BAB III

METODOLOGI DAN PERANCANGAN

3.1 Metode Penelitian

Pada bab sebelumnya telah dijelaskan mengenai dasar-dasar teori yang digunakan dalam penelitian seperti penjelasan mengenai algoritma-algoritma apa saja yang digunakan. Selanjutnya penelitian dimulai dengan tahapan-tahapan sebagai berikut:

1. Studi Literatur

   Pengumpulan data-data yang diperlukan serta mempelajari konsep-konsep yang akan diperlukan dalam pelaksanaan penelitian. Pengumpulan data dilakukan dengan mencari dan mempelajari jurnal-jurnal yang berhubungan dengan penelitian yang dilakukan serta mencari artikel-artikel yang dapat membantu penelitian.

2. Perancangan *Design* Aplikasi

   Perancangan *design* aplikasi dilakukan dalam dua tahap yaitu

   a. Merancang *flowchart diagram* sebagai pemodelan aliran data dalam aplikasi. Aliran data yang diperlukan dalam pembangunan aplikasi akan dicatat dan dibuat diagramnya sehingga mempermudah dan memperjelas alur proses penulisan kode.

   b. Merancang GUI (*Graphical User Interface*). Tampilan antar-muka merupakan salah satu aspek penting yang harus diperhatikan. Aplikasi harus dibuat *user-friendly* sehingga tidak membuat *user* kesulitan dalam

penggunaan aplikasi yang dibangun. Tampilan aplikasi harus dirancang sesuai dengan aturan-aturan interaksi manusia dan komputer.

3. Pembuatan Aplikasi

   Proses penulisan kode untuk membangun aplikasi. Aplikasi berbentuk *windows form application* yang merupakan *desktop application*. Aplikasi dibangun menggunakan *framework* .NET Microsoft Visual Studio 2008 dengan menggunakan bahasa pemrograman C#.

4. Uji Coba

   Setelah aplikasi selesai dibangun, dilakukan proses uji coba untuk mengetahui bagaimana aplikasi berjalan. Uji coba dilakukan dengan cara sebagai berikut.

   a. Tahap pengujian *software* yang dibuat apakah sudah sesuai dengan *requirement* atau belum.

   b. Mencari kekurangan program dan *bug*. Jika terdapat *bug* atau kekurangan, maka aplikasi akan diperbaiki hingga *bug* tidak terjadi lagi dan bagian-bagian yang dirasa belum mencukupi telah dikembangkan sehingga dapat memenuhi seluruh *requirement*.

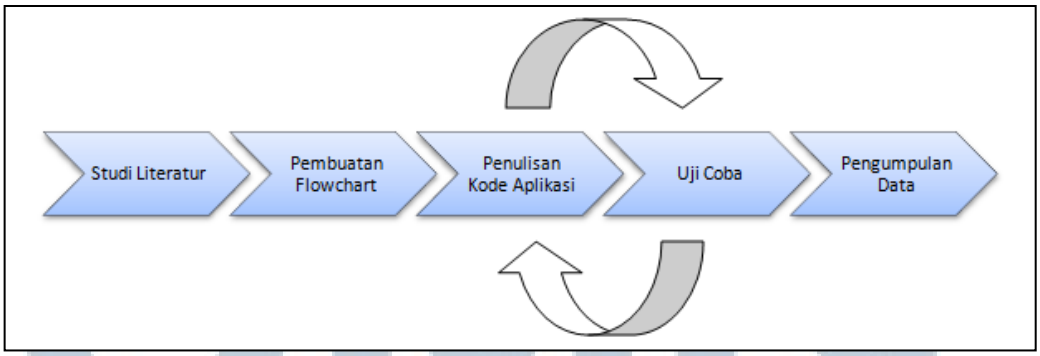
5. Pengumpulan Data Hasil

   Setelah aplikasi berjalan dengan sempurna, data-data yang dihasilkan akan dikumpulkan dan dicatat untuk kemudian hasilnya dibandingkan hingga dapat ditarik kesimpulan.

Proses pembangunan aplikasi ini menggunakan metode *prototyping* untuk *software development*-nya. Pembangunan aplikasi didahului dengan

pengumpulan data-data referensi yang berguna untuk pembangunan aplikasi serta pengumpulan *hardware* maupun *software* yang diperlukan. Ketika kebutuhan-kebutuhan tersebut telah terpenuhi, dibuatlah diagram-diagram yang merupakan rancangan dalam pembangunan aplikasi agar penulisan kode lebih terarah.

Setelah itu aplikasi akan dibangun mengikuti rancangan yang telah dibuat sebelumnya, kemudian diuji coba untuk mengetahui apakah aplikasi sudah bekerja dengan baik atau belum dan apakah sudah sesuai dengan kebutuhan atau belum. Tahap uji coba ini termasuk uji coba per modul dan uji coba aplikasi secara keseluruhan. Kedua tahap ini (pembangunan dan uji coba) dilakukan berulang-ulang hingga aplikasi berjalan sesuai dengan yang diharapkan.



Gambar 3.1 Proses pembuatan aplikasi

3.2 Design Sistem

Sebagai garis besar dari apa yang harus dilakukan oleh sebuah aplikasi, diperlukan rancangan umum mengenai apa saja yang diperlukan terutama input dan output karena akan langsung berhubungan dengan *user*. Rancangan untuk

input dan output yang diperlukan aplikasi yang dibangun tercantum sebagai berikut.

Masukan yang dibutuhkan oleh aplikasi:

1. *String* yang akan digunakan sebagai *string* dimana *keyword* akan dicari. *String* ini dapat ditulis secara manual maupun dengan *load* teks dari *text file* (doc, docx, php, html, txt). Untuk selanjutnya *string* ini akan disebut dengan *text*.
2. *String keyword* yang merupakan *string* yang akan dicari di dalam *text*. Untuk selanjutnya *string* ini akan disebut dengan *pattern*.

Output yang akan dihasilkan oleh aplikasi:

1. Jumlah total *pattern* yang ditemukan dalam *text*.
2. Waktu yang diperlukan oleh masing-masing algoritma untuk mencari *pattern* dalam *text*.
3. *Pattern* yang ditemukan di *text* akan ter-*highlight* dengan warna kuning.

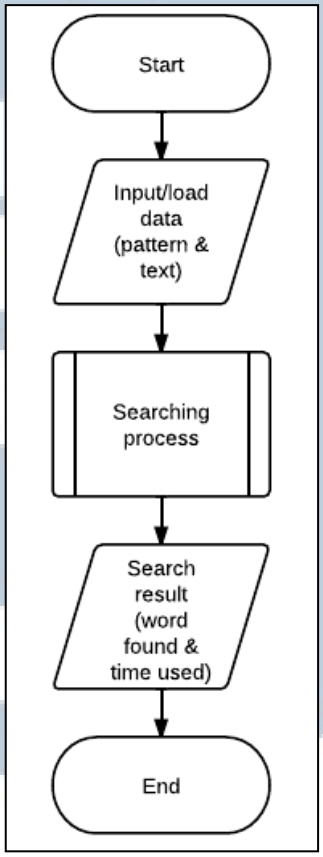
3.3 Diagram Sistem

Pada bagian ini akan dijelaskan proses-proses yang akan dilakukan aplikasi menggunakan media diagram.

3.3.1 *Flowchart*

Diagram alir atau *flowchart* merupakan bagan yang memperlihatkan urutan dan hubungan antar proses beserta instruksinya (Maing, 2008). Fungsinya adalah memberikan gambaran secara garis besar untuk program atau aplikasi yang dibuat.

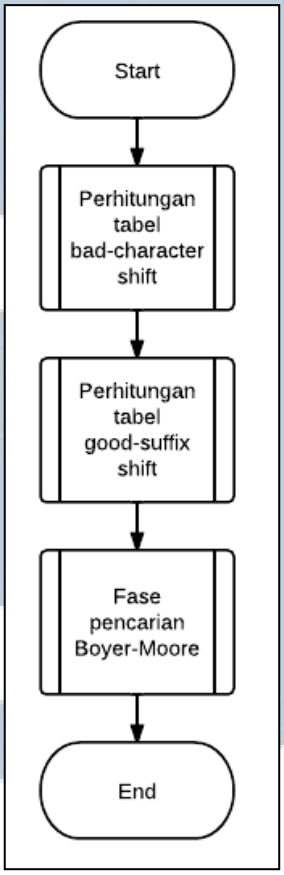
A. *Flowchart* program



Gambar 3.2 *Flowchart* aplikasi

Diagram menggambarkan alur kerja dari program yang digunakan untuk pengetesan kecepatan algoritma. Program dapat menerima input teks yang ditulis langsung oleh *user* atau mengambil teks dari *text file* yang telah ada di hard drive seperti txt *file*, doc atau docx *file*, html *file*, php *file*, dan *text file* lainnya. Untuk proses pencarian, diperlukan juga teks yang ingin dicari (*pattern*). Ketika data-data yang diperlukan telah dipenuhi, maka pencarian dapat dilakukan. Setelah pencarian dilakukan, akan ditampilkan jumlah hasil pencarian kata yang diinginkan serta waktu yang diperlukan untuk pencarian tersebut.

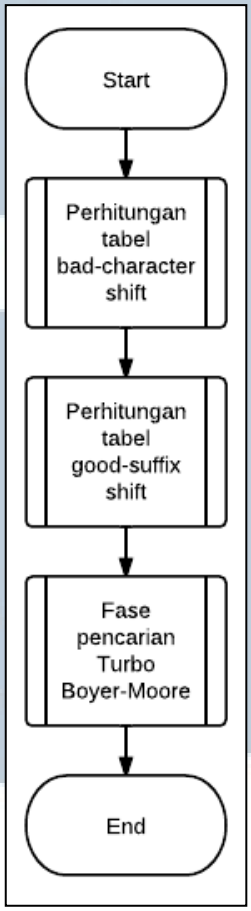
B. *Flowchart* algoritma Boyer-Moore



Gambar 3.3 *Flowchart* garis besar algoritma Boyer-Moore

Dalam proses pencarian dari algoritma Boyer-Moore, proses dimulai dengan membangun tabel *bad character shift* dan tabel *good suffix shift*. Kedua tabel ini diperlukan untuk menentukan seberapa banyak *pattern* bisa bergeser atau melompat dalam proses pencocokannya. Algoritma ini bergerak mencocokan teks dengan *window* yang bergerak dari kiri ke kanan teks dan proses pencocokan karakter yang dimulai dari kanan ke kiri. Ketika ditemukan karakter yang tidak cocok, maka *pattern* akan digeser sesuai dengan nilai yang terdapat di tabel *bad character shift* atau *good suffix shift* sesuai dengan aturan yang berlaku.

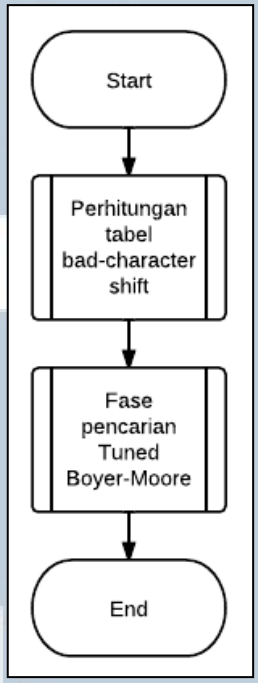
C. *Flowchart* algoritma Turbo Boyer-Moore



Gambar 3.4 *Flowchart* garis besar algoritma Turbo Boyer-Moore

Pencarian pada Turbo Boyer-Moore pada dasarnya sama dengan algoritma pendahulunya, Boyer-Moore. Kedua algoritma ini diharuskan membangun dua tabel *pre-processing* yang sama yaitu tabel *bad character shift* dan *good suffix shift*. Yang membedakan adalah variabel pengingat pergeseran yang berada di proses pencarian Turbo Boyer-Moore yang akan dibahas pada sub bab selanjutnya yang akan membahas tentang proses pencarian dalam algoritma Turbo Boyer-Moore.

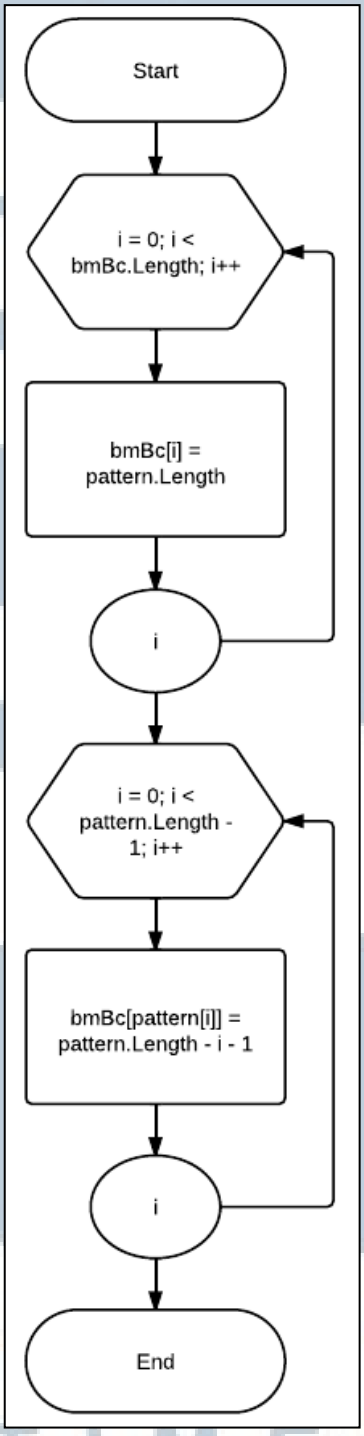
D. *Flowchart* algoritma Tuned Boyer-Moore



Gambar 3.5 *Flowchart* garis besar algoritma Tuned Boyer-Moore

Gambar 3.5 menunjukkan garis besar alur algoritma Tuned Boyer-Moore. Algoritma ini memerlukan satu tabel *pre-processing* yaitu tabel *bad character shift* yang kemudian nilai-nilai dari tabelnya akan digunakan untuk menentukan seberapa besar pergeseran yang perlu dilakukan ketika ditemukan ketidakcocokan dalam proses pencarian.

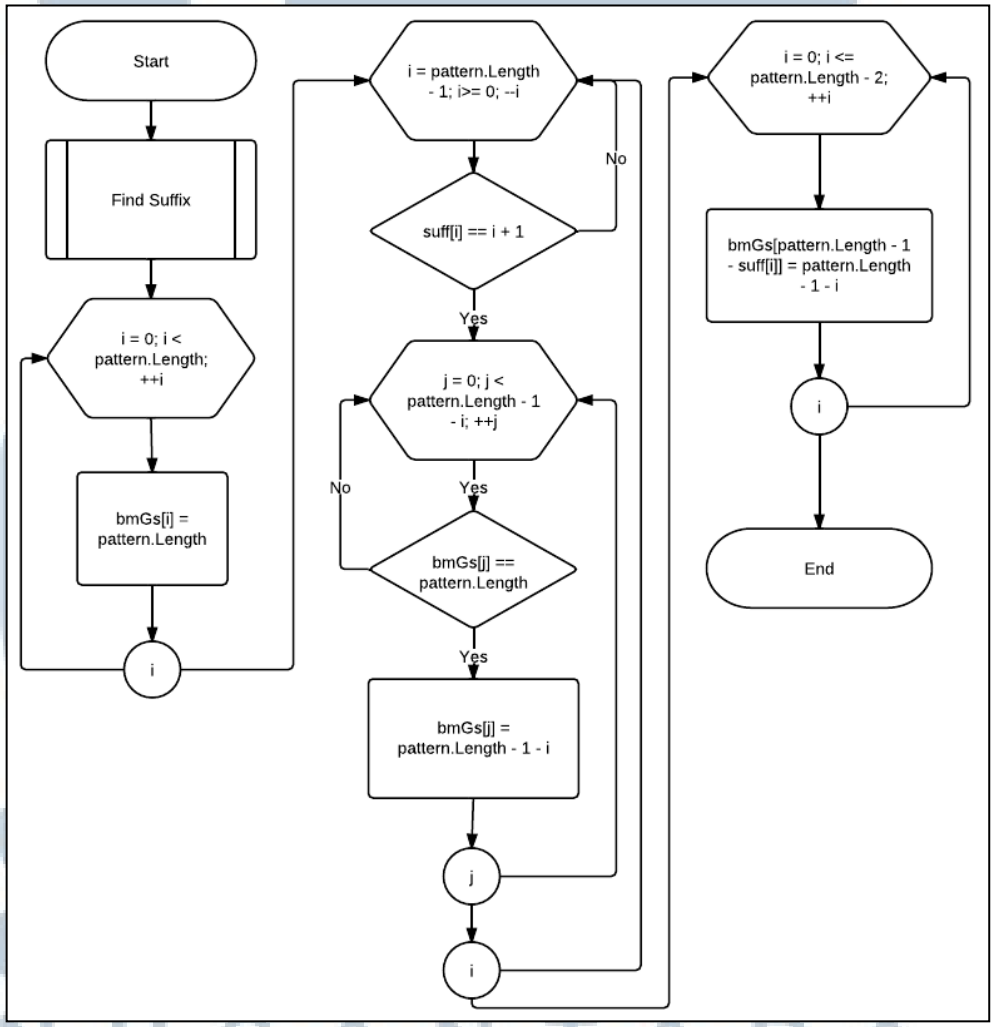
E. *Flowchart* proses perhitungan tabel *bad-character shift*



Gambar 3.6 *Flowchart* proses pembangunan tabel bad-character shift

Tabel *bad character shift* berisi nilai-nilai pergeseran yang dapat dilakukan untuk setiap karakter yang terdapat di alfabet, nomor, dan tanda baca yang umum digunakan. Setiap karakter yang ada di *pattern* diberi nilai sesuai dengan ukuran jauhnya karakter tersebut dari karakter paling akhir dari *pattern* dan untuk karakter yang tidak terdapat di dalam *pattern* akan diberi nilai yang sama yaitu panjang dari *pattern* yang dimasukkan.

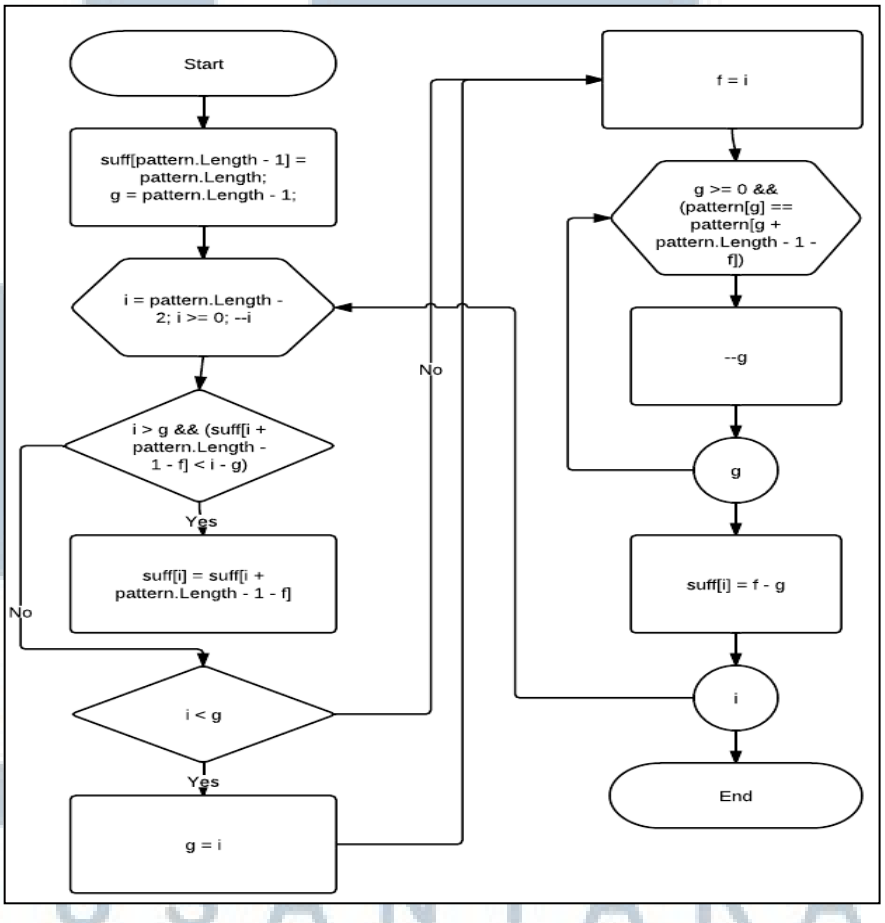
F. *Flowchart* proses perhitungan tabel *good-suffix shift*



Gambar 3.7 *Flowchart* proses pembangunan tabel *good-suffix shift*

Tabel *good suffix shift* berisi nilai yang dapat digunakan untuk pergeseran ketika ketidakcocokan ditemukan berdasarkan karakter pada posisi keberapa yang menyebabkan ketidakcocokan. Untuk menentukan nilai-nilainya, perlu dihitung dulu tabel *suffix* yang fungsinya memberi tanda jika terdapat perulangan akhiran. Dari tabel *suffix* inilah tabel *good suffix shift* akan dihitung. Nilai dari tiap karakter yang ada dalam *pattern* bergantung pada apakah adanya perulangan akhiran (*suffix*) atau tidak. Semakin banyak perulangan, semakin kecil nilai pergeseran.

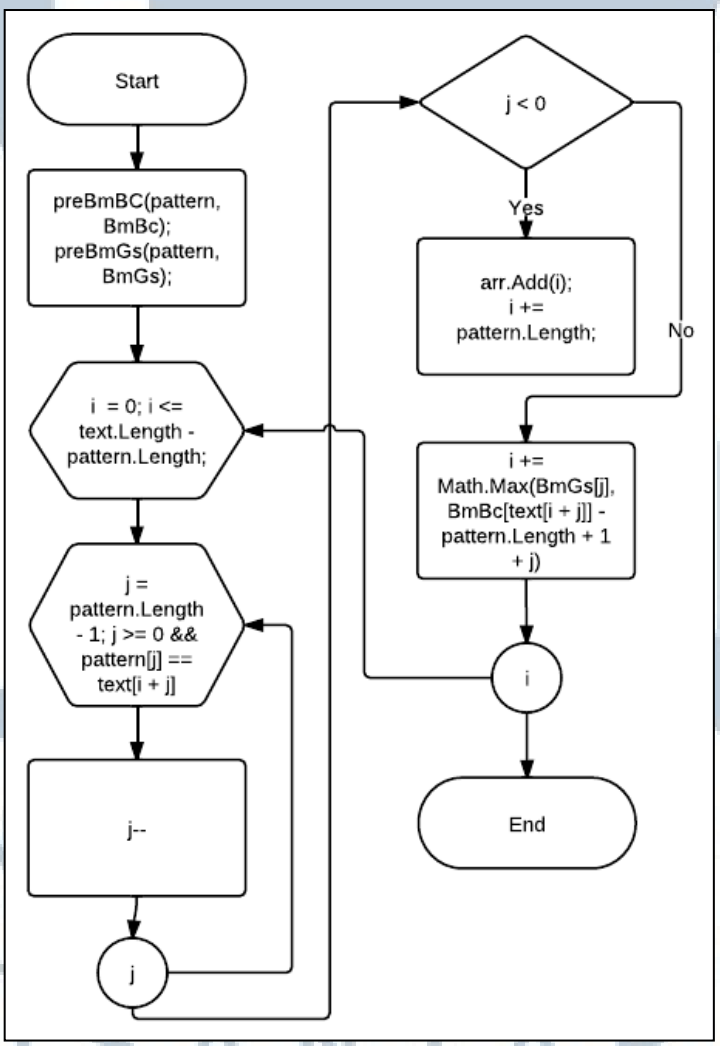
G. *Flowchart* proses perhitungan tabel *suffix*



Gambar 3.8 *Flowchart* proses perhitungan tabel *suffix*

Gambar dibawah menunjukkan alur perhitungan tabel *suffixes* yang berguna untuk menghitung tabel *good suffix shift* nantinya. Tabel ini berisi nilai-nilai dari tiap karakter yang ada di *pattern* yang menunjukkan adanya perulangan akhiran (*suffix*) atau tidak dan dimana letak perulangan tersebut sehingga ketika proses perhitungan tabel *good suffix shift*, dapat diketahui seberapa banyak karakter yang dapat digeser untuk pencocokan karakter yang selanjutnya.

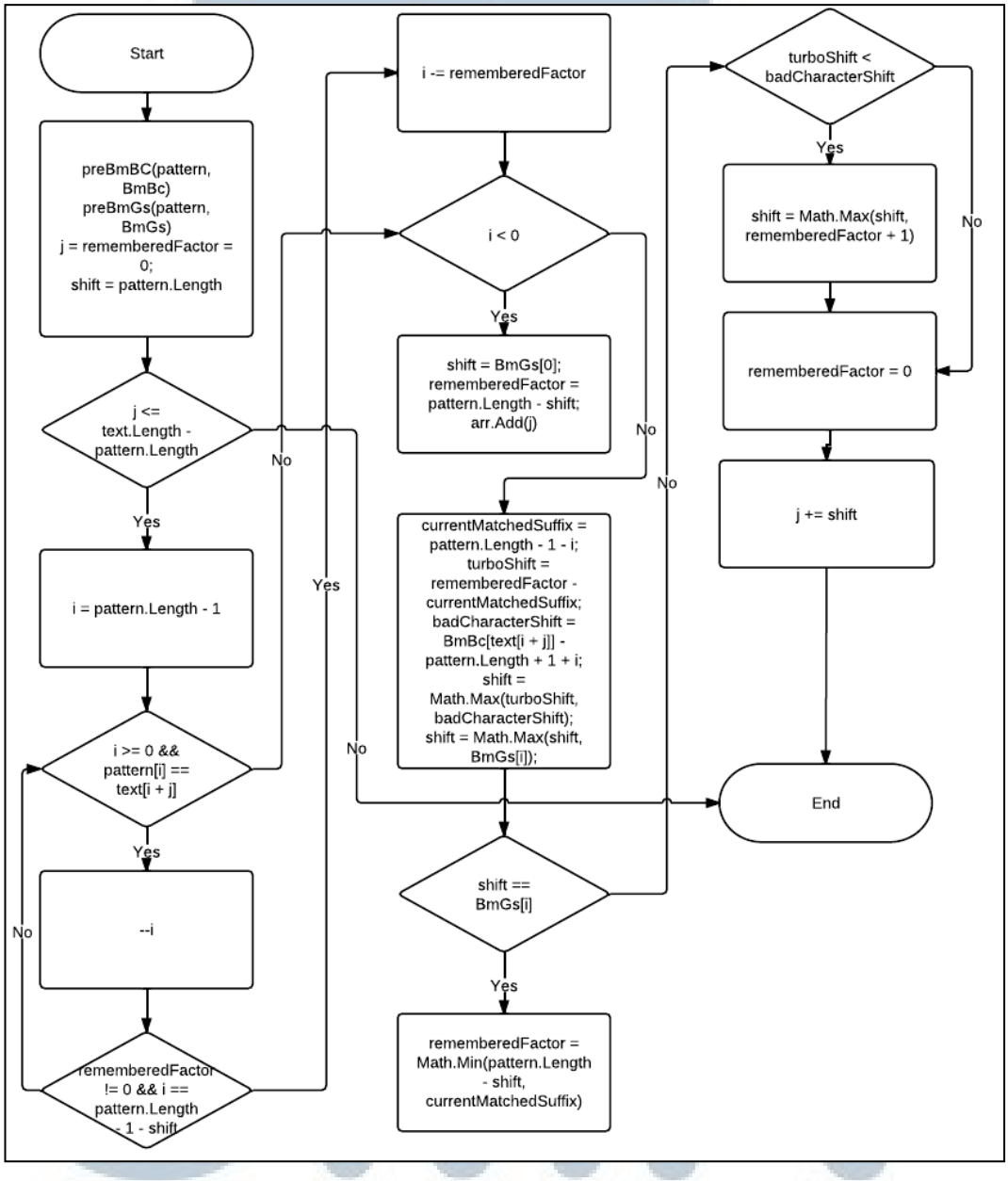
H. *Flowchart* proses fase pencarian Boyer-Moore



Gambar 3.9 *Flowchart* tahap pencarian algoritma Boyer-Moore

Dalam pencarian Boyer-Moore seperti yang ditunjukkan gambar 3.9 proses awal yang dilakukan adalah penempatan *window* dari *pattern* di *text* yang tersedia. Proses pencarian dimulai dari karakter paling kanan *pattern*. Setiap karakter akan dibandingkan satu per satu. Jika terjadi ketidakcocokan, maka akan dicek nilai pergeseran yang mungkin dilakukan ke tabel good *suffix shift* dan *bad character shift*. Nilai terbesar yang didapat akan diambil dan pergeseran *window* akan dilakukan sesuai dengan nilai tersebut.

I. *Flowchart* proses fase pencarian Turbo Boyer-Moore

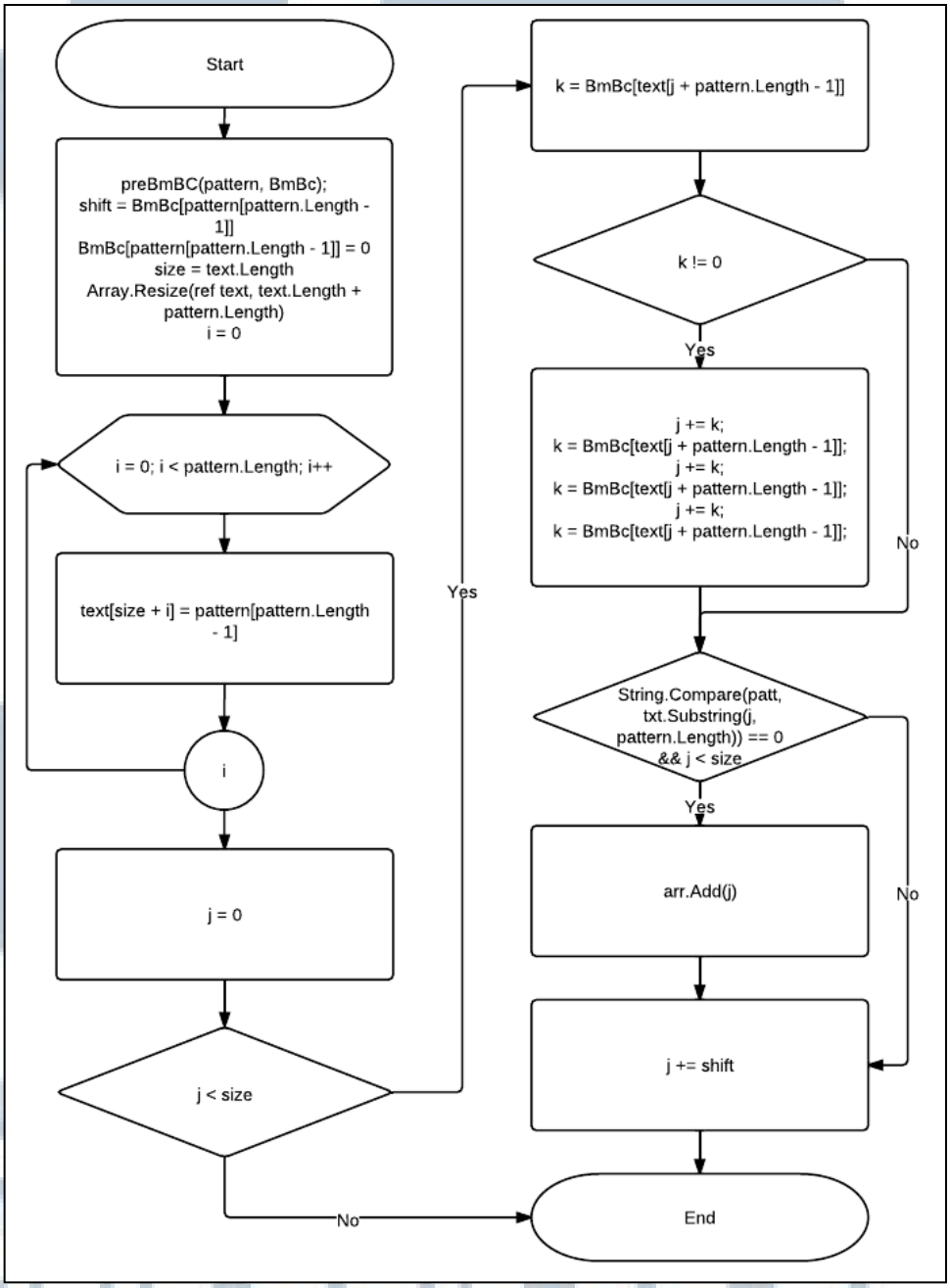


Gambar 3.10 *Flowchart* proses pencarian algoritma Turbo Boyer-Moore

Untuk fase pencarian dalam algoritma Turbo Boyer-Moore, proses yang dilakukan hampir sama dengan fase pencarian pada algoritma Boyer-Moore. Yang membedakan adalah adanya variabel yang berfungsi untuk menampung nilai pergeseran apabila di putaran sebelumnya nilai yang diambil untuk pergeseran

berasal dari tabel *good suffix shift*. Nilai ini nantinya akan digunakan sebagai kandidat nilai yang mungkin digunakan untuk melakukan pergeseran *pattern*.

J. *Flowchart* proses fase pencarian Tuned Boyer-Moore

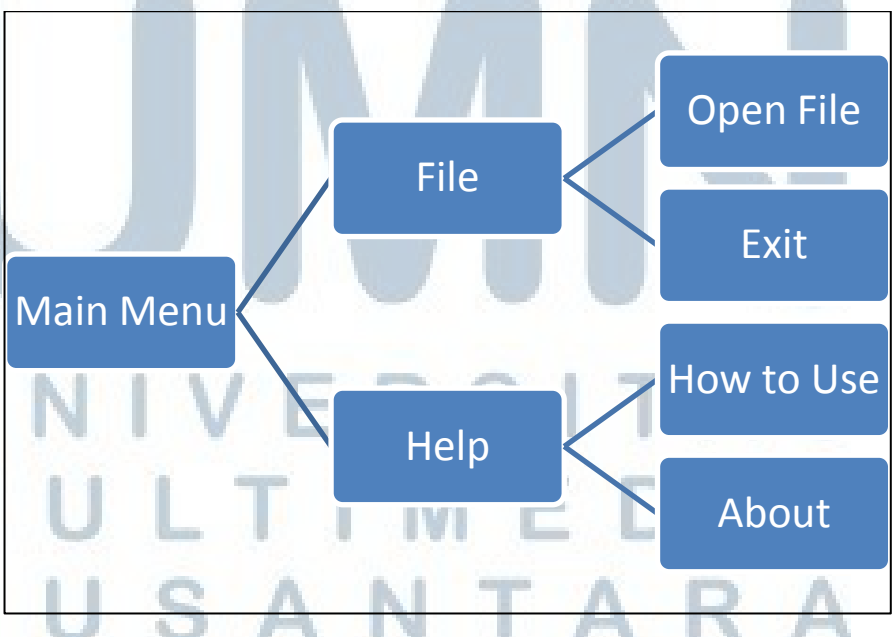


Gambar 3.11 *Flowchart* proses pencarian algoritma Tuned Boyer-Moore

Algoritma Tuned Boyer-Moore akan melakukan tiga kali *blind-shift* pada tahap awal pencariannya. *Shift* yang dilakukan pada tiga kali *blind-shift* ini mengikuti aturan pergeseran sesuai tabel *bad-character shift*. Setelah tiga kali dilakukan *blind-shift*, karakter dari *pattern* akan mulai dibandingkan dengan *text*. Jika terjadi ketidakcocokan, *pattern* akan digeser lagi sesuai dengan nilai *shift* yang sebelumnya telah diinisialisasi.

3.3.2 Rancangan Navigasi

Berikut adalah rancangan navigasi dari aplikasi yang akan dibangun untuk penelitian. Open File berfungsi untuk memanggil *window* untuk memilih file yang akan di-load isinya ke dalam aplikasi, Exit berfungsi untuk keluar dari aplikasi, How to Use memberikan informasi tentang cara penggunaan aplikasi, dan About berisi sedikit penjelasan mengenai aplikasi.

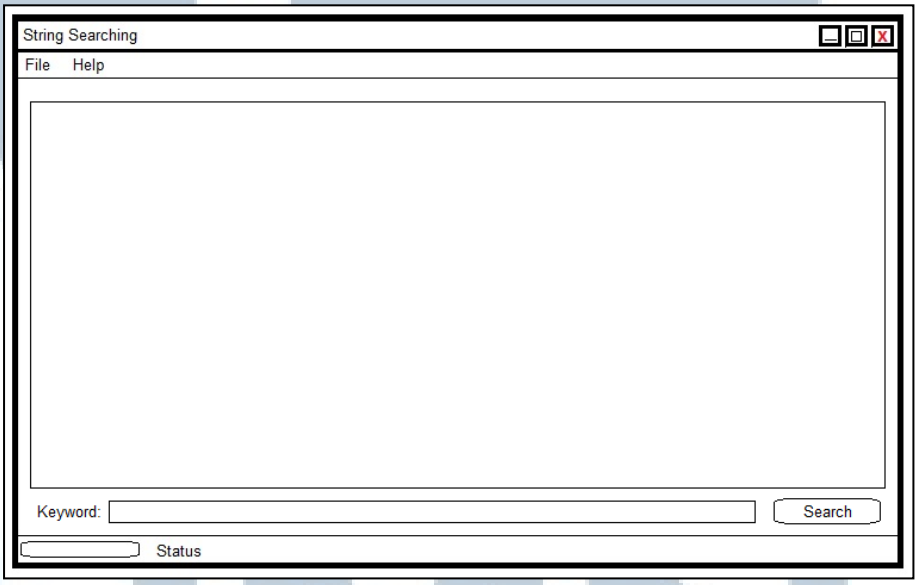


Gambar 3.12 Rancangan navigasi aplikasi

3.4 Sketsa Graphical User Interface Aplikasi

Sebagai penghubung antara *user* dengan aplikasi, tentu diperlukan tampilan atau *graphical user interface* yang simpel dan mudah dimengerti. Sketsa *user interface* ini merupakan rancangan garis besar yang akan mempermudah pembangunan aplikasi terutama dalam bidang *design*.

1. Halaman Utama



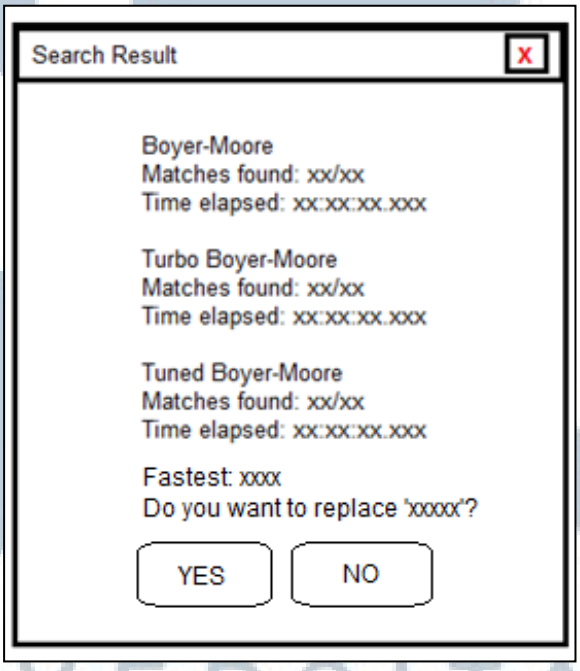
Gambar 3.13 Sketsa halaman utama

Halaman ini adalah halaman yang pertama akan muncul saat aplikasi dijalankan. Terdapat menu File yang memiliki *sub-menu* Open File dan Exit serta menu Help yang memiliki *sub-menu* How to Use dan About. Open File berfungsi untuk menampilkan *open file box* dimana *user* dapat memilih *file* yang ingin di-*load* isinya ke dalam aplikasi. *File* yang dipilih isinya akan muncul di *text box* yang disediakan. *Sub-menu* Exit berfungsi untuk menutup aplikasi. *Sub-menu*

How to Use akan menampilkan *step-by-step* bagaimana cara menggunakan aplikasi, sedangkan *sub-menu* About akan berisi sekilas tentang aplikasi ini.

Text box keyword adalah tempat *user* menulis *string* yang ingin dicari (*pattern*). Tombol *search* ketika ditekan akan memicu pencarian *string* dan menampilkan hasil *search* setelah proses *search* selesai dilakukan. Di paling bawah *form* terdapat *progress bar* yang berfungsi menunjukkan apakah suatu proses sedang berjalan atau tidak dan *label status* yang akan memberikan info apakah yang sedang aplikasi tersebut sedang lakukan.

2. Halaman *Output* Hasil *Search*

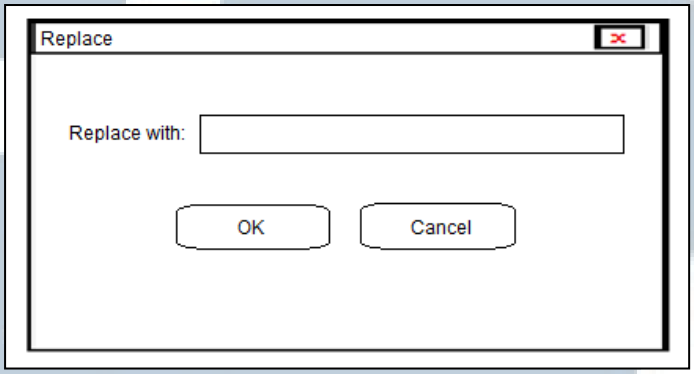


Gambar 3.14 Sketsa halaman *result*

Halaman ini ditampilkan setelah proses *search* selesai dilakukan. Halaman ini berisi informasi hasil *search* dari masing-masing algoritma. Informasi yang

diberikan adalah jumlah *pattern* yang ditemukan dalam *text* dan waktu yang dibutuhkan untuk mencari *pattern* tersebut.

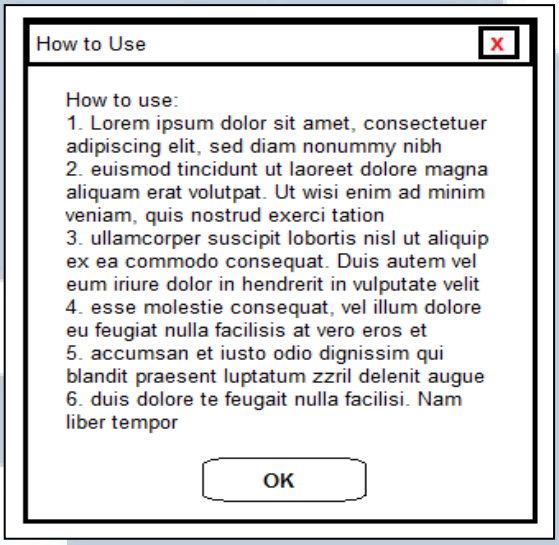
3. Halaman *Replace*



Gambar 3.15 Sketsa halaman *Replace*

Halaman ini berfungsi untuk memfasilitasi *user* yang ingin mengubah *string* yang baru saja dicarinya menjadi *string* lain. Terdapat sebuat *text box* yang berguna untuk menampung '*string* baru' yang nantinya akan digunakan untuk mengganti '*string* lama'. Terdapat pula tombol 'OK' untuk menyetujui proses *replace* dan 'Cancel' untuk membatalkan.

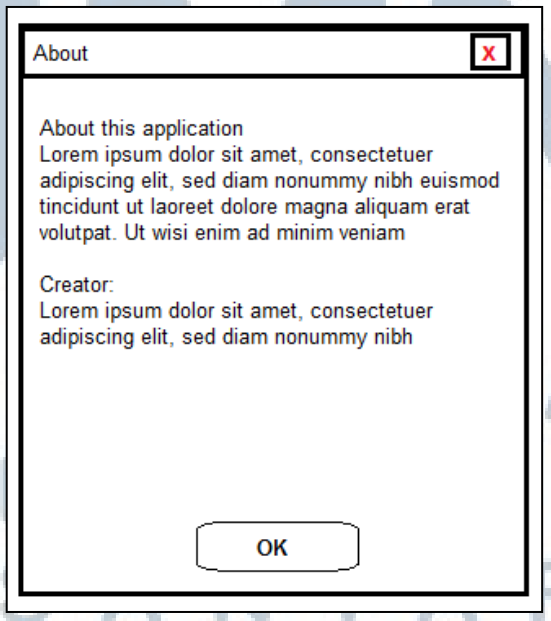
4. Halaman Cara Penggunaan



Gambar 3.16 Sketsa halaman cara penggunaan

Halaman ini berisi tentang *step-by-step* penggunaan aplikasi supaya *user* dapat menggunakan aplikasi dengan maksimal.

5. Halaman About



Gambar 3.17 Sketsa halaman About

Gambar 3.17 menunjukkan sketsa untuk halaman About. Halaman ini memberikan sedikit informasi tentang aplikasi serta fungsinya dan nama pembuat aplikasi.