Googology

Aaryan Sukhadia

Minicourse, PROMYS 2022

1 Introduction

Readers may remember a childhood experience of competing against their friends or rivals in a race to name the biggest number they could. Obviously, having grown up and matured as mathematicians now, the majority of us have realized this is a ridiculous and futile exercise. Some of those kids, however, never seemed to get the memo. These people are googologists.

We will explore different ways of expressing larger and larger numbers, and what they could possibly be useful for.

2 Graham's Number, Knuth's Arrow Notation

Let us consider an n-dimensional hypercube, and let us draw a line from each vertex to every other vertex.

Now, we color all the edges (there are $\binom{2^n}{2}$ of them) either red or blue. We want to do this in a way to avoid a complete graph on 4 vertices that form a square, as shown in Figure 1.

Q:. *What is the smallest n for which every possible coloring produces the configuration we are trying to avoid?*

We have a lower bound of 13, and an upper bound of Graham's Number. This is a very, *very* bad range. To describe Graham's number we need to introduce Knuth's arrow notation.

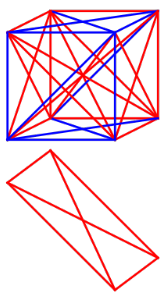

Figure 1: A coloring of the 3-cube. This configuration is something we are trying to avoid.

Definition. Let $a \uparrow^n b$ represent $a \underbrace{\uparrow \cdots \uparrow}_{n \text{ times}} b$. This is defined by:

$$a \uparrow^n b = \begin{cases} a^b, & \text{if } n = 1 \\ 1, & \text{if } n > 1 \text{ and } b = 0 \\ a \uparrow^{n-1} (a \uparrow^n (b-1)) & \text{otherwise} \end{cases}$$

Remark. The convention is that this arrow notation is right-associative. In other words, $a \uparrow b \uparrow c = a \uparrow (b \uparrow c)$ rather than $(a \uparrow b) \uparrow c$

For example, $3 \uparrow 3 = 3^3 = 27$. Let us compute $3 \uparrow\uparrow 3$:

$$\begin{aligned} 3 \uparrow\uparrow 3 &= 3 \uparrow (3 \uparrow\uparrow 2) \\ &= 3 \uparrow (3 \uparrow (3 \uparrow\uparrow 1)) \\ &= 3 \uparrow (3 \uparrow (3 \uparrow (3 \uparrow 0))) \\ &= 3^{3^3} = 3^{27} \\ &\approx 7.6 \times 10^{12} \end{aligned}$$

Exercise 2.0.1. *Show that* $a \uparrow\uparrow b = \left. a^{a^{\cdot^{\cdot^{a}}}} \right\} b$ *times*

Now let us compute $3 \uparrow\uparrow\uparrow 3$. We have:

$$\begin{aligned} 3 \uparrow\uparrow\uparrow 3 &= 3 \uparrow\uparrow (3 \uparrow\uparrow 3) \\ &= \left. 3^{3^{\cdot^{\cdot^{\cdot^{3}}}}} \right\} 3 \uparrow\uparrow 3 \text{ times} \end{aligned}$$

This number is expected to have more digits than can fit in the universe. It is beyond any sort of human comprehension. Now that we have this intution for the sheer stupid magnitude of how fast this arrow notation grows, we are ready to define Graham's Number.

We let $G_1 := 3 \uparrow^4 3$ (which is ridiculously larger than $3 \uparrow^3 3$, which we just calculated). We then create the following recurrence relation:

$$G_{n+1} = 3 \uparrow^{G_n} 3 = 3 \underbrace{\uparrow \cdots \uparrow}_{G_n \text{ arrows}} 3$$

In other words, the number of arrows itself in G_2 is incomprehensibly large, which makes the number of arrows in in G_3 double incomprehensibly large, and so on. Graham's Number is defined as G_{64}.

Thus, the answer to our question is $13 \leq n \leq G_{64}$.

3 Posets and SSCG(3)

Definition. A pair $(P, \leq)$ is a *partially ordered set* (**poset**) if P is a non-empty set and $\leq$ is a *relation* on P such that:

- $\forall x \in P, x \leq x$ (reflexivity)
- $\forall x, y, \in P,\ x \leq y, y \leq x \implies x = y$ (symmetry)
- $\forall x, y, z \in P,\ (x \leq y) \wedge (y \leq z) \implies x \leq z$ (transitivity)

Example 3.0.1. *The usual ordering on* $(\mathbb{R}, \leq)$ *is a poset.*

Note that not every element has to be comparable to every other element in a poset, however. If this is the case, as it is with $(\mathbb{R}, \leq)$, we call it a **totally ordered set**.

Example 3.0.2. *Given any set X, consider the poset of the powerset $P(X)$ ordered by inclusion. (i.e $A \leq B$ iff $A \subseteq B$).*
Verify that these satisfy the poset axioms. Moreover, note that if $X = \{1, 2, 3, 4\}$, then $A = \{1, 2\}$ and $B = \{3, 4\}$ are not comparable.

Definition. Given a poset $(P, \leq)$, a **chain** $\mathcal{C} \subseteq P$ is a set of elements where everything is comparable to everything else. An **antichain** $\mathcal{A} \subseteq P$ is a set of elements where nothing is comparable to anything (except itself).

Example 3.0.3. *If $X = \{1, 2, 3, 4\}$, and our poset is $P(X)$ ordered by inclusion, then $\mathcal{C} = \{\emptyset, \{1\}, \{1, 3\}, X\}$ is a chain and $\mathcal{A} = \{\{1\}, \{2, 3\}, \{4, 3\}\}$ is an antichain.*

Definition. We say a poset is **well-quasi-ordered** if there is no infinite descending chain, and if there is no infinite anti-chain.

Example 3.0.4. $(\mathbb{N}, \leq)$ *is a well-quasi-ordered poset.*

Let us take a sharp turn and discuss graphs. Given a graph G, if G' can be obtained from G be the following 3 steps:

1. Deleting an edge
2. Deleting a vertex
3. Deleting an edge and making the two endpoints of the edges the same vertex,

then G' is said to be a **minor** of G.

Exercise 3.0.5. *Let $\mathcal{G}$ denote the set of all graphs. Show that $\mathcal{G}$ ordered by the minor relationship (i.e $H \leq G$ iff H is a minor of G) gives us a poset*

Theorem 3.0.6 (Robertson-Seymour)**.** *The set of graphs ordered by the minor relation is well-quasi-ordered.*

Showing that $\mathcal{G}$ has no infinite descending chain is quite simple; every minor of a graph has fewer edges or vertices, therefore at some point we must remove everything and we'll be left with the empty graph, and thus we can never have an infinite descending chain. The non-trivial part however, is that there is no infinite anti-chain. Essentially what this is saying is that there is no infinite collection of graphs such that no graph is the minor of another.

Definition. A **simple sub-cubic graph** (SSCG) is a graph where all edges have multiplicity 1 and all vertices have degree at most 3.

For some given integer k, suppose we have a sequence of simple sub-cubic graphs S_1, S_2... such that S_i has at most $i + k$ vertices, and that $\forall i, j, i < j \implies S_i$ is not a minor of S_j. Let a length of a maximal such sequence be denoted $SSCG(k)$.

Note by Robertson-Seymour that $SSCG(k)$ is finite for all k, else that would imply an infinite antichain. However, these sequences can grow absurdly large.

Exercise 3.0.7. *Show $SSCG(0) = 1$, $SSCG(1) = 5$*

$SSCG(3)$ is **MUCH** bigger than Graham's number. In fact, for readers familiar with the TREE function, $SSCG(3) >> TREE(3)$. What's wilder is that it is known that $SSCG(13) >>> TREE^{TREE(3)}(3) = \underbrace{TREE(TREE \cdots TREE(}_{TREE(3)\text{ times}}3)) \cdots)$.

Remark. Though the definition of the TREE function is outside the scope of these notes, it is worth emphasizing that $TREE(3) > G_{G_{64}}$, i.e continuing the recursive formula used for Graham's Number G_{64} times.

4 Turing Machines and Busy Beaver Numbers

A **Turing Machine** is perhaps the most elementary model of computation, in which you have a roll of tape and a machine that can go around writing symbols on that tape, as shown in Figure 2.

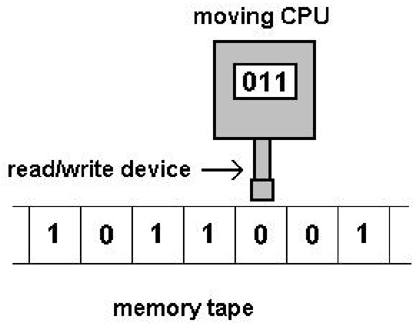

Figure 2: The TM can read and write on one square of the tape at a time, and can move left or right by one square

Pretty much any computation is equivalent to a Turing Machine performing the following steps, known as the n**-state Busy Beaver game**:

- The TM has n operational 'states', plus a halting state in which it stops and does nothing. These states are numbered $1, 2...n$ and it starts on state 1. We can denote the halt state as state 0.
- The TM starts on an infinite roll of tape filled with squares, and each square can be either a 0 or a 1. In the initial stage, all squares are 0s.
- The TM has a *transition function* with two inputs:
  1. Current (non-Halt) state of the TM
  2. Symbol in the current tape square

  and three outputs:
  1. Whether to keep the square the same value or flip the bit.
  2. Whether to move left or right after the step
  3. The state to transition into, potentially the same exact state or the halt state.

For each state, for each input we have 2 options of flipping or preserving the bit given an input, 2 options of the direction to move in afterwards, and $(n+1)$ options for which state to go into. This gives us a total of $4(n+1)$ options for each input on each state, giving us $(4n+4)^2$ options on each state. Doing this for all n states gives us $(4n+4)^{2n}$ possible n-state Turing machines.

Observation. *There are two options: either the TM will continue looping forever, or at some point it will halt*

Let's just consider the set of TMs on n states that eventually halt at some point. We call the number of 1s left on the tape after the halting point the *score* of the n-state Busy Beaver game.

Definition. The Turing Machine with the largest score is called the **champion** of the n-state Busy Beaver game. The function $BB(n)$ returns the score of the champion.

Theorem 4.0.1. *$BB(n)$ grows faster than any computable function (by computable, we mean any function that can be simulated on some Turing Machine).*

Proof. Suppose there existed a way for a Turing Machine M to compute $BB(n)$ for any input n. Let M have m different states. Take another Turing Machine L that takes any input of n 1s and outputs $2n$ 1s, and let L have ℓ states. We then perform the following algorithm using a Turing Machine K:

- Print n 1s *(requires n states)*
- Print $2n$ 1s *(requires ℓ states)*
- Print $BB(BB(2n))$ 1s *(requires $2m$ states)*

Thus, K has $\leq 2m + \ell + n$ states, but it can print $BB(BB(2n))$ 1s. Thus, $BB(n + 2m + \ell) \leq BB(BB(2n))$ Note that the Busy Beaver function is strictly monotone increasing, and that m and ℓ are fixed. Thus, increasing n to arbitrarily large numbers gives us a contradiction (e.g if $n > 2m+\ell$) $\square$

Q:. *Can we disprove Goldbach's Conjecture by only testing a finite number of counterexamples?*

Goldbach's conjecture asks if every even number can be written as the sum of two primes. As of now there is no known counterexample, but there is no known proof either. Note that testing if an even number is the sum of two smaller primes is a computable function. Thus, if we can compute an n-state Turing Machine to test if a given even number is a counter-example, and print the smallest one.

Using $BB(n)$, we can extract essentially the largest number N that an n-state Turing machine can output before halting. In otherwords, if our specially constructed TM does not halt after testing if N is a counterexample, we know that it won't halt, and therefore there *cannot* be a counterexample. Thus, we only have to check a finite number of values! Simple!

What makes this practically intractable is the sheer size of these Busy Beaver numbers. Graham's function, the $TREE$ function and the $SSCG$ function are all computable. This means that the busy beaver numbers grow faster than all of them.

References

Gary Googology.