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

ANALISIS DAN PERANCANGAN

3.1 Metode Penelitian

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

a. Studi Literatur

Melakukan studi mengenai teori-teori dan konsep yang berkaitan dengan pokok bahasan penelitian, seperti teori mengenai metode Deflate, LZ77, Huffman Code, SDK Dropbox untuk windows, dan berbagai konsep pendukung lainnya. Referensi-referensi yang digunakan dapat berupa buku, jurnal ilmiah, artikel, dokumentasi SDK, dan lain-lain.

b. Perancangan Aplikasi

Melakukan perancangan awal terhadap aplikasi yang dibangun, meliputi analisis hasil studi literatur dan sistem yang dibuat, serta perancangan alur aplikasi dalam bentuk *system flow* dan rancangan antarmuka.

c. Implementasi

Melakukan pembangunan terhadap aplikasi dengan mengimplementasikan rancangan yang telah didefinisikan sebelumnya dengan menggunakan bahasa pemrograman yang telah ditentukan.

Implementasi Algoritma ..., Agi Reza Jasuma S, FTI UMN, 2013

d. Uji Coba dan Analisa Aplikasi

Melakukan uji coba terhadap aplikasi disertai dengan evaluasi hasil yang didapatkan.

3.2 Analisis

Dalam penelitian yang dilakukan, algoritma yang digunakan adalah algoritma Deflate. Pada algoritma tersebut, algoritma Huffman Code diganti dengan algoritma length limited Huffman Code. Aplikasi ini dibangun dengan bahasa pemrograman C#, dimana *framework* aplikasi ini adalah .NET *framework* 4. Untuk menghubungkan aplikasi ini dengan Dropbox, digunakan library SharpBox 1.2 yang dibangun oleh Codeplex. Aplikasi yang dibangun merupakan aplikasi desktop untuk windows. Cara kerja dari aplikasi tersebut mengikuti aplikasi Dropbox pada windows. Hanya saja *file* yang diunggah adalah *file* hasil kompresi. Dengan cara ini, *file* asli tetap berada pada *folder* dan yang tersimpan pada server Dropbox adalah hasil kompresi. Sedangkan, ketika *file* diunduh dari server Dropbox, *file* didekompresi terlebih dahulu. Dengan mengikuti cara kerja pada aplikasi Dropbox, pengguna aplikasi tidak mengalami kesulitan dalam menggunakan. Aplikasi ini diberi nama CompressBox.

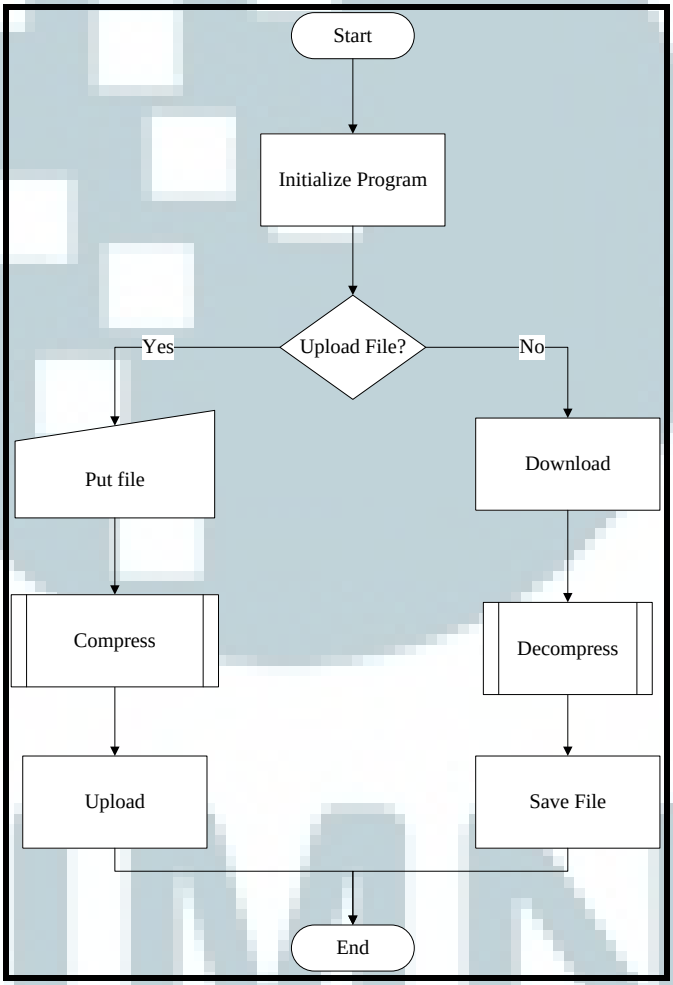
3.3 Perancangan

Pada subbab ini dipaparkan hasil rancangan untuk aplikasi CompressBox ini.

3.3.1 Diagram Sistem

Proses kerja aplikasi yang dibuat memiliki *system flow* seperti gambar dibawah ini. Aplikasi ketika dijalankan selalu mengecek apakah *file* diunggah atau

diunduh. Pada saat terdapat file yang diunggah, maka aplikasi mengkompres terlebih dahulu kemudian diunggah. Apabila terdapat *file* yang diunduh, *file* tersebut didekompresi terlebih dahulu, setelah itu diletakkan di *folder* CompressBox. Aplikasi ini menggunakan modul Deflate. Dalam modul daflate terdapat modul LZ77 dan modul Huffman..



Gambar 3.1 Diagram System Flow Aplikasi

3.3.2 Fungsionalitas Sistem

Sistem ini memiliki beberapa fungsionalitas, sebagai berikut:

1. Menerima *input* dalam bentuk *array of byte* dari sebuah file yang akan dikompres.
2. Melakukan pengurangan jumlah pattern yang sama pada input dengan menggunakan algoritma LZ77.
3. Melakukan penentuan *prefix* pada setiap *literal* atau *length* dengan menggunakan algoritma Huffman Code.
4. Melakukan pemetaan *literal*, *length* dan *distance* terhadap *prefix*.
5. Melakukan penyusunan *bit* untuk *output* berdasarkan hasil dari algoritma LZ77.
6. Melakukan pencarian *prefix* dalam *array of byte* pada saat dekompresi.

3.3.3 Masukan dan Keluaran Sistem

Masukkan yang diperlukan oleh sistem adalah sebagai berikut:

1. Masukan pada sistem secara umum adalah *file* yang akan dikompresi atau didekompresi.
2. Modul Deflate menerima alamat lokasi dari *file* yang dikompresi atau didekompresi serta status kompresi *file*.
3. Masukan pada modul LZ77 adalah *array of Byte* dari *file* yang dikompresi atau didekompresi.
4. Masukan pada modul Huffman adalah *List literal* dan *length* dari proses LZ77.

Keluaran yang dihasilkan oleh sistem adalah sebagai berikut:

1. Keluaran sistem secara umum adalah *file* hasil proses kompresi atau dekompresi.
2. Keluaran modul Deflate adalah *array of byte* hasil proses kompresi atau dekompresi.
3. Keluaran modul LZ77 terdiri dari dua jenis keluaran yaitu untuk proses kompresi berupa *array of byte* hasil pemetaan *literal*, *length*, *distance* dalam bentuk *byte*. Dan untuk proses dekompresi berupa *array of byte* hasil pemetaan *prefix* ke *character* dari setiap *byte* yang dibaca.
4. Keluaran modul Huffman adalah *array of byte* yang berisi panjang *bit* masing-masing *literal* dan *length.*

3.3.4 Desain Modul

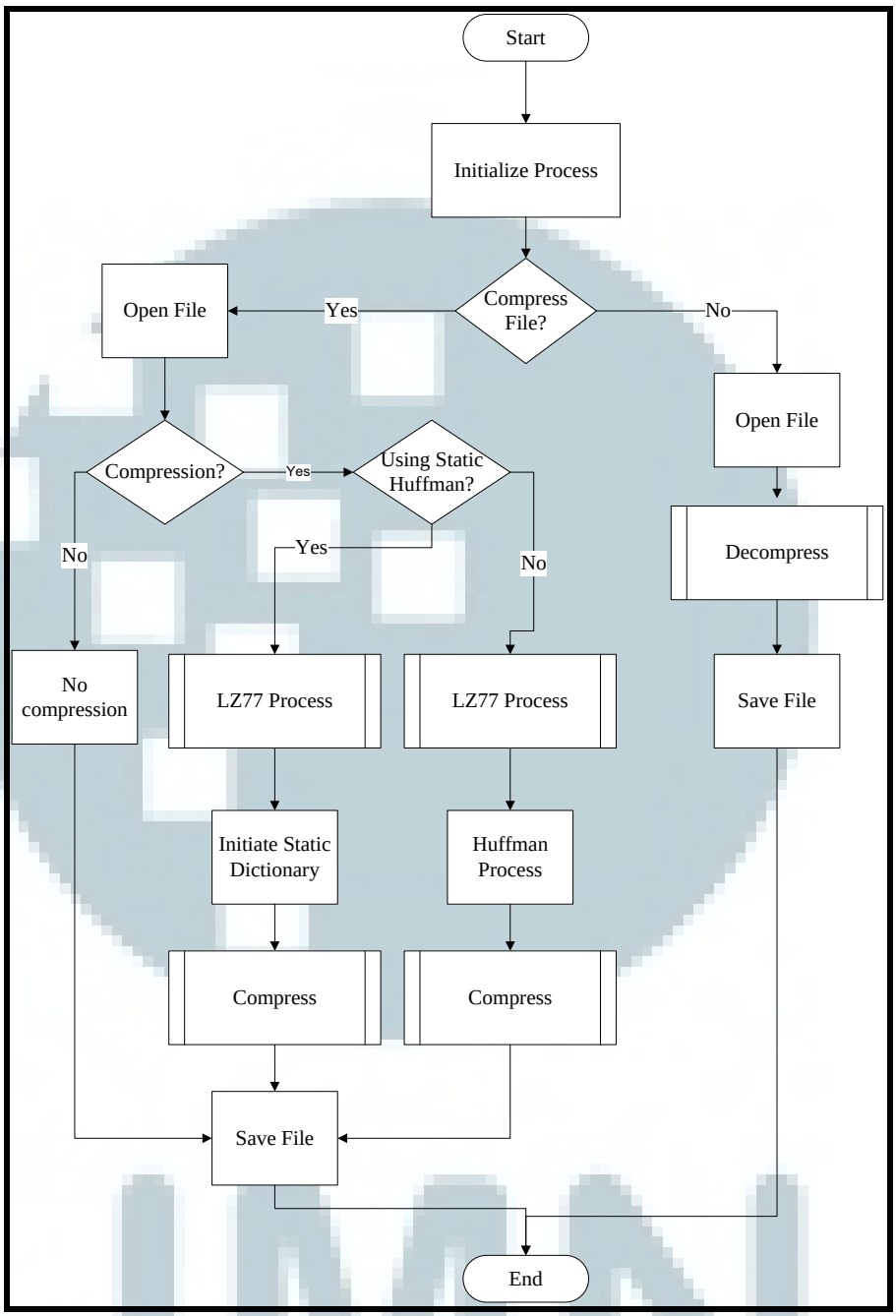
Aplikasi ini menggunakan modul deflate untuk melakukan kompresi dan dekompresi. Pada modul deflate, terdapat modul LZ77 dan Huffman untuk membantu proses kompresi maupun dekompresi. Berikut adalah penjelasan rancangan dari masing-masing modul yang digambarkan dengan diagram alir, hirarki.

1. Modul Deflate

Modul ini memiliki dua proses, proses pertama yaitu kompresi dan kedua yaitu dekompresi. Pada kedua proses ini modul deflate menggunakan modul LZ77 dan modul Huffman untuk melakukan kompresi ataupun dekompresi. Modul LZ77

digunakan untuk melakukan proses yang menggunakan algoritma LZ77. Modul Huffman digunakan untuk melakukan proses yang menggunakan algoritma Huffman code.

Pada proses kompresi, hal pertama yang dilakukan adalah membaca *byte* yang terdapat pada *file*. Hasil pembacaan ini akan disimpan dalam *array of byte*. Selanjutnya, sesuai dengan status yang digunakan, kompresi akan dilakukan. Terdapat tiga status untuk melakukan kompresi. Pada status pertama, *file* tidak akan dikompresi sama sekali. Pada status kedua, *file* dikompresi menggunakan Fixed Huffman code. Pada status ketiga *file* dikompresi menggunakan Dynamic Huffman code. Jika status yang digunakan adalah dua atau tiga, selanjutnya fungsi Lz77Process dipanggil untuk melakukan penghilangan *pattern* yang sama pada input menggunakan algoritma LZ77. Dari fungsi yang telah dipanggil, didapatkan *list* berupa pasangan jarak, panjang dan *character* dalam bentuk *byte*.



Gambar 3.2 *System flow* Modul Deflate

Apabila status dua digunakan, panjang bit sudah ditentukan oleh sistem sehingga tidak diperlukan pencarian *prefix* menggunakan Huffman code.

Berikutnya proses yang pertama dilakukan adalah melakukan inisialisasi panjang *bit* setiap *prefix*. Setelah proses inisialisasi panjang bit, proses pembuatan *file* kompresi dilakukan. Pembuatan kompresi *file* menggunakan fungsi compress dari modul LZ77, *output* dari proses tersebut adalah *list of byte* yang merupakan hasil kompresi *file*.

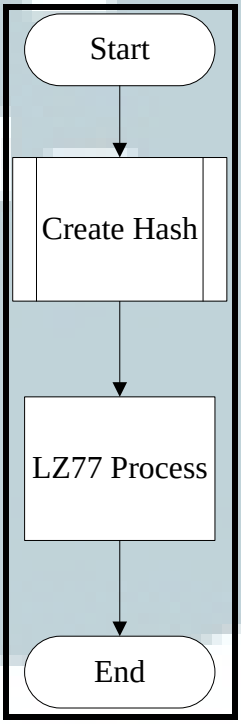
Jika status tiga digunakan, maka panjang *bit* untuk setiap *prefix* akan dicari menggunakan Huffman code. Setelah proses Huffman code, pembuatan *file* kompresi dilakukan. Kompresi sendiri memanggil fungsi compress pada modul LZ77. Dihasilkan *Output* berupa *list of byte* yang merupakan hasil kompresi.

Selanjutnya output dari proses tersebut digabungkan dengan *array of byte* dari panjang *bit*. Kedua output ini digabungkan dengan format *array of byte* panjang *bit* berada didepan, diikuti *array of byte* hasil proses kompresi dibelakang. Setelah penggabungan dilakukan, *array of byte* yang dihasilkan dapat disimpan dalam bentuk *file*.

Pada proses dekompresi, hal yang dilakukan adalah memanggil fungsi decompress pada modul LZ77. *Array of byte* dihasilkan dari fungsi ini. *Array of byte* tersebut kemudian disimpan dalam bentuk *file*. *File* ini merupakan *file* yang sama persis seperti pada saat sebelum dikompresi.

2. Modul LZ77

Modul LZ77 digunakan untuk melakukan proses yang berhubungan dengan algoritma LZ77. Beberapa fungsi pada modul ini dipanggil dalam modul Deflate. Contohnya pada saat proses kompresi, ketika status kompresi adalah dua atau tiga, fungsi Lz77Process dan compress akan dipanggil.

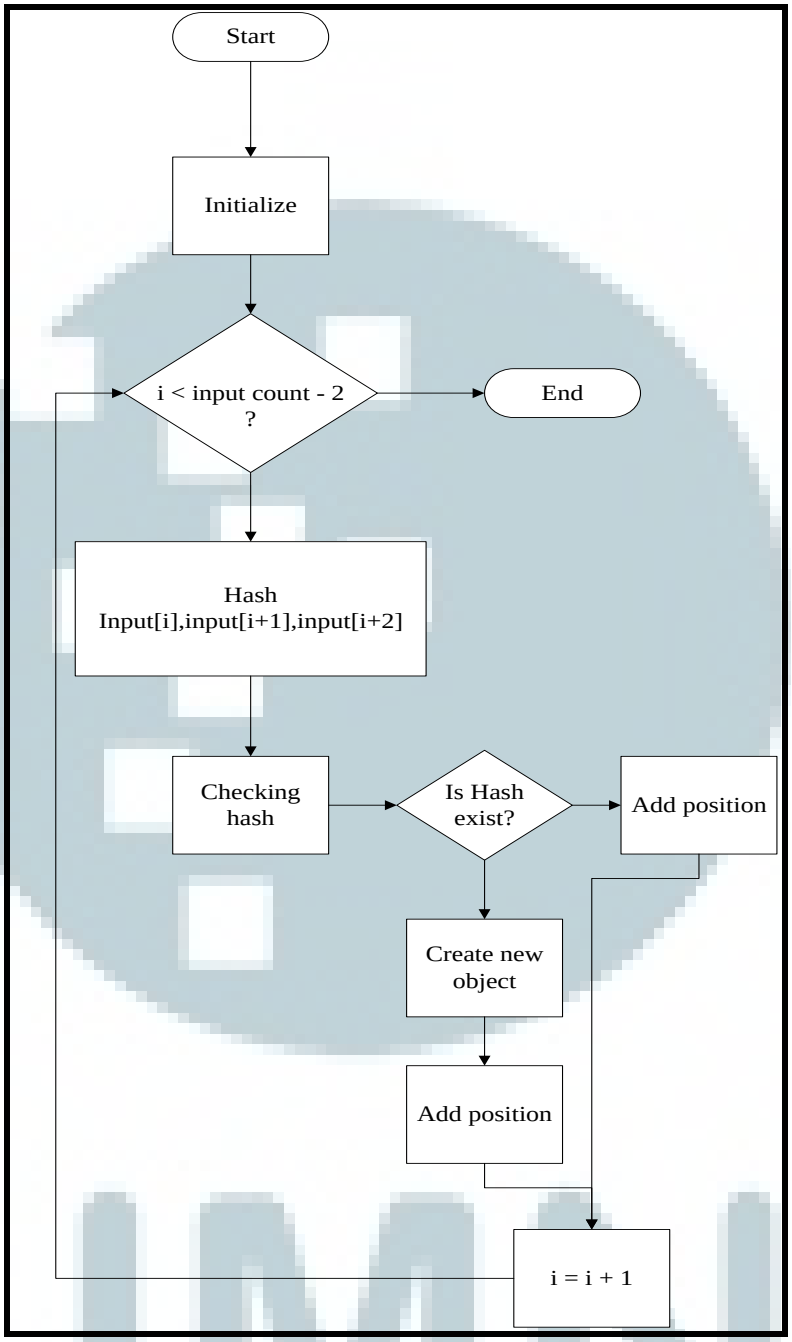


Gambar 3.3 *System flow* dari fungsi Lz77Proces

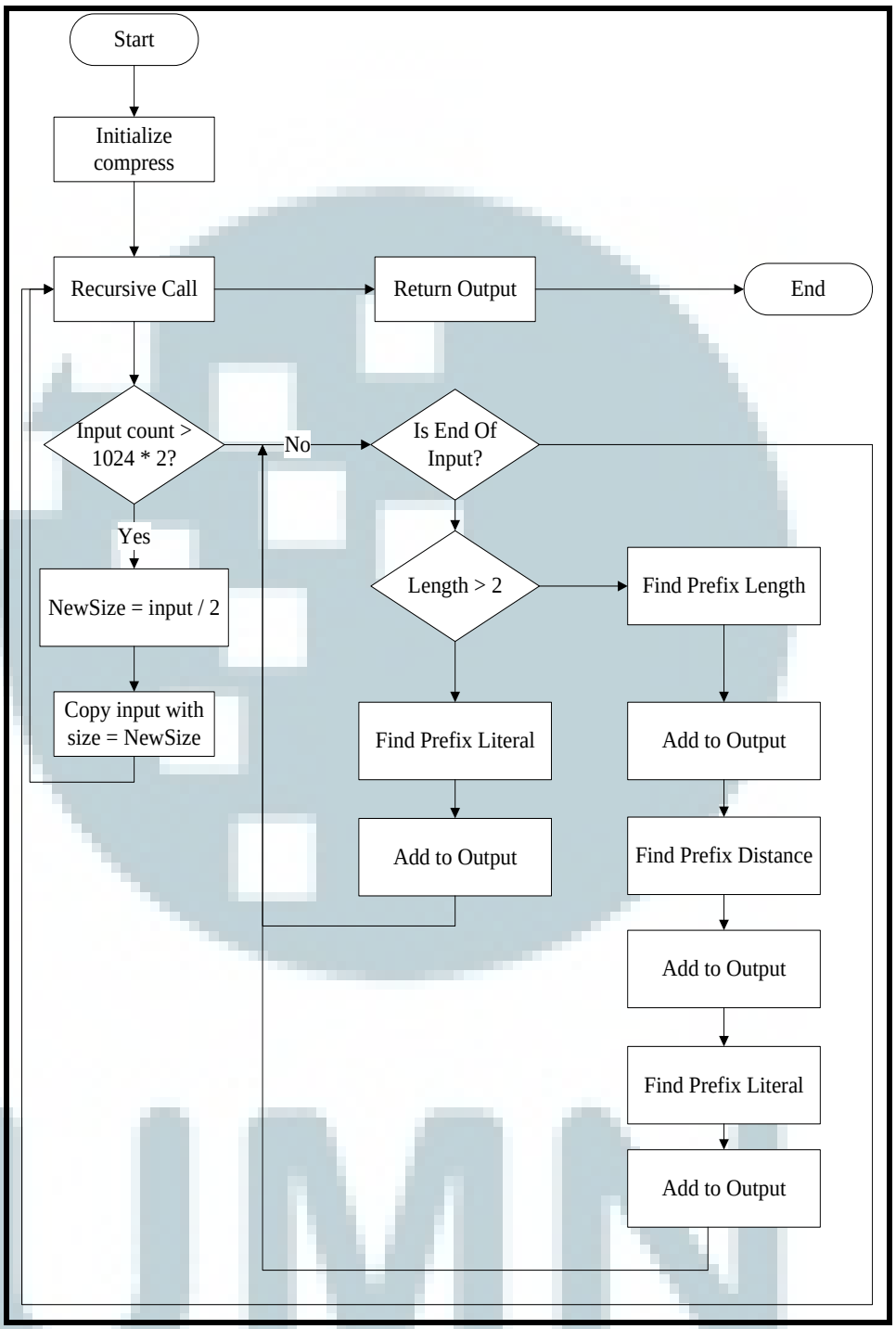
Fungsi LZ77Process dipanggil ketika sebuah proses ditujukan untuk mengurangi jumlah *byte* berdasarkan kesamaan *pattern.* Hal pertama yang dilakukan adalah menjalankan fungsi create hash. Pada proses ini, *pattern* dan posisi *pattern* dari input akan disimpan. *Pattern* disini merupakan penggabungan tiga *character*. Sebagai contoh apabila input berupa “abrac”, maka *pattern* yang

akan disimpan adalah “abr” dengan posisi lima, “bra” dengan posisi empat, “rac” dengan posisi tiga. Posisi yang dimiliki sebuah *pattern* bisa lebih dari satu, maka posisi ini disimpan kedalam bentuk *list*. Tujuannya adalah untuk mempercepat proses pencarian *pattern* pada algoritma LZ77. Flowchart fungsi ini dapat dilihat pada gambar 3.4. Setelah melakukan pemetaan berikutnya adalah pengurangan jumlah *byte* dengan algoritma LZ77. Proses pengurangan akan mengembalikan hasil *list* berupa pasangan jarak, panjang dan *character*.

Ketika proses ditujukan untuk membuat *file* hasil kompresi maka fungsi compress akan dipanggil. Fungsi tersebut diawali dengan proses inisialisasi *dictionary*. *Dictionary* memiliki *properties edoc*, *prefix*, *length*, *distance* dan *character*. *Prefix* pada *dictionary* dibuat dengan konsep Huffman canonical. Setelah proses inisialisasi dilakukan, berikutnya mengubah isi dari list yang dihasilkan fungsi LZ77Process menjadi *array of byte*. Pada proses ini dilakukan iterasi satu per satu pada setiap isi list. Setiap *character*, *distance* atau *length* yang ditemukan, akan dicari *prefix*nya pada *dictionary*. Apabila *length* yang ditemukan berupa nol, maka yang dicari hanya *prefix character*nya saja. *Prefix* ini kemudian dimasukkan kedalam *array of byte* yang merupakan keluaran dari fungsi ini. Ketika iterasi telah selesai dilakukan, maka ditambahkan *prefix* pada *edoc* 256. *Prefix* ini digunakan untuk memberi tanda akhir dari blok yang dikompresi.

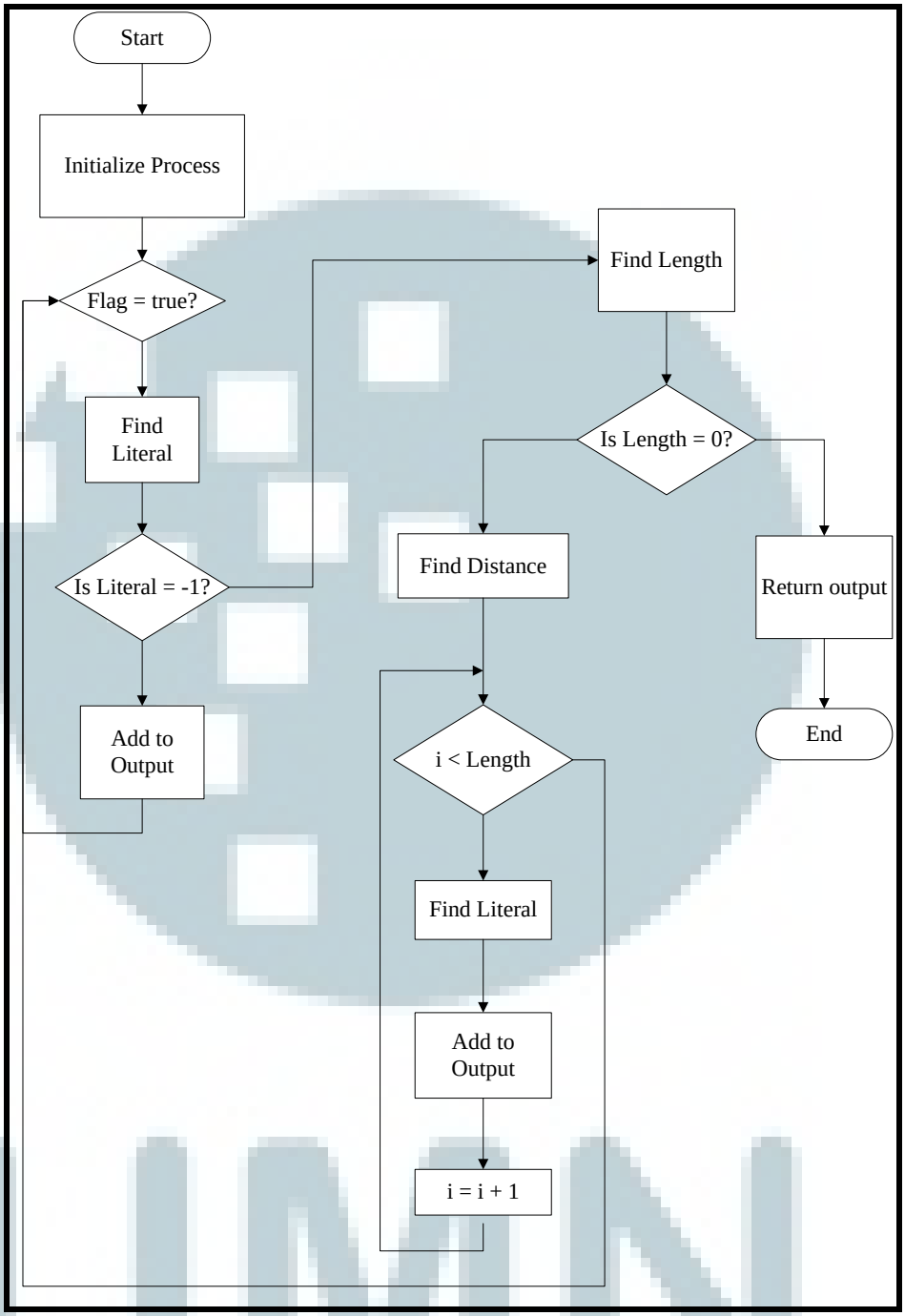


Gambar 3.4 Flowchart fungsi Lz77 – create hash



Gambar 3.5 *Flowchart* fungsi Lz77 – compress

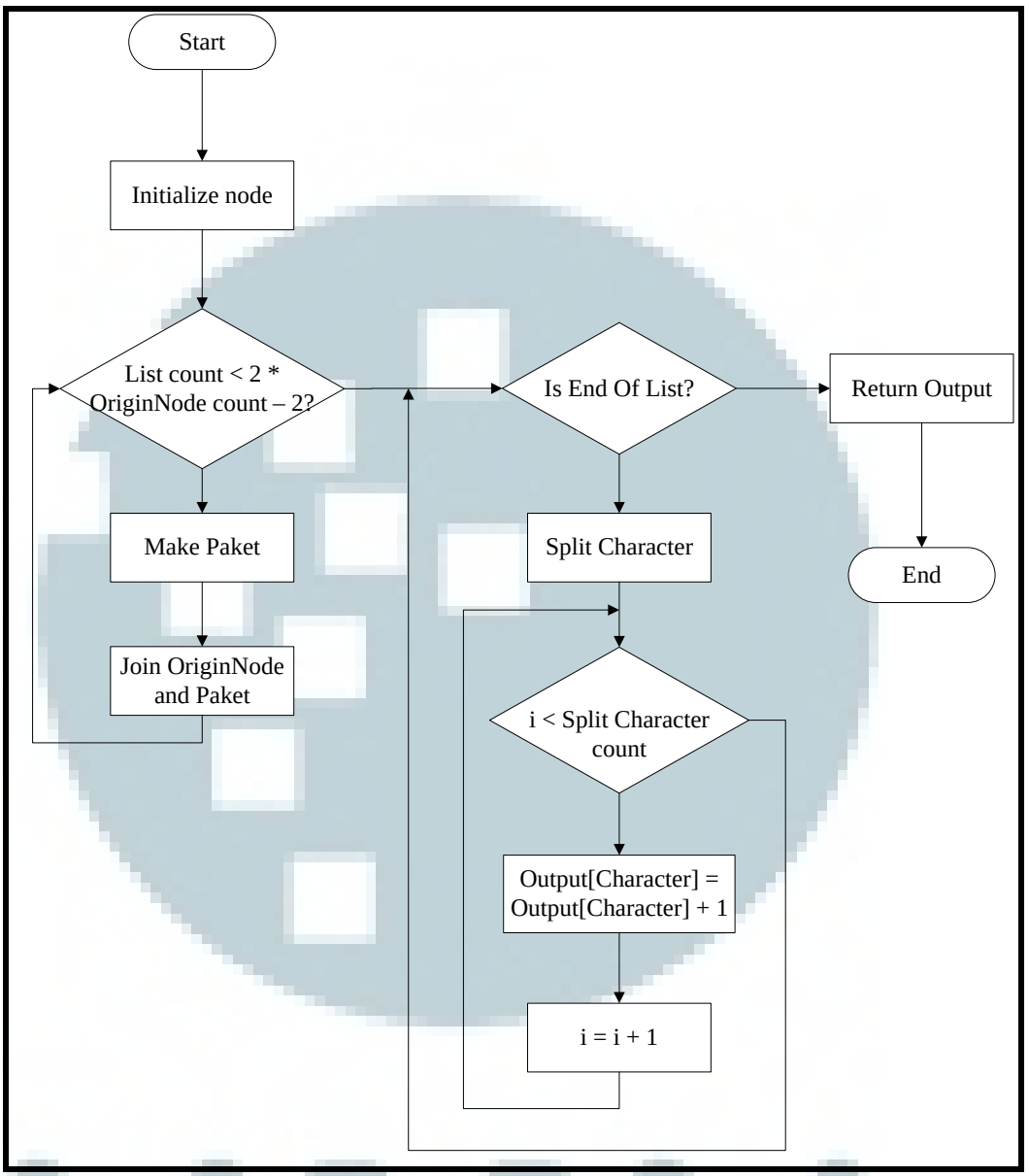
Sedangkan fungsi decompress akan dipanggil ketika sebuah proses ditujukan untuk membuat *file* hasil dekompresi. Sama seperti fungsi compress, fungsi ini diawali dengan proses inisialisasi *dictionary*. Setelah proses inisialisasi *dictionary*, iterasi untuk membaca *byte* pada *file* akan dilakukan. Iterasi ini akan berhenti apabila ditemukan prefix yang menunjukkan *end of block*. Setiap *bit* yang dibutuhkan akan diambil dari setiap *byte* yang dibaca. Proses dekompresi *file* ini dimulai dari mencari *prefix literal* terlebih dahulu. Apabila *prefix literal* ditemukan, maka akan dicari *literal* yang sesuai. Setelah *literal* ditemukan kemudian dimasukkan ke dalam *output*. Sedangkan apabila *prefix literal* tidak ditemukan, maka *prefix length* dicari. Setelah *prefix length* ditemukan, akan dicari *length* yang sesuai. Tahap selanjutnya adalah mencari *prefix distance* untuk mendapatkan *distance* yang sesuai. Kemudian *literal* akan ditambahkan sebanyak *length* dengan *distance* yang telah ditemukan kedalam *output*. Proses ini kembali mencari *prefix literal* seperti pertama kali dan diulang hingga *prefix end of block* didapatkan.



Gambar 3.6 *Flowchart* fungsi Lz77 – decompress

3. Modul Huffman

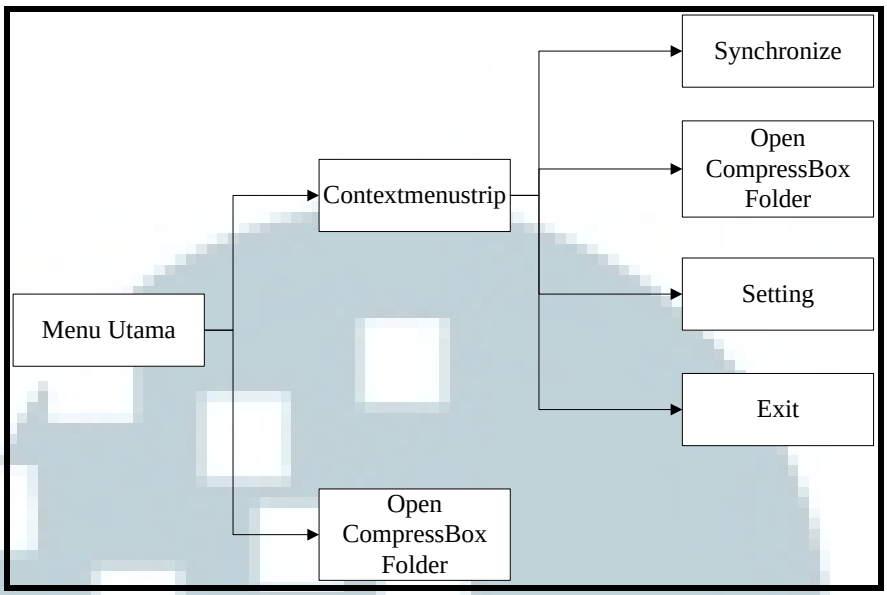
Modul Huffman digunakan untuk melakukan proses yang berhubungan dengan algoritma Huffman. Fungsi HuffmanCode pada modul ini dipanggil ketika status kompresi adalah tiga. Fungsi HuffmanCode digunakan untuk mendapatkan panjang *bit*. Panjang *bit* dibutuhkan untuk menentukan *prefix* dari setiap *character* atau *length*. Algoritma yang digunakan adalah limited length Huffman. Setelah ditemukannya panjang bit, fungsi ini akan mengembalikan nilai dari panjang bit yang sudah didapatkan dalam bentuk *array*.



Gambar 3.7 *Flowchart* fungsi Huffman

3.3.5 Hirarki Menu

Pada gambar 3.8 diperlihatkan rancangan hirarki menu dari aplikasi yang dibangun.



Gambar 3.8 Hirarki Menu

Pada aplikasi ini menu utama terdapat pada *menu bar* atau dipojok kanan bawah tampilan windows. Dalam menu ini, apabila pengguna melakukan *double click*, aplikasi membuka *folder* CompressBox. Sedangkan apabila pengguna melakukan *right click*, maka muncul *context menu*. *Context menu* adalah menu yang tampil ketika pengguna melakukan *right click* pada suatu aplikasi. Menu ini memiliki pilihan Synchronize, Open CompressBox Folder, Setting dan Exit.

3.3.6 Tampilan Antarmuka

Berikut akan dijelaskan rancangan tampilan antarmuka pada aplikasi yang dibangun. Tampilan antarmuka dibagi menjadi lima bagian.

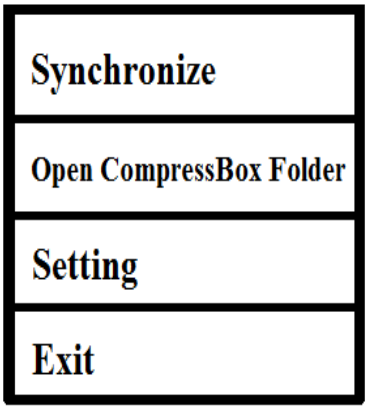
1. Menu Utama

Menu utama adalah tampilan yang terdapat pada *menu bar*. Menu utama digunakan untuk mengakses menu atau proses lain. Dengan melakukan *double click*, menu utama akan membuka *folder* CompressBox. Pengguna dapat melakukan *right click* untuk membuka *context menu*. Rancangan antarmuka menu utama diperlihatkan pada gambar 3.9.

Gambar 3.9 Rancangan antarmuka menu utama pada *menu bar*

2. Context Menu

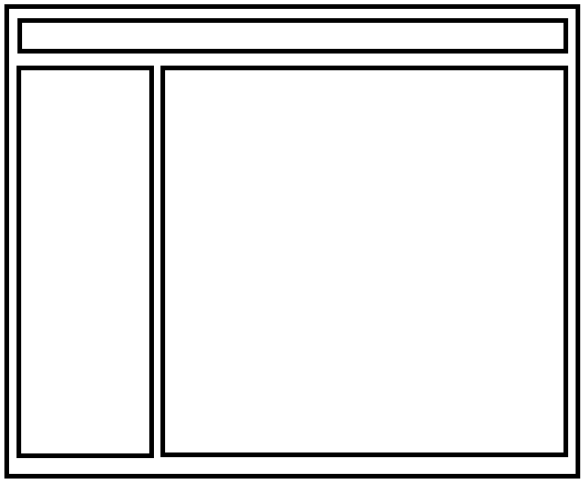
Context menu adalah tampilan saat pengguna melakukan *right click* pada *Icon* CompressBox di *menu bar*. Fungsinya untuk menggakses menu setting, membuka CompressBox *folder*, melakukan proses sinkronisasi dan menutup aplikasi CompressBox. Gambar 3.10 merupakan rancangan context menu.



Gambar 3.10 Rancangan antarmuka *context menu*

3. Open CompressBox Folder

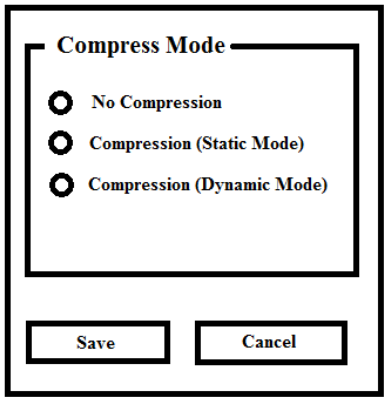
CompressBox folder akan terbuka apabila *icon* CompressBox pada *menu bar* di*double click* atau dengan memilih Open CompressBox Folder pada *context menu*. *Folder* ini digunakan untuk meletakkan *file* yang akan diunggah ke Dropbox. Selain itu, *folder* ini merupakan tempat *file* yang berhasil diunduh. Rancangan *folder* yang dibuat sama seperti *folder* pada windows. Di bawah ini merupakan gambar rancangan *folder* pada CompressBox.



Gambar 3.11 Rancangan antarmuka *folder* CompressBox

4. Setting

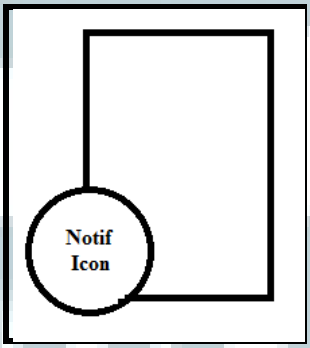
Setting adalah tempat dimana mode kompresi diatur. Dalam setting terdapat tiga pilihan mode, yaitu No Compression, *file* yang diletakkan di *folder* CompressBox tidak akan dikompres. Kedua, Compression (Fixed Mode), yang berarti *file* yang diletakkan di *folder* CompressBox akan dikompres menggunakan *fixed dictionary*. Mode terakhir yaitu Compression (Dynamic Mode), berarti *file* yang diletakkan di *folder* CompressBox akan dikompres menggunakan *dictionary* yang dihasilkan dari Huffman Code. Di bawah ini merupakan rancangan antar muka untuk *setting*.



Gambar 3.12 Rancangan antarmuka *Setting*

5. File Icon

Pada CompressBox *folder*, *file* memiliki tampilan *icon* seperti halnya Dropbox. *Icon* ini digunakan sebagai penanda *file* sedang diproses atau sudah selesai diproses. *Icon* pada *file* dapat berubah sesuai dengan kondisi *file* tersebut. Dapat diartikan pada saat *file* dikompresi *icon file* tersebut berbeda dengan pada saat *file* telah selesai dikompresi. Dibawah ini merupakan gambar rancangan *icon* pada *file*.



Gambar 3.13 Rancangan Antarmuka *File Icon*