CS 170 Homework 12

Due **Monday 11/20/2023, 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, you must explicitly write "none".

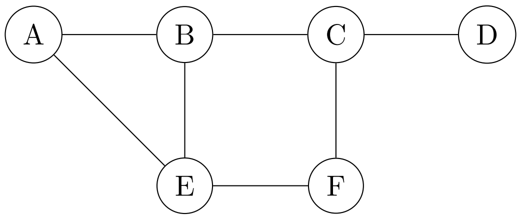
2 Vertex Cover to Set Cover

To help jog your memory, here are some definitions:

Vertex Cover: given an undirected unweighted graph $G = (V, E)$, a vertex cover C_V of G is a subset of vertices such that for every edge $e = (u, v) \in E$, at least one of u or v must be in the vertex cover C_V.

Set Cover: given a universe of elements U and a collection of sets $\mathcal{S} = \{S_1, \ldots, S_m\}$, a set cover is any (sub)collection C_S whose union equals U.

In the *minimum vertex cover problem*, we are given an undirected unweighted graph $G = (V, E)$, and are asked to find the smallest vertex cover. For example, in the following graph, $\{A, E, C, D\}$ is a vertex cover, but not a minimum vertex cover. The minimum vertex covers are $\{B, E, C\}$ and $\{A, E, C\}$.



Then, recall in the *minimum set cover problem*, we are given a set U and a collection $\mathcal{S} = \{S_1, \ldots, S_m\}$ of subsets of U, and are asked to find the smallest set cover. For example, given $U := \{a, b, c, d\}$, $S_1 := \{a, b, c\}$, $S_2 := \{b, c\}$, and $S_3 := \{c, d\}$, a solution to the problem is $C_S = \{S_1, S_3\}$.

Give an efficient reduction from the minimum vertex cover problem to the minimum set cover problem. Briefly justify the correctness of your reduction (i.e. 1-2 sentences).

Solution:

Algorithm Description: Let $G = (V, E)$ be an instance of the minimum vertex cover (MVC) problem. Create an instance of the minimum set cover (MSC) problem where $U = E$ and for each $u \in V$, the set S_u contains all edges incident to u.

Let $C_S = \{S_{u_1}, S_{u_2}, \ldots, S_{u_k}\}$ be a set cover, where $k = |C_S|$. Then our corresponding vertex cover will be $C_V = u_1, u_2, \ldots, u_k$.

Proof of Correctness: To see that C_V is a vertex cover, take any $(u, v) \in E$. Since $(u, v) \in U$, there is some set S_{u_i} containing (u, v), so u_i equals u or v and (u, v) is covered in the vertex cover. Thus, every vertex cover in G corresponds to a set cover in $U, \mathcal{S}$.

Now take any vertex cover $u_1, \ldots, u_k$. To see that $S_{u_1}, \ldots, S_{u_k}$ is a set cover, take any $(u, v) \in E$. By the definition of vertex cover, there is an i such that either $u = u_i$ or $v = u_i$. So $(u, v) \in S_{u_i}$, so $S_{u_1}, \ldots, S_{u_k}$ is a set cover. Thus, every set cover in $U, \mathcal{S}$ corresponds to a vertex cover in G.

Since every vertex cover has a corresponding set cover (and vice-versa) and minimizing set cover minimizes the corresponding vertex cover, the reduction holds.

3 Reduction to 3-Coloring

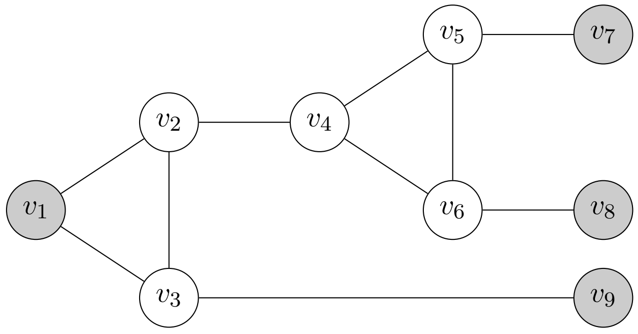
Given a graph $G = (V, E)$, a valid 3-coloring assigns each vertex in the graph a color from {red, green, blue} such that for any edge (u, v), u and v have different colors. In the 3-coloring problem, our goal is to find a valid 3-coloring if one exists. In this problem, we will give a reduction from 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 blue to any of the gray vertices (v_1, v_7, v_8, v_9). Show that if v_1 is assigned the color green, then at least one of $\{v_7, v_8, v_9\}$ is assigned the color green.

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 (v, v_{FALSE}) and (v, v_{BASE}) in the graph, then in any valid 3-coloring v 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 red or green.

(2) v_1 is assigned the color green.

(3) At least one of $\{v_7, v_8, v_9\}$ is assigned the color green.

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 red, then the gray vertex on the left must be assigned the color red as well. Consider the triangle on the right. Since all the gray vertices are assigned red, the two right points must be assigned the colors green and blue, and so the left point in this triangle must be assigned red 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 red 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 red, v_{TRUE} the color green, and v_{BASE} the color blue; assign x_i the color green if x_i is v_{TRUE} and red if x_i is v_{FALSE} (vice-versa for $\neg x_i$). Assign each C_j the color green. 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 green, 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 red, v_{TRUE} is colored green, and v_{BASE} is colored blue. Then for each literal where x_i is color green, that literal is true in the satisfying assignment. By part a, we know that exactly one of $x_i, \neg x_i$ is colored green, so this produces a valid assignment. By observation (i), we also know every node C_j must be colored green. All literal nodes are colored red or green, so by part b, this implies that for every clause, one of the gray literal nodes in the clause gadget is colored green, i.e. the clause will be satisfied in the 3-SAT assignment.

4 k-XOR

In the k-XOR problem, we are given n boolean variables $x_1, x_2, \ldots, x_n$, a list of m clauses each of which is the XOR of exactly k distinct variables (that is, the clause is true if and only if an odd number of the k variables in the clause are true), and an integer r. Our goal is to decide if there is some assignment of variables that satisfies at least r clauses.

(a) In the Max-Cut problem, we are given an undirected unweighted graph $G = (V, E)$ and integer c and want to find a cut $S \subseteq V$ such that at least c edges cross this cut (i.e. have exactly one endpoint in S). Give and argue correctness of a reduction from Max-Cut to 2-XOR.

Hint: every clause in 2-XOR is equivalent to an edge in Max-Cut.

(b) Give and argue correctness of a reduction from 3-XOR to 4-XOR.

Solution:

(a) We create a variable for each vertex i, and a clause x_i XOR x_j for each edge (i, j). We choose $c = r$.

For any assignment, consider the cut such that S contains all vertices i for which x_i is true in this assignment. For each edge (i, j) crossing this cut, its corresponding clause is true because exactly one of x_i, x_j is true. For each edge not crossing this cut, its corresponding clause is false because either both x_i, x_j are true or neither is true. This proves correctness of the reduction.

(b) The reduction is to add a new variable y and add y to every clause in the Max 3-XOR instance and choose the same value of r to get a Max 4-XOR instance.

Given an assignment that satisfies r clauses in the Max 3-XOR instance, the same assignment plus y set to false satisfies r clauses in the Max 4-XOR instance. Given an assignment that satisfies r clauses in the Max 4-XOR instance, if y is false, then the same assignment minus y satisfies r clauses in the Max 3-XOR instance. Otherwise, y is true, and the same assignment minus y and negating all other variables satisfies r clauses in the Max 3-XOR instance.

To see this, consider any clause that was true in Max 4-XOR. Deleting y takes it from having an odd number of true variables to an even number. Then, since 3 is odd, negating all variables brings it back to having an odd number of true variables. Similarly, any clause that was false in the Max 4-XOR instance has an even number of true variables. Deleting y causes it to have an odd number of true variables, and then negating all other variables brings it back to even.

5 Dominating Set (Optional)

A dominating set of a graph $G = (V, E)$ is a subset D of V, such that every vertex not in D is a neighbor of at least one vertex in D. Let the Minimum Dominating Set problem be the task of determining whether there is a dominating set of size $\leq k$. Show that the Minimum Dominating Set problem is NP-Complete. You may assume that G is connected.

Hint: Try reducing from Vertex Cover or Set Cover.

Solution:

Proof that Minimum Dominating Set is in NP.
Given a possible solution, we can check that it is a solution by iterating through the vertices not in the solution subset D and checking if it has a neighbor in D by iterating through the adjacent edges. This would take at most E time because you would at most check every edge twice. We should also check that the number of vertices in D is less than k, which would take at most V as k should be less than E. So, we can verify in $O(E)$ time which is polynomial.

Proof that Minimum Dominating Set is NP-Hard.
Reduction from Vertex Cover to Dominating Set
Minimum Vertex Cover is to find a vertex cover (a subset of the vertices) which is of size $\leq k$ where all edges have an endpoint in the vertex cover.

Suppose $G(V, E)$ is an instance of vertex cover and we are trying to find a vertex cover of size $\leq k$. For every edge (u, v), we will add a new vertex c and two edges (c, u), and (c, v). We will pass in the same k to the dominating set. This is a polynomial reduction because we are adding $|E|$ vertices and $2|E|$ edges.

Proof of Correctness of Reduction
1. **If minimum vertex cover has a solution, then minimum dominating set that corresponds to it has a solution.**
Suppose we have some minimum vertex cover of size k. By definition of vertex cover, every edge has either one or both endpoints in the vertex cover. Thus if we say that the vertex cover is a dominating set, every original vertex must either be in the dominating set or adjacent to it. Now we have to figure out whether the vertices we added for every edge are covered. Since every edge has to have one or both endpoint in the vertex cover, the added vertices must be adjacent to at least one vertex in the vertex cover. Thus the vertex cover maps directly to a dominating set in the transformed problem.

2. **If minimum dominating set in the transformed format has a solution, then the corresponding minimum vertex cover has a solution.**
Suppose we have some minimum dominating set of size k of the transformed format. Vertices in the dominating set either must come from the original vertices or the vertices we added. If they don't come from the vertices we added in the transformation, then from the logic stated in the previous direction, the dominating set corresponds directly to the vertex cover. If some vertices come from the transformation, then we can substitute the vertex for either

of the endpoints of the edge it corresponds to without changing the size of the dominating set. Thus, we can come up with a dominating set of size k that only uses vertices from our original problem, which will directly match to a minimum vertex cover of size k in the original problem.

Alternative Proof that Minimum Dominating Set is NP-Hard.
Reduction from Set Cover to Dominating Set

Minimum Set Cover is to find a set cover (a subset of all the sets) which is of size $\leq k$ where all elements are covered by at least one set of the set cover.

Suppose (S, U) is an instance of set cover where U denotes the set of all distinct elements and S is a set of subsets S_i of U. We will construct a graph $G = (V, E)$ as follows. For each element u in U construct a vertex; we will call these "element vertices". For each possible S_i construct a vertex; we will call these "set vertices". Connect each vertex S_i to all u in S_i.

Notice that if we were to run dominating set on the graph right now, we would be able to cover all the element vertices with any valid set cover, but we would have to pick every single set vertex in order to ensure that all set vertices are covered. To rectify this, connect every set vertex to every other set vertex, forming a clique. This ensures that we can cover all the set vertices by picking just one. This way we really only need to worry about covering the element vertices.

Proof of Correctness of Reduction

1. **If minimum set cover has a solution, then minimum dominating set that corresponds to it has a solution.**

Suppose we have some minimum set cover of size k. This will correspond to a dominating set of size k as well. For each set in our minimum set cover, pick the corresponding set vertex. It follows directly from the construction of the graph and the definition of a set cover that all set and element vertices are covered.

2. **If minimum dominating set in the transformed format has a solution, then the corresponding minimum set cover has a solution.**

Suppose we have some dominating set D of size k. We can find a set cover of size $\leq k$. To do this, we will construct a new dominating set D' that contains only set vertices. Include every set vertex in D in D'. For each vertex in our dominating set that is an element vertex, pick any random neighboring set vertex and add it to D'. Observe that $|D'| \leq |D|$. Thus if there is a dominating set in G of size $\leq k$, there must be a set cover of size $\leq k$

Since this problem is in NP and is NP-Hard, it must be NP-Complete.

6 Orthogonal Vectors (Optional)

In the 3-SAT problem, we have n variables and m clauses, where each clause is the OR of (at most) three of these variables or their negations. The goal of the problem is to find an assignment of variables that satisfies all the clauses, or correctly declare that none exists.

In the orthogonal vectors problem, we have two sets of vectors A, B. All vectors are in $\{0,1\}^m$, and $|A| = |B| = n$. The goal of the problem is to find two vectors $a \in A, b \in B$ whose dot product is 0, or correctly declare that none exists. The brute-force solution to this problem takes $O(n^2 m)$ time: We compute all $|A||B| = n^2$ dot products between two vectors in A, B, and each dot product takes $O(m)$ time.

Show that if there is a $O(n^c m)$-time algorithm for the orthogonal vectors problem for some $c \in [1,2)$, then there is a $O(2^{cn/2} m)$-time algorithm for the 3-SAT problem. For simplicity, you may assume in 3-SAT that the number of variables must be even.

Hint: Try splitting the variables in the 3-SAT problem into two groups.

Solution: We use an $O(2^{n/2} m)$-time reduction from 3-SAT to orthogonal vectors. We split the variables into two groups of size $n/2$, V_1, V_2. For each group, we enumerate all $2^{n/2}$ possible assignments of these variables. For each assignment x of the variables in V_1, let v_x be the vector where the ith entry is 0 if the ith clause is satisfied by one of the variables in this assignment, and 1 otherwise. We ignore the variables in the clause that are in V_2. For example, if clause i only contains variables in V_2, then $v_x(i)$ for all x. Let A be the $2^{n/2}$ vectors produced this way.

We construct B containing $2^{n/2}$ vectors in a similar manner, except using V_2 instead of V_1.

We claim that the 3-SAT instance is satisfiable if and only if there is an orthogonal vector pair in $A \times B$. Given this claim, we can solve 3-SAT by making the orthogonal vectors instance in $O(2^{n/2} m)$ time, and then solving the instance in $O((2^{n/2})^c m) = O(2^{cn/2} m)$ time.

Suppose there is a satisfying assignment to 3-SAT. Let x_1 be the assignment of variables in V_1, and x_2 be the assignment of variables in V_2. Let v_1, v_2 be the vectors in A, B corresponding to x_1, x_2. Since every clause is satisfied, one of $v_1(i)$ and $v_2(i)$ must be 0 for every i, and so $v_1 \cdot v_2 = 0$. So there is also a pair of orthogonal vectors in the orthogonal vectors instance.

Suppose there is a pair of orthogonal vectors v_1, v_2 in the orthogonal vectors instance. Then for every i, either $v_1(i)$ or $v_2(i)$ is 0. In turn, for the corresponding assignment of variables in V_1, V_2, the combination of these assignments must satisfy every clause. In turn, the combination of these assignments is a satisfying assignment for 3-SAT.

Comment: It is widely believed that SAT has no $O(2^{.999n} m)$-time algorithm - this is called the Strong Exponential Time Hypothesis (SETH). So it is also widely believed that orthogonal vectors has no $O(n^{1.99} m)$-time algorithm, since otherwise SETH would be violated. It turns out that we can reduce orthogonal vectors to string problems such as edit distance and longest common subsequence, and so if we belive SETH then we also believe those problems also don't have $O(n^{1.99})$-time algorithms. The field of research studying reductions between problems with polynomial-time algorithms such as these is known as fine-grained complexity, and orthogonal vectors is one of the central problems in this field.