Group Homework

Problem (1)

(a)

For each assignment $x_1, \ldots, x_n$ such that $f(x_1, \ldots, x_n)$ is true we create a $c_i$ by taking the AND of each variable $x_1, \ldots, x_n$ corresponding to its value in the assignment. That is, if $x_j$ in the assignment is true then we take the AND of $x_j$ in $c_i$ and if $x_j$ is false in the assignment then we take the AND of $\sim x_j$ in $c_i$. Therefore, $c_i$ is the AND of $n$ literals and more importantly $c_i$ is true if and only if $x_1, \ldots, x_n$ have the particular assignment that was used to create $c_i$.

Let $s$ be the number of assignments of $x_1, \ldots, x_n$ for which $f(x_1, \ldots, x_n)$ is true. Since there are at most $2^n$ possible assignments we have that $s \leq 2^n$. We claim that $f = c_1 \lor \ldots \lor c_s$.

Proof of this claim: Let $x_1, \ldots, x_n$ be an arbitrary assignment. If $f(x_1, \ldots, x_n)$ is true then there is some $c_i$ corresponding to this assignment. Therefore $c_i$ is true and in turn $c_1 \lor \ldots \lor c_s$ is true. If $f(x_1, \ldots, x_n)$ is false then every $c_i$ is false since each $c_i$ corresponds to a truth assignment $x'_1, \ldots, x'_n$ for which $f(x'_1, \ldots, x'_n)$ is true. Hence $c_1 \lor \ldots \lor c_s$ is false. This completes the proof that $f = c_1 \lor \ldots \lor c_s$. 
We apply part (a) to $\sim f$ and so we have that $\sim f = c_1 \lor \ldots \lor c_s$ where each $c_i$ is an AND of $n$ literals. Then by De Morgan’s law we have that $f = \sim (c_1 \lor \ldots \lor c_s) = \sim c_1 \land \sim c_2 \land \ldots \land \sim c_s$.

Letting $c'_i$ be the result of De Morgan’s law applied to $c_i$ we have that $c'_i$ is a OR of $n$ literals. Thus we obtain the desired result, $f = c'_1 \land \ldots \land c'_s$. 
Problem (2)

To show that 3COLOUR is NP-complete we need to (i) show that it is in NP and (ii) to show that all problems in NP can be reduced to 3COLOUR in polynomial time.

(i)
3COLOUR is in NP since we can non-deterministically guess a colouring of the vertices of the graph with 3 colours and then check if this gives a valid 3-colouring of the graph. Checking if the colouring is valid will require at most polynomial number of steps in the size of the graph. Therefore, the non-deterministic TM deciding 3COLOUR will take at most non-deterministic polynomial number of steps to halt.

(ii)
To show that all problems in NP can reduce to 3COLOUR in polynomial time it is enough to give a polynomial-time reduction from 3SAT to 3COLOR, using the subgraphs in the hint for Problem 7.29 in [Sip].

Consider the palette as shown in 7.29, where T is the colour turquoise, F is the colour fuchsia, and the third node is G for grey. [Note: the textbook uses T and F to suggest true and false, and this has advantages and disadvantages]

As in 7.29, each variable $x_i, i = 1, ..., n$ has vertices, $v_{i,1}, v_{i,2}$, joined by an edge, and both vertices also joined to grey. It follows that $v_{i,1}, v_{i,2}$ are coloured turquoise and fuchsia in any 3-colouring of the graph constructed so far.
To illustrate the use of the OR-gadgets, consider the clause \( x_i \lor \neg x_j \lor \neg x_k \).

Let us use the OR-gadget to express \( x_i \lor \neg x_j \) we introduce one “OR-gadget” -i.e., a triangle as in 7.29- three vertices, two “bottom” vertices and one “top” vertex, each pair of which are joined by an edge - and where we add the following edges: (1) the bottom left vertex has an edge to \( v_{i,1} \), (2) the bottom right an edge to \( v_{j,2} \), and (3) the top vertex has an edge to G. Let us label the top vertex \( x_i \lor \neg x_j \).

We claim that (1) if \( v_{i,1} \) and \( v_{j,2} \) are both coloured F then the top vertex must be coloured F (and the same for T), and (2) if \( v_{i,1} \) and \( v_{j,2} \) are coloured differently, then the top vertex can be coloured T (in the graph we have built so far).
We then add another OR-gadget (another triangle), whose bottom vertices are connected to the $x_i \lor \neg x_j$ vertex and to $v_{k,2}$. Similarly, we have that the new top vertex - label it $x_i \lor \neg x_j \lor \neg x_k$ - can be coloured T, turquoise, (in the graph built so far) iff at least one of $v_{i,1}, v_{j,2}, v_{k,2}$ is coloured T; otherwise the new top vertex must be coloured F, fuchsia. Now introduce an edge from this new top vertex to the palette vertex coloured F, fuchsia. We then do a similar construction for every clause.

This takes a Boolean formula $f$ in 3CNF form and produces a graph in polynomial time, since the number of vertices and edges in this graph is linear in the number of variables and clauses in $f$. If $f$ is satisfiable, then consider any satisfying assignment for $x_1, \ldots, x_n$ if $x_i$ is true in the assignment, colour $v_{i,1}$ turquoise and $v_{i,2}$ fuchsia. Otherwise, reverse the colours. Then each vertex corresponding to a clause that is the OR of three variables can be coloured T, turquoise, so that the entire graph is 3-colourable.
Conversely, given a 3-colouring of this graph, consider the assignment of each variable $x_i$ to true if $v_{i,1}$ is coloured turquoise, and false if fuchsia. Then we see that for each clause at least one of the vertices, $v_{i,1}$ or $v_{i,2}$, corresponding one of the variables, $x_i$ or $\neg x_i$ (respectively) in the clause must be coloured turquoise. Hence this assignment of the $x_i$ is a satisfying assignment.

Therefore, the graph produced from the 3CNF is 3-colourable iff $f$ is satisfiable. Since the algorithm to produce the graph runs in polynomial time, this gives a reduction SAT $\leq_P$ 3COLOR.

We’ve shown both (i) and (ii) so 3COLOUR is NP-Complete.
Problem (3)

To show that 3COLOUR is NP-complete we need to (i) show that it is in NP and (ii) to give a polynomial time reduction from 3COLOUR to 4COLOUR (since we are assuming that 3COLOUR is NP-complete)

(i)

Similar to problem(2). 4COLOUR is in NP since we can non-deterministically guess a colouring of the vertices of the graph with 4 colours and then check if this gives a valid 4-colouring of the graph. Checking if the colouring is valid will require at most polynomial number of steps in the size of the graph. Therefore, the non-deterministic TM deciding 4COLOUR will take at most non-deterministic polynomial number of steps to halt.

(ii)

Given a graph $G$ which we want to check for 3-colourability, generate a new graph $G'$, such that $G'$ is exactly the same as $G$, but we have added a new vertex $v$ that is connected to all other vertices in $G'$.

If $G$ is 3-colourable, then $G'$ is 4-colourable. We take the colouring for $G$ and apply it to the vertices of $G'$ and use a different colour for $v$.

If $G$ is not 3-colourable, then we have 2 cases:

Case 1: $G$ is 4-colourable
The $G'$ cannot be 4-colourable, since $G$ is a subgraph of $G'$ that is not 3-colourable but it 4-colourable, we require a 5th colour for $v$ to have a valid colouring for $G'$.

Case 2: $G$ is not 4-colourable
The $G'$ cannot be 4-colourable, since $G$ is a subgraph of $G'$.

We’ve shown that $G'$ is in 4COLOUR iff $G$ is in 3COLOUR. Since $G'$ has one more vertex and $|V(G)|$ more edges than $G$, we have a polynomial time reduction from 3COLOUR to 4COLOUR.

We’ve shown (i) and (ii), therefore 4COLOUR is NP-complete.
Problem (4)

To show that 4SAT is NP-complete we need to (i) show that it is in NP and (ii) to give a polynomial time reduction from 3SAT, a known NP-complete problem, to 4SAT.

(i)

4SAT is in NP since we can non-deterministically guess an assignment to the boolean variables and then verify that each clause is satisfied. Verifying that each clause is satisfied takes at most a polynomial number of steps in the size of the 4CNF formula. Therefore, the non-deterministic Turing Machine deciding 4SAT will take at most non-deterministic polynomial number of steps to halt.

(ii)

Let \( f = c_1 \land c_2 \land \ldots \land c_m \) be an arbitrary 3CNF formula over boolean variables \( x_1, \ldots, x_n \). We first give a polynomial time procedure that converts \( f \) into a 4CNF formula, \( f' \). For each clause \( c_l \) of \( f \) we create two corresponding clauses and a new variable. Without loss of generality let \( c_l = (x_i \lor x_j \lor x_k) \). Let \( a_l \) be a new variable, \( c_l^1 = (x_i \lor x_j \lor x_k \lor a_l) \), \( c_l^2 = (x_i \lor x_j \lor x_k \lor \neg a_l) \). Then we let \( f' = c_1^1 \land c_2^1 \land c_2^1 \land \ldots \land c_m^2 \) over boolean variables \( x_1, \ldots, x_n, a_1, \ldots, a_m \). This procedure takes polynomial time since the number of clauses in \( f \) is polynomial in the size of \( f \) and this procedure creates two clauses and a new variable for each clause of \( f \).

Now it remains to show that \( f \) is satisfiable if and only if \( f' \) is satisfiable. Let \( a_1, \ldots, a_m = T \). Then for any arbitrary pair of clauses \( c_l^1 \) and \( c_l^2 \) we have that

- \( c_l^1 = (x_i \lor x_j \lor x_k \lor a_l) = (x_i \lor x_j \lor x_k) = c_l \)
- \( c_l^2 = (x_i \lor x_j \lor x_k \lor \neg a_l) = (x_i \lor x_j \lor x_k) = c_l \)

With this assignment of the \( a_l \)'s we have that \( c_l^1, c_l^2 = c_l \) and so for any assignment of the boolean variables \( x_1, \ldots, x_n \), \( f \) is satisfied if and only if \( f' \) is satisfied.
Note that setting any one the $a_i$'s to false will not satisfy $f'$. Thus $f \in 3\text{SAT}$ if and only if $f' \in 4\text{SAT}$. We have shown (i) and (ii), therefore $4\text{SAT}$ is NP-complete.
Problem (5)

We reduce $L_1$ to $L_3$ by first reducing $L_1$ to $L_2$ and then reducing from $L_2$ to $L_3$. We argue that the time complexity is $O(n^{15})$. Let $w$ be an arbitrary instance to the decision problem of $L_1$, where the size of $w$ is $n$. We can convert $w$ to an instance of the decision problem of $L_2$, calling it $w'$, by a Turing Machine that runs in $O(n^3)$ time. This time complexity of $O(n^3)$ implies that the Turing Machine can only write on $O(n^3)$ consecutive cells of the tape and so the size of $w'$ is $O(n^3)$. Let $m$ be the size of $w'$. Then the reduction of $w'$ to an instance of the decision problem of $L_3$ takes $O(m^5)$ time which is $O(n^{15})$. Since the the time complexity for the first reduction is $O(n^3)$ and the time complexity for the second reduction is $O(n^{15})$ then the overall time complexity for the reduction from $L_1$ to $L_3$ is $O(n^{15})$. 