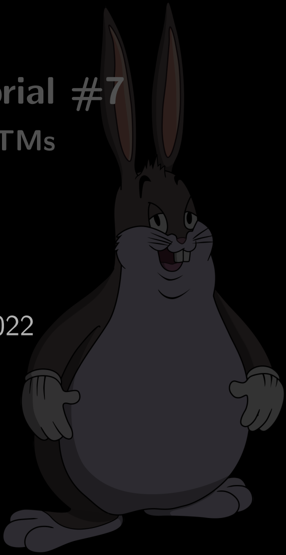
CSC363 Tutorial #7

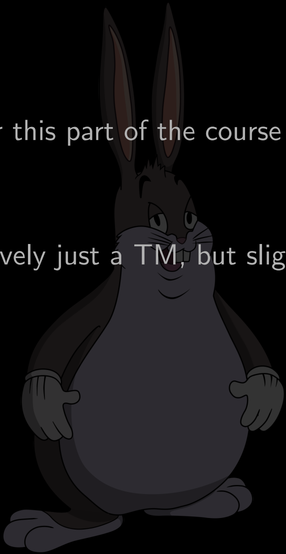
Alternative TMs



March 9, 2022

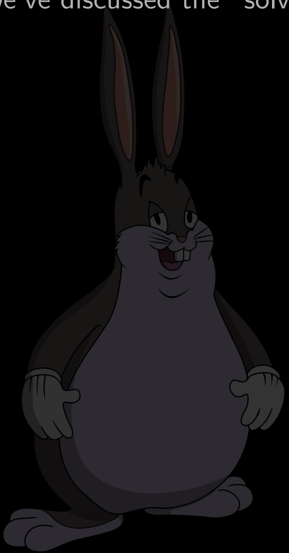
Learning objectives this tutorial

- Define some aliases we'll be using for this part of the course (complexity).
- Describe a "multi-tape" TM.
- Show that a multi-tape TM is effectively just a TM, but slightly better.



Some aliases

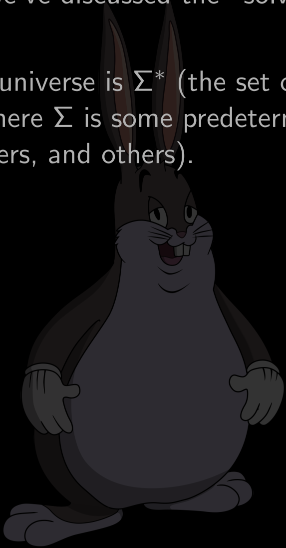
In the *computability* part of the course, we've discussed the "solvability" of *sets* (all subsets of our universe $\mathbb{N}$).



Some aliases

In the *computability* part of the course, we've discussed the "solvability" of *sets* (all subsets of our universe $\mathbb{N}$).

In the *complexity* part of the course, our universe is Σ^* (the set of strings using characters from Σ) instead of $\mathbb{N}$, where Σ is some predetermined alphabet (binary, decimals, ASCII characters, and others).

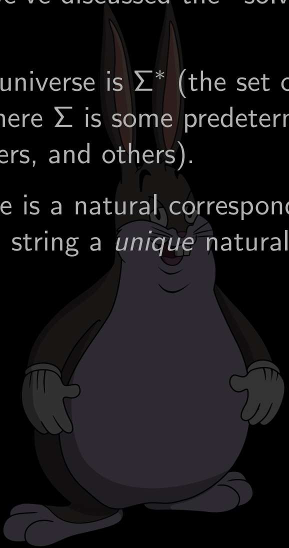


Some aliases

In the *computability* part of the course, we've discussed the "solvability" of *sets* (all subsets of our universe $\mathbb{N}$).

In the *complexity* part of the course, our universe is Σ^* (the set of strings using characters from Σ) instead of $\mathbb{N}$, where Σ is some predetermined alphabet (binary, decimals, ASCII characters, and others).

Task: Let Σ be any finite alphabet. There is a natural correspondence between Σ^* and $\mathbb{N}$, as we can assign each string a *unique* natural number. Recall what we used to show this.



Some aliases

In the *computability* part of the course, we've discussed the "solvability" of *sets* (all subsets of our universe $\mathbb{N}$).

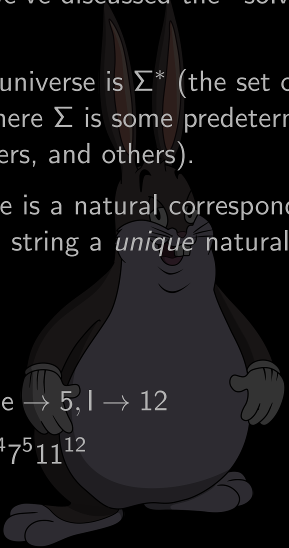
In the *complexity* part of the course, our universe is Σ^* (the set of strings using characters from Σ) instead of $\mathbb{N}$, where Σ is some predetermined alphabet (binary, decimals, ASCII characters, and others).

Task: Let Σ be any finite alphabet. There is a natural correspondence between Σ^* and $\mathbb{N}$, as we can assign each string a *unique* natural number. Recall what we used to show this.

Ans: Gödel Numbers!

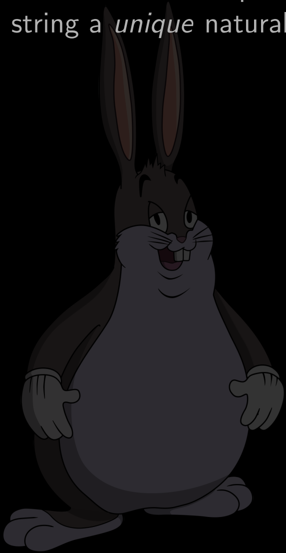
$$\mathrm{g} \to 7, \mathrm{o} \to 15, \mathrm{d} \to 4, \mathrm{e} \to 5, \mathrm{l} \to 12$$

$$\mathrm{godel} \to 2^7 3^{15} 5^4 7^5 11^{12}$$



Some aliases

Task: Let Σ be any finite alphabet. There is a natural correspondence between Σ^* and $\mathbb{N}$, as we can assign each string a *unique* natural number. Recall what we used to show this.



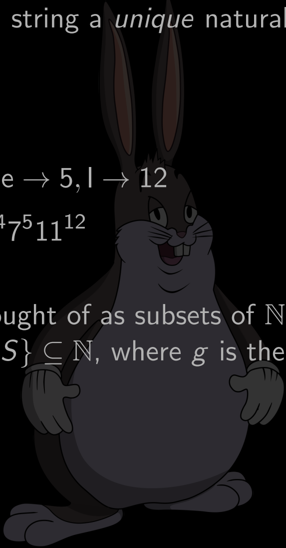
Some aliases

Task: Let Σ be any finite alphabet. There is a natural correspondence between Σ^* and $\mathbb{N}$, as we can assign each string a *unique* natural number. Recall what we used to show this.

Ans: Gödel Numbers!

$$\mathrm{g} \to 7, \mathrm{o} \to 15, \mathrm{d} \to 4, \mathrm{e} \to 5, \mathrm{l} \to 12$$

$$\mathrm{godel} \to 2^7 3^{15} 5^4 7^5 11^{12}$$



So in this sense, subsets of Σ^* can be thought of as subsets of $\mathbb{N}$ by mapping $S \subseteq \Sigma^*$ to $g(S) = \{g(w) : w \in S\} \subseteq \mathbb{N}$, where g is the Gödel mapping function.

Some aliases

Task: Let Σ be any finite alphabet. There is a natural correspondence between Σ^* and $\mathbb{N}$, as we can assign each string a *unique* natural number. Recall what we used to show this.

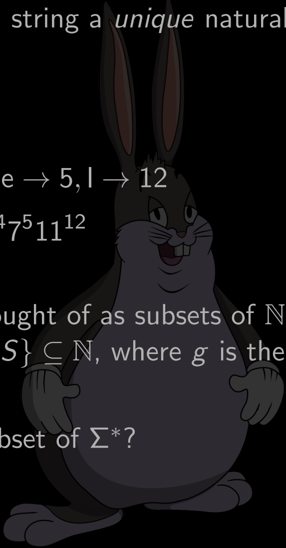
Ans: Gödel Numbers!

$$\mathsf{g} \to 7, \mathsf{o} \to 15, \mathsf{d} \to 4, \mathsf{e} \to 5, \mathsf{l} \to 12$$

$$\mathsf{godel} \to 2^7 3^{15} 5^4 7^5 11^{12}$$

So in this sense, subsets of Σ^* can be thought of as subsets of $\mathbb{N}$ by mapping $S \subseteq \Sigma^*$ to $g(S) = \{g(w) : w \in S\} \subseteq \mathbb{N}$, where g is the Gödel mapping function.

Question: What is another term for a subset of Σ^*?



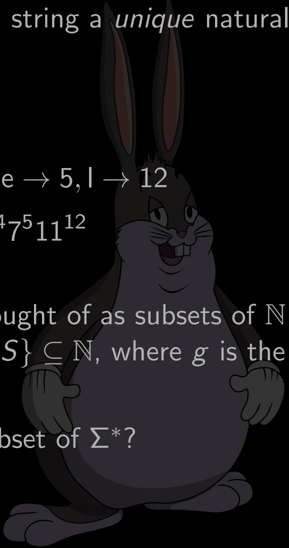
Some aliases

Task: Let Σ be any finite alphabet. There is a natural correspondence between Σ^* and $\mathbb{N}$, as we can assign each string a *unique* natural number. Recall what we used to show this.

Ans: Gödel Numbers!

$$\text{g} \to 7, \text{o} \to 15, \text{d} \to 4, \text{e} \to 5, \text{l} \to 12$$

$$\text{godel} \to 2^7 3^{15} 5^4 7^5 11^{12}$$



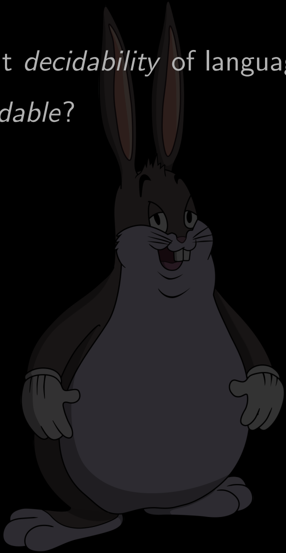
So in this sense, subsets of Σ^* can be thought of as subsets of $\mathbb{N}$ by mapping $S \subseteq \Sigma^*$ to $g(S) = \{g(w) : w \in S\} \subseteq \mathbb{N}$, where g is the Gödel mapping function.

Question: What is another term for a subset of Σ^*?
Ans: *Language*.

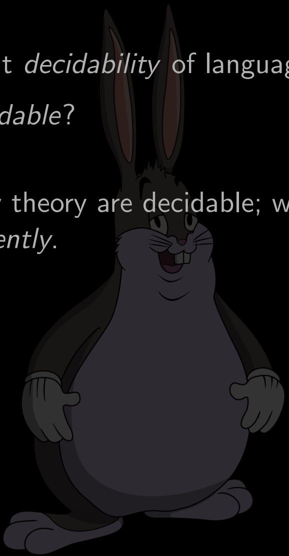
Some aliases

In complexity, we will consider the efficient *decidability* of languages.

Question: What is another word for *decidable*?



Some aliases



In complexity, we will consider the efficient *decidability* of languages.

Question: What is another word for *decidable*?
Ans: *Computable*.

Most languages we consider in complexity theory are decidable; we examine whether they are decidable *efficiently*.

Some aliases

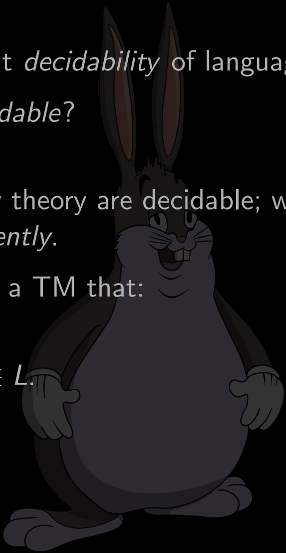
In complexity, we will consider the efficient *decidability* of languages.

Question: What is another word for *decidable*?
Ans: *Computable*.

Most languages we consider in complexity theory are decidable; we examine whether they are decidable *efficiently*.

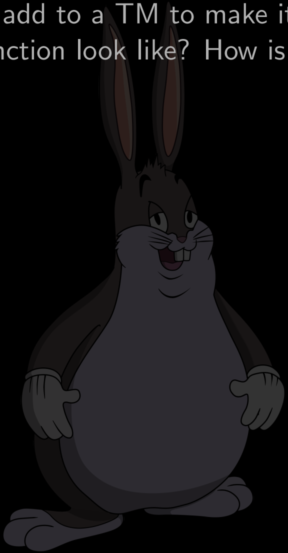
Definition: A *decider* for a language L is a TM that:

1. always halts *on any input*,
2. accepts the input x if and only if $x \in L$.



Multi-tape TM

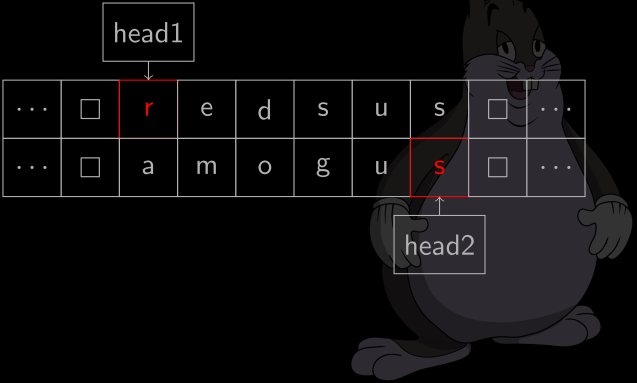
Task: Describe some attributes we could add to a TM to make it "multi-tape". What will the transition function look like? How is input handled?



Multi-tape TM

Task: Describe some attributes we could add to a TM to make it "multi-tape". What will the transition function look like? How is input handled?

Here's what I have in mind:

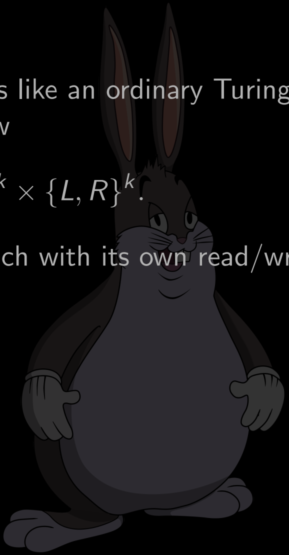


Multi-tape TM

Definition: A *k*-**tape Turing Machine** is like an ordinary Turing Machine, but its transition function is now

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k.$$

In effect, this gives us *k* distinct tapes, each with its own read/write head. We read and write *k* symbols at once.



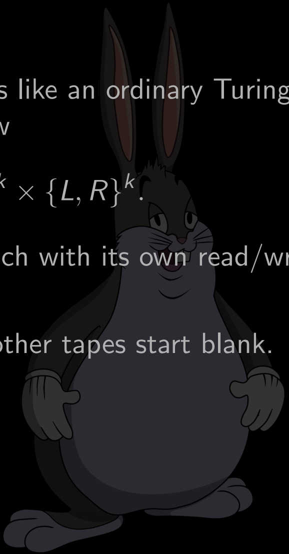
Multi-tape TM

Definition: A *k*-**tape Turing Machine** is like an ordinary Turing Machine, but its transition function is now

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k.$$

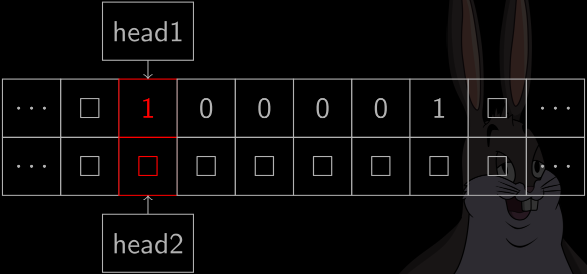
In effect, this gives us k distinct tapes, each with its own read/write head. We read and write k symbols at once.

The input is placed on the first tape; all other tapes start blank.



Multi-tape TM Example

Let's construct a multi-tape TM over the alphabet $\{0, 1\}$ that accepts palindromes. Here's what we will do:



1. Copy the string on tape 1 to tape 2.
2. Move head1 to the beginning of the first tape.
3. Compare characters from head1 and head2, scanning right and left respectively.

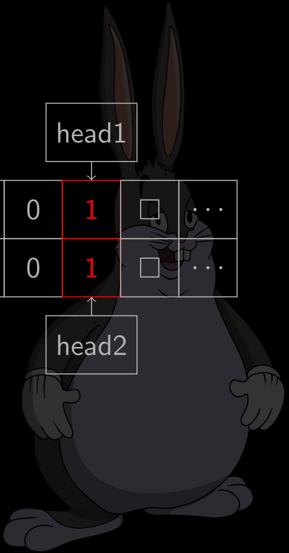
Multi-tape TM Example

1. Copy the string on tape 1 to tape 2.

head1

⋯	□	1	0	0	0	0	1	□	⋯
⋯	□	1	0	0	0	0	1	□	⋯

head2



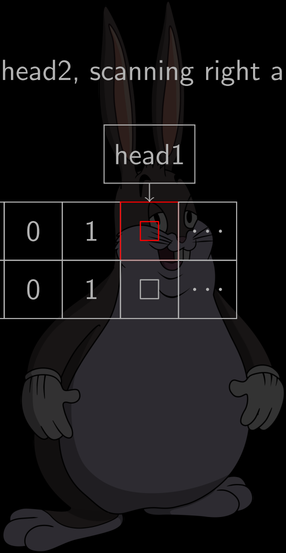
Multi-tape TM Example

3. Compare characters from head1 and head2, scanning right and left respectively.

head1

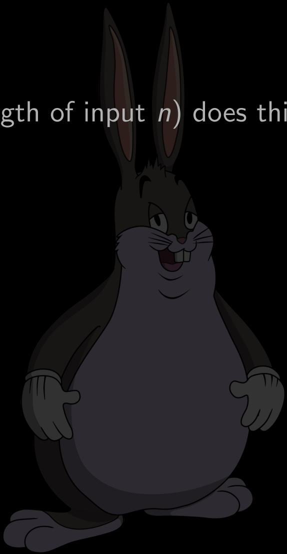
⋯	□	1	0	0	0	0	1	□	⋯
⋯	□	1	0	0	0	0	1	□	⋯

head2



Multi-tape TM Example

Question: What runtime (in terms of length of input n) does this 2-tape TM have?



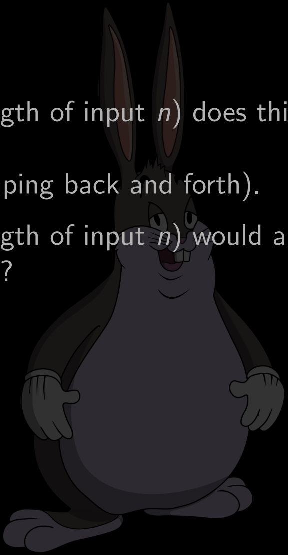
Multi-tape TM Example

Question: What runtime (in terms of length of input n) does this 2-tape TM have?
Ans: $O(n)$ (because we have to keep jumping back and forth).

Multi-tape TM Example

Question: What runtime (in terms of length of input n) does this 2-tape TM have?
Ans: $O(n)$ (because we have to keep jumping back and forth).

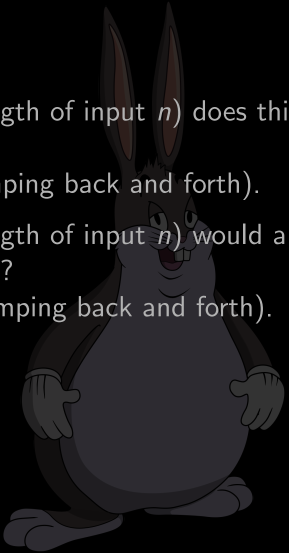
Question: What runtime (in terms of length of input n) would a "naive" single-tape TM use to detect palindromes?



Multi-tape TM Example

Question: What runtime (in terms of length of input n) does this 2-tape TM have?
Ans: $O(n)$ (because we have to keep jumping back and forth).

Question: What runtime (in terms of length of input n) would a "naive" single-tape TM use to detect palindromes?
Ans: $O(n^2)$ (because we have to keep jumping back and forth).



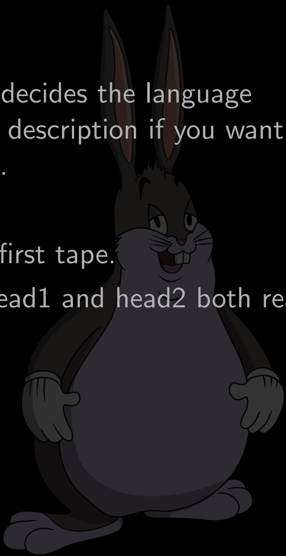
Multi-tape TM Example

Task: Construct a $O(n)$ 2-tape TM that decides the language $\{0^n 1^n : k \in \mathbb{N}\}$. You may use a high-level description if you want.

Multi-tape TM Example

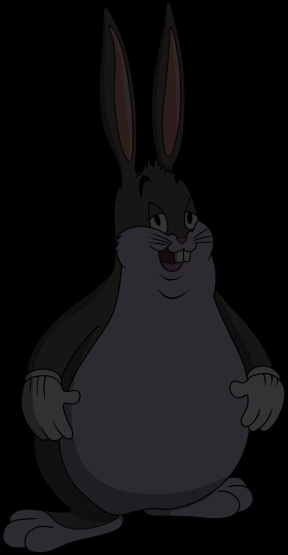
Task: Construct a $O(n)$ 2-tape TM that decides the language $\{0^n 1^n : k \in \mathbb{N}\}$. You may use a high-level description if you want.
Ans: Here's the procedure I have in mind.

1. Copy the string on tape 1 to tape 2.
2. Move head1 to the beginning of the first tape.
3. Compare character by character; if head1 and head2 both read 0 or both read 1, then reject.



Multi-tape TM Example

Question: How fast can we decide $\{0^n 1^n : k \in \mathbb{N}\}$ with a TM?



Multi-tape TM Example

Question: How fast can we decide $\{0^n1^n : k \in \mathbb{N}\}$ with a TM?
Ans: Naively, $O(n^2)$. The procedure is as follows:

1. Cross out a 0; move to the right end of the string.
2. Cross out a 1; move to the left end of the string.

Try `https://turingmachinesimulator.com/shared/prsswhkkyb`.

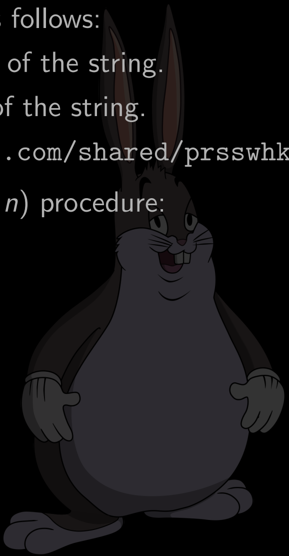
Multi-tape TM Example

Question: How fast can we decide $\{0^n 1^n : k \in \mathbb{N}\}$ with a TM?
Ans: Naively, $O(n^2)$. The procedure is as follows:

1. Cross out a 0; move to the right end of the string.
2. Cross out a 1; move to the left end of the string.

Try `https://turingmachinesimulator.com/shared/prsswhkkyb`.

But we can do better! There is a $O(n \log n)$ procedure:



Multi-tape TM Example

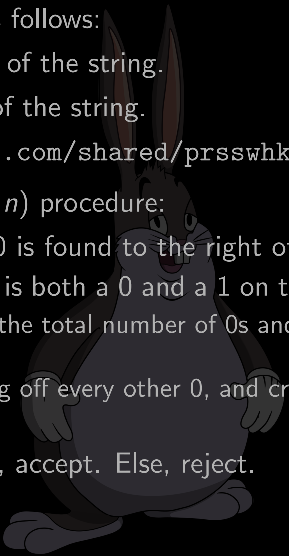
Question: How fast can we decide $\{0^n 1^n : k \in \mathbb{N}\}$ with a TM?
Ans: Naively, $O(n^2)$. The procedure is as follows:

1. Cross out a 0; move to the right end of the string.
2. Cross out a 1; move to the left end of the string.

Try `https://turingmachinesimulator.com/shared/prsswhkkyb`.

But we can do better! There is a $O(n \log n)$ procedure:

1. Scan across the tape and reject if a 0 is found to the right of a 1.
2. Repeat the following as long as there is both a 0 and a 1 on the tape:
   2.1 Scan across the tape, and reject if the total number of 0s and 1s remaining is odd.
   2.2 Scan again across the tape, crossing off every other 0, and crossing off every other 1.
3. If the tape doesn't have any 0s or 1s, accept. Else, reject.



Multi-tape TM Example

Question: How fast can we decide $\{0^n 1^n : k \in \mathbb{N}\}$ with a TM?
Ans: Naively, $O(n^2)$. The procedure is as follows:

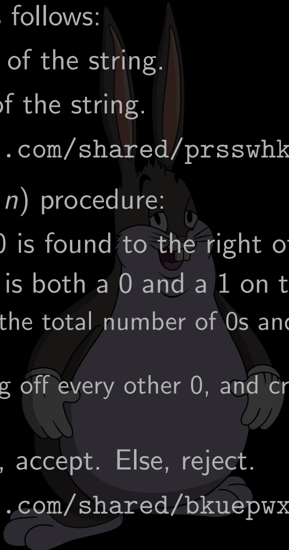
1. Cross out a 0; move to the right end of the string.
2. Cross out a 1; move to the left end of the string.

Try `https://turingmachinesimulator.com/shared/prsswhkkyb`.

But we can do better! There is a $O(n \log n)$ procedure:

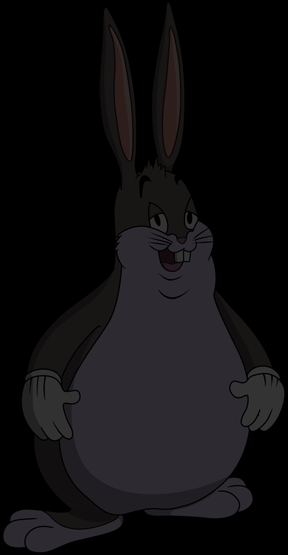
1. Scan across the tape and reject if a 0 is found to the right of a 1.
2. Repeat the following as long as there is both a 0 and a 1 on the tape:
   2.1 Scan across the tape, and reject if the total number of 0s and 1s remaining is odd.
   2.2 Scan again across the tape, crossing off every other 0, and crossing off every other 1.
3. If the tape doesn't have any 0s or 1s, accept. Else, reject.

Try `https://turingmachinesimulator.com/shared/bkuepwxgdh`.



Multi-tape TMs in P

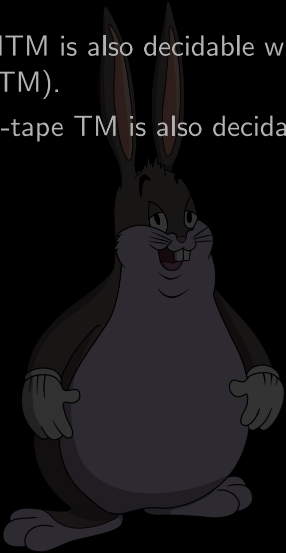
It turns out, just as with NTMs, the question of *decidability* doesn't change with multi-tape TMs.



Multi-tape TMs in P

It turns out, just as with NTMs, the question of *decidability* doesn't change with multi-tape TMs.

- Everything that is decidable with a NTM is also decidable with a TM (since we can simulate a NTM on a TM).
- Everything that is decidable with a k-tape TM is also decidable with a TM.



Multi-tape TMs in P

It turns out, just as with NTMs, the question of *decidability* doesn't change with multi-tape TMs.

- Everything that is decidable with a NTM is also decidable with a TM (since we can simulate a NTM on a TM).
- Everything that is decidable with a k-tape TM is also decidable with a TM.

In fact, there is an even stronger theorem.

Theorem: Everything that is decidable with a k-tape TM in $O(f(n))$ time is decidable with a TM in $O((f(n))^2)$ time. (See Sipser page 137)

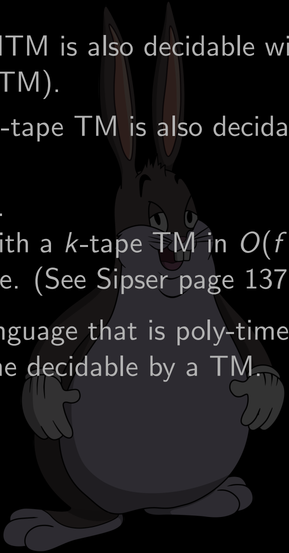
Multi-tape TMs in P

It turns out, just as with NTMs, the question of *decidability* doesn't change with multi-tape TMs.

- Everything that is decidable with a NTM is also decidable with a TM (since we can simulate a NTM on a TM).
- Everything that is decidable with a k-tape TM is also decidable with a TM.

In fact, there is an even stronger theorem.
Theorem: Everything that is decidable with a k-tape TM in $O(f(n))$ time is decidable with a TM in $O((f(n))^2)$ time. (See Sipser page 137)

Task: Using the above, show that any language that is poly-time decidable by a k-tape TM is also poly-time decidable by a TM.



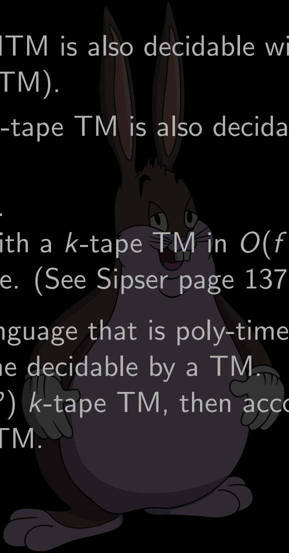
Multi-tape TMs in P

It turns out, just as with NTMs, the question of *decidability* doesn't change with multi-tape TMs.

- Everything that is decidable with a NTM is also decidable with a TM (since we can simulate a NTM on a TM).
- Everything that is decidable with a k-tape TM is also decidable with a TM.

In fact, there is an even stronger theorem.
Theorem: Everything that is decidable with a k-tape TM in $O(f(n))$ time is decidable with a TM in $O((f(n))^2)$ time. (See Sipser page 137)

Task: Using the above, show that any language that is poly-time decidable by a k-tape TM is also poly-time decidable by a TM.
Ans: If a language is decidable by a $O(n^p)$ k-tape TM, then according to the theorem, it is decidable by a $O(n^{2p})$ TM.



Multi-tape TMs in P

It turns out, just as with NTMs, the question of *decidability* doesn't change with multi-tape TMs.

- Everything that is decidable with a NTM is also decidable with a TM (since we can simulate a NTM on a TM).
- Everything that is decidable with a k-tape TM is also decidable with a TM.

In fact, there is an even stronger theorem.
Theorem: Everything that is decidable with a k-tape TM in $O(f(n))$ time is decidable with a TM in $O((f(n))^2)$ time. (See Sipser page 137)

Task: Using the above, show that any language that is poly-time decidable by a k-tape TM is also poly-time decidable by a TM.
Ans: If a language is decidable by a $O(n^p)$ k-tape TM, then according to the theorem, it is decidable by a $O(n^{2p})$ TM.

In effect, this shows that multi-tape TMs are "better", but don't fundamentally change the set of poly-time decidable languages.