EECS101001 Logic Design

Bonus Project

Deadline: 2024/04/25 23:59

1. Introduction

You are requested to implement a two-level logic optimization program. In this program, you have to reduce the literal count of given Boolean equations. Assume these equations are all written in the sum of product (SOP) form. You can solve this problem by using the Quine-McClusky approach you have learned in class. Also, any heuristic methods in your algorithm are acceptable.

2. Input file format (*.in*)

The first two lines list the total number of variables and the total number of product terms, respectively. The next n (n = total number of product terms) lines list the literals of this product term. Each of these n lines contains several characters, which represent the logic value (0, 1) of the variables in the product term. The number of the variables shown in the first line will be less than or equal to **12**. The variable letters will appear in a lexicographic order. For example, if the total number of variables is 6, the letters representing these variables would be ***a***, ***b***, ***c***, ***d***, ***e***, ***f***.

Sample input:

```
6                      // total number of variables = 6
8                      // total number of product terms = 8
101101                 // product term = ab’cde’f
101100
111111
101110
101111
111001
111000
011111
```

3. Output file format (*.out*)

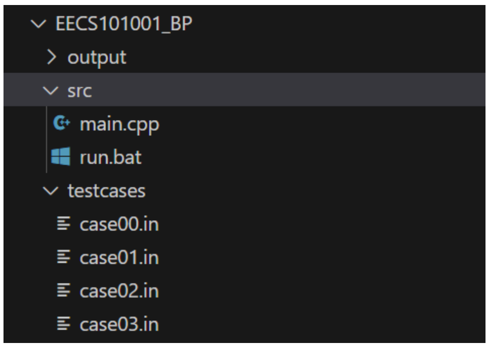
The first line lists the literal count of the optimized equation. The second line lists the total number of product terms remaining in the optimized equation. The next m (m = total number of remaining product terms) lines list all the remaining product terms after optimization. The hyphen (-) means the corresponding variable is a don’t care.

Sample output:

```
14                     // literal count = 14
3                      // total number of product terms = 3
11100-                 // optimized product term = abcd’e’
-11111
1011--
```

4. Implementation

A sample directory is provided with a file name, `EECS101001_BP.zip`. Please decompress it. Then a directory named `EECS101001_BP/` with a file structure like the following figure will be generated.



```
EECS101001_BP
  output
  src
    main.cpp
    run.bat
  testcases
    case00.in
    case01.in
    case02.in
    case03.in
```

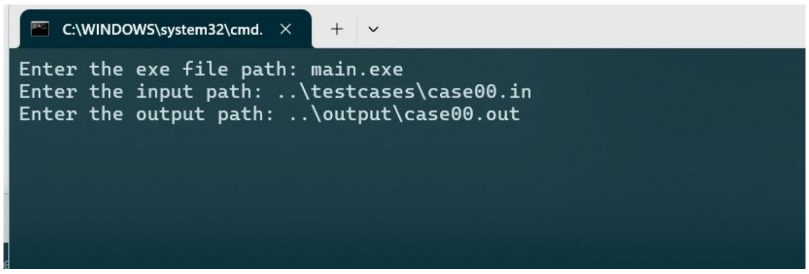
You have to write this program in C/C++. **Please ensure your source code can be compiled by g++ and follows the C++11 rules.** TAs will compile your source code and run all the testcases (10 hidden testcases) by the following command in Linux.

```
$ ./your_executable_file <.in file> <.out file>
```

(e.g., `$ ./bp case00.in case00.out`)

We provide the input and output functions in the main.cpp file for you. You can change it and design yours instead. But ensure to follow the same behavior as the original function. That is, to ensure that your program follows the format and accepts the corresponding arguments.

If you use Windows system and do not know how to use the command to read the file path, for your convenience, we include the **run.bat** file in the src folder. You can use it to run the program instead. All you need to do is double-click the .bat file after compiling your executable file. Your command line should show up and you can type the file path to execute your program.



```
Enter the exe file path: main.exe
Enter the input path: ..\testcases\case00.in
Enter the output path: ..\output\case00.out
```

We also provide a verification function for you to check your answer in source code file. Please make sure your result can pass it before you submit your code.

5. Submitted items

Two files are needed to be submitted to NTHU eeclass platform.

- `BP_{StudentID}.zip`
- `BP_{StudentID}.pdf`

Please compress `EECS101001_BP/` into one zip file with your **SOURCE Code in C/C++** under `src/`

A **REPORT** with a file name `BP_{StudentID}.pdf` is also needed, in which you should contain (1) your name and student ID (2) introduction of the data structure and the algorithm you used, and (3) other details of your implementation. The page limit of this **REPORT** is **5**.

6. Grading policies

Your program should be scalable for up to 12 input variables. The output result needs to be functionally equivalent to the function in the input file. Otherwise, you will get 0 points for this assignment. This assignment will be scored according to the quality of the results.

- ***0.2 points will be given if your literal counts are <= 50% of the input file literals for each case.***
- ***Additional 0.3 points will be given if all your literal counts are optimal for each case.***

We will test your program using 10 hidden testcases. **Total 5 points will be given** if you have done everything correctly.

Note that all the public testcases will not take into account of evaluation. Also, you must submit your report, or you will get a penalty of -1 point applied to the above grading policies.

The following information is the baseline and optimal literal count of each testcase. For each testcase, if your literal count is greater than the baseline literal count or ***TLE**, you will get 0 points. If you solve this problem using the Quine-McClusky approach, you should get the result of optimal literal count.

Testcase	case00.in	case01.in	case02.in	case03.in
50% baseline	8	24	36	80
Optimal	6	14	23	53

***TLE** (time limit exceeded): runtime longer than 15 minutes.

Other Notes

- If you have any questions, please post them on the Discussion board of NTHU eeclass instead of sending emails to TAs since there may be someone else having the same question.
- Your program should be single-threaded. Parallel computation with multiple threads or processes is **not allowed**.
- Plagiarism is not allowed. If you are caught, you will **FAIL this Logic Design course**, not just getting 0 points in this bonus project.