# A DISCUSSION OF DIFFICULTY

Building Towards Complexity Measures

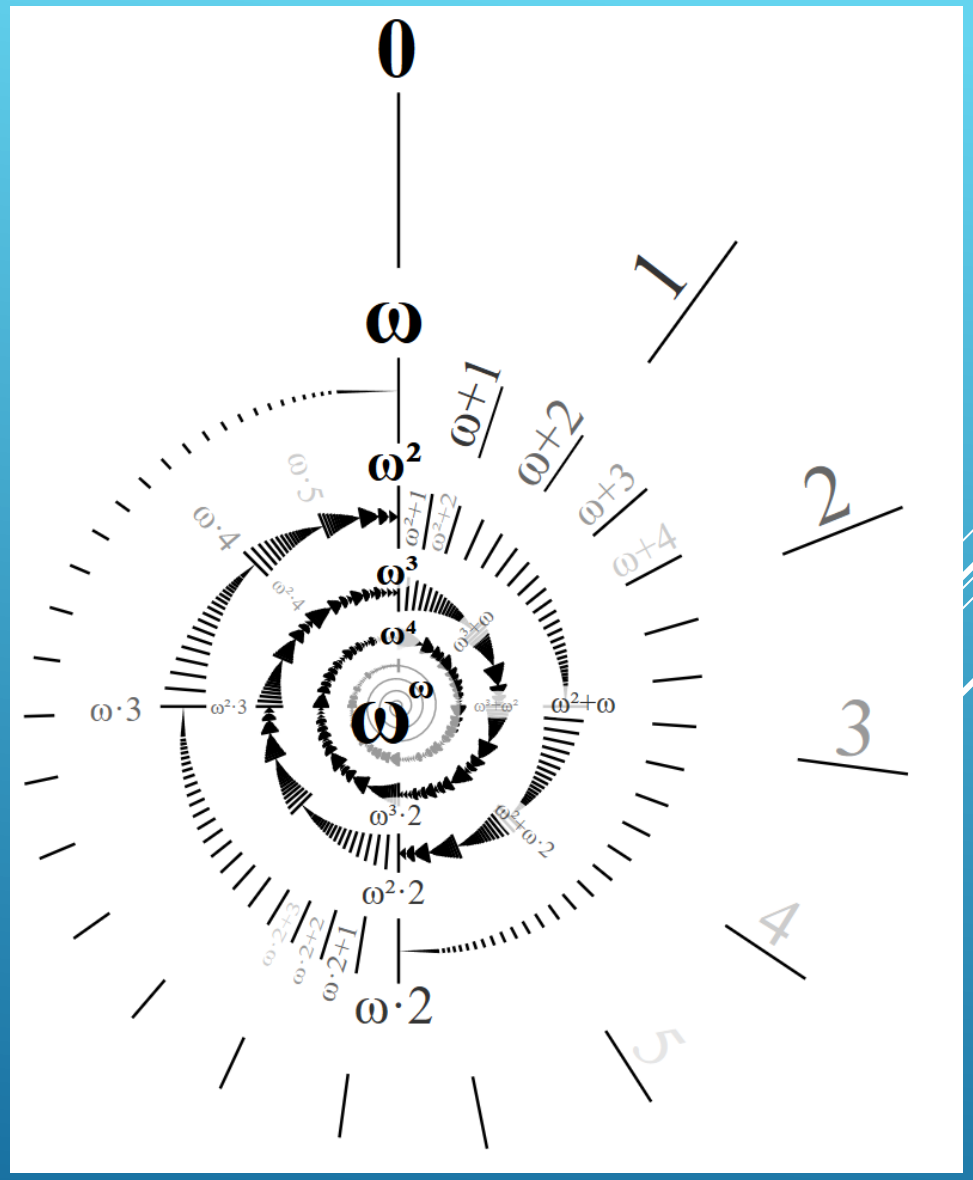
# THE QUESTION AT HAND

- In computability theory, we are always interested in analyzing the **difficulty** computational problems
- But what *is* difficulty? A philosophical question requires a philosophical approach.
- First of all, difficulty is a quantity. We talk of *how difficult* something is. We say this thing is more difficult than that thing. We naturally treat it as a quantity, and this is our starting point.
- There is a complication even here, however, in that we also have difficulty as a binary category. Even now we have some natural candidates for what makes a problem *hard* or *not hard*.

# DIFFICULTY = COMPUTABLE?

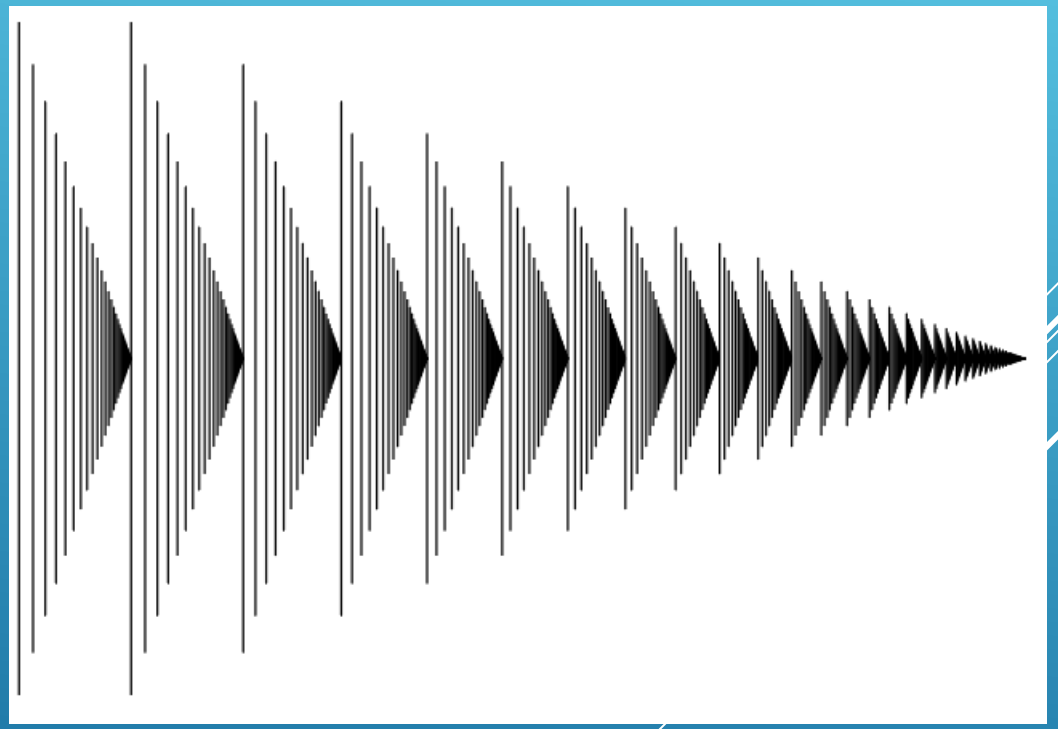
- It should go without saying that if a problem is not computable, then it is difficult. In fact, it's *infinitely more difficult than any computable problem*
- Just as clear is that computable problems can still be very difficult. We should be able to in some sense quantify and characterize difficulty within the sphere of computable problems.
- But once we leave the sphere of computable problems, there could *still* be some problems which are more difficult then others.
  - (There could be problems such that if we gave our model more power in a particular form, we would be able to solve it, but then others where this power is still not enough.

Thus difficulty is in some sense an *ordinal quantity*.



# QUANTITY YIELDING QUALITY YIELDING QUANTITY

- We need to find a way to measure difficulty of **particular computations**
- From this, we will be able to define **qualitative categories** of difficulty.
- These qualitative categories will arrange themselves into a **naturally ordered hierarchy,** from which we will have found a quantification of the difficulty of problems via this order.
- The result will look something like what you see on the right, a visualization of $\omega^2$.
- ($\omega$ is the number form of the 'first' infinity, as a set.) What you see on the right is $\omega$ many times this, i.e. $\omega^2$



# HOW TO BEGIN

- Have to acknowledge that difficulty is a fuzzy concept. We discussed it as a quantity before even considering how to measure it, and it is largely subjective in the way we use it. Yet it is also clearly real. The difficulty of a problem is a concrete, real thing, whether we find what it is or not.
- For inspiration, we look to the social scientists.
- We must settle temporarily for an **operation definition** of difficulty.
- **Assertion**: If a problem is difficult, it will require much of *some* resource of interest within *any* computational model.
- Thus we will **fix** a computational model (Turing machines), and define resources which we can keep track of. The amount of the resources used form an operational measure of a computation's difficulty.
- Turing machine's have two extremely natural resources: **Time and Space**

# NATURAL RESOURCES

- By **time,** we mean the number of *steps* which it takes for a computation to halt
  - Naturally, this corresponds to the actual amount of time a computation would take by a person (or any modern computer) in seconds
- By **space**, we mean the maximum number of tape cells used at any point during a computation.
  - Naturally, this corresponds to the amount of scratch paper used during a computation by a person (or memory of a modern computer) in 'bits'
- There are others, but they are unfortunately largely neglected to this day.
  - For example. **ink**, or the number of times that a symbol is replaced by a distinctly different symbol (as opposed to left alone)
    - Naturally, this corresponds to the amount of ink (or lead used) to write things down, or alternatively the amount of erasing effort.
    - For modern computers though, it corresponds to erasing, which by Landauer's Principle, amounts to the amount of **heat generated by the computation.**

# MICRO TO MACRO

- We can track the amount of resources used during a *specific* computation, but we are interested in problems, not computations.
- Any single decision problem will involve infinitely many computations.

- Can always assume that when we have a Turing machine which decides a problem, then *all computations on an input of a given length will use the same amount of time and space*.
- **Reasonable Assumption: The more complicated (longer) an input, the more resources we should expect to be used in the computation.**
  - Thus, to extrapolate observations about computation to characterizations of problems, we need to inspect **how resource usage scales with the input size.**

Suppose that $L_1$ and $L_2$ are languages (decision problems) which are computable by Turing machines in time $f(n)$ and $g(n)$ respectively. To compare the difficulty of these problems is to compare the *growth rates* of the functions $f$ and $g$.

Suppose that $f(n) = g(n) + 1$. Is $L_1$ really harder than $L_2$?

What about the case $f(n) = cg(n)$ for some positive constant $c$?