James Sweetman (jts2939)
Jung Yoon (jey283)
Unique #51835

HW #9

Q1.

a. Proof of NP-complete:
   Let's note that in order to show that 4-SAT is NP-complete, we have to prove that it is in NP and NP-hard, by the definition of NP-complete.

   First, to show that 4-SAT is in NP:

   We can devise an algorithm such that it takes in a 4-SAT instance and a proposed truth assignment as input. Then, the algorithm evaluates the two inputs with each other. If it is true, outputs yes, otherwise outputs no. Runtime is in polynomial time since it's a nondeterministic polynomial-time algorithm.

   Second, to show that 4-SAT is NP-hard:

   To do this we must reduce 3-SAT to 4-SAT such that Φ is an instance of 3-SAT and n Φ' is an instance of 4-SAT. Then, we transform each clause in 3-SAT Φ $(X \lor Y \lor Z)$ such that $(X \lor Y \lor Z \lor W) \land (X \lor Y \lor Z \lor {\sim}W)$ such that W is the new variable added for 4-SAT Φ'. Runtime is in polynomial time.

   We can now see the following two things:

   1) If a clause $(X \lor Y \lor Z)$ satisfies the truth assignment, then $(X \lor Y \lor Z \lor W) \land (X \lor Y \lor Z \lor {\sim}W)$ must also satisfy the truth assignment since it doesn't matter what W is; it'll be true as well regardless. That is, W can be any arbitrary set and it would still be satisfiable. Thus, if Φ can be met (regarded as true), then so can Φ'. That is, both are satisfiable.
   2) Let's say that Φ' is regarded as true given a truth assignment T. Then, $(X \lor Y \lor Z \lor W) \land (X \lor Y \lor Z \lor {\sim}W)$ must also be true within truth assignment T. Since W and ~W are opposite truth values, $X \lor Y \lor Z$ will be true under T as well. Thus, Φ is satisfiable.

   Thus, 4-SAT is NP-complete.

b. Proof of NP-complete:

   We can prove that T-SAT is NP-complete by a reduction of the 3-SAT.

We can construct a graph such that if we can find M vertices that form a subset such that it is a dominating set, we can set those M variables as true in Φ, where Φ is an instance of a 3-SAT. Each clause must contain at least one of these vertices in M.

Thus, we have proved that T-SAT is satisfiable if and only if 3-SAT ϕ is as well.

c. Proof of NP-complete:
Let's first note that from what we are given, the number of clauses and variables are equal. In this case specifically, there are 3 clauses and 3 variables.

There are two ways that we see that we could prove NP-completeness.

The first approach would be to construct a bipartite graph with all the nodes and to find a perfect matching. Should one exist, it would indicate that it is satisfiable. This would be done in polynomial time.

The second approach (the arguably better, more complete approach) needs a longer explanation:

We can prove that P-SAT is NP-complete by a reduction of the 3-SAT. Like in the previous solutions, Φ is an instance of 3-SAT such that there are A variables and B clauses. There are several cases:

1) A and B are equal. In this case we do nothing.
2) If A > B then we add a new variable(s) to Φ such that Wi is not in Φ already. The number of new variables should be A-B+1 so we add that many new variables until we have A = B.
3) If A < B, then we add a 3-literal clause (X v Y v Z) where X,Y,Z are new variables. We would increase A by 3 and M by 1 with each addition and we'd keep adding new clauses in Φ until A=B OR the difference between A and B is either one or three. In the case that the difference is one, adding one more 3-literal clause causes A>B so we would revert back to (2) to make A=B. In the case where the difference is 2, adding a final new clause would be A=B. And in the case where the difference is 3, we would add a 2-clause (X v Y), thereby increasing A by 2 and B by 1 making A=B.

Thus, we have proved that P-SAT is satisfiable if and only if 3-SAT ϕ is as well.

Q2 .

First, let's note that this problem is NP since by definition of NP, we can get a set X and check the size of the intersection with every set Ai in A1,...,Am.

Let's suppose that there is a 3D matching such that there are sets X, Y, Z (all of size n) and that that is a set of m triples from X x Y x Z. We will call this set of triples T, which happens to also be an intersection inference by its definition, from now on for our convenience.

The purpose is to show that 3D matching <= intersection inference. That is, 3D matching reduces down to intersection inference.

So, now, let's say that T = S, where for every element such that j is an element of the union of X, Y, Z (that is, j is an element of X ∪ Y ∪ Z),, a set Aj is created of the triples that contain j.

From this, we can check to see if there exists a set R such that R is a subset of S, such that R has an intersection of size 1 which every set Aj. If it exists, then such sets R would consist of exactly those collection of triples such that every element of X x Y x Z appears in exactly one. This means that R represents the perfect 3D matchings.

Then, this means that our example set T has a positive answer if and only if the original instance of 3D matching does as well.

Thus, we know that intersection inference is NP-complete.

Q3.

First, let's note that zero-weight-cycle is in NP because we can get a cycle in G and we can consistently check that the cycle's sum of the edge weights is 0.

The purpose is to show that subset sum <= zero-weight-cycle. That is, subset sum reduces down to zero-weight-cycle. This means that given w1,...wn, we want to see if there is a subset whose sum is W.

Construction of graph: To do this, we make a zero-weight-cycle such that the graph has nodes 0,1,...n and such that there is an edge (n,0) of weight -W. There should also be an edge (a,b) for all pairs where a<b such that (a,b) is equal to wj.

From this, let's say that there is a subset that adds up to exactly W if and only if G has a zero-weight cycle. If such a subset S exists, then we know that a cycle starts at 0, traverses through the nodes that are in S, and then returns to 0 on the edge (n,0) whose weight is -W. Since the weight of edge (n,0) is -W, it cancels out the sum of the other edge weights which is W.

All cycles of G must use edge (n,0) so assuming that there is a zero-weight cycle, we know that the other edges summed up must cancel -W. That is to say, the values of the subset must add up to exactly W. From this, we know that the subset sum must be less than or equal to zero-weight cycle.

Thus, this problem is NP-complete.