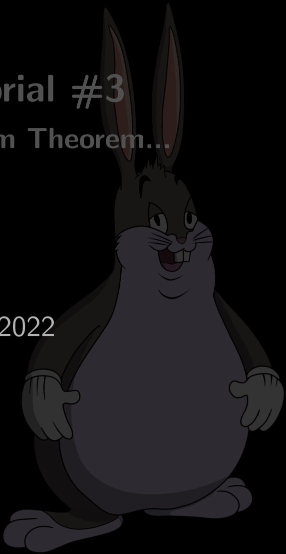
CSC363 Tutorial #3

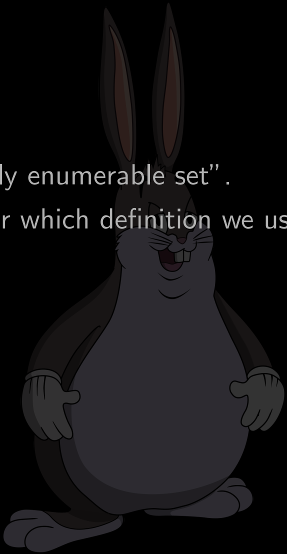
CE sets, Normal Form Theorem...

February 02, 2022



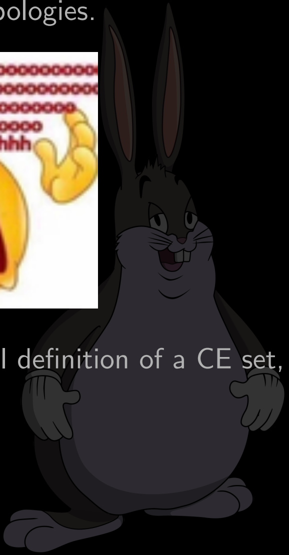
Learning objectives this tutorial

- Talk about the definition “computably enumerable set”.
- Conclude that it doesn’t really matter which definition we use!



Computably Enumerable Sets

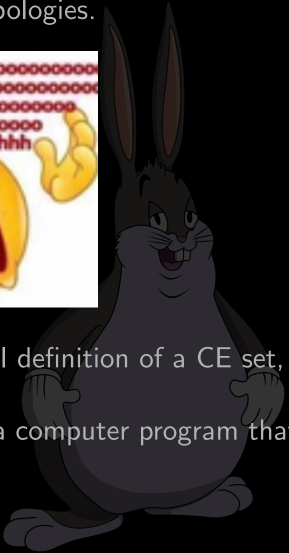
Assignment 1 recall time! My sincerest apologies.



Question: What was our original informal definition of a CE set, from the first assignment?

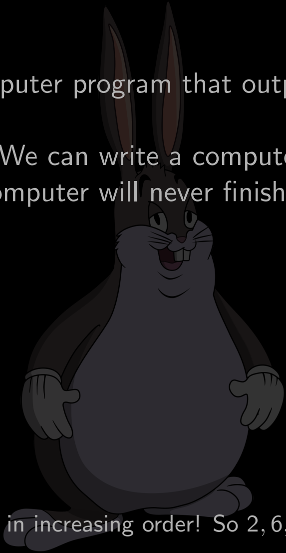
Computably Enumerable Sets

Assignment 1 recall time! My sincerest apologies.



Question: What was our original informal definition of a CE set, from the first assignment?
Ans: A set $M \subseteq \mathbb{N}$ is CE if we can write a computer program that outputs the elements of M in a list.

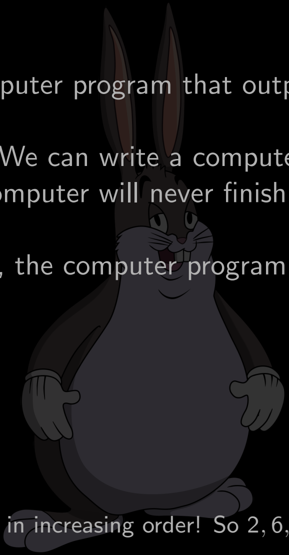
Computably Enumerable Sets



A set $M \subseteq \mathbb{N}$ is CE if we can write a computer program that outputs the elements of M in a list.
But how do we "output" an infinite set? We can write a computer program that prints $2, 4, 6, 8, \ldots$, but a computer will never finish outputting all the even numbers!

[1] It is not necessary that we print the numbers in increasing order! So $2, 6, 4, 8, \ldots$ is also a valid way to enumerate the evens.

Computably Enumerable Sets



A set $M \subseteq \mathbb{N}$ is CE if we can write a computer program that outputs the elements of M in a list.

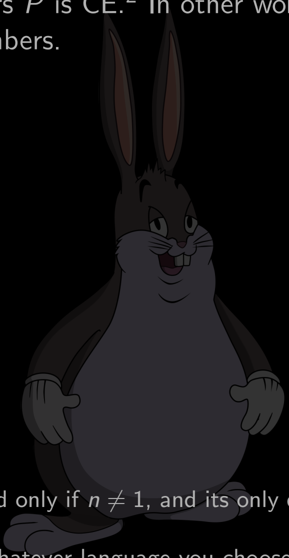
But how do we "output" an infinite set? We can write a computer program that prints $2, 4, 6, 8, \ldots$, but a computer will never finish outputting all the even numbers!

What we mean here is: given any $m \in M$, the computer program will eventually print out m.[1]

[1] It is not necessary that we print the numbers in increasing order! So $2, 6, 4, 8, \ldots$ is also a valid way to enumerate the evens.

Computably Enumerable Sets

Task: Show that the set of prime numbers P is CE.[2] In other words, write a program[3] that prints out the prime numbers.



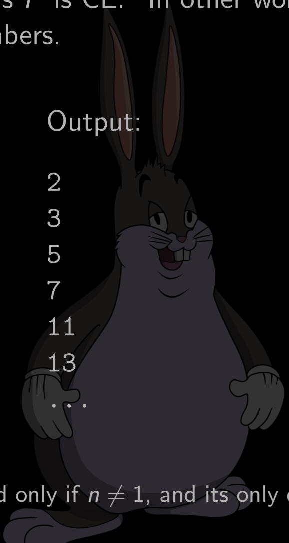
[2]Recall that a natural number n is prime if and only if $n \neq 1$, and its only divisors are 1 and n

[3]In Python, C, Minecraft, ChungusCode, or whatever language you choose!

Computably Enumerable Sets

Task: Show that the set of prime numbers P is CE.[2] In other words, write a program[3] that prints out the prime numbers.
Ans:

```
i = 2
while True:
  is_prime = True
  for j in range(i):
    if i % j == 0:
      is_prime = False
  if is_prime:
    print(i)
  i += 1
```

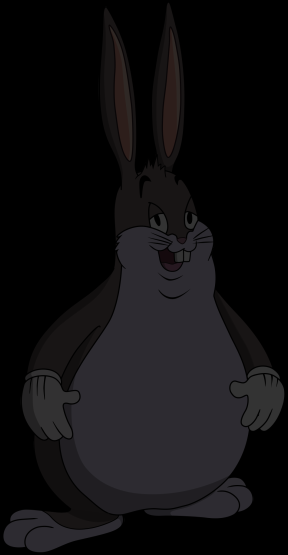


Output:

2
3
5
7
11
13
...

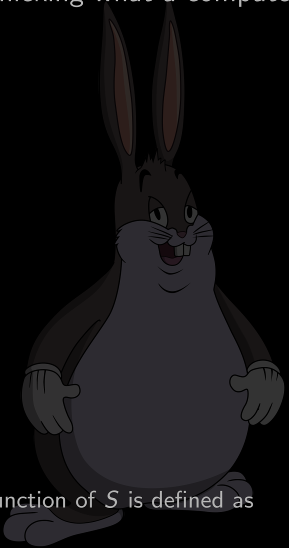
[2]Recall that a natural number n is prime if and only if $n \neq 1$, and its only divisors are 1 and n

[3]In Python, C, Minecraft, ChungusCode, or whatever language you choose!

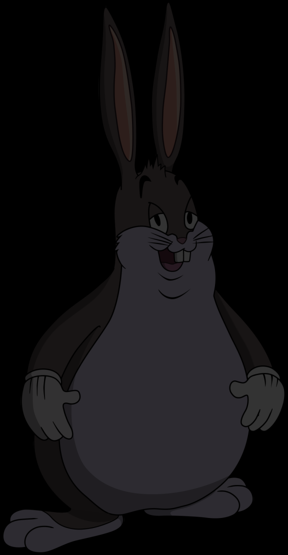


Formal definition of CE set

Recall in Lecture 3 that we built up a set of functions called the "partial recursive" functions, in an attempt to mimicking what a computer can do.



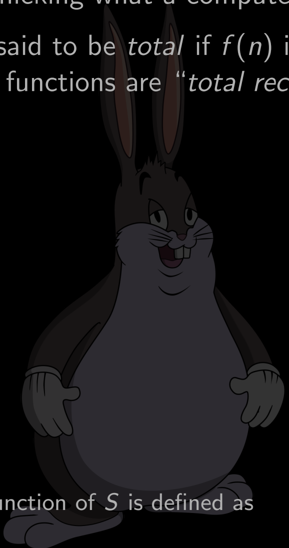
[4]Recall: If $S \subseteq \mathbb{N}$ is a set, the characteristic function of S is defined as
$$\chi_S(n) = \begin{cases} 1 & n \in \mathbb{N} \\ 0 & n \notin \mathbb{N}. \end{cases}$$



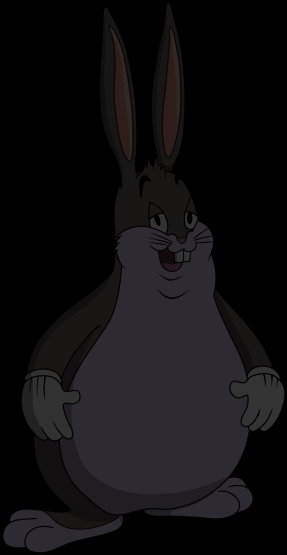
Formal definition of CE set

Recall in Lecture 3 that we built up a set of functions called the "partial recursive" functions, in an attempt to mimicking what a computer can do.

A partial recursive function $f : \mathbb{N} \to \mathbb{N}$ is said to be *total* if $f(n)$ is defined for all $n \in \mathbb{N}$. Some synonyms for "total" functions are "*total recursive*" and "**computable**".



[4]Recall: If $S \subseteq \mathbb{N}$ is a set, the characteristic function of S is defined as
$$\chi_S(n) = \begin{cases} 1 & n \in \mathbb{N} \\ 0 & n \notin \mathbb{N}. \end{cases}$$

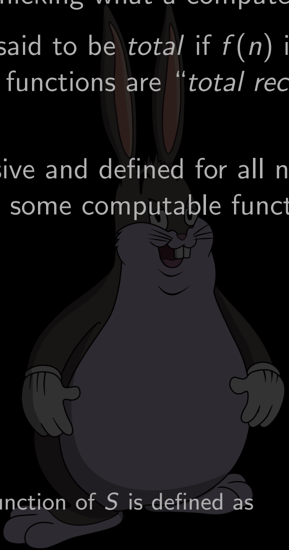


Formal definition of CE set

Recall in Lecture 3 that we built up a set of functions called the “partial recursive” functions, in an attempt to mimicking what a computer can do.

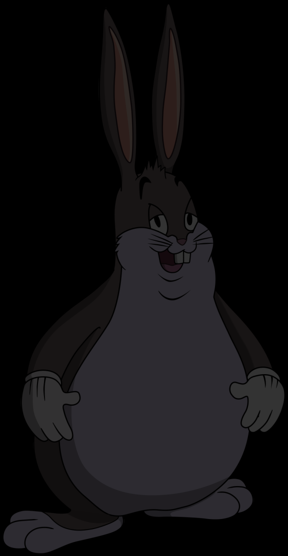
A partial recursive function $f : \mathbb{N} \to \mathbb{N}$ is said to be *total* if $f(n)$ is defined for all $n \in \mathbb{N}$. Some synonyms for “total” functions are “*total recursive*” and “**computable**”.

All primitive recursive functions are recursive and defined for all natural numbers, so they are all computable! But some computable functions are not primitive recursive.



[4] Recall: If $S \subseteq \mathbb{N}$ is a set, the characteristic function of S is defined as

$$\chi_S(n) = \begin{cases} 1 & n \in \mathbb{N} \\ 0 & n \notin \mathbb{N}. \end{cases}$$



Formal definition of CE set

Recall in Lecture 3 that we built up a set of functions called the "partial recursive" functions, in an attempt to mimicking what a computer can do.

A partial recursive function $f : \mathbb{N} \to \mathbb{N}$ is said to be *total* if $f(n)$ is defined for all $n \in \mathbb{N}$. Some synonyms for "total" functions are "*total recursive*" and "**computable**".

All primitive recursive functions are recursive and defined for all natural numbers, so they are all computable! But some computable functions are not primitive recursive.

Correction to last week's tutorial: Again, we lied to you!

- Last week's definition: A computable set is a set whose characterstic function[4] is primitive recursive.
- This week's definition: A computable set is a set whose characterstic function is computable (as we have just defined).

[4]Recall: If $S \subseteq \mathbb{N}$ is a set, the characteristic function of S is defined as
$$\chi_S(n) = \begin{cases} 1 & n \in \mathbb{N} \\ 0 & n \notin \mathbb{N}. \end{cases}$$

Formal definition of CE set

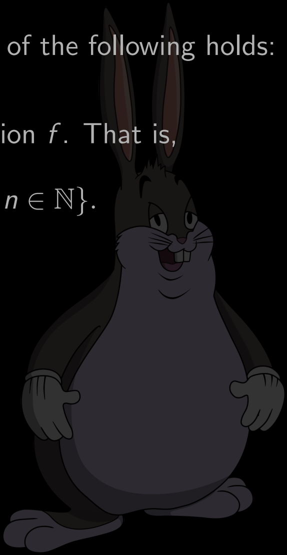
Now we will present the formal definition of a CE set (from Lecture 3 also).

Definition: A set $S \subseteq \mathbb{N}$ is **CE** when one of the following holds:

- $S = \emptyset$;
- S is the range of a computable function f. That is,

$$S = \{f(n) : n \in \mathbb{N}\}.$$

Write this down!!



Formal definition of CE set

Now we will present the formal definition of a CE set (from Lecture 3 also).

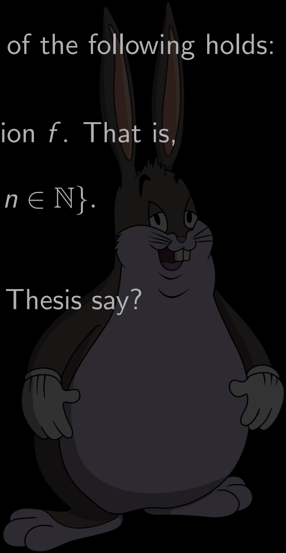
Definition: A set $S \subseteq \mathbb{N}$ is **CE** when one of the following holds:

- $S = \emptyset$;
- S is the range of a computable function f. That is,

$$S = \{f(n) : n \in \mathbb{N}\}.$$

Write this down!!

Question: What does the Church-Turing Thesis say?



Formal definition of CE set

Now we will present the formal definition of a CE set (from Lecture 3 also).

Definition: A set $S \subseteq \mathbb{N}$ is **CE** when one of the following holds:

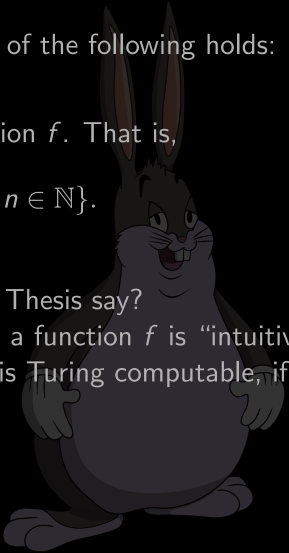
- $S = \emptyset$;
- S is the range of a computable function f. That is,

$$S = \{f(n) : n \in \mathbb{N}\}.$$

Write this down!!

Question: What does the Church-Turing Thesis say?

Ans: The Church-Turing Thesis says that a function f is "intuitively computable" iff it is total recursive (iff it is Turing computable, iff it is URM computable, etc).



Formal definition of CE set

Now we will present the formal definition of a CE set (from Lecture 3 also).

Definition: A set $S \subseteq \mathbb{N}$ is **CE** when one of the following holds:

- $S = \emptyset$;
- S is the range of a computable function f. That is,

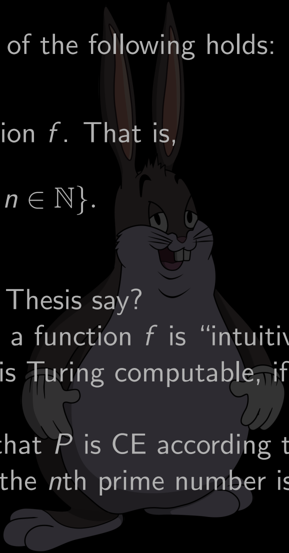
$$S = \{f(n) : n \in \mathbb{N}\}.$$

Write this down!!

Question: What does the Church-Turing Thesis say?

Ans: The Church-Turing Thesis says that a function f is "intuitively computable" iff it is total recursive (iff it is Turing computable, iff it is URM computable, etc).

Task: Let P be the set of primes. Show that P is CE according to the above definition, by showing that $f(n) =$ the nth prime number is computable using the CT Thesis.



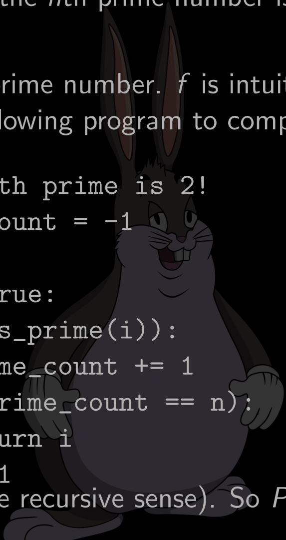
Formal definition of CE set

Task: Let P be the set of primes. Show that P is CE according to the above definition, by showing that $f(n) =$ the nth prime number is computable using the CT Thesis.

Ans: Define $f : \mathbb{N} \to \mathbb{N}$, $f(n) =$ the nth prime number. f is intuitively computable, because we can write the following program to compute f:

```
def is_prime(i):
  for j in range(i):
    if i % j == 0:
      return False
  return True
```

```
def f(n):
  # the 0th prime is 2!
  prime_count = -1
  i = 2
  while True:
    if (is_prime(i)):
      prime_count += 1
    if (prime_count == n):
      return i
    i += 1
```



By the CT Thesis, f is computable (in the recursive sense). So P, which is the range of f, is a CE set.

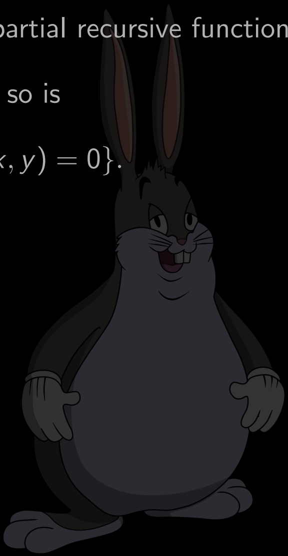
Equivalent definition 2

We will now prove the following:

$$S \text{ is CE} \Leftrightarrow S \text{ is the domain of a partial recursive function.}$$

Recall: if $g(x, y)$ is partial recursive, then so is

$$f(x) = \min\{y : g(x, y) = 0\}.$$



Equivalent definition 2

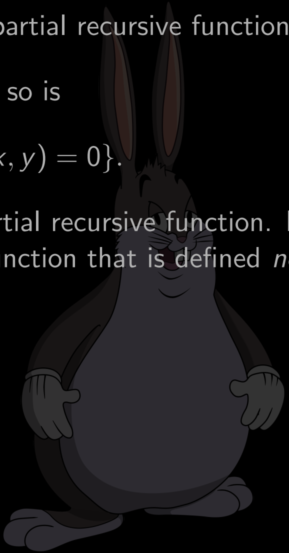
We will now prove the following:

$$S \text{ is CE} \Leftrightarrow S \text{ is the domain of a partial recursive function.}$$

Recall: if $g(x, y)$ is partial recursive, then so is

$$f(x) = \min\{y : g(x, y) = 0\}.$$

Task: Show that $\emptyset$ is the domain of a partial recursive function. In other words, come up with a partial recursive function that is defined *nowhere*!



Equivalent definition 2

We will now prove the following:

$$S \text{ is CE} \Leftrightarrow S \text{ is the domain of a partial recursive function.}$$

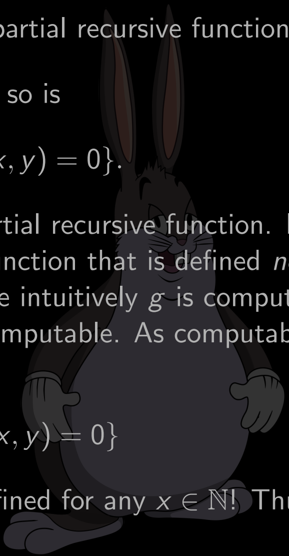
Recall: if $g(x, y)$ is partial recursive, then so is

$$f(x) = \min\{y : g(x, y) = 0\}.$$

Task: Show that $\emptyset$ is the domain of a partial recursive function. In other words, come up with a partial recursive function that is defined *nowhere*!
Ans: Define $g(x, y) = 1$ for all x, y. Since intuitively g is computable (just return 1 regardless of input), g is computable. As computable functions are (partial) recursive,

$$f(x) = \min\{y : g(x, y) = 0\}$$

is also partial recursive. But $f(x)$ is undefined for any $x \in \mathbb{N}$! Thus domain$(f) = \emptyset$.



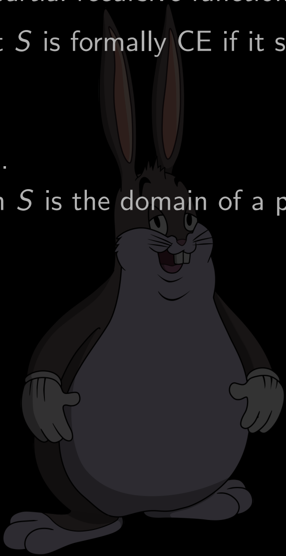
Equivalent definition 2

$$S \text{ is CE} \Rightarrow S \text{ is the domain of a partial recursive function.}$$

Let's prove the theorem! Recall that a set S is formally CE if it satisfied one of the following:

- $S = \emptyset$.
- $S = \text{range}(f)$ for some computable f.

Task: Show that if S is formally CE, then S is the domain of a partial recursive function.



Equivalent definition 2

$$S \text{ is CE} \Rightarrow S \text{ is the domain of a partial recursive function.}$$

Let's prove the theorem! Recall that a set S is formally CE if it satisfied one of the following:

- $S = \emptyset$.
- $S = \text{range}(f)$ for some computable f.

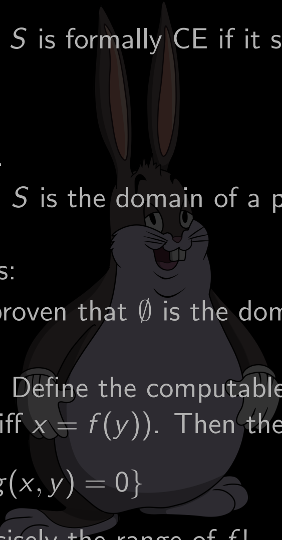
Task: Show that if S is formally CE, then S is the domain of a partial recursive function.

Ans: Suppose S is CE. We have two cases:

- $S = \emptyset$: On the previous slide, we've proven that $\emptyset$ is the domain of a partial recursive function.
- $S = \text{range}(f)$ where f is computable. Define the computable function $g(x, y) = |x - f(y)|$ (so $g(x, y) = 0$ iff $x = f(y)$). Then the function

  $$h(x) = \min\{x : g(x, y) = 0\}$$

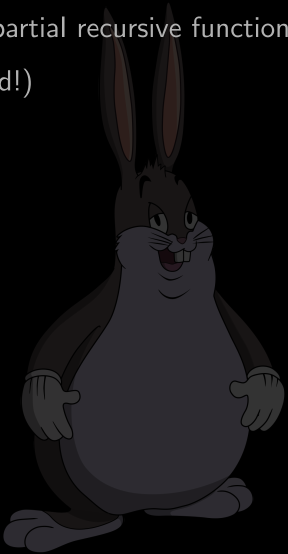
  is partial recursive. h's domain is precisely the range of f!



Equivalent definition 2

$$S \text{ is CE} \Leftrightarrow S \text{ is the domain of a partial recursive function.}$$

What about the other direction? (It's hard!)

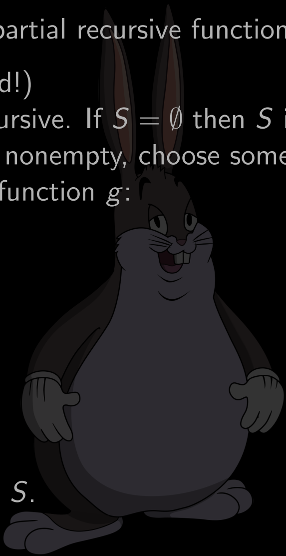


Equivalent definition 2

$$S \text{ is CE} \Leftrightarrow S \text{ is the domain of a partial recursive function.}$$

What about the other direction? (It's hard!)
Let $S = \text{domain}(f)$, where f is partial recursive. If $S = \emptyset$ then S is CE and we're done, so suppose $S \neq \emptyset$. Since S is nonempty, choose some $p \in S$. We may define the following computable function g:

```
def g(x, s):
  try to compute f(x) for s steps
  if f(x) returns within s steps:
    return x
  else:
    return p
```

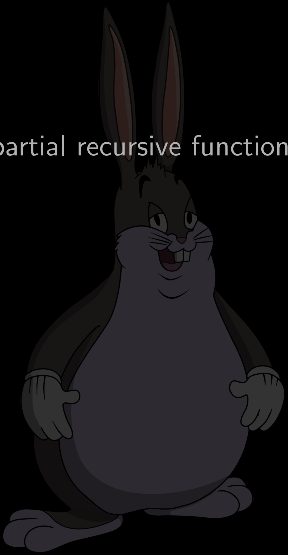


Task: Show that the range of g is indeed S.

Equivalent definition 2

So we've proven the following!

$$S \text{ is CE} \Leftrightarrow S \text{ is the domain of a partial recursive function.}$$



Equivalent definition 2

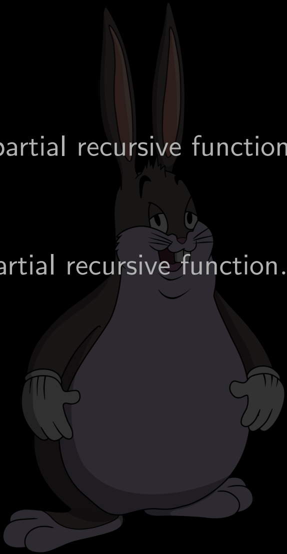
So we've proven the following!

$$S \text{ is CE} \Leftrightarrow S \text{ is the domain of a partial recursive function.}$$

It also turns out that

$$S \text{ is CE} \Leftrightarrow S \text{ is the range of a partial recursive function.}$$

But we don't have time to prove this! :(



Equivalent definition 2

So we've proven the following!

$$S \text{ is CE} \Leftrightarrow S \text{ is the domain of a partial recursive function.}$$

It also turns out that

$$S \text{ is CE} \Leftrightarrow S \text{ is the range of a partial recursive function.}$$

But we don't have time to prove this! :(
This equivalence of definitions is called the **Normal Form Theorem**.