CS 325: Analysis of Algorithms
Group Assignment 4

Busra Dedeoglu
Hyuntaek Oh
Yu-Hao, Shih

14 March 2024

Description of Algorithm

In this assignment, we are required to get answers such as “Yes” or “No” from the program that turns off all the lights in the OSU campus. To find the answers, we need to convert the lights and switches problem to 2-CNF problems since the turning a switch twice means not turning it, and changing the order of toggling the switches does not change the results. The reduction from the problem to 2-SAT, which encapsulates the problem within black box that is used as 2-CNF problem and can be applied for solving 2-SAT problems. We will use the given 2-SAT solver, which uses Kosaraju’s algorithm and finds contradiction with strong connected components.

We are required to define Boolean variables, which represent the action of toggling the switches. If the switch is toggled, the variable can be True whereas if the switch is not toggled, the variable can be False.

To build clauses for the 2-SAT problem, the program checks each initial state of the lights one by one and its connection to two switches. For example, one light has two switches: S_1 and S_2. For each light, if the initial state of the light is “1”, which means “ON”, we need to turn off the light to change current state. The lights can be changed by other switch since each light is connected to two switches, which means we carefully choose which switch is toggled. So, we choose the clauses below that work together as a formula, ensuring that at least one switch is toggled to turn off.

$$(S_1 \vee S_2)$$

$$(\neg S_1 \vee \neg S_2)$$

$$(S_1 \vee S_2) \wedge (\neg S_1 \vee \neg S_2)$$

On the other hand, if the initial state of the lights is “0”, which means “OFF”, we need to maintain this state. If both switches are toggled, it can be achieved since one switch would turn on and another switch would turn off the light. The clauses below ensure that

remaining its state.

$$(S_1 \vee \neg S_2)$$

$$(\neg S_1 \vee S_2)$$

$$(S_1 \vee \neg S_2) \wedge (\neg S_1 \vee S_2)$$

Thus, these meet the condition of turning off each light or remaining its state. With these clauses, we use the given 2-SAT solver function, which provides the result of the problem that whether 2-CNF is satisfiable or not. Then, we can finally get the answer “Yes” or “No”.

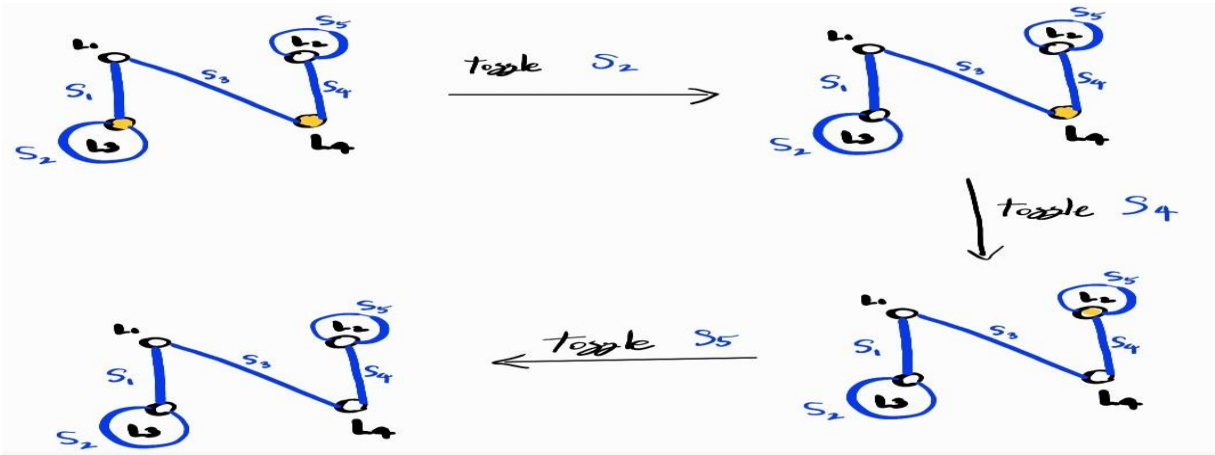
Running Time Analysis

In the reduction process, there are two things we need to consider: the number of lights and switches, which are m and n respectively. For each light, the program generates two clauses checking the connected switches and the light’s initial state, which takes linear time $O(m)$. For each switch, the program changes them into Boolean variables, which are direct and take linear time $O(n)$. Therefore, the overall time complexity of the reduction is polynomial, which is $O(m+n)$, solving the problem efficiently and meeting the condition of polynomial time solvability with 2-SAT algorithms.

Proof of Correctness

To prove the correctness of the reduction, we can assume that if there is a feasible solution to the 2-SAT problem, there would be a map of switches that turn off all the lights. Since each clause is based on the initial state of a light, and the light connected switches, a satisfiable 2-SAT instance exists and corresponded to a switch toggling map that can turn off all the lights.

For example, the picture below proves the process of True case that turns off all the lights.



In this case, one light has two switches, and each switch can affect another light. To turn off all the lights, we start from L_1 to L_4 to turn off the light one by one. L_1 and L_2 is not on, so we can skip this process to turn off. When we check the L_3 is on, we need to choose which switch to turn off. In the picture, when we choose S_2 to toggle, it does not affect other lights because it connects to L_3 only. After that, when we toggle the S_4, L_4 is off but L_2 is on because the switch is connected. Then, we can use S_5 to turn off L_2. The result of the problem can be "Yes" since all the lights are "OFF". If we use other clauses to turn off the lights like $(\neg S_1 \vee \neg S_2)$ and $(\neg S_1 \vee S_2)$ to turn off, the light would be on and off repeatedly, and the result shows False even though it is True case.

Moreover, If there is a way to turn off all lights, then the switch states corresponding to the method above satisfy all clauses constructed for each light, which means we can find a satisfiable true assignment for the 2-SAT problem meeting all clauses, proving that the 2-SAT instance is satisfiable.

Therefore, our reduction proves the 2-SAT instance can also be correct.