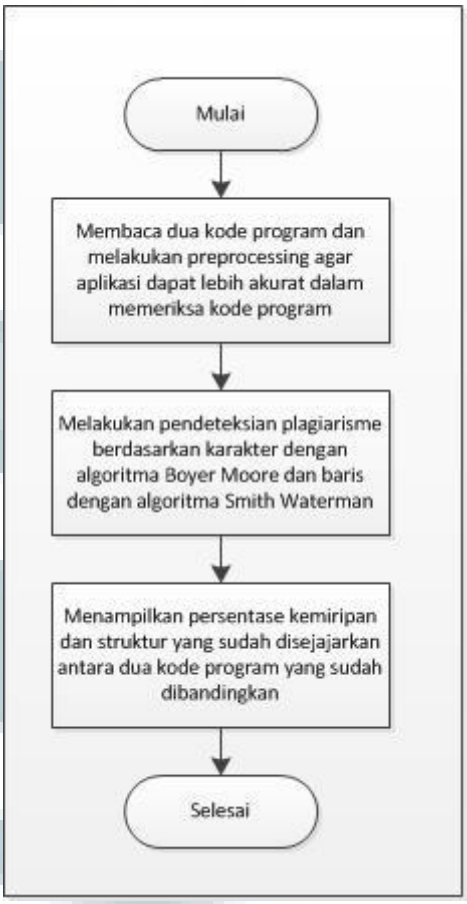
Berdasarkan analisis yang dilakukan, masalah pada pendeteksian plagiarisme secara manual adalah membutuhkan waktu yang lama. Ditambah lagi pelaku plagiarisme dapat melakukan modifikasi kode program sehingga tidak 100% sama dengan aslinya. Perubahan yang dilakukan biasanya berupa mengubah komentar, mengubah *string output* program, mengubah nama variabel, mengubah nama fungsi, mengubah nama parameter fungsi, mengubah bentuk pengulangan, mengubah *condtitional statement,* dan melakukan *insertion* ataupun *deletion* pada baris kode program.

Perubahan-perubahan tersebut tidak mudah dideteksi begitu saja, diperlukan aplikasi pendeteksi plagiarisme yang menggunakan metode dan algoritma yang dapat mendeteksi plagiarisme kode program secara akurat. Setelah plagiarisme terdeteksi masalah selanjutnya yang mungkin muncul adalah bagaimana mengetahui bagian mana dari kode program tersebut yang memiliki kesamaan, dan yang mengalami *insertion* atau *deletion*. Jika diperlukan pemeriksaan secara manual untuk mengetahui bagian-bagian yang diplagiarisme, akan dihadapi masalah yang sama dengan sebelumnya.

3.2 Usulan Solusi Masalah

Solusi yang dapat ditawarkan adalah pembuatan sebuah aplikasi pendeteksi plagiarisme kode program dengan algoritma Boyer Moore dan Smith-

Waterman. Dengan tahapan-tahapan yang dilakukan aplikasi sesuai pada gambar 3.1.



Gambar 3.1 *Flowchart* Umum Aplikasi

Aplikasi melakukan teknik *preprocessing* sebelum mendeteksi plagiarisme. Dalam *preprocessing* dilakukan modifikasi pada dua kode program untuk mengurangi efek dari perubahan leksikal maupun struktural. Dengan begitu kode program dapat dibandingkan dengan lebih akurat. Hasil *preprocessing* tersebut dicocokkan dengan algoritma *Boyer Moore* berdasarkan karakter. Pendeteksian berdasarkan karakter ini di optimalkan oleh algoritma Boyer Moore dengan melakukan perhitungan tabel *suffixes* untuk mengetahui apakah terdapat *substring* yang berulang pada pola pencarian. Untuk pola yang tidak memiliki dua

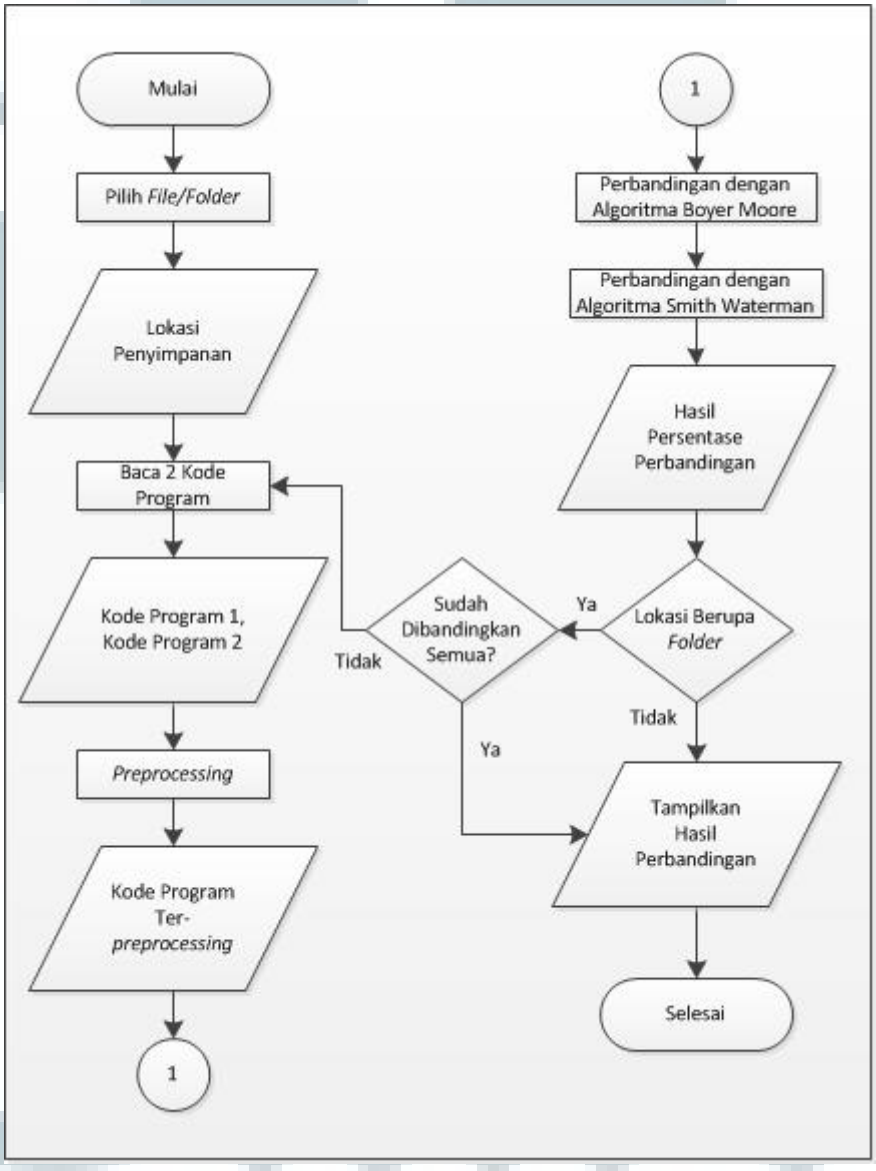
substring yang sama akan dilakukan pergeseran berdasarkan tabel *bad character*, sedangakan untuk pola berulang, pergeseran yang optimal dalam pencarian pola akan dihitung berdasarkan tabel *good suffix*. Hasli persentase dari algoritma *Boyer Moore* didapat berdasarkan jumlah kecocokan dan jumlah *pattern.*

Dalam pendeteksian plagiarisme struktural, hasil *preprocessing* juga dicocokkan dengan algoritma *Smith-Waterman* dalam pencocokan berdasarkan baris. Algoritma Smith-Waterman menentukan nilai kemiripan struktur kedua kode program dengan menghitung sebuah *scoring matrix* dimana kemiripan setiap dua baris kode dihitung dengan mempertimbangkan kode-kode sebelumnya pada kedua kode program. *Cell* dengan nilai terbesar dalam *scoring matrix* tersebut kemudian di *traceback* untuk dapat merekonstruksi kode program berdasarkan *insertion* dan *deletion* yang terdeteksi. Hasil persentasi dari algortima Smith-Waterman didapat berdasarkan jumlah *match, insertion* dan *deletion* kode program yang sudah disejajarkan.

Hasil kemiripan dari kedua kode program yang telah didapat ditambil kan beserta kode program yang sudah sejajarkan dari rekonstruksi *scoring matrix.* Selain itu ditampilkan juga kode program awal dan hasil *preprocessing.*

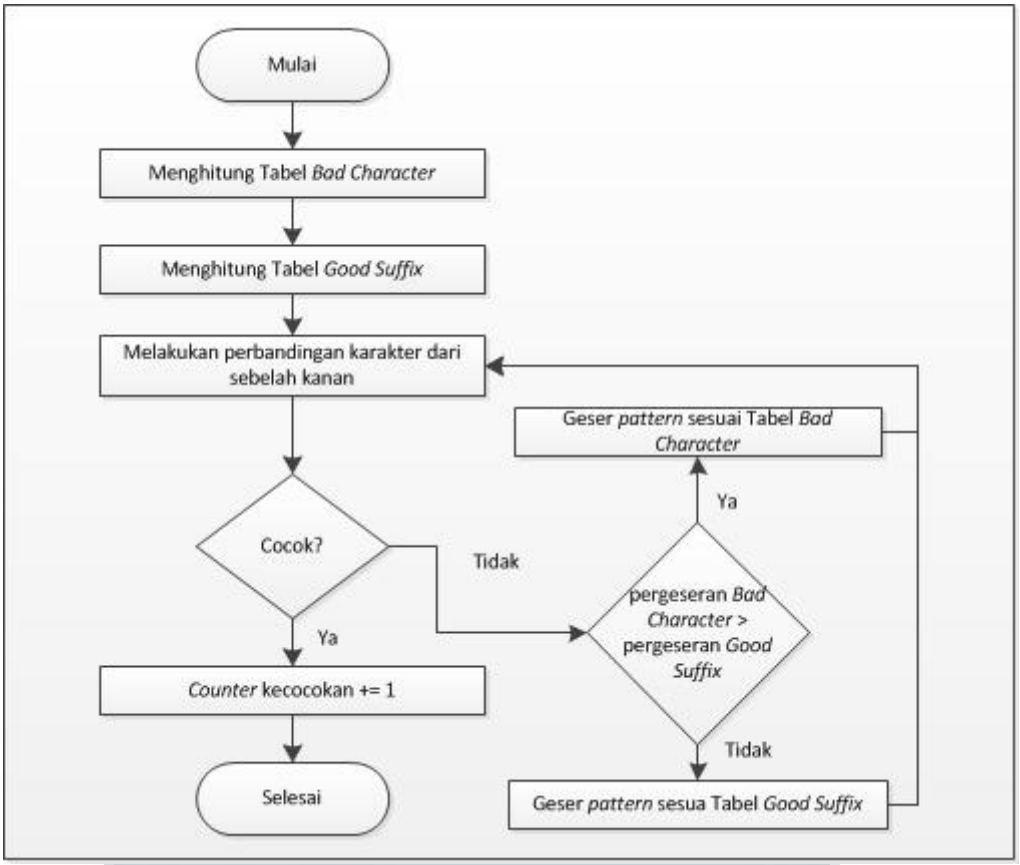
3.3 Spesifikasi Umum Aplikasi

Untuk mewujudkan solusi pembuatan aplikasi pendeteksi plagiarisme pada kode program dengan algoritma Boyer Moore dan Smith-Waterman, aplikasi ini melalui tahapan-tahapan yang dijelaskan pada *flowchart* berikut.



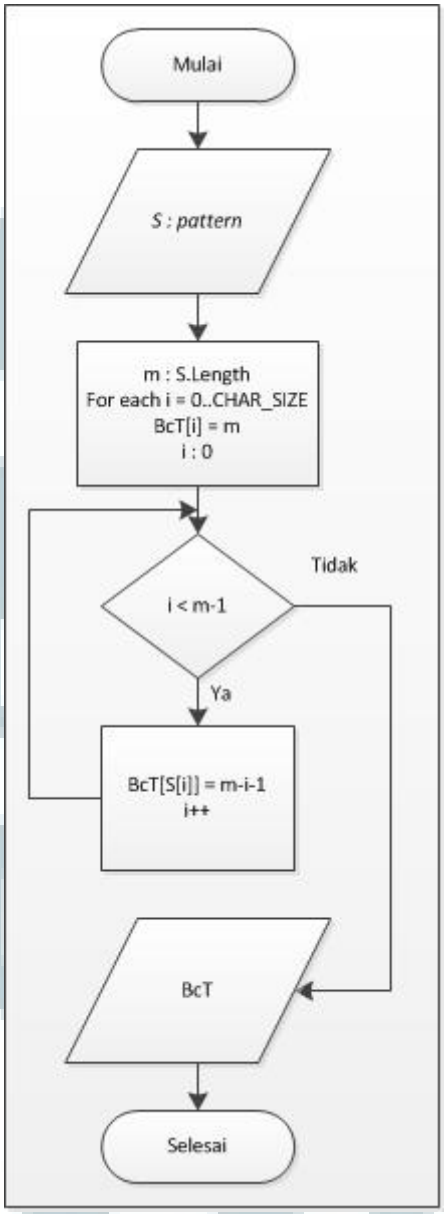
Gambar 3.2 *Flowchart* Aplikasi Pendeteksi Plagiarisme

Langkah pertama adalah *user* dapat memilih apakah akan membandingkan dua buah kode program saja atau membandingkan setiap dua kode program dalam satu *folder*. Dalam membandingkan dua kode program, *user* hanya dapat memilih kode program ber-ekstensi .c atau .cpp, sedangkan jika memilih *folder*, *user* harus memilih *folder* yang berisi sedikitnya dua buah kode program ber-ekstensi .c atau .cpp. Dari pilihan *user* tersebut akan dibaca dua buah kode program yang kemudian akan melalui tahap *preprocessing* agar hasil perbandingan lebih akurat. Dalam tahap *preprocessing* kode program akan dimodifikasi dengan penghapusan komentar, *header*, isi *string*, spasi, dan beberapa tanda baca dan simbol. Dalam *preprocessing* juga dilakukan penyeragaman bentuk *if, for, do, do…while, conditional statement*, dan penamaan variabel, fungsi, dan parameter. Setelah tahap *preprocessing* menghasilkan kode program ter-*preprocessing*, aplikasi akan membandingkan kode program ter-*preprocessing* tersebut dengan algoritma Boyer Moore. Setiap pencocokkan melalui tahap-tahap dalam algoritma Boyer Moore pada gambar 3.3.



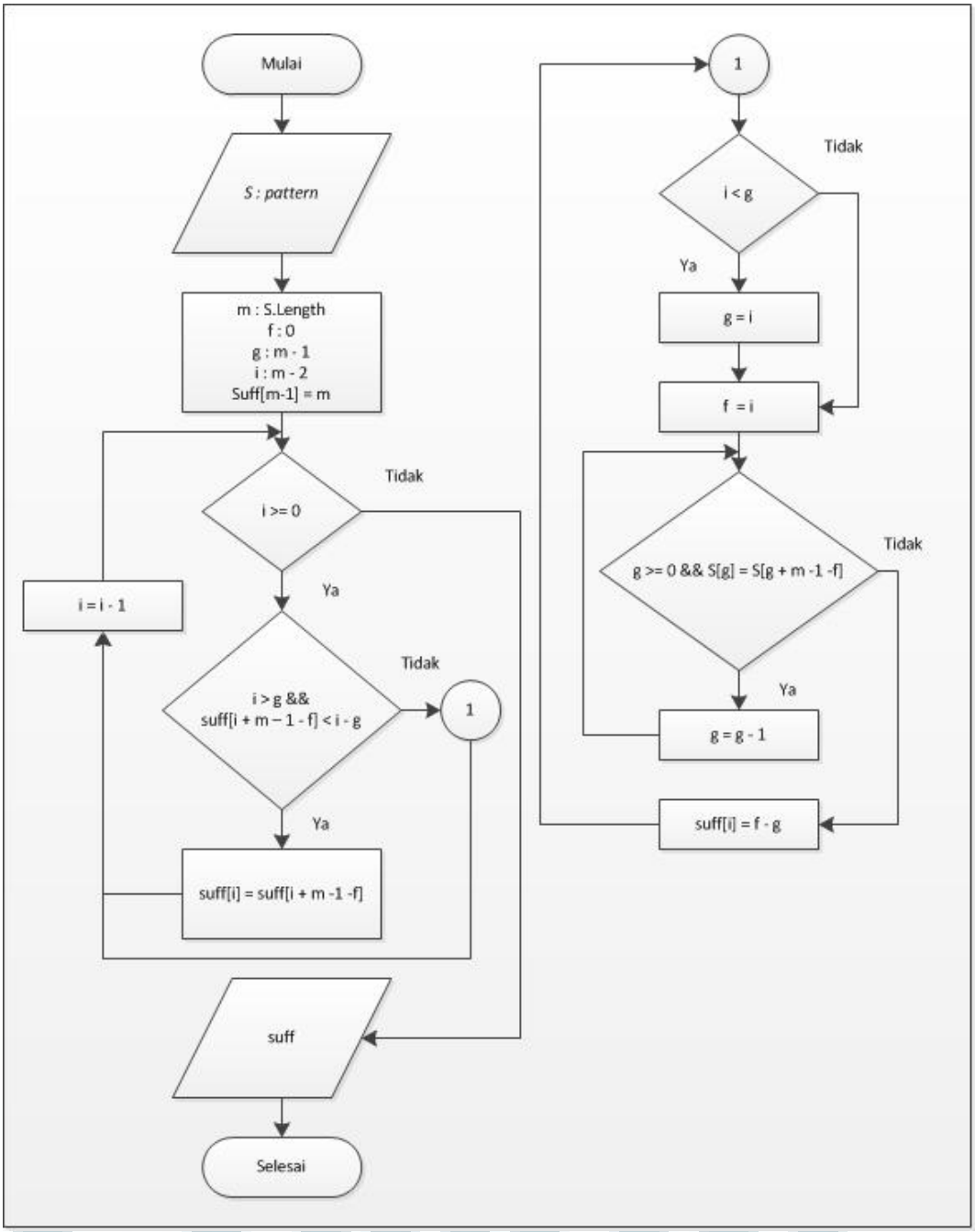
Gambar 3.3 *Flowchart* algortima Boyer Moore

Gambar 3.3 merupakan *flowchart* algoritma Boyer Moore dalam setiap pencocokkan yang dilakukan. Sebelum dibandingkan, aplikasi akan menentukan kode program yang lebih besar sebagai *text*, dan kode program yang lebih kecil sebagai penguji. Selanjutnya setiap *string* pada penguji akan dianggap *pattern* dan dihitung tabel *bad character* nya sesuai gambar 3.4.



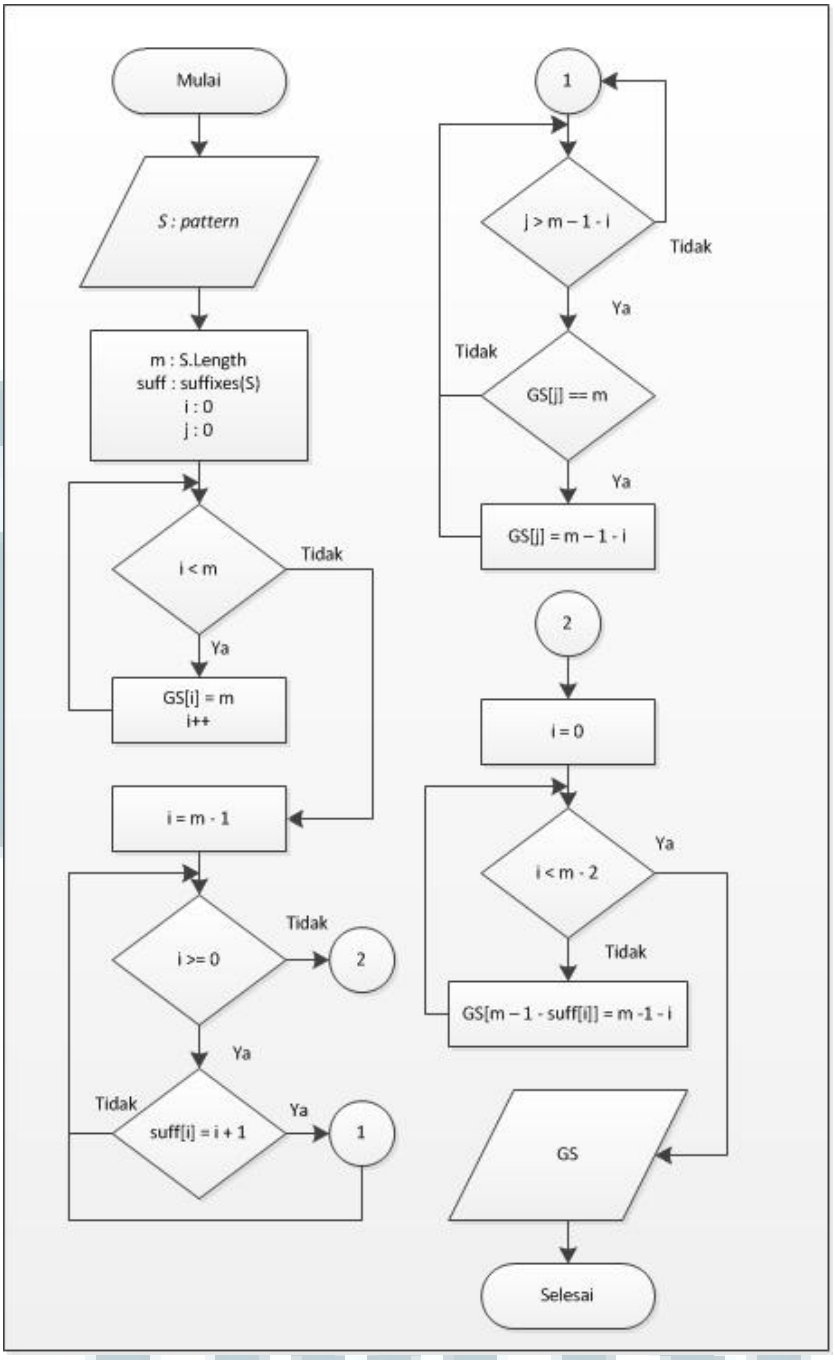
Gambar 3.4 *Flowchart* Tabel *Bad Character*

Tabel *bad character* menghitung pergeseran *pattern,* dengan maksimal pergeseran karakter sebesar panjang *pattern* bila karakter tidak terdapat dalam *pattern*. Untuk karakter yang terdapat dalam *pattern* jumlah pergeseran akan ditentukan index dari karakter tersebut dalam *pattern*. Bila ditemukan karakter yang sama nilai pergeseran yang disimpan adalah index terkecil.



Gambar 3.5 *Flowchart* Tabel *suffixes*

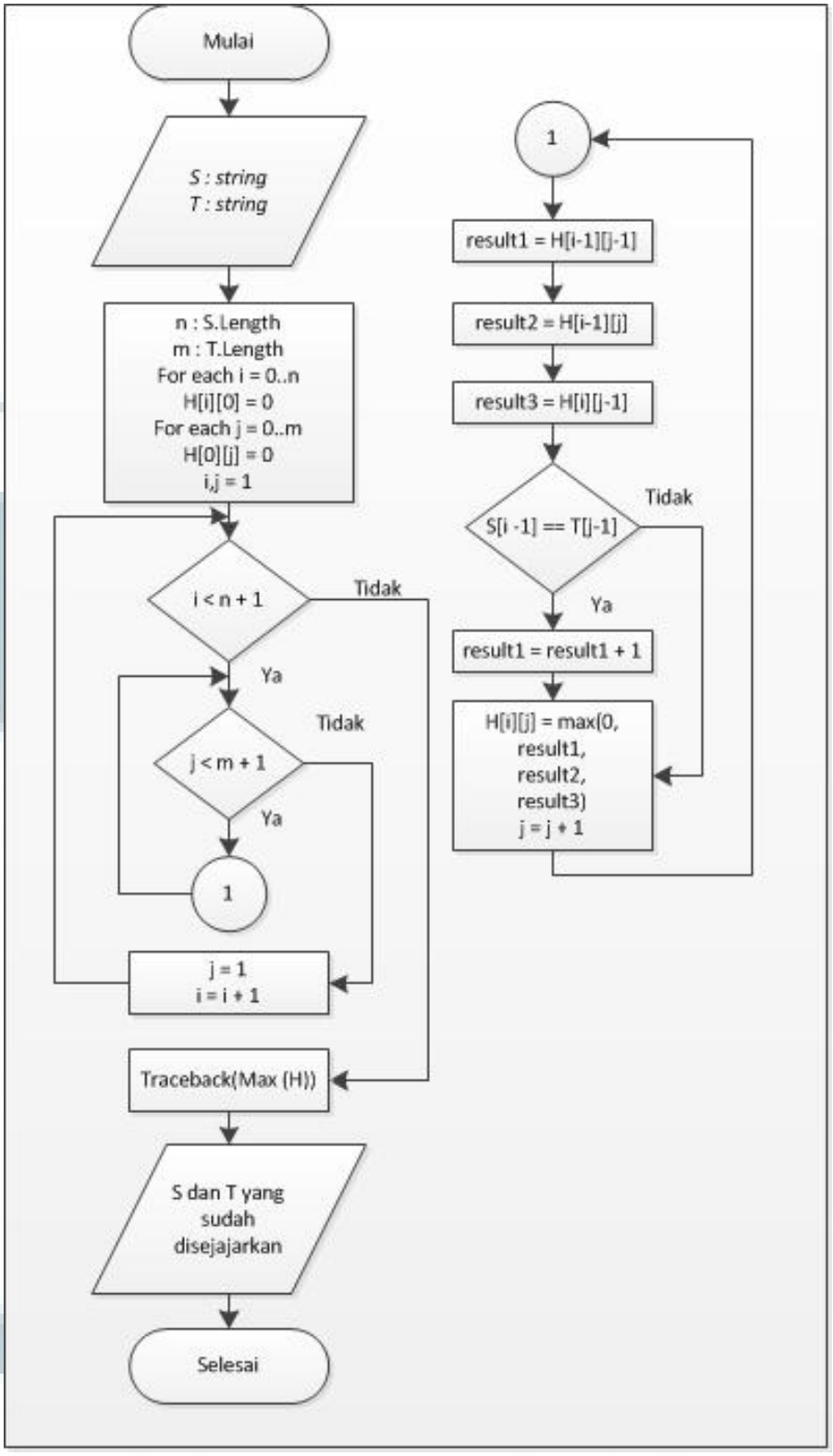
Gambar 3.5 merupakan tahap selanjutnya dari algoritma Boyer Moore, yaitu perhitungan tabel *suffixes* untuk mengetahui apakah terdapat *substring* yang berulang dalam *pattern,* hasil tabel ini digunakan dalam menghitung tabel *good-suffix*.



Gambar 3.6 *Flowchart* Tabel *good-suffix*

Tabel *good-suffix* lah yang menentukan pergeseran karakter bila dalam *pattern* terdapat *substring* yang berulang, dengan menentukan pergesaran karakter yang optimal dan tidak melewatkan *string* yang mempunyai kemungkinan kecocokan terhadap *pattern*.

Setelah menghitung ketiga tabel tersebut dimulailah perbandingan *pattern* terhadap *text.* Perbandingan *pattern* terhadap *text* dilakukan dari kanan, tetapi posisi *pattern* terhadap *text* tetap dilakukan dari kiri, dengan pencocokkan tiap karakter pada *pattern* dengan tiap karakter pada *text,* bilang terdapat karakter yang tidak cocok, *pattern* akan digeser sesuai dengan tabel *bad character* bila *pattern* tidak berulang, dan berdasarkan tabel *good suffix* bila *pattern* memiliki pola berulang. Bila pencocokkan tiap karakter menghasilkan kecocokan, maka *counter* kecocokkan akan bertambah dan aplikasi melanjutkan dengan *pattern* berikutnya hingga setiap *string* pada penguji selesai dicocokkan. Setelah didapat, nilai *counter* kecocokkan akan dihitung persentase kecocokannya terhadap jumlah *pattern.*

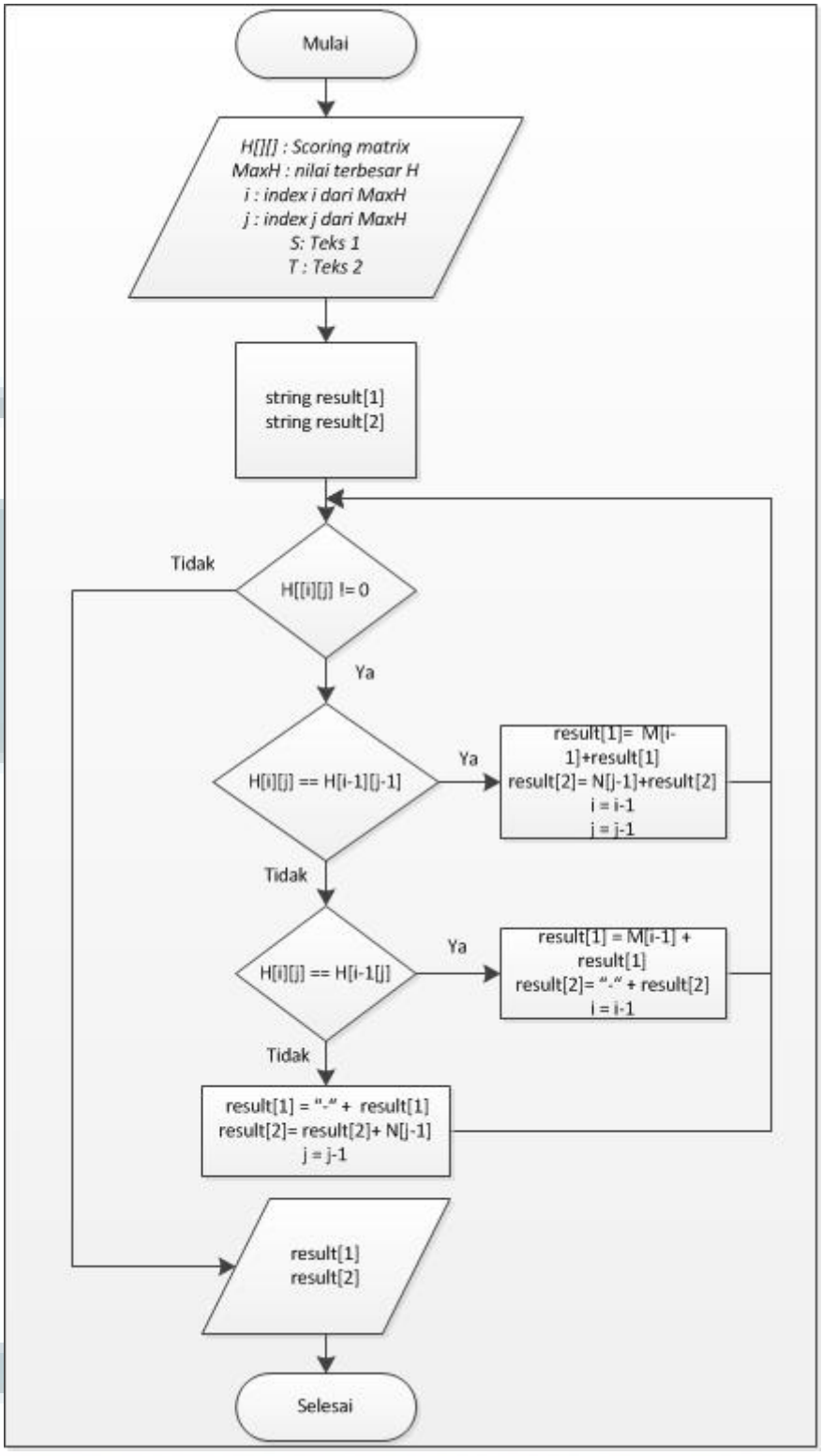


Gambar 3.7 *Flowchart* algoritma Smith-Waterman

Selain menggunakan algoritma Boyer Moore kode program yang telah di-*preprocessing* juga melalui perbandingan dengan algoritma Smith-Waterman untuk menghitung kecocokkan berdasarkan baris dan melakukan penyejajaran struktur kode program terhadap *insertion* dan *deletion*. Dalam algoritma Smith-Waterman tidak ditentukan kode program *text* ataupun penguji, kode program hanya dibedakan sebagai Teks S dan Teks T.

Integer i dan j sebagai *counter* pengulangan untuk mengisi *scoring matrix*. *Scoring matrix* tersebut berfungsi untuk mencari penyejajaran yang optimal berdasarkan struktur baris kedua kode program Dengan nilai tiap cell pada *matrix* dihitung dari tiga cell sebelumnya (H[i-1][j-1], H[i-1][j], dan H[i][j-1]), nilai tiap cell akan menentukan *match, mismatch, deletion,* atau *insertion* yang optimal.

Setelah *scoring matrix* selesai dihitung dilakukan *traceback* untuk merekonstruksi kode program yang telah disejajarkan. Gambar 3.8 menunjukkan *flowchart* langkah-langkah dalam *traceback scoring matrix.*



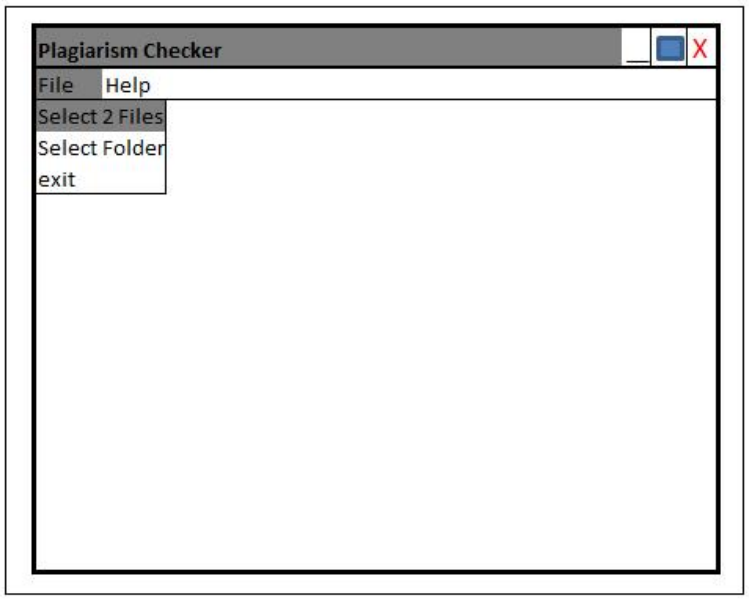
Gambar 3.8 *Flowchart Traceback Scoring Matrix*

Dalam proses *traceback* dari nilai terbesar dalam *scoring matrix*, cell tersebut dibandingkan dengan cell diagonal (i-1, j-1), jika nilai cell sama, berarti terjadi *match* atau *mismatch* di cell tersebut. Jika tidak sama cell akan dibandingkan dengan cell vertical (i-1,j), jika nilai cell sama berarti terjadi *insertion* pada S dan *deletion* pada T. Sebaliknya, jika tidak sama berarti terjadi *deletion* pada S dan *insertion* pada T.

Setelah melakukan *traceback* hasil *result[1]* dan *result[2]* akan ditampilkan beserta persentase kecocokan kode program yang telah disejajarkan. Selain itu ditampilkan juga kode program awal dan kode program *hasil preprocessing*.

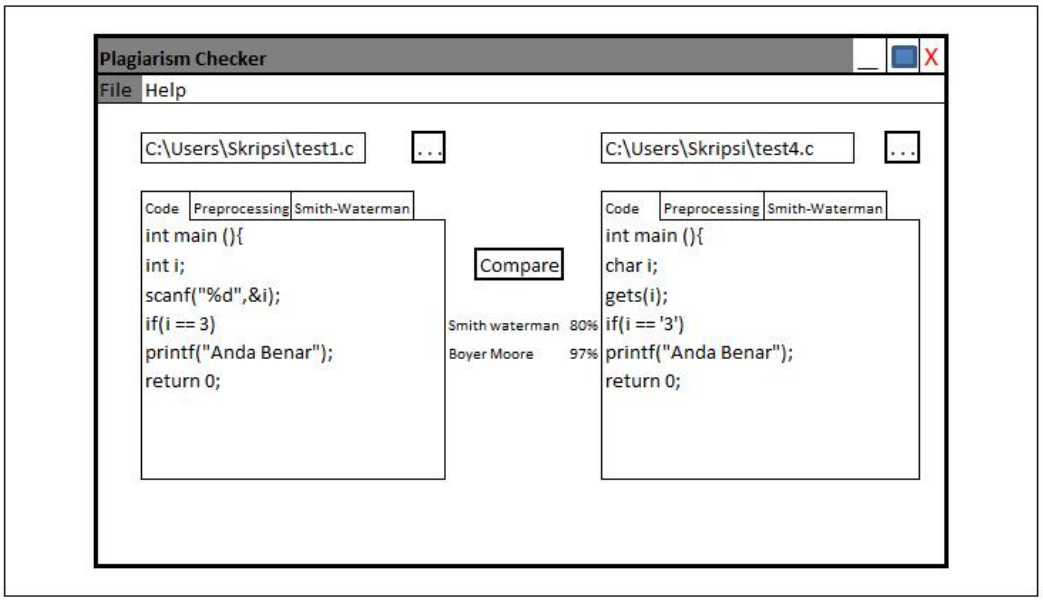
3.4 Desain Antarmuka Aplikasi

Berikut adalah desain antarmuka dari aplikasi pendeteksi plagiarisme pada kode program. Gambar 3.9 adalah halaman utama dari aplikasi.



Gambar 3.9 Desain Antarmuka Apliakasi Pendeteksi Plagiarisme pada Kode Program

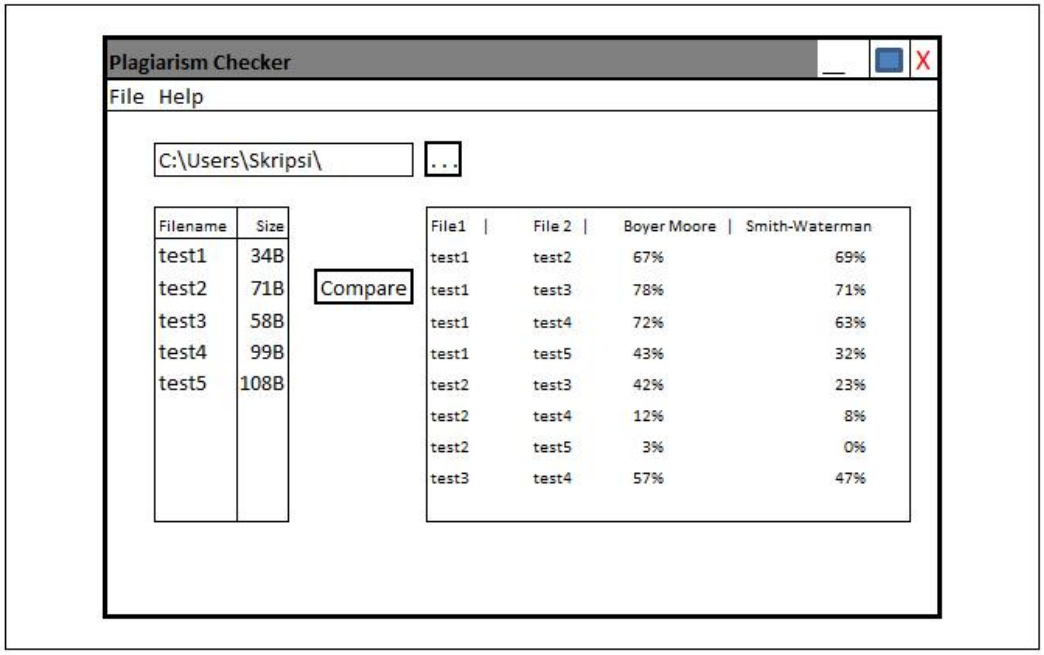
Halaman utama berfungsi sebagai halaman awal saat aplikasi dibuka dan sebagai pusat dari pemilihan menu yang disediakan. Pada halaman ini terdapat menu utama "*File" dan "Help",* dimana menu "*File"* terdiri dari submenu "*Select 2 Files", "Select a Folder",* dan "*Exit"* serta "*Help*" yang terdiri dari submenu "*How To Use"* dan *"About"*.



Gambar 3.10 Desain Antarmuka Halaman "*Select 2 Files*"

Gambar 3.10 adalah tampilan halaman jika *user* memilih submenu "*Select 2 Files*". Pada halaman ini terdapat tombol "**. . .**" yang berfungsi untuk memilih *file* pertama dan *file* kedua yang akan dibandingkan. Ketika ditekan akan memunculkan *Open File Dialog* untuk memilih *file*. Setelah itu terdapat tiga buah *tab*, yang berfungsi menampilkan kode programa awal, kode program setelah *preprocessing* dan kode program hasil penyejajaran algoritma Smith-Waterman. Setelah *user* memilih *file*, isi *file* tersebut akan ditampilkan pada *tab "Code"*. Kemudian user dapat menekan tombol "*Compare*" yang berfungsi memulai proses pendeteksian plagiarisme. Setelah proses selesai hasil kode program ter-

preprocessing akan ditambpilkan pada *tab "Preprocessing"* dan kode program yang sudah disejajarkan pada *tab "Smith-Waterman"*. Diantara kedua *tab* tersebut ditampilkan juga hasil persentase kemiripan dari kedua algoritma yang digunakan.



Gambar 3.11 Desain Antarmuka Halaman *"Select a Folder"*

Pada gambar 3.11 ditunjukkan desain untuk halaman submenu "*Select a Folder*". Pada halaman ini juga terdapat tombol "**. . .**" yang berfungsi untuk memilih *folder* yang akan diproses. Setelah memilih *folder*, daftar *file* kode program berekstensi .c dan .cpp dalam *folder* tersebut akan ditampilkan pada *listview*. Selain itu, terdapat tombol "*Compare*" yang berfungsi untuk memulai proses perbandingan setiap *file* yang ada pada *folder* yang dipilih. Hasil persentase akan ditampilkan pada *textbox* yang ada di sebelah kanan tombol.