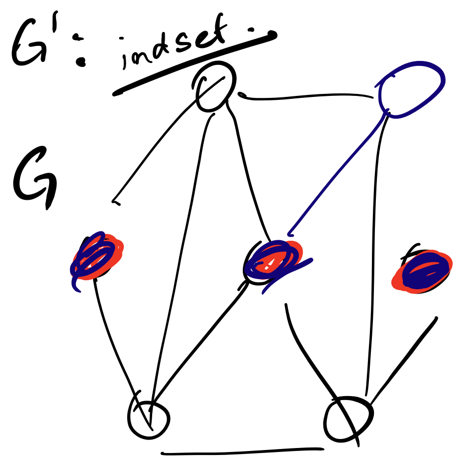
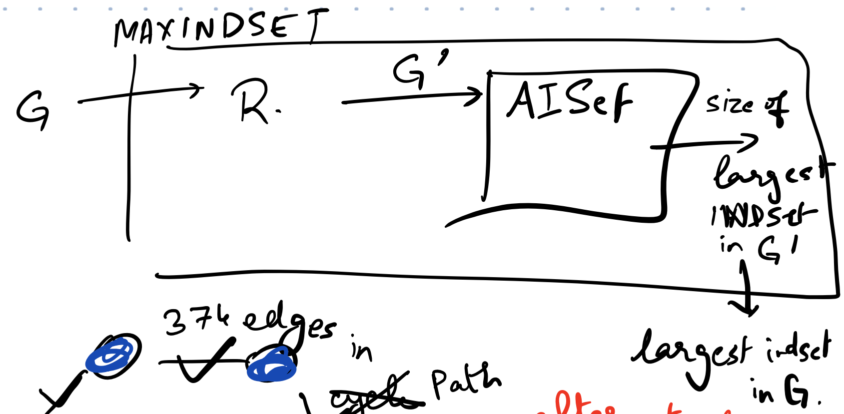
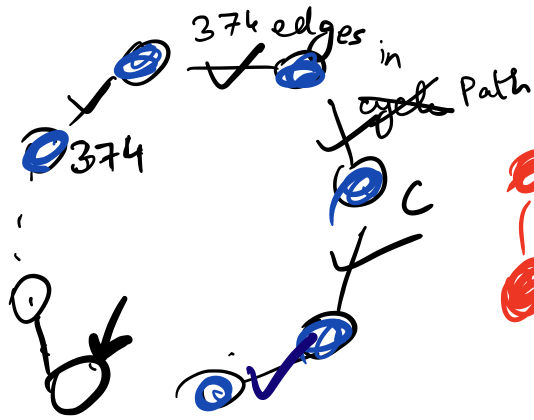
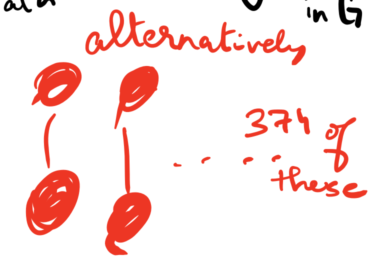
NP-HARDNESS REVIEW, UNDECIDABILITY.

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.









✓ S is independent set in $G \implies S \cup C$ 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'

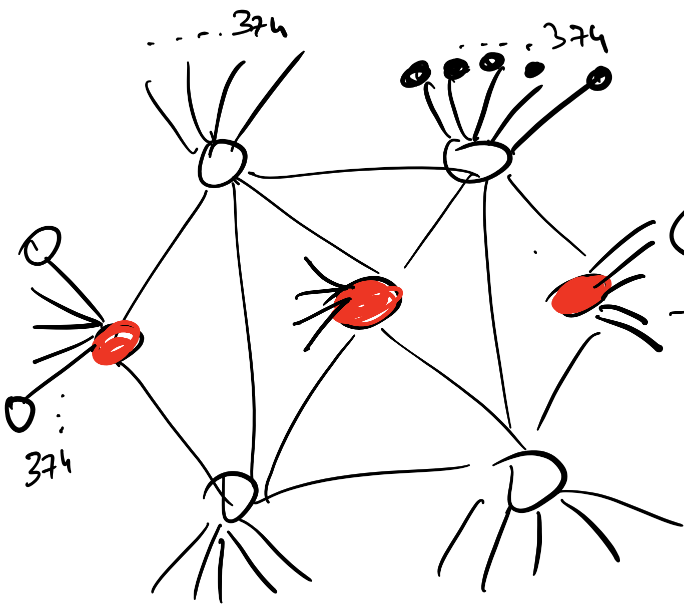
Suppose S' does not contain a vertex in C

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

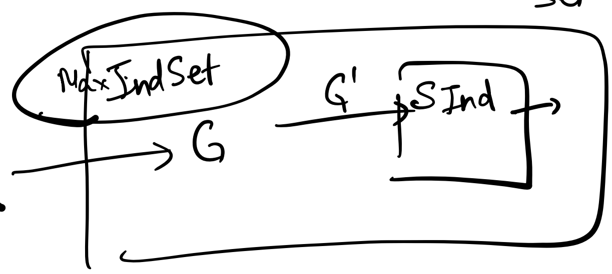
The resulting set S'' is still an almost-independent set, where $|S''| = |S'|$

$\therefore$ There is at least 1 largest almost-independent set in G' that contains all of C. Removing $C \Rightarrow$ 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.



smells like independent set.



Set of all dummy vertices = L

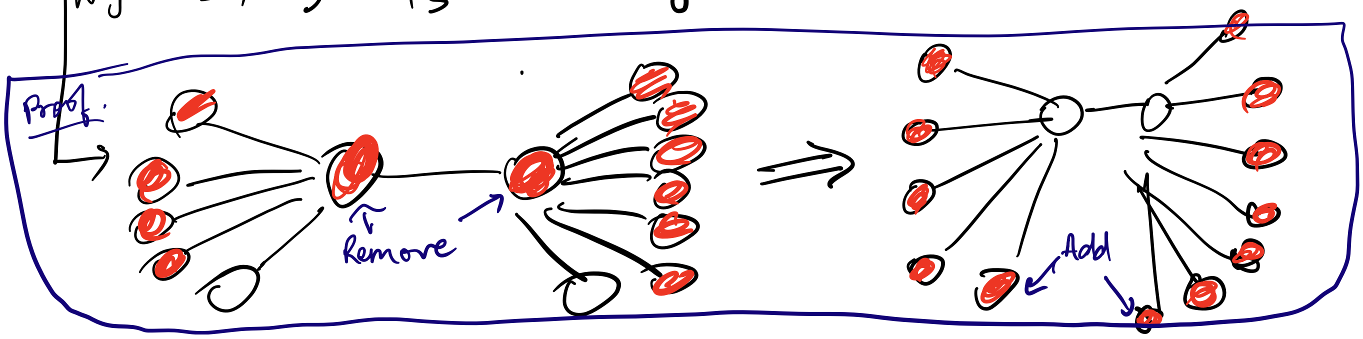
If S is ind set in G
$\Rightarrow S \cup L$ is sort of ind set in G'.

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

Call it $S' \cup L$

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

Proof:



NP-hard : no fast algorithm

Undecidable : no algorithm.

HALTING PROBLEM.

Problem: Given M represented as $\langle M \rangle$.

Turing machine

Executable program

String encoding / description of TM

Source code

Language = collection of strings.

Turing Machine Y decides language L if

$\forall\ x \in L,\ Y(x) = 1$

$\forall\ x \notin L,\ Y(x) = 0.$

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

Language of satisfiable Boolean formulae is decidable.

$$\boxed{L_{SELFHALT} = \{ \langle M \rangle \mid M \text{ on input } \langle M \rangle \text{ halts} \}}$$

executable program

source code

stops running and outputs 0/1.

$\boxed{\text{H. P. is undecidable}}$

There is no TM that given x outputs 1 when $x \in L_{SELFHALT}$, 0 otherwise.

$$L_{SELFNOTHALT} = \{ \langle M \rangle \mid M \text{ on input } \langle M \rangle \text{ does not halt} \}$$

CLAIM: $L_{SELFNOTHALT}$ is undecidable.

Proof: Suppose that L_{SNH} is decidable.

This M_1 exists.

$\langle x \rangle$ → If x doesn't halt on input $\langle x \rangle$ → YES

x does halt on $\langle x \rangle$ → NO

M_1 exists $\Rightarrow$ M_2 exists

$\langle x \rangle$ → If x doesn't halt on $\langle x \rangle$ → YES

x does halt on $\langle x \rangle$ → loop forever / hang

$M_2(\langle \cancel{x} \rangle)$ [M_2] ~~says YES~~ halts $\iff$ $\cancel{x}$ [M_2] doesn't halt on input $\langle \cancel{x} \rangle$ [M_2]

$M_2(\langle \cancel{x} \rangle)$ [M_2] ~~loops forever~~ does not halt $\iff$ $\cancel{x}$ [M_2] halts on input $\langle \cancel{x} \rangle$ [M_2]

What if $x = M_2$?

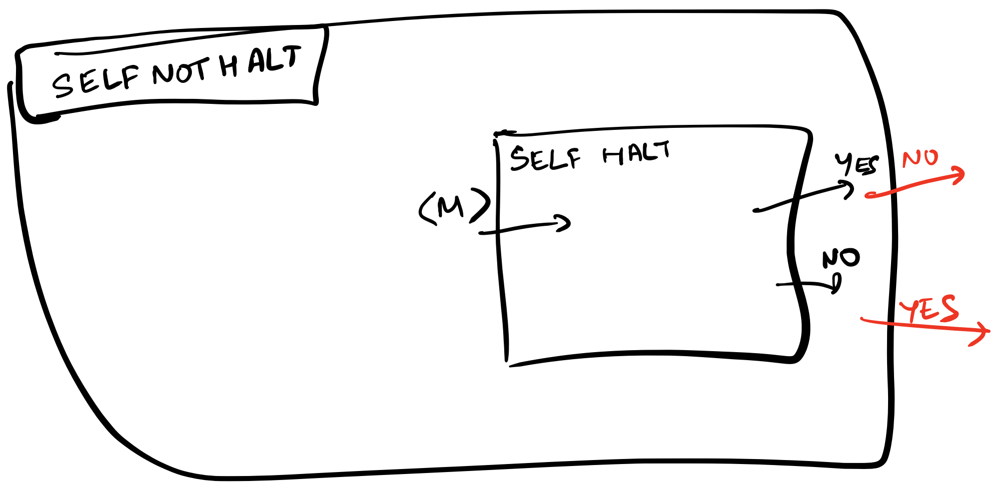
$M_2 \langle M_2 \rangle$ halts $\iff$ $M_2 \langle M_2 \rangle$ doesn't halt.

OOPS! Contradiction.

Claim. [If L_{SNH} is undecidable,] - This is true
then L_{SH} is also undecidable.

Suppose self halt were decidable.

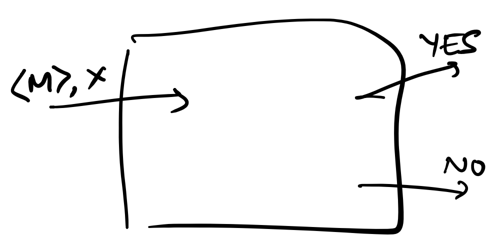
Then

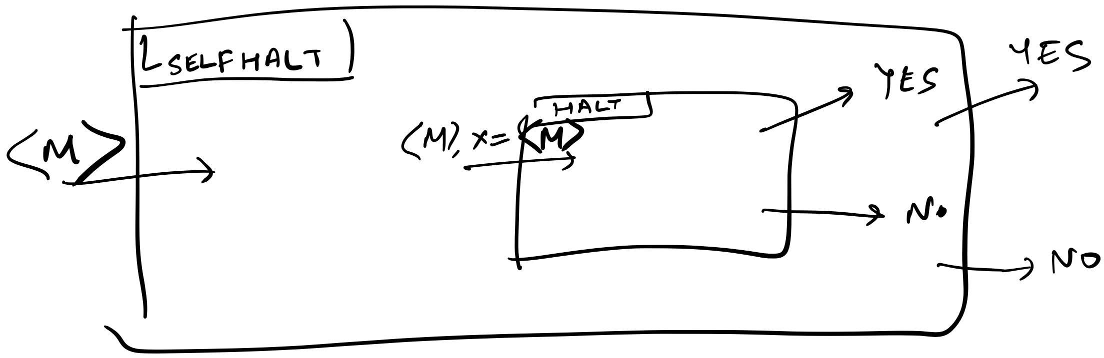


Self not halt is also decidable.
But we know SNH is undecidable.
$\Rightarrow$ SH is also undecidable.

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

Is L_{HALT} decidable?
Suppose it is.





$\Rightarrow$ $L_{SELFHALT}$ is decidable.

$\not\Leftrightarrow$

$\therefore$ HALT is not decidable.