<u>Goal:</u> Show that there is a language that is Turing-enumerable but not Turing-decidable.

<u>Def'ns from 10 days ago:</u>

<u>Def:</u> 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.

<u>Def:</u> A language L is Turing-acceptable if there is a TM M so that M ~~eventu~~ 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).

<u>Def:</u> A language L is Turing-enumerable if there is a TM M that prints out every string in L (if L is infinite, ~~this~~ M runs forever, but every string is printed out at some point)

10 days ago- we proved that ~~lang~~ T-acceptable lang are T-enumerable and vice versa.

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

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

Then L is not TD.

~~Lem~~

<u>Thm:</u> ~~If~~ L ~~and $\overline{L}$ are~~ is TD if and only if both L and $\overline{L}$ are ~~TE~~ TA.

<u>Pf:</u> 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 $\bar{L}$ are both TA. This means we have TMs M and $\bar{M}$ that accept L & $\bar{L}$ - in other words, M halts in a final state for L, and $\bar{M}$ halts in a final state for $\bar{L}$.

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 $\bar{M}$, halting when either one of them halts and outputting the appropriate answer.

Finding a language L that is TA but $\overline{L}$ is not TA.

We'll think only about TMs whose input alphabet ~~has~~ is $\{1\}$, 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 & 1's.

$$(q_\phi, \ell_3) \longrightarrow (q_3, \ell_5, R)$$

$$\Downarrow$$

$$1011101110\ 1111101\ 1$$

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.

$1010101 01 \to (q_1, \ell_1) \to (q_1, \ell_1, L)$

$$L = \{ 1^n \mid 1^n \text{ is accepted by the } n\text{-th TM} \}$$

Input

TM	1	2	3	4	5			
1	N	N	N	N	N	...		
2	N	N	N	N	N	...		
3								
4								
5	N	Y	Y	N	Y	N	Y	...
6								
⋮								
	Y	N	Y	N	N	Y	N	...

fill this table w/ N's & 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).

$\overline{L}$ 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 $1^n \in L$, so $1^n \notin \overline{L}$, so the n-th TM (which is supposed to accept $\overline{L}$) should reject 1^n.

If the n-th TM ~~rej~~ doesn't accept 1^n, then $1^n \notin L$, so $1^n \in \overline{L}$, so the n-th TM should accept 1^n.

So whichever TM we purported accepted $\overline{L}$ 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 $\overline{L}$ is not TA, and so L is not TD.