Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

Team project ©2017
Dony Pratidana S. Hum | Bima Agus Setyawan S. IIP

BAB III

METODE DAN PERANCANGAN SISTEM

3.1 Metode Penelitian

Metode penelitian yang digunakan dalam penelitian ini antara lain adalah sebagai berikut.

1. Studi Literatur

   Penelitian dimulai dengan studi kepustakaan melalui berbagai macam media, baik media cetak maupun media internet. Studi yang dilakukan berfokus kepada penelitian-penelitian sebelumnya dalam bentuk jurnal maupun artikel, terkait dengan kompresi citra digital, algoritma transformasi Burrows-Wheeler, dan algoritma kompresi Lempel-Ziv-Welch.

2. Perancangan Aplikasi

   Tahap perancangan aplikasi mencakup perancangan alur kerja aplikasi serta perancangan tampilan atau *Graphical User Interface* (GUI) aplikasi dalam bentuk sketsa. Alur kerja aplikasi dirancang dengan menggunakan beberapa *flowchart* yang menjelaskan proses-proses yang ada dalam aplikasi.

3. Pembuatan Aplikasi

   Pada tahap pembuatan aplikasi, kode diimplementasi berdasarkan *flowchart* dan perancangan GUI yang telah dilakukan sebelumnya. Kode dibuat menggunakan aplikasi Visual Studio 2010 dengan bahasa pemrograman C#.

4. Uji Coba dan Evaluasi

   Setelah aplikasi dibuat, uji coba dilakukan untuk memperoleh data yang selanjutnya akan dievaluasi. Uji coba dilakukan menggunakan beberapa citra

Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016

digital yang diperoleh dari halaman *website Signal and Image Processing Institute, University of Southern California* (University of Southern California, 1981) untuk dikompresi menggunakan aplikasi hasil implementasi. Kemudian evaluasi dilakukan berdasarkan hasil dari uji coba yang diperoleh. Selain itu, evaluasi juga akan dilakukan dengan membandingkan aplikasi yang dibuat dengan aplikasi kompresi lainnya.

5. Penulisan Laporan

   Membuat laporan dari hasil penelitian yang telah dilakukan, mulai dari tahap studi literatur, perancangan aplikasi, pembuatan aplikasi, hingga uji coba dan evaluasi.

3.2 Alat yang Diperlukan

Dalam penelitian ini, digunakan perangkat keras dan lunak yang diperlukan untuk membuat aplikasi pendukung penelitian yang dapat berfungsi dengan baik. Berikut adalah perangkat-perangkat keras dan lunak yang digunakan.

1. Perangkat Keras

   Perangkat keras yang digunakan untuk penelitian adalah laptop ASUS X450J dengan spesifikasi sebagai berikut.

   a. Prosesor : Intel® Core™ i7 4700HQ
   b. Memori : 4 GB
   c. Kapasitas Penyimpanan : 1TB

2. Perangkat Lunak

   a. Sistem operasi Windows 10 Pro 64 bit.

Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016

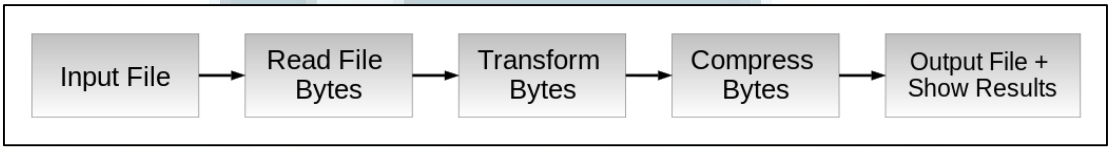
b. Visual Studio 2010 sebagai Integrated Development Environment (IDE) untuk penulisan kode aplikasi.

3.3 Fitur Aplikasi

Penelitian ini dilakukan dengan menggunakan sebuah aplikasi yang dibuat. Secara garis besar, aplikasi memiliki dua buah fitur, yaitu kompresi dan dekompresi. Pada sub bab selanjutnya akan dijelaskan secara singkat tahapan yang dilalui untuk melakukan kompresi serta dekompresi.

3.3.1 Kompresi

Kompresi dilakukan dalam 5 tahap yang mencakup memberikan masukan data berupa *file*, membaca *file*, mentransformasi, mengkompresi, dan menyimpan keluaran serta menampilkan hasil proses kompresi.



Gambar 3.1 Tahapan Kompresi Aplikasi

Tahapan-tahapan yang dilalui dalam proses kompresi dapat dijelaskan sebagai berikut.

1. *Input File*

Tahap pertama proses kompresi adalah *input file*. Pada tahap ini, pengguna akan diminta untuk memilih sebuah *file* citra digital untuk dikompresi. *File* citra

Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016

digital yang diterima oleh aplikasi adalah *file* dengan ekstensi jpg, png, gif, bmp, dan tiff.

2. *Read File Bytes*

Pada tahap ini, aplikasi akan membaca *byte-byte* data dari *file* yang dipilih untuk dikompresi pada tahap selanjutnya.

3. *Transform Bytes*

Transformasi hanya dilakukan untuk kompresi menggunakan metode BWT + LZW. Pada tahap ini aplikasi akan mentransformasi data dalam bentuk *byte* yang telah dibaca pada tahap sebelumnya dan mengubahnya menggunakan algoritma BWT. Hasil dari tahap ini tidak mengalami perubahan ukuran, atau dengan kata lain tidak terkompresi. Kompresi data terjadi pada tahap selanjutnya. Tahap ini bertujuan untuk mengurutkan *byte-byte* data agar dapat dikompresi dengan lebih baik. Untuk penjelasan lebih lanjut mengenai tahap ini, dapat dilihat pada subbab 3.5 yang membahas tentang perancangan algoritma.

4. *Compress Bytes*

Pada tahap ini, aplikasi akan mengolah *byte-byte* data dari proses sebelumnya dan *byte-byte* tersebut akan dikompresi menggunakan algoritma LZW. Hasil dari tahap ini adalah sekumpulan *byte-byte* baru yang terkompresi. Untuk penjelasan lebih lanjut mengenai tahap ini, dapat dilihat pada subbab 3.5 yang membahas tentang perancangan algoritma.

5. *Output File + Show Results*

Pada tahap ini, *byte-byte* yang dihasilkan oleh tahap sebelumnya disimpan ke dalam *file* bersama dengan data-data tambahan. Untuk metode kompresi LZW, data tambahan yang dimaksud meliputi ekstensi *file* citra digital asli dan panjang

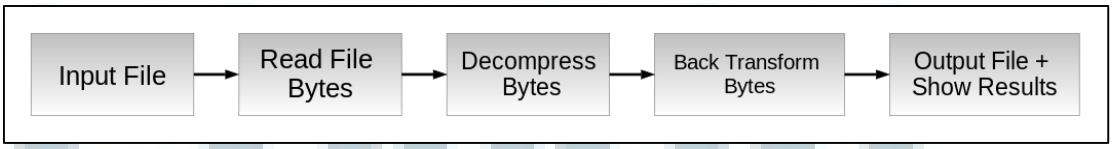
Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016

byte-byte terkompresi. Sedangkan untuk metode kompresi BWT + LZW, data tambahan tersebut meliputi ekstensi *file* citra digital asli, indeks dari tahap transformasi Burrows-Wheeler, dan panjang *byte-byte* terkompresi. Data-data tambahan ini diletakan sebelum *byte-byte* hasil kompresi. Data tersebut diperlukan untuk mengembalikan *byte-byte* hasil kompresi menjadi citra digital yang sama seperti aslinya.

File yang dihasilkan dari proses kompresi menggunakan metode LZW memiliki ekstensi lzw, sedangkan *file* yang dihasilkan dari proses kompresi menggunakan metode BWT + LZW memiliki ekstensi blzw. Di samping menyimpan *file*, hasil proses kompresi yang berupa waktu dan rasio kompresi ditampilkan.

3.3.2 Dekompresi

Serupa dengan fitur kompresi, dekompresi dilakukan dalam 5 tahap yang mencakup memberikan masukan data berupa *file*, membaca *file*, dekompresi, transformasi balik, dan menyimpan keluaran serta menampilkan hasil proses dekompresi.



Gambar 3.2 Tahapan Dekompresi Aplikasi

Tahapan-tahapan yang dilalui dalam proses dekompresi dapat dilejaskan sebagai berikut.

1. *Input File*

Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016

Tahap pertama proses dekompresi adalah *input file*. Pada tahap ini, pengguna akan diminta untuk memilih sebuah *file* khusus yang merupakan *output* dari proses kompresi untuk didekompresi. *File* khusus yang diterima oleh aplikasi adalah *file* dengan ekstensi blzw atau lzw.

2. *Read File Bytes*

Pada tahap ini, aplikasi akan membaca *byte-byte* data dari *file* yang dipilih. *Byte-byte* tersebut tidak dibaca seluruhnya sekaligus, melainkan dibaca sesuai komposisi *byte* yang terdapat dalam *file*, yaitu.

a. Ekstensi *file* citra digital asli.
b. Indeks dari tahap transformasi Burrows-Wheeler apabila *file* dengan ekstensi blzw. Komposisi ini tidak terdapat dalam *file* dengan ekstensi lzw.
c. Panjang *byte* terkompresi.
d. *Byte-byte* terkompresi sesuai panjang yang diperoleh pada poin c. *Byte-byte* inilah yang digunakan pada tahap selanjutnya.

3. *Decompress Bytes*

Pada tahap ini, *byte-byte* terkompresi akan didekompresi menggunakan algoritma LZW. Hasil dari tahap ini adalah *byte-byte* baru yang merupakan hasil dekompresi *byte-byte* terkompresi. Untuk penjelasan lebih lanjut, dapat dilihat pada subbab 3.5 yang membahas tentang perancangan algoritma.

4. *Back Transform Bytes*

Transformasi balik hanya dilakukan apabila *file* yang didekompresi memiliki ekstensi blzw. Pada tahap ini, *byte-byte* yang merupakan hasil tahap sebelumnya ditransformasi balik menggunakan algoritma BWT. Hasil dari tahap ini merupakan *byte-byte* yang membentuk *file* citra digital asli. Untuk penjelasan

Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016

lebih lanjut, dapat dilihat pada subbab 3.5 yang membahas tentang perancangan algoritma.

5. *Output File + Show Results*

Pada tahap ini, *byte-byte* yang telah diproses sebelumnya disimpan ke dalam *file*. Nama *file* yang dihasilkan disesuaikan dengan nama *file input*. Sedangkan ekstensi dari *file* yang dihasilkan disesuaikan dengan data yang diperoleh dari dalam *file input*. Selain penyimpanan *file*, hasil proses dekompresi yang berupa waktu dekompresi juga ditampilkan.

3.4 Rancangan Implementasi Algoritma

Implementasi algoritma dilakukan dengan pembelajaran mengenai cara kerja algoritma. Setelah pembelajaran selesai, langkah selanjutnya adalah meneliti cara untuk mengimplementasi algoritma tersebut ke dalam aplikasi. Dalam tahap penelitian ini, dibuat diagram *flowchart* yang menggambarkan alur kerja aplikasi dalam mengimplementasikan algoritma.

3.4.1 Diagram Alir Kompresi

Proses kompresi dimulai dengan tahap masukan dimana pengguna diminta untuk memilih *file* citra yang ingin dikompresi. Selanjutnya, aplikasi akan membaca data dari *file* citra yang dipilih. Kemudian data tersebut ditransformasi menggunakan algoritma BWT dan dikompresi menggunakan algoritma LZW. Setelah itu, data-data proses kompresi, yang berupa waktu dan rasio kompresi, ditampilkan dan data hasil kompresi disimpan ke *file*.

Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016

Gambar 3.3 Diagram Alir Kompresi

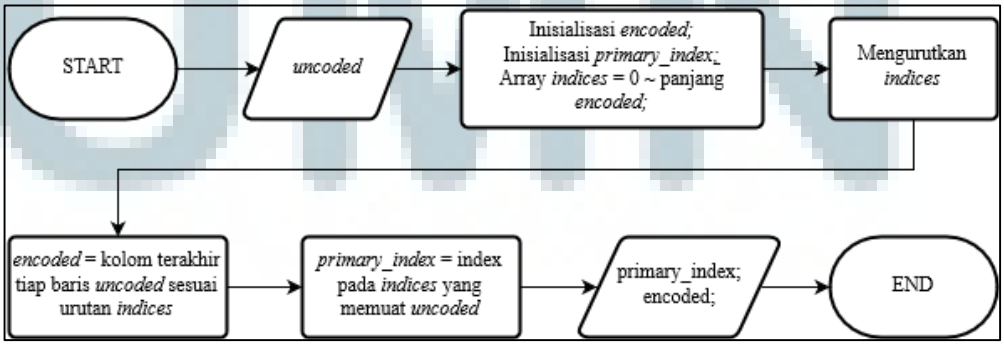
3.4.2 Diagram Alir Dekompresi

Seperti proses kompresi, proses dekompresi juga dimulai dengan tahap masukan dimana pengguna diminta untuk memilih *file* hasil kompresi untuk didekompresi. Selanjutnya, aplikasi akan membaca data dari *file* citra yang dipilih. Kemudian data tersebut didekompresi menggunakan algoritma LZW dan ditransformasi balik menggunakan algoritma BWT. Setelah itu, data proses dekompresi, yang berupa waktu dekompresi, ditampilkan dan hasil dekompresi disimpan ke *file*.

Gambar 3.4 Diagram Alir Dekompresi

3.4.3 Rancangan Implementasi Algoritma Burrows-Wheeler-Transform

Gambar 3.5 adalah diagram alir yang menunjukkan algoritma transformasi Burrows-Wheeler yang akan diimplementasikan.



Gambar 3.5 Diagram Alir Transformasi BWT

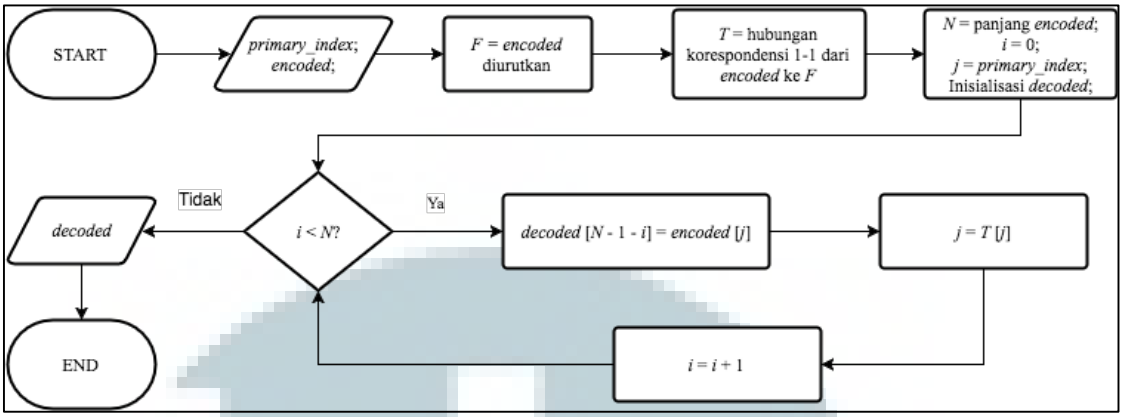
Transformasi pada metode Burrows-Wheeler yang akan diimplementasi diawali dengan tahap inisialisasi variabel-variabel. Variabel-variabel tersebut adalah variabel *encoded* dan *primary_index* yang akan dikembalikan sebagai *output*, serta variabel *indices* yang merupakan sebuah *array* yang berisi angka *index* dari 0 hingga N-1, dimana N adalah panjang *array byte* data yang akan ditransformasi.

Algoritma Burrows-Wheeler yang diimplementasi tidak mengurutkan baris-baris pada matriks yang berisi pergeseran data *input*, melainkan hanya mengurutkan *index-index* baris yang ada dengan menggunakan pembanding khusus. Pembanding ini menyimpan data *input* untuk digunakan pada proses pembandingan, tetapi tidak terjadi duplikasi data *input* menjadi matriks berukuran N x N yang tentunya memerlukan memori sebesar N x N. Hal ini dilakukan, selain untuk mempersingkat waktu pengurutan, tetapi juga untuk mengatasi keterbatasan memori yang tersedia.

Setelah *indices* dirurutkan, maka langkah selanjutnya adalah memperoleh kolom-kolom terakhir dari data *input*, sesuai urutan *indices*, dan menyimpannya ke dalam variabel *encoded*. Selanjutnya, *primary_index* diisi dengan *index* pada *indices* yang memuat data *input* asli, atau dengan kata lain memuat nilai 0. Hasil dari proses transformasi ini adalah data setelah ditransformasi, yaitu *encoded*, dan *index* yang memuat data asli, yaitu *primary_index*.

Gambar 3.6 menunjukkan diagram alir algoritma transformasi balik Burrows-Wheeler yang akan diimplementasikan.

Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016



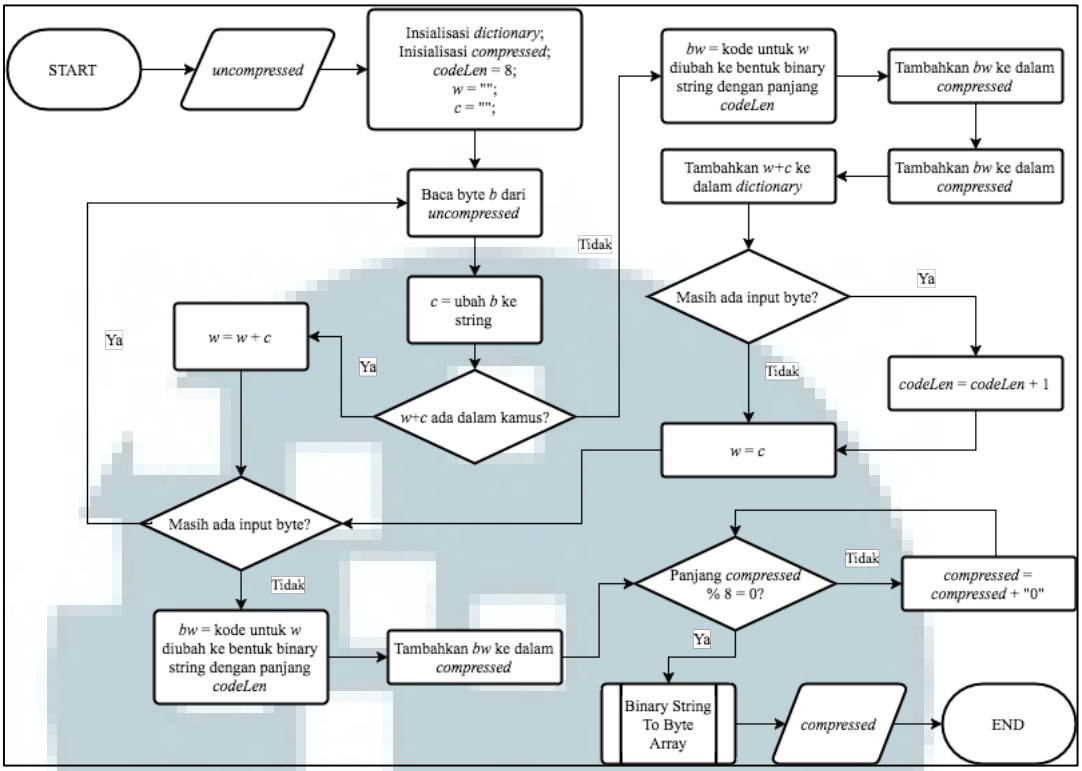
Gambar 3.6 Diagram Alir Transformasi Balik BWT

Proses transformasi balik pada algoritma Burrows-Wheeler diawali dengan membuat *array* F yang merupakan kolom pertama dari matriks pada proses transformasi Burrows-Wheeler. F dapat diperoleh dengan mengurutkan data *encoded*. Setelah mendapatkan F, langkah selanjutnya adalah membangun *array* T yang merupakan hubungan korespondensi satu-satu dari *encoded* ke F. Setelah memperoleh T, data asli dapat dibuat kembali. Proses pembentukan *array decoded* dilakukan dalam sebuah pengulangan, yang dimulai dari *index* pada *primary_index*, berdasarkan data *encoded*. Setiap pengulangan, *index* selanjutnya diperoleh dengan menggunakan *array* T. Pengulangan dilakukan sebanyak N kali, dimana N adalah panjang data *encoded*.

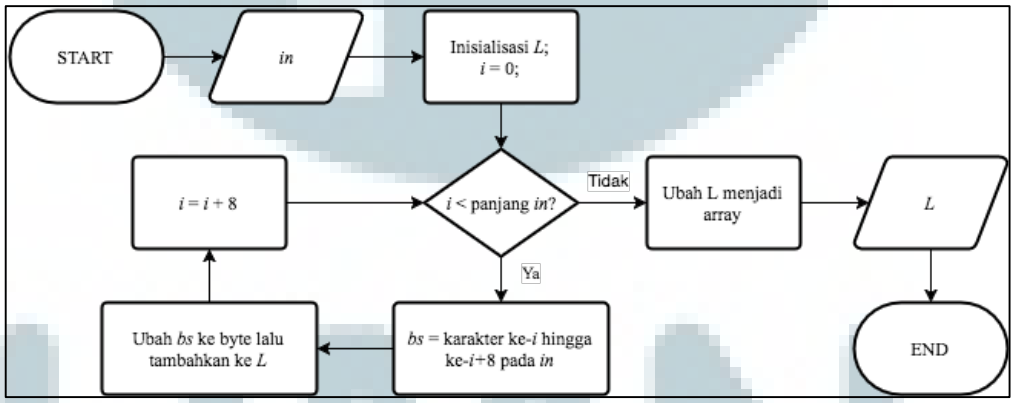
3.4.4 Rancangan Implementasi Algoritma Lempel-Ziv-Welch

Gambar 3.7 adalah diagram alir yang menunjukkan algoritma kompresi Lempel-Ziv-Welch yang akan diimplementasi.

Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016



Gambar 3.7 Diagram Alir Kompresi LZW



Gambar 3.8 Diagram Alir *Binary String To Byte Array*

Tahap kompresi diawali dengan inisialisasi kamus dan variabel. Selanjutnya, setiap *byte* dalam *byte array* pada *uncompressed* akan diproses satu persatu dalam sebuah pengulangan. *Byte* tersebut akan diubah menjadi *string* dengan menggunakan *encoding* Base64. Kemudian akan diperiksa apakah gabungan

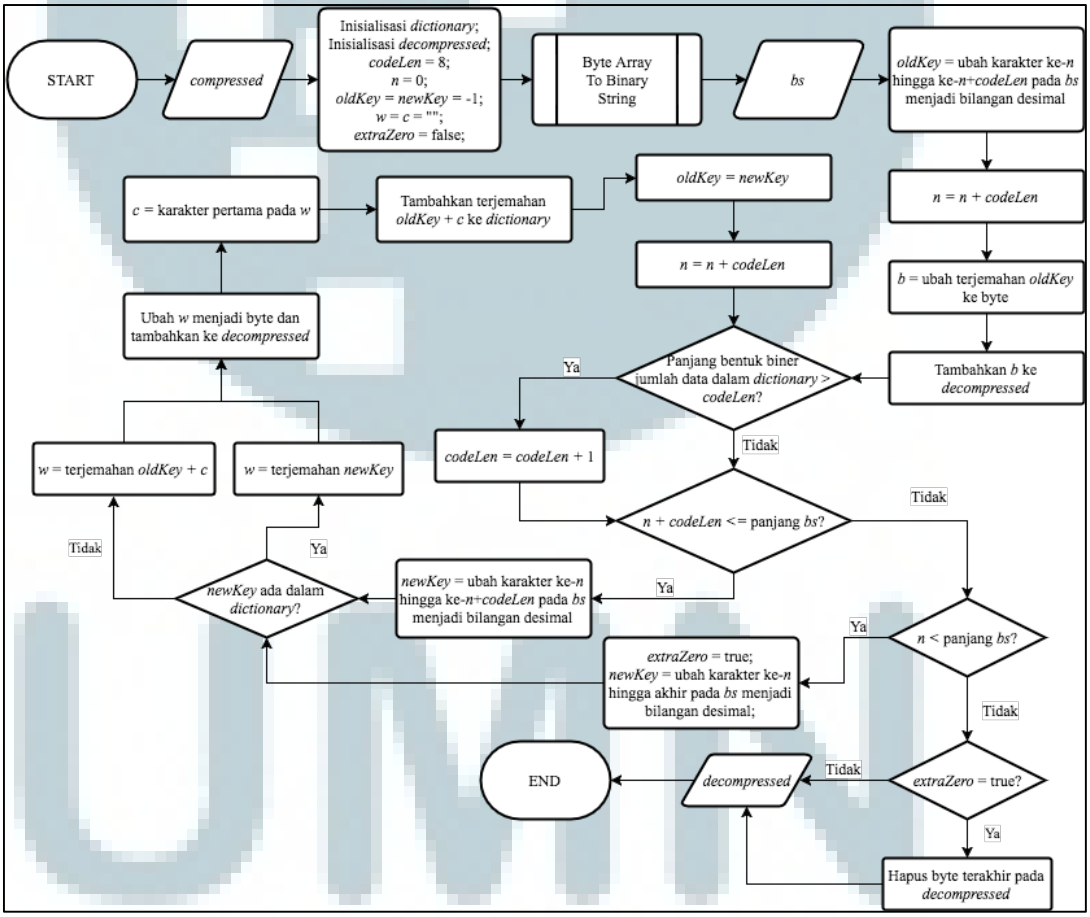
string lama dengan *string* baru terdapat dalam kamus. Jika ya, maka akan dilanjutkan ke proses *byte* selanjutnya. Apabila tidak, maka *string* lama akan dikonversi ke bilangan biner dengan panjang sesuai dengan *codeLen* dan ditambahkan ke dalam *compressed* dan gabungan *string* lama dengan *string* baru ditambahkan ke dalam kamus. Setelah penambahan data dalam kamus, dilakukan pemeriksaan panjang kamus. Apabila panjang kamus dalam bentuk biner melebihi *codeLen*, maka *codeLen* akan ditambah dengan 1. Lalu *string* lama diisi dengan *string* baru. Proses ini akan diulang hingga *byte* terakhir dalam *byte array uncompressed*. Setelah *byte* terakhir diproses, masih dilakukan penambahan ke dalam *compressed*, yaitu *string* lama dikonversi ke bilangan biner dengan panjang sesuai *codeLen*.

Sebelum tahap kompresi berakhir, akan dilakukan pemeriksaan terhadap panjang data *compressed*. Selama panjang data *compressed* tidak habis dibagi 8, maka akan ditambahkan karakter '0' ke dalam *compressed*. Penambahan karakter '0' ini dilakukan untuk mencegah terjadinya kerusakan data pada tahap selanjutnya dimana *binary string* akan diubah menjadi *byte array*. Sebagai contoh, apabila terdapat urutan *binary string* seperti 00110100101 dengan panjang 11 bit, bagian terakhir akan tersisa menjadi 3 bit (00110100 | 101) dan ketika diubah ke bilangan biner, maka akan dianggap sebagai 00000101 dan mengubah data asli ketika dibaca kembali (00110100 | 00000101). Dengan adanya penambahan '0', maka urutan *binary string* akan menjadi 0011010010100000 (00110100 | 10100000). Kelebihan *byte* karena adanya tambahan 0 tidak akan mengakibatkan kesalahan karena dapat dihapus pada tahap dekompresi.

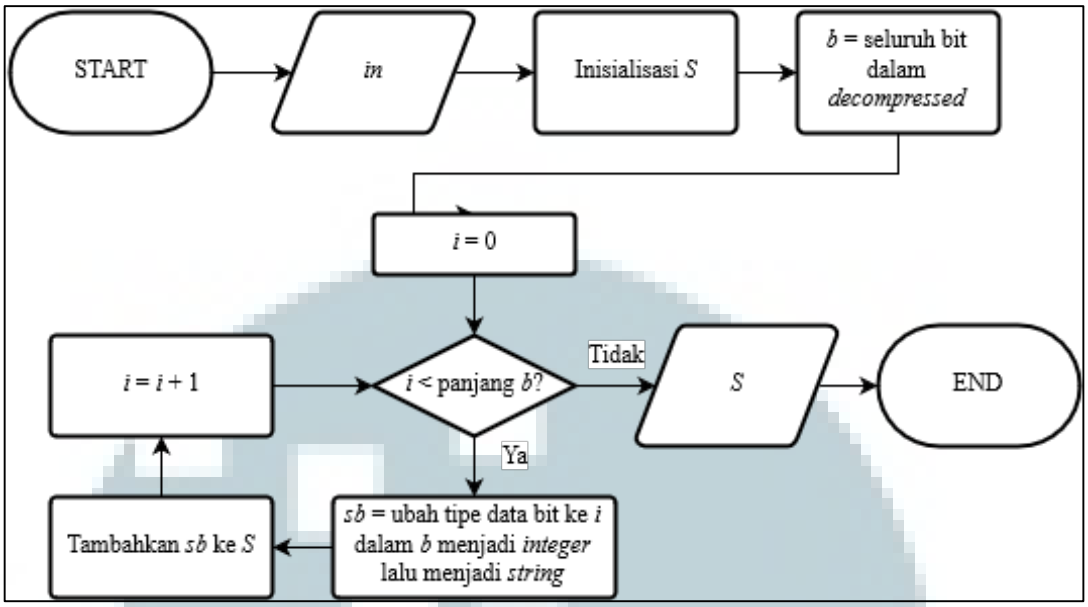
Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016

Panjang data *compressed* yang tidak habis dibagi 8 sendiri diakibatkan karena *codeLen* yang berubah seiring dengan berjalannya proses kompresi. *CodeLen* memegang peran yang cukup penting dalam proses kompresi karena panjang bit yang ditulis sebagai *output* dapat diperpendek dengan *codeLen*. Tanpa *codeLen*, *byte-byte* yang ditulis sebagai *output* harus disamakan panjangannya agar tidak merusak data, panjang bit yang digunakan harus mengikuti panjang bit terbesar.

Gambar 3.9 menunjukkan algoritma dekompresi Lempel-Ziv-Welch yang akan diimplementasi.



Gambar 3.9 Diagram Alir Dekompresi LZW



Gambar 3.10 Diagram Alir *Byte Array To Binary String*

Seperti pada tahap kompresi, tahap dekompresi dimulai dengan inisialisasi kamus dan variabel. Kemudian data yang dalam bentuk *byte array* diubah menjadi *binary string* untuk diproses. Selanjutnya, sebelum memasuki tahap pengulangan, dilakukan proses pertama kali untuk mendapatkan nilai *oldKey*, yaitu membaca karakter biner dari posisi n sebanyak *codeLen* karakter pada *string* dan mengubahnya menjadi desimal, lalu menambahkan n dengan *codeLen*, setelah itu menambahkan terjemahan *oldKey* dalam kamus ke dalam *decompressed*.

Tahap pengulangan diawali dengan pemeriksaan terhadap panjang kamus. Apabila panjang kamus dalam bentuk biner melebihi *codeLen*, maka *codeLen* akan ditambah dengan 1. Setelah itu akan diperiksa apakah n + *codeLen* tidak melebihi panjang *string*. Apabila ya, maka nilai *newKey* akan diisi dengan bentuk desimal dari katakter biner pada *string* dari posisi n sebanyak *codeLen* karakter. Apabila tidak, maka akan diperiksa apakah n tidak melebihi panjang *string*. Apabila ya, maka nilai *newKey* akan diisi dengan bentuk desimal dari karakter

Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016

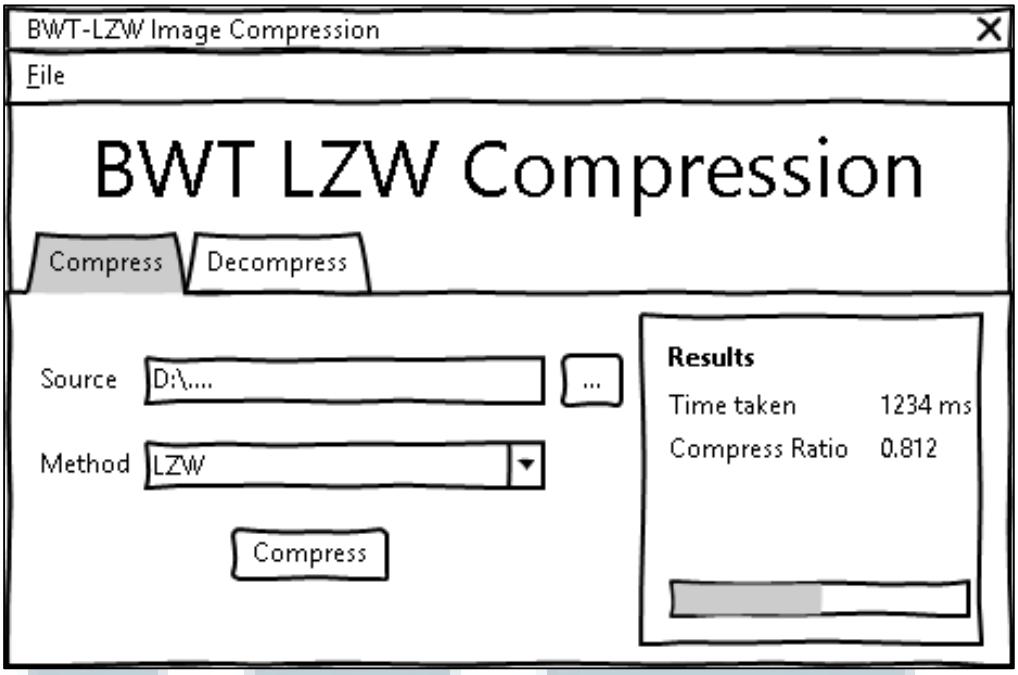
pada *string* dari posisi n sampai habis dan *extraZero* diisi dengan nilai *true*. Apabila tidak, maka pengulangan akan diakhiri. Setelah mendapatkan nilai *newKey*, akan diperiksa apakah *newKey* terdapat dalam kamus. Jika ya, maka w akan diisi dengan terjemahan *newKey*. Jika tidak, maka w akan diisi dengan terjemahan *oldKey* dan digabungkan dengan karakter c. Selanjutnya, w akan diubah menjadi *byte* dan ditambahkan ke dalam *decompressed*. Kemudian karakter c diisi dengan karakter pertama dalam *string* w. Tambahkan terjemahan *oldKey* digabung dengan c ke dalam kamus. Selain itu, n ditambahkan dengan *codeLen* untuk pembacaan selanjutnya, *oldKey* diisi dengan *newKey* dan kembali ke awal pengulangan.

Setelah pengulangan selesai, akan diperiksa apakah *extraZero* bernilai *true*. Apabila ya, maka *byte* terakhir pada *decompressed* akan dihapus seperti yang telah dijelaskan pada tahap kompresi.

3.5 Sketsa Tampilan Aplikasi

Aplikasi dibuat memliki tampilan berupa antarmuka grafis atau *Graphical User Interface* (GUI). GUI ditujukan untuk mempermudah pengguna untuk menggunakan aplikasi. Berikut adalah sketsa tampilan aplikasi yang akan diimplementasi.

Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016



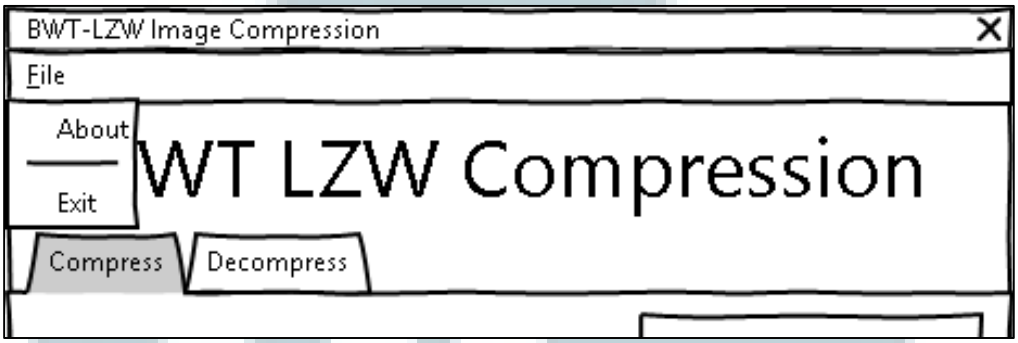
Gambar 3.11 Sketsa Tampilan Utama

Gambar 3.11 menunjukkan sketsa tampilan utama aplikasi. Pada halaman ini, terdapat tab untuk kompresi dan dekompresi, sebuah *text box* dengan tombol untuk memilih *file* citra digital, sebuah *drop down* untuk memilih metode kompresi yang digunakan, dan tombol untuk memulai kompresi atau dekompresi. Pengguna harus memilih citra untuk dikompresi dan metode kompresi yang dipilih sebelum dapat memulai kompresi. Pada dekompresi, pengguna hanya diwajibkan untuk memilih *file* untuk didekompresi dan pengguna tidak dapat memilih metode dekompresi melainkan ditentukan oleh aplikasi berdasarkan ekstensi *file* yang dipilih.

Informasi *file* citra yang dipilih oleh pengguna akan digunakan untuk mengisi *text box* yang ada dengan *path* menuju *file* citra tersebut. Kemudian pengguna dapat memilih metode yang ingin digunakan untuk melakukan kompresi pada *drop down* yang berisi LZW dan BWT + LZW. Setelah memilih metode

Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016

kompresi, pengguna dapat menekan tombol Compress untuk memulai kompresi. Ketika proses kompresi selesai, hasil kompresi yang berupa waktu yang dibutuhkan serta rasio kompresi akan ditampilkan pada panel di sebelah kanan yang bertuliskan Results.



Gambar 3.12 Sketsa Tampilan Menu *File*

Gambar 3.12 menunjukkan menu *drop down* yang berisi sub-menu *About* untuk menampilkan informasi singkat mengenai aplikasi dan pembuat aplikasi dan sub-menu *Exit* untuk keluat dari aplikasi.

Gambar 3.13 Sketsa Tampilan *About*

Gambar 3.13 menunjukkan tampilan ketika sub-menu *About* dipilih. Di dalamnya akan dijabarkan secara singkat mengenai aplikasi, cara penggunaan, serta pembuat aplikasi.

3.6 Rancangan Skenario Pengujian

Dalam pengujian terhadap implementasi algoritma Burrows-Wheeler dan Lempel-Ziv-Welch untuk kompresi citra ini, akan digunakan 6 citra digital yang diasumsikan dapat mewakili citra digital lainnya. Pengujian akan dilakukan terhadap seluruh citra digital dengan format BMP, GIF, JPG, PNG, dan TIFF serta resolusi 256 x 256 *pixel* dan 512 x 512 *pixel*. Pengujian akan dilakukan dengan dua metode berbeda, yaitu pertama hanya menggunakan algoritma LZW dan kedua menggunakan algoritma BWT dan LZW. Pengujian dilakukan bertujuan untuk mengetahui dampak dari algoritma BWT terhadap algoritma LZW dalam melakukan kompresi terhadap citra digital, dilihat dari segi waktu kompresi dan dekompresi serta rasio kompresi.

Di samping itu, akan dilakukan pengujian menggunakan aplikasi kompresi yang sudah ada, yaitu 7zip. Pengujian ini ditujukan untuk menjadi validasi terhadap implementasi yang dibuat dengan algoritma pada aplikasi kompresi lainnya.

Dari uji coba yang dilakukan akan diperoleh data yang berupa rasio kompresi, waktu kompresi dan waktu dekompresi yang diperoleh. Dari data-data hasil pengujian tersebut dapat disimpulkan bagaimana kinerja dari implementasi algoritma BWT pada algoritma LZW untuk melakukan kompresi pada citra digital, serta kinerjanya dibandingkan dengan aplikasi kompresi lain.

Implementasi algoritma..., Daniel Sukmana, FTI UMN, 2016