УКРАЇНСЬКИЙ КАТОЛИЦЬКИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНИХ НАУК

Комп’ютерні науки

Побудова досконалої кон’юнктивної форми булевої функції, заданої таблицею.

Автор: Володимир Савчук

31 травня 2020

1. Вступ.

Проєкт реалізовано на мові програмування Python. Для його реалізації мені звісно знадобилися базові знання з об’єктно орієнтованого програмування, знання з курсу Дискретної математики – Тема 4. Булеві функції.

Проєкт було реалізовано у двох модулях. Один із них - bool_func.py, є функціональним та реалізує абстрактний тип даних – Булева функція, де й містить окремий метод для побудови досконалої кон’юктивної нормальної форми. Другий(main.py) забезпечує взаємодію з користувачем та отримання вхідних даних, тим самим демонструючи роботу першого модуля, а саме, побудову досконалої кон’юктивної нормальної форми булевої функції від n змінних, заданої таблично.

2. Псевдокод.

1. Модуль **bool_func.py**

1) Функція **__init__ (var_num, func_value)** забезпечує створення нового елементу класу(булева функція), де var_num – це кількість змінних функції, func_value – це список значень функції на 2**n наборів.

```
"""
That module realizes ADT for boolean function.
"""
from itertools import product
from tabulate import tabulate


class BoolFunction:
    """
    Class represents the boolean function.
    """
    def __init__(self, var_num, func_value=None):
        """
        Initialization of BoolFunction.
        :param var_num: int
        :param func_value: list
        """
        self.var_num = var_num
        self.func_value = func_value
        self.variables_lst = ['x{}'.format(i + 1)
                              for i in range(var_num)]
        self.table = list()
        self.cnf = list()
        self.table_int = list()
        self.table_creation()
        self.table_to_cnf()
        self.zero_one_table_creation()

```

2) Функція **table_creation()** забезпечує створення таблиці у вигляді списку списків(де кожна комірка - це булеве значення True або False) для заданої функції.

```
def table_creation(self):
    """
    It creates the table representation
    of the boolean function with defined
    number of variables.
    :return: None
    """
    for i, val in enumerate(product([False, True], repeat=self.var_num)):
        val = list(val)
        val.append(bool(int(self.func_value[i])))
        self.table.append(val)
```

3) Функція zero_one_table_creation() забезпечує створення таблиці у вигляді списку списків(де кожна комірка - це string значення ‘0’ або ‘1’) для заданої функції. Вона створена як допоміжна для функції __str__().

```
def zero_one_table_creation(self):
    """
    It creates the table representation
    of the boolean function with defined
    number of variables.
    :return: None
    """
    for i, val in enumerate(product(['0', '1'], repeat=self.var_num)):
        val = list(val)
        val.append('1' if int(self.func_value[i]) else '0')
        self.table_int.append(val)
```

4) Функції __str__() забезпечує візуальну інтерпретацію булевої функції у вигляді таблиці.

```
def __str__(self):
    """
    String representation of the boolean function.
    :return: str
    """
    headers_lst = self.variables_lst
    headers_lst.append('F()')
    return tabulate(self.table_int, headers=headers_lst)
```

5) Функція table_to_cnf() будує безпосередньо доскноналу кон’юктивну форму.

```
def table_to_cnf(self):
    """
    It creates the  conjunctive normal form
    from the table as the representation
    of boolean function.
    :return: None
    """
    for i in self.table:
        if i[-1]:
            continue
        self.cnf.append([not value for value in i[:-1]])
```

6) Функція str_cnf() забезпечує візуальну інтерпретацію доскноналої кон’юктивної форми.

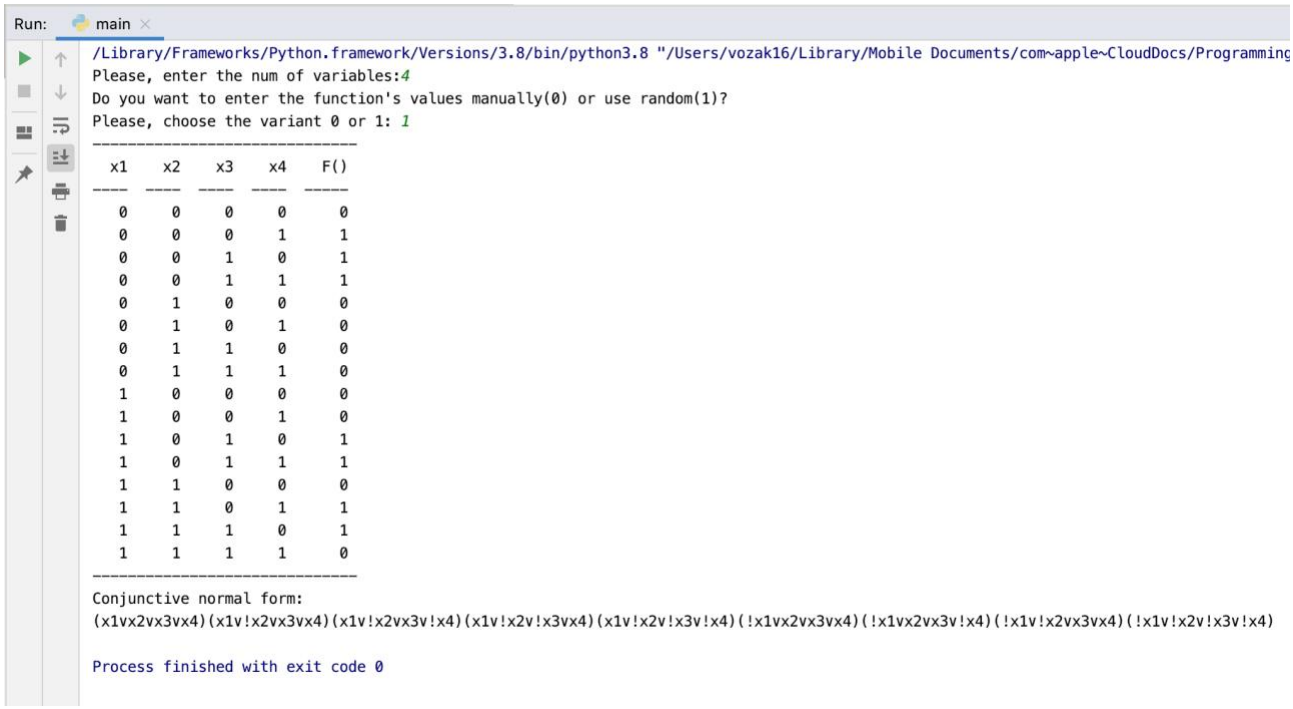
```
def str_cnf(self):
    """
    String representation of CNF.
    :return: str
    """
    result_str = ''
    for collection in self.cnf:
        temp_dnf_lst = [self.variables_lst[i] if collection[i] else '!'
                        + self.variables_lst[i] for i in range(len(collection))]
        temp_dnf_str = "v".join(temp_dnf_lst)
        result_str += '(' + temp_dnf_str + ')'
    return result_str
```

2. Модуль **main.py**

1) Функція main() забезпечує взаємодію з користувачем та отримання вхідних даних. Також тут забезпечено можливість для користувача обрати рандомне генерування списоку значень функції на 2**n наборів.

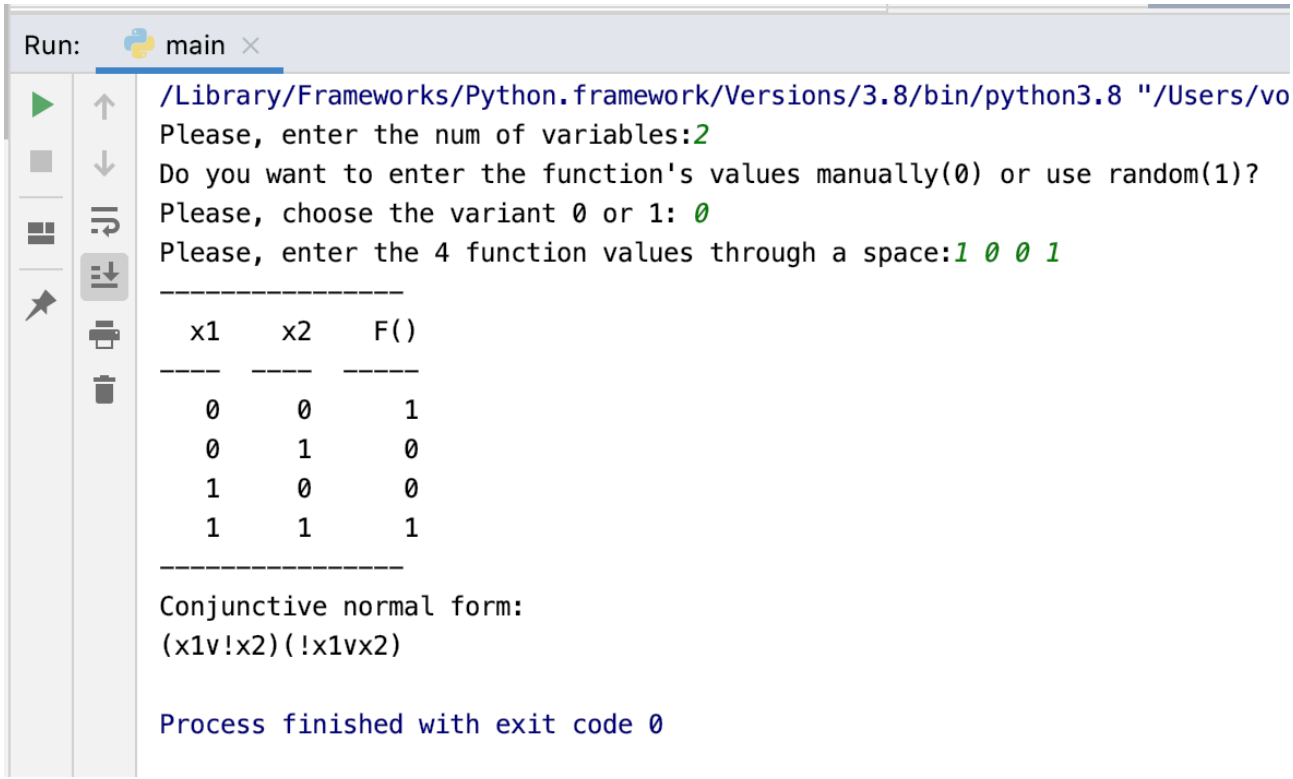
```
9
10  def main():
11      """
12      Runs the process of making CNF from the the table representation
13      of the boolean function.
14      :return: None
15      """
16      var_num = int(input("Please, enter the num of variables:"))
17      print("Do you want to enter the function's "
18            "values manually(0) or use random(1)?")
19      logs = int(input("Please, choose the variant 0 or 1: "))
20      if not logs:
21          func_value = input("Please, enter the {} function values through "
22                             "a space:".format(2 ** var_num)).split()
23      else:
24          func_value = [str(choice(('0', '1'))) for _ in range(2 ** var_num)]
25
26      bool_func = BoolFunction(var_num, func_value)
27      print(4 * '----')
28      print(bool_func)
29      print(4 * '----')
30      print("Conjunctive normal form:")
31      print(bool_func.str_cnf())
32
```

2) Приклад роботи програми:



```
Run:  main
/Library/Frameworks/Python.framework/Versions/3.8/bin/python3.8 "/Users/vozak16/Library/Mobile Documents/com~apple~CloudDocs/Programming
Please, enter the num of variables:4
Do you want to enter the function's values manually(0) or use random(1)?
Please, choose the variant 0 or 1: 1
------------------------------
  x1   x2   x3   x4   F()
----  ----  ----  ----  -----
   0    0    0    0     0
   0    0    0    1     1
   0    0    1    0     1
   0    0    1    1     1
   0    1    0    0     0
   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     1
   1    1    0    0     0
   1    1    0    1     1
   1    1    1    0     1
   1    1    1    1     0
------------------------------
Conjunctive normal form:
(x1vx2vx3vx4)(x1v!x2vx3vx4)(x1v!x2vx3v!x4)(x1v!x2v!x3vx4)(x1v!x2v!x3v!x4)(!x1vx2vx3vx4)(!x1vx2vx3v!x4)(!x1v!x2vx3vx4)(!x1v!x2v!x3v!x4)

Process finished with exit code 0
```



```
Run:  main
/Library/Frameworks/Python.framework/Versions/3.8/bin/python3.8 "/Users/vo
Please, enter the num of variables:2
Do you want to enter the function's values manually(0) or use random(1)?
Please, choose the variant 0 or 1: 0
Please, enter the 4 function values through a space:1 0 0 1
-----------------
  x1    x2    F()
----  ----  -----
   0     0     1
   0     1     0
   1     0     0
   1     1     1
-----------------
Conjunctive normal form:
(x1v!x2)(!x1vx2)

Process finished with exit code 0
```

3. Висновки.

Протягом роботи над проєктом, я поглибив й застосував на практиці свої знання з курсу Дискретної Математики, а саме, що таке досконала кон’юктивна нормальна форма та принцип її побудови. Також я мав можливість ще раз повправлятися у роботі з системою контролю версій Git, програмуванні на Python, проектуванні та реалізації абстрактних типів даних(булева функція), поєднав свої з дискретної математики та уміння програмувати.

Результатом роботи я задоволений, програма працює коректно в усіх випадках і написана досить якісно та елегантно.