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 literals (now variables) and multiplying those variables by -1 that are the negated literals. Make this sum $\leq -1 * (the\ number\ of\ literals) + 1$.

Ex: $x \vee y \vee \neg z$ becomes $x + y - z \leq -2$
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.