The hidden sequences of Pascal's triangle

Tom Coleman

Summary

In the first of four linked presentations, this session looks at Pascal's triangle and the various number sequences contained within. Some of these sequences are on the surface, and some of the sequences are incredibly hidden! Once these sequences are found, they will be examined and compared against each other, where their role in one of the most famous/infamous open problems in theoretical computer science will be revealed.

Pascal's triangle

Central to many different and disparate areas of mathematics, one of the most interesting objects in the subject is **Pascal's triangle**.

> **How to construct Pascal's triangle**
>
> Start with 1 at the top point of a triangle.
> Write 1's down the sides of the triangle.
> To fill in the rest of the entries, add two numbers next to each other in a row, and write the answer below and between both of them.

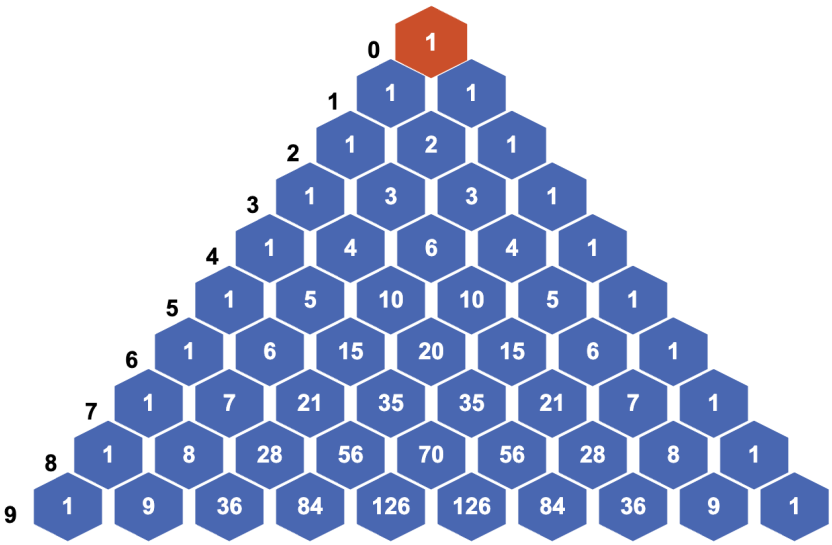

Figure 1: A figure demonstrating the first 10 rows of Pascal's triangle, with row numbers in black and to the left.

Why Pascal's triangle? A brief history

While this is known as Pascal's triangle in most of Western civilization, the idea of generating numbers in a triangle by adding the above two entries predates Pascal the mathematician by several hundred years. In fact, the history of the triangle lives on in the name given to it by other countries:

- In Iran, it's called **Khayyam's triangle** after the Persian mathematician Omar Khayyam, who used the construction in the 11th century to find nth roots of numbers and describe bracket expansions.. In fact, they were not the first Persian to discuss the triangle: this was al-Karaji.
- In China, it's called **Yang Hui's triangle** after the Chinese mathematician Yang Hui, using it to find the sum of triangular numbers (more on that later...). They also wrote many more mathematical texts on subjects from algebra to magic squares.
- In Italy, it's called **Tartaglia's triangle** after the Italian mathematician Niccolò Tartaglia, who (amongst other things) helped to solve equations involving cubic terms x^3. Notably, they participated in what can only be described as the 16th century Italian version of a rap battle.

Pascal himself did expose the deep connections between the triangle and probability theory, and so the common name Pascal's triangle is not without merit!

Here's something you can figure out.

! Try it yourself 1

Using the defining relation and the figure above, work out the eleventh row of Pascal's triangle. Write down any observations you have about the triangle so far.

i Answer to try it yourself 1

The next row should be

$$1 \;\; 10 \;\; 45 \;\; 120 \;\; 210 \;\; 252 \;\; 210 \;\; 120 \;\; 45 \;\; 10 \;\; 1$$

where each term is obtained by adding the two terms directly above it.
You can notice that

- every row is symmetric down the middle (and therefore so is the triangle)
- every even numbered row has a unique number in the centre
- every odd numbered row has two numbers the same in the middle.
- there is $1, 2, 3, 4, 5, 6, \ldots$ down the diagonals in numerical order.

You will see more on these properties in the next exploration, which can be found at Exploration: How to win at cards - combinations and permutations.

Sequences hidden in Pascal's triangle

A **sequence** is any finite or infinite list of numbers. Any number in a sequence is called a **term**. The term in position n of the sequence is called the n**th term** of the sequence.

Here are some sequences of numbers:

$$(1) \quad 1, 1, 1, 1, 1$$

$$(2) \quad 1, 1, 1, 1, 1, \ldots$$

$$(3) \quad 1, 2, 3, 4, 99, \ldots$$

$$(4) \quad 1, 99, 2, 3, 4, \ldots$$

$$(5) \quad 1, 4, 6, 9, 16, \ldots$$

$$(6) \quad 1, 1, 2, 3, 5, \ldots$$

Here, (1) is a finite sequence with five terms; all of the others $(2) - (6)$ are infinite sequences - which you can see by the three dots ... at the end of each.

As evidenced by (2), every term in the sequence can be the same; such a sequence is called a **constant sequence**.

Sequence (3) shows that a sequence doesn't have to follow a prescriptive pattern; they really can be any list of numbers. The 5th term of the sequence is 99.

Sequences (3) and (4) show that the order of the terms in the sequence matters; these are not the same sequence.

Sequence (5) is an example of a sequence where the nth term is given by a formula involving n. In this case, the nth term of the sequence is given by n^2.

Sequence (6) is an example of a sequence where the nth term is given by some combination of previous terms. It can be defined by setting the first and second term to be 1, and then every subsequent term is the sum of the previous two terms. (This is the famous **Fibonacci sequence**.)

What's the point?

Sequences of numbers help to measure **growth** of things. Knowing which sequences grow faster than others is a critical skill in measuring the speed of computer algorithms, as well as behaviour of some mathematical objects as they approach infinity.

Triangular numbers

Looking down the diagonals of Pascal's triangle (in either direction) gives infinite sequences of numbers.

- The diagonal starting at row 0 (the side of the triangle) gives an infinite sequence of 1's, which is pretty boring.
- The diagonal starting at row 1 gives the sequence of positive whole numbers $1, 2, 3, 4, 5, ...$, which is more interesting.
- The diagonal starting at row 2 gives the sequence

$$1, 3, 6, 10, 15, ...$$

  Because of the way that Pascal's triangle is created, you can get the terms of the

sequence from the two entries above it:

$$\begin{aligned} 1 &= 1 \\ 3 &= 1 + 2 \\ 6 &= 3 + 3 \\ 10 &= 6 + 4 \\ 15 &= 10 + 5 \\ \vdots \quad & \quad \vdots \end{aligned}$$

But you can notice that the first terms of these sums are the sequence on the left, allowing you to write

$$\begin{aligned} 1 &= 1 \\ 3 &= 1 + 2 \\ 6 &= 1 + 2 + 3 \\ 10 &= 1 + 2 + 3 + 4 \\ 15 &= 1 + 2 + 3 + 4 + 5 \end{aligned}$$

You can visualize these numbers by **triangles**, where each term of the sum is the number of dots in a subsequent row. This sequence of numbers is therefore called the **triangular numbers**. The nth term of a triangular number is written as T_n, so you could write

$$T_1 = 1,\ T_2 = 3,\ T_3 = 6,\ T_4 = 10,\ T_5 = 15,\ ...$$

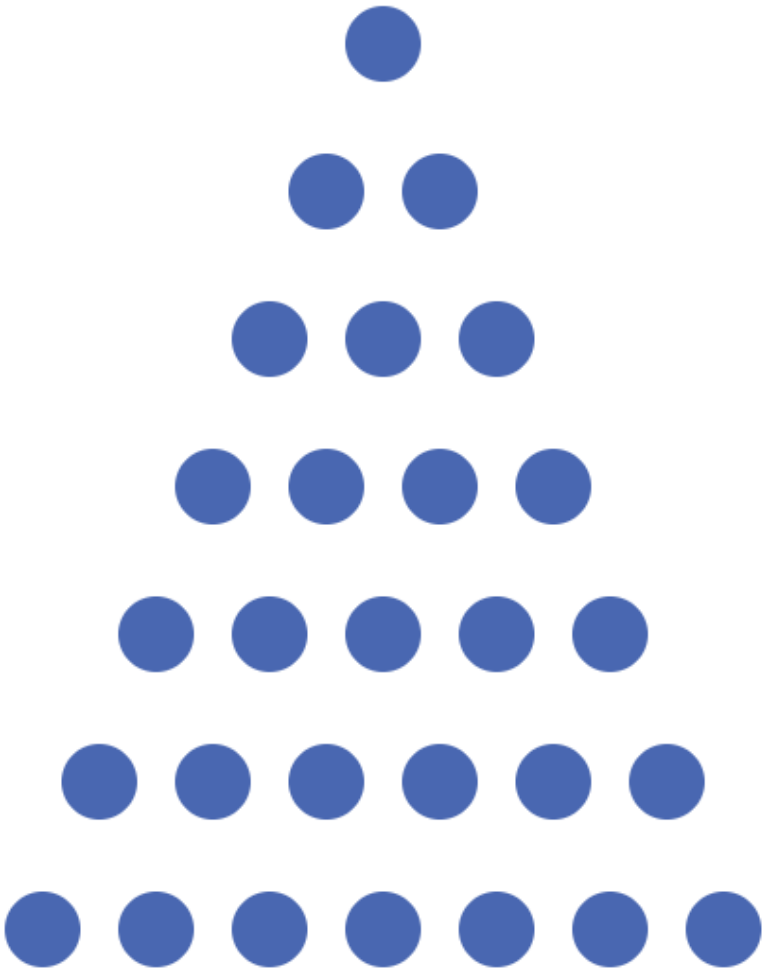

Figure 2: A figure illustrating the seventh triangular number T_7.

Formula

You have already seen that you can get the nth triangular number T_n by adding n to the previous triangular number T_{n-1}; so

$$T_n = T_{n-1} + n.$$

However, mathematicians (yourself included) shouldn't be satisfied with this. Using this formula, to work out the 100th triangular number, you need to work out the previous 99. Is there a way to work out the nth term of the sequence **without** working out all of the previous $n-1$ terms? This is the same question as asking - is there a formula for the nth triangular number? The answer is yes:

Formula for nth triangular number

The formula for the nth triangular number is given by

$$T_n = \frac{n(n+1)}{2}.$$

Next, you need to **prove** that this is true. It's all well and good saying that something is true, but you need to be able to back that claim up. In maths, this is done by starting with an assumption, following a series of logical steps until you get to a conclusion. Here, the assumption is that $T_n = 1 + 2 + 3 + ... + n$ and the conclusion is that $T_n = \frac{n(n+1)}{2}$.

Proof of the formula for T_n

To show that this is true, you can use some geometry. The idea is to count the number of dots in the triangle, which can be found from the fact that $T_n = 1 + 2 + 3 + ... + n$. You can left-align the triangle to create a right angled triangle with the same number of dots. Next, do this again. Put the two right-angled triangles together to get a rectangle, as shown in the picture.

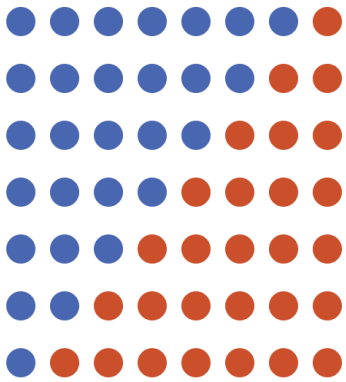

Figure 3: A pictorial proof of the formula for T_n.

The rectangle then has n dots along the vertical side and $n+1$ dots along the horizontal side. This means that there are $n(n+1)$ dots in the rectangle. But the number you are looking for is the number of dots in triangle, which is exactly half of the rectangle. So

$$T_n = \frac{n(n+1)}{2}.$$

You can see this in detail by clicking the 'show double-triangle proof' on the interactive element above.

History time: Gauss and his teacher

According to history, this problem was solved by Carl Friedrich Gauss at the age of **seven years old** when he realized that you can work out $1+2+3+...+n$ by reversing the order of the sum to get $n+(n-1)+...+2+1$ and adding to get n lots of $(n+1)$:

$$\begin{array}{rcccccccccc} T_n & = & 1 & + & 2 & + & 3 & + & ... & + & n \\ + \; T_n & = & n & + & (n-1) & + & (n-2) & + & ... & + & 1 \\ \hline 2T_n & = & (n+1) & + & (n+1) & + & (n+1) & + & ... & + & (n+1) \end{array}$$

You can then rearrange to get $T_n = n(n+1)/2$ as you have already seen.
To do this at the age of seven gave a small hint of the genius to follow!

Square numbers

It's all well and good to place two triangular numbers together to make a rectangle - can you get squares instead? The number of dots in the squares are called

As it turns out, you can. You need to add the $(n-1)$th triangular number to the nth to get a square shape, so you can reasonably deduce that:

Formula for nth square number

$$n^2 = T_n + T_{n-1}$$

Once again, you need to back up this statement with a proof. You can approach this proof in two ways. You could use a geometric argument as above (you can view this on the webpage by clicking 'show square-number proof' on the interactive figure).

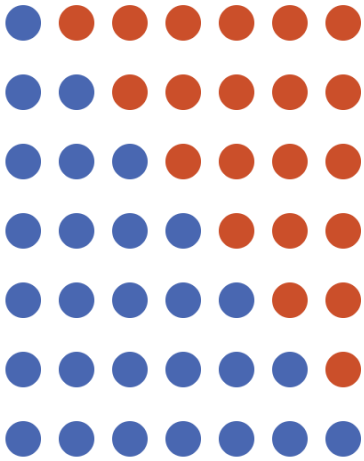

Figure 4: A pictorial proof of the formula for n^2 in terms of triangular numbers.

Or you could do some algebra. You know already that $T_n = n(n+1)/2$, and $T_{n-1} = (n-1)n/2$. Adding these together and factorizing gives

$$\begin{aligned} T_n + T_{n-1} &= \frac{n(n+1)}{2} + \frac{(n-1)n}{2} \\ &= \frac{n}{2}\left((n+1) + (n-1)\right) \\ &= \frac{n}{2}(2n) = \frac{2n^2}{2} = n^2 \end{aligned}$$

In fact, this formula does a lot more than this. This tells you explicitly that square numbers are always larger than triangular numbers for $n > 1$:

! Try it yourself 2: formula comparing

Show that $n^2 > T_n$ for $n > 1$.

i Answer to try it yourself 2

You know that $n > 1$ and so $n \geq 2$. This means that $n - 1 \geq 1$ and so T_{n-1} is a positive integer. It follows from there that $n^2 - T_{n-1} < n^2$. Since $T_n = n^2 - T_{n-1}$ from above, it follows that

$$T_n = n^2 - T_{n-1} < n^2$$

and the result follows.

What's the point?

Triangular numbers are used in two quite disparate areas:

- If you are running a sports/gaming league with n teams/players/things in it, the amount of matches required so that every competitor plays every other competitor exactly once is T_{n-1}.
- Triangular numbers are used in accounting to calculate the depreciation of assets over a period of years.

Square numbers n^2 are the start of investigation of **polynomial growth**. A sequence has polynomial growth if the nth term of the sequence can be expressed as some polynomial expression with largest term n^k. Since the sequence of triangular numbers can be expressed by $(n^2 + n)/2$, this sequence has polynomial growth.

Interlude: Tetrahedral numbers

You can continue working your way down the diagonals of Pascal's triangle. What if you start your diagonal at row 3? You get the sequence of numbers:

$$1,\ 4,\ 10,\ 20,\ 35,\ ...$$

In the same way as the triangular numbers, using the defining property of Pascal's triangle you can write each of these as

$$\begin{aligned} 1 &= 1 \\ 4 &= 1 + 3 \\ 10 &= 4 + 6 \\ 20 &= 10 + 10 \\ 35 &= 20 + 15 \\ \vdots &\quad\ \vdots \end{aligned}$$

But you can notice that the first terms of these sums are the sequence on the left, allowing you to write

$$\begin{aligned} 1 &= 1 \\ 4 &= 1 + 3 \\ 10 &= 1 + 3 + 6 \\ 20 &= 1 + 3 + 6 + 10 \\ 35 &= 1 + 3 + 6 + 10 + 15 \end{aligned}$$

These are consecutive triangular numbers added together! In fact, much in the same way you can organise triangular numbers into triangles, you can organise these into stacks of triangles - **tetrahedrons**. So these are called **tetrahedral numbers** Te_n, and are defined by the formula

$$Te_n = Te_{n-1} + T_n.$$

There is also a closed form formula which you *can* prove geometrically (but suggest you don't), given by

$$Te_n = \frac{n(n+1)(n+2)}{6}.$$

Notice that there is an n^3 term in here - this signifies a shift to three dimensions, and also that tetrahedral numbers grow faster than both triangular and square numbers, but still have polynomial growth.

Interlude: Fibonacci numbers

Here's a neat thing hiding in Pascal's triangle.

- Left-align Pascal's triangle using the tool above (or in Interactive: Pascal's triangle).
- Starting in the left-most column of 1's, add up entries in diagonals in the northeast direction.
- You should get the sequence $1, 1, 2, 3, 5, 8, 13, 21, ...$, which is the famous **Fibonacci sequence**.

The Fibonacci sequence F_n is defined by the following relation:

$$F_0 = 1, \quad F_1 = 1, \quad F_n = F_{n-1} + F_{n-2}$$

and it is **everywhere** in mathematics and nature:

- The Fibonacci sequence is closely related to the **golden ratio** $\varphi = (1 + \sqrt{5})/2 = 1.618...$, which appears in art as the proportion of the golden rectangle
- In biology, flowers tend to grow a Fibonacci number of petals as opposed to any other type of number
- In genetics, the number of possible ancestors on an X chromosome line is a Fibonacci number
- In computer science, Fibonacci numbers provide 'worst case scenarios' for the Euclidean algorithm and working out continued fractions.

How fast does the Fibonacci sequence grow? Well, a closed form formula for the Fibonacci numbers is

$$F_n = \frac{\left(\frac{\sqrt{5}+1}{2}\right)^n + \left(\frac{\sqrt{5}-1}{2}\right)^n}{\sqrt{5}}$$

and so this is actually *not* polynomial growth, as it's to a power of n rather than n to the power of something.

Powers of 2

! Try it yourself 3

Add up each row of Pascal's triangle in turn, and write up your answers. What do you get?

i Answer to try it yourself 3

Here's a table of the answers.

Row number	0	1	2	3	4	5	6	7	8	9
Sum of numbers in row	1	2	4	8	16	32	64	128	256	512

Here, the sum of numbers in row n is twice the sum of numbers in row $n-1$. Since row 0 sums to 1, it follows that the sequence of sums of numbers in rows are precisely the **powers of two**.

A proof of this try it yourself will be seen in the next exploration, as it involves choice: Exploration: How to win at cards - combinations and permutations.

i Powers of two, powers of c

The sequence of numbers given by

$$p_0 = 1, \quad p_n = 2 \cdot p_{n-1}$$

are the **powers of two**, with nth term given by $p_n = 2^n$. Here, the notation 2^n means

$$2^n = \underbrace{2 \cdot 2 \cdot 2 \cdot \ldots \cdot 2}_{n \text{ times}}$$

More generally, if c is any positive number, then the sequence of numbers given by

$$c_0 = 1, \quad c_n = c \cdot p_{n-1}$$

are the **powers of** c, with nth term given by $c_n = c^n$.

Tip

Here, the convention is to start the sequence at 0 rather than 1, as the first term is obtained by adding the numbers in row 0 of Pascal's triangle.
For more about powers of numbers, see Guide: Laws of indices.

How do the powers of two compare to the square numbers? Are they bigger, or smaller? Well, comparing the two seems to suggest that powers of two are smaller than square numbers to begin with (for $n = 1, 2, 3$), are equal at $n = 4$, but quickly overtake for larger n. In fact, you can make the statement:

Sequence comparing 1

$2^n > n^2$ for $n > 4$.

How could you show this? The idea is to take the **ratio of the two sequences**, using the mathematical principle that is a, b are positive numbers with $a < b$, then $a/b < 1$. Then, by investigating how the ratio changes as n changes, you're able to determine the behaviour of the sequences.

Sequence comparing 1

Showing that $2^n > n^2$ for $n > 4$ is the same as showing that $n^2/2^n < 1$ for $n > 4$. Write $a_n = n^2/2^n$; you can notice that if $n = 4$, then $a_n = 1$. Therefore, if $a_n < 1$ for $n > 4$, you're done.
So how does a_{n+1} relate to a_n? You can actually work this out. Start with

$$a_{n+1} = \frac{(n+1)^2}{2^{n+1}}.$$

Next, you want to manufacture a_n in this expression. You can do this by taking a factor of n^2 out of the numerator and 2^n out of the denominator (for more about factorization, see Guide: Factorization). Doing this gives, by the laws of ind

$$a_{n+1} = \frac{n^2}{2^n} \cdot \frac{\left(1 + \frac{1}{n}\right)^2}{2} = a_n \cdot \frac{\left(1 + \frac{1}{n}\right)^2}{2}$$

You want a_{n+1} to be smaller than a_n, as the whole object of the argument is to show that $a_n < 1$ for $n > 4$. So how big a_{n+1} is compared to a_n entirely depends on if $\frac{(1+\frac{1}{n})^2}{2}$ is bigger than or smaller than 1.
As it turns out, $(1+1/n)^2/2 < 1$ whenever $n > 2$; this means that $(1+1/n)^2/2 < 1$ whenever $n > 4$ as well. This means that $a_{n+1} < a_n$ whenever $n > 4$ - which means that $n^2/2^n < 1$ for $n > 4$, proving the result.

In fact, for any positive number k, there is a large enough number m such that $2^n > n^k$ for all $n > m$. This means that **exponential growth is always eventually faster than polynomial growth**.

What's the point?

The sequence of powers c^n of any number $c > 1$ can be used to describe **exponential growth**. For instance, bacteria that split in half every 24 hours are an example of exponential growth.

Conversely, the sequence of powers c^n of any number $0 < c < 1$ can be used to described **exponential decay** - which is particularly useful when it comes to management of nuclear waste.

Factorials

As you saw earlier, adding up the first n numbers gives you the nth triangular number T_n. What happens when you **multiply** the first n numbers together?

i Factorials

For a positive whole number n, the number $n!$, called n **factorial**, is defined to be

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n$$

which is the first n whole numbers multiplied together.
You can define a sequence of factorials by

$$f_0 = 1, \quad f_n = f_{n-1} \cdot n$$

So it follows that

$$(n+1)! = (n+1) \cdot n!$$

for all natural numbers

Try it yourself 4

Starting at $n = 0$, work out the first 7 factorials up to $6!$.

Answer to try it yourself 4

Here's a table of the answers.

number n	0	1	2	3	4	5	6
n factorial $(n!)$	1	1	2	6	24	120	720

You can notice that $n!$ starts to grow extremely quickly. How quickly? The answer is: *quicker than exponential growth*.

Sequence comparing 2

$n! > 2^n$ for $n > 3$.

Again, the idea is to take the **ratio of the two sequences**.

Sequence comparing 2

Showing that $n! > 2^n$ for $n > 3$ is the same as showing that $n!/2^n < 1$ for $n > 3$. Write $b_n = 2^n/n!$; you can notice that if $n = 3$, then $b_n = 8/6 > 1$. Therefore, if $b_n < 1$ for $n > 3$, you're done.

So how does a_{n+1} relate to a_n? You can actually work this out. Start with

$$a_{n+1} = \frac{2^{n+1}}{(n+1)!}.$$

Next, you want to manufacture a_n in this expression. You can do this by taking a factor of 2^n out of the numerator and $n!$ out of the denominator. Doing this gives

$$a_{n+1} = \frac{2^n}{n!} \cdot \frac{2}{n+1} = a_n \cdot \frac{2}{n+1}$$

You want a_{n+1} to be smaller than a_n, as the whole object of the argument is to show that $a_n < 1$ for $n > 3$. So how big a_{n+1} is compared to a_n entirely depends on if $\frac{2}{n+1}$ is bigger than or smaller than 1.

As it turns out, $2/(n+1) < 1/2 < 1$ whenever $n > 3$. This means that $a_{n+1} < a_n$ whenever $n > 3$. Since $a_4 = 16/24 < 1$, the result follows.

In fact, you can show that for any positive number c, there is a large enough number m such that $n! > c^n$ for all $n > m$. This means that **factorial growth is always eventually faster than exponential growth**. The exact m can be obtained using something called **Stirling's approximation**.

Growth of sequences

! Try it yourself 4

Fill in the following table of sequences to give a full comparison of their properties. For fun, there is also a column for the **super-exponential** sequence n^n.

n	T_n	n^2	F_n	2^n	$n!$	n^n
0	0	_	_	_	_	undef
1	_	_	_	_	_	_
2	_	_	_	_	_	_
3	_	_	_	_	_	__
4	__	__	_	__	__	___
5	__	__	_	__	___	____
6	__	__	__	__	___	_____
7	__	__	__	___	____	______
8	__	__	__	___	_____	_______
9	__	__	__	___	______	________
10	__	___	__	____	_______	_________

! Try it yourself 5

Explain why $n^n > n!$ for all $n > 1$.

i Answer to try it yourself 5

You can see that $n > k$ for all $k = 1, 2, \ldots, n-1$. Therefore, since $ac < bc$ for all

positive numbers a, b, c with $a < b$, it follows that

$$\begin{aligned} n! &= 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n \\ &< n \cdot n \cdot n \cdot \ldots \cdot n \cdot n \\ &= n^n \end{aligned}$$

and this is enough.

You can also investigate the growth via graphing:

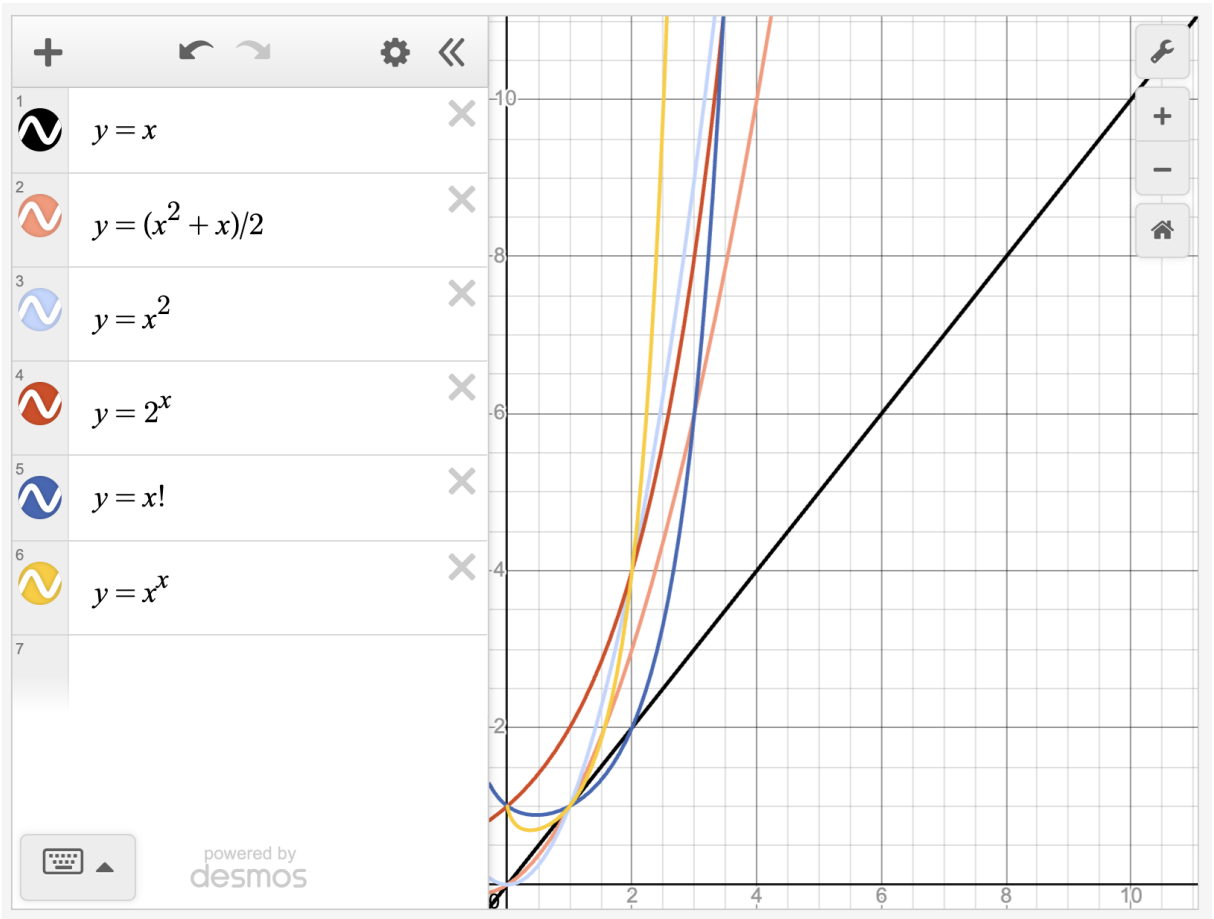

Figure 5: Graphs comparing growth rates $y = x$, $y = x(x+1)/2$, $y = x^2$, $y = 2^x$, $y = x!$, $y = x^x$. More detailed explanations can be found in the html version of this page.

What's the point?

So far, you have seen for all $n > 4$ (at least)

$$T_n < n^2 < 2^n < n! < n^n$$

In fact, some of these sequences are the cornerstone of **computational complexity**, which is the study of **how fast computer programs run**.

Computational complexity

Nowadays, computer programs underpin every facet of our lives. From shopping online, to gaming, to the financial markets, to transporting goods up and down the country, they really are everywhere.

The mathematical basis for computer programs is the study of **algorithms**. An algorithm takes in an input or inputs, performs a series of computational steps, and gives an output. It's critical to know exactly how long an algorithm takes, as this will affect how long a computer program takes to run. In real life, you would want algorithms that run the fastest and consumes the least resources. Typically, this is measured in the number of computational operations inside the algorithm, and amount of these computational operations tend to scale with the size of the input. In other words, the number of computational operations is a **sequence** in which you are trying to **minimize the growth rate**.

Measuring the amount of these computational operations is typically written in something called **big-O notation**, and this notation measures the **computational complexity** of the algorithm. For instance, if an algorithm has input of size n and needs (roughly) n^2 many computational operations to complete, you would say that the complexity of the algorithm is $O(n^2)$.

The smaller the growth rate of the sequence, the better the algorithm's performance. Typically, you would want an algorithm to complete in **polynomial time**; that is $O(n^k)$ for some positive number k. However, there are some algorithms (like the famous **travelling salesman problem**) that require at least **exponential time** $O(c^n)$ (for some $c > 1$) to complete.

P versus NP

Suppose that you are given a puzzle to solve, like a Sudoku. An algorithm could help you with this problem in one of two ways:

- given an attempt at a solution as an input, it could **check your solution** for you, or
- given the empty puzzle as an input, it could **solve the puzzle** for you.

Of these, the algorithm to check the solution should be far quicker than the algorithm to solve the problem - it has to do less work to check an answer than to find one itself. It could be that both algorithms have the same computational complexity, or they have different computational complexities.

- Problems with algorithms that **solve that problem** in polynomial time are said to be in the computational complexity class **P**.
- Problems with algorithms that can **check a solution** in polynomial time are said to be in the computational complexity class **NP**.

Here are some examples of problems in each complexity class:

P

- Testing to see if a given number is prime or not
- Calculating the highest common factor of any two integers
- Sorting a set of inputs into an ordered list
- Wikipedia racing; going from one Wikipedia page to another in the fewest amount of clicks

NP

- Finding all prime factors of a given integer
- The travelling salesman problem
- Finding a group of people on Facebook who are all friends with each other
- Games and puzzles including Sudoku, Battleships, Tetris, and Super Mario Bros
- Creating a perfect Bitcoin block

However, because humanity hasn't yet found algorithms that solve the problems in **NP** in polynomial time, it **doesn't mean they don't exist**. (You can compare this to the existence of extra-terrestrial life, or of the Loch Ness monster.) This is one of the most significant open problems in theoretical computer science, the **P vs NP** problem:

> If the solution to a problem can be checked in polynomial time, must the problem be solvable in polynomial time?

If the answer is **yes** (implying that **P = NP**), then humanity will have some significant issues to overcome. For instance, internet security relies on the fact that 'finding all prime factors' is **not** in **P** - so if it is, then a whole new security protocol will need to be invented. If the answer is **no**, then there will always be hard bounds on what computers can achieve.

A solution either way will net you a cool million dollars and make you famous for ever.

[Note: All of this framework is for **regular computers** - however, due to the fundamentally different nature of **quantum computers**, this problem might even be moot in a few years time...]

Version history

v1.0: initial version created 06/26 by tdhc, originally for presentation 1 of 4 for Sutton Trust Summer School 2026.