Lindsey Bieda and Joe Frambach
Dynamic Programming Problems
10.19.2011

5. Show that if one of the following three problems has a polynomial time algorithm then they all do.

- The Independent Set Problem: The input is a graph G. The problem is to find the largest independent set in G. In an independent set all vertices are mutually nonadjacent.
- The Clique Problem: The input is a graph G. The problem is to find the largest clique in G. In a clique all vertices are mutually adjacent.
- The Vertex Cover Problem: The input is a graph G. The problem is to find the smallest vertex cover in G. A set S is a vertex cover if each edge in G is incident to a vertex in S.

Independent Set $\leq$ Clique

```
Program Independent Set:
    read G
    Construct $G'$ with the same vertices
    For each vertex pair: $(v_1, v_2) \ni v_1 \neq v_2$ This will take $O(n^2)$ time to check all pairs
        If there is no edge from $v_1$ to $v_2$ in $G$:
            Create edge from $v_1$ to $v_2$ in $G'$
    output Clique($G'$)
```

Since the independent set wants only non-adjacent nodes we can generate edges between all non-adjacent nodes and then find the clique of that in order to determine which ones are in the independent set, since the state of being a clique is the inverse of being an independent set.

Clique $\leq$ Vertex Cover:

```
Program Clique:
    read $G$
    Construct $G'$ with the same vertices
    For each vertex pair: $(v_1, v_2) \ni v_1 \neq v_2$ This will take $O(n^2)$ time to check all pairs
        If there is no edge from $v_1$ to $v_2$ in $G$:
            Create edge from $v_1$ to $v_2$ in $G'$
    $S$ = Set of vertices in $G$
    $S'$ = VertexCover($G'$)
    output $S$ - $S'$
```

Since the independent set of the inverse of G will try to create the smallest independent set, we know that there must not be an edge between the two edges in that set. We also know that there must then be edges between all of the other remaining nodes and, therefore, there is a clique.

Vertex Cover $\leq$ Independent Set

```
Program Vertex Cover:
    read $G$
    $S$ = IndependentSet($G$)
    output $G$ - $S$ This will take $O(n)$ time to perform set subtraction
```

Since we have the largest Independent Set we know that the items in this set all must not be adjacent to each other, therefore, the smallest possible vertex cover must contain all vertices not in this set. If one vertices in this set was in the smallest vertex cover then there would be two adjacent nodes in the vertex cover and it would no longer be the smallest one.

8. Show that the subset sum problem is self-reducible. The decision problem is to take a collection of positive integers $x_1, \ldots, x_n$ and an integer L and decide if there is a subset of the x_is that sum to L. The optimization problem asks you to return the actual subset if it exits. So you must show that if the decision problem has a polynomial time algorithm then the optimization problem also has a polynomial time algorithm.

subset sum optimization $\leq$ subset sum decision.

```
program subset sub optimization:
    read $x_1, \ldots, x_n$, $L$
    If $x_1, \ldots, x_n$, $L$ decision is 1 then:
        S = $[x_1, \ldots, x_n]$
        foreach $x$ in $S$:
            if $S \backslash x$, $L$ decision is 1 then: *if the subset sum is still possible without x remove it*
                remove $x$ from $S$
    output $S$
```

We examine if each of the numbers is required to sum to the desired L. If it is not required then we can remove it from the set.

12. Consider the following 2Clique problem:
INPUT: A undirected graph G and an integer k.
OUTPUT: 1 if G has two vertex disjoint cliques of size k, and 0 otherwise.
Show that this problem is NP-hard. Use the fact that the clique problem in NP-complete. The input to the clique problem is an undirected graph H and an integer j. The output should be 1 if H contains a clique of size j and 0 otherwise. Note that a clique is a mutually adjacent collection of vertices. Two cliques are disjoint if they do not share any vertices in common.

Clique $\leq$ 2-Clique

```
Program Clique:
    read $H$, $j$
    G = H + H, where both copies of H are disjoint
    output 2-Clique($G$,$j$)
```

Therefore, 2Clique is NPH.

If there is a clique in H, there must be two cliques in a graph made up of two disjoint copies of H.