More NP-complete problems

A Boolean expression is in k-CNF if (1) it is a conjunction of disjunctions and (2) every clause has exactly k literals.

Example:

The Boolean expression $(x_2 \vee \overline{x_1} \vee x_3) \wedge (\overline{x_3} \vee x_4 \vee x_2) \wedge (x_4 \vee x_2 \vee x_1)$ is in 3-CNF

The Boolean expression $(x_2 \vee \overline{x_1}) \wedge (x_3 \vee x_2 \vee x_1) \wedge (x_4 \vee x_2 \vee x_1)$ is not in 3-CNF.

The 3-Satisfiability Problem (3SAT)
Instance: A Boolean expression ϕ in 3-CNF
Question: Is ϕ satisfiable (that is, does there exist an assignment of truth values to the variables that makes ϕ true)?

Theorem 3SAT is NP-complete
(proof omitted)

A set of vertices U is **independent** in a graph G if there is no edge incident with any two vertices of U. The size of a maximal independent set in G is called the **independence number** of G and is written $\alpha(G)$.

Example:

$\alpha(G) =$

Determine $\alpha(K_{m,n})$

The **Independent Set (Optimization) Problem** is to find the size of a maximal independent set (i.e., find the independence number).

Example: State the **Independent Set Problem** as a decision problem.

Independence Set Problem
Instance: A graph $G = (V, E)$, integer $k \leq |V|$
Question: Is $\alpha(G) \geq k$ (that is, does G contain an independent set of size at least k?)

Theorem The Independent Set Problem is NP-complete.
Proof
First, show that Independent Set is in NP.
(Discuss…have students write some details)

Next, we will show that 3SAT reduces to Independent Set.

We must write a polynomial algorithm that takes any Boolean expression ϕ in 3-CNF and constructs a graph G and defines an integer k such that ϕ is satisfiable if and only if G has an independent set of cardinality at least k.

Let ϕ be a Boolean expression in 3-CNF. Then ϕ has some number, say n, of clauses, each consisting of a disjunction of 3 variables. (The input size of ϕ is $O(n)$.)

Construct a graph $G = (V, E)$ as follows:

- for each clause C_i $(1 \le i \le n)$ of ϕ, construct a 3-cycle with vertices labeled with the variables of the clause C_i. This constructs $3n$ vertices and $3n$ edges. Thus these vertices and edges can be constructed in a loop taking time $O(n)$.

```
for i = 1 to n
  construct 3 vertices labeled with elements of C_i
  construct 3 edges between these 3 vertices
```

- add edges to the graph between any two literals that are contradictory, that is, add edges between pairs of vertices labeled x_j and $\overline{x_j}$. Adding these edges can be performed in a doubly-nested loop and thus will take time $O(n^2)$.

```
for i = 1 to n
   for j = 1 to n
      if any vertex corresponding to clause C_i is contradictory to any
vertex corresponding to clause C_j
       add an edge
```

- define $k = n$

This construction takes time $O(n^2)$, which is polynomial.

We claim that ϕ is satisfiable if and only if G has an independent set of at least $k = n$ vertices.

Assume that G contains an independent set of size at least $k = n$. Let U be a set of exactly n vertices. Then U must contain exactly one vertex from each 3-cycle since there are edges between any two vertices of the same 3-cycle. Since U is an independent set, no two of the variables corresponding to the vertices in U are contradictory. Assign the value "true" to each of the variables in U. Since U contains a vertex from each 3-cycle, it follows that one variable from each clause C_i has been

assigned true. And since each clause C_i is a disjunction of variables, it follows that the truth value of C_i is true. Finally since ϕ is a disjunction of all clauses, it follows that the value of ϕ is true, that is, ϕ is satisfiable.

For the converse, assume that ϕ is satisfiable. Fix a truth assignment for the variables of ϕ so that the value of ϕ is true. With this truth assignment, at least one variable in each clause must be assigned true. Construct the set U by selecting one such variable from each clause. Then U contains $n = k$ vertices. Since each variable chosen has been assigned true, it follows that there cannot be contradictory variables chosen (we cannot assign both x_j and $\overline{x_j}$ to be true). Thus there are no edges among the vertices in U, that is, U is an independent set of size $n = k$.

Applications of Independent Set:

- dispersion problems (identify locations for a new franchise service so that no two locations are close enough to compete with each other)
- coding theory (vertices represent words, edges between two words sufficiently similar to be confused due to noise in transmission; independent set gives highest capacity code for the given communication channel)

A set of vertices U is a **clique** in a graph $G = (V, E)$ if all edges between any two vertices of U belong to E. The size of a maximal clique in G is called the **clique number** of G and is written $\omega(G)$.

Example:

$\omega(G) =$

Determine $\omega(K_{m,n})$

The **Clique (Optimization) Problem** is to find the size of a maximal clique (i.e., find the clique number).

Example: State the **Clique Problem** as a decision problem.

Clique Problem
Instance: A graph $G = (V, E)$, integer $k \leq |V|$

Question: Is $\omega(G) \geq k$ (that is, does G contain a clique of size at least k?)

Theorem The Clique Problem is NP-complete.

Proof
Show Clique is in NP.
Show Independent Set reduces to Clique.

Graph 3-colorability
Instance: A graph $G = (V, E)$, integer $k \leq |V|$
Question: Is $\chi(G) \leq k$ (that is, can G be colored with at most k colors?)

Theorem The 3-Colorability Problem is NP-complete.

Proof
We have already shown that k-colorability is in NP. Thus 3-Colorability is in NP.
Show 3SAT reduces to Clique.

Practice for Final Exam

Vertex Cover (Optimization)
Given a graph $G = (V, E)$, find the smallest subset S of vertices of G such that every edge of G is incident with at least one vertex belonging to S.

State Vertex Cover as a decision problem
Show Vertex Cover is in NP
Show Vertex Cover is NP-complete
(reduce Independent Set to Vertex Cover)

Subset-Sum Problem
Given a set of n numbers $S = \{s_1, s_2, \ldots, s_n\}$ and a number k, is there a subset of S whose sum is k?

Show Subset Sum is in NP
Subset Sum is NP-complete
(can be shown that 3SAT reduces to Subset Sum or Vertex Cover reduces to Subset Sum)

Knapsack Problem

Given a set of n items with weights $w_1, w_2, \ldots, w_n$ and profits $p_1, p_2, \ldots, p_n$ and a knapsack of capacity C, find a subset of items whose total weight is at most C that maximizes the total profit.

State Knapsack as a decision problem
Show Knapsack is in NP
Show Knapsack is NP-complete
(reduce Subset Sum to Knapsack)