LAPORAN APLIKASI

Huffman Coding

Ditulis sebagai tugas mata kuliah Struktur Data dan Algoritma
Program Studi D4 Teknik Informatika

Oleh :

Dewanto Joyo Pramono	(181524005)
Mufqi Uwais Nastiar Salim	(181524017)

JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
POLITEKNIK NEGERI BANDUNG

2019

1. DESKRIPSI PROGRAM

a. Nama Program

Program ini bernama “Huffman Coding”.

b. Fungsi Program

Program ini berfungsi untuk membuat pengkodean Huffman dari setiap simbol yang diinput.

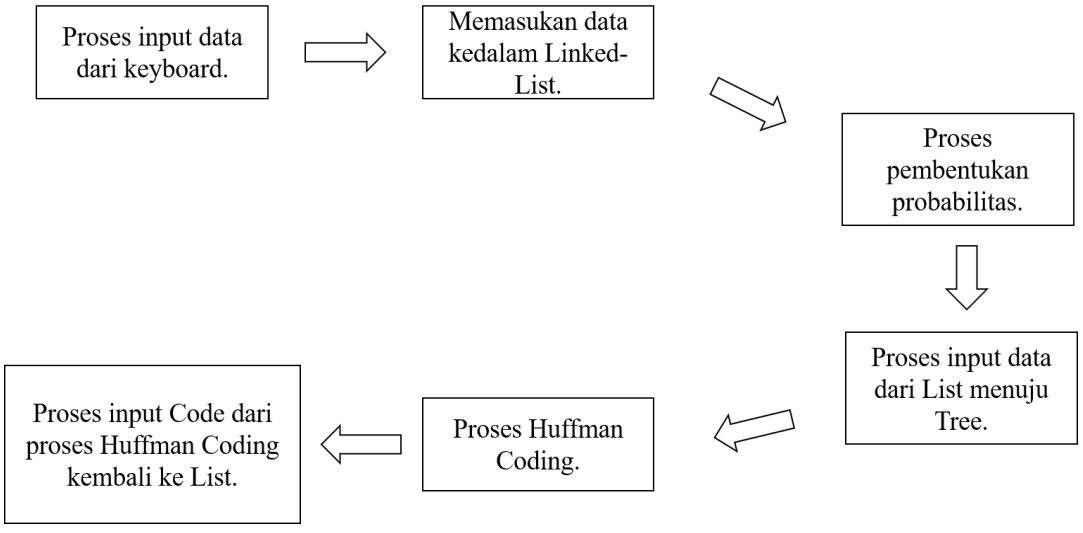
c. Tujuan Program

Tujuan dari program ini adalah sebagai proses pembelajaran bagi pembuat program maupun penggunanya dalam memahami implementasi pengkodean Huffman.

d. Konsep Program

Program ini menerima inputan pengguna yang berupa text lalu mengolahnya sehingga menghasilkan probabilitas kemunculan setiap symbol. Probabilitas tersebut digunakan untuk pembuatan tree dan tabel. Dari tree yang sudah dibuat maka kode Huffman setiap symbol akan dihasilkan.

e. Diagram Kerja Program



2. RANCANGAN

a. Rancangan Data

Data Node dalam Linked-List, tipe data record

Field:

- Huruf *char*
- Code *pointer of char*
- Jumlah *float*
- Next *pointer of Node-List*

Data Node dalam Tree, tipe data record

Field:

- Probabilitas *float*
- Symbol *char*
- Parent *pointer of Node-Tree*
- Left *pointer of Node-Tree*
- Right *pointer of Node-Tree*
- Status *integer*
- Code *pointer of char*

b. Rancangan Modul

No	Nama Modul	Fungsi
1.	void CreateList(List *L)	Untuk inisiasi Linked List
2.	bool ResetList(List *L)	Untuk return true dan menghapus isi Linked List atau return false dan tidak menghapus isi Linked List
3.	void initiateTable(addressT *Table, int ArrayLength)	Untuk inisiasi pembuatan tabel
4.	void initiateTree(BinTree *tree)	Untuk inisiasi pembuatan tree
5.	void InsertHuruf(List *L, infotype Huruf)	Untuk memasangkan simbol ke Linked List dan menghitung jumlah totalnya
6.	void CreateProbabilty(List *L, float Jumlah)	Untuk membuat probabilitas kemunculan dari setiap simbol berdasarkan jumlah huruf dalam teks
7.	void HuffmanCodingProccess(addressT Tree[TreeArrayLength], List theList)	Untuk menginput simbol-simbol beserta probabilitasnya yang berada dalam list menuju tree

8.	void executeHuffman(addressT *Table, BinTree *newTree)	Untuk menjalankan proses pengkodean Huffman
9.	void MoveCodeToList(addressT *Table, List *theList)	Untuk memasukan data dari tree menuju list
10.	bool isEmpty(List L)	Untuk return true jika list kosong atau return false jika list berisi
11.	void printTree(BinTree T, int space)	Untuk mencetak tree
12.	void PrintInfoList(List L)	Untuk mencetak isi Linked List dalam bentuk tabel

Nama Modul	void CreateList(List *L)
Fungsi	Untuk inisiasi Linked List
Jenis	Procedure
I.S	Variabel L belum diinisiasi
F.S	Variabel L sudah diinisiasi dan siap dipakai
Algoritma	1. First dari list diisi dengan NULL

Nama Modul	bool ResetList(List *L)
Fungsi	Untuk return true dan menghapus isi Linked List atau return false dan tidak menghapus isi Linked List
Jenis	Function
I.S	List masih berisi
F.S	List sudah kosong dan return true/List tetap berisi dan return false
Algoritma	1. Cek apakah List berisi, jika true maka lanjutkan, jika false maka return false 2. Konfirmasi penghapusan list, jika setuju maka list dihapus dan return true, jika tidak setuju maka return false

Nama Modul	void initiateTable(addressT *Table, int ArrayLength)
Fungsi	Untuk inisiasi pembuatan tabel
Jenis	Procedure
I.S	Variabel Table belum dialokasi
F.S	Variabel Table sudah dialokasi dengan panjang sesuai ArrayLength
Algoritma	1. Setiap anggota array/variabel Table dialokasi sepanjang ArrayLength

Nama Modul	void initiateTree(BinTree *tree)
Fungsi	Untuk inisiasi pembuatan tree
Jenis	Procedure
I.S	Variabel Tree belum dialokasi
F.S	Variabel Tree sudah dialokasi
Algoritma	1. Variabel Tree dialokasi

Nama Modul	void InsertHuruf(List *L, infotype Huruf)
Fungsi	Untuk memasangkan huruf ke Linked List dan menghitung jumlah totalnya
Jenis	Procedure
I.S	Variabel Huruf belum terpasang ke List
F.S	Variabel Huruf sudah terpasang ke List dan jumlahnya sudah terakumulasi
Algoritma	1. Cek apakah list kosong, jika true maka huruf akan menjadi First dari List, jika false maka lanjut ke proses ke-2 2. Cari huruf yang sama yang sudah ada di List, jika ada maka akumulasikan jumlahnya, jika tidak ada maka buat Node baru dalam List

Nama Modul	void CreateProbabilty(List *L, float Jumlah)
Fungsi	Untuk membuat probabilitas kemunculan dari setiap simbol berdasarkan jumlah huruf dalam teks
Jenis	Procedure
I.S	Simbol-simbol dalam List belum memiliki probabilitas
F.S	Simbol-simbol dalam List sudah memliki probabilitas
Algoritma	1. Ambil simbol dari First List lalu lakukan pembagian jumlah kemunculan simbol dengan jumlah keseluruhan/Variabel Jumlah dikali dengan 100. Hasil pembagian tersebut menjadi probabilitas bagi simbol itu. 2. Lakukan proses ke-1 dari First List hingga akhir List

Nama Modul	void HuffmanCodingProccess(addressT Tree[TreeArrayLength], List theList)
Fungsi	Untuk menginput simbol-simbol beserta probabilitasnya yang berada dalam list menuju tree
Jenis	Procedure
I.S	Tree kosong
F.S	Tree berisi simbol-simbol dan probabilitasnya
Algoritma	1. Ambil simbol dari First List lalu lakukan alokasi Node dalam tree berdasarkan simbol tersebut dan probabilitasnya 2. Lakukan proses ke-1 dari First List hingga ke akhir List

Nama Modul	void executeHuffman(addressT *Table, BinTree *newTree)
Fungsi	Untuk menjalankan proses pengkodean Huffman
Jenis	Procedure
I.S	Variabel newTree kosong dan kode belum dibuat
F.S	Variabel newTree sudah terisi dan kode setiap simbol sudah

	dibuat
Algoritma	1. Cari 2 Node dengan probabilitas terkecil 2. Buat parent dari kedua Node tersebut dengan simbol kosong dan probabilitas gabungan dari keduanya 3. Probabilitas paling kecil diantara keduanya menjadi anak kiri, dan yang lainnya menjadi anak kanan 4. Jadikan parent tersebut menjadi newTree yang baru 5. Lakukan proses 1, 2, 3, dan 4 hingga semua Node bersimbol sudah memiliki parent dan isi Table berjumlah ((Jumlah Simbol*2) - Jumlah Simbol) 6. Buat code setiap simbol dengan memanfaatkan newTree sebagai jalur dan proses concatenate 2 string sebagai penampung kode 7. Jika Node simbol merupakan anak kiri, maka buat string “0”, jika anak kanan maka buat string “1” 8. Lakukan proses ke-7 hingga menuju root dengan proses concatenate 9. Saat proses concatenate selesai maka kode untuk simbol itu sudah dibuat dan disimpan dalam Node-nya 10. Lakukan proses 6, 7, 8, dan 9 hingga semua simbol sudah memiliki kode

Nama Modul	void MoveCodeToList(addressT *Table, List *theList)
Fungsi	Memindahkan hasil pembentukan Huffman Code dari setiap simbol ke List agar mudah diakses.
Jenis	Procedure
I.S	Code Huffman setiap simbol dalam list kosong.
F.S	Code Huffman setiap simbol dalam list terisi.
Algoritma	1. Baca setiap simbol dari sebuah List. 2. Banding setiap simbol yang sudah dibaca dengan setiap simbol yang berada dalam Tabel Tree. 3. Jika simbolnya sama maka pindahkan Code Huffman yang telah terbentuk ke List.

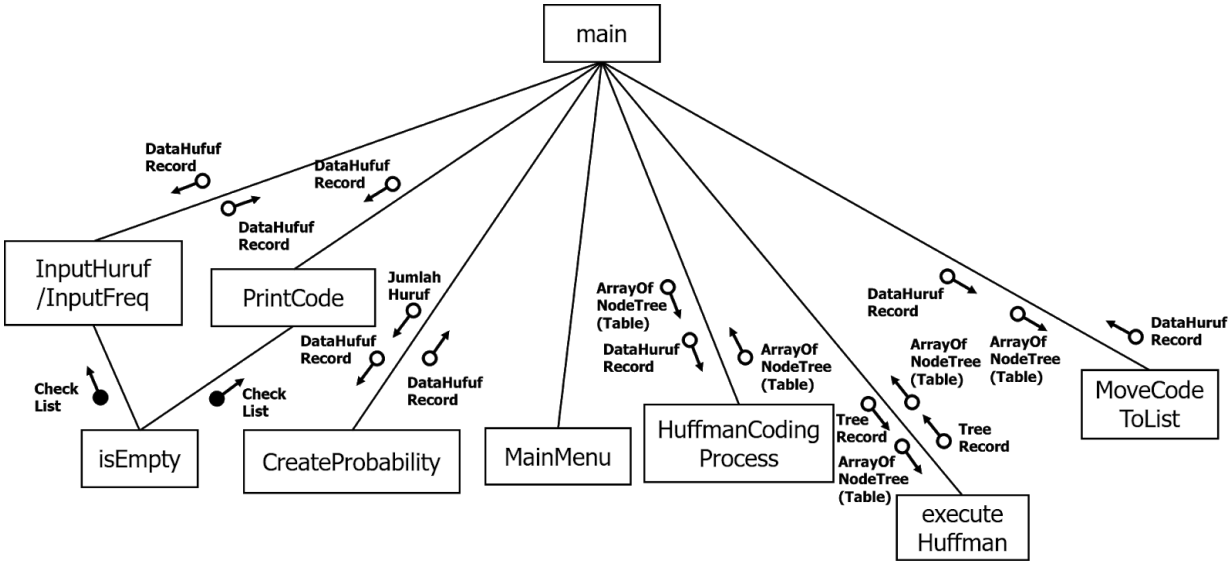
Nama Modul	bool isEmpty(List L)
Fungsi	Memeriksa List kosong/tidak.
Jenis	Fungsi

I.S	-
F.S	-
Algoritma	1. Baca node pertama dalam List. 2. Mengembalikan nilai true apabila node pertama dalam List bernilai NULL dan mengembalikan nilai false apabila node pertama dalam List tidak NULL.

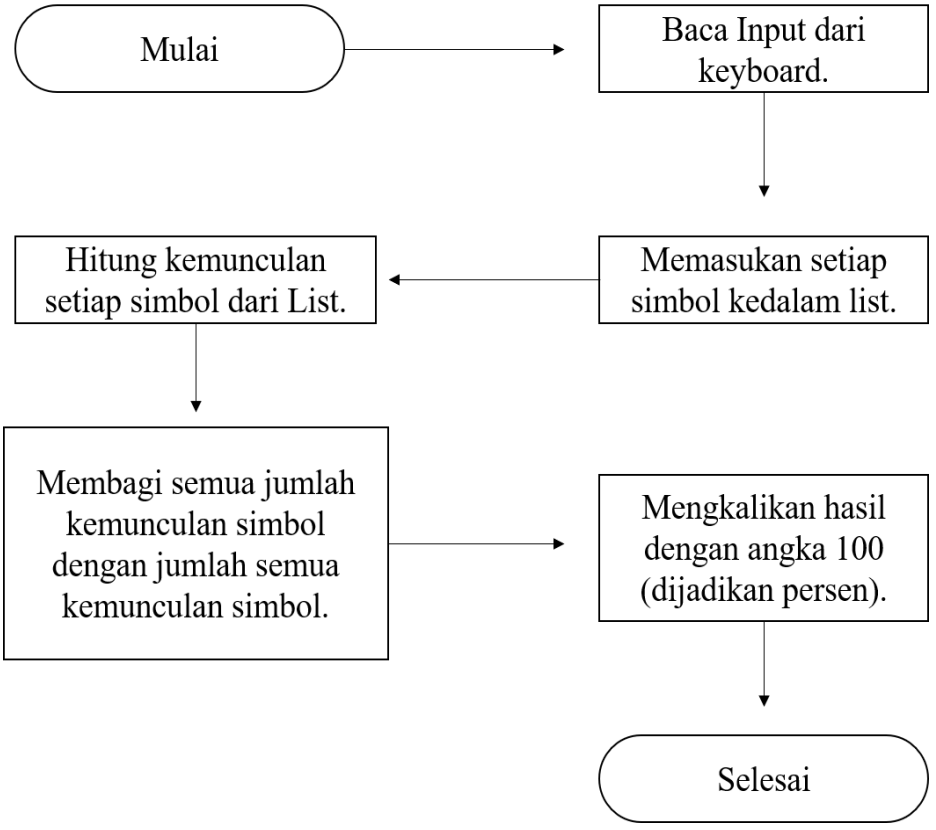
Nama Modul	void printTree(BinTree T, int space)
Fungsi	Untuk mencetak tree berdasarkan hierarki.
Jenis	Procedure
I.S	-
F.S	Tree dicetak di layar
Algoritma	1. Dahulukan anak kanan untuk dicetak 2. Lalu cetak parent 3. Lalu cetak anak kiri 4. Setiap level di pisah dengan spasi/variabel space

Nama Modul	void PrintInfoList(List L)
Fungsi	Menampilkan semua info node dalam Linked-List yang berupa Simbol, Probabilitas, serta Code Huffmanl.
Jenis	Procedure
I.S	-
F.S	Simbol, Probabilitas, seta Code dari List muncul di layar.
Algoritma	1. Membaca node pertama dari List. 2. Jika ada, maka tampilkan Simbol, Probabilitas dan Code Huffman. 3. Baca node selanjutnya dan ulangi langkah 2 hingga node terujung dari List.

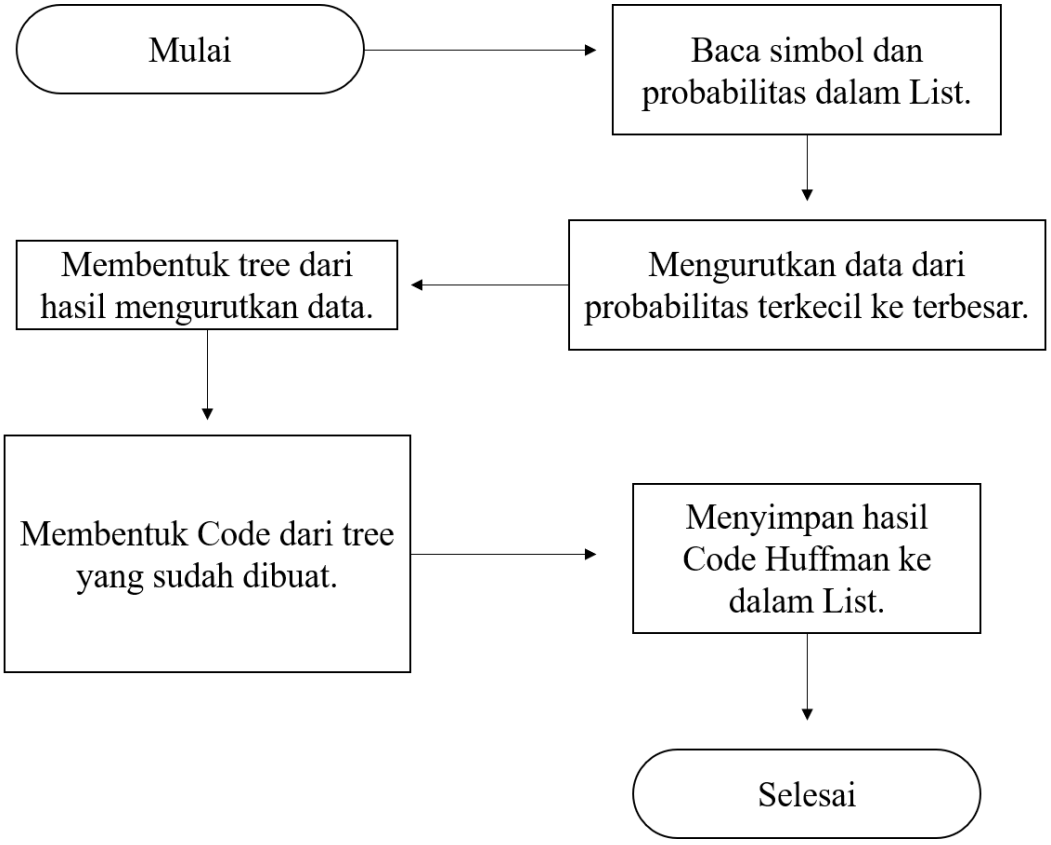
c. Proses dan relasi antar modul



d. Proses Pemodelan dan Probabilitas



e. Proses Huffman Encoding



3. KENDALA

Berikut beberapa kendala kami dalam membuat program ini.

1. Program akan error apabila suatu variabel diisi dengan input yang berbeda dengan tipe variabelnya.
2. Dikarenakan program masih menerima input berupa teks dari keyboard maka jumlah elemen array harus disesuaikan dengan input.
3. Program akan error apabila jumlah semua simbol yang berbeda melebihi jumlah maksimal elemen dari array yang dipakai saat membentuk tree. Oleh karena itu jumlah element kami sesuaikan dengan jumlah kode ASCII sehingga dapat meminimalisir error.
4. Saat pertama kali menentukan jumlah probabilitas kami sempat mengalami error ketika probabilitas semua huruf jika dijumlahkan mencapai angka 100 lebih yang dikarenakan probabilitas dari huruf bernilai 33,3334 dikali 3(Jumlah huruf).

4. LESSON LEARNED

Dalam pembuatan program ini terdapat beberapa hal yang kami peroleh antara lain.

1. Kami dapat mengetahui proses sebenarnya yang terjadi saat mengkompres suatu data.
2. Kami dapat mengetahui salah satu fungsi dari sebuah struktur data tree yang sangat berguna dalam bidang kami.
3. Kami dapat mengetahui mengenai prosedur “concat” untuk membentuk suatu Code Biner (Huffman).

5. Referensi

Function concat:
https://stackoverflow.com/questions/8465006/how-do-i-concatenate-two-strings-in-c/8465083
Function printTree :
https://www.geeksforgeeks.org/print-binary-tree-2-dimensions/