TUGAS BESAR – MINIMISASI FUNGSI LOGIKA

Kelompok 13
Muhammad Fauzan (13220009)
Deovie Lentera (18320037)
Muhammad Rafli F.Y (13220037)
Bayu Aji Nugroho (13221601)
Tanggal Percobaan: 18/04/2022
EL2008-Pemecahan Masalah dengan C
Laboratorium Dasar Teknik Elektro - Sekolah Teknik Elektro dan Informatika ITB

Abstrak

Dalam sistem digital, telah dikenal berbagai metode minimisasi fungsi logika salah satu dari metode tersebut adalah metode minimisasi dengan menggunakan algoritma Quine-McCluskey. Algoritma Quine-McCluskey merupakan algoritma yang dapat dikerjakan baik secara manual maupun dengan mesin. Pada tugas besar ini, akan dilakukan percobaan implementasi algoritma Quine-McCluskey menggunakan program bahasa C. Dari percobaan yang dilakukan, telah berhasil memperoleh minimisasi fungsi logika yang sesuai dengan perhitungan manual. Dengan menggunakan program bahasa C, algoritma Quine-McCluskey lebih cepat diselesaikan dengan hasil yang baik.

Kata kunci: Minimisasi, Fungsi, Quine-McCluskey.

1. PENDAHULUAN

Dalam system digital, minimisasi fungsi logika merupakan sebuah proses untuk menyederhanakan sebuah fungsi logika. Proses ini digunakan dalam melakukan desain perangkat elektronika digital dan desain sirkuit terpadu. Pada umumnya, sirkuit dibatasi area chip minimum yang telah ditentukan. Selain itu pengurangan dari fungsi logika mengurangi waktu respons penundaan (delay) dalam suatu proses. Sehingga minimisasi fungsi logika sangat penting dalam dunia digital.

Ada beberapa banyak cara yang dapat digunakan untuk melakukan minimisasi fungsi logika. Salah satu cara penyederhanaan fungsi logika adalah dengan menggunakan algoritma Quine-McClusky. Metode Quine-McCluskey sering juga disebut dengan metode tabulasi. Metode ini dapat digunakan untuk menyelesaikan minimasi fungsi dengan lebih dari lima variabel.

Dalam minimisasi menggunakan algoritma Quine-McCluskey pada dasarnya dapat dilakukan dengan menggunakan proses perhitungan tangan. Akan tetapi, hal ini tentunya memerlukan proses yang panjang dan lama. Oleh karena itu pada percobaan tugas besar yang disusun ini akan dilakukan implementasi algoritma Quine-McCluskey pada program bahasa C dalam minimisasi fungsi logika. Penyelesaian masalah dengan program bahasa C tentunya akan memberikan hasil penyelesaian yang tepat dan cepat.

2. STUDI PUSTAKA

Dalam melakukan penelitian atau percobaan tentunya membutuhkan referensi dan literature yang dapat digunakan sebagai landasan teori. Hal tersebut dilakukan agar penelitian dan percobaan yang dilakukan valid dan dapat teruji secara ilmiah. Dalam percobaan minimisasi fungsi logika, dasar teori yang digunakan adalah algoritma Quine-McCluskey.

2.1 ALGORITMA QUINE-MCCLUSKEY

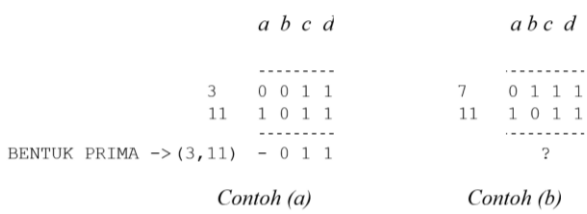
Metode Quine-McCluskey adalah sebuah metode yang digunakan untuk menyederhanakan fungsi Boolean, khususnya fungsi Boolean yang memiliki jumlah peubah yang besar (di atas 6 buah). Metode Quine-McCluskey dikembangkan oleh W.V. Quine dan E.J. McCluskey pada tahun 1950.

Metode ini mengubah sebuah fungsi Boolean menjadi sebuah himpunan bentuk prima, dimana sebanyak mungkin peubah dieliminasi (dihilangkan) secara maksimal, hingga didapat fungsi Boolean yang paling sederhana. Ini dapat dilakukan dengan melakukan perulangan penggunaan hukum komplemen, $a + a' = 1$. Sebagai contoh, fungsi Boolean dengan empat peubah dalam bentuk SOP :

$$f(a,b,c,d)=\sum(3,11)=\sum(0011,1011)=a'b'cd + ab'cd$$

dan

$$f(a,b,c,d)=\sum(7,11)=\sum(0111,1011)=a'b'cd + ab'cd.$$



Gambar 2.1 Bentuk prima dari (3,11)

Pada gambar 2.1(a), kedua *minterm* tersebut dapat dikombinasikan menjadi sebuah bentuk prima yaitu (3,11), karena memiliki tepat satu perbedaan bit pada posisi bit nomor satu. Hasil kombinasi dalam bentuk prima (3,11) menyatakan bahwa peubah *a*" telah dieleminasi. Hal ini sesuai dengan hukum komplemen, *a*+*a*'=1.

Pada Gambar 2.1(b), kedua *minterm* tersebut tidak dapat dikombinasikan menjadi sebuah bentuk prima, karena memiliki dua perbedaan bit pada posisi bit nomor satu dan dua. Setiap kombinasi dari *minterm* yang dapat membentuk sebuah bentuk prima baru harus memiliki tepat satu perbedaan bit pada posisi yang sama.

Secara umum, langkah – langkah metode Quine-McCluskey untuk menyederhanakan fungsi Boolean dalam bentuk SOP adalah sebagai berikut:

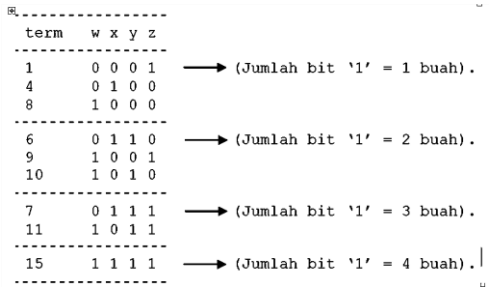
1. Nyatakan tiap *minterm* dalam *n* peubah menjadi *string bit* yang panjangnya *n*, yang dalam hal ini peubah komplemen dinyatakan dengan „0", peubah yang bukan komplemen dengan „1".
2. Kelompokkan tiap *minterm* berdasarkan jumlah „1" yang dimilikinya.
3. Kombinasikan *minterm* dalam *n* peubah dengan kelompok lain yang jumlah „1"-nya berbeda satu, sehingga diperoleh bentuk prima (*prime-implicant*) yang terdiri dari *n* - 1 peubah. *Minterm* yang dikombinasikan diberi tanda "√".
4. Kombinasikan *minterm* dalam *n* – 1 peubah dengan kelompok lain yang jumlah „1"-nya berbeda satu, sehingga diperoleh bentuk prima yang terdiri dari *n* – 2 peubah.
5. Teruskan langkah 4 sampai diperoleh bentuk prima yang sesederhana mungkin.
6. Ambil semua bentuk prima yang tidak bertanda "√". Buatlah tabel baru yang memperlihatkan *minterm* dari ekspresi Boolean semula yang dicakup oleh bentuk prima tersebut (tandai dengan "√"). Setiap *minterm* harus dicakup oleh paling sedikit satu buah bentuk prima.
7. Pilih bentuk prima yang memiliki jumlah literal paling sedikit namun mencakup sebanyak mungkin *minterm* dari ekspresi Boolean semula. Hal ini dapat dilakukan dengan cara berikut :
   a. Tandai kolom – kolom yang mempunyai satu buah tanda "x" dengan tanda "*", lalu beri tanda "√" di sebelah kiri bentuk prima yang berasosiasi dengan tanda "*" tersebut. Bentuk prima ini telah dipilih untuk fungsi Boolean sederhana.
   b. Untuk setiap bentuk prima yang telah ditandai dengan "√", beri tanda *minterm* yang dicakup oleh bentuk prima tersebut dengan tanda "√" (di baris bawah setelah "*").
   c. Periksa apakah masih ada *minterm* yang belum dicakup oleh bentuk prima terpilih. Jika ada, pilih dari bentuk prima yang tersisa yang mencakup sebanyak mungkin *minterm* tersebut. Beri tanda "√" bentuk prima yang dipilih itu serta *minterm* yang dicakupnya.
   d. Ulangi langkah c sampai seluruh *minterm* sudah dicakup oleh semua bentuk prima.

Langkah – langkah penyederhanaan metode Quine-McCluskey di atas juga berlaku untuk penyederhanaan fungsi Boolean dalam bentuk POS. Dapat dilihat bahwa bentuk fungsi *output* selalu sama dengan bentuk fungsi *input*, artinya *input* dalam bentuk SOP akan menghasilkan *output* dalam bentuk SOP, dan demikian pula untuk bentuk POS.

Langkah–langkah penyederhanaan metode Quine-McCluskey di atas juga berlaku untuk penyederhanaan fungsi Boolean dalam bentuk POS. Dapat dilihat bahwa bentuk fungsi *output* selalu sama dengan bentuk fungsi *input*, artinya *input* dalam bentuk SOP akan menghasilkan *output* dalam bentuk SOP, dan demikian pula untuk bentuk POS. Agar lebih jelas, perhatikan contoh berikut. Bentuk *input* dalam bentuk SOP:

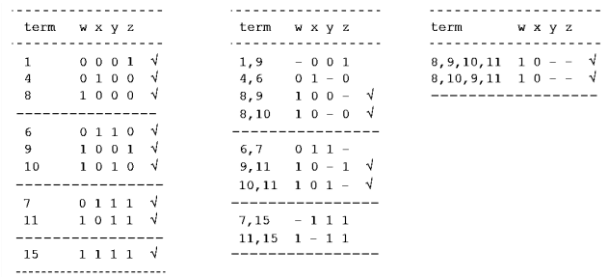
$f(w, x, y, z) = \sum(1, 4, 6, 7, 8, 9, 10, 11, 15)$

Langkah – langkah minimisasi yang dilakukan adalah sebagai berikut: (Langkah 1 dan langkah 2) Konversikan nilai *minterm* ke bentuk biner dengan panjang sebesar n peubah (4 bit) dan kelompokkan tiap minterm berdasarkan jumlah bit „1" yang dimilikinya.



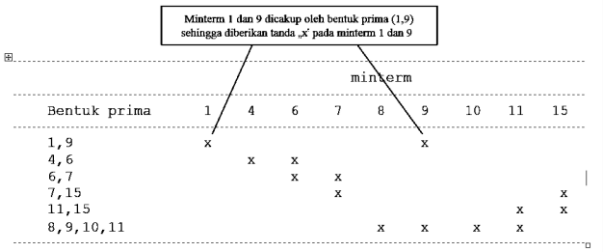
Gambar 2.2 Langkah 3 dan 5

(Langkah 3 sampai 5) Kombinasikan *term* atau bentuk prima yang memiliki perbedaan tepat satu bit pada posisi yang sama. Hasil kombinasi merupakan bentuk prima baru. Lakukan hingga didapat bentuk prima sesederhana mungkin. *Term* atau bentuk prima yang dikombinasikan diberi tanda '√'.



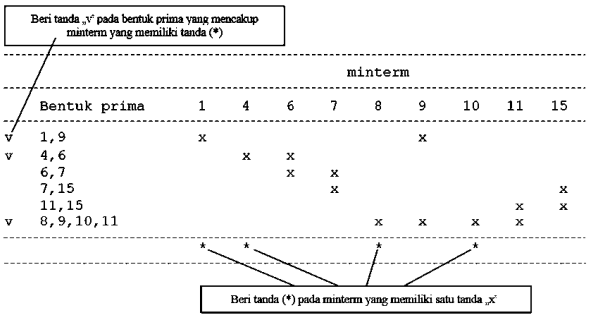
Gambar 2.3 Langkah 6

(Langkah 6) Ambil semua bentuk prima yang tidak bertanda "√". Buat tabel baru yang memperlihatkan *minterm* dari ekspresi Boolean semula yang dicakup oleh bentuk prima tersebut (tandai dengan "x"). Setiap *minterm* harus dicakup oleh paling sedikit satu buah bentuk prima.



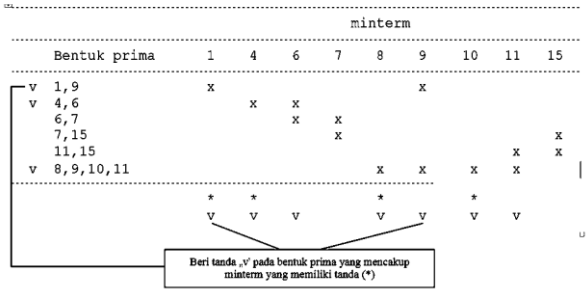
Gambar 2.4 Langkah 7a

(Langkah 7.a) Tandai kolom–kolom yang mempunyai satu buah tanda "x" dengan tanda "*", lalu beri tanda "√" di sebelah kiri bentuk prima yang berasosiasi dengan tanda "*" tersebut. Bentuk prima ini telah dipilih untuk fungsi Boolean sederhana.



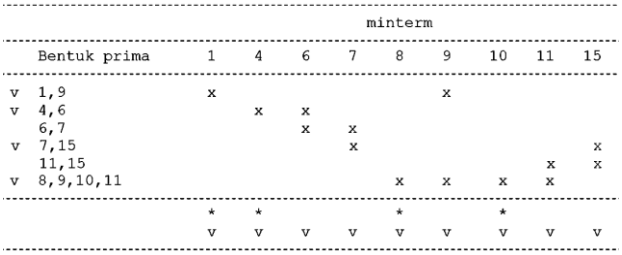
Gambar 2.5 Langkah 7b

(Langkah 7.b) Untuk setiap bentuk prima yang telah ditandai dengan "√", beri tanda *minterm* yang dicakup oleh bentuk prima tersebut dengan tanda "√" (di baris bawah setelah "*").



Gambar 2.6 Langkah 7c dan 7d

(Langkah 7.c dan 7.d) Sampai tahap ini, masih ada *minterm* yang belum tercakup dalam bentuk prima terpilih, yaitu 7, 15. Untuk mencakup *minterm* tersebut, maka dipilih bentuk prima (7,15), karena mencakup *minterm* 7 dan 15 sekaligus.



Gambar 2.7 Hasil akhir

Semua minterm sudah tercakup dalam bentuk prima terpilih. Bentuk prima yang terpilih adalah :

1,9 yang bersesuaian dengan term $x'\,y'\,z$

4,6 yang bersesuaian dengan term $w'\,xz'$

7,15 yang bersesuaian dengan term xyz

8,9,10,11 yang bersesuaian dengan term wx'

Dengan demikian, fungsi Boolean hasil penyederhanaan dengan metode Quine-McCluskey adalah:

$$f(w, x, y, z) = x'y'z + w'xz' + xyz + wx'$$

3. METODOLOGI

Dalam menyelesaikan tugas besar ini, kelompok kami melaksanakan beberapa tahap kegiatan. Pertama, dilakukan pengkajian dan studi literatur dengan membaca referensi dari berbagai literatur yang berkaitan. Kemudian memilih literatur yang dapat dijadikan acuan dalam pembuatan program. Setelah diperoleh beberapa literatur yang sesuai, dilakukan perangkuman untuk menentukan algoritma yang sesuai sekaligus membahas poin-poin penting dari tujuan yang ingin dicapai. Kegiatan berikutnya dilakukan pembuatan flowchart program.

Setelah flowchart program dibuat, tahap selanjutnya adalah implementasi flowchart ke sebuah program yang ditulis dalam bahasa C. Pembuatan kode program ini dilakukan terus-menerus dengan menggunakan metode trial and error sehingga perlu dilakukan revisi hingga kode program yang dibuat dapat mendapatkan output yang optimal dan sesuai dengan spesifikasi yang diharapkan. Kemudian dari program tersebut dilakukan perbandingan hasil hitung manual dengan menggunakan program. Apabila hasil telah sesuai, selanjutnya dilakukan analisa kompleksitas waktu dan ruang. Tahap terakhir dari penelitian adalah pemaparan kode yang berhasil dijalankan tersebut ke dalam bentuk laporan. Urutan dalam metodologi penelitian dapat dilihat pada gambar 3.1 dibawah ini.

Gambar 3-1 Metodologi Pembuatan Program

4. DESAIN PROGRAM

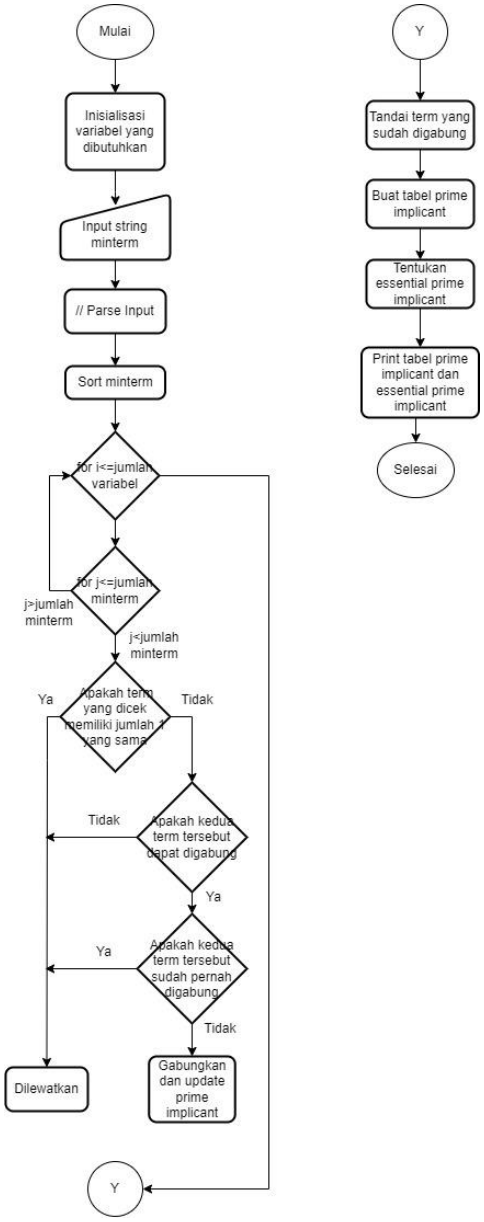
Dalam melakukan desain program kita perlu menentukan kriteria dan format input dan output program. Kriteria dan format ini berfungsi sebagai aturan dalam memasukkan input dan membaca output. Dari perancangan yag telah dilakukan, desain input dan output adalah sebagai berikut:

Input : String minterm (tanpa don't care)

Output : Essential prime implicant

Setelah menentukan desain input dan output, selanjutnya di rancang diagram alir (flowchart) program. Flowchart menggambarkan fungsi-fungsi program yang berkaitan dan aliran perintah pada program. Desain flowchart pada program ini ditunjukkan pada gambar 4.1 di bawah ini.

Flowchart Fungsi Utama :



Gambar 4.1 Flowchart fungsi utama

Pada flowchart fungsi utama terdapat fungsi-fungsi pendukung lainnya. Flowchart fungsi pendukung dapat dilihat di lampiran 1. Sedangkan nama dan deskripsi dari fungsi tersebut dijelaskan di bawah ini.

Fungsi List_Init() : digunakan untuk memasukkan minterm ke list.

Fungsi List_Insert() : digunakan untuk menyisipkan newElement ke dalam list

Fungsi list_merge() : digunakan untuk mengambil dua linked list dan menggabungkannya menjadi satu

Fungsi list_equal() : digunakan untuk membandingkan kedua list apakah isinya sama atau tidak

Fungsi list_print() : digunakan untuk mencetak seluruh nilai curr->id (index prime implicant)

Fungsi Max() : membandingkan nilai a dan b. Jika a > b, kembali ke nilai a. Jika a < b, kembali ke nilai b.

Fungsi Str_Reverse() : Ketika low < high, kondisi akan tetap. Ketika low > high, kondisi berubah low++, high- - .

Fungsi CompareMintermByRepr : digunakan untuk Membandingkan nilai minterm dengan nilai representasi, jika nilai cPosbit pertama sama dengan nilai kedua maka kembali ke fungsi strchamp(), jika tidak nilai tetap

Fungsi CompareMintermById : digunakan untuk membandingkan nilai minterm dan mengurutkan nilainya

Fungsi ParseInput() : digunakan untuk merubah minterm yang diinput menjadi bentuk binary-nya

Fungsi CanFormGroup() : digunakan untuk mengecek apakah kedua group yang dimasukkan fungsi dapat membentuk group atau tidak

Fungsi CreateNewGroupRepr() : digunakan untuk membentuk group representative baru dari dua group berbeda

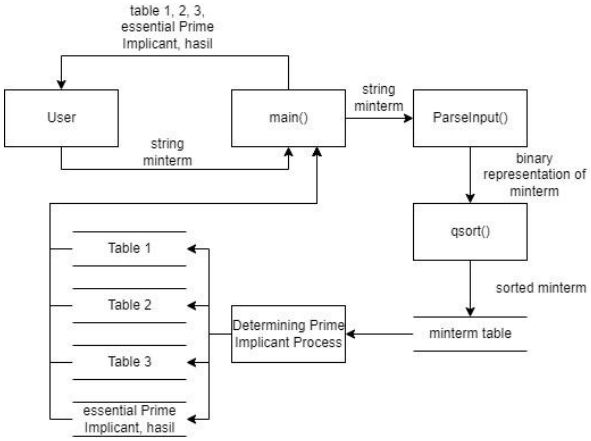
Fungsi GetPrimeImplicants() : digunakan untuk Mendapatkan array prime implicant dari tabel (array dari kolom)

Fungsi CreatePrimeChart() : digunakan untuk membuat array prime implicant dari nilai-nilai pada tabel (array dari kolom)

Fungsi GetEssentialPrimeImplicant() : digunakan untuk menyimpan implicant-implicant yang penting dan membuang yang tidak

Data Flow Diagram (DFD)

Data Flow Diagram menjelaskan aliran data masukan sampai data tersebut diproses dan menjadi keluaran. Data masukan berupa string minterm yang dimasukkan oleh user kemudian masuk ke main program dan selanjutnya string minterm dikirim ke fungsi Parseinput(). Pada fungsi Parseinput() string minterm diubah menjadi representasi bilangan biner. Setelah menjadi biner, data tersebut dikirim ke fungsi qsort() untuk diurutkan. Data biner yang telah diurutkan selanjutnya dikirim ke fungsi minterm table untuk disusun sedemikan rupa ke dalam table. Dari table tersebut akan ditentukan essential prime implicant nya dan disimpan ke dalam table table oleh fungsi Getessentialprime implicant(). Setelah menjadi tabel dan hasil essential prime implicant, data dikirim ke main function lagi untuk ditampilkan hasilnya. Data Flow Diagram dari penjelasan diatas dapat dilihat pada gambar 4.4 di bawah ini.



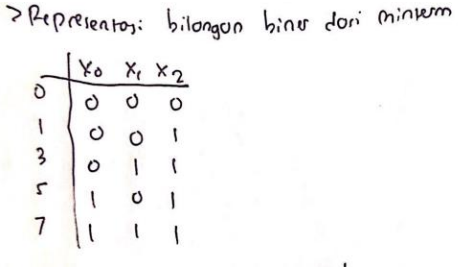
Gambar 4.2 Data Flow Diagram Program

5. HASIL DAN ANALISIS

Analisis dilakukan dengan menggunakan 2 masalah. Masalah pertama berupa masukan minterm dengan jumlah variabel 3 dan masalah kedua berupa minterm dengan jumlah variabel 4.

- Problem 1

Kita akan menggunakan algoritma Quine McCluskey untuk meminimisasi $f(x_0, x_1, x_2) = \sum(1, 3, 5, 7)$.



Pertama-tama kita mengubah bilangan minterm menjadi representasi binernya. Tampak bahwa bilangan biner tersebut sudah terurut berdasarkan jumlah "1" yang dimiliki sehingga kita tidak perlu mengurutkannya lagi.

> Tabel 1

	x_0	x_1	x_2	
0	0	0	0	✓
1	0	0	1	✓
3	0	1	1	✓
5	1	0	1	✓
7	1	1	1	✓

Selanjutnya berdasarkan tabel representasi bilangan biner dari minterm, kita harus mengelompokkan minterm-minterm yang memiliki perbedaan pada representasi binernya (melalui Tabel 1). Setiap bilangan yang sudah pernah mendapat kelompok, kita beri tanda centang. Pengelompokkan ini dapat dilihat pada Tabel 2.

> Tabel 2

	x_0	x_1	x_2	
0,1	0	0	–	(P_0)
1,3	0	–	1	✓
1,5	–	0	1	✓
3,7	–	1	1	✓
5,7	1	–	1	✓

Kita lakukan hal yang sama seperti pada Tabel 1 sebelumnya pada Tabel 2 untuk menghasilkan Tabel 3.

> Tabel 3

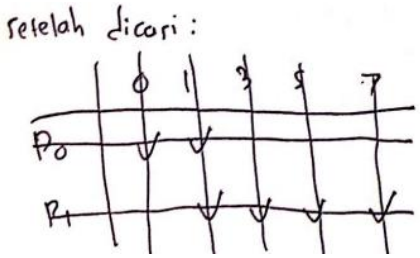
	x_0	x_1	x_2	
1,3,5,7	–	–	1	] sama (P_1)
1,5,3,7	–	–	1	

Setelah tidak ada kelompok minterm yang bisa dikelompokkan lagi, lihat pada Tabel 1 hingga Tabel 3 minterm/kelompok minterm mana yang belum dicentang. Minterm/kelompok minterm itulah yang merupakan *prime implicant*. Dari *prime implicant* ini akan dicari *essential prime implicant*-nya.

> Prime chart

	0	1	3	5	7
P_0	✓	✓			
P_1		✓	✓	✓	✓

Setelah dilakukan perhitungan:



didapatkan *essential prime implicant* : $P_0 + P_1$ dengan hasil akhir : $x_0'x_1' + x_2$.

Kita akan mengecek hasil yang didapatkan dengan menggunakan kode yang telah kami desain.

```
############### TABEL 1 ###############
[  0 ]  000  OK
[  1 ]  001  OK
[  3 ]  011  OK
[  5 ]  101  OK
[  7 ]  111  OK


############### TABEL 2 ###############
[  0  1 ]  00-   *
[  1  3 ]  0-1  OK
[  1  5 ]  -01  OK
[  3  7 ]  -11  OK
[  5  7 ]  1-1  OK


############### TABEL 3 ###############
[  1  3  5  7 ]  --1   *


############### PRIME IMPLICANT ########

[P 0]: 00-  [  0  1  ]
[P 1]: --1  [  1  3  5  7  ]
```

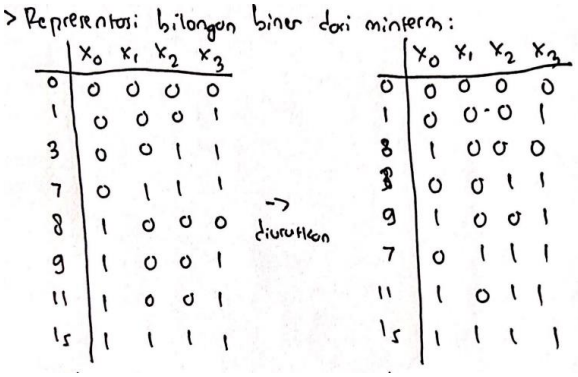
Tampak bahwa Tabel 1 hingga tabel *prime implicant* yang dihasilkan pada kode program sama dengan perhitungan tangan.

```
############### PRIME CHART ###############

        |  0|  1|  3|  5|  7|
 ---------------------------
[P 0]:  |  X|  X|   |   |   |
[P 1]:  |   |  X|  X|  X|  X|


############### ESSENTIAL PRIME IMPLICANT ####

[P 0]: 00-  [  0  1  ]
[P 1]: --1  [  1  3  5  7  ]

Hasil = x'[0]x'[1] + x[2]
```

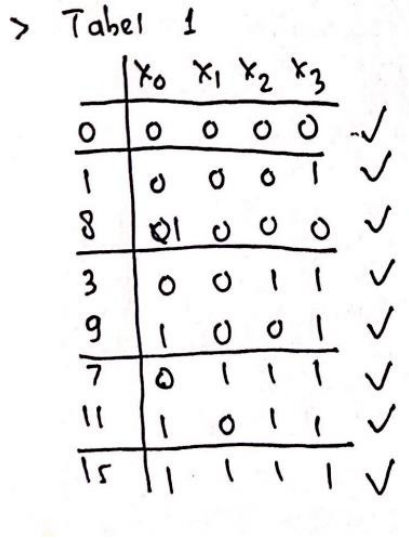
Tampak bahwa hasil akhir yang didapatkan sama dengan hasil perhitungan tangan.

➢ Problem 2

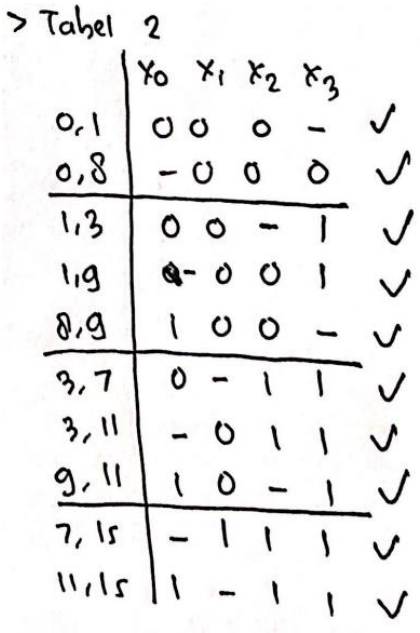
Kita akan menggunakan algoritma Quine McCluskey untuk meminimisasi $f(x_0, x_1, x_2, x_3) = \sum(0, 1, 3, 7, 8, 9, 11, 15)$.



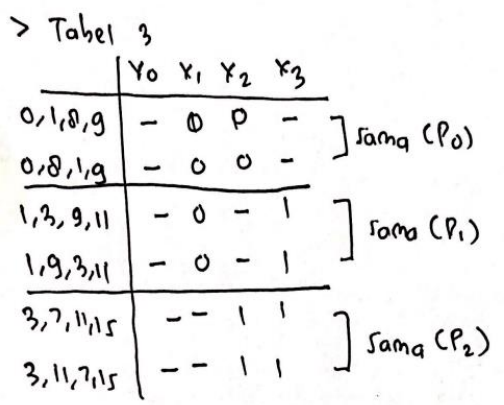
Pertama-tama kita mengubah bilangan minterm menjadi representasi binernya. Tampak bahwa bilangan biner 8 hanya memiliki 2 angka "1" dan bilangan biner 7 memiliki angka 3 angka "1". Oleh karena itu kita ubah posisinya berdasarkan jumlah angka "1" yang dimilikinya.



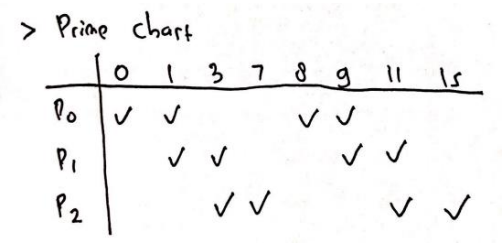
Selanjutnya berdasarkan tabel representasi bilangan biner dari minterm, kita harus mengelompokkan minterm-minterm yang memiliki perbedaan pada representasi binernya (melalui Tabel 1). Setiap bilangan yang sudah pernah mendapat kelompok, kita beri tanda centang. Pengelompokkan ini dapat dilihat pada Tabel 2.



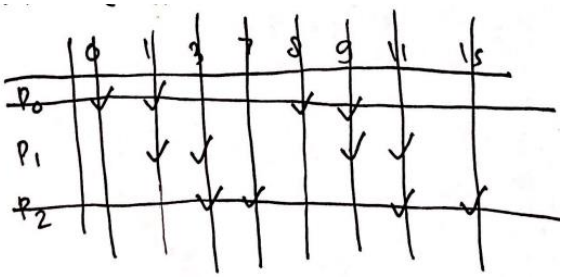
Kita lakukan hal yang sama seperti pada Tabel 1 sebelumnya pada Tabel 2 untuk menghasilkan Tabel 3.



Setelah tidak ada kelompok minterm yang bisa dikelompokkan lagi, lihat pada Tabel 1 hingga Tabel 3 minterm/kelompok minterm mana yang belum dicentang. Minterm/kelompok minterm itulah yang merupakan *prime implicant*. Dari *prime implicant* ini akan dicari *essential prime implicant*-nya.



Setelah dilakukan perhitungan:



didapatkan *essential prime implicant* : P_0 + P_2 dengan hasil akhir : $x_1'x_2'$ + x_2x_3.

Kita akan mengecek hasil yang didapatkan dengan menggunakan kode yang telah kami desain.

```
################ TABEL 1 ###############
[  0 ]  0000  OK
[  1 ]  0001  OK
[  8 ]  1000  OK
[  3 ]  0011  OK
[  9 ]  1001  OK
[  7 ]  0111  OK
[ 11 ]  1011  OK
[ 15 ]  1111  OK


################ TABEL 2 ###############
[  0  1 ]  000-  OK
[  0  8 ]  -000  OK
[  1  3 ]  00-1  OK
[  1  9 ]  -001  OK
[  8  9 ]  100-  OK
[  3  7 ]  0-11  OK
[  3 11 ]  -011  OK
[  9 11 ]  10-1  OK
[  7 15 ]  -111  OK
[ 11 15 ]  1-11  OK


################ TABEL 3 ###############
[  0  1  8  9 ]  -00-   *
[  1  3  9 11 ]  -0-1   *
[  3  7 11 15 ]  --11   *
```

Tampak bahwa Tabel 1 – 3 hasil simulasi sama dengan Tabel 1-3 hasil perhitungan tangan.

```
############## PRIME CHART ##############

       |  0|  1|  3|  7|  8|  9| 11| 15|
-----------------------------------------
[P 0]:  |  X|  X|   |   |  X|  X|   |   |
[P 1]:  |   |  X|  X|   |   |  X|  X|   |
[P 2]:  |   |   |  X|  X|   |   |  X|  X|


############### ESSENTIAL PRIME IMPLICANT #

[P 0]: -00-  [  0  1  8  9  ]
[P 2]: --11  [  3  7 11 15  ]

Hasil = x'[1]x'[2] + x[2]x[3]
```

Tampak bahwa hasil akhir yang didapatkan sama dengan hasil perhitungan tangan.

6. KESIMPULAN DAN LESSONS LEARNED

- Salah satu metode yang dapat digunakan untuk *logic minimization* adalah menggunakan Algoritma Quine-McCluskey atau biasa disebut dengan metode tabular.
- Algoritma QMC atau metode tabular ini menggunakan prinsip sebuah fungsi Boolean menjadi sebuah himpunan bentuk prima, dimana sebanyak mungkin peubah dieliminasi (dihilangkan) secara maksimal, hingga didapat fungsi Boolean yang paling sederhana. Ini dapat dilakukan dengan melakukan perulangan penggunaan hukum komplemen, $a + a' = 1$.
- Algoritma QMC dapat digunakan sebagai alternatif minimisasi logika selain K-map.
- Kelebihan dari penggunaan algoritma QMC ini adalah dapat dengan mudah diimplementasikan pada kode program, dapat menerima input yang banyak.
- Kekurangan dari penggunaan algoritma QMC ini adalah semakin banyaknya input maka semakin banyak tabel yang digunakan sehingga rentan terjadi kesalahan hitung (error), memiliki time and space complexity yang tinggi karena memerlukan banyak tabel dan perhitungan di dalamnya.

DAFTAR PUSTAKA

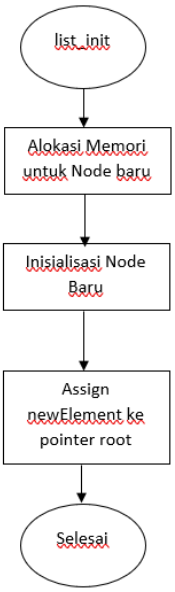
[1] Rimansyah, Rinaldi. 2012. Penyederhanaan Fungsi Boolean dengan metode Quine-McCluskey. *Skripsi*. Bandung: UNIKOM.

[2] https://en.wikipedia.org/wiki/Quine-McCluskey_algorithm , diakses tanggal 9 April 2022 jam 22.00 wib.

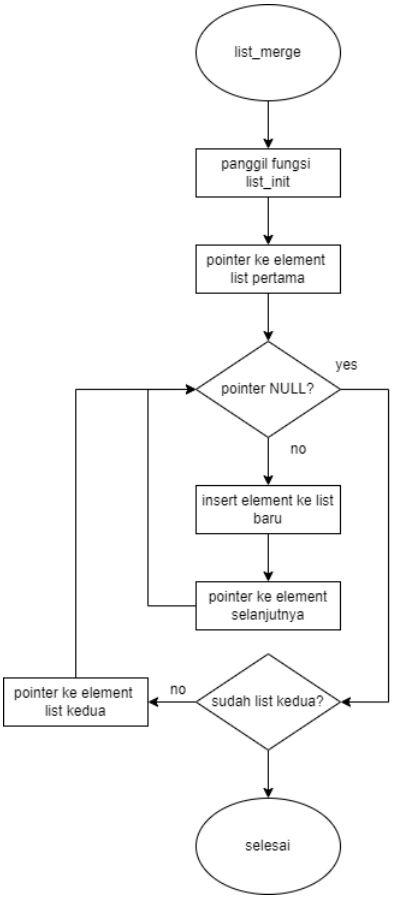
[3] https://www.youtube.com/watch?v=l1jgq0R5EwQ , diakses pada tanggal 12 April 2022 jam 20.00 wib

LAMPIRAN 1

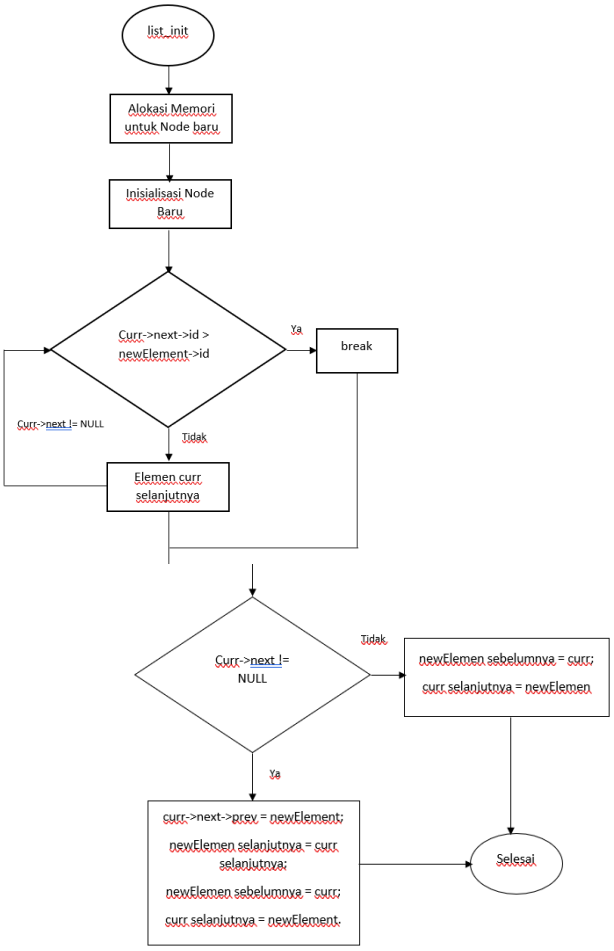
Fungsi List_Init()



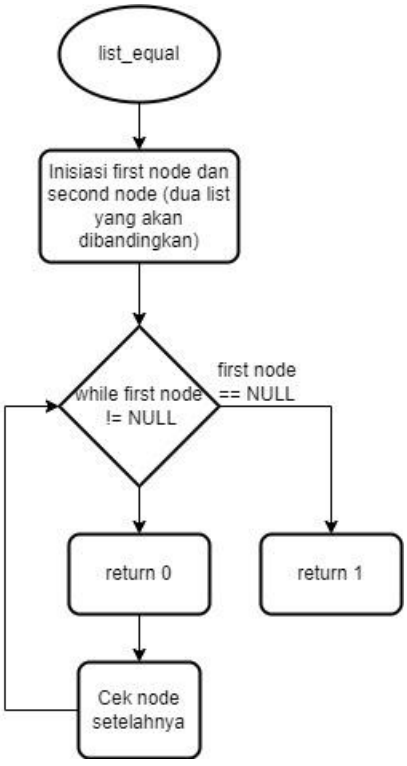
Fungsi list_merge()



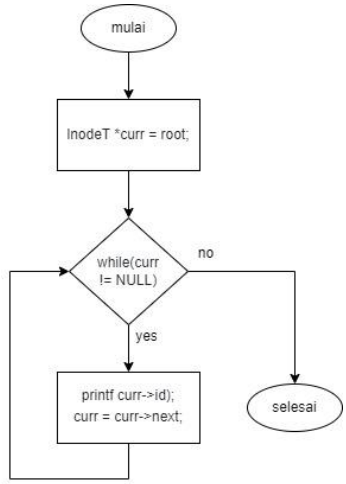
Fungsi List_Insert()



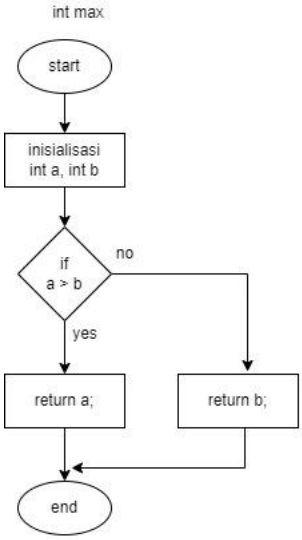
Fungsi list_equal()



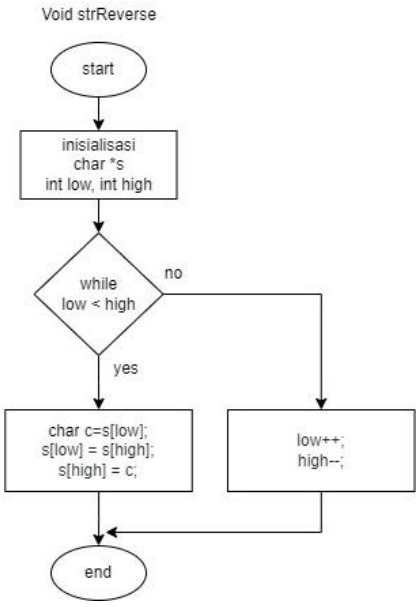
Fungsi list_print()



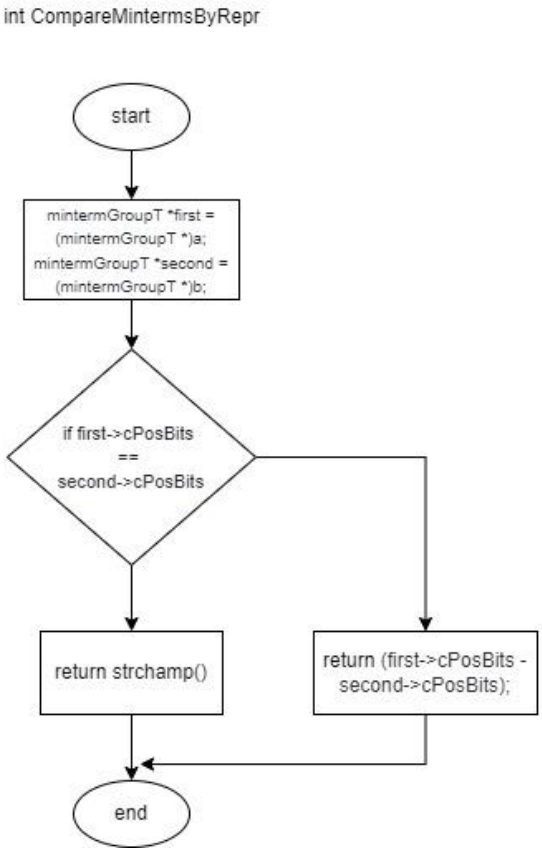
Fungsi Max()



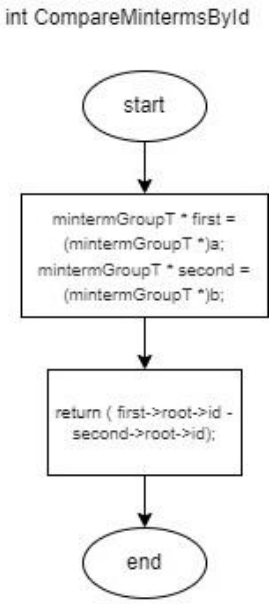
Fungsi Str_Reverse()



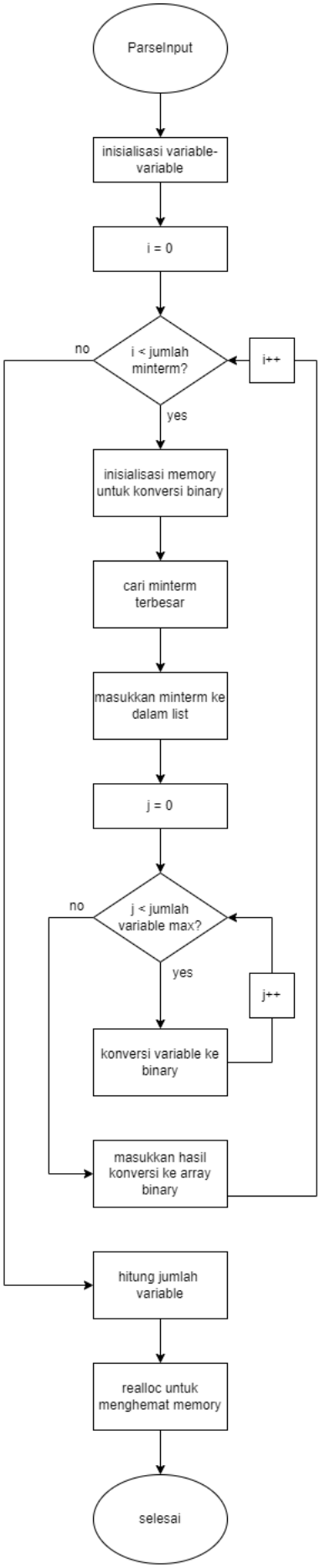
Fungsi CompareMintermByRepr()



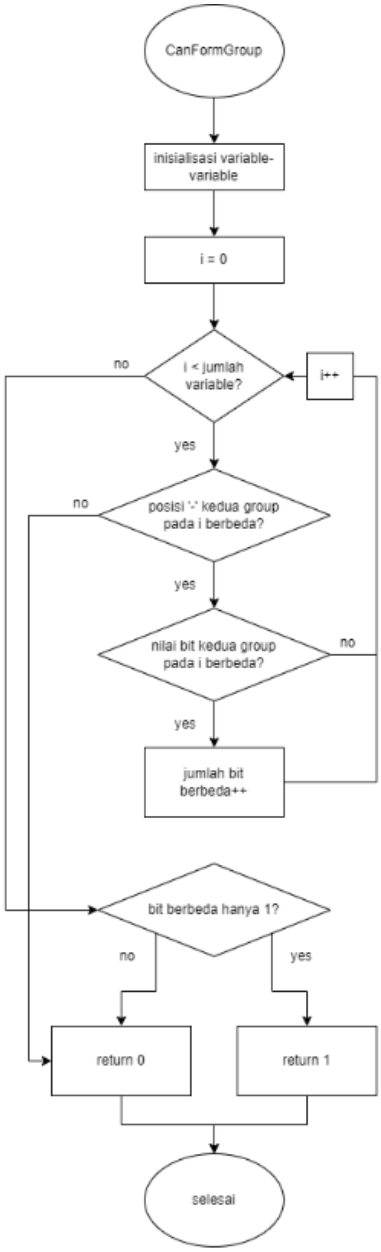
Fungsi CompareMintermById()



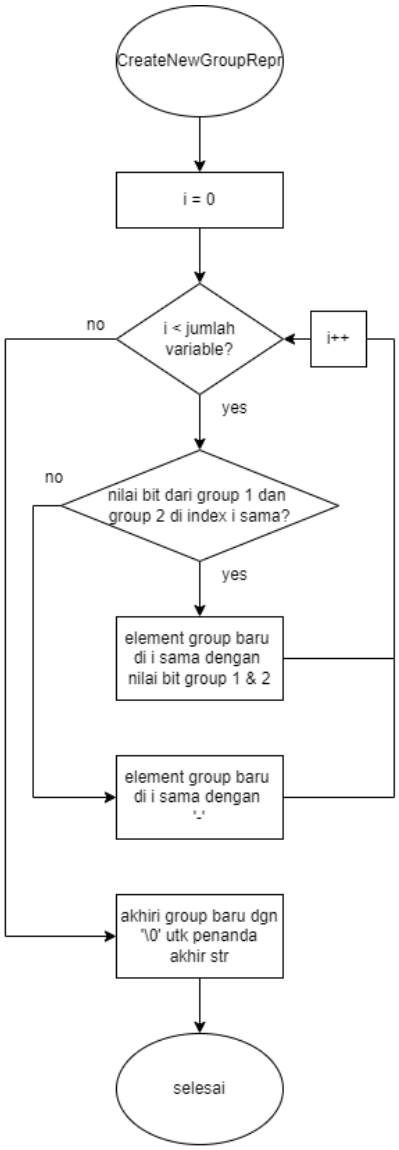
Fungsi ParseInput()



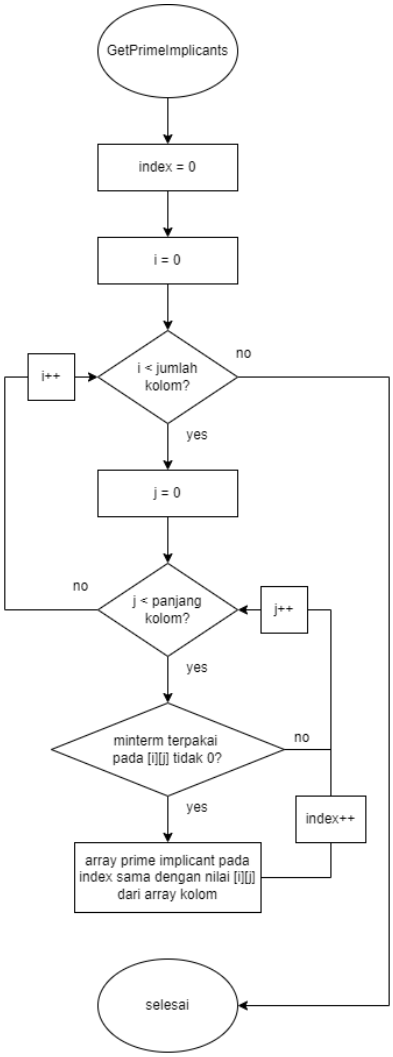
Fungsi CanFormGroup()



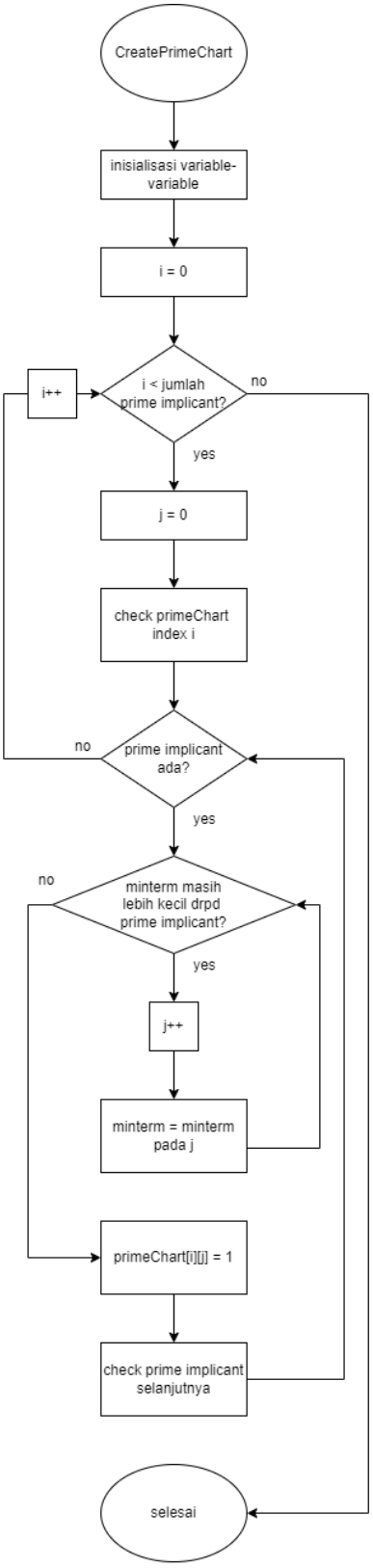
Fungsi CreateNewGroupRepr()



Fungsi GetPrimeImplicants()



Fungsi CreatePrimeChart()



Fungsi GetEssentialPrimeImplicant()