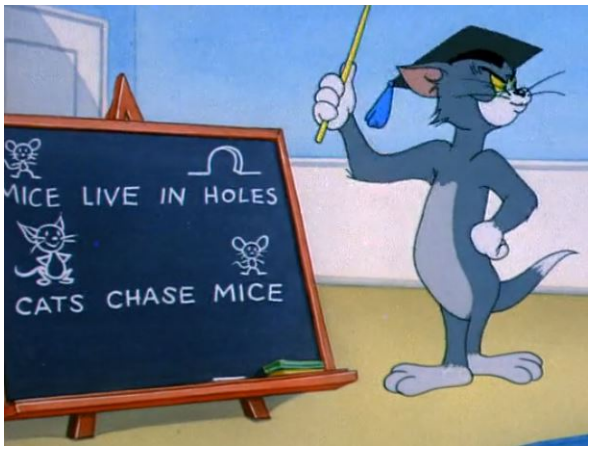
Lab 1

October 3, 2019



Deadline

Due: November 3, 2019, 23:59. Good luck!

Prerequisite

1. This lab should be implemented in Linux environment, such as debian and ubuntu. VM installed in ICS is OK.
2. Open a terminal and input commands below one by one to install necessary software:

   sudo apt-get install build-essential

   sudo apt-get install minisat

   sudo apt-get install zlib1g

   sudo apt-get install zlib1g-dev
3. Source codes are under lab1 directory. You should write codes in lab1.cpp. After writing the codes, you can open a terminal in the directory and input "make run", then the files will be compiled and executed. Detailed information are in lab1/README.txt.
4. You may want to install VS code in Linux to edit the files.

Problem

There are n stones in the river. Every stone has two states: above and below the water. There are m switches. Every switch has two states: open and close. These switches can control the state of stones. Every switch can control one or two stones. Every stone is controlled by one or two switches. Every time you change the state of a switch, the states of stones which are controlled by the switch will change.

Now every switch is close. Some stones are above the water and the others are below the water. Determine how to set the states of switches to make every stone above the water with SAT solver Minisat.

Input

Input data are in test.txt.

The first line inludes a number iter, which is the number of test cases. Then there are iter groups of test cases. Every two cases are separated with a blank line.

For every test case, the first line includes two numbers m and n. m is the number of switches and n is the number of stones. The next line includes n numbers, which means the initial states of n stones. 0 means the stone is below the water and 1 means the stone is above the water. Then there are m lines, whcih are the control information of m swithes. The first number of every line means the number of stones controlled by the switch. The following numbers means the indexes of stones controlled by the switch. Th index of stone starts from 1 instead of 0.

Output

The program will output your answer to answer.txt. If there are iter test cases, there will be iter lines in answer.txt. Every line includes some 0 and 1. 0 means the switch shoud be close and 1 means the switch should be open. So all stones will be above the water. Or some line will include "UNSAT", which means it's impossible to make all stones above the water.

Example

For example, test.txt is

1

2 2

1 0

1 1

1 2

This means there is 1 test case. there are 2 switches and 2 stones. Initially, stone No. 1 is above the water and stone No. 2 is below the water. The first switch control stone No. 1 and the second switch control stone No. 2. So after running your program, the answer.txt should be

0 1

It means that the first switch should be close and the second switch should be open.

Framework

We have implemented the code framework in main.cpp and lab1.cpp, including reading test.txt and output your answer to answer.txt. All you need to do is to implement the function called lab1 in lab1.cpp. You can write some other functions invoked by lab1. But You can only modify lab1.cpp. You must not change the other files or create new files.

lab1 function has 5 arguments.

- n is the number of stones.
- m is the number of switches.
- The array called states is the initial states of n stones. Its length may be larger than n. Ignore the extra part in the array.
- Button is a 2-dimension array. The array records the control information. Every item in the array denotes the index of stones. For example, button[3][0] and button[3][1] means the indexes of stones controlled by the fourth switch. The index starts from 1 instead of 0. If button[3][0] = 1 and button[3][1] = 0, it means the button only control stone No. 1. The array may be larger than necessary. Ignore the extra part.
- Array answer records your answer to the problem. For example, answer[2] = true means the third switch should be open. The array may be larger than necessary. Ignore the extra part.

- The return value of lab1 function is bool type. If it returns true, it means there is a solution for the problem. If it returns false, it means you cannot make every stone above the water.

We have given some codes in lab1 function. You can change them if you want. But your implementation should be consist with description above.

You must use Minisat to implement lab1. There is an example in minisat.cpp. The program finds an assignment for $(\neg A \vee \neg B \vee C) \wedge (\neg A \vee \neg B \vee \neg C) \wedge (A \vee \neg B \vee C)$. You can input command "make example" in terminal under lab1 directory to compile and execute the example program. Then you will see the answer of the SAT problem.

Don't use IO operations in your codes, including scanf, printf and all file operations.

Submission

Please submit a zip package to JBox https://jbox.sjtu.edu.cn/l/g1ydsL. The name of package should be like "123456张三.zip". The zip should include only lab1.cpp.It should not incude any directories. Don't put lab1.cpp under some directory.

If you want to change your submission in JBox, you should submit a new zip, whose name is like "123456张三-2.zip".

We will use automatic tools to unzip the package, test your codes and record the grade. If you violate the requirements and cause some bad or wrong results, we will reduce your points.