Lindsey Bieda and Joe Frambach
Dynamic Programming Problems
10.28.2011

11. Show that the Vertex Cover Problem is self-reducible. The decision problem is to take a graph G and an integer k and decide if G has a vertex cover of size k or not. The optimization problem takes a graph G, and returns a smallest vertex cover in G. So you must show that if the decision problem has a polynomial time algorithm then the optimization problem also has a polynomial time algorithm. Recall that a vertex cover is a collection S of vertices with the property that every edge is incident to a vertex in S.

VERTEXCOVER-OPT $\leq$ VERTEXCOVER-DEC

```
Program VERTEXCOVER-OPT:
    Read graph $G$, integer $k$.
    First, find the smallest size of the smallest vertex cover.
    For $k = 1$ to $n$:
        If VERTEXCOVER-DEC($G$,$k$) returns true:
            size $s = k$.
            End for-loop.
    Next, find the actual cover by removing the vertices NOT in the cover
    For each vertex $v$ in $G$:
    $G' = G - v$
        If VERTEXCOVER-DEC($G'$, $s$) returns true:
            $G = G'$.
    G is the graph containing only the smallest cover.
    Return $G$.
```

14. Consider the problem where the input is a collection of linear inequalities. For example, the input might look like $3x - 2y \leq 3$ and $2x - 3y \geq 9$. The problem is to determine if there is an integer solution that simultaneously satisfies all the inequalities. Show that this problem is NP-hard using the fact that it is NP-hard to determine if a Boolean formula in conjunctive normal form is satisfiable.

CNF-SAT $\leq$ LINEAR-INEQ

```
Program CNF-SAT:
    Read formula $F$
    $F'$ = ConstructEquation($F$)
    Output LINEAR-INEQ($F'$)
```

ConstructEquation(F):
Separate the CNF formula by the conjuncts to create n separate statements.
For each of these n statements constuct one side of the inequality by summing each of the positive literals (now variables) and for the negated literals make them (1 - *literal*). Make this sum ≥ 1. This is done to enforce that at least one of the literals must be true in order to satisfy this part of the formula.

For each of the literals add the following inequality $0 \leq literal \leq 1$. The purpose of this is to ensure that each literal can only have a value of 0 or 1 relating to the possibility of it bring true or false in the formula.
Ex: $x \vee y \vee \neg z$ becomes:
$x + y + (1 - z) \geq 1$ and
$0 \leq x \leq 1$,
$0 \leq y \leq 1$,
$0 \leq z \leq 1$

The fact that the formula is satifiable if and only if the linear inequality has a solution that satisfies all inequalities is obvious from the above construction.

16. The input to the three coloring problem is a graph G, and the problem is to decide whether the vertices of G can be colored with three colors such that no pair of adjacent vertices are colored the same color. The input to the four coloring problem is a graph G, and the problem is to decide whether the vertices of G can be colored with four colors such that no pair of adjacent vertices are colored the same color. Show by reduction that if the four coloring problem has a polynomial time algorithm then so does the three coloring problem.

3-Coloring $\leq$ 4-Coloring

```
Program 3-Coloring:
    Read $G$
    $G'$ = G + node *Where the added node connects to all other nodes in the graph*
    Output 4-Coloring($G'$)
```

Since the added node connects to all other nodes in the graph, it forces that additional color to be used. The only way the graph could be 4 colorable is if it was 3 colorable, otherwise, this added node is the same color as one of the nodes it is connected it.