4005-800 Algorithms

Homework 7

Christopher Wood

May 11, 2012

PROBLEM 1 - 34.2-1. *Consider the language GRAPH-ISOMORPHISM* $= \{\langle G_1, G_2 \rangle : G_1$ *and* G_2 *are isomorphic graphs* $\}$. *Prove that GRAPH-ISOMORPHISM* $\in NP$ *by describing a polynomial-time algorithm to verify the language.*

Solution. By the defintion of graph isomorphism, two graphs G_1 and G_2 are isomorphic if and only if there exists a bijection $m : V(G_1) \to V(G_2)$ such that any two vertices v_i and v_j of G_1 are adjacent in G_1 if and only if $m(v_i)$ and $m(v_j)$ are adjacent in G_2. With this definition, it is enough to check the bijection m to see if it fulfills this property to verify that two graphs isomorphic. We can easily devise a polynomial-time algorithm to solve this as follows:

Algorithm 1: GRAPH-ISOMORPHISM-VERIFIER

function VerifyGraphIsomorphism(m)
    for all $v_i \in V(G_1)$ **do**
        vCount = 0 ▷ Count number of times v_i appears in G_2
        for all $v_j \in V(G_2)$ **do**
            if $m(v_i) == v_j$ **then**
                vCount = vCount + 1
            end if
        end for
        if vCount $\neq 1$ **then** ▷ v_i should only appear once in G_2
            return False
        end if
    end for

    for all $v_i \in V(G_1)$ **do**
        for all $v_j \in V(G_1)$ **do**
            if $(v_i, v_j) \in E(G_1)$ and $(m(v_i), m(v_j)) \notin E(G_2)$ **then**
                return False
            end if
        end for
    end for

    for all $v_i \in V(G_1)$ **do**

```
        for all v_j ∈ V(G_1) do
            if (m(v_i), m(v_j)) ∈ E(G_2) and (v_i, v_j) ∉ E(G_1) then
                return False
            end if
        end for
    end for

    return True
end function
```

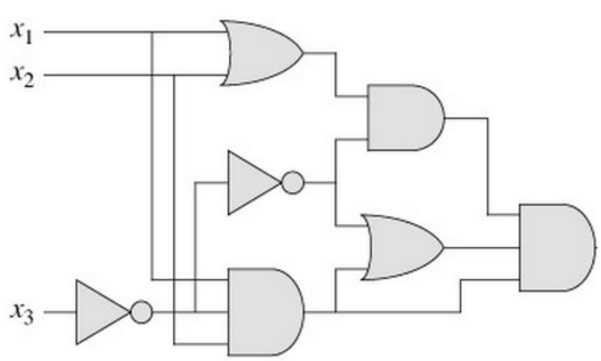
Note that m denotes the bijective mapping (the permutation) from $V(G_1)$ to $V(G_2)$. It is clear that the permutation check runs in $O(V)$ time and the edge check runs in $O(V^2)$ time. Thus, we conclude that this algorithms runs in $O(V^2)$ time and thus verifies the solution (i.e. the permutation mapping m) to the GRAPH-ISOMORPHISM problem in polynomial time.

PROBLEM 2 - 34.2-10. *Prove that if $NP \neq$ co-NP, then $P \neq NP$.*

Solution. If $NP \neq$ *co*-NP, then we know there exists a problem $Q \in NP$ such that $Q \notin$ *co*-NP. Furthermore, by definition, we know that $P \in$ *co*-$NP \cap NP$. Now, let Q be a problem in NP that is not in *co*-NP. By the definition of the set intersection, that means that $Q \notin$ *co*-$NP \cap NP$, and thus we know that $Q \in NP$ and $Q \notin P$ (becuse P is a subset of *co*-$NP \cap NP$). Therefore, since there exists a problem that is in NP but not in P, we conclude that $P \neq NP$.

PROBLEM 3 - 34.3-1. *Verify that the circuit in Figure 34.8(b) is unsatisfiable.*

Solution.



The logical equivalent expression for this circuit is as follows:

$$((x_1 \vee x_2) \wedge x_3) \wedge (x_3 \vee (x_1 \wedge x_2 \wedge \neg x_3)) \wedge (x_1 \wedge x_2 \wedge \neg x_3)$$

x_1, x_2, x_3	$x_1 \vee x_2$	$(x_1 \vee x_2) \wedge x_3$	$x_1 \wedge x_2 \wedge \neg x_3$	$x_3 \vee (x_1 \wedge x_2 \wedge \neg x_3)$	Final AND Gate
F,F,F	F	F	F	F	F
F,F,T	F	F	F	T	F
F,T,F	T	F	F	F	F
F,T,T	T	T	F	T	F
T,F,F	T	F	F	F	F
T,F,T	T	T	F	T	F
T,T,F	T	F	T	T	F
T,T,T	T	T	F	T	F

Table 1: Truth table for problem #3, where T = True and F = False.

To show that this circuit is unsatisfiable, we simply build a truth table for the boolean expression that considers all logical values for x_1, x_2, and x_3, as shown in table :

Therefore, since there is no possible combination of logical values for x_1, x_2, and x_3 such that the boolean expression is true, we conclude that it is unsatisfiable.

PROBLEM 4 - 34.4-5. *Show that the problem of determining the satisfiability of boolean formulas in disjunctive normal form is polynomial-time solvable.*

Solution. We show that the problem of determining the satisfiability of boolean formulas in disjunctive normal form is polynomial-time solvable by providing a polynomial-time algorithm that performs this task. This algorithm is realized below in Algorithm 2.

Algorithm 2: DNF-SOLVER

```
function SolveDNF(ψ)
    for all v_i ∈ V(G_1) do
        vCount = 0                              ▷ Count number of times v_i appears in G_2
        for all v_j ∈ V(G_2) do
            if m(v_i) == v_j then
                vCount = vCount + 1
            end if
        end for
        if vCount ≠ 1 then                      ▷ v_i should only appear once in G_2
            return False
        end if
    end for

    return True
end function
```

PROBLEM 5 - 34.5-5. *The* ***set-partition problem*** *takes as input a set S of numbers. The question is whether the numbers can be partitioned into two sets A and $A' = S - A$ such that $\sum_{x \in A} x = \sum_{x \in A'} x$. Show that the set-partition problem is NP-complete.*

Solution. In order to show that the set-partition problem, Q is NP-complete, we show that it reduces to the subset-sum problem, Q'. That is, we prove $Q \leq_p Q'$.

PROBLEM 6. *TODO*

Solution.

ALGORITHM 3: RECURSIVE-KNAPSACK

```
TODO
```
