

Programming Assignment #1: 2-Way F-M Circuit Partitioning; due 5pm online, March 27, 2022 (Sunday)

ATTENTION: All programming submissions will be subject to duplication-checking with our database consisting of all submissions from previous and the current PD classes; those with $\geq 40\%$ similarity will be penalized. (In fact, most of earlier students with this problem eventually failed this course.)

Modified from Problem #3 of the 2001 IC/CAD Contest (Source: Faraday Technology Corp.)

1. Problem Description

Let $C = \{c_1, c_2, c_3, \dots, c_k\}$ be a set of k cells and $N = \{n_1, n_2, n_3, \dots, n_m\}$ be a set of m nets. Each net n_i connects a subset of the cells in C . The 2-way partitioning problem is to partition the set C of k cells into two *disjoint, balanced* groups, G_1 and G_2 , such that the overall cut size is minimized; in other words, no cell replication is allowed. The cut size s is given by the number of nets between G_1 and G_2 . For a given *balance degree* d , $0 < d < 1$ (with $d = 0$ being the most balanced and $d = 1$ the least), the objective of this assignment is to minimize s under the constraint

$$\frac{1-d}{2} \times k \leq \#(G_1), \#(G_2) \leq \frac{1+d}{2} \times k$$

by the Fiduccia-Mattheyses heuristic introduced in class.

2. Input

The input format and a sample input are given as follows:

Input Format	Sample Input
<Balance Degree> NET <Net Name> [<Cell Name>]+;	0.5 NET n1 c2 c3 c4 ; NET n2 c3 c6 ; NET n3 c3 c5 c6 ; NET n4 c1 c3 c5 c6 ; NET n5 c2 c4 ; NET n6 c4 c6 ; NET n7 c5 c6 ;

The input file starts with the balance degree d , followed by the description of m nets. The description of each net contains the keyword NET, followed by the net name and a list of the connected cells, and finally the symbol ‘;’. See the sample input for the format of a circuit with seven nets and six cells.

3. Output

In the program output, you are asked to give the cut size, the sizes of G_1 and G_2 , and the contents of G_1 and G_2 . The following table gives the output format and an output to the sample input. (Note that the solution may not be the optimal one.)

Output Format	Sample Output
Cutsizes = <Number> G1 <Size> [<Cell Name>]+; G2 <Size> [<Cell Name>]+;	Cutsizes = 5 G1 3 c1 c2 c3 ; G2 3 c4 c5 c6 ;

4. Provided Program Structure

We provide some sample data and program structures for your reference. Below list several related files which are available on the submission web page (the file named “sample codes.tgz”):

Makefile /bin /src /main.cpp: main program /cell.h: cell and node information /net.h: net information /partitioner.h: header file for the partitioner /partitioner.cpp: source code for the partitioner
--

To generate an executable binary, simply type “make” in the current directory. To regenerate a new executable binary, type “make clean” and then “make”. Please note that you can add, delete, and/or revise any part of the sample codes. If so, please remember to revise your makefile accordingly. You are also recommended to construct your own program structure because it might be more difficult to understand others’ programs than writing one from the scratch by yourself.

Command-line Parameter:

The executable binary must be named as “**fm**” and use the following command format:

```
./fm <input_file_name> <output_file_name>
```

For example, if you would like to run your binary for the input file `input_0.dat` and generate a solution named `output_0.dat`, the command is as follows:

```
./fm input_0.dat output_0.dat
```

Required Files:

You need to create a directory named `<student_id>_pa1/` (e.g. `f08943000_pa1/`) which must contain the following materials:

- A directory named **src/** containing your source codes: only `*.h`, `*.hpp`, `*.c`, `*.cpp` are allowed in `src/`, and no directories are allowed in `src/`;
- A directory named **bin/** containing your executable binary named **fm**;
- A makefile named **makefile** or **Makefile** that produces an executable binary from your source codes by simply typing “make”: the binary should be generated under the directory `<student_id>_pa1/`;
- A text readme file named **readme.txt** describing how to compile and run your program.
- A report named **report.pdf** on the data structures used in your program and your findings in this programming assignment.

Then please use the following command to compress your directory into a `.tgz` file:

```
tar zcvf <filename>.tgz <your directory>
```

The submission file should be named as `<student_id>_pa1.tgz` (e.g. `f08943000_pa1.tgz`). For example, if your student ID is `f08943000`, then you should use the command below:

```
tar zcvf f08943000_pa1.tgz f08943000_pa1/
```

Please submit your `.tgz` file to the NTU COOL system before **5pm, March 27, 2022 (Sunday)**. To

make sure that your submission satisfies all the requirements, we provide a script `checkSubmitPA1.sh` to check your `.tgz` file by the command below:

```
bash checkSubmitPA1.sh <your submission>
```

For example,

```
bash checkSubmitPA1.sh f08943000_pa1.tgz
```

Language/Platform:

- (a) Language: C or C++.
- (b) Platform: Linux. (For a fair evaluation of the submitted program, your developed programs will be tested on EDA Union servers. Submitted programs that fail to be executed on these servers by the TA will incur a significant penalty.)

5. Grading Policy

This programming assignment will be graded based on (1) **correctness**, (2) **solution quality**, (3) **running time**, (4) **readme.txt**, and (5) **report.pdf**. Please check these items before your submission. The runtime limit for each case is 60 minutes. For fair evaluation, please apply the **-O3** optimization for the compilation.

6. Online Resources

Sample input files (`input.dat`), `readme.txt`, `report.pdf`, and a checker to verify your program results can be found at the NTU COOL course link below: <https://cool.ntu.edu.tw/courses/12758/assignments/74460>.

For any questions, please email Shao-Yu at sylo@eda.ee.ntu.edu.tw. Thank you so much for your cooperation. Have fun with this programming assignment!