Goal: Show that there is a language that is Turing-enumerable but not Turing-decidable.

Def's from 10 days ago:

Def: A language \( L \) is Turing-decidable if there is a TM \( M \) so that \( M \) halts on any input and halts in a final state if and only if the input is in \( L \).

Def: A language \( L \) is Turing-acceptable if there is a TM \( M \) so that \( M \) eventually halts in a final state if and only if the input is in \( L \) (but it might halt if the input is not in \( L \)).

Def: A language \( L \) is Turing-enumerable if there is a TM \( M \) that prints out every string in \( L \) (if \( L \) is infinite, \( M \) runs forever, but every string is printed out at some point).
10 days ago we proved that T-acceptable lang are T-enumerable and vice versa.

How do we find a language that is not TD but is TE/TA?

We will find a language L so that L is TE but \( \overline{L} \) (everything not in L) is not TE.

Then L is not TD.

Thm: If L and \( \overline{L} \) are is TD if and only if both L and \( \overline{L} \) are TA.

Pf: If L is TD, then \( \overline{L} \) is TD by switching final and non-final states.
So both are TA. (since any lang that is TD is TA).

Now suppose \( L \) and \( \overline{L} \) are both TA. This means we have TMs \( M \) and \( \overline{M} \) that accept \( L \cup \overline{L} \) — in other words, \( M \) halts in a final state for \( L \), and \( \overline{M} \) halts in a final state for \( \overline{L} \).

To build a TM \( \hat{M} \) that always halts and does so in a final state for \( L \), we have \( \hat{M} \) alternately do one step of \( M \) and one step of \( \overline{M} \), halting when either one of them halts and outputting the appropriate answer.
Finding a language \( L \) that is TA but \( E \) is not TA.

We'll think only about TMs whose input alphabet is \( \{1, 3\} \) and think of inputs as numbers in unary.

When we talked about universal TMs, we came up with a way of representing a TM as a string of 0's and 1's.

\[
(q_1, l_3) \rightarrow (q_3, l_5, R) \\
\downarrow
\]

\[
10111011101111011
\]

and separate transitions with another 0.

Once we fix a scheme for repr. TMs, we can talk about the first TM, the second TM, by their numeric order thought of as binary #s.
\[ L = \{ 1^n \mid 1^n \text{ is accepted by the } n\text{-th TM} \} \]

Fill this table with N's and Y's for whether that TM accepts that input (as a unary number).

Look at diagonal:

A number is in \( L \) if that entry in the diagonal is a "Y".
$L$ is TA - to check if $1^n \in L$,
we 1) figure out what the $n$-th TM is
2) running (as on universal TM) the $n$-th TM on the input $1^n$.
3) give whatever answer the $n$-th TM gives (or run forever if the $n$-th TM runs forever on that input).

$L$ is not TA.
Suppose for contradiction that $\overline{L}$ is TA. Then is some TM that accepts $\overline{L}$.
This TM is on my list of TMs.

So this TM is the $n$-th TM for some $n$. Is the $n$-th TM supposed to accept $1^n$ or not?
If the $n$-th TM accepts $1^n$, then $1^n \in \mathcal{L}$, so $1^n \notin \mathcal{L}$, so the $n$-th TM (which is supposed to accept $\mathcal{L}$) should reject $1^n$.

If the $n$-th TM $\tau_n$ doesn't accept $1^n$, then $1^n \notin \mathcal{L}$, so $1^n \notin \mathcal{L}$, so the $n$-th TM should accept $1^n$.

So whichever TM we purported accepted $\mathcal{L}$ must make a mistake at some point—when it encounters the input that corresponds to itself in the ordering of TMs.

So there is no such machine.

So $\mathcal{L}$ is not TA, and so $\mathcal{L}$ is not TD.