Homework 11: CS 603

(Polynomial Reductions)

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

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 with:
  - $P = \{p_1, p_2, \ldots, p_n\}$ be the size of each platoon (an input of the troop problem).
  - $B_1 \subset \{1, 2, \ldots, n\}$ be the indices of those platoons assigned to battle 1 (a certificate of P).
  - $P_1 = \{p_{b_1} \forall\, b_1 \in B_1\} \subset P$ be those platoons assigned to battle 1.
  - $B_2 \subset \{1, 2, \ldots, n\} = \{1, 2, \ldots, n\} \setminus B_1$ be the indices of be those platoons assigned to battle 2 (an equivalent certificate of P).
  - $P_2 = \{p_{b_2} \forall\, b_2 \in B_2\} = P \setminus P_1$ be those platoons assigned to battle 2.
- S be the subset sum problem with:
  - W the target value
  - $E = \{e_1, e_2, \ldots, e_n\}$ the value of each element
  - $F \subseteq E$ such that $\sum_{e \in F} e = W$ be a certificate of T.

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 $|P|$ and the certifier of T returns true for P and B_1.
2. There is some NP-complete problem Y such that it can be polynomial time reduced into T by proving:
   a. There is a problem Y that can be reduced to T.
   b. The reduction from Y to T is polynomial in the input to Y.

Proving 1a:

We can define a certifier for T as:

Algorithm 1 Certifier

```
t ← 0
s_1 ← 0
s_2 ← 0
for i ∈ {1, 2, ..., n} do
    t ← t + 1
    if i ∈ B_1 then
        s_1 ← s_1 + p_i
    else if i ∈ B_2 = {1, 2, ..., n} \ B_1 then
        s_2 ← s_2 + p_i
    else
        return False
if t ≠ n then
    return False
return s_1 = s_2
```

Algorithm 1 certainly certifies T, as it checks that $\sum_{b_1 \in B_1} p_{b_1} = \sum_{b_2 \in B_2} p_{b_2}$, $B_1 \cup B_2 = P$, and $B_1 \cap B_2 = \varnothing$ and takes $O(|P|)$ time because we iterate $|B_1 \cup B_2| = |P|$ times and we do a constant amount of work per iteration. Since $O(|P|)$ is polynomial in $|P|$ and the algorithm correctly certifies T, so 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 2a:

In order to prove that there is some NP-complete problem Y such that Y can be polynomial time reduced to T, we must first select some NP-complete problem Y to reduce. There are many NP-complete problems to pick from, and our choice greatly impacts the method of polynomial reduction to T. Selecting Y that is similar (in some sense) to T presents the most straightforward polynomial reduction.

For the troop problem, we must select B_1 such that $\sum_{p_i \in P_1} p_i = \sum_{p_j \in P_2} p_j$. A similar NP-complete problem that seems reasonable to select is subset sum. In order to reduce an instance of subset sum $S = (E, W)$ to an instance of $T = (P)$, we must construct a set of platoons P such that the troop problem is true if and only if the subset sum problem is true. So what should P be?

In order for T to be true, P must be able to be spit into two sets P_1 and P_2 with equal sums, so $\sum_{p \in P} p$ must be even. In order for $S \Leftrightarrow T$, we must have $E \subseteq P$, since removing any element in E might change the outcome of a given instance of S. So, we must construct P by adding some elements A to E such that P has an even sum: $P = E \cup A \mid \sum_{p \in P} p \equiv 0 \pmod 2$.

But there are lots of even numbers - what should $\sum_{p \in P} p$ be? Consider what would happen if $A = \{a_1, a_2\}$ and a_1 and a_2 were too large to both be in either P_1 or P_2. Since P gets split into P_1 and P_2 and a_1 and a_2 must be too large to be in either P_1 or P_2, we must have:

$$\sum_{a \in A} a > \frac{\sum_{p \in P} p}{2}$$

And since $P = E \cup A$,

$$\sum_{p \in P} p = \sum_{e \in E} e + \sum_{a \in A} a$$

$$\Rightarrow \sum_{a \in A} a > \frac{\sum_{e \in E} e + \sum_{a \in A} a}{2}$$

$$\Rightarrow \sum_{a \in A} a > \sum_{e \in E} e$$

What happens if we select A with the minimal sum satisfying this relation:

$$\sum_{a \in A} a = \sum_{e \in E} e + 1$$

$$\Rightarrow \sum_{p \in P} p = \sum_{a \in A} a + \sum_{e \in E} e = 2 \sum_{e \in E} e + 1$$

But this is odd, which we cannot have if the troop problem is to be solvable. What if it was just a little larger?

$$\sum_{a \in A} a = \sum_{e \in E} e + 2$$

$$\Rightarrow \sum_{p \in P} p = \sum_{a \in A} a + \sum_{e \in E} e = 2 \sum_{e \in E} e + 2$$

Since this is even, we have a valid instance of the troop problem. Now, we have a value for $a_1 + a_2$, but what should the values be? Consider what it means for a_1 and a_2 to be in different partitions of P. Without loss of generality, let $a_1 \in P_1$. Since a_1 and a_2 are in different partitions, P_1 must contain a_1 as well as some subset of E. This sounds exactly like the subset sum problem! In order for this to be exactly analogous to the subset sum problem, the sum of the subset of elements in P_1 must be W:

$$\sum_{p \in P_1} p = a_1 + W$$

And since we are splitting P into P_1 and P_2 that have equal sums,

$$2 \left(\sum_{e \in E} e + 1 \right) = \sum_{p \in P} p = \sum_{p \in P_1} p + \sum_{p \in P_2} p = 2 \sum_{p \in P_1} p = 2 (a_1 + W)$$

$$\Rightarrow 2 \left(\sum_{e \in E} e + 1 \right) = 2 (a_1 + W)$$

$$\Rightarrow \sum_{e \in E} e + 1 = a_1 + W$$

$$\Rightarrow a_1 = \sum_{e \in E} e + 1 - W$$

Now we know a value for a_1, but what about a_2? We know the value for $a_1 + a_2$, so we can simply subtract a_1 from that value to get a_2:

$$a_1 + a_2 = \sum_{e \in E} e + 2$$

$$\Rightarrow a_2 = \sum_{e \in E} e + 2 - a_1$$

$$\Rightarrow a_2 = \sum_{e \in E} e + 2 - \left(\sum_{e \in E} e + 1 - W \right)$$

$$\Rightarrow a_2 = W + 1$$

Now, we have fully defined an instance of the troop problem given a general instance of the subset sum problem $S = (E, W)$:

$$P = E \cup \left\{ \sum_{e \in E} e + 1 - W, W + 1 \right\}$$

By construction, solving this instance of the troop problem is exactly analogous to selecting a subset of the element of the subset sum problem whose values add up to W, and therefore, their outcomes must be equivalent.

Proving 2b:

In order to translate a subset sum instance into a troop problem instance, we add two elements ($O(2)$ is polynomial in any input) whose values we compute by summing all subset sum elements E (in $O(|E|)$ time, which is polynomial in input size). Additionally, we use polynomially many calls to a black box solver of the troop problem (1 call is polynomial in any input). Overall, this is polynomial in $|E|$, so this reduction is indeed polynomial time. This proves that $S \leq_p T$, and S is NP-complete, so T must also be NP-complete.
