NP-HARDNESS REVIEW, UNDECIDABILITY.

A subset $S$ of vertices in an undirected graph $G$ is *almost independent* if at most 374 edges in $G$ have both endpoints in $S$. Prove that finding the size of the largest almost-independent set of vertices in a given undirected graph is NP-hard.

![Graph diagram]

\[ S \text{ is independent set in } G \implies S \cup C \text{ is almost independent set in } G' \]

? $S \cup C$ is almost independent set in $G' \implies S$ is independent set in $G$.

Let $S'$ be an almost independent set in $G'$.

Suppose $S'$ does not contain a vertex in $C$.

Then you can add a vertex in $C$ to $S'$, remove one from $G$.

The resulting set $S''$ is still an almost-independent set, where $|S''| = |S'| + 1$.

There is at least 1 largest almost-independent set in $G'$ that contains all of $C$. Removing $C \implies$ independent set in $G$. 
A subset $S$ of vertices in an undirected graph $G$ is sort-of-independent if if each vertex in $S$ is adjacent to at most 374 other vertices in $S$. Prove that finding the size of the largest sort-of-independent set of vertices in a given undirected graph is NP-hard.

If $S$ is ind set in $G$

$\Rightarrow$ $S \cup L$ is sort of ind set in $G'$.

Claim:

Some largest sort of ind set in $G'$ should contain all dummy vertices.

Call it $S' \cup L$

$\Rightarrow S'$ is a largest ind. set in $G$. 

Proof:

Remove

$\Rightarrow$

Add
NP-hard: no fast algorithm
Undecidable: no algorithm.

**HALTING PROBLEM.**

Problem: Given \( M \) represented as \( \langle M \rangle \).

- Turing machine
- String encoding
description of TM
- Source code
- Executable program

Language = collection of strings.

Turing Machine \( Y \) decides language \( L \) if

\[
\forall x \in L, \quad Y(x) = 1 \\
\forall x \notin L, \quad Y(x) = 0.
\]

Q: Can every language be decided by a Turing Machine?

Language of satisfiable Boolean formulae is decidable.

\[
L_{\text{SELFHALT}} = \{ \langle M \rangle \mid M \text{ on input } \langle M \rangle \text{ halts?} \}
\]
H. P. is undecidable

There is no TM that given \( x \) outputs 1 when \( x \in L_{\text{SELFNOTHALT}} \), 0 otherwise.

\[
L_{\text{SELFNOTHALT}} = \frac{1}{2} <M> | M \text{ on input } <M> \text{ does not halt}
\]

**CLAIM:** \( L_{\text{SELFNOTHALT}} \) is undecidable.

**Proof:** Suppose that \( L_{\text{SNH}} \) is decidable.

This \( M_1 \) exists.

\[
\begin{array}{c}
\langle x \rangle \\
\downarrow
\end{array}
\begin{array}{c}
\text{If } x \text{ doesn't halt on } <x> \\
\text{X does halt on } <x>
\end{array}
\begin{array}{c}
\text{YES} \\
\text{NO}
\end{array}
\]

\( M_1 \) exists

\[ \Rightarrow \exists M_2 \]

\[
\begin{array}{c}
\langle x \rangle \\
\downarrow
\end{array}
\begin{array}{c}
\text{If } x \text{ doesn't halt on } <x> \\
\text{X does halt on } <x>
\end{array}
\begin{array}{c}
\text{YES} \\
\text{Loop forever}
\end{array}
\]

\( M_2 (\langle x \rangle) \) halts \( \iff \) \( x \) doesn't halt on input \( <x> \)

\( M_2 (\langle x \rangle) \) loops forever \( \iff \) \( x \) halts on input \( <x> \)

What if \( x = M_2 \) ?

\( M_2 <M_2> \) halts \( \iff \) \( M_2 <M_2> \) doesn't halt.

**OOPS!** Contradiction.
Claim: [If \( L_{SNH} \) is undecidable, then \( L_{SH} \) is also undecidable.]

Suppose self-halt were decidable.

Then

Self not halt is also decidable.
But we know SNH is undecidable.
\( \implies \) SH is also undecidable.

\[ L_{HALT} = \{ \langle M \rangle, x \mid \text{such that } M \text{ on input } x \text{ halts} \} \]

Is \( L_{HALT} \) decidable?
Suppose it is.

\[ \langle M \rangle, x \]

Is \( L_{HALT} \) decidable?
Suppose it is.
\[ L_{\text{SELFHALT}} \]

\[ \langle M \rangle \rightarrow \langle M, x = \text{HALT} \rangle \rightarrow \text{YES} \]

\[ \rightarrow \text{NO} \rightarrow \text{NO} \rightarrow \text{YES} \]

\[ \Rightarrow \text{L}_{\text{SELFHALT}} \text{ is decidable.} \]

\[ \therefore \text{HALT is not decidable.} \]