CS325: Analysis of Algorithms, Winter 2024

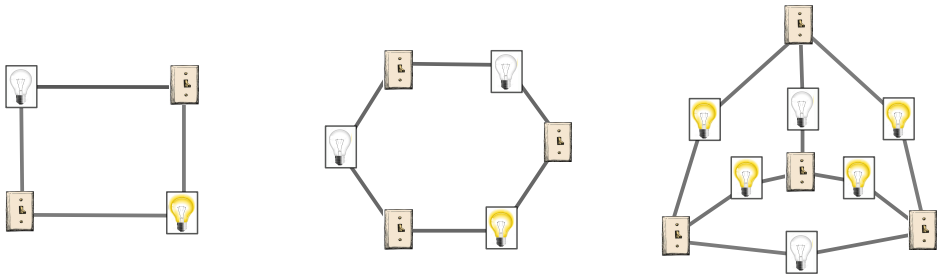
Group Assignment 4

Due: Thur, 3/14/24

Homework Policy:

1. Students should work on group assignments in groups of preferably three people. Each group submits to CANVAS a zip file that includes their source code and their *typeset* report. The report must include the name of all group members. Specifically, for this assignment your zipped folder should contain two files named `assignment4.pdf`, and `assignment4.py`. One submission from each group is sufficient.
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.

We are getting close to the break, and all of us are ready to go home after an eventful Winter in Corvallis. We are almost done with the quarter, and the final exam is not that far. Yet, there will be one more task that needs to be done before leaving. We need to turn off all the lights of the campus. Unfortunately, we learned that this task is a bit more complicated that we thought because each light is connected to multiple switches and each switch to multiple lights. Toggling a switch alters the state of every light that is connected to that switch. So, the challenge is to find a set of switches to toggle to turn all the lights off. The good news is we just found an old map of the wiring of the campus. So, we know which switch is connected to which light. Further, we observed that each light is connected to exactly two switches, while each switch might be connected to multiple lights. Here are a few examples of what this wiring and the initial state of the lights can look like.



2-SAT. Next to the old wiring map of the building we founds a blackbox that can solve the 2-SAT problem efficiently! 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

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

the blackbox returns "no".

Since the blackbox was next to the map, we suspect that it can help us turn off all the lights. Hence, we are after a polynomial time reduction from our problem of lights and switches to 2-SAT.

- Describe your reduction and show that it takes polynomial time.
- Prove that your reduction is correct. The 2-SAT problem that you obtain is satisfiable if and only if it is possible to turn off all the lights.

Report (60%). In your report, include the description of your reduction, its running time analysis and proof of correctness. Your reduction should have polynomial time complexity. Also, you need to prove that the 2-SAT instance that you build is satisfiable if and only if there is a way to turn off all the lights.

Code (40%). Submit a python program that decides if it is possible to turn off the lights. Your program will be tested against several test cases, for correctness and efficiency. For each test case, the program will be automatically stopped after 30 seconds if it is not done in that time. In this case, the group will miss the points of that test case.

Note: it is important that your output is formatted as described below, since your codes will be tested automatically. Specifically, you must implement the function "can_turn_off_lights" in the following code. The code you submit will be an implementation of this procedure in a file named "assignment4.py".

```
'''
    This file contains the template for Assignment4.  For testing it, I will place it
    in a different directory, call the function <can_turn_off_lights>,
    and check its output. So, you can add/remove  whatever you want to/from this file.  But,
    don't change the name of the file or the name/signature of the following function.

    Also, I will use <python3> to run this code.
'''

def can_turn_off_lights(input_file_path, output_file_path):
    '''
        This function will contain your code.  It wil read from the file <input_file_path>,
        and will write its output to the file <output_file_path>.
    '''
    pass

'''
    To test your function, you can uncomment the following command with the the input/output
    files paths that you want to read from/write to.
'''
# can_turn_off_lights('', '')
```

You can use available codes on the internet for solving the 2-Sat problem. Here is an example that I found easy to use: `https://github.com/arunptl100/SAT-Solver/blob/master/sat-solver.py`.

Input/Output Each input contains two instances of the switches and lights problem. That appear one after the other. Each instance starts with a line that contains '***'. The first line of each instance 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. The second line of each instance 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 of each instance specify the set of lights connected to each switch. Specifically, line i contains a list of all lights connected to the ith switch.

The output file must composed of two line each line contains exactly one of the words 'yes' or 'no': 'yes' if it is possible to turn off all lights, and 'no' if it is not possible to turn off all the lights.

Sample Input (1):

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

Sample Output (1):

```
no
no
```

Sample Input (2):

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

Sample Output (2):

```
yes
no
```

Sample Input (2):

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

Sample Output (2):

```
no
yes
```