# Homework 11: CS 603

(Polynomial Reductions)

Algorithms (Fall 2025)
University of San Francisco
Due: Friday, November 21$^{st}$

**Brandon Paulsen**

---

**Problem 1:** You live in a very bizarre apartment with a very long hallway filled with unusual light switches. You need to go to bed, and thus want to turn off all the lights. Say the light switches are numbered $1, 2, 3, \ldots, n$ as you walk down the hallway. There are also $n$ lights that you encounter as you walk down the hallway, one next to each switch, so you would guess that the switch changes the light is is next to. But no, your mischievous landlord has set up the light switches to do something unexpected: when you flip switch 1, all the lights $1, 2, 3, \ldots n$ change. (If they were on, they're now off. If they were off, then they're now on.) And in general, if you flip the $i^th$ switch, then lkights $i, i+1, i+2, \ldots, n$ change. You want to get to bed as quickly as possible because you know that sleep is good for your brain and body.

Please write a method `minFlips` that will calculate the minimum number of light switches you need to flip in order to turn off all the lights. The input for this method is an array `lights[]` whose entries are 0 or 1; a value of 0 indicates the light is off, and a value of 1 indicates the light is on. The starter code for this problem is here: `https://classroom.github.com/a/ioSsBG73`.

Please also state and explain the time complexity of your solution.

EXTRA CREDIT!! Please explain (prove) why your solution works. (Not only is this problem worth the extra credit of one "activity", it is good proof practice for those of you wanting more practice!)

**Solution 1:**

---

**Problem 2:** The problem is to make sure you understood the gadget construction which was used to show that 3-dimensional matching is NP-complete. Suppose we have an instance of 3-SAT which is the following: There are 3 variables and 2 clauses: $x_1, x_2, x_3$ are the variables and the clauses are:

$$C_1 = x_1 \vee x_1 \vee \overline{x_3}$$

$$C_2 = \overline{x_1} \vee x_2 3 \vee x_3$$

Please draw the gadget construction which corresponds to the instance of 3-dimensional matching that we constructed from this instance of 3-SAT. Don't forget the cleanup gadgets (although if you want, you can just describe what they are and how many there are, so your drawing doesn't get too messy).

**Solution 2:**

---

**Problem 3:** General Leia Organa has $n$ resistance platoons, with $a_i$ troops in platoon $i$, $i \in 1, 2, 3, \ldots, n$. There are two different battles she needs to send the troops to, and she'd like to split these platoons into two groups so that there is an equal number of troops going to each of the two battles. In other words, she wants to know if there exists $S \subset 1, 2, \ldots, n$ such that

$$\sum_{i \in S} a_i = \sum_{j \in 1,2,\ldots,n, j \notin S} a_j$$

Please prove to General Organa that this problem is NP-complete.

**Solution 3:** Let's start with some general notation. Let:

- $T$ be the troop problem
- $P = \{p_1, p_2, \ldots, p_n\}$ be the size of each platoon (an input of the troop problem).
- $B_1 \subset \{1, 2, \ldots, n\}$ be those platoons assigned to battle 1 (a certificate of $P$).
- $B_2 \subset \{1, 2, \ldots, n\} = \{1, 2, \ldots, n\} \setminus B_1$ be those platoons assigned to battle 2 (an equivalent certificate of $P$).

In order to prove that the troop problem $T$ is NP-complete, we must prove the following:

1. $T$ is in $\mathcal{NP}$ by proving:
    a. There is a polynomial time certifier of $T$ given input $P$ and certificate $B_1$.
    b. There is a certificate $B_1$ such that $|B_1|$ is polynomial in the size of input $|P|$.
2. There is some algorithm $Y$ in $\mathcal{NP}$ such that it can be polynomial time reduced into $T$.

The certifier $C$ can be defined as:

**Algorithm 1** Certifier

```
1: c_1 ← 0
2: s_1 ← 0
3: for each p_i ∈ B_1 do
4:     c_1 ← c_1 + 1
5:     s_1 ← s_1 + p_i
6: c_1 ← 0
7: S_2 ← 0
8: for each p_i ∈ B_2 = S \ B_1 do
9:     c_2 ← c_2 + 1
10:    S_2 ← S_2 + p_i
11: if c_1 ≠ c_2 then
12:    return False
13: return True
```

Proving 1a:

Algorithm 1 certainly certifies $T$, as it checks that $B_1$ and $B_2$ are the same size and have the same sum and takes $O(|S|)$ time because we iterate $B_1 \cup B_2 = S$ times and we do a constant amount of work per iteration. Since $O(|S|)$ is polynomial in $|S|$ and the algorithm correctly certifies $T$, $T$ has an efficient certifier.

Proving 1b:

Since $B_1 \subset P$, $|B_1| \leq |P|$, so $|B_1| \in O(|P|)$ which is polynomial in $|P|$.

Proving 2:

We will prove that there is some algorithm $Y$ in $\mathcal{NP}$