CSCI 2210: Theory of Computation

Problem Set 7 (due 12/6)

Student: Rafael Almeida *Collaborators:* Cal Thompson

Problem 1. Show that the following languages are in the class P.

(a) TRIANGLE $= \{\langle G\rangle \mid G$ is an undirected graph that contains a clique of size $3\}$

G is represented as an adjacency list

```
M = "on input G:
    1. For all vertex V in G:
        2. For neighbor N of V:
            3. If {neighbors of V} ∩ {neighbors of N} ≠ ∅:
                4. **Accept**
    5. **Reject**"
```

Above is a decider for TRIANGLE that runs in polynomial time, O(|V| + |E|^2) where V are the vertices and E are the edges in G.

It works because if V shares a neighbor with N that means a triangle (clique of 3) exists

(b) CONNECTED $= \{\langle G\rangle \mid G$ is an undirected graph and G is connected$\}$

```
M = "on input G:
    1. Choose a vertex V in G
    2. Mark all nodes reachable from V
    3. Traverse all nodes in G
        4. If a node is unmarked, **reject**
    5. **Accept**"
```

Above is a decider for TRIANGLE that runs in polynomial time, O(|V| + |E|) where V are the vertices and E are the edges in G.

It works because if there are unmarked nodes it means not all nodes were reachable from the arbitrary vertex V. Therefore the graph is not connected.

Problem 2. Show that class P is closed under complement.

If a problem A is in P, then there is a halting TM M_A for A that runs in polynomial time. Create another halting TM M_CA using M_A that decides the complement of A.

```
M_CA = "on input W of format M_A:
        1. Simulate M_A on W
             2. If M_A accepts, reject. Otherwise, accept"
```

Above is a decider for the complement of A that runs in polynomial time.
Therefore, class P is closed under complement.

Problem 3. Consider the following variation of the SAT problem.

$$\text{DOUBLE-SAT} = \{\langle\phi\rangle \mid \phi \text{ is CNF formula and it has at least two satisfying assignments}\}$$

(a) Give an example of a formula $\phi \in$ DOUBLE-SAT.

Using the symbol "!" as negation.

(x or !x)

(b) Give an example of a formula $\phi \notin$ DOUBLE-SAT.

(x or !y) and (!x or y) and (!x or !y)

Only satisfiable assignment is X = F and Y = F

(c) Show that DOUBLE-SAT is NP-complete.

To verify a solution **ϕ** for **DOUBLE-SAT**, iterate over the **m** clauses and plug in the assignment to the **n** literals in each and evaluate the clauses. Verify that all clauses are true. This algorithm runs in **O(nm)**. Therefore, **DOUBLE-SAT** is **NP**

3SAT ≤*p* DOUBLE-SAT
Use the following reduction:

f(ϕ) = ϕ and (x or y) and (!x or !y)
where x and y are new variables in ϕ

Essentially adding to the original formula the equivalent of **(x XOR y)**, which maintains the satisfiability of **ϕ** while adding two new potential satisfiable assignments, namely **1) x = T, y = F 2) x = F, y = T**

Consider **S(ϕ) = the number of satisfiable assignments for ϕ**

If **ϕ** ∈ **3SAT**, **S(ϕ) ≥ 1**. Therefore, since **S(x XOR y) = 2**, **S(f(ϕ)) ≥ 3**, which means **f(ϕ)** ∈ **DOUBLE-SAT**

If **ϕ** ∉ **3SAT**, **S(ϕ) = 0** which means at least one of the clauses of **ϕ** are always false. Thus, **S(f(ϕ)) = 0** because the "and" added by f ensures that, because **ϕ** did not have any satisfiable assignments, **f(ϕ)**, despite having two new satisfiable clauses, will not have any satisfiable assignments since one of the clauses originally in **ϕ** are going to be false.

Problem 4. Let U be a finite set and $\mathcal{S}$ a collection of nonempty subsets of U, i.e., $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$ and $S_i \subseteq U$.

$$\textsf{HittingSET} = \{\langle U, \mathcal{S}, k\rangle \mid \text{there exists a set } H \subseteq U \text{ with } k \text{ elements such that for all } S_i \in \mathcal{S} \text{ we have } H \cap S_i \neq \emptyset\ \}$$

Show that HittingSET is NP-complete.

To verify a solution **H** to **HittingSet**, iterate over the **|S|** sets in **S** and verify that **H** ∩ **S_i** is not empty**.** This algorithm runs in **O(|S| * n)** where **n** is the max sized set in **{H}** ∪ **S.** Therefore, **HittingSet** is NP

VCOVER ≤*p* HittingSET
Use the following reduction:

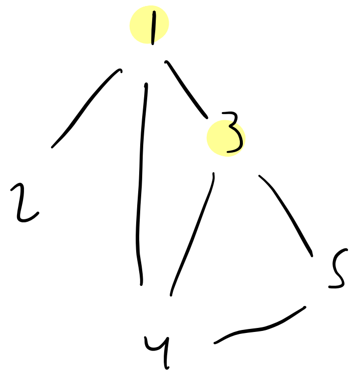
f(G, k) -> U, S, k
  U = **{**vertex **v | v** ∈ **G}**
  S = **{{v}** ∪ **(**neighbors of **v) | v** ∈ **G}**
  k

Consider **C(G, k) = {**the vertices in a vertex cover of size **k** for **G,** if any**}**

If **<G, k>** ∈ **VCOVER** then **<U, S, k>** ∈ **HittingSET** because at least one of the nodes in **C(G)** will be present in each set of **S.** This is true because since **{C(G)}** ∪ **{**neighbors of **v | v** ∈ **C(G)} = all vertices of G**, meaning all nodes in **G** are at most one away from (is a neighbor of) some node in **C(G)**, and since the sets in **S** represents a vertex and its neighbors, all of the sets will contain at least one vertex from **C(G)**

If **G, k** ∉ **VCOVER** then **U, S, k** ∉ **HittingSET** because at least one set of **S** will not share any nodes with any attempt of a **C(G)** of size **k.** This is true because since **C(G)** is incomplete, it means there is some vertex in **G** that is more than one away from all vertices in **C(G)**, and since each set in **S** represents a vertex and its neighbors, picking **C(G)** as the hitting set would mean some set in **S** has nothing in common with this **C(G)**

Example



1 : 2 3 4

2 : 1

3 : 1 4 5

4 : 1 3 5

5 : 3 4