

**Python Programming and Practice**

# **Boolean Function Simplifier Program**

**Proposal**

Date : 2023/11/03

Name : kimsanha

ID :230382

# 1. Introduction

## 1) Background

In my logic circuits course, I was working on a problem to simplify a Boolean function, and when my answer and the answer sheet were incorrect, I had a hard time figuring out where I went wrong. So to make it easier for me and my future students, I wrote a program to draw a k-map of the input Boolean function and simplify it using the sum of products (SOP) and the product of sums (POS).

## 2) Project goal

Write a program that draws a k-map for a given Boolean function and simplifies it using the sum of products (SOP) and product of sums (POS).

## 3) Differences from existing programs

Existing programs that simplify Boolean functions often don't know what rules they use or just give you the answer, but the program I'm going to build will draw and use K-maps to simplify, making it easy to understand visually.

# 2. Functional Requirement

## 1) Function 1 (Draw a K-map)

- Description (Ability to represent an input Boolean function as a K-map)

### (1) Detailed function 1 (Draw a table based on the number of variables)

- Description (If the number of variables entered is two, draw a table of the form  $2 \times 2$ , three  $4 \times 2$ , and four  $4 \times 4$ .)

### (2) Detailed function 2 (Populating a table 1)

- Description (In each column of the table, write down the term that the column represents. ( $m_0 = x'y'$ ))

### **(3) Detailed function 3** (Populating a table 2)

- Description (Display the values of the Boolean function you entered as 0s and 1s in the table you created in Detailed function 2.)

## **2) Function 2** (Using K-maps to simplify Boolean functions)

- Description (Output the result of simplifying the input Boolean function to (sum of products) sop and (sum of products) pos using K-maps, respectively.)

### **(1) Detailed function 1** (Group adjacent cells with the same value together.)

- Description (For example, if the value of the cell is 1, then the cells around it are bound in a rectangular shape. This should be as large as you can make it, with only 1s and even numbers across and down. The same goes for 0)

### **(2) Detailed function 2** (Make the enclosed result a Boolean function)

- Description (A set of 1's is represented as a sum of products by expressing one set as a product and adding them together, and a set of 0's is represented as a sum of products by expressing one set as a sum and adding them together, and (1) output with the K-map of Detailed function 3.)

## **3. Progress**

### **1) Implementing features**

#### **(1) Create a Karnaugh Map**

- Input: Takes in a number between 2 and 4 and outputs a Karnaugh Map with as many variables as the input number.
- Description: Generates a Karnaugh Map for a Boolean algebra with a number of variables between 2 and 4.

- Applied lessons: Conditional statements, loops, functions, modules.

-Code

screenshot

```
: 1 import pandas as pd
2 import itertools
3
4 def generate_gray_code(n):
5     """
6     n 비트에 대한 그레이 코드를 생성합니다.
7     """
8     gray_code = []
9     for i in range(2 ** n):
10         gray_code.append((i >> 1) ^ i)
11     return gray_code
12
13 def draw_karnaugh_map(variables):
14     """
15     주어진 변수의 개수에 대한 카르노 맵을 그립니다.
16     """
17     n = 2 ** variables
18     gray_code = generate_gray_code(variables)
19
20     # 카르노 맵을 나타내는 DataFrame 생성
21     karnaugh_map = pd.DataFrame(index=range(n), columns=range(variables))
22
23     # 각 칸에 해당하는 값 적기
24     for i in range(n):
25         for j in range(variables):
26             karnaugh_map.at[i, j] = gray_code[i] % 2
27             gray_code[i] //= 2
28
29     print("카르노 맵:")
30     print(karnaugh_map)
31
32     # 각 칸의 숫자와 이진수를 매핑하는 딕셔너리 생성
33     mapping_dict = {}
34     for i in range(n):
35         binary_value = ''.join(map(str, karnaugh_map.iloc[i].tolist()))
36         mapping_dict[i] = binary_value
37
38     print("#n카르노 맵 숫자와 이진수 매핑:")
39     print(mapping_dict)
40
41     # 사용자로부터 변수의 개수를 입력으로 받기
42     num_variables = int(input("변수의 개수를 입력하세요 (2에서 4까지): "))
43
44     if num_variables < 2 or num_variables > 4:
45         print("변수의 개수는 2에서 4까지의 값이어야 합니다.")
46     else:
47         draw_karnaugh_map(num_variables)
48
```

## (2) Optimize Boolean functions

- Input: Takes a Boolean function as input and outputs the optimized result

- Description: Optimize and output the input Boolean function using sum of products.

There are no errors, but it often produces incorrect results and needs to be improved.

- Applied lessons: Conditional statements, loops, functions, modules.

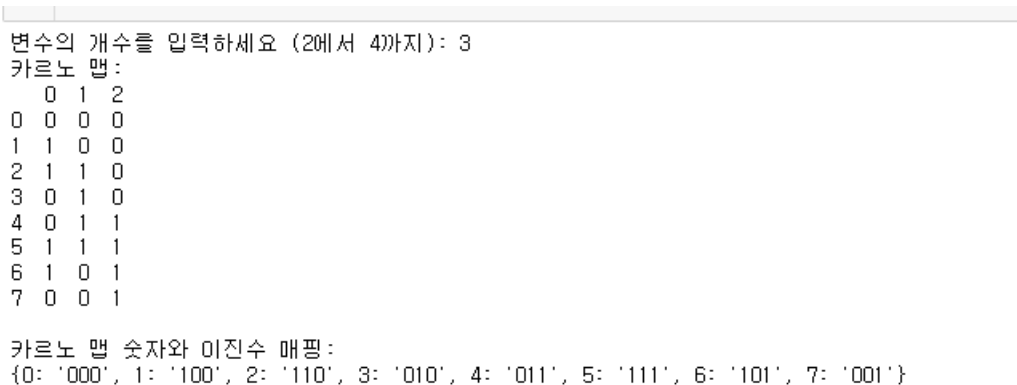
```
def optimize_boolean_function(karnaugh_map, mapping_dict, boolean_function):  
    """  
    카르노 맵을 이용하여 부울 함수를 최적화합니다.  
    """  
    num_rows, num_columns = karnaugh_map.shape  
    optimized_terms = []  
  
    for i in range(num_rows):  
        value = karnaugh_map.iloc[i].tolist()  
        if value.count(1) == 1 and boolean_function[i] == '1':  
            term = mapping_dict[i]  
            optimized_terms.append(term)
```

## 2) Test results

### (1) Create a Karnaugh Map

- Description: Generates a Karnaugh Map for a Boolean algebra with a number of variables between 2 and 4.

- Screenshot of test results



```
변수의 개수를 입력하세요 (2에서 4까지): 3  
카르노 맵:  
  0 1 2  
0 0 0 0  
1 1 0 0  
2 1 1 0  
3 0 1 0  
4 0 1 1  
5 1 1 1  
6 1 0 1  
7 0 0 1  
  
카르노 맵 숫자와 이진수 매핑:  
{0: '000', 1: '100', 2: '110', 3: '010', 4: '011', 5: '111', 6: '101', 7: '001'}
```

### (2) Optimize Boolean functions

- Description: Optimize and output the input Boolean function using sum of products. -  
Screenshot of test results

변수의 개수를 입력하세요 (2에서 4까지): 4

카르노 맵:

	0	1	2	3
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

카르노 맵 숫자와 이진수 매핑:

{0: '0000', 1: '0001', 2: '0011', 3: '0010', 4: '0110', 5: '0111', 6: '0101', 7: '0100', 8: '1100', 9: '1101', 10: '1111', 11: '1110', 12: '1010', 13: '1011', 14: '1001', 15: '1000'}

부울 함수를 입력하세요:  $x_1 + x_1 * x_2 + x_1 * x_3 + x_1 * x_4 + x_2$

최적화된 부울 함수: 0001

## 4. Changes in Comparison to the Plan

### 1) None

## 5. Schedule

