Homework 23

Nicholas Amoscato
naa46@pitt.edu

Josh Frey
jtf15@pitt.edu

October 25, 2013
CS 1510 Algorithm Design

1. **Reduction Problem 16:**

   Let $SATCNF$ be the decision version of the SAT-CNF problem that determines whether or not a Boolean formula F in conjunctive normal form is satisfiable. Let $3COLOR$ be the decision version of the 3-coloring problem that determines whether the graph G is a valid 3-coloring.

   Theorem: If $SATCNF$ has a polynomial time algorithm then $3COLOR$ has a polynomial time algorithm.

   Proof: In order to prove this theorem, we must show that $3COLOR \leq SATCNF$. That is, we must derive a polynomial algorithm for $3COLOR$ that calls $SATCNF$. This requires us to transform the input of $3COLOR$ (a graph G) into a Boolean formula F in conjunctive normal form.

   First, we realize that each vertex in G can be assigned one of three colors. Thus, for each vertex $v_i \in G$, create three variables $v_i^{(a)}, v_i^{(b)}$ and $v_i^{(c)}$ cooresponding to the three possible colorings of v_i. We create a clause that represents this vertex:

   $$(v_i^{(a)} \vee v_i^{(b)} \vee v_i^{(c)}) \tag{1}$$

   Note that this clause is true if *at least* one variable is true. However, in order to accurately represent $3COLOR$, this clause should only be true if *exactly* one variable is true. This comes from the fact that each vertex in a graph can only be assigned one color. Thus, we add a few more clauses that should be ANDed to clause (1):

$$(\bar{v}_i^{(a)} \vee \bar{v}_i^{(b)}) \wedge (\bar{v}_i^{(b)} \vee \bar{v}_i^{(c)}) \wedge (\bar{v}_i^{(a)} \vee \bar{v}_i^{(c)}) \tag{2}$$

The only way (2) can be true is if exactly one of the variables $v_i^{(a)}, v_i^{(b)}$ or $v_i^{(c)}$ is true.

This takes care of each vertex individually; however, we must also account for adjacent vertices. That is, a graph is only a valid 3-coloring if no pair of adjacent vertices are colored the same color. Given two adjacent vertices v_i and v_j, this translates to Boolean logic like so:

$$\overline{(v_i^{(x)} \wedge v_j^{(x)})} = (\bar{v}_i^{(x)} \vee \bar{v}_j^{(x)}) \tag{3}$$

where vertices v_i and v_j both are assigned color x.

Thus, for each pair of adjacent vertices v_i and v_j in G, the following clauses should be ANDed to clauses (1) and (2):

$$(\bar{v}_i^{(a)} \vee \bar{v}_j^{(a)}) \wedge (\bar{v}_i^{(b)} \vee \bar{v}_j^{(b)}) \wedge (\bar{v}_i^{(c)} \vee \bar{v}_j^{(c)}) \tag{4}$$

The only way (4) can be true is if no two adjacent vertices have the same color. Otherwise, at least one of these clauses will be false.

In summary, the input G can be transformed to a CNF Boolean formula by creating three variables for every vertex $v_i \in G$. Each vertex v_i is represented by four clauses, and each pair of adjacent vertices v_i and v_j is represented by three additional clauses. All of these clauses should be ANDed together:

$$F = \left(\bigwedge_{v_i \in G} (1)_i \wedge (2)_i \right) \wedge \left(\bigwedge_{v_i, v_j \in G} (4)_{i,j} \right) \tag{5}$$

Clearly this transformation can happen in polynomial time. And the output of $SATCNF(F)$ will be true if and only if the graph G is a valid 3-coloring.

2. **Reduction Problem 18:**

   Let $HAMP$ be the Fixed Hamiltonian Path Problem that determines if there is a simple path between two vertices x and y in an undirected graph $G = (V_G, E_G)$ that spans all the vertices in G. Let $HAMC$ be the Hamiltonian Cycle Problem that determines whether or not an undirected graph $H = (V_H, E_H)$ has a Hamiltonian cycle. A Hamiltonian cycle is a simple cycle that spans H.

Theorem: If the $HAMP$ has a polynomial time algorithm then $HAMC$ has a polynomial time algorithm

Proof: In order to prove this theorem, we must show that $HAMC \leq HAMP$. That is, we must derive a polynomial algorithm for $HAMC$ that calls $HAMP$. This requires us to transform the input of $HAMC$ (an undirected graph H) to the input of $HAMP$ (an undirected graph G, and two vertices x and y).

This transformation involves iterating through each edge $e = (x, y) \in H$ where x and y are the vertices that comprise this edge. Using these two vertices as input to $HAMP$, we realize that if there exists a Fixed Hamiltonian path, then this edge e can not exist in it. This follows from the definition of a Fixed Hamiltonian path; specifically, given that x must be visited first, y must be visited last and no vertex can be visited more than once, edge e will never be traveled. Otherwise, y would be visited twice.

(Note that if the graph only contains two vertices x and y, edge e must be traveled. However, this graph will always have a Hamiltonian cycle, so this edge case is not a problem.)

The reduction is described in the pseudocode below:

```
problem HAMC(G)
H = G
for each e = (x, y) ∈ E_H do                    ▷ for each edge in H
    if HAMP(H, x, y) then                       ▷ if there is a path from x to y
        return true                             ▷ there exists a cycle with inclusion of e
    end if
end for
```

Clearly if $HAMP$ can be solved in polynomial time, this algorithm can be solved in polynomial time.