Homework 10

Due: Friday, Nov 21, 11:59 PM PT

1. The Sushi Express wants to implement a new feature, "order batching." The menu consists of a set of desired food items *F* and a set of combos $c_1, \ldots, c_n$ where each combo is a set of food items along with a price - we represent combo $c_i = (\{f_i^1, f_i^2, \ldots, f_i^k\}, p_i)$, where each f_i^j is a food item in the combo. If a student orders c_i, they receive each of the food items, and they pay the cost p_i. In the new order batching system, the students would input a list of food items that they want, and the order batching system should find the **cheapest** set of combos, which would include **at least one of each desired food** item. Write the decision version of this problem, and show via reduction that this decision problem is NP Hard. (20 points)

   Example:
   Foods F = {Fries, Burger, Pizza, drink, sandwich, cookie, chips}
   C_1 = ({Fries, Pizza, cookie}, 10$)
   C_2 = ({Fries, drink, sandwich}, 15$)
   C_3 = ({drink, Pizza, chips, burger}, 17$)

   Input order: {Fries, drink, pizza} output: $\{C_1, C_2\}$ (for 25$) is the cheapest
   Input order: {burger} output: $\{C_3\}$ is the cheapest

   Solution:
   First, we write the decision version of this problem, call it SUSHI:
   Given set of food items *F*, set of combos C, subset of food (order) O, and total price *k*, does there exist a subset of combos of price at most *k* which contain all the items in O.

   Now, we take the SETCOVER problem (known to be NP-Hard) and reduce it to SUSHI (SETCOVER $\leq_p$ SUSHI)

   Construction: Suppose we have a SETCOVER instance with element set U, sets $S_1, \ldots, S_m$,, and integer *k*. Then, we construct a SUSHI instance in which we construct the set of foods F by having a food item f_u for every $u \in U$. We let combos $c_1, \ldots, c_m$ correspond to $S_1, \ldots, S_m$, i.e., C_i includes food item f_u for every $u \in S_i$. We let the price of every item be 1, the food order O to be the same as all of F, and the total price k.
   Note that the construction takes only polynomial time.

   Claim: There exists a valid set cover of at most *k* sets, if and only if there is a set of combos in the constructed SUSHI instance with price at most *k*, that covers order O.

   Proof:
   Forward) Suppose we have a set cover Z of at most k sets. Then, for each S_i in Z, we select the combo c_i in our SUSHI solution. Since Z has at most k sets, at most k combos are selected, and because they all have price 1, these combos will be within the price bound k. Since every element u is covered by some set S_i in the set cover Z, the corresponding food f_u in O is included in c_i by

construction, thus proving we have a valid SUSHI solution.
Backward) Suppose we have our SUSHI solution - for each selected combo c_i in there, we pick S_i to be in the set cover. Since each combo has a price of 1, a valid solution with price at most k, will have at most k combos, and thus, our set cover has at most k sets. Further, since each food f_u is included in some combo c_i, each element u is covered by the corresponding S_i, thus, we have a valid set cover.

Thus, this poly-time reduction from SETCOVER to SUSHI proves that SUSHI is NP-Hard.

Rubric:

- 4 points to write the decision version
- 16 points for showing NP-Hardness.
  - 9 points for construction
  - 1 point for correct claim
  - 3 points for forward proof
  - 3 points for backward proof

2. Problem CLIQUE(G,k) asks whether, given a graph G, does G contain a k-clique? A k-clique is defined to be a set of k vertices such that each pair of these vertices shares an edge. Show via reduction that CLIQUE is NP-complete (20 points)

Solution:
First, we show that CLIQUE(G,k) ∈ NP. CLIQUE(G,k) has a polynomial-size certificate as a set of k vertices. The certifier checks each of the "k choose 2" pairs of given vertices if they are connected by an edge, to determine if these k nodes form a clique. This takes polynomial time as required.
To show NP-Hardness, we reduce from independent set (IS) which is known to be NP-Hard. Suppose we have an instance of IS (G,k). Construct a new graph G′ which is the ***complement*** of G: G′ has the same set of nodes as G, and for every pair of vertices u, v in G, if (u, v) is an edge in G, it is not in G′, and if (u, v) is not an edge in G, it IS an edge in G′. We use the same k for the Clique instance.
Note that the construction takes only polynomial time.

Claim: G has an ind. set of size k if and only if G' has a clique of size k.
Forward) Suppose S is an ind. set of size k in G. Then, for each pair u, v in S, (u,v) is not an edge in G (since S is an ind. set). Hence, (u,v) is an edge in G' by construction. Since this is true for each pair u, v in S, S is a clique (of size k) in G'.
Backward) Suppose S is a clique (of size k) in G'. Then, for each pair u, v in S, (u,v) is an edge in G' (since S is a clique). Hence, (u,v) is not an edge in G by construction. Since this is true for each pair u, v in S, S is an ind. set of size k in G.
construction takes only polynomial time, CLIQUE is NP-hard. And since also CLIQUE(G,k) ∈ NP, therefore CLIQUE(G,k) is NP-complete.

Thus, this poly-time reduction from IS to CLIQUE proves that CLIQUE is NP-Hard.

Rubric:

- 4 points to show poly-time certification.
- 16 points for showing NP-Hardness.
  - 9 points for construction
  - 1 point for correct claim
  - 3 points for forward proof
  - 3 points for backward proof

3. Recall the 3-SAT problem. Given a 3CNF input formula, it tries to find an assignment of variables that satisfies ALL given clauses. Now, we consider a variation - the **partial** satisfiability problem, denoted as 3-Sat(α), for a specified constant α. Here, we are given a collection of k clauses, each of which contains exactly three literals, and we are asked to determine whether there is an assignment of true/false values to the literals such that at least αk clauses will be true. Note that for α=1, we have the problem 3-Sat(1) which is exactly the 3-SAT problem as per definition. We want to analyze that the problem corresponding to a smaller α, in particular, $\alpha=\frac{15}{16}$ which gives us the problem 3-Sat($\frac{15}{16}$). Show that the problem 3-Sat($\frac{15}{16}$) is NP-complete. (20 points)

Hint: If x, y, and z are variables, there are eight possible clauses containing them: $(x \vee y \vee z), (!x \vee y \vee z), (x \vee !y \vee z), (x \vee y \vee !z), (!x \vee !y \vee z), (!x \vee y \vee !z), (x \vee !y \vee !z), (!x \vee !y \vee !z)$

Solution:

To prove it's in NP: given a truth value assignment as certificate, we can count how many clauses are satisfied and compare it to $15k/16$.

To prove it's NP-hard:
We will show that 3-SAT $\leq_p$ 3-SAT(15/16). For each set of 8 original clauses, create 8 new clauses using 3 new variables, by constructing all the possible 8 clauses on the 3 new variables. If the number of clauses is a multiple of 8, then we have created m new clauses for the m existing ones. Now, any assignment will satisfy only 7/8 of the new clauses by construction, so we can say, all of the original clauses in a valid solution can be satisfied if and only if 15/16 of all the new clauses can be satisfied (we will prove this statement both ways). If the number of clauses is not a multiple of 8, say it is of the form 8a + b with b < 8. Then we follow the procedure above for the first a groups of 8 clauses, leading to a total of 16a + b clauses in the new instance.

Claim: 3-SAT instance is satisfiable if and only if the constructed instance is (15/16)-satisfiable.

→) Suppose 3-SAT instance is satisfiable. Thus, for some assignments of the variables, all m clauses can be satisfied. We take the same assignments for the new instance, however, all the newly introduced variables are yet to be assigned. We set any assignments to them. Then, we know that 7/8 of the new clauses are satisfied. Therefore, 8a+b original clauses + 7a (out of 8a)

new clauses are satisfied as per the assignments above. Thus 15a+b out of 16a+b total clauses are satisfied. It can be shown that this fraction $\frac{15a+b}{16a+b}$ is at least 15/16.

←) If the new constructed instance is (15/16) satisfiable. Thus, we have assignments to all the variables so that 15/16 of all clauses are satisfied. Since we have 16a+b clauses by construction, '15/16 of them' comes out to '15a + b (- b/16)' clauses. But, b/16 is less than half (with b<8) and 'the no. of satisfied clauses' must be an integer, so at least 15a + b clauses are satisfied, or at most a are not satisfied. But we know that exactly a out of the 8a new clauses are not satisfied (for any given assignments).Thus, all of the original clauses must be satisfied under these assignments. So, these assignments are a solution to make the original instance satisfiable.

Thus, this poly-time reduction from 3-SAT to 3-SAT(15/16) proves that 3-SAT(15/16) is NP-Hard.

Example to demonstrate reduction: If original given formula has 8 clauses
$(a \vee b \vee c) \wedge (!a \vee b \vee c) \wedge (a \vee !b \vee c) \wedge (a \vee b \vee !c) \wedge (!a \vee !b \vee c) \wedge (d \vee e \vee f) \wedge (g \vee !b \vee !c) \wedge (!a \vee !h \vee !c)$ So we add our 8 new clauses with new variables say x,y,z so that formula now contains total 16 clauses.i.e.:
$(a \vee b \vee c) \wedge (!a \vee b \vee c) \wedge (a \vee !b \vee c) \wedge (a \vee b \vee !c) \wedge (!a \vee !b \vee c) \wedge (d \vee e \vee f) \wedge (g \vee !b \vee !c) \wedge (!a \vee !h \vee !c) \wedge$
$(x \vee y \vee z) \wedge (!x \vee y \vee z) \wedge (x \vee !y \vee z) \wedge (x \vee y \vee !z) \wedge (!x \vee !y \vee z) \wedge (!x \vee y \vee !z) \wedge (x \vee !y \vee !z) \wedge (!x \vee !y \vee !z)$

Rubric:

- 4 points to show poly-time certification.
- 16 points for showing NP-Hardness.
    - 9 points for construction
    - 1 point for correct claim
    - 3 points for forward proof
    - 3 points for backward proof

Ungraded Problems:

1. Given a graph G = (V, E) and two integers k, m, the Dense Subgraph Problem is to find a subgraph G′ = (V′, E′) of G, such that V′ has at most k vertices and E′ has at least m edges. Prove that the Dense Subgraph Problem is NP-Complete. HINT: Use Independent Set for your reduction. (20 points)

   Solution:

   For efficient certification: Take a subgraph G′ = (V′, E′) of G as certificate. The certifier checks that G′ has at most k vertices and E′ has at least m edges. This takes polynomial time.

   Next, we prove that the Independent set problem $\leq_p$ Dense Subgraph Problem. Given a graph $G(V, E)$ and an integer k, an independent set decision problem outputs yes, if the graph contains an independent set of size k. For an arbitrary graph $G = (V, E)$ of n vertices, we first get the complementary graph G_c of G.

   A clique is a subset of vertices of an undirected graph G such that every two distinct vertices in the clique are adjacent; that is, its induced subgraph is complete. We know that a clique will always contain $k(k-1)/2$ edges if there are k vertices in the clique, and that an independent set in G is a clique in G_c (the complement graph of G) and vice versa.

   Then we set m to $k(k-1)/2$ and test with the dense subgraph problem.

   Claim: There exists an independent set of size k in G (equivalently, a clique in G_c of size k), iff there exists a subgraph of G_c with at most k vertices and at least $m = k(k-1)/$ edges. $\rightarrow$) If there exists a clique in G_c of size at least k, then there exists a subgraph of G_c with at most k vertices and at least $k(k-1)/2$ edges.

   If there is a clique of size at least k then there is a clique of size exactly k. Moreover, by definition, a clique of size k would have $k(k-1)/2$ edges. $\leftarrow$) If there exists a subgraph of G_c with at most k vertices and at least $k(k-1)/2$ edges, then there is a clique of size at least k.

   For a subgraph to have $k(k-1)/2$ edges, implies there are k vertices. So this subset with k vertices forms a clique in G_c of size k.

   Example: Consider a graph $G = (V, E)$ where $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2),(2, 3)\}$. To determine if there's an independent set in G of size 3, we first generate G_c, where $E_c = \{(1, 3),(1, 4),(2, 4),(3, 4\}$. Since there's a dense graph with 3 vertices and $3 \cdot 2/2 = 3$ edges in G_c, which is $V' = \{1, 3, 4\}$, $E' = \{(1, 3),(1, 4),(3, 4)\}$, we can say there's an independent set in G of size 3. On the other hand,if we set $k = 4$ and $m = 4 \cdot 3/2 = 6$, then there's no such dense graph in G_c, thus no independent set of size 4 in G.

2. DOUBLECLIQUE(G, k) asks whether, given a graph G, does G contain 2 vertex-disjoint cliques of size at least k? Show via reduction that DOUBLECLIQUE is NP-Hard. HINT: Use CLIQUE for your reduction. (12 points)

   Solution:
   We will use the CLIQUE problem, which we demonstrated in the last part is NP-hard. Say that we have a CLIQUE instance, G, k. Construct a new graph G′ which is just G with k-additional vertices, where each pair of those new vertices shares an edge. Note that if G′ contains 2 cliques, then at least one of those cliques must have existed in G. Therefore, G must contain a clique as well. Then, note that if G contains a clique, G′ will contain that clique in addition to the new clique we have added. Clearly, adding these new vertices takes only polynomial time, so we see that DOUBLECLIQUE is NP-hard.

3. DOMINATINGSET(G,k) asks whether given a graph G, is there a set S of at most k vertices such that each vertex in G is either in S or adjacent to at least one vertex in S. Show that DOMINATINGSET is NP-Hard. HINT: Use VERTEXCOVER. (12 points)

   Solution:
   Take some connected vertex cover instance G, k. Create a new graph G' with the same vertices and edges as G. For each edge (u, v) in G, add a new vertex w with edges (u', w) and (v', w). Note that if G has a vertex cover S of size at most k, then this vertex cover still contains a vertex adjacent to all of the old vertices in G' because the old edges were imported directly. In addition, note that because S includes a vertex incident on each edge in G, it also contains a vertex adjacent to each of the newly added vertices in G'. Therefore, it is by definition a dominating set. Going the other way, if G' contains a dominating set of size at most k, we will show that G contains a vertex cover of size at most k. If this dominating set contains only vertices from the original G, then we are done. If the dominating set in G' contains one of the newly added vertices, then note that by instead including one of the vertices adjacent to the new vertex, the resulting set is still a dominating set. Therefore, if G' has a dominating set of size k, it also has a dominating set of size at most k where all of the vertices in the dominating set appeared in G. Therefore, if G' contains a dominating set, then G must have an independent set. Because the construction of G' took only polynomial time, this reduction is sufficient to show that the dominating set is NP-hard.

4. DELIVERY(S,k,d,m) asks whether, given a set S of major cities, and an arbitrary distance function $d : S \times S \rightarrow R+$, can we find a subset of these major cities of size at most k where we can place distribution points in order to deliver to every city in S in such a way that the distribution center is at most distance m from the destination? Show via reduction that DELIVERY is NP-Hard. NOTE: The distance function does not need to satisfy properties one would ordinarily expect distance functions to satisfy, such as the triangle inequality. HINT: Use DOMINATINGSET. (12 points)

Solution:
Take some dominating set instance G, k. Create a DELIVERY instance with the following parameters. Let the cities be vertices, k be the same as in the DOMINATINGSET instance, d be 1 for any pair of vertices which share an edge, and 3 otherwise, and m be 2. Note that 2 cities are close enough for delivery iff they correspond to adjacent edges in the original graph. Therefore, a dominating set in G corresponds precisely to a set of cities for DELIVERY. In addition, a set of cities which can deliver to each other corresponds with a dominating set in the original graph. Therefore, because this conversion was polynomial time, the DELIVERY problem is NP-hard.