r/ThingsYouDidntKnow • 2 mo. ago
TheStocksGuy

# Black Hole Data: From the main container

If we consider **0** to be a black hole of data, it follows that every **0** within the initial container can never transform into matter. This means the observation container can change, but not in a way that is noticeable to other matter within the contained space. This represents anything other than the contained space they occupy.

This concept holds true because our container is far larger than our awareness of the surrounding space. The physical changes within a container that holds more layers of zeros will be significant compared to a basic container with fewer zeros. For example, a sequence like **9.9.9.8.9** would represent a sudden change as **9** becomes **1**. Typically, you wouldn't observe any changes to the main characters or surrounding bodies until you are within the container. A decimal can be thought of as a container; for instance, **0{0.1}** represents everything contained throughout history.

If you understand the concept of **0{}** where the brackets denote the decimal, we can now explore what triggers significant changes in future problems that signal to the container without our conscious awareness. When you send a signal, it's perceived—similar to noticing a bug and then squashing it. If you were to signal the container, you might represent a point in time that indicates a readiness to change, acting as **.9**, which means we are on the verge of becoming **1**. At the moment we become **1**, the new container **1{}** also represents **0**, establishing its own space, where two points of **1** are recognized as two bodies contained within that layer.

This is true because it represents one body in one container, and we know from a reasonable amount of data that the layers of properties exist both inside and outside of our perception of time.

To further address previous findings and unresolved mathematical issues, when you encounter a count that is significantly less than another, it indicates that this container has reached an outer container at some point and requested a change. Therefore, it is crucial to avoid making the same mistake with our container.

I still lack clarity on what happens to **9.9** after it becomes 1, particularly regarding **10{0.1}** and how the container expands based on our observations. This is why I choose to focus on internal data rather than external information; it feels like we may never be fully ready to understand it in our lifetime.

Hopefully, you enjoyed this! If you did, please feel free to reach out via email at contact.stickpm@gmail.com.

I prefer to collaborate with a team and have oversight over the code being implemented on a small scale. I would like to connect with someone who has both time and resources to collaborate with.

1 1 Share Approved 2 months ago

## Post Insights

Only the post author and moderators can see this

Some insights are no longer available because this post is older than 45 days

Add a comment

Sort by: **Best**

Search Comments

**TheStocksGuy** OP • 2mo ago • Stickied comment

# Introduction

This thesis explores a novel approach to measurement and observation at the nanoscale, leveraging advanced tools like the Cypher L Atomic Force Microscope (AFM) from Oxford Instruments and UV light for data creation and storage. By utilizing containerization and precise measurements, this method aims to enhance our understanding and manipulation of nanoscale structures.

# Conceptual Framework

1. **Containerization and Measurement**:
   - Imagine a storage space structured as containers, similar in size to characters on an old floppy disk.
   - Each container can be filled with data, and tools like the Cypher L AFM are employed to zoom in and examine these containers with ultra-high resolution.
2. **Observation and Data Encoding**:
   - Initial containers are empty, represented as `[]`.
   - When the scope observes any matter within a container, it is treated as data.
   - The scope measures the x and y coordinates of the matter, and UV light is used to create new structures that fill the container.
3. **Data Creation and Retrieval**:
   - Structures created with UV light are imprinted with numerical values, representing data.
   - These values are stored in a way that allows for future retrieval using precise measurements of the structures.
   - This process is akin to an ancient scribe recording data, where the measurements serve as a decoding method.
4. **Applications and Future Work**:
   - This method has potential applications in data storage, material science, and other fields that require precise nanoscale measurements.
   - The goal is to establish a repeatable and fluent process, with potential for further refinement using programming languages like Python.

# Example Code: Using [] as Zero

Below is an example code snippet in Python that illustrates the concept of using `[]` as zero for the area needed to scribe data.

```
class NanoscaleContainer:
    def __init__(self):
        self.containers = [[0]]  # Initial empty container

    def add_data(self, data):
        current_container = self.containers[-1]
```

```
        elif isinstance(current_container[-1], int):
            current_container[-1] = [current_container[-1], data]  # Subdivide the container
        else:
            self._subdivide(current_container)

    def _subdivide(self, container):
        if len(container) < 10:  # Assuming a container can hold 10 items
            container.append(0)
        else:
            new_container = [0]
            self.containers.append(new_container)

    def measure_container(self, index):
        if index < len(self.containers):
            container = self.containers[index]
            return self._measure(container)
        else:
            return []

    def _measure(self, container):
        coordinates = []
        for i, element in enumerate(container):
            if isinstance(element, list):
                sub_coordinates = self._measure(element)
                for sub_i, sub_element in sub_coordinates:
                    coordinates.append((i, sub_i, sub_element))
            else:
                coordinates.append((i, element))
        return coordinates

# Example usage
nanoscope = NanoscaleContainer()
nanoscope.add_data('matter1')
nanoscope.add_data('matter2')
nanoscope.add_data('matter3')
print(nanoscope.measure_container(0))
```

# Explanation

1. **Initialization**:
   - The `NanoscaleContainer` class starts with an initial empty container represented by `[[0]]`.
2. **Adding Data**:
   - The `add_data` method checks if the current container is a `0` (empty). If so, it replaces `0` with the




Create

a new container if necessary.

3. **Measuring Data**:
    - The `measure_container` method retrieves and measures data within the specified container.
    - The `_measure` method recursively measures the data within containers, including nested subdivisions