CSE 421 Winter 2026: Section 8

February 26, 2026

Instructions: This section worksheet is designed to assist you in working through some example problems and developing your basics. You are encouraged to collaborate on the problems in this section worksheet as well as use the course's generative AI.

1 A Tour of Canonical NP-Complete Problems

In 1971, Stephen Cook proved a remarkable theorem: the Boolean satisfiability problem (SAT) is complete for the class NP. Independently, Leonid Levin proved the same result in the Soviet Union. This result, now known as the Cook–Levin Theorem, established the first NP-complete problem and introduced a new method of reasoning about computational difficulty. The importance of the theorem lies in the following insight:

> If any NP-complete problem can be solved in polynomial time, then every problem in NP can be solved in polynomial time.

Shortly afterward, in 1972, Richard Karp published a paper identifying 21 natural combinatorial problems that are all polynomial-time equivalent to SAT. These problems came from diverse areas of computer science, including logic, graph theory, scheduling, integer optimization, and combinatorics.
Karp's work demonstrated that NP-completeness is not an isolated phenomenon but rather a structural property shared by a wide range of natural problems.

Since then, thousands of problems have been shown NP-complete.

In this worksheet, we explore several of the most fundamental NP-complete problems across logic, graphs, and optimization. Learning these canonical problems — and the reductions between them that we will see in class — provides a foundational toolkit for recognizing NP-completeness in new settings. Very often, when faced with a new computational problem, the first step toward proving NP-completeness is to reduce from one of these canonical problems.

For each of the following problems, identify which of the two instances given is the YES instance and which is the NO instance. Then, prove that the problem is in NP by describing the certificate/witness for the problem as well as why the certificate can be efficiently verified.

In the solution sections of the worksheet, we identify which instance is a YES and which is a NO instance. Furthermore, we will give a proof that the problem is in NP by arguing what the certificate/witness for the problem is as well as why the certificate can be efficiently verified.

1.1 3-SAT

Input. A Boolean formula φ in conjunctive normal form (CNF) where each clause contains exactly three literals. Let n be the number of variables and let m be the number of clauses. We view the instance size as being determined by (n, m).

Question. Is there a truth assignment to the n variables that satisfies all m clauses?

Determine which is a YES instance and which is a NO instance:

- **Instance A**

$$(x_1 \lor x_2 \lor \neg x_3) \land (\neg x_1 \lor x_3 \lor x_4) \land (x_2 \lor \neg x_4 \lor x_5)$$

- **Instance B**

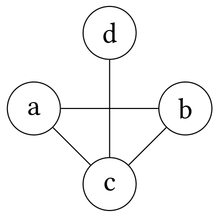
$$(x_1 \lor x_1 \lor x_1) \land (\neg x_1 \lor \neg x_1 \lor \neg x_1) \land (x_1 \lor x_1 \lor \neg x_1)$$

1.2 Vertex Cover

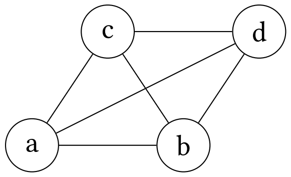
Input. A graph $G = (V, E)$ and an integer k. Let $n := |V|$ be the number of vertices and $m := |E|$ be the number of edges. We view the instance size as being determined by (n, m) (and k, which can be encoded in $O(\log n)$ bits).

Question. Is there a subset $S \subseteq V$ with $|S| \leq k$ such that every edge in E has at least one endpoint in S? We say that S *covers* an edge if at least one of its endpoints lies in S.

- **Instance A** $k = 2$



- **Instance B** $k = 2$



1.3 Integer Programming

Input. An integer matrix $A \in \mathbb{Z}^{m \times n}$, and a vector $b \in \mathbb{Z}^m$. We view the instance size as being determined by (m, n) together with the bit-length needed to write down the entries of A and b.

The goal of the problem is to decide if there exists an *integer* vector $x \in \mathbb{Z}^m$ such that

$$Ax \leq b, \text{ and } x \geq 0 .$$

Specifically, if

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \text{ then } Ax = \begin{pmatrix} a_{11}x_1 + \cdots + a_{1n}x_n \\ \vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n \end{pmatrix}.$$

Thus,

$$Ax \leq b \iff \begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n \leq b_1 \\ \vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n \leq b_m \end{cases}$$

Question. Does there exist an integer vector $x \geq 0$ satisfying both $Ax \leq b$ and $x \geq 0$?

- **Instance A** $\quad A \in \mathbb{Z}^{2\times 2}$

$$A = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

$$Ax \leq b = \begin{cases} x_1 + x_2 \leq 2 \\ 2x_1 + x_2 \leq 3 \end{cases}$$

- **Instance B** $\quad A \in \mathbb{Z}^{2\times 2}$

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$$

$$Ax \leq b = \begin{cases} x_1 + x_2 \leq 1 \\ x_1 + x_2 \geq 3 \end{cases}$$

1.4 Knapsack

Input. A list of items $1, \dots, n$, where item i has a nonnegative integer weight w_i and value v_i, along with a capacity W and target value T. We view the instance size as being determined by n together with the bit-length needed to write down the integers w_i, v_i, W, T.

Question. Is there a subset of items whose total weight is at most W and whose total value is at least T?

- **Instance A** $W = 10,\ T = 15$

Item	Weight	Value
1	5	8
2	4	7
3	6	9

- **Instance B** $W = 5,\ T = 12$

Item	Weight	Value
1	4	5
2	3	4
3	2	3

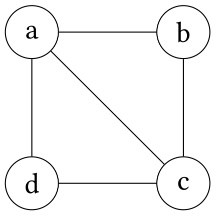
1.5 Hamiltonian Cycle

A *simple cycle* is any cycle that contains no repeated vertices.

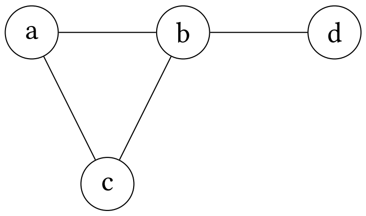
Input. A graph $G = (V, E)$. Let $n := |V|$ be the number of vertices and $m := |E|$ be the number of edges. We view the instance size as being determined by (n, m).

Question. Does G contain a simple cycle that visits every vertex exactly once?

- **Instance A**



- **Instance B**

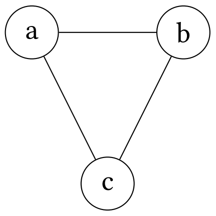


1.6 3-Coloring

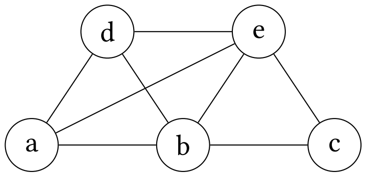
Input. A graph $G = (V, E)$. Let $n := |V|$ and $m := |E|$. We view the instance size as being determined by (n, m).

Question. Can each vertex be assigned one of three colors so that adjacent vertices receive different colors?

- **Instance A**



- **Instance B**



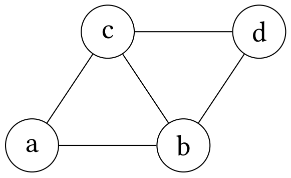
1.7 Clique

A *clique of size* k is a set of k vertices such that every pair of vertices in the set is connected by an edge.

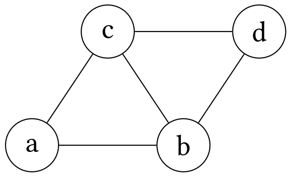
Input. A graph $G = (V, E)$ and an integer k. Let $n := |V|$ and $m := |E|$. We view the instance size as being determined by (n, m) (and k, which can be encoded in $O(\log n)$ bits).

Question. Is there a subset of k vertices that are all pairwise adjacent? Equivalently, does G contain a complete subgraph on k vertices?

- **Instance A** $k = 3$



- **Instance B** $k = 4$



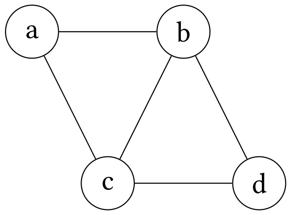
1.8 Max Cut

A *cut* (or partition) divides the vertices into two disjoint sets. An edge crosses the cut if its endpoints lie on opposite sides.

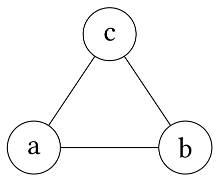
Input. A graph $G = (V, E)$ and an integer k. Let $n := |V|$ and $m := |E|$. We view the instance size as being determined by (n, m) (and k, which can be encoded in $O(\log m)$ bits).

Question. Is there a partition such that at least k edges cross the cut?

- **Instance A** $k = 3$



- **Instance B** $k = 5$



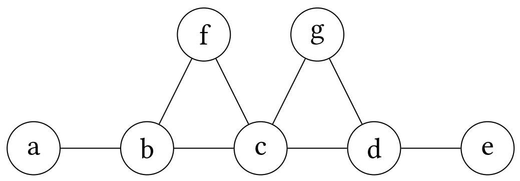
1.9 Longest Simple Path

A *simple path* is a path that does not repeat any vertex.
Input. A graph $G = (V, E)$ and an integer k. Let $n := |V|$ and $m := |E|$. We view the instance size as being determined by (n, m) (and k, which can be encoded in $O(\log n)$ bits).

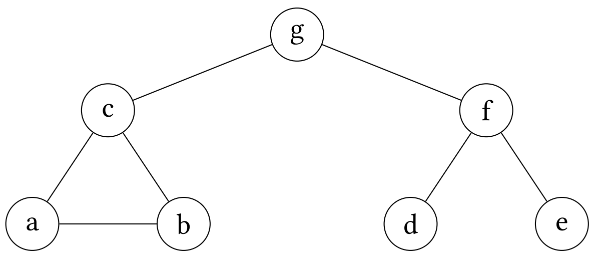
Question. Is there a simple path containing at least k vertices?

Remark. This contrasts with *Shortest Path*, which can be solved in polynomial time using BFS or Dijkstra's algorithm.

- **Instance A** $k = 7$



- **Instance B** $k = 7$



2 Mechanical calculations: SAT

Determine whether each instance of 3-SAT is satisfiable. If it is, list a satisfying variable assignment.

1. $(\neg a \vee \neg b \vee c) \wedge (a \vee c \vee \neg d) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee b \vee c) \wedge (\neg b \vee c \vee \neg d)$

2. $(\neg a \vee b \vee d) \wedge (\neg b \vee c \vee d) \wedge (a \vee \neg c \vee d) \wedge (a \vee \neg b \vee \neg d) \wedge (b \vee \neg c \vee \neg d) \wedge (\neg a \vee c \vee \neg d) \wedge (a \vee b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$

3. $(a \vee \neg c \vee d) \wedge (\neg a \vee b \vee c) \wedge (b \vee \neg c \vee \neg d) \wedge (a \vee \neg c \vee d) \wedge (a \vee \neg b \vee c) \wedge (\neg a \vee b \vee \neg d) \wedge (\neg a \vee c \vee d) \wedge (b \vee \neg c \vee d) \wedge (a \vee c \vee \neg d)$

4. $(\neg a \vee \neg b \vee c) \wedge (a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee \neg c) \wedge (\neg a \vee b \vee c) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee b \vee \neg c) \wedge (a \vee b \vee c) \wedge (a \vee \neg b \vee c)$

3 Solutions

3.1 A tour of canonical NP-complete problems

3.1.1 3-SAT

Verification: The certificate for a 3-SAT problem is the assignment vector $x \in \{0, 1\}^n$ which is polynomial length in the input.

Given an assignment, we evaluate each clause. Each clause contains exactly three literals, so evaluating one clause takes constant time. If there are m clauses, checking all clauses takes $O(m)$ time. Thus, verification is in polynomial time.

- **Instance A is YES.**

  A certificate is a truth assignment to the variables. For example:

  $$x_1 = 0, \quad x_2 = 1, \quad x_3 = 0, \quad x_4 = 0, \quad x_5 = 0.$$

  Under this assignment, each clause evaluates to true.

- **Instance B is NO.**

  The first clause forces $x_1 = 1$. The second clause forces $x_1 = 0$. No assignment satisfies both simultaneously. Therefore no certificate exists.

3.1.2 Vertex Cover

Verification: The certificate for a Vertex Cover problem is a subset S of the input graph's vertices .

To verify a vertex cover, we check every edge. For each edge, we test whether one of its endpoints lies in S. If there are m edges, this requires checking m edges. Each membership test can be done in constant time using a boolean array. Thus verification takes $O(m)$ time.

- **Instance A is YES.** A certificate is the set $S = \{a, c\}$. Every edge has at least one endpoint in S.
- **Instance B is NO.**

  Any set of size two fails to cover at least one edge. Hence no certificate of size $k = 2$ exists.

3.1.3 Integer Programming

Verification: The certificate for an Integer Programming problem is a vector x with nonnegative integer entries of length n.

To verify a candidate vector x, we compute Ax. Matrix multiplication for an $m \times n$ matrix takes $O(mn)$ time. Then we compare each coordinate of Ax to the corresponding entry of b, which takes $O(m)$ time. Thus verification is polynomial time.

- **Instance A is YES.**

  A certificate is the vector

  $$x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

  Substituting:

  $$1 + 1 \le 2, \quad 2(1) + 1 \le 3.$$

  Both inequalities hold.

- **Instance B is NO.**

  The constraints require:

  $$x_1 + x_2 \le 1 \quad \text{and} \quad x_1 + x_2 \ge 3.$$

  No nonnegative integer vector satisfies both simultaneously. Therefore no certificate exists.

3.1.4 Knapsack

Verification: The certificate for a Knapsack problem is a subset $S \subseteq [n]$ of the input items.

To verify a subset, we sum the weights and values. If there are n items, this requires scanning the list once, which takes $O(n)$ time. Then we compare to W and T. Thus verification is polynomial time.

- **Instance A is YES.**

  A certificate is the subset of items $\{1, 2\}$.

  Total weight:

  $$5 + 4 = 9 \leq 10.$$

  Total value:

  $$8 + 7 = 15 \geq 15.$$

- **Instance B is NO.**

  Any subset of items either exceeds weight 5 or has total value strictly less than 12. Hence no certificate exists.

3.1.5 Hamiltonian Cycle

Verification: The certificate for a Hamiltonian Cycle problem is a sequence of the input graph's edges that are adjacent to each other.

To verify a Hamiltonian cycle:

- Check that the sequence contains each vertex exactly once. This can be done using a boolean array in $O(n)$ time.
- Check that each consecutive pair of vertices is connected by an edge. There are n edges to check, so this takes $O(n)$ time.

Thus verification is polynomial time.

- **Instance A is YES.**

  A certificate is the cycle:

  $$a \to b \to c \to d \to a.$$

  This visits each vertex exactly once and returns to the start.

- **Instance B is NO.**

  Vertex d has degree 1, so it cannot lie on a simple cycle that visits every vertex. Hence no Hamiltonian cycle exists.

3.1.6 3-Coloring

Verification: The certificate for a 3-coloring problem is an assignment $C : V \to \{0, 1, 2\}$ of each of the input graph's vertices to one of 3 colors.

To verify a coloring, check every edge. For each edge, confirm the endpoints have different colors. If there are m edges, this takes $O(m)$ time. Thus verification is polynomial time.

- **Instance A is YES.**

  A certificate is a coloring:

  $$a = \text{red}, \quad b = \text{blue}, \quad c = \text{green}.$$

  Adjacent vertices receive different colors.

- **Instance B is NO.**

  The graph contains a substructure that forces four mutually constrained vertices, requiring four distinct colors. Thus no valid 3-coloring exists.

3.1.7 Clique

Verification: The certificate for a Clique problem is a subset $S \subseteq V$ of the input graph's vertices.

To verify a clique of size k, we check every pair of vertices in the proposed set. There are $\binom{k}{2}$ pairs. Checking adjacency for each pair yields $O(k^2)$ time. Since $k \leq n$, this is polynomial.

- **Instance A is YES.**

  A certificate is the vertex set

  $$\{a, b, c\}.$$

  Every pair among these vertices is connected by an edge.

- **Instance B is NO.**

  The graph has only four vertices but is missing at least one edge among any four, so no complete subgraph of size 4 exists.

3.1.8 Max Cut

Verification: The certificate for a Max Cut problem is a partition of the input graph's vertices into two subsets (a cut): $(S, V \setminus S)$.

To verify a cut, we check every edge and count how many have endpoints in opposite sets. If there are m edges, this takes $O(m)$ time. Thus verification is polynomial.

- **Instance A is YES.**

  A certificate is the partition:

  $$\{a, c\} \mid \{b, d\}.$$

  Under this partition, at least 3 edges cross the cut.

- **Instance B is NO.**

  The graph has only three edges total, so no partition can produce 5 crossing edges.

3.1.9 Longest Simple Path

Verification: The certificate for a Longest Simple Path problem is a sequence of the input graph's edges.

To verify a simple path:

- Check that vertices are not repeated (use a boolean array in $O(k)$).
- Check that consecutive vertices are adjacent (at most k checks).

Thus verification is polynomial time.

- **Instance A is YES.**

  A certificate is the path:

  $$a \to b \to f \to c \to g \to d \to e.$$

  This is a simple path with 4 vertices.

- **Instance B is NO.**

  The graph contains no simple path containing all 7 vertices.

3.2 Mechanical calculations

1. This is satisfiable: $a = T, b = F, c = T, d = F$ makes each clause true, so the overall formula is true.
2. This is unsatisfiable. (Brute force) One possible way of verifying this is by enumerating all 16 possibilities and seeing that a clause is unsatisfied in each instance.
3. This is satisfiable: $a = F, b = T, c = T, d = T$ makes each clause true, so the overall formula is true.
4. This is unsatisfiable. One can notice that each of the clauses disallows exactly one of the 8 possible assignments to the three variables. As no clauses are equal and there are 8 such clauses, no possible assignment is viable.