Project

In SoC design, dictionary-based test data compression is an important approach for reducing test data volume. This project requires you to select the entries in the dictionary for a circuit. You need to model the test set as a graph and apply heuristic procedure to solve a clique partitioning problem for the graph. You can read the paper for details. We will ONLY consider test sets provided.

Two words $u_1u_2 \cdots u_m$ and $v_1v_2 \cdots v_m$ are defined to be compatible (i.e. there is an edge) to each other if for any position i, u_i and v_i are either equal to each other or at least one of them is a don't-care bit:

W_1: 1010 1101
W_2: 101x 1x0x
W_3: 1100 0011
W_4: 1010 1x01

W_1 is compatible with W_2 and W_4, but not with W_3. Following this principle, we can get the graph below from test data.

Table I. An Example of Test Data for Multiple Scan Chains

Scan chain index	Word index															
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1	1	1	0	1	1	1	1	0	0	0	X	X	0	X	0	1
2	X	0	1	1	0	0	1	X	X	X	1	X	X	0	1	0
3	X	X	X	X	0	X	0	1	0	0	1	1	0	X	X	X
4	X	0	X	0	X	X	0	X	0	0	0	0	0	X	0	1
5	0	0	0	0	X	0	X	0	X	X	X	0	X	1	0	X
6	0	X	1	0	1	0	X	X	1	X	0	0	X	0	X	X
7	1	0	1	X	X	X	X	1	1	0	X	1	0	0	1	0
8	1	X	0	X	0	1	X	1	0	X	X	X	X	X	X	1

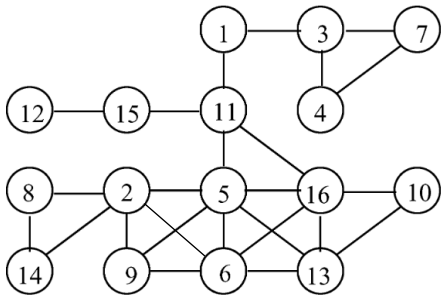

Fig. 4. The graph G for the example of Table I.

You can use the following heuristic procedure for partitioning:
(1) Copy the graph G to a temporary data structure G′.
(2) Find the vertex v with the maximum degree in G′.
(3) Establish a subgraph that consists of all the vertices connected to v. Copy this subgraph to G′ and add v to a set C. (The subgraph thus formed does not include the vertex v.)
(4) If G′ is not empty, go to Step (2). Otherwise, a clique C has been formed consisting of all the vertices found in Step (2).
(5) Remove the vertices in the clique C from G and copy G − C to G′. Go to Step 2 and repeat until |D| cliques are found.
Using the greedy algorithm or dictionary size |D| = 4, we obtain four cliques: {5, 6, 13, 16}, {2, 8, 14}, {3, 4, 7} and {1, 11}. (Here we use the word indices of Table I to represent the vertices.)

Your program should be in either C/C++. **Make sure your program runs on the departmental server. Download NoMachine and connect to engnx.utdallas.edu.**
Run **source /opt/rh/devtoolset-12/enable** to get the latest C++ version. You may need to log off and back on before your terminal recognizes this in some cases.
Your code MUST RUN in this server. If it doesn't, you will receive no credits. Please submit a zipped folder with your code and a Makefile. Please DO NOT include anything else. We DON'T need a README. We will compile your code by using "make". Your executable output MUST be named *dict*. We will run your code as:
./dict s15850f.test 4 32 s15850f.dict

where s15850f.test is the input, 4 is the number of dictionary entries (this can change), 32 is the length of each test vector (we will assume that the length can be either 8 bits, 16 bits, 32 bits or 64 bits) and s15850f.dict is the output file (containing the dictionary entries). For this project, you can assume all test data are 32 bits each. If the input file name is wrong, your program should output "**Wrong file name**" and terminate. The file should be available in the same directory your output is created. The command line should have 5 arguments exactly. Anything different, your program should output "**Incorrect number of arguments**" and terminate. If the number of dictionary entries requested (*e.g.,* 1000) is more than the number of unique cliques possible, your program should terminate normally with the dictionary stored in the *dict file, but output the following message "Only X dictionary entries are possible", where X is the number of unique cliques/dictionary entries.
Four test cases have been provided to you. We have also provided a sample output named sample.dict.

Please remember we will do a check of your code. Any similarity with your colleagues' codes or any open-source code, including GitHub, will receive ZERO credits.