Analysis of Algorithms

Homework 7 – *NP*

Abraham Murciano

January 22, 2021

1 Subgraph Isomorphism

We are given two graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. The subgraph isomorphism problem (SGI) is to determine if there exists a subgraph of G_2 which is isomorphic to G_1.

Part A

To prove that SGI $\in NP$ we must show that there is a way to verify a solution to the problem in polynomial time.

If we are given a solution to an SGI problem in the form of $G_3 = (V_3, E_3)$, which is a subgraph of G_2; and a bijection $f : V_1 \to V_3$,which is a map from the vertices of G_1 to the vertices of G_3, then we can verify that they are in fact isomorphic as follows.

```
function ISOMORPHIC(G_1, G_3, f)
    for (u, v) ∈ E_1 do
        if (f(u), f(v)) ∉ E_3 then return false
    return true
```

This algorithm is clearly $O(E)$, so is polynomial. Therefore $SCI \in NP$.

Part B

To prove that SGI is *NP*-complete, we will show that the Clique problem (which is known to be *NP*-complete) is reducible to SGI.

The Clique problem is to determine, given a graph $G = (V, E)$ and a constant $c \leq |V|$, whether or not K_c (a complete graph with c vertices) is a subgraph of G.

To reduce the Clique problem to SGI, we must convert every instance of the Clique problem into one of SGI, such that the solution will be the same for both.

Consider some inputs to the Clique problem; a graph G, and a constant c. We can feed K_c and G as the inputs G_1 and G_2 of the SGI problem respectively. Then any SGI algorithm will tell us whether or not $G = G_2$ has a subgraph isomorphic to $K_c = G_1$, which is precisely the definition of the Clique problem.

2 Simple Cycle Problem

This problem is to find whether or not a given graph has a simple cycle (a cycle with no repeating vertices, other than the first and last) of k distinct vertices.

Part A

This problem is in *NP*. This can be easily shown since if we are given a cycle, we can walk the cycle and verify that there are precisely k vertices in linear time, and we can verify that they are all distinct in linear time with the use of a hash-set, or certainly in quadratic time by comparing all vertices to all previously seen ones. Therefore the problem is in *NP*.

Part B

We will now show that our problem is *NP*-complete by showing that the Hamiltonian Cycle problem, which is known to be *NP*-complete reduces to it.

The Hamiltonian Cycle problem goes as follows. Given a graph, is there a path that traverses all the vertices without repeating any, then ends back at the start?

For any instance of the Hamiltonian Cycle problem, supposing the input is the graph $G = (V, E)$, we can construct an equivalent Simple Cycle problem by asking is there a simple cycle in G of $k = |V|$ distinct vertices. If there is one, that is a hamiltonian cycle. Otherwise, there is no hamiltonian cycle. Thus the Simple Cycle problem is also *NP*-complete.

Part C

We are presented with another similar problem. Given a graph, does it contain *any* cycles? This problem is different to the Simple Cycle problem, in that we do not care about the size of the cycle, nor do we care if there are smaller cycles within it.

This problem can be solved in polynomial time by performing a breadth-first search on the tree, and if back-edges are found, then there must be a cycle. Otherwise there cannot be one.

3 Cliques and 3-CNF

We are given the following boolean formula in 3-CNF.

$$\varphi = (a \vee b \vee \neg c) \wedge (a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee c) \wedge (a \vee \neg b \vee \neg c)$$

In the reduction from a Clique problem to 3-CNF, the graph in figure 1 would reduce to φ to serve as input into the 3-CNF problem.

Figure 1: A graph which reduces to φ when reducing to 3-CNF.