Sara Pesavento
CSCI 570 Homework 10

1. Consider the partial satisfiability problem, denoted as 3-SAT(α). 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 3-SAT(1) is exactly the 3-SAT problem from the lecture.
   Prove that 3-SAT(15/16) is **NP-complete**.
   Hint: If x, y, and z are literals, there are eight possible clauses containing them:
   (x∨y∨z),(!x∨y∨z),(x∨!y∨z),(x∨y∨!z),(!x∨!y∨z),(!x∨y∨!z),(x∨!y∨!z),(!x∨!y∨!z)
   If we can prove that this problem is both NP and NP-hard, we can show that it is NP-complete (their intersection).

   1) To prove that this problem is within NP, we can do the following. Given we are provided with a truth value assignment, we can check if at least α*k or (15/16)*k clauses are satisfied in polynomial time.

   2) To show it's NP-Hard, we can show that an 3-SAT $\leq_p$ 3-SAT(15/16). For <u>each set of 8</u> clauses in the original k collection of clauses given to us, we can create three new variables, (e.g. x, y, and z) and construct the 8 possible combinations of these 3 numbers clauses (shown in the problem statement).
   For each set of eight possible combinations clauses we find that 7/8 of the clauses evaluate to 1 no matter what the assignment is to x, y, and z.
   If our original k is some multiple of 8 (e.g. 16, 24), then any assignment we use will satisfy 7/8 of the new clauses. To get to 15/16 satisfied, we must satisfy all of the original clauses!
   If our original k is NOT some multiple of 8, (e.g. 10) then we will have more than 15/16 of clauses satisfied if all our original clauses are satisfied.
   If one or more of the original clauses are not satisfied, then less than 15/16 of the clauses are satisfied.
   E.g. (For k=10 clauses we need at least 9.3 = True. 10/10 original + 7/8 new = (10 + 7) True / 18 Total > 15/16. vs. 9/10 original + 7/8 new = (9 + 7) True / 18 < 15/16.

   So we are asking the providing the black box of 3-SAT(15/16) with k clauses and asking if over 15/16 of clauses are True. If this box returns "Yes", then our answer is that 3-SAT is satisfied in k clauses (all clauses in k are True).

   We have thus proved 3-SAT(α) is NP-Complete.

2. Consider modified SAT problem, SAT' in which given a CNF formula having m clauses in n variables x1,x2,...,xn, the output is YES if there is an assignment to the variables such that **exactly** m−2 clauses are satisfied, and NO otherwise. Prove that SAT' is also NP-Complete.
   If we can prove that this problem is both NP and NP-hard, we can show that it is NP-complete (their intersection).

   1) To prove that SAT' is within NP, we can do the following.
   Certificate: Given an instance of SAT', with assignment of truth values to variables (x1…..xn)
   Certifier: Evaluate the clauses and check if the number of satisfied clauses is m - 2.

   2) To show it's NP-Hard, we can take an NP-complete problem (SAT) and show that SAT $\leq_p$ SAT'.
   So basically, we want to send our m clauses in SAT to the blackbox SAT', and if it returns "YES", then we can conclude that all m of our clauses evaluate to True.
   Construction: Add onto our m clauses four more clauses: x1, x2, $\overline{x1}$, $\overline{x2}$.
   - Proof 1. If we have an assignment which satisfies SAT', then there is an assignment which satisfies SAT. If SAT' x1…...xn + x1 + x2 + $\overline{x1}$ + $\overline{x2}$ returns YES, then the only unsatisfied clauses must be one of each: x1 or $\overline{x1}$ and x2 or $\overline{x2}$. Since there are pairs of x1 and x2. In order for m - 2 clauses to evaluate to True, it must be that x1…..xn are True.
   - Proof 2. If we have an assignment which satisfies SAT, then there is an assignment which satisfies SAT'.

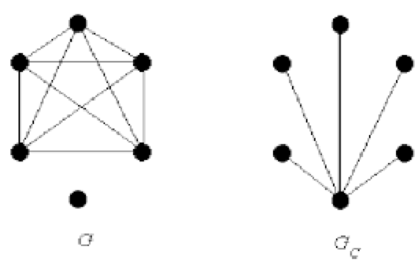
If an assignment x1…..xn evaluates to True, then exactly two of the four extra clauses we added will evaluate to True, giving m' - 2 satisfied clauses for SAT'.

3. Given a graph G = (V,E) and two integers k, m, the *Dense Subgraph Problem* is to find a subset V' of V, whose size is **at most k** and are connected by **at least m** edges.
   Prove that the Dense Subgraph Problem is **NP-Complete**.
   If we can prove that this Dense Subgraph Problem is both NP and NP-hard, we can show that it is NP-complete (their intersection).

   1) To prove that Dense Subgraph is within NP, we can do the following.
   Certificate: Given an instance of Dense Subgraph.
   Certifier: Check to ensure that the size of this subgraph is at most k, and has at least m edges.

   2) To show it's NP-Hard, we can take an NP-complete problem (Independent Set) and show that Independent Set $\leq_p$ Dense Subgraph.
   In an instance of the independent set problem, we are given a graph G = (V, E) and an integer k. The problem evaluates to "Yes" if the graph contains an independent set of size k.

   A clique creates a COMPLETE graph - it's a subset of vertices of an undirected graph such that every two distinct vertices in the clique are adjacent (connected by one edge). As we can see, this is the opposite of the independent set problem, where no two nodes in S are joined by an edge.



   An independent set in G, becomes a clique in the complement graph of G.

   The number of edges in a complete graph is n*(n - 1)/2.

   So we can keep k as is and set m to be k*(k - 1)/2. These two along with the complement of G can be sent to the black box of Dense Subgraph. Then only if there exists a subgraph in the complement of G with at most k vertices and at least k*(k-1)/2 edges, there exists an independent set of size k in G.

   - Proof 1. Subgraph → Independent Set. If there is a subgraph in the complement of G that has at most k vertices and at least k*(k - 1)/2 edges, then there is a clique of size at least k.
     In order for a subgraph to have k*(k - 1)/2 edges, there must be k vertices. So this subgraph with k vertices form a clique of size k in the complement of G, thus an independent set of size k in G.

   - Proof 2. Independent Set → Subgraph. If there is a **clique** in the complement of G that has at least k vertices, then there exists a **subgraph** in the complement of G with at most k vertices and at least k*(k - 1)/2 edges.
     We said we have a clique of at least k vertices, which means there also exists a clique of exactly k vertices, and this clique would have k*(k - 1)/2 edges.