CPSC 421 - Homework 9

Tristan Rice, q7w9a, 25886145

1. Partition

1.1 Argue that PARTITION is in NP

We aim to show that SUBSET-SUM is in NP by constructing a verifier in P.

V = "On input $\langle n_1, \dots, n_m \rangle, I$

1. Test that each element of $I \in \{1, \dots, m\}$ and each element appears at most once.
2. Test that $\sum_{i \in I} n_i = \sum_{i \notin I} n_i$.
3. If both pass, accept; otherwise, reject."

Each of these checks can be done in $O(n)$ time, thus $V \in P$ and $PARITION \in NP$.

1.2 Reduction of SUBSET-SUM to PARTITION

We can reduce SUBSET-SUM to PARTITION by adding an element $(\sum n_i) - 2t$.

We can see that this works since if there is a subset that satisfies SUBSET-SUM there must be a set of numbers that add up to t. Thus we can satisfy the constrain for partition:

$$\sum_{i \in I} n_i = \sum_{i \notin I} n_i$$

$$t + (\sum n_i) - 2t = (\sum n_i) - t$$

$$(\sum n_i) - t = (\sum n_i) - t$$

1.3 Explain reduction is in P time

You can compute the sum of the set $\sum n_i$ in $O(n)$ time since it requires only one pass over each element. Adding an element to a set can occur in $O(1)$ time. Thus, the transformation takes $O(n)$ time which is clearly in polynomial time.

2. Exercies 7.24 of Sip.

2.a Show that $CNF_2 \in P$

We can construct a polynomial time algorithm that solves a CNF_2 problem to show that it is in P.

We can recursively eliminate variables until we can show that it's satisfiable or not.

Once the problem has been reduced to empty we know it can be satisfiable.

We can consider each term that is connected by ands. If there are two terms that can be written as $a \wedge \neg a$ the expression can't be satisfied.

If there are two terms $a \wedge a$ we can remove them since they're easily satisfiable.

Any variable that is only used once can trivially be satisfied and the entire term can be removed from the problem.

If we have an expression in the form $(a \vee \dots) \wedge (a \vee \dots)$ we can remove those terms since you can satisfy both by setting a to true.

If there is an expression in the form $(a \vee C) \wedge (\neg a \vee D)$ we can replace it with $(C \vee D)$. (Similar to the inverse of how we reduce to 3SAT)

Since each of these steps remove a variable in constant time, and there's a linear number of variables. This algorithm runs in $O(n)$ and takes polynomial time. Thus, CNF_2 can be solved in polynomial time and is in P.

2.b Show that CNF_3 is NP-complete

First we need to show that CNF_3 is in NP. We can do this by creating a polynomial time verifier.

V = Given a solution to CNF_3 we can subsitute in the boolean values for each variable in the problem definition. Evaluate the boolean formula and accept if it evaluates to true; otherwise reject.

Each operation can have a logical not applied to it, which can be computed in constant time. There are $n-1$ binary operations since there are n variables in the statement. Each operation takes constant time so thus, the verifier runs in $O(n)$ time.

To show NP-completeness we can do a reduction from SAT to CNF_3.

We can create a chain of variables by adding clauses that force them to all be equivalent. We can force a_1, a_2, a_3 to be equivalent by adding clauses $(\neg a_1 \vee a_2) \wedge (\neg a_2 \vee a_3) \wedge (\neg a_3 \vee a_1)$. To transform any SAT problem into CNF_3 we can replace each variable with the a_i version of it and add a series of clauses in to the end of the CNF expression. We can do this transformation in $O(n)$ time where n is the number of variables since for each variable we have to do a constant amount of work. Thus, we can reduce from SAT to CNF_3 in polynomial time and therefore CNF_3 must be NP-complete.

3. Exercies 7.29 of Sip.

We can see that 3COLOR is in NP by showing it has a polynomial time verifier.

V = for every node verify that it has max of 2 adjacent nodes and every adjacent node has a different color.

This runs in $O(N)$ since it does a constant amount of work for each node since the number of adjacent nodes is bounded.

We can do a reduction from SAT to 3COLOR to show that 3COLOR is NP-complete.

We assign one color to be true, one color to be false and the third color doesn't represent anything.

Using the 3 subgraphs we can see that the "palette" forces any node that it is connected to either be true or false.

The "OR-gadget" allows us to compute a logical "or".

We can create a "not" gate by adding a single edge and node. Since using the palette subgraph we can force each node it's connected to to either be true or false, the new node has to be the opposite of what it's connected to.

Since we can construct "and" using only "or" and "not" we can represent any SAT problem using a coloring problem. Thus, we can see if a problem is satisfiable using a 3COLOR solver.

If we use conjunctive normal form, we see that the number of logical operators for n nodes is linear with respect to n. Adding each one of the logical subgraph constructs above is $O(1)$, thus, overall to construct the SAT graph the runtime is $O(n)$ and in polynomial time. Since we can reduce SAT to 3COLOR in polynomial time, that means that $3COLOR$ must be NP-complete.

4. Problem 4 of Final Exam from December 2014

We can show that DOUBLE-SAT is in NP by creating a polynomial time verifier.

V = Given a set of inputs to a boolean formula and a description of the formula, solve each pairwise logical operation until we have a $true$ or $false$ answer. If it is true, accept; otherwise reject.

Since this requires constant amount of work for each logical operator it is in $O(n)$ time. Thus, DOUBLE-SAT is in NP.

We can reduce SAT to DOUBLE-SAT by adding a $\vee x$ clause where x can be either true or false. If there is only a single satisfying assignment, that means that x must be true and the original clause doesn't have a satisfying assignment. If there are at least two satisfying assignments, that means the original clause must have at least one satisfying assignment.

Adding a single new clause is $O(1)$ and thus in polynomial time. Since SAT is NP-complete and you can reduce it to DOUBLE-SAT in polynomial time, that means that DOUBLE-SAT is NP-complete.