CS 170 Homework 11

Due **Friday 11/15/2024, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, explicitly write "none".

2 Decision vs. Search vs. Optimization

Recall that a vertex cover is a set of vertices in a graph such that every edge is adjacent to at least one vertex in this set.

The following are three formulations of the VERTEX COVER problem:

- As a *decision problem*: Given a graph G, return TRUE if it has a vertex cover of size at most b, and FALSE otherwise.
- As a *search problem*: Given a graph G, find a vertex cover of size at most b (that is, return the actual vertices), or report that none exists.
- As an *optimization problem*: Given a graph G, find a minimum vertex cover.

At first glance, it may seem that search should be harder than decision, and that optimization should be even harder. We will show that if any one can be solved in polynomial time, so can the others.

(a) Suppose you are handed a black box that solves VERTEX COVER (DECISION) in polynomial time. Give an algorithm that solves VERTEX COVER (SEARCH) in polynomial time.

(b) Similarly, suppose we know how to solve VERTEX COVER (SEARCH) in polynomial time. Give an algorithm that solves VERTEX COVER (OPTIMIZATION) in polynomial time.

Solution:

(a) If given a graph G and budget b, we first run the DECISION algorithm on instance (G, b). If it returns "FALSE", then report "no solution".

If it comes up "TRUE", then there is a solution and we find it as follows:

- Pick any node $v \in G$ and remove it, along with any incident edges.
- Run DECISION on the instance $(G \setminus \{v\}, b - 1)$; if it says "TRUE", add v to the vertex cover. Otherwise, put v and its edges back into G.
- Repeat until G is empty.

Correctness: If there is no solution, obviously we report as such. If there is, then our algorithm tests individual nodes to see if they are in any vertex cover of size b (there may be multiple). If and only if it is, the subgraph $G \setminus \{v\}$ must have a vertex cover no larger than $b-1$. Apply this argument inductively.

Running time: We may test each vertex once before finding a v that is part of the b-vertex cover and recursing. Thus we call the DECISION procedure $O(n^2)$ times. This can be tightened to $O(n)$ by not considering any vertex twice. Since a call to DECISION costs polynomial time, we have polynomial complexity overall.

Note: this reduction can be thought of as a greedy algorithm, in which we discover (or eliminate) one vertex at a time.

(b) Binary search on the size, b, of the vertex cover.

Correctness: This algorithm is correct for the same reason as binary search.

Running time: The minimum vertex cover is certainly of size at least 1 (for a nonempty graph) and at most $|V|$, so the SEARCH black box will be called $O(\log |V|)$ times, giving polynomial complexity overall.

Finally, since solving the optimization problem allows us to answer the decision problem (think about why), we see that all three reduce to one another!

3 Some Sums

Given an array $A = [a_1, a_2, \dots, a_n]$ of nonnegative integers, consider the following problems:

1 **Partition**: Determine whether there is a subset $S \subseteq [n]$ ($[n] := \{1, 2, \cdots, n\}$) such that $\sum_{i \in S} a_i = \sum_{j \in ([n] \setminus S)} a_j$. In other words, determine whether there is a way to partition A into two disjoint subsets such that the sum of the elements in each subset equal.

2 **Subset Sum**: Given some integer x, determine whether there is a subset $S \subseteq [n]$ such that $\sum_{i \in S} a_i = x$. In other words, determine whether there is a subset of A such that the sum of its elements is x.

3 **Knapsack**: Given some set of items each with weight w_i and value v_i, and fixed numbers W and V, determine whether there is some subset $S \subseteq [n]$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i \geq V$.

For each of the following clearly describe your reduction and justify its correctness.

(a) Find a linear time reduction from Subset Sum to Partition.

(b) Find a linear time reduction from Subset Sum to Knapsack.

Solution:

(a) Suppose we are given some A with target sum x. Let s be the sum of all elements in A. If $s - 2x \geq 0$, generate a new set $A' = A \cup \{s - 2x\}$. If A' can be partitioned, then there is a subset of A that sums to x.

We know that the two sets in our partition must each sum to $s - x$ since the sum of all elements will be $2s - 2x$. One of these sets, must contain the element $s - 2x$. Thus the remaining elements in this set sum to x.

If $s - 2x \leq 0$, generate a new set $A' = A \cup \{2x - s\}$. If A' can be partitioned, then there is a subset of A that sums to x.

We know that the two sets in our partition must each sum to x since the sum of all elements will be $2x$. The set that does not contain $\{2x - s\}$ will be our solution to subset sum.

(b) Suppose we are given some set A with target sum x. For each element a_i of the set, create an item with weight a_i and value a_i. Let $V = x$ and $W = x$. We know Knapsack will determine if there is a combination of items with sum of weights $\leq x$ and values $\geq x$. Because the weights and values are the same, we know (Sum of chosen weights) $=$ (Sum of chosen values) $= x$. And since each weight/value pair is exactly the value of one of the original elements of A, we know that there will be a solution to our Knapsack problem iff there is one for our subset sum problem.

4 Reduction to 3-Coloring

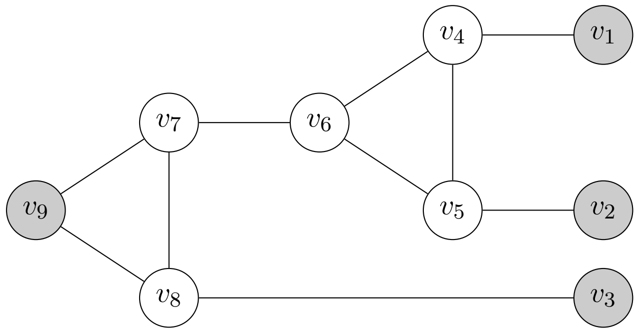
Given a graph $G = (V, E)$, a valid 3-coloring assigns each vertex in the graph a color from $\{\text{red, green, blue}\}$ such that for any edge (u, v), u and v have different colors. In the 3-coloring problem, our goal is to find a valid 3-coloring if one exists. In this problem, we will give a reduction from 3-SAT to the 3-coloring problem. Since we know that 3-SAT is NP-Hard (there is a reduction to 3-SAT from every NP problem), this will show that 3-coloring is NP-Hard (there is a reduction to 3-coloring from every NP problem).

In our reduction, the graph will start with three special vertices, labelled v_{TRUE}, v_{FALSE}, and v_{BASE}, as well as the edges $(v_{\mathsf{TRUE}}, v_{\mathsf{FALSE}})$, $(v_{\mathsf{TRUE}}, v_{\mathsf{BASE}})$, and $(v_{\mathsf{FALSE}}, v_{\mathsf{BASE}})$.

(a) For each variable x_i in a 3-SAT formula, we will create a pair of vertices labeled x_i and $\neg x_i$. How should we add edges to the graph such that in any valid 3-coloring, one of $x_i, \neg x_i$ is assigned the same color as v_{TRUE} and the other is assigned the same color as v_{FALSE}?

Hint: any vertex adjacent to v_{BASE} must have the same color as either v_{TRUE} or v_{FALSE}. Why is this?

(b) Consider the following graph, which we will call a "gadget":



Consider any valid 3-coloring of this graph that does *not* assign the color red to any of the gray vertices (v_1, v_2, v_3, v_9). Show that if v_9 is assigned the color blue, then at least one of $\{v_1, v_2, v_3\}$ is assigned the color blue.

Hint: it's easier to prove the contrapositive!

(c) We have now observed the following about the graph we are creating in the reduction:

(i) For any vertex, if we have the edges (u, v_{FALSE}) and (u, v_{BASE}) in the graph, then in any valid 3-coloring u will be assigned the same color as v_{TRUE}.

(ii) Through brute force one can also show that in a gadget, if all the following hold:

(1) All gray vertices are assigned the color green or blue.

(2) v_9 is assigned the color blue.

(3) At least one of $\{v_1, v_2, v_3\}$ is assigned the color blue.

Then there is a valid coloring for the white vertices in the gadget.

Using these observations and your answers to the previous parts, **give a reduction from 3-SAT to 3-coloring. Prove that your reduction is correct (you do not need to prove any of the observations above).**

Hint: create a new gadget per clause!

Solution:

(a) We add the edges $(x_i, \neg x_i)$, (x_i, v_{BASE}) and $(\neg x_i, v_{\mathsf{BASE}})$. Since $x_i, \neg x_i$ are both adjacent to v_{BASE} they must be assigned a different color than v_{BASE}, i.e. they both are assigned either the color of v_{TRUE} or the color of v_{FALSE}. Since we added an edge between x_i and $\neg x_i$, they can't be assigned the same color, i.e. one is assigned the same color as v_{TRUE} and one the same color as v_{FALSE}.

(b) It is easier to show the equivalent statement that if all the gray vertices on the right are assigned the color green, then the gray vertex on the left must be assigned the color green as well. Consider the triangle on the right. Since all the gray vertices are assigned green, the two right points must be assigned the colors red and blue, and so the left point in this triangle must be assigned green in any valid coloring. We can repeat this logic with the triangle on the left, to conclude that the gray vertex on the left must be assigned green in any valid coloring.

(c) Given a 3-SAT instance, we create the three special vertices and edges described in the problem statement. As in part a, we create vertices x_i and $\neg x_i$ for each variable x_i, and add the edges we gave in the answer to part a. For clause j, we add a vertex C_j and edges $(C_j, v_{\mathsf{FALSE}})$, (C_j, v_{BASE}). Lastly, for clause j we add vertices and edges to create a gadget where the three gray vertices on the right of the gadget are the vertices of three literals in the clause, and the gray vertex on the left is the vertex C_j (All white vertices in the gadget are only used in this clause's gadget).

If there is a satisfying 3-SAT assignment, then there is a valid 3-coloring in this graph as follows. Assign v_{FALSE} the color green, v_{TRUE} the color blue, and v_{BASE} the color red; assign x_i the color blue if x_i is v_{TRUE} and green if x_i is v_{FALSE} (vice-versa for $\neg x_i$). Assign each C_j the color blue. Lastly, fix any gadget. Since the 3-SAT assignment is satisfying, in each gadget at least one of the gray vertices on the right is assigned blue, so by the observation (ii) in the problem statement the gadget can be colored.

If there is a valid 3-coloring, then there is a satisfying 3-SAT assignment. By symmetry, we can assume v_{FALSE} is colored green, v_{TRUE} is colored blue, and v_{BASE} is colored red. Then for each literal where x_i is color blue, that literal is true in the satisfying assignment. By part a, we know that exactly one of $x_i, \neg x_i$ is colored blue, so this produces a valid assignment. By observation (i), we also know every node C_j must be colored blue. All literal nodes are colored green or blue, so by part b, this implies that for every clause, one of the gray literal nodes in the clause gadget is colored blue, i.e. the clause will be satisfied in the 3-SAT assignment.