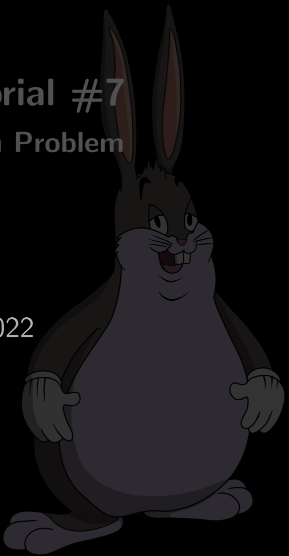
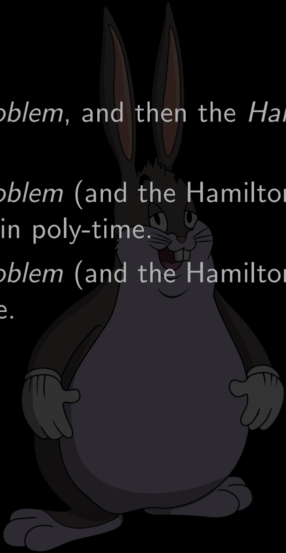
CSC363 Tutorial #7

Hamiltonian Path Problem

March 9, 2022



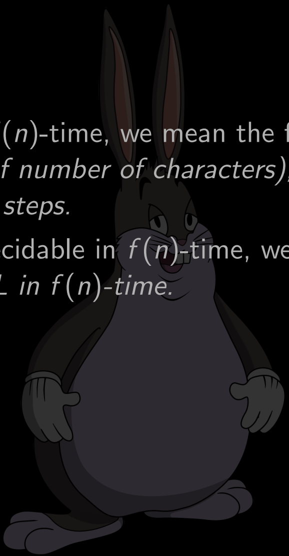
Learning objectives this tutorial



- Formulate the *Hamiltonian Cycle Problem*, and then the *Hamiltonian Path Problem*.
- Show that the *Hamiltonian Cycle Problem* (and the Hamiltonian Path Problem) can be decided by a NTM in poly-time.
- Show that the *Hamiltonian Cycle Problem* (and the Hamiltonian Path Problem) can be *verified* in poly-time.

Some Clarifications

- When we say that a TM M runs in $f(n)$-time, we mean the following: *For all inputs of length n (in terms of number of characters), the computation $M(n)$ halts within $f(n)$ steps.*
- When we say that a language L is decidable in $f(n)$-time, we mean that *there is some TM that decides L in $f(n)$-time.*



Hamilton

Question: Who's this person?

Hamilton

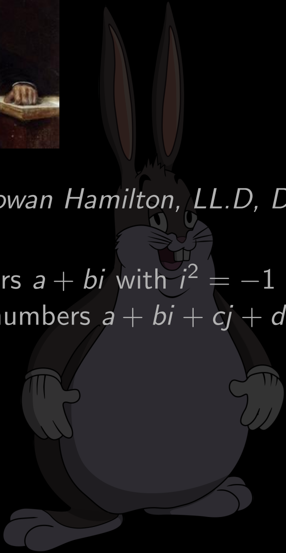
Question: Who's this person?

Ans: Sir William Rowan Hamilton, LL.D, DCL, MRIA, FRAS.

Hamilton Lore

List of things attributed to *Sir William Rowan Hamilton, LL.D, DCL, MRIA, FRAS*:

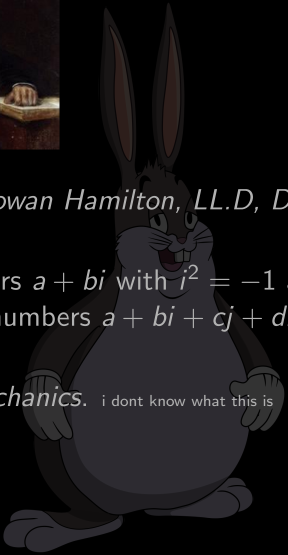
Hamilton Lore



List of things attributed to *Sir William Rowan Hamilton, LL.D, DCL, MRIA, FRAS*:

- Quaternions. (Think complex numbers $a + bi$ with $i^2 = -1$ aren't enough? Introducing 4-dimensional numbers $a + bi + cj + dk$ with $i^2 = j^2 = k^2 = ijk = -1$!)

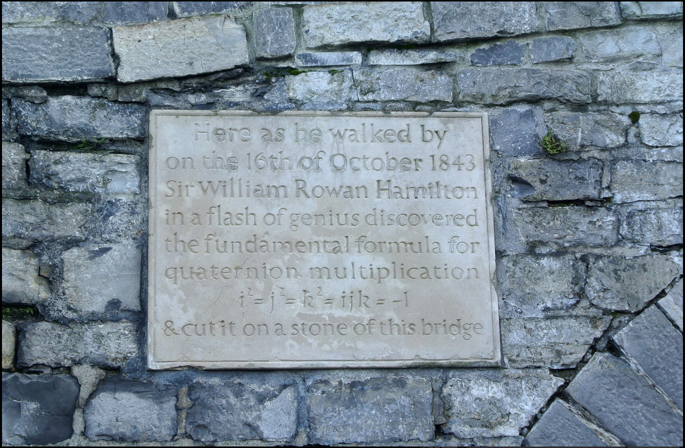
Hamilton Lore



List of things attributed to *Sir William Rowan Hamilton, LL.D, DCL, MRIA, FRAS*:

- Quaternions. (Think complex numbers $a + bi$ with $i^2 = -1$ aren't enough? Introducing 4-dimensional numbers $a + bi + cj + dk$ with $i^2 = j^2 = k^2 = ijk = -1$!)
- Physics, specifically *Hamiltonian Mechanics*. i dont know what this is
- Astronomy. i dont know astronomy either
- Some graph theory.
- Other uninteresting stuff

Hamilton Lore

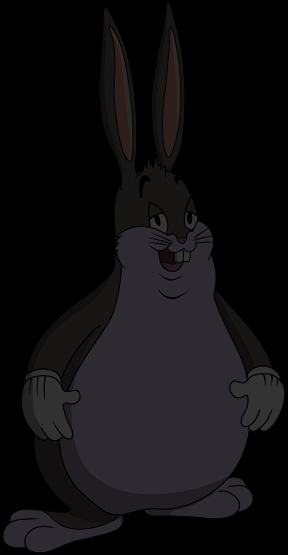


Around Broom Bridge, Dublin. Vandalized many times, of course.

There's also a *Hamilton Walk* event that takes place every year from Dunsink Observatory in Dublin Broom Bridge. Would be funny if they walked in a *Hamiltonian path*, eh.

Hamilton

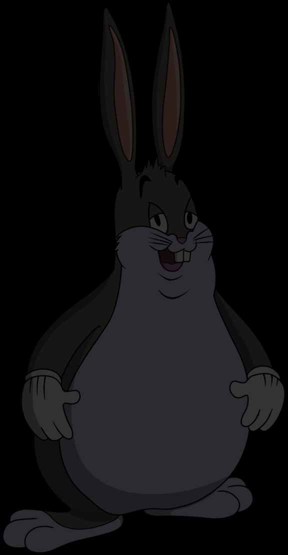
Question: What would someone, living in 19th century Ireland, do when they were bored?



[1] Not sure about other 19th century Irishpeople, but Hamilton certainly did.

Hamilton

Question: What would someone, living in 19th century Ireland, do when they were bored?
Ans: Math.[1]

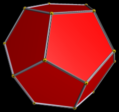


[1] Not sure about other 19th century Irishpeople, but Hamilton certainly did.

Hamilton

Question: What would someone, living in 19th century Ireland, do when they were bored?

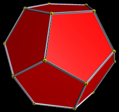
Ans: Math.[1] More specifically, think about whether they can "walk" across a dodecahedron in a loop, visiting every vertex once.



[1]Not sure about other 19th century Irishpeople, but Hamilton certainly did.

Hamilton

Question: What would someone, living in 19th century Ireland, do when they were bored?
Ans: Math.[1] More specifically, think about whether they can "walk" across a dodecahedron in a loop, visiting every vertex once.



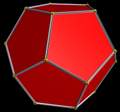
Let's try it! Here's the dodecahedron projected to 2D space.

[1]Not sure about other 19th century Irishpeople, but Hamilton certainly did.

Hamilton

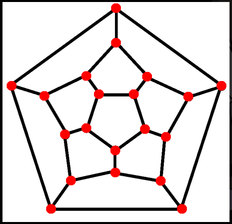
Question: What would someone, living in 19th century Ireland, do when they were bored?

Ans: Math.[1] More specifically, think about whether they can "walk" across a dodecahedron in a loop, visiting every vertex once.



Let's try it! Here's the dodecahedron projected to 2D space.

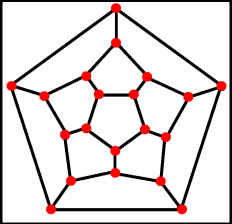
Task: Can you find a cycle in the graph below that visits every vertex exactly once?

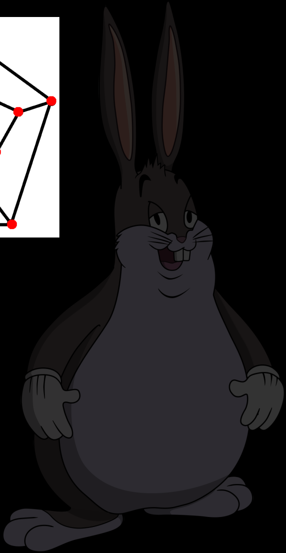


[1] Not sure about other 19th century Irishpeople, but Hamilton certainly did.

Hamilton

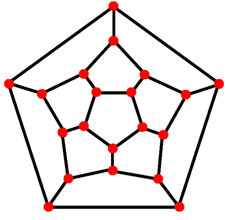
Task: Can you find a cycle in the graph below that visits every vertex exactly once?



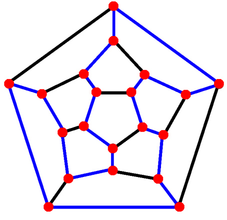


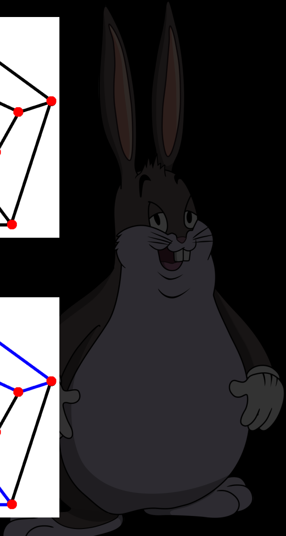
Hamilton

Task: Can you find a cycle in the graph below that visits every vertex exactly once?



Ans: Yes!

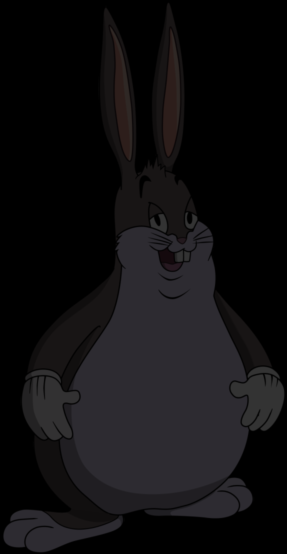




This kinda reminds me of "connect the dots" games I used to play as a kid

Hamiltonian Cycle Problem

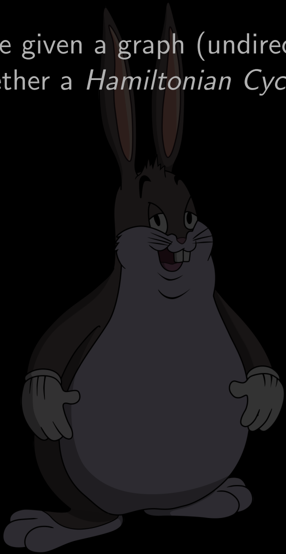
The previous "connect the dots in a loop" game is a specific instance of the **Hamiltonian Cycle Problem**.



Hamiltonian Cycle Problem

The previous “connect the dots in a loop” game is a specific instance of the **Hamiltonian Cycle Problem**.

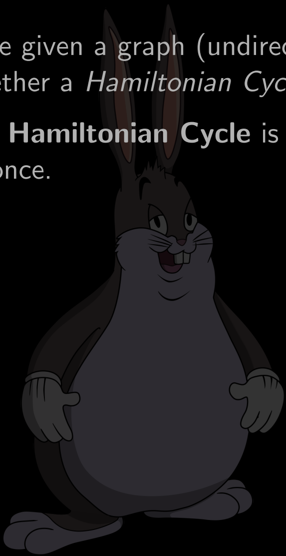
In the Hamiltonian Cycle Problem, you are given a graph (undirected in our context), and asked to determine whether a *Hamiltonian Cycle* exists.



Hamiltonian Cycle Problem

The previous "connect the dots in a loop" game is a specific instance of the **Hamiltonian Cycle Problem**.
In the Hamiltonian Cycle Problem, you are given a graph (undirected in our context), and asked to determine whether a *Hamiltonian Cycle* exists.

Definition: Given a graph $G = (V, E)$, a **Hamiltonian Cycle** is a cycle that contains each vertex $v \in V$ exactly once.

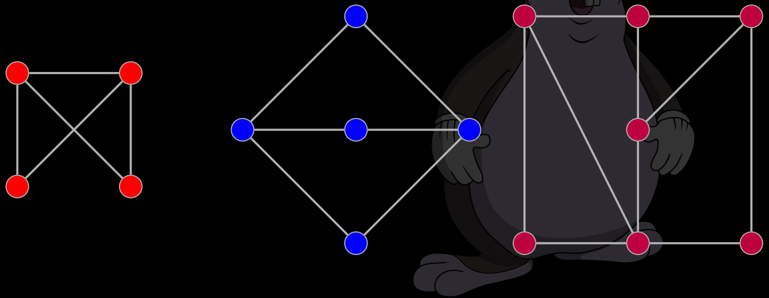


Hamiltonian Cycle Problem

The previous “connect the dots in a loop” game is a specific instance of the **Hamiltonian Cycle Problem**.
In the Hamiltonian Cycle Problem, you are given a graph (undirected in our context), and asked to determine whether a *Hamiltonian Cycle* exists.

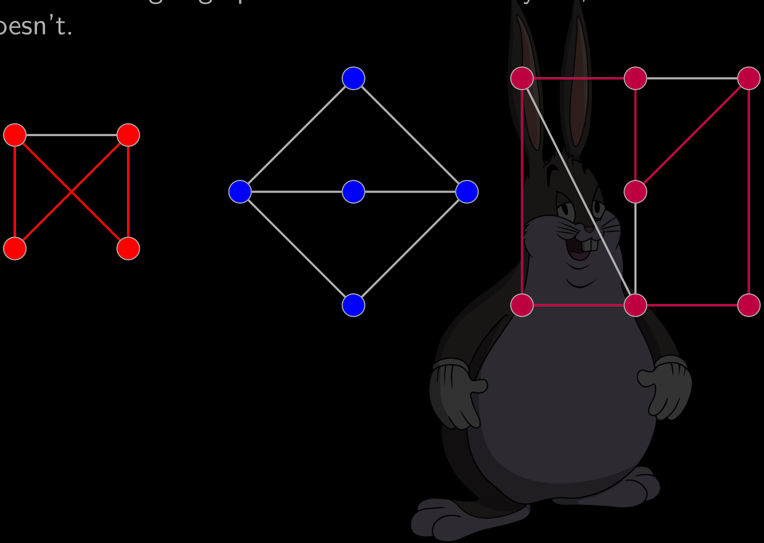
Definition: Given a graph $G = (V, E)$, a **Hamiltonian Cycle** is a cycle that contains each vertex $v \in V$ exactly once.

Task: Determine whether Hamiltonian cycles exist in the following graphs.



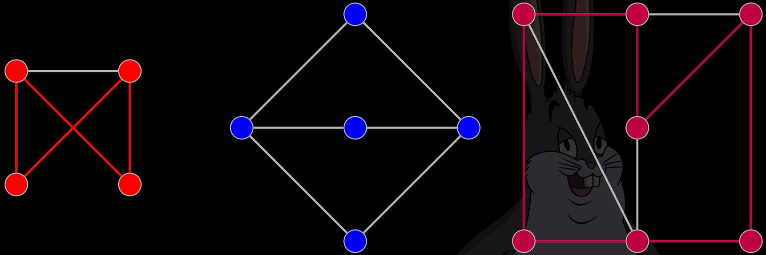
Hamiltonian Cycle Problem

Ans: The left and right graphs have Hamiltonian cycles, while the middle graph doesn't.



Hamiltonian Cycle Problem

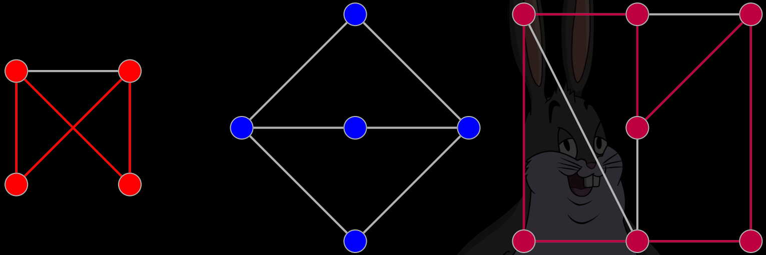
Ans: The left and right graphs have Hamiltonian cycles, while the middle graph doesn't.



Question: How do we know that there is no Hamiltonian cycle for the middle graph?

Hamiltonian Cycle Problem

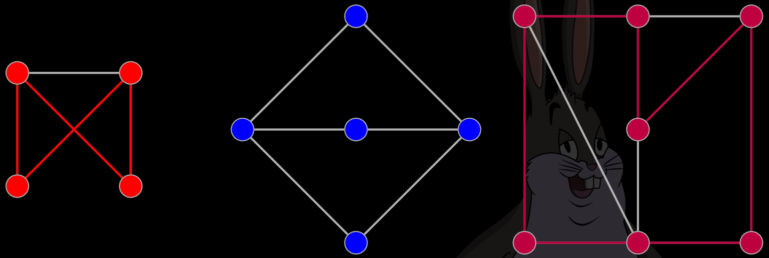
Ans: The left and right graphs have Hamiltonian cycles, while the middle graph doesn't.



Question: How do we know that there is no Hamiltonian cycle for the middle graph?
Ans: 💀brute force.

Hamiltonian Cycle Problem

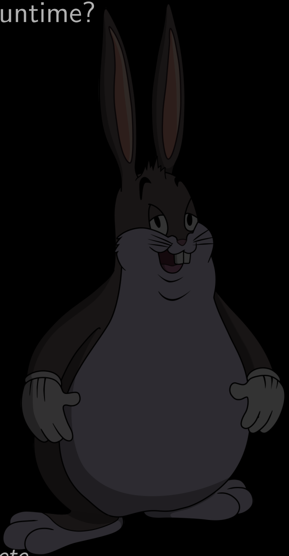
Ans: The left and right graphs have Hamiltonian cycles, while the middle graph doesn't.



Question: How do we know that there is no Hamiltonian cycle for the middle graph?
Ans: 💀 brute force.
Insight: This gives us another "hard to solve, easy to verify" problem.

Hamiltonian Cycle Problem

Task: Describe, in pseudocode, how you can "brute-force" the Hamiltonian cycle problem. What is the runtime?



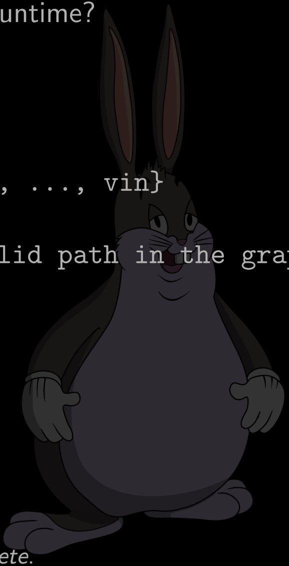
[2] The Hamiltonian Cycle problem is *NP-complete*.

Hamiltonian Cycle Problem

Task: Describe, in pseudocode, how you can "brute-force" the Hamiltonian cycle problem. What is the runtime?
Ans:

```
def has_hcycle(V, E):
  suppose V = {v1, v2, ..., vn}
  for every permutation {vi1, vi2, ..., vin}
    of {v1, v2, ..., vn}:
    if vi1->vi2->...->vin is a valid path in the graph:
      return True
  return False
```

The runtime is...



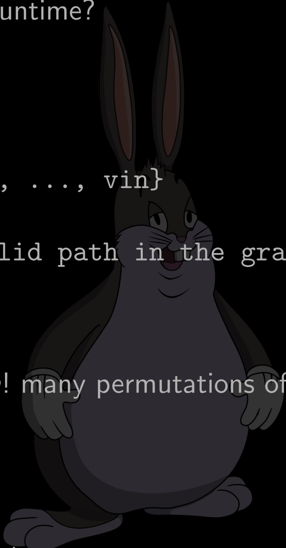
[2] The Hamiltonian Cycle problem is *NP-complete*.

Hamiltonian Cycle Problem

Task: Describe, in pseudocode, how you can "brute-force" the Hamiltonian cycle problem. What is the runtime?
Ans:

```
def has_hcycle(V, E):
  suppose V = {v1, v2, ..., vn}
  for every permutation {vi1, vi2, ..., vin}
    of {v1, v2, ..., vn}:
    if vi1->vi2->...->vin is a valid path in the graph:
      return True
  return False
```

The runtime is... $O(n!)$ (since there are $n!$ many permutations of n vertices).

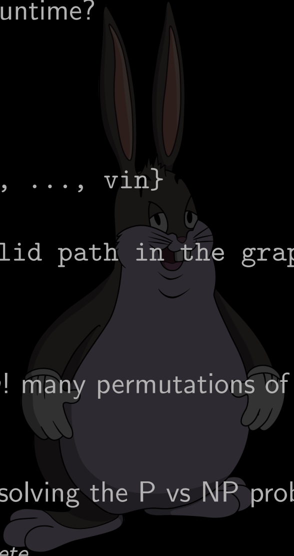


[2]The Hamiltonian Cycle problem is *NP-complete*.

Hamiltonian Cycle Problem

Task: Describe, in pseudocode, how you can "brute-force" the Hamiltonian cycle problem. What is the runtime?
Ans:

```
def has_hcycle(V, E):
  suppose V = {v1, v2, ..., vn}
  for every permutation {vi1, vi2, ..., vin}
    of {v1, v2, ..., vn}:
    if vi1->vi2->...->vin is a valid path in the graph:
      return True
  return False
```

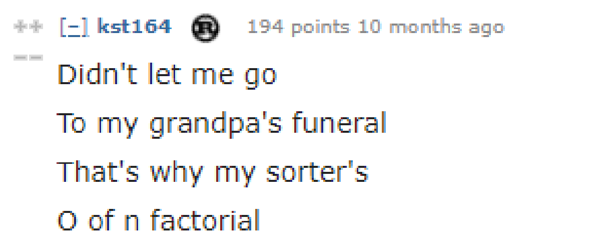


The runtime is... $O(n!)$ (since there are $n!$ many permutations of n vertices).
Question: Can we do better?
Ans: 💀 we don't know... (this is akin to solving the P vs NP problem)[2]

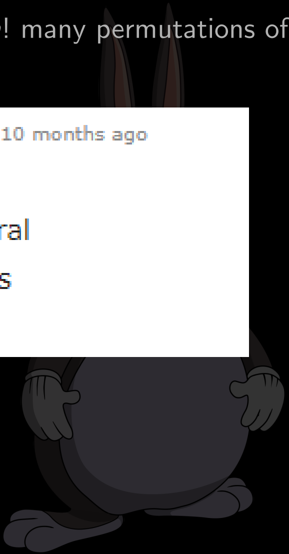
[2] The Hamiltonian Cycle problem is *NP-complete*.

Hamiltonian Cycle Problem

The runtime is... $O(n!)$ (since there are $n!$ many permutations of n vertices).

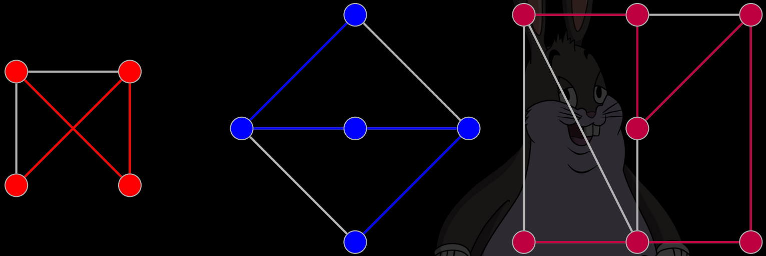


source



Hamiltonian Cycle Problem

There's a similar problem, called the **Hamiltonian Path Problem**, which involves visiting every vertex exactly once in a path (without having to loop back to the beginning).



The middle graph has a solution to the Hamiltonian Path Problem.

NTM solution to Hamiltonian Cycle Problem

Task: Build a poly-time NTM that decides the language

$$\mathrm{HC} = \{G : \text{There is a Hamiltonian Cycle in } G\}.$$

You should define `in_HC(V, E)` (where (V, E) is the graph).

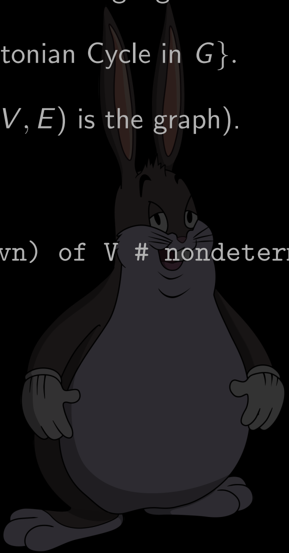
NTM solution to Hamiltonian Cycle Problem

Task: Build a poly-time NTM that decides the language

$$\mathrm{HC} = \{G : \text{There is a Hamiltonian Cycle in } G\}.$$

You should define `in_HC(V, E)` (where (V, E) is the graph).
Ans:

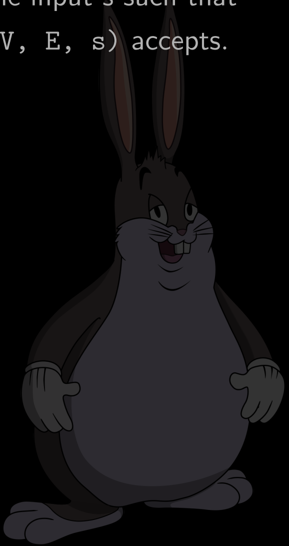
```
in_HC(V, E):
  choose a permutation (v1, ..., vn) of V # nondeterministic!
  for i in 1, ..., n-1:
    if (vi, v(i+1)) not in E:
      reject
  if (vn, v1) not in E:
    reject
  accept
```



NTM solution to Hamiltonian Cycle Problem

Task: Build a poly-time TM `verify_HC(V, E, s)` such that

$(V, E) \in \mathrm{HC} \Leftrightarrow$ There is some input s such that `verify_HC(V, E, s)` accepts.



NTM solution to Hamiltonian Cycle Problem

Task: Build a poly-time TM `verify_HC(V, E, s)` such that

$$(V, E) \in \mathrm{HC} \Leftrightarrow \text{There is some input s such that } \texttt{verify\_HC(V, E, s)} \text{ accepts.}$$

Ans:

```
verify_HC(V, E, s: string):
  let V = (v1, ..., vn)
  if s is not of the form "(v{k1}, ..., v{kn}})":
    reject
  parse s to extract vertices vk1, ..., vkn
  for i in range(n-1):
    if (vki, vk(i+1)) not in E:
      reject
  if (vkn, vk1) not in E:
    reject
  accept
```

NTM solution to Hamiltonian Cycle Problem

Task: Build a poly-time TM `verify_HC(V, E, s)` such that

$$(V, E) \in \mathrm{HC} \Leftrightarrow \text{There is some input s such that } \texttt{verify\_HC(V, E, s)} \text{ accepts.}$$

Ans:

```
verify_HC(V, E, s: string):
  let V = (v1, ..., vn)
  if s is not of the form "(v{k1}, ..., v{kn}})":
    reject
  parse s to extract vertices vk1, ..., vkn
  for i in range(n-1):
    if (vki, vk(i+1)) not in E:
      reject
  if (vkn, vk1) not in E:
    reject
  accept
```

`verify_HC(V, E, s)` acts as a *verifier*: it checks whether a prospective "solution" `s` to the Hamiltonian Cycle problem actually works.

NTM poly-time versus verifiable in poly-time

There is a more general definition of a *verifier*.

Definition: A verifier V for a language L is a Turing machine that satisfies

$$x \in L \Leftrightarrow (\exists s)\text{V(x, s) accepts.}$$

A string s is called a **certificate** if $V(x, s)$ accepts.