1. Recall that $L_u = \{\langle M, w\rangle \mid M \text{ accepts } w\}$ is language of a UTM, and $L_{HALT} = \{\langle M\rangle \mid M \text{ halts on blank input}\}$ is the Halting language.

   - Let $L_{374\text{A}} = \{\langle M\rangle \mid M \text{ accepts at least 374 distinct input strings}\}$. Prove that $L_{374\text{A}}$ is undecidable.

Solution: For the sake of contradiction, suppose there is an algorithm DECIDEL374A that correctly decides the language L_{374A}. Then we can solve L_{HALT}:

```
DECIDEHALT(⟨M⟩):
  Encode the following Turing machine M′:
      M′(x):
        run M on blank input
        return TRUE
  if DECIDEL374A(⟨M′⟩)
      return TRUE
  else
      return FALSE
```

We prove this reduction correct as follows:

$\Longrightarrow$ Suppose M halts blank input.
Then M' halts on and accepts *every* input string x.
Therefore, M' halts on at least 374 distinct input strings.
So DECIDEL374A accepts the encoding $\langle M'\rangle$.
So DECIDEHALT correctly accepts the encoding $\langle M, w\rangle$.

$\Longleftarrow$ Suppose M does not halt on blank input.
Then M' diverges (i.e., does not halt) on *every* input string x.
Therefore, M' does not halt on at least 374 distinct input strings.
So DECIDEL374A rejects the encoding $\langle M'\rangle$.
So DECIDEHALT correctly rejects the encoding $\langle M, w\rangle$.

In both cases, DECIDEHALT is correct, that is, there exists a decider for L_{HALT}. But that is impossible, because L_{HALT} is undecidable. We conclude that the algorithm DECIDEL374A does not exist, i.e., L_{374A} is undecidable. ■

   - Prove that $L_u \leq L_{HALT}$

Solution: We will reduce L_u to L_{HALT} as follows. Given $\langle M, w\rangle$, let M' be the following Turing Machine:

```
M′():
  run M on input w
  if M accepted
      return TRUE
  else
      loop forever
```

We claim that $\langle M, w\rangle \in L_u$ if and only if $\left\langle M'\right\rangle \in L_{HALT}$.

$\Longrightarrow$ Suppose M accepts on input w. Then M' halts on blank input.

$\Longleftarrow$ Suppose M does not accept input w, then either M halts and rejects, or does not halt. In either case, M' does not halt on blank input. ■

Rubric: 10 points. 5 points each: scaled Undecidability reduction rubric (see last page).

2. Consider an instance of the Satisfiability Problem, specified by clauses $C_1, \dots, C_m$ over a set of Boolean variables $x_1, \dots, x_n$. We say that the instance is *monotone* if each term in each clause consists of a nonnegated variable; that is each term is equal to x_i, for some i, rather than $\bar{x}_i$. Monotone instance of Satisfiability are very easy to solve: They are always satisfiable, by setting each variable equal to 1.

   For example, suppose we have the three clauses

$$(x_1 \vee x_2), (x_1 \vee x_3), (x_2 \vee x_3)$$

   This is monotone, and indeed the assignment that sets all three variables to 1 satisfies all the clauses. But we can observe that this is not the only satisfying assignment; we could also have set x_1 and x_2 to 1 and x_3 to 0. Indeed, for any monotone instance, it is natural to ask how few variables we need to set to 1 in order to satisfy it.

   Given a monotone instance of Satisfiability, together with a number k, the problem of *Monotone Satisfiability with Few True Variables* asks: Is there a satisfying assignment for the instance in which at most k variables are set to 1? Describe a polynomial time reduction from Vertex Cover to this problem. You should also prove the correctness of the reduction.

Solution: The input to the Vertex Cover problem is the graph $G = (V, E)$ and integer k, while the input to the Monotone Satisfiability with Few True Variables (MSFTV) problem is a boolean formula Φ and another integer ℓ.

We reduce Vertex Cover to MSFTV as follows. For each vertex $u \in V$, we define a variable x_u. A variable being 1 in a satisfying assignment of Φ will imply that the corresponding vertex is in the vertex cover.

A solution to the Vertex Cover must satisfy the criteria that for each edge uv, at least one of the two endpoints u or v must be in the vertex cover. Correspondingly, we will define the following clause for each edge $uv \in E$: $(x_u \vee x_v)$. Each clause ensures that at least one of the variables corresponding to the endpoings of the edge is set to 1.

In summary, $\Phi = \bigwedge_{uv \in E}(x_u \vee x_v)$, which is a monotone instance. We will set $\ell = k$.

We claim that G has a vertex cover of size at most k if and only if Φ has a satisfying assignment in which at most k variables are set to 1.

$\Longrightarrow$ Let $S \subseteq V$ be a vertex cover of G of size at most k. We set variable x_u to 1 if $u \in S$, and x_u to 0 otherwise. This sets at most k variables to 1. For each edge $uv \in E$, we know that $u \in S$ or $v \in S$, i.e., $(x_u \vee x_v)$ is satisfied, so we have a satisfying assignment to Φ with at most k variables set to 1.

$\Longleftarrow$ Take a satisfying assignment x to Φ with at most k variables set to 1. Define the set $S := \{u \in V \mid x_u = 1\}$. This set consists of at most k vertices. For each edge $uv \in E$, the corresponding clause $(x_u \vee x_v)$ is satisfied, so we know that $x_u = 1$ or $x_v = 1$, i.e., $u \in S$ or $v \in S$, so we have a vertex cover of G of size at most k.

Φ can clearly be constructed in time polynomial (in fact linear) in the size of G, so this is a polynomial time reduction. ■

Rubric: 10 points. Standard polynomial-time reduction rubric (see last page).

3. Given an undirected graph $G = (V, E)$, a partition of V into $V_1, V_2, \ldots, V_k$ is said to be a clique cover of size k if each V_i is a clique in G. CLIQUE-COVER is the following decision problem: given G and integer k, does G have a clique cover of size at most k?

   - Describe a polynomial-time reduction from CLIQUE-COVER to SAT. Does this prove that CLIQUE-COVER is NP-Complete? For this part you just need to describe the reduction clearly, no proof of correctness is necessary. *Hint:* Use variable $x(u, i)$ to indicate that node u is in partition i.

Solution: The input to the CLIQUE-COVER problem is the graph $G = (V, E)$ and integer k, while input to the SAT problem is a boolean formula Φ.

We reduce CLIQUE-COVER to SAT as follows. For each vertex $u \in V$, we define k variables $x(u, i)$, $1 \leq i \leq k$. A variable being 1 in a satisfying assignment of Φ will imply that the corresponding node is in the partition i in the CLIQUE-COVER instance.

A solution to the CLIQUE-COVER problem must satisfy some criteria. To have the corresponding SAT solution reflect these criteria, we define different types of clauses, one for each criterion, for the SAT instance as follows.

- To encode the constraint that non-adjacent vertices cannot belong in the same clique, we will define k clauses for each non-edge $uv \notin E$: $(\overline{x(v, i)} \vee \overline{x(u, i)})$, $1 \leq i \leq k$. These clauses ensure that in a satisfying assignment, variables corresponding to two non-adjacent vertices and the same partition cannot both be 1.
- To encode the constraint that every vertex should belong in exactly one partition, we define the following clauses for each vertex u: $\left(\bigvee_{1 \leq i \leq k} x(u, i)\right) \wedge \left(\bigwedge_{1 \leq i < j \leq k} (\overline{x(u, i)} \vee \overline{x(u, j)})\right)$. For each vertex u, the first clause requires that at least one of the variables $x(u, i)$ is set to 1 in a satisfying assignment; the subsequent clauses ensure that two variables $x(u, i)$ and $x(u, j)$ where $i \neq j$ cannot both be simultaneously true in a satisfying assignment. Together these clauses ensure that in a satisfying assignment, exactly one of the variables $x(u, i)$ can be set to 1.

The boolean formula Φ is simply the AND of all the above clauses.

If the graph has n vertices, Φ is a formula of exactly nk variables. Counting the clauses of the first type, we see that Φ has k clauses of size 2 for each non-edge, and there can be at most $\binom{n}{2}$ non-edges. As for the clauses of the second type, for each vertex we have a clause of size k, as well as $\binom{k}{2}$ clauses of size 2. Thus, the total number of clauses is at most $O(k^2 + kn^2)$, which is polynomial in the size of the CLIQUE-COVER instance. ■

- Describe a polynomial-time reduction from k-Color to CLIQUE-COVER. You should also prove the correctness of the reduction.

Solution: The input to k-Color is a graph $G = (V, E)$.

We define the complement of the graph G, denoted by $\overline{G} = (V, \overline{E})$, as the graph where the vertex set is the same, and the edges are all non-edges of G, that is $uv \in \overline{E} \iff uv \notin E$.

We claim that G has a proper coloring using at most k colors if and only if $\overline{G}$ has a Clique-cover of size at most k.

$\Longrightarrow$ Given a proper coloring of G, we assign all vertices of color i to a set V_i. Each set V_i colored by one color in a proper coloring of G forms an independent set (no pair of vertices is adjacent), and thus forms a clique (every pair of vertices is adjacent) in $\overline{G}$. So if G can be colored using k colors, there will be k cliques in $\overline{G}$.

$\Longleftarrow$ Given a clique cover of size k, we color all vertices in the same clique V_i using color i. Every pair of vertices in V_i are adjacent in $\overline{G}$, and thus non-adjacent in G. As a result, coloring them using the same color does not violate coloring constraints.

$\overline{G}$ can clearly be constructed in time polynomial in the size of G, so this is a polynomial time reduction. ■

Rubric: 10 points.

- 5 points:
  - + 1 point for the edge constraints (non-adjacent vertices cannot be in the same clique).
  - + 2 points for the vertex constraints (a vertex must be assigned to exactly one clique).
  - + 1 point for describing the resulting formula.
  - + 1 point for writing "polynomial time".
  - – A reduction in the wrong direction is worth 0/10.
- 5 points: scaled Standard polynomial-time reduction rubric (see last page).

Rubric (Standard undecidability reduction rubric): 10 points =

+ 2 points for the reduction itself
+ 4 points for the "if" proof of correctness
+ 4 points for the "only if" proof of correctness

- A reduction in the wrong direction is worth 0/10.

Rubric (Standard polynomial-time reduction rubric): 10 points =

+ 3 points for the reduction itself
+ 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.