CS325: Analysis of Algorithms, Fall 2017

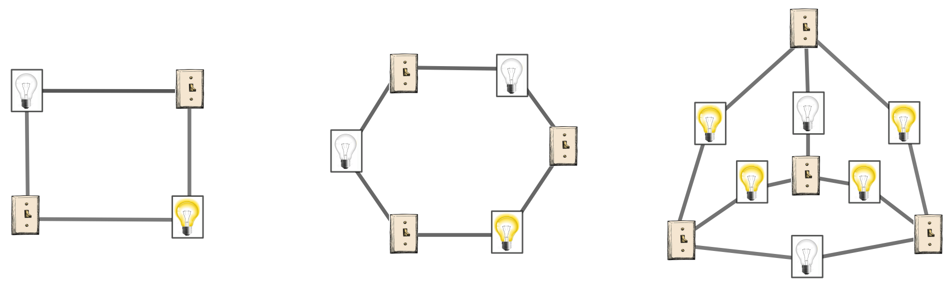
Group Assignment 4

Due: Thr, 11/30/17

Homework Policy:

1. Students should work on group assignments in groups of preferably three people. Each group submits to TEACH a zip file that includes their source code and their *typeset* report. Each group, also, hands in a printed hard copy of the report in class or slides the hard copy under my door before the midnight of the due day. The hard copy will be graded, and the codes submitted to teach will be tested.
2. The goal of the homework assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.
3. You are allowed to discuss the problems with other groups, and you are allowed to use other resources, but you *must* cite them. Also, you must write everything in your own words, copying verbatim is plagiarism.
4. *I don't know policy:* you may write "I don't know" *and nothing else* to answer a question and receive 25 percent of the total points for that problem whereas a completely wrong answer will receive zero.
5. Algorithms should be explained in plain english. You can use pseudocodes if it helps your explanation, but the grader will not try to understand a complicated pseudocode.
6. More items might be added to this list. ☺

Lights and switches. As the quarter ends and before we leave for the Winter break, someone needs to turn off all campus lights. Unfortunately though, we recently learned that the wiring map of OSU campus is a little bit strange. Each light of the campus is connected to exactly two switches and each switch is connected to multiple lights. Toggling a switch changes the on/off state of every light that is connected to the switch. The challenge is to find a set of switches to toggle to turn off all campus lights. The good news is we have a complete map of the wirings between lights and the switches. Here are smaller examples of light-switch maps from a smaller college (OSU's is much more complicated). It is not possible to turn off all lights in the first two examples. How about the third one?



2-SAT. Upon the advice of a computer scientist, OSU has recently purchased a blackbox 2-SAT solver to solve the lights and switched problem. Given a 2-CNF formula (i.e. conjunctions of clauses, where each clause is a disjunction of exactly two literals), this blackbox outputs "yes" if the formula is satisfiable, and "no" otherwise. For example, given

$$(x_1 \vee x_2) \wedge (x_2 \vee \neg x_1)$$

the blackbox returns "yes", and given

$$(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \wedge x_2) \wedge (\neg x_1 \wedge \neg x_2)$$

the blackbox returns "no".

The computer scientist more recently noticed that using this blackbox for the lights and switches problem might not be as straightforward as he thought. So, he would like to ask the students in the algorithm class to come up with a polynomial time reduction from the lights and switches problem to the 2-SAT problem.

Report (60%). In your report, describe a polynomial time reduction from the lights and switches problem to 2-SAT. Analyze the time complexity of your reduction and prove that it is correct.

Code (40%). Write a program that solves the lights and switches problem assuming an available blackbox for 2-SAT. We will post codes that solve the 2-SAT problem on the webpage of the course. Your program can use these codes to solve the lights and switches problem. Your program will be tested against several test cases, for correctness and efficiency. For each test case, the program will be automatically stopped after 20 seconds if it is not done in that time. In this case, the group will miss the points of that test case. Your program must be written in one of the following languages: Python, C++, or Java.

Input/Output Your program reads from one text file, "input.txt". The first line contains two integers $1 \leq n, m \leq 1000$ separated by comma, where n is the number of switches and m is the number of lights. Next line contains a list of m binary values specifying the initial state of the lights, 1 is on, and 0 is off. The following n lines specify the set of lights connected to each switch. Specifically, line i contains a list of all lights connected to the ith switch.

Your output must be written in "output.txt". The output is exactly one word for each test case: "yes" of it is possible to turn off all lights, and "no" otherwise.

Sample Input (1):

```
2,2
0,1
1,2
1,2
```

Sample Output (1):

```
no
```

Sample Input (2):

```
3,,3
0,0,1
1,2
2,3
1,3
```

Sample Output (2):

```
no
```

Sample Input (2):

```
4,6
1,0,1,1,1,0
1,2,3
1,4,6
3,5,6
2,4,5
```

Sample Output (2):

```
yes
```

Submission Each group submits to TEACH a zip file that includes their source code (*which must be just one file with name "lights.cpp", "lights.java", or "lights.py"*) and their report in pdf format. This file can be submitted by any member of the group, but all names must be listed in the submitted report. Each group, also, hands in a printed hard copy of the report in class or slides the hard copy under the door of my office before the midnight of the due day. The hard copy will be graded, and the submitted code to teach will be tested.

Your codes will be tested *automatically*. So, you need to carefully follow all formatting requirements mentioned above. To summarize:

(1) Your source code file should be named "lights.cpp", "lights.java", or "lights.py".

(2) It reads from files "input.txt" in the current folder.

(3) It writes to the file "output.txt" in the current folder. It should write exactly one word into "output.txt", "yes" or "no", with no extra symbol.

Test your code with the sample test files (`http://web.engr.oregonstate.edu/~nayyeria/CS325/Fall17/hws/test4.zip`) before submitting them, to make sure there is no formatting error.