# Laporan Tugas Besar

# EL2008 – Pemecahan Masalah dengan C


Fannan Bachtiar (18320021)

Christopher Chandra (18320033)

Febrian Vivaldi (18320035)

Sekolah Dasar Teknik Elektro

Institut Teknologi Bandung

Bandung

# Daftar Isi

## 1. Studi Pustaka

### 1.1 Minimisasi

Minimisasi, pada konteks ini minimisasi logic, adalah proses menyederhanakan sebuah fungsi boolean. Minimisasi penting karena minimisasi mengurangi jumlah komponen yang diperlukan dalam sirkuit, yang akan mengakibatkan pengurangan harga dan kompleksitas. Terdapat dua metode minimisasi yang umum digunakan, yaitu aljabar dan karnaugh map [4].

### 1.2 K-Map

K-Map atau Karnaugh map adalah salah satu metode minimisasi aljabar boolean. Metode ini memanfaatkan kemampuan "*pattern-recognition*" manusia. Pada implementasinya di dunia asli, K-Map digunakan untuk melakukan minimisasi pada rangkaian logic gate dengan tujuan mengimplementasi rangkaian tersebut dengan *logic gate* sesedikit mungkin.

K-Map merupakan metode *pictorial* dengan pengelompokan bersama ekspresi yang memiliki faktor umum yang sama kemudian mengeliminasi variabel yang tidak diinginkan. K-Map juga dapat dideskripsikan sebagai *special arragement* dari *truth table*, [1].

| A | B | F |
|---|---|---|
| 0 | 0 | a |
| 0 | 1 | b |
| 1 | 0 | c |
| 1 | 1 | d |

Truth Table.

| A / B | 0 | 1 |
|---|---|---|
| 0 | a | b |
| 1 | c | d |

F.

Gambar 1. Truth table dan Karnaugh Map

Pada gambar diatas menunjukkan transformasi dari *truth table* menuju K-map, nilai pada K-map didapatkan dari kolom output pada *truth table.* Maka dari itu, terdapat satu kotak pada k-map untuk setiap baris dari *truth table*. Di pinggir dari Karnaugh Map merupakan nilai dari dua input variabel, [1]

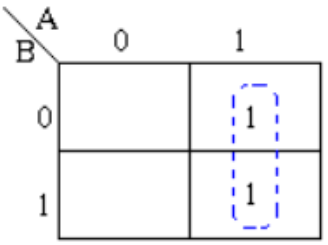
Pada truth table sebelumnya, fungsi yang didapat adalah F(A,B) = $A\overline{B} + AB$. Apabila diselesaikan dengan menggunakan metoda aljabar :

F(A,B) = $A\overline{B} + AB$

F(A,B) = $A(\overline{B} + B)$

F(A,B) = A

Ketika digunakan Karnaugh maps, akan didapati map,




Gambar 2. Minimalisasi Logika menggunakan Karnaugh Map

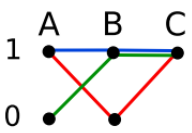
Kedua kotak yang berisi 1 *(true)* dikelompokkan bersama. Dengan melihat pada kedua kotak variabel B memiliki nilai yang berbeda yakni 0 *(false)* dan 1, kita dapat mengeliminasi variabel B dan didapati F = A.

### 1.3 T-Graph

T-Graph atau Truth Graph adalah salah satu metode minimisasi aljabar boolean dengan menggunakan grafik. Metode ini muncul sebagai alternatif bagi K-map karena K-map sulit untuk menyelesaikan permasalahan dengan lebih dari empat variabel. Metoda ini menggunakan notasi visual berupa garis dan node untuk mempermudah pemahaman dan minimisasi.

*Truth table* merupakan alat yang umum untuk menunjukkan *truth value*. Pada truth graph, kombinasi yang bernilai false atau 0 tidak dimasukkan dalam proses minimalisasi, sehingga *truth graph* hanya berisi kombinasi yang bernilai true atau 1, [2].

| A | B | C | Output |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

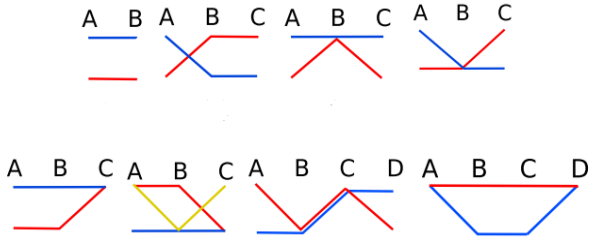



Gambar 3. Truth Value dan Truth Graph

Minimisasi pada metode truth graph memiliki dua syarat :

1. Jika dua garis (path) yang bertemu dengan variabel dan nilai yang sama, kecuali untuk satu variabel dimana satu melalui 0 dan yang lain 1, maka variabel tersebut dieleminasi. Contohnya $ABC + \overline{A}BC = BC$.
2. Jika satu path bertemu dengan variabel yang lebih kecil daripada path lain, dan path yang lebih pendek bertemu dengan variabel dan nilai yang sama , kecuali untuk satu variabel, maka nilai yang dapat dieleminasi hanya dari path yang memiliki variabel lebih banyak. Contohnya $AB + \overline{A}BC = AB + BC$.

Minimisasi pada truth graph tidak dapat dilakukan pada kombinasi dari 3 variabel atau lebih yang berbagi 1 variabel yang sama,



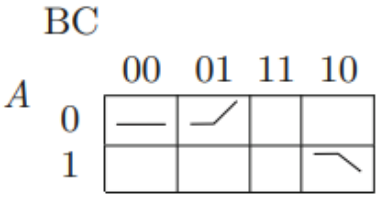

Gambar 4. Graph yang Tidak Bisa Diminimisasi

Sebagai contoh minimisasi, K-map pada tabel di bawah merepresentasikan nilai 000 + 001 + 110.

| A \ BC | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |

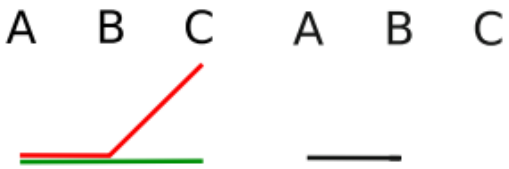
Gambar 5. K-Map dengan Truth Value

Kemudian *truth value* diubah menjadi *truth graph* dengan menggambarkan garis yang naik, turun, maupun lurus.



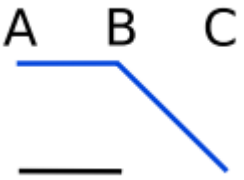

Gambar 6. K-Map dengan Truth Value

Truth path pada baris pertama digabungkan dan didapatkan hasil sebagai berikut,




Gambar 7. Truth Value dari Baris Pertama dan Truth Value Setelah di Eliminasi

Terlihat garis merah dan hijau memiliki nilai yang sama kecuali untuk variabel C, sehingga variabel C dieliminasi dan didapatkan hasil seperti gambar. Dengan menambahkan path ketiga dari baris kedua dengan hasil eliminasi, didapatkan path sebagai berikut,




Gambar 8. Hasil Truth Value Akhir

Dari path di atas, terlihat tidak ada nilai yang sama untuk garis biru dan hitam. Sehingga hasil akhir yang merupakan persamaan paling sederhana adalah $\overline{AB} + AB\overline{C}$.

### 1.4 Metode Quine-McCluskey

Metode Quine-McCluskey pertama kali dikembangkan oleh W.V. Quine dan E.J. McCluskey pada tahun 1950. Metode ini terdiri dari dua bagian yaitu penentuan kandidat *prime implicant* dan pemilihan *prime implicant* itu sendiri. *Prime implicant* adalah suku *(minterm)* yang menjadi calon untuk dicantumkan dalam fungsi yang disederhanakan. Kemudian dari semua p*rime implicant* tersebut akan dipilih suku *(minterm)* yang akan menghasilkan fungsi paling sederhana, [3].

## 2. Metodologi

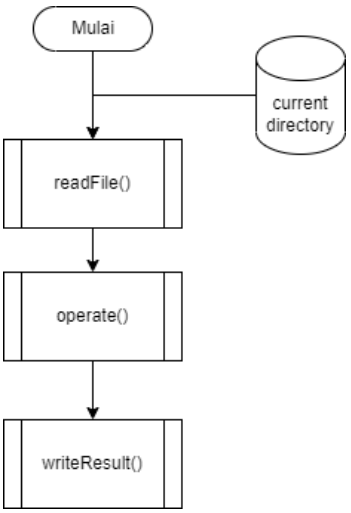
### 2.1 Spesifikasi

Program yang dibuat menggunakan algoritma *brute force* yang menggunakan campuran dari metode T-graph dan Quine-McCluskey. Program tersusun atas eksekutor utama main.c. Main.c secara sekuensial akan menjalan fungsi readFile(), operate(), dan

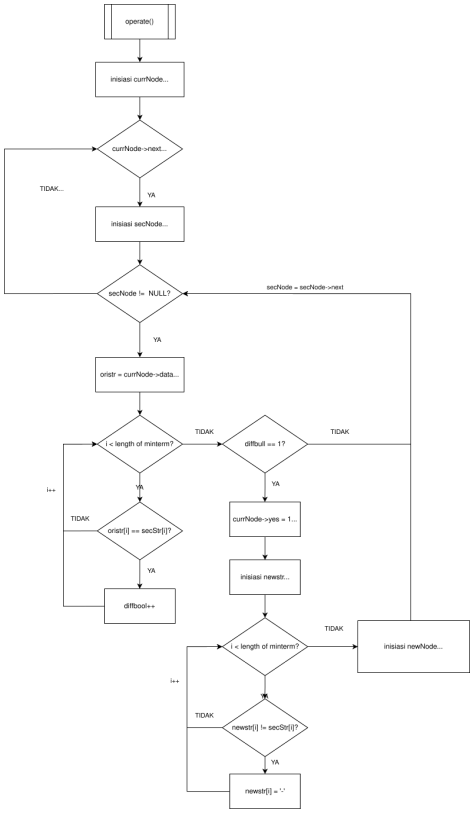
writeResult(). Pada bagian input, program menerima file .txt dari pengguna yang berisi minterm dan menyimpan minterm pada *node stack*. Stack ini kemudian diakses oleh fungsi operate() yang menjalankan skema campuran antara metoda T-graph dan Quine McCluskey untuk memperoleh simplifikasi dari minterm. Minterm yang telah disimplifikasi ini disusun dalam stack yang kemudian diakses oleh writeResult(). Pada bagian output, writeResult() menuliskan tiap minterm yang diterjemahkan dalam definisi aljabar boolean.

## 2.2 FLOWCHART

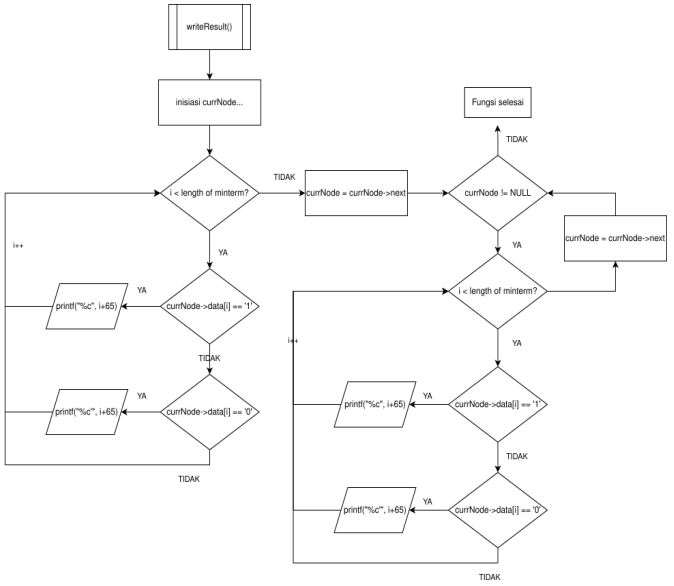
a. main()




b. operate()

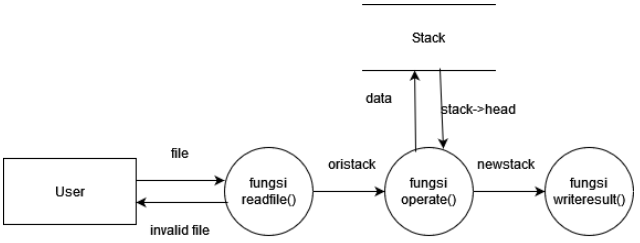



c. writeResult()




d. readFile()

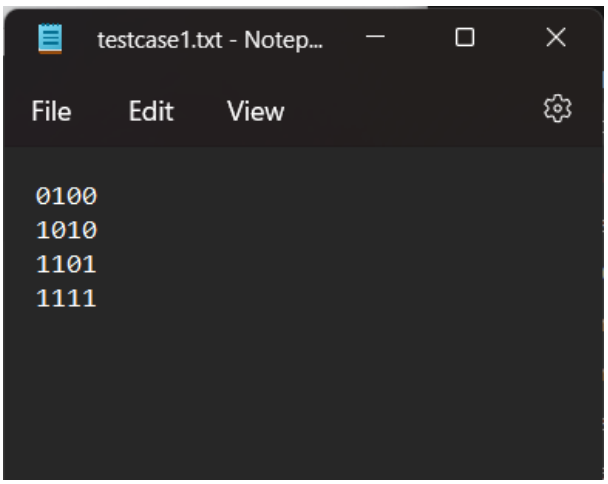
## 2.3 DATA FLOW DIAGRAM



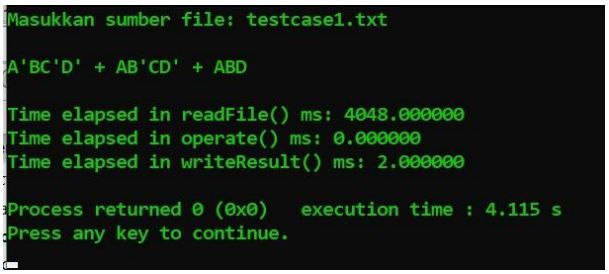

## 3. HASIL DAN ANALISIS

### 3.1 HASIL

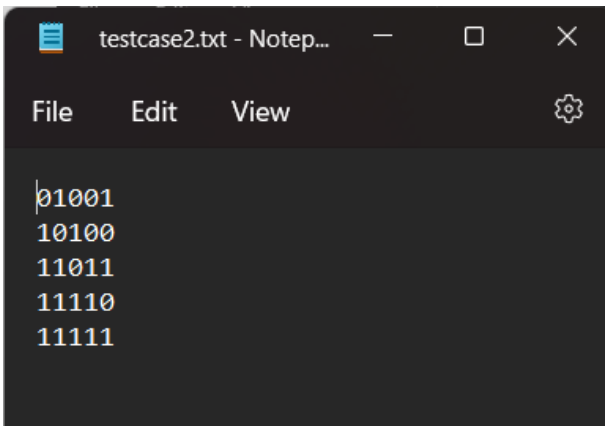
Pengujian pertama, program di test dengan file sebagai berikut,



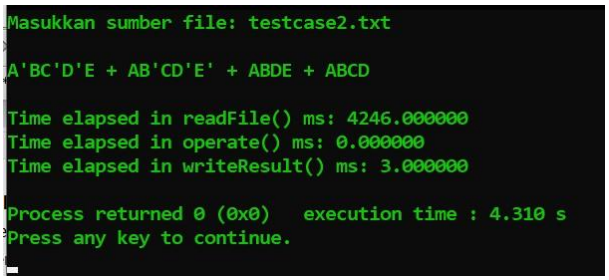

Didapat hasil keluaran program,



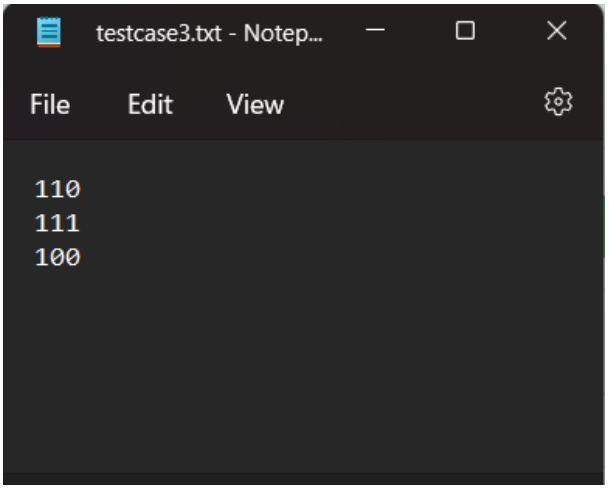

Untuk pengujian kedua, program di test dengan file sebagai berikut,



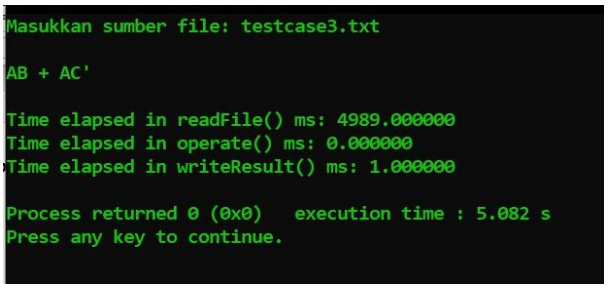

Didapat hasil keluaran,




Untuk pengujian ketiga, program di test dengan file sebagai berikut,




Didapat hasil keluaran,




Dari hasil pengujian, didapati program dapat melakukan minimisasi logika, dan memiliki hasil yang sama dengan hasil dari *website online minimization.* Bahkan untuk *test case* 1, hasil keluaran program memiliki hasil yang lebih sederhana dibandingkan hasil *website*.

### 3.2 Analisis Kompleksitas Waktu

Kompleksitas waktu program dapat ditinjau dari beban terbesar di antara kumpulan fungsi yang dipanggil. Pada fungsi readFile() pembacaan tiap baris fail dan pengisian nodus merupakan suatu biaya waktu sebut O(k). Kasus terburuk dari deskripsi demikian adalah seluruh minterm dituliskan, yaitu 2^n, untuk n tiap penambahan variabel boolean. Dengan demikian, kompleksitas waktu terburuk readFile adalah O(2^n).

Di sisi lain, kompleksitas waktu pada fungsi operate() dapat ditinjau dari dua ciri program: skema *handshake* dan skema rekursi. Skema *handshake* mengacu pada tiap nodus pada tumpukan orisinal mengecek sesamanya. Kasus terburuk dari banyaknya *handshake* ini adalah (2^n)! perulangan, sehingga dapat disebut sebagai O(2^n)!. Ada pula hasil dari tiap *handshake* ini saling mengecek kembali jika simplifikasi belum berakhir. Dengan asumsi simplifikasi hingga ketunggalan, rekursi memberikan beban kelipatan O(n!) pada beban tiap eksekusi fungsi. Dengan itu, kompleksitas waktu terburuk fungsi operate adalah O((2^n!)!).

Ada pula bagian fungsi writeSource() yang hanya kelipatan 2^n dari ekspansi ukuran minterm dengan asumsi terburuk tidak ada simplifikasi. Berdasarkan pertimbangan tersebut, total kompleksitas waktu terburuk program adalah O((2^n!)!).

### 3.3 Analisis Kompleksitas Ruang

Kasus terburuk dari masukan minterm adalah 2^n, yaitu semua minterm dari sejumlah n variabel. Dengan itu, biaya ruang untuk

inisiasi nodus untuk tiap baris pada fungsi readFile() akan berlaku kelipatan tersebut, yaitu S(2^n). Demikian pula dengan writeResult().

Lain halnya dengan operate(), rekursi *handshake* untuk kasus terburuk mengakibatkan biaya pemanggilan nodus S(k) dilipatgandakan dengan (2^n!)!, sehingga kompleksitas ruang fungsi operate adalah S((2^n!)!).

Oleh karena itu, kompleksitas ruang terburuk program adalah S((2^n!)!).

## 4. Kesimpulan dan Lesson Learned

Pada tugas ini, didapat kesimpulan bahwa,

1. Dengan menggunakan metode campuran, masalah minimisasi dapat diselesaikan.
2. Program yang dibuat lebih pendek dibandingkan cara lain seperti McCluskey.
3. Untuk beberapa test case, program mengeluarkan output yang lebih sederhana dari website penyederhana aljabar boolean.

Dari tugas ini, didapat pelajaran bahwa terdapat berbagai macam metode untuk melakukan minimisasi, dan cara-cara implementasi metode tersebut dalam bahasa pemrograman C.

## 5. Pembagian Tugas

| NIM | Tugas |
|---|---|
| 18320021 | |
| 18320033 | |
| 18320035 | |

## Daftar Pustaka


[1] http://www.ee.surrey.ac.uk/Projects/Labview/minimisation/karnaugh.html, Diakses pada 18 Mei 2022, Pukul 18.20 WIB

[2] Eisa A., *Truth Graph: A Novel Method for Minimizing Boolean Algebra Expressions by Using Graphs*, Kuwait Oil Company, Kuwait, 2020.

[3] https://wirasetiawan29.wordpress.com/2011/10/11/metode-quine-mc-cluskey/, Diakses 20 Mei 2022. 2.16 WIB.

[4] https://www.geeksforgeeks.org/minimization-of-boolean-functions/, Diakses 17 Mei 2022. 14.30 WIB.