COMP 3804 Assignment 4

April 12, 2021

Question 1.

Name: Braeden Hall
Student Number: 101143403

Question 2.

To prove: show that $INTPROG$ can be verified in $O((nm)^c)$ time.

Proof: Input: $m \times n$ matrix A, vector c of length m;
Certificate: binary vector $x = (x_1, \ldots, x_n)$

Steps:

- Check that for each $1 \leq i \leq n$: $x_i \in \{0, 1\}$. Time: $O(n)$
- Check that $|(x_1, \ldots, x_n)| = n$. Time: $O(n)$
- Check that $Ax \leq c$ component-wise. I.e. multiply A by x, then for each $1 \leq i \leq m$ check that $(Ax)_i \leq c_i$. Time: $O(nm)$
- If each of the previous checks are successful return YES, otherwise, return NO. Time: $O(1)$

Total time to run verification algorithm: $O(nm)$. Since the size of the input (the matrix and vector c) is $O(nm)$ the algorithm runs in polynomial time. Since the size of the certificate is $O(n)$ it is also polynomial in the size of the input. Therefore, the language $INTPROG$ is in NP.

Question 3.

To prove that $SubsetSum \leq_P INTPROG$ we need: a function f where $f(a_1, \ldots, a_m, b) \to (A, c)$, such that $\exists I \subseteq \{1, \ldots, m\}$, $b = \sum_{i \in I} a_i \iff \exists x \in \{0, 1\}^n, Ax \leq c$ component-wise.

Definition of function f:

- Let A be a $2 \times m$ matrix where the first row is the sequence $a_1, \ldots, a_m$. The second row is the same sequence, however, with every term negated. Therefore it is the sequence $-a_1, -a_2, \ldots, -a_m$. Time: $O(m)$
- Let c be a vector of length 2 where the first component is the integer b and the second component is the negated integer b, so $-b$. Time: $O(1)$
- Run the algorithm to solve $INTPROG$ with the input (A, c).

After the $INTPROG$ algorithm runs, if no vector x is found then no subset $I \subseteq \{1, \ldots, m\}$ exists to satisfy $b = \sum_{i \in I} a_i$. However, if a vector x is found that satisfies $Ax \leq c$, then the subset I can be constructed as follows. Since x is a binary vector all of its components will either be 0 or 1. So for each $1 \leq i \leq m$: if $x_i = 1$ then i will be in the set I, otherwise i will not be in the set.

This will satisfy $b = \sum_{i \in I} a_i$ because of the way the input of $INTPROG$ was formatted. Since we are assuming that a satisfactory vector x exists we know that $(Ax)_1 \leq c_1$; i.e. $b \geq \sum_{i \in I} a_i$. The second components ($(Ax)_2$ and c_2) effectively mean that $(Ax)_1 \geq c_1$ (since values in the second components are just the negated values of the first components); i.e. $b \leq \sum_{i \in I} a_i$.

We have now seen that b is both less than or equal and greater than or equal to the sum of some elements of the sequence $a_1, \ldots, a_m$. From this we can conclude that b must be equal to the sum of those elements.

Therefore, we have seen the definition of a function f that, in $O(m)$ time, converts inputs from the $SubsetSum$ problem into input to the $INTPROG$ problem such that $(a_1, \ldots, a_m, b) \in SubsetSum$ if and only if $(A, c) \in INTPROG$. This means that $SubsetSum \leq_P INTPROG$.

Question 4.

1. To prove: show that $3COLOR$ can be verified in $O((|V| + |E|)^c)$ time.

   Proof: Input: Graph $G = (V, E)$;
   Certificate: sequence of colours $(c_1, \ldots, c_n)$
   Assume: $n = |V|$, $m = |E|$, for each $i \in \{1, \ldots, n\}$, c_i is colour of vertex v_i.

   Steps:

   - Check that $|(c_1, \ldots, c_n)| = n$. Time: $O(n)$
   - Check that no one vertex has multiple colours. Time: $O(n)$
   - Check that the amount of distinct colours in $(c_1, \ldots, c_n) \leq 3$. Time: $O(n)$
   - For each edge $\{v_i, v_j\} \in E$: check that $c_i \neq c_j$. Time: $O(n + m)$
   - If each of the previous checks are successful return YES, otherwise, return NO. Time: $O(1)$

   Total time to run verification algorithm: $O(n+m)$. Since the size of the input (the graph) is $O(n+m)$ the algorithm runs in polynomial time. Since the size of the certificate is $O(n)$ it is also polynomial in the size of the input. Therefore, the language $3COLOR$ is in NP.

2. To prove: show that $CLIQUECOVER$ can be verified in $O((|V| + |E|)^c)$ time.

   Proof: Input: Graph $G = (V, E)$, integer k;
   Certificate: set of subsets $\{V_1, \ldots, V_k\}$
   Assume: $n = |V|$, $m = |E|$, $k \leq n$

   Steps:

   - Check that $|\{V_1, \ldots, V_k\}| = k$. Time: $O(k)$
   - Check that $|\{V_1 \cup V_2 \cup \cdots \cup V_k\}| = n$. Time: $O(n)$
   - Check that no single vertex appears in multiple subsets V_i and V_j etc. Time: $O(n^2)$
   - For each $i \in \{1, \ldots, k\}$: let $p = |V_i|$, check that edges connect each of the p vertices to all other $p - 1$ vertices in the subset. I.e. edges: $\{v_{ij}, v_{i1}\}, \{v_{ij}, v_{i2}\}, \ldots, \{v_{ij}, v_{ip}\}$ exist for each $v_{ij} \in V_i$. Essentially this checks that each subset V_i forms a clique. Time: $O(n + m)$
   - If each of the previous checks are successful return YES, otherwise, return NO. Time: $O(1)$

   Total time to run verification algorithm: $O(max(n+m, n^2)) = O((n+m)^2)$. Since the size of the input (the graph and integer k) is $O(n+m)$ the algorithm runs in polynomial time. Since the size of the certificate is $O(n)$ it is also polynomial in the size of the input. Therefore, the language $CLIQUECOVER$ is in NP.

3. To prove that $3COLOR \leq_P CliqueCover$ we need: a function f where $f(G) \to (G', k)$ such that G is 3-colourable $\iff$ (G', k) is "clique-coverable".

   Definition of f :

   - Define G' as the complement of the graph G from $3COLOR$. Time: $O(n+m)$
   - Run the algorithm to solve $CliqueCover$ with the input $(G', 3)$.

   After $CliqueCover$ runs if no subsets V_1, V_2, V_3 (since k is always 3) that form cliques to cover G' exist, then G is not 3-colourable. However, if such subsets exist then G can be coloured simply by giving the vertices in each subset the same colour, but giving different colours to vertices in different subsets.

   We know colouring the graph like this works because all the vertices in the one subset form a clique in the complement of G: G'. This means that those vertices form an independent set in G, i.e. none of the vertices are adjacent to each other so they can get the same colour. We always pass $k = 3$ because we want to see if the graph can be coloured using 3 colours so we only want 3 sets of vertices.

   Therefore, we have seen a definition of a function f that, in $O(n+m)$ time, converts inputs to $COLOR$ into inputs to $CliqueCover$ such that $(G) \in 3COLOR$ if only and only if $(G', k) \in CliqueCover$. This that $3COLOR \leq_P CliqueCover$.

Question 5.

To show that $PROBLEMWITHOUTANAME$, from now on referred to as PWN, is NP-complete we must show the it is in NP and prove that it is "more" difficult than some other NP-complete problem. The NP-complete problem I will be using is $3SAT$ (in class we have seen that $3SAT$ is indeed NP-complete).

First, show that PWN is in NP:

To prove: show that PWN can be verified in $O((k+l)^c)$ time (since the sequence $\mathcal{A}$ has $3k$ elements and the sequence $\mathcal{B}$ has $2l$ elements).

Proof: Input: sequence $\mathcal{A}$, sequence $\mathcal{B}$
Certificate: set T (note that $|T| = O(k+l)$ since it can have at most $3k+l$ elements).

Steps:

- Check that T contains at least 1 element of each A_i for $i \in \{1, \ldots, k\}$.
  Time: $3k \cdot (3k+l) = O(k^2+l) = O((k+l)^2)$
- Check that T contains at most 1 element of each B_j for $j \in \{1, \ldots, l\}$.
  Time: $2l \cdot (3k+l) = O(k+l^2) = O((k+l)^2)$
- If each of the previous checks are successful return YES, otherwise, return NO. Time: $O(1)$

Total time to run verification algorithm: $O((k+l)^2)$. Since the size of the input (the 2 sequences) is $O(k+l)$ the algorithm runs in polynomial time. Since the size of the certificate is $O(k+l)$ it is also polynomial in the size of the input. Therefore, the language PWN is in NP.

Next, show that $3SAT \leq_P PWN$:

We need: a function f where $f(\varphi) \to (\mathcal{A}, \mathcal{B})$ such that φ is satisfiable $\iff$ $(\mathcal{A}, \mathcal{B})$ is good.

Definition of function f:

- Assume the equation φ has k clauses and variables $x_1, x_2, \ldots, x_l$.
- Define $\mathcal{A}$ as the sequence $(A_1, \ldots, A_k)$ where each A_i consists of the 3 terms in clause C_i. Time: $O(k)$
- Define $\mathcal{B}$ as the sequence $(B_1, \ldots, B_l)$ where each B_j consists of term x_j and its negation.
  Ex: $B_1 = (x_1, \neg x_1)$. Time: $O(l)$
- Run the algorithm to solve PWN with input $(\mathcal{A}, \mathcal{B})$.

After the PWN algorithm runs if no set T is found then the equation φ is not satisfiable. However, if some set T is found such that $(\mathcal{A}, \mathcal{B})$ is good, then the equation φ can be satisfied. To do this you simply need to assign values each variable such that each element in the set T results in "true" in the equation. So if $x_i \in T$ and $\neg x_j \in T$ then $x_i \leftarrow true$ and $x_j \leftarrow false$. Any variable that does not appear in T can be assigned an arbitrary value.

We know that this will produce a satisfiable equation because from the way $\mathcal{A}$ was constructed each clause will yield true since some term of each clause is in T. From the way $\mathcal{B}$ was constructed we know that for some variable x_j, x_j and $\neg x_j$ cannot both be in T; so it can never be the case that both x_j and $\neg x_j$ need to be true.

Therefore, we have seen the definition of a function f that, in $O(k + l)$ time, converts inputs to $3SAT$ into inputs to PWN such that $(\varphi) \in 3SAT$ if and only if $(\mathcal{A}, \mathcal{B}) \in PWN$. This means that $3SAT \leq_P PWN$.

Finally, since we knoe that $3SAT$ is NP-complete and it has been shown that PWN is in NP and that it is "harder" than $3SAT$ we can conclude that PWN is also NP-complete.