Directions for Homework 7: You may assume from class that the following problems (both the search and decision versions) are NP-complete when doing your reductions. We will (eventually) cover the reasons that all these problems are NP-complete in class.

- SAT : Given a boolean formula F in CNF form with n variables and m clauses, determine if there is an assignment of the n variables that satisfies F.

- $3 - \text{SAT}$: A special case of SAT where every clause has at most 3 literals.

- Integer Linear Programming : Given a set of n variables and m linear constraints, and a linear objective function, find an *integer* solution to the constraints that optimizes the objective.

- Independent Set : Given graph G and $k \in \mathbb{Z}^+$, determine if G has an indep. set of size $\geq k$.

- Vertex Cover : Given graph G and $k \in \mathbb{Z}^+$, determine if G has a vertex cover of size $\leq k$.

- Clique : Given graph G and $k \in \mathbb{Z}^+$, determine if G has a clique of size $\geq k$.

- Subset Sum : Given a list of integers A and a target T, determine if some sublist of A adds up to T.

- Knapsack : Given a list of n items with weights w_i and values v_i a maximum capacity C and a target value V, determine if some subset of of the items have a total weight $\leq C$ and a total value $\geq V$.

- Hamiltonian Path (Rudrata): Given graph G, determine if G has a path that visits every vertex exactly once.

- Hamiltonian Cycle (Rudrata): Given graph G, determine if G has a cycle that visits every vertex exactly once.

1. (20 points) We define the problem Clique3(G, k). You are tasked with determining if G has a clique of size k, but only on speical inputs where the max degree of any vertex of G is 3. In otherwords, this is a special case of the Clique(G, k) problem, but we are restricted to graphs in which every vertex has degree at most 3.

    (a) Prove that Clique3 is in NP.

    > **Solution:**

    (b) I claim that Clique3 is NP-complete.

    > PROOF: We showed that Clique3 is in NP in part a.
    >
    > We show a reduction from Clique3 to Clique , a known NP-hard problem. Clique is a generalizaion of Clique3, since Clique3 consists of the special case of Clique where the input graph has max degree 3. Therefore we can solve Clique3(G, k) by returning Clique(G, k) on exactly the input to Clique3 unchanged.
    >
    > This proves the correctness of the reduction and, therefore the NP-completeness of Clique-3. ∎

    What is wrong with the above proof?

> **Solution:**

(c) Here's a true fact that you can use without proof.

Theorem 1.1: Independent Set3(G, k), the special case of Independent Set(G, k) restricted to graphs of degree at most 3, is NP-complete.

Here's another fact that we learned in lecture when showing that Clique was NP-complete.

Theorem 1.2: A subset $X \subseteq V$ is an independent set in G if and only if X is a clique in the "opposite" graph $G' = (V, E' = \binom{v}{2} \setminus E)$. Therefore G has a independent set of size $\geq k$ if and only G' has a clique of size $\geq k$.

Now lets try again.

Claim: Clique3 is NP-complete.

PROOF: Clique3 is in NP as proved in part a.

We present a reduction from Independent Set3 to Clique3.

To solve Independent Set3(G, k), we first create G' by switching all the edges of G, and returning Clique3(G', k).

Proof of correctness: By the second fact, G has an independent set of size k if and only if G' has a clique of size k. This proves the correctness of the reduction.

By the first fact, Independent Set3 is NP-complete, and therefore Clique3 is NP-complete.■

What's wrong with the proof this time?

> **Solution:**

(d) Describe any poly-time algorithm for Clique3. Hint: How big can a clique get?

> **Solution:**

2. (40 points) Generalizations: Show that each of the following problems are NP-hard. (Don't worry about showing that these problems are in NP). Each reduction should be "simple" - each problem below is either a direct generalization of a known NP-hard problem, or solves a known NP-hard problem with a relatively simple transformation (like from Independent Set to Clique).

(a) (EXAMPLE) Weighted Independent Set: Given a graph $G = (V, E)$ with integer weights on the vertices V, and number k, determine if G has an independent set of total weight $\geq k$.

Prove that this problem is NP-complete.

> **Solution:**
>
> - First we show this is in NP. Say we are given the inputs G, K as well as an independent set with total weight $\geq k$. Our verifier will check that
>
>   - The given vertices have total weight $\geq k$ in at most $O(n)$ time.

– That this is a valid independent set, by checking each edge and making sure that both endpoints are never in the set. This will take $O(E)$ time.

– Accept if both of the above are True

This will accept iff the certificate is a valid independent set of total weight $\geq k$, which exists iff there is an independent set of total weight $\geq k$.

• We show that this problem is a generalization of Independent Set .

Our reduction: Given an instance (G, k) of Independent Set, create a weighted version of G where every vertex has weight 1. Call this graph G'. Now, return Weighted Independent Set(G', k).

PROOF: By construction, any set of k vertices in G has total weight k in G' and vice versa. Thus an independent set of k vertices exists in G iff that independent set has total weight k in G'. ∎

(b) (EXAMPLE) $\mathsf{SAT}_{\leq k}$: Given a SAT instance with n variables and m clauses, and an integer $k \leq n$, decide if there is a satisfying assignment in which *at most k* of the n variables are assigned TRUE.

Prove that this problem is NP-complete.

Solution:

• First we show this is in NP. Say we are given the input formula f, the integer k, and a satisfying assignment that sets at most k variables ot true. Our verifier will check that

– At most k of the variables are set to true. We can check this in $O(n)$ time.

– That this is a satisfying assignment by evaluating the formula. Each clause takes $O(n)$ time to check as there are at most n variables per clause. There are m total clauses, so this takes a total of in $O(mn)$ time.

– Accept if both of the above are True

This will accept iff there is a valid assignment

• We show that this problem is a generalization of SAT .

Our reduction: On an input F to SAT , we return $\mathsf{SAT}_{\leq n}(F)$.

PROOF: Since the restriction "at most n of the variables are set to True" applies to any assignment, the problem $\mathsf{SAT}_{\leq n}$ is exactly the problem of $\mathsf{SAT}(F)$, and SAT is a special case of $\mathsf{SAT}_{\leq n}$. ∎

(c) Given a set of n elements E, and a family of m subsets $S_i \subseteq E$, and "budget" b, determine if there are some b of these subsets $\{S_i\}$ such that their union is E.

Solution:

(d) Longest $\mathsf{s-t}$ Path: Given a graph G, two vertices s and t in V, and an integer k determine

if G has a simple path (no repeat vertices) that uses k or more edges from s to t.

> **Solution:**

(e) Traveling Salesman Problem : Given a weighted graph G and bound B, determine if there is a cycle in G that visits every vertex exactly once and has the total sum of edge weights $\leq B$.

> **Solution:**

(f) Degree Restricted MST: Given an undirected graph $G = (V, E)$ and an integer k, find a spanning tree T of G such that each vertex of the tree has maximum degree k, if such a tree exists.

> **Solution:**

3. (EXAMPLE) Hamiltonian Path : Show that the following problem is NP-complete by reducing from Hamiltonian Cycle .

Hamiltonian Path : Given graph G, determine if G has a simple **path** (no repeated vertices) that visits every vertex exactly once. Here the start and end vertices of the path do not need to be the same (or adjacent).

> **Solution:** Note: This solution explains more things than you need to in your solutions. This is meant to be instructive.
>
> - To show this problem is in NP, say we are given an input which is the graph G, as well as the vertices of the path (listed in order). Our verifier would check that every vertex was contained in this path, and that every proposed edge between adjacent vertices was in the graph. If both of these conditions are true, then this is a Hamiltonian Path.
>
> - Here's our reduction.
>
>   Remember - we are trying to find an algorithm that can efficiently solve Hamiltonian Cycle , a known NP=hard problem, under the assumption that we can run an algorithm that can solve Hamiltonian Path , the problem that we are trying to show is NP-hard.
>
>   Here's the algorithm: Given graph G, we're going to construct G' as follows. Pick any vertex v, and make an extra copy of it called v', and add all the same edge connections as v. Now construct another new vertex s and connect it only to v, and another new vertex t, and connect it only to v'. Now, we return Hamiltonian Path(G').
>
>   Here's the proof of correctness of this reduction. We need to show that this is CORRECT - that this reduction is an algorithm that actually solves Hamiltonian Cycle . If the input G does have a Hamiltonian Cycle , we need to return YES. If the input G does not have a Hamiltonian Cycle , we need to return NO.
>
>   In this case, that means that showing that G has a Hamiltonian Cycle if and only if G' has an Hamiltonian Path . Here's the proof:
>
>   – G has a Hamiltonian Cycle $\rightarrow$ G' has a Hamiltonian Path

PROOF: Say G has a Hamiltonian Cycle . Then since a cycle visits every vertex, including v, write the vertices of the cycle as $(v, x_1), (x_1, x_2), \ldots, (x_{n-1}, v)$. By construction, the edges $(v, x_1), (x_1, x_2), \ldots, (x_{n-1}, v')$ are in the graph G', as are (s, v) and (v', t). Since the only edges we added were v', s, t, this new set of edges are a Hamiltonian Path in G'. ∎

– G' has a Hamiltonian Path $\rightarrow$ G has a Hamiltonian Cycle

PROOF: Say G' has a Hamiltonian Path . Because s and t both have degree 1, any Hamiltonian Path in G' must have s and t as its endpoints. Since s only connects to v and t to v', then the path must go from s to v to every other vertex to v' to t. The middle part of this path is a path from v to every other vertex back to v' in G'. The equivalent edges in G would form a path from v to every other vertex and back to v again, which are a Hamiltonian Cycle . ∎

Another valid approach is to add t to a different neighbor of v, one at a time, until you find a Hamiltonian Path . This would require at most $O(n)$ calls to Hamiltonian Path , once for each neighbor of v, which is a valid polytime reduction.

4. (20 points) Search SAT $\rightarrow$ Decision SAT . Describe an algorithm that can **find** a satisfying assignment for a given SAT instance (search), if one exists, using an algorithm that can only determine **whether or not** a SAT instance *has* a satisfying assignment (decision).

Remember, you may do polynomial work, and call the Decision version of the algorithm polynomially many times to solve the Search version.

Hint: If there is a valid solution to the SAT instance - use your algorithm for Decision SAT to determine a valid assignment for each variable, one at a time.

Solution:

5. (20 points) Partition: Show that the following problem is NP-complete by reducing from Subset Sum .

Given a list of integers A, determine if the integers of A can be partitioned into two disjoint lists A_1, A_2 that have the same sum.

For example, the list $A = [7, 3, 2, 9, 6, 13]$ can be partitioned into $[7, 13]$, $[2, 3, 6, 9]$ each with sum 20. The list $A = [4, 6, 8, 12]$ cannot be partitioned into two equal pieces.

Solution:

6. (Bonus) Consider the following problem. You are a hiring manager at a consulting firm. You have a set of n available workers W and a set of m tasks T that your clients want you to do. Each worker w knows how to complete some subset $T[w] \subseteq T$ of the tasks, and requires a certain salary to be hired, $S[w]$. Each worker, once hired, can complete all the tasks that they know how to do. Each completed task earns the company a revenue $R[t]$ per task t.

You goal is to maximize your profit, the sum of all revenues earned minus the sum of all salaries paid. Not all tasks have to be assigned; sometimes the revenue for completing some tasks may not be worth hiring a worker to do it.

The decision version of this question asks: Given all this information, $n, m, T[w], S[w], R[t]$, can you achieve a profit of at least k? Show that this problem is NP-complete.

Solution: