1. Consider the following problem, called BoxDEPTH: Given a set of \( n \) axis-aligned rectangles in the plane, how big is the largest subset of these rectangles that contain a common point?

(a) Describe a polynomial-time reduction from BoxDEPTH to MaxClique.

**Solution:** Given a set \( R \) axis-aligned rectangles in the plane, we construct an undirected graph \( G = (V, E) \) as follows:

- \( V = R \)
- \( E = \{ \{r, s\} | r \cap s \neq \emptyset \} \)

That is, \( G \) is the intersection graph of the rectangles in \( R \). We can easily construct \( G \) in brute force in \( O(n^2) \) time, where \( n = |R| \). Yes, that’s the entire reduction.

To prove the reduction is correct, we need to prove that \( G \) has a clique of size \( k \) if and only if some subset of \( k \) rectangles contains a common point.

\[ \implies \] Suppose \( G \) contains \( k \) vertices \( r_1, r_2, \ldots, r_k \) such that \( (r_i, r_j) \in E \) for all indices \( i \) and \( j \) where \( 1 \leq i < j \leq k \). Each rectangle \( r_i \) is the product of two intervals

\[ r_i = [\min x_i, \max x_i] \times [\min y_i, \max y_i], \]

where \( \min x_i < \max x_i \) and \( \min y_i < \max y_i \). For all indices \( i \) and \( j \), rectangles \( r_i \) and \( r_j \) have non-empty intersection, which implies the following inequalities:

\[ \min x_i < \max x_j \quad \min y_i < \max y_j \quad \min x_j < \max x_i \quad \min y_j < \max y_i. \]

Now define four values:

\[ \overline{\min x} := \max_i \min x_i \quad \overline{\max x} := \min_j \max x_j \]
\[ \overline{\min y} := \max_i \min y_i \quad \overline{\max y} := \min_j \max y_j \]

Since the previous inequalities hold for all \( i \) and \( j \), we infer that \( \overline{\min x} < \overline{\max x} \) and \( \overline{\min y} < \overline{\max y} \), so the rectangle

\[ \bigcap_{i=1}^k r_i = [\overline{\min x}, \overline{\max x}] \times [\overline{\min y}, \overline{\max y}] \]

is non-empty. Thus, \( R \) contains a subset of \( k \) rectangles with a non-empty common intersection.

\[ \iff \] Suppose there are \( k \) rectangles \( r_1, r_2, \ldots, r_k \in R \) that all contain a common point \( p \). Then for all indices \( i \) and \( j \) such that \( 1 \leq i < j \leq k \), we have \( p \in r_i \cap r_j \) and therefore \( r_i \cap r_j \neq \emptyset \), which implies \( (r_i, r_j) \in E \). We conclude that \( G \) contains a clique of size \( k \).

\[ \blacksquare \]

**Rubric:** 5 points: standard poly-time reduction rubric (scaled)
(b) Describe and analyze a polynomial-time algorithm for \text{BoxDepth}.

\textbf{Solution:} Intuitively, the following algorithm extends the sides of the rectangles into infinite lines, splitting the plane into an irregular \((2n + 1) \times (2n + 1)\) grid of boxes. Then for each of those \(O(n^2)\) boxes, the algorithm chooses a point in the interior of the box, and then counts the number of input rectangles that contains it in \(O(n)\) time by brute force.

\begin{verbatim}
BOXDEPTH(minx[1..n], maxx[1..n], miny[1..n], maxy[1..n]):
   ⟨⟨— Extend the box sides into a grid ——⟩⟩
   for i ← 1 to n
      X[i] ← minx[i];    X[n + i] ← maxx[i]
      Y[i] ← miny[i];    Y[n + i] ← maxy[i]
   sort X
   sort Y
   ⟨⟨— Compute the depth of each grid cell and keep max ——⟩⟩
   maxdepth ← 0
   for i ← 0 to 2n
      for j ← 0 to 2n
         ⟨⟨find a point \((x, y)\) inside cell \((i, j)\)⟩⟩
         x ← (X[i] + X[i + 1])/2
         y ← (Y[j] + Y[j + 1])/2
         ⟨⟨count boxes containing \((x, y)\)⟩⟩
         depth ← 0
         for k ← 1 to n
            if minx[k] ≤ x ≤ maxx[k] and miny[k] ≤ y ≤ maxy[k]
               depth ← depth + 1
         maxdepth ← max{maxdepth, depth}
   return maxdepth
\end{verbatim}

This algorithm runs in \(O(n^3)\) time. ■

\textbf{Rubric:} 4 points. The English description is enough for full credit; the pseudocode is not necessary. This is not the only correct algorithm, nor the fastest algorithm; any correct polynomial-time algorithm is enough for full credit.

(c) Why don’t these two results imply that \(P=NP\)?

\textbf{Solution:} The reduction in part (a) goes in the wrong direction. If we want to prove that \text{BoxDepth} is NP-hard, or that \text{MaxClique} can be solved in polynomial time, we need to describe a reduction \textit{from} \text{MaxClique} \textit{to} \text{BoxDepth}. ■

\textbf{Rubric:} 1 point; all or nothing.
2. (a) Describe a polynomial-time reduction from \textsc{SubsetSum} to \textsc{Partition}.

**Solution (add two elements):** Let \(x\) denote the sum of the elements of \(X\).
- If \(k > x\), then \(\textsc{SubsetSum}(X, k) = \text{FALSE}\).
- Otherwise, if \(k = x/2\), then \(\textsc{SubsetSum}(X, k) = \textsc{Partition}(X)\).
- Otherwise, \(\textsc{SubsetSum}(X, k) = \textsc{Partition}(X \cup \{2x + k, 3x - k\})\).

The reduction is trivially correct when \(k > x\), and the proof of correctness when \(k = x/2\) is the same as part (a), so let’s assume that \(k \leq x\) and \(k \neq x/2\). Let \(Y = X \cup \{3x - k, 2x + k\}\). Observe that \(\sum Y = 6x\) and \(2x + k \neq 3x - k\).

\(\implies\) Suppose \(X\) has a subset \(X’\) whose elements sum to \(k\). Let \(Y_1 = X’ \cup \{3x - k\}\), and let \(Y_2 = Y \setminus Y_1\). and \(\sum Y_1 = \sum X + 3x - k = 3x\), so \(\sum Y_2 = 3x\) as well. So \(Y\) can be partitioned into two subsets with the same sum.

\(\impliedby\) On the other hand, suppose \(Y\) can be partitioned into two subsets \(Y_1\) and \(Y_2\) with the same sum, which must be \(3x\). Neither set can contain both \(2x + k\) and \(3x - k\), because \((2x + k) + (3x - k) = 3x > x\). So suppose without loss of generality that \(3x - k \in Y_1\), and let \(X’ = Y_1 \setminus \{3x - k\}\). Then \(X’\) is a subset of \(X\) whose elements sum to \(k\).

The reduction takes \(O(n)\) time (to compute \(x\)).

**Solution (add one element):** Let \(x\) denote the sum of the elements of \(X\).
- If \(k = x/2\), then clearly \(\textsc{SubsetSum}(X, k) = \textsc{Partition}(X)\).
- Otherwise, \(\textsc{SubsetSum}(X, k) = \textsc{Partition}(X \cup \{|x - 2k|\})\).

The reduction is trivially correct when \(k > x\), and the proof of correctness when \(k = x/2\) is the same as part (a), so let’s assume that \(k \leq x\) and \(k \neq x/2\).

Let \(Y = X \cup \{|x - 2k|\}\), and let \(y = \sum Y\). If \(x > 2k\), we have \(y = 2x - 2k\); otherwise, we have \(y = 2k\).

\(\implies\) Suppose \(X\) has a subset \(X’\) whose elements sum to \(k\). If \(x > 2k\), let \(Y’ = X’ \cup \{|x - 2k|\}\); in this case, we have \(\sum Y’ = x - k = y/2\). On the other hand, if \(x < 2k\), let \(Y’ = X’\); in this case, we have \(\sum Y’ = k = y/2\). In either case \(Y\) can be partitioned into two subsets (\(Y’\) and \(Y \setminus Y’\)) with the same sum.

\(\impliedby\) On the other hand, suppose \(Y\) can be partitioned into two subsets \(Y_1\) and \(Y_2\) with the same sum. Without loss of generality, suppose \(|x - 2k| \in Y_1\). If \(x > 2k\), let \(X’ = Y_1 \setminus \{x - 2k\}\); otherwise, let \(X’ = Y_2\). In both cases, \(X’\) is a subset of \(X\) whose elements sum to \(k\).

The reduction takes \(O(n)\) time (to compute \(x\)).

**Rubric:** 5 points standard reduction rubric (scaled). These are not the only correct reductions. No penalty if the output of the reduction could be a multiset instead of a set. \(-\frac{1}{2}\) if the output of the reduction can contain negative integers.
(b) Describe a polynomial-time reduction from \textsc{Partition} to \textsc{SubsetSum}.

\textbf{Solution:}  \textsc{Partition}(Y) = \textsc{SubsetSum}(Y, y/2), where $y$ is the sum of the elements of $Y$.

$\implies$ Suppose $Y$ can be partitioned into two subsets $Y_1$ and $Y_2$ with the same sum. Then $\sum Y_1 = \sum Y_2 = y/2$. So $Y$ has a subset whose elements sum to $y/2$.

$\impliedby$ On the other hand, suppose $Y$ has a subset $Z$ whose elements sum to $y/2$. Then the elements of $Y \setminus Z$ sum to $y/2$ as well, which implies that $\sum Z = \sum (Y \setminus Z)$.

So $Y$ can be partitioned into two subsets with the same sum.

The reduction requires $O(n)$ time (to compute $y$).

\begin{rubric}
5 points standard reduction rubric (scaled). This is not the only correct reduction.
\end{rubric}
3. Prove that the following problem is NP-hard: Given an undirected graph \( G \) and an integer \( k \), decide whether the vertices of \( G \) can be partitioned into \( k \) cliques.

**Solution:** We prove the problem is NP-hard using a reduction from \( k\text{COLOR} \), which we proved NP-hard in Friday’s lab.

Let \( G = (V, E) \) be an arbitrary undirected graph. Let \( \overline{G} = (V, \overline{E}) \) denote the edge-complement of \( G \), where \( uv \in \overline{E} \) if and only if \( uv \notin E \), for all vertices \( u \) and \( v \). I claim that \( G \) is \( k \)-colorable if and only if the vertices of \( \overline{G} \) can be partitioned into \( k \) cliques.

\[ \implies \] Suppose \( G \) is \( k \)-colorable. Fix a proper \( k \)-coloring, and let \( V_1, V_2, \ldots, V_k \) be the subsets of \( V \) of each color. By definition of “proper coloring”, for every index \( i \) and every pair of indices \( u, v \in V_i \), we have \( uv \notin E \). Thus, for every index \( i \) and every pair of indices \( u, v \in V_i \), we have \( uv \in \overline{E} \). In other words, each subset \( V_i \) is a clique in \( \overline{G} \). We conclude that the vertices of \( \overline{G} \) can be partitioned into \( k \) cliques.

\[ \iff \] Suppose the vertices of \( \overline{G} \) can be partitioned into \( k \) cliques \( V_1, V_2, \ldots, V_k \). By definition of “clique”, for every index \( i \) and every pair of indices \( u, v \in V_i \), we have \( uv \in \overline{E} \). Thus, for every index \( i \) and every pair of indices \( u, v \in V_i \), we have \( uv \notin E \). In other words, each subset \( V_i \) is an independent set in \( \overline{G} \); equivalently, if we assign “color” \( i \) to each vertex in \( V_i \), for every index \( i \), we obtain a proper coloring of \( G \). We conclude that \( G \) is \( k \)-colorable.

We can build \( \overline{G} \) from \( G \) in polynomial time by brute force. ■

**Rubric:** 10 points: Standard reduction rubric. This is more detail than necessary for full credit.
Solved Problem

4. Consider the following solitaire game. The puzzle consists of an \( n \times m \) grid of squares, where each square may be empty, occupied by a red stone, or occupied by a blue stone. The goal of the puzzle is to remove some of the given stones so that the remaining stones satisfy two conditions: (1) every row contains at least one stone, and (2) no column contains stones of both colors. For some initial configurations of stones, reaching this goal is impossible.

A solvable puzzle and one of its many solutions. An unsolvable puzzle.

Prove that it is NP-hard to determine, given an initial configuration of red and blue stones, whether this puzzle can be solved.

Solution: We show that this puzzle is NP-hard by describing a reduction from 3SAT.

Let \( \Phi \) be a 3CNF boolean formula with \( m \) variables and \( n \) clauses. We transform this formula into a puzzle configuration in polynomial time as follows. The size of the board is \( n \times m \). The stones are placed as follows, for all indices \( i \) and \( j \):

- If the variable \( x_j \) appears in the \( i \)th clause of \( \Phi \), we place a blue stone at \((i, j)\).
- If the negated variable \( \overline{x_j} \) appears in the \( i \)th clause of \( \Phi \), we place a red stone at \((i, j)\).
- Otherwise, we leave cell \((i, j)\) blank.

We claim that this puzzle has a solution if and only if \( \Phi \) is satisfiable. This claim immediately implies that solving the puzzle is NP-hard. We prove our claim as follows:

\( \implies \) First, suppose \( \Phi \) is satisfiable; consider an arbitrary satisfying assignment. For each index \( j \), remove stones from column \( j \) according to the value assigned to \( x_j \):

- If \( x_j = \text{True} \), remove all red stones from column \( j \).
- If \( x_j = \text{False} \), remove all blue stones from column \( j \).

In other words, remove precisely the stones that correspond to \( \text{False} \) literals. Because every variable appears in at least one clause, each column now contains stones of only one color (if any). On the other hand, each clause of \( \Phi \) must contain at least one \( \text{True} \) literal, and thus each row still contains at least one stone. We conclude that the puzzle is satisfiable.

\( \impliedby \) On the other hand, suppose the puzzle is solvable; consider an arbitrary solution. For each index \( j \), assign a value to \( x_j \) depending on the colors of stones left in column \( j \):

- If column \( j \) contains blue stones, set \( x_j = \text{True} \).
- If column \( j \) contains red stones, set \( x_j = \text{False} \).
- If column \( j \) is empty, set \( x_j \) arbitrarily.

In other words, assign values to the variables so that the literals corresponding to the remaining stones are all \( \text{True} \). Each row still has at least one stone, so each clause of \( \Phi \) contains at least one \( \text{True} \) literal, so this assignment makes \( \Phi = \text{True} \). We conclude that \( \Phi \) is satisfiable.
This reduction clearly requires only polynomial time.

**Rubric (for all polynomial-time reductions):**

- 10 points = 
  - + 3 points for the reduction itself
    - For an NP-hardness proof, the reduction must be from a known NP-hard problem. You can use any of the NP-hard problems listed in the lecture notes (except the one you are trying to prove NP-hard, of course).
  - + 3 points for the “if” proof of correctness
  - + 3 points for the “only if” proof of correctness
  - + 1 point for writing “polynomial time”

- An incorrect polynomial-time reduction that still satisfies half of the correctness proof is worth at most 4/10.
- A reduction in the wrong direction is worth 0/10.