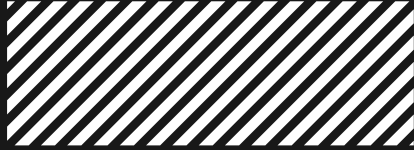


TUGAS BESAR



PEMECAHAN MASALAH DENGAN C

MINIMISASI LOGIKA

KELOMPOK 3

Fannan Bachtiar (18320021)

Christopher Chandra (18320033)

Febrian Vivaldi (18320035)



K-01

TABLE OF CONTENTS

01



Masalah

02



Studi Pustaka

03

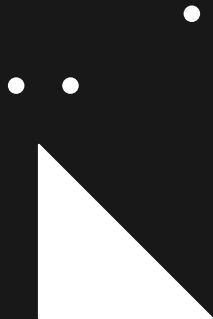


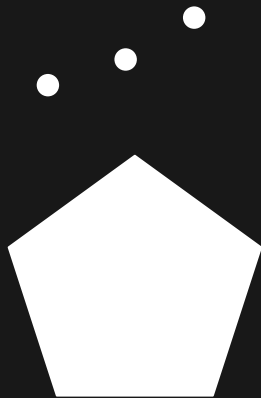
Program

04



Analisis Program



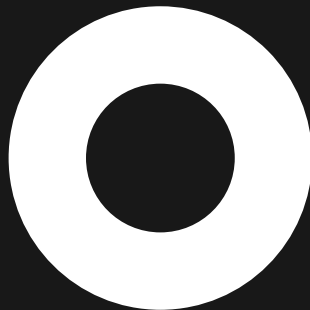
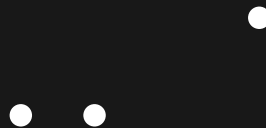


01

MASALAH

MINIMISASI LOGIKA +
PROGRAM DALAM BAHASA C

MINIMISASI???



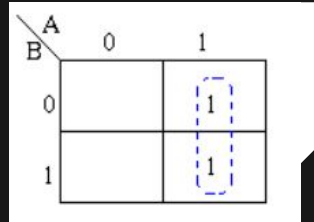
MINIMISASI

Dapat dilakukan dengan



$$\begin{aligned}F(A,B) &= \overline{A}\overline{B} + AB \\F(A,B) &= A(\overline{B} + B) \\F(A,B) &= A\end{aligned}$$

Aljabar Boolean



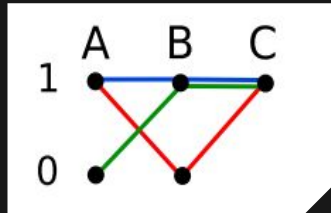
Karnaugh Map

- 1 (0 0 0 1) ditandai dengan A
- 0,2,8,10 (- 0 - 0) ditandai dengan B
- 10,11,14,15(1 - 1 -) ditandai dengan C

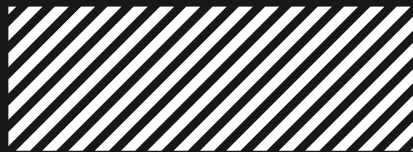
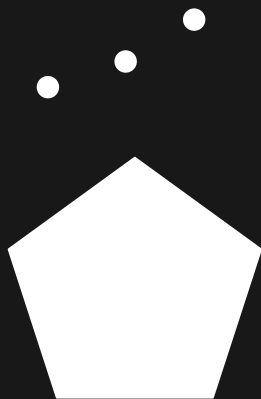
	0	1	2	8	10	11	14	15
A		X						
B	X		X	X	X			
C					X	X		
	√	√	√	√		√	√	√

Tanda O : berarti yang harus dipilih

Quine Mc-Cluskey



Truth Graph

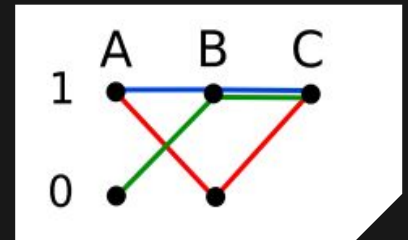


02

STUDI
PUSTAKA



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



TRUTH GRAPH

Step-by-step

BC


A

00 01 11 10

0	1	1	0	0
1	0	0	0	1

Truth Value

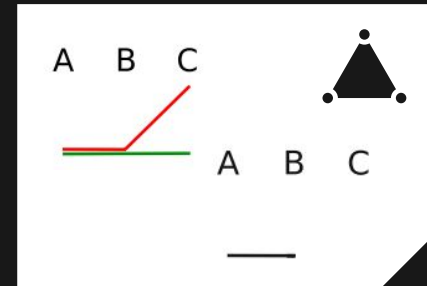
Memulai dengan membuat K-map yang berisi truth value



BC					
A		00	01	11	10
	0	—	—		
	1				—

Truth Graph

Kemudian dari truth value diubah menjadi truth graph

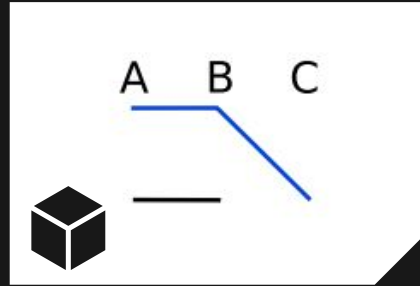


Eliminasi

Dilanjut dengan eliminasi dari truth graph dari baris pertama, didapatkan truth graph dengan garis hitam

TRUTH GRAPH

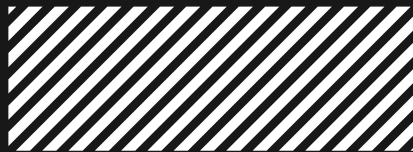
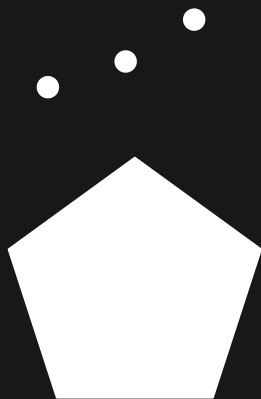
Step-by-step



Menambahkan Path

Ditambahkan path dari baris kedua. Dari graph akhir terlihat tidak ada nilai yang sama, maka hasil akhir $A'B' + ABC'$





03

PROGRAM

SPESIFIKASI



INPUT

Program menerima file
.txt dan menyimpan
minterm pada node stack

readFile()

operate()

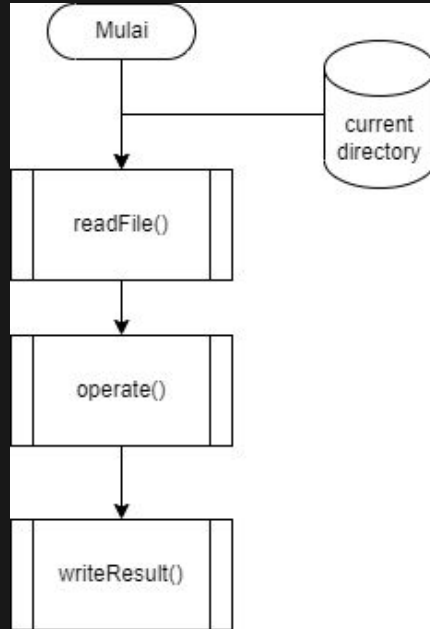
writeResult()

MAIN

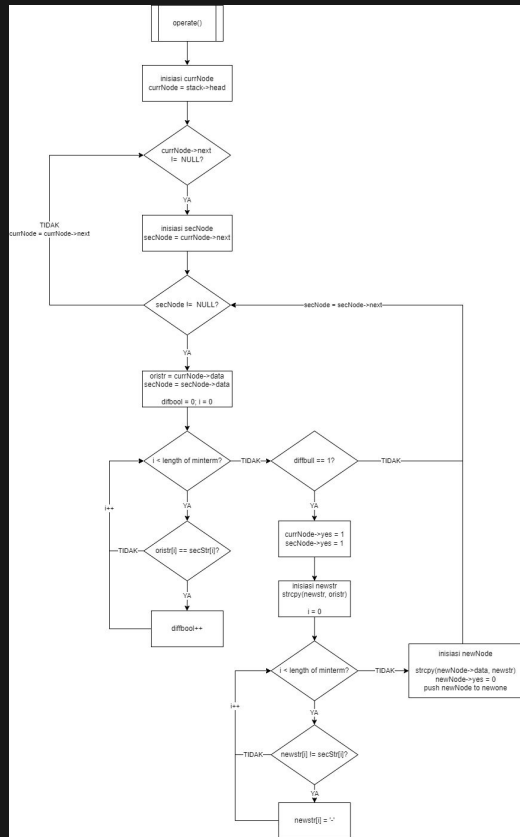
Menjalankan skema campuran
antara metoda T-graph dan Quine
McCluskey untuk memperoleh
simplifikasi dari minterm.

OUTPUT

Menuliskan tiap minterm hasil
simplifikasi yang diterjemahkan
dalam definisi aljabar boolean.



main()




operate()



```
int operate(  
    Stack *orione,        // koleksi path  
    Stack *newone,        // koleksi baru  
    int n)  
{  
    Node * currNode = orione->head;  
    while (currNode->next != NULL) {  
        Node * secNode = currNode->next;  
        while (secNode != NULL) {  
  
            char oristr[n];  
            strcpy(oristr, currNode->data);  
            char secStr[n];  
            strcpy(secStr, secNode->data);  
  
            int difbool = 0;  
            for (int i=0; i<n; i++) {  
                if (oristr[i] != secStr[i]) {  
                    difbool++;  
                }  
            }  
        }  
    }  
}
```

```
if (difbool == 1) {  
    currNode->yes = 1;  
    secNode->yes = 1;  
  
    char newstr[n];  
    strcpy(newstr, oristr);  
    for (int i=0; i<n; i++) {  
        if (newstr[i] != secStr[i]) {  
            newstr[i] = '-';  
        }  
    }  
    Node * newNode = (Node *) malloc(sizeof(Node));  
    strcpy(newNode->data, newstr);  
    newNode->yes = 0;  
    newNode->next = newone->head;  
    newone->head = newNode;  
}  
  
    secNode = secNode->next;  
}  
currNode = currNode->next;  
}
```

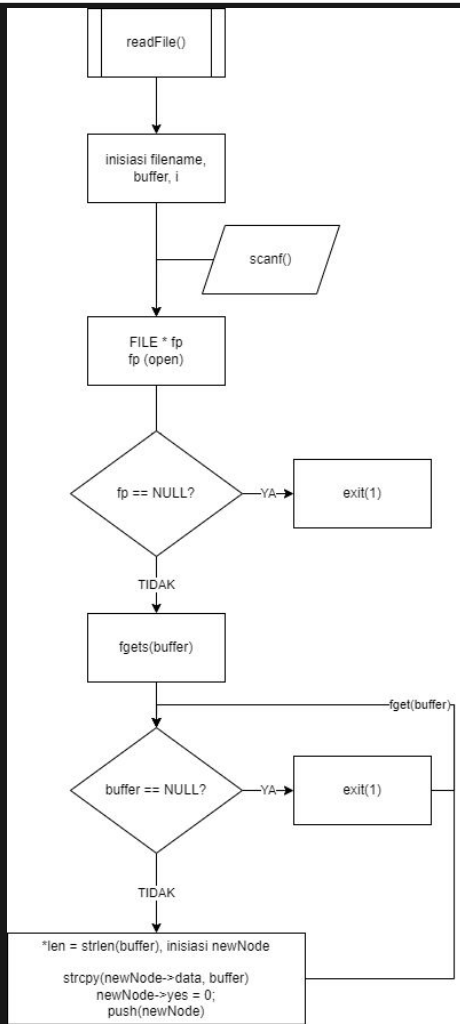


```
int recurse = 0;
while(checkNode != NULL) {
    if (checkNode->yes == 0) {
        Node * newNode = (Node *) malloc(sizeof(Node));
        strcpy(newNode->data, checkNode->data);
        newNode->yes = 0;
        newNode->next = newone->head;
        newone->head = newNode;

    }
    else {
        recurse = 1;
    }
    checkNode = checkNode->next;
}

Stack *emptystack = (Stack *) malloc (sizeof(Stack));
emptystack->head = NULL;

if (recurse) {
    operate(newone, emptystack, n);
}
```

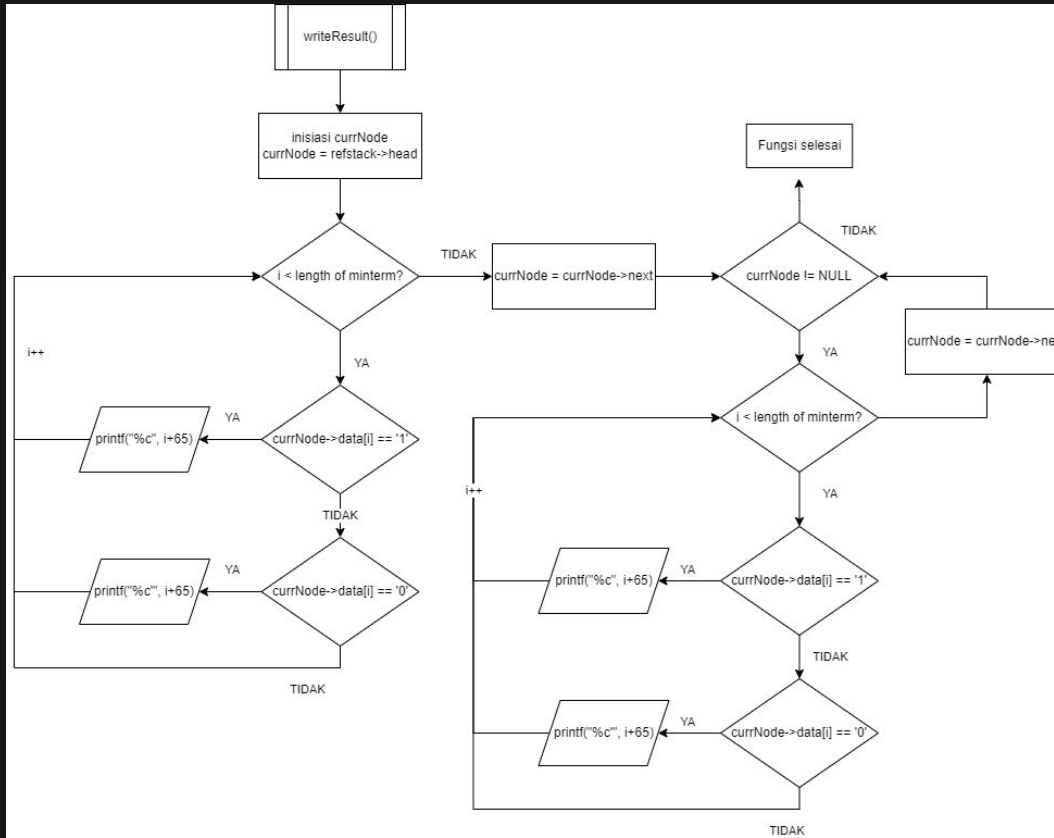


readFile()



```
void readFile(  
    Stack * astack,  
    int *len  
)  
  
    char filename[MAX_STR], buffer[MAX_STR], *token;  
    int i = 0;  
  
    //input nama file dan membukanya  
    printf("Masukkan sumber file: ");  
    scanf("%s", filename);  
    FILE *fp = fopen(filename, "r");  
  
    //beri output error jika file tidak ditemukan (filename: "file_tidak_ada.txt")  
    if (fp == NULL){  
        printf("\nError: file invalid!\n");  
        fclose(fp);  
        exit(1);  
    }
```

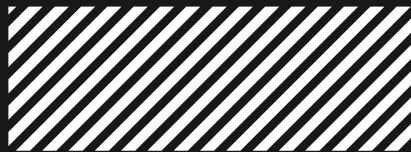
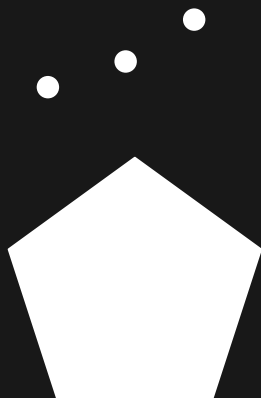
```
//mengambil dimensi matriks  
fgets (buffer, MAX_STR, fp);  
  
//beri output error jika file kosong (filename: "file_kosong.txt")  
if(buffer == NULL){  
    printf("\nError: file empty!\n");  
    fclose(fp);  
    exit(1);  
}  
  
*len = strlen(buffer);  
Node * newNode = (Node *) malloc((sizeof(Node)));  
strcpy(newNode->data, buffer);  
newNode->yes = 0;  
  
newNode->next = astack->head;  
astack->head = newNode;  
  
//mengambil data pada file .txt dan menyimpannya pada Stack  
while (fgets (buffer, MAX_STR, fp) != NULL){  
  
    Node * newNode = (Node *) malloc((sizeof(Node)));  
    strcpy(newNode->data, buffer);  
    newNode->yes = 0;  
  
    newNode->next = astack->head;  
    astack->head = newNode;  
}
```



writeResult()

```
void writeResult(Stack *refstack, int n)
{
    printf("\n");
    Node * currNode = refstack->head;
    for (int i=0; i<n; i++) {
        if (currNode->data[i] == '1') {
            printf("%c", i+65);
        }
        else if (currNode->data[i] == '0') {
            printf("%c'", i+65);
        }
    }
    currNode = currNode->next;
    while (currNode != NULL) {
        printf(" + ");
        for (int i=0; i<n; i++) {
            if (currNode->data[i] == '1') {
                printf("%c", i+65);
            }
            else if (currNode->data[i] == '0') {
                printf("%c'", i+65);
            }
        }
        currNode = currNode->next;
    }
    printf("\n");
}
```





04

ANALISIS PROGRAM

HASIL PROGRAM



Truth table (4input)

a	b	c	d	Output
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

0100
1010
1101
1111

```
Masukkan sumber file: testcase1.txt
```

```
A'BC'D' + AB'CD' + ABD
```

```
Time elapsed in readFile() ms: 4048.000000
```

```
Time elapsed in operate() ms: 0.000000
```

```
Time elapsed in writeResult() ms: 2.000000
```

```
Process returned 0 (0x0)   execution time : 4.115 s
```

```
Press any key to continue.
```

Output program

DNF (with overline) = $\overline{a}bcd + a\overline{b}cd + ab\overline{c}d + abcd$

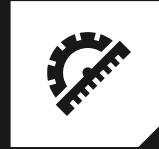
Hasil menggunakan web

ANALISIS KOMPLEKSITAS



Time Complexity

Dengan menganalisis kode, didapatkan kompleksitas waktu terburuk dari program adalah $O((2^n)!)!$



Space Complexity

Kasus terburuk dari masukan minterm adalah 2^n , yaitu semua minterm dari sejumlah n variabel. Oleh karena itu, kompleksitas ruang terburuk program adalah $S((2^n)!)!$.

TERIMAKASIH!

