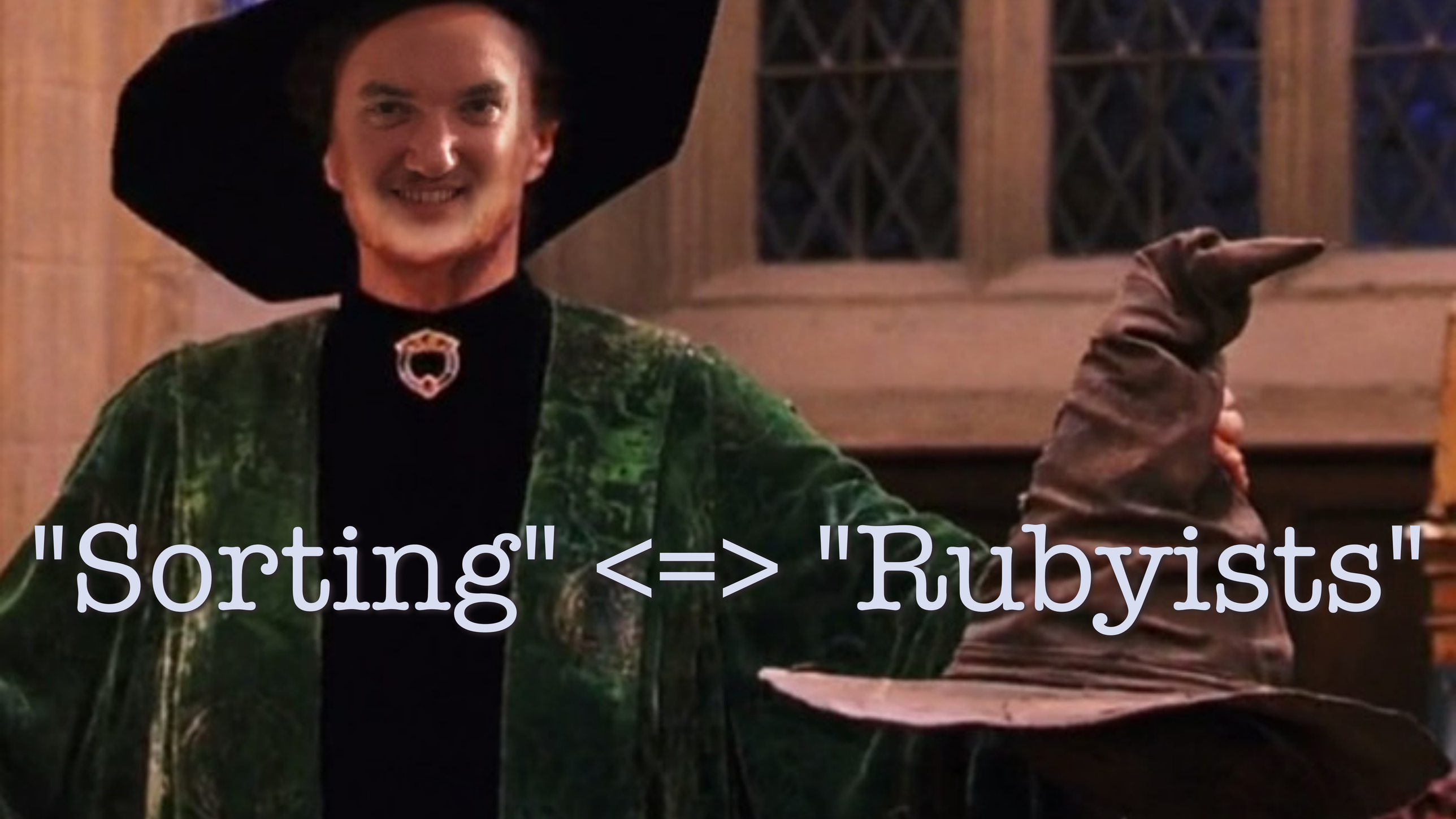
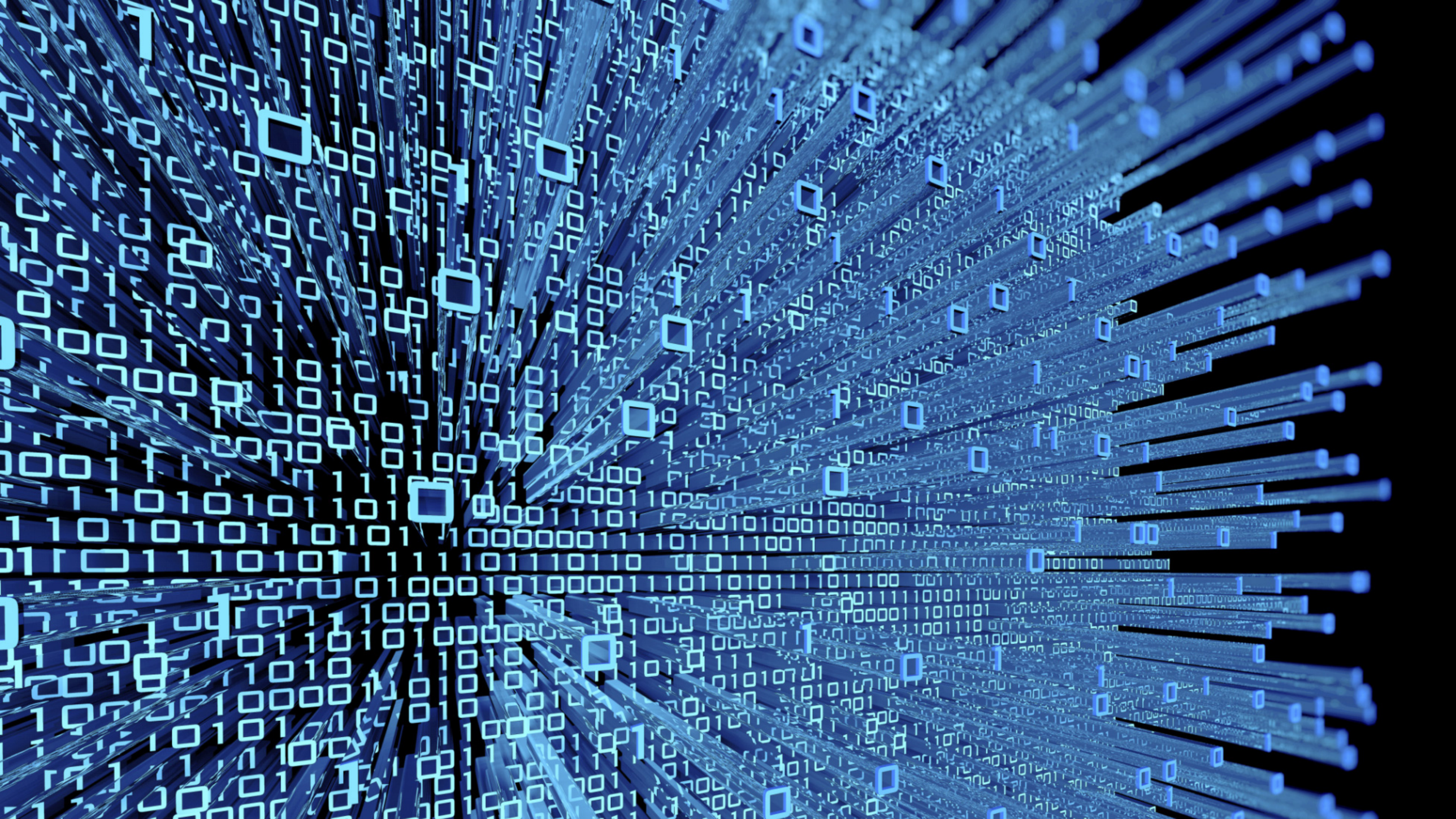
THE GOBBLING GLUTTONS

ONCE UPON A TIME, WALDO EMBARKED UPON A FANTASTIC JOURNEY. FIRST, AMONG A THRONG OF GOBBLING GLUTTONS, HE MET WIZARD WHITEBEARD, WHO COMMANDED HIM TO FIND A SCROLL AND THEN TO FIND ANOTHER AT EVERY STAGE OF HIS JOURNEY. FOR WHEN HE HAD FOUND 12 SCROLLS, HE WOULD UNDERSTAND THE TRUTH ABOUT HIMSELF.

IN EVERY PICTURE FIND WALDO, WOOF (BUT ALL YOU CAN SEE IS HIS TAIL), WENDA, WIZARD WHITEBEARD, ODLAW, AND THE SCROLL. THEN FIND WALDO'S KEY, WOOF'S BONE (IN THIS SCENE IT'S THE BONE THAT'S NEAREST TO HIS TAIL), WENDA'S CAMERA, AND ODLAW'S BINOCULARS.

THERE ARE ALSO 25 WALDO-WATCHERS, EACH OF WHOM APPEARS ONLY ONCE SOMEWHERE IN THE FOLLOWING 12 PICTURES. AND ONE MORE THING! CAN YOU FIND ANOTHER CHARACTER, NOT SHOWN BELOW, WHO APPEARS ONCE IN EVERY PICTURE EXCEPT THE LAST?


"Sorting" <=> "Rubyists"



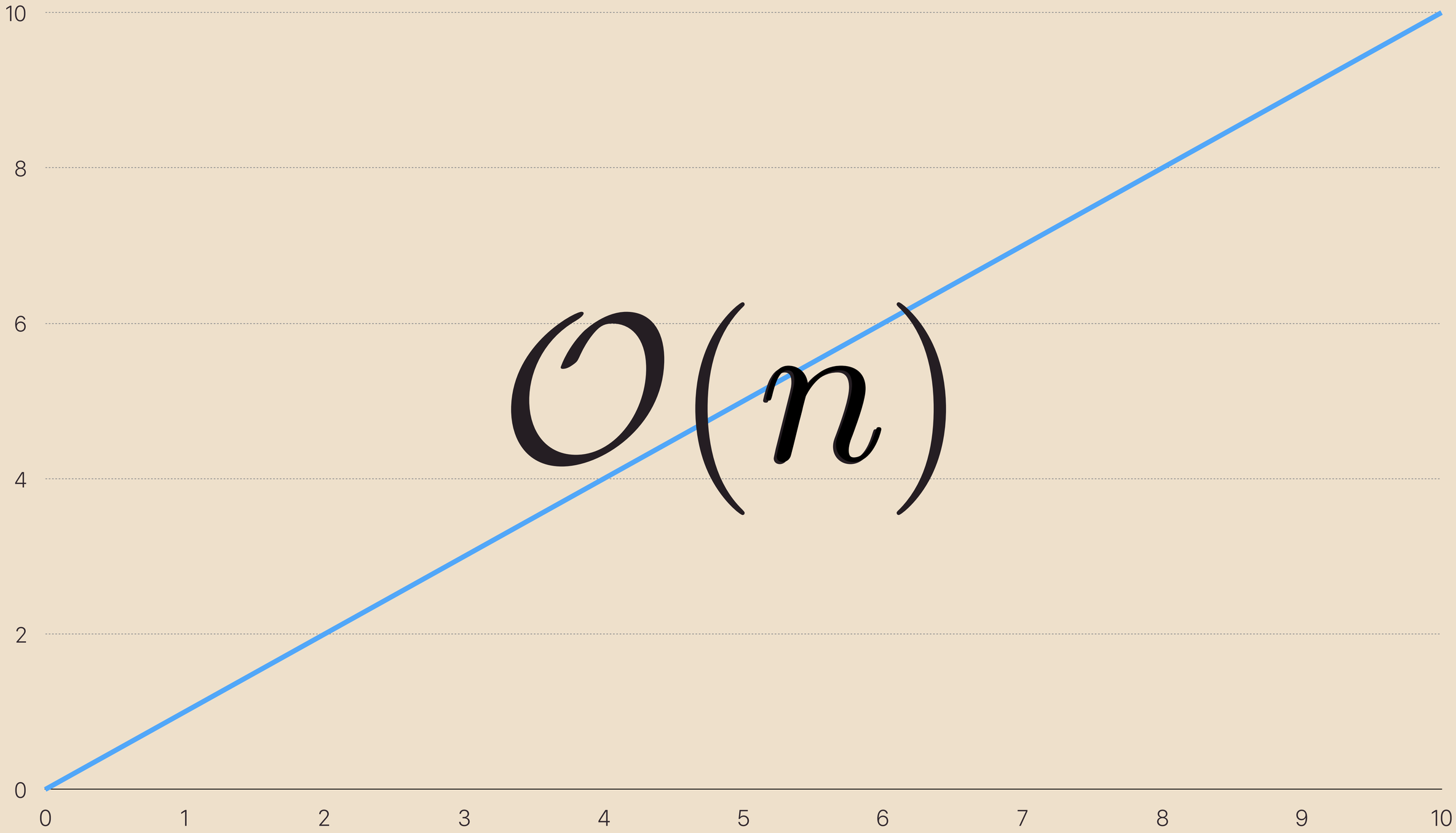
```
/*
 *  call-seq:
 *     ary.sort!                    -> ary
 *     ary.sort! { |a, b| block }  -> ary
 *
 *  Sorts +self+ in place.
 *
 *  Comparisons for the sort will be done using the <code><=></code> operator
 *  or using an optional code block.
 *
 *  The block must implement a comparison between +a+ and +b+ and return
 *  an integer less than 0 when +b+ follows +a+, +0+ when +a+ and +b+
 *  are equivalent, or an integer greater than 0 when +a+ follows +b+.
 *
 *  The result is not guaranteed to be stable.  When the comparison of two
 *  elements returns +0+, the order of the elements is unpredictable.
 *
 *     a = [ "d", "a", "e", "c", "b" ]
 *     a.sort!                    #=> ["a", "b", "c", "d", "e"]
 *     a.sort! { |x,y| y <=> x }  #=> ["e", "d", "c", "b", "a"]
 *
 *  See also Enumerable#sort_by.
 */

VALUE
```

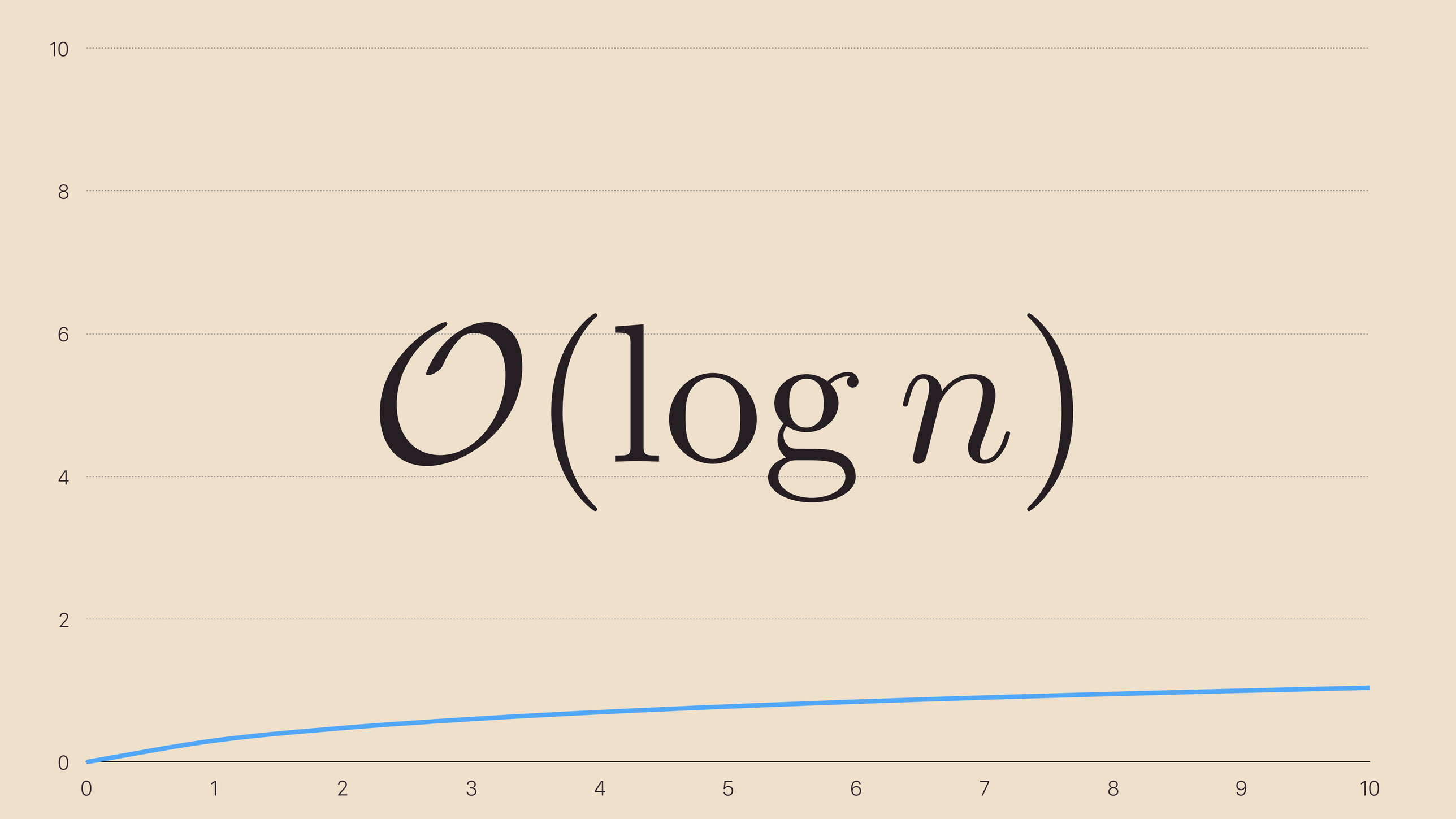
```
/*
 *  call-seq:
 *     ary.sort!                    -> ary
 *     ary.sort! { |a, b| block }  -> ary
 *
 *  Sorts +self+ in place.
 *
 *  Comparisons for the sort will be done using the <code><=></code> operator
 *  or using an optional code block.
 *
 *  The block must implement a comparison between +a+ and +b+ and return
 *  an integer less than 0 when +b+ follows +a+, +0+ when +a+ and +b+
 *  are equivalent, or an integer greater than 0 when +a+ follows +b+.
 *
 *  The result is not guaranteed to be stable.  When the comparison of two
 *  elements returns +0+, the order of the elements is unpredictable.
 *
 *     a = [ "d", "a", "e", "c", "b" ]
 *     a.sort!                    #=> ["a", "b", "c", "d", "e"]
 *     a.sort! { |x,y| y <=> x }  #=> ["e", "d", "c", "b", "a"]
 *
 *  See also Enumerable#sort_by.
 */

VALUE
```

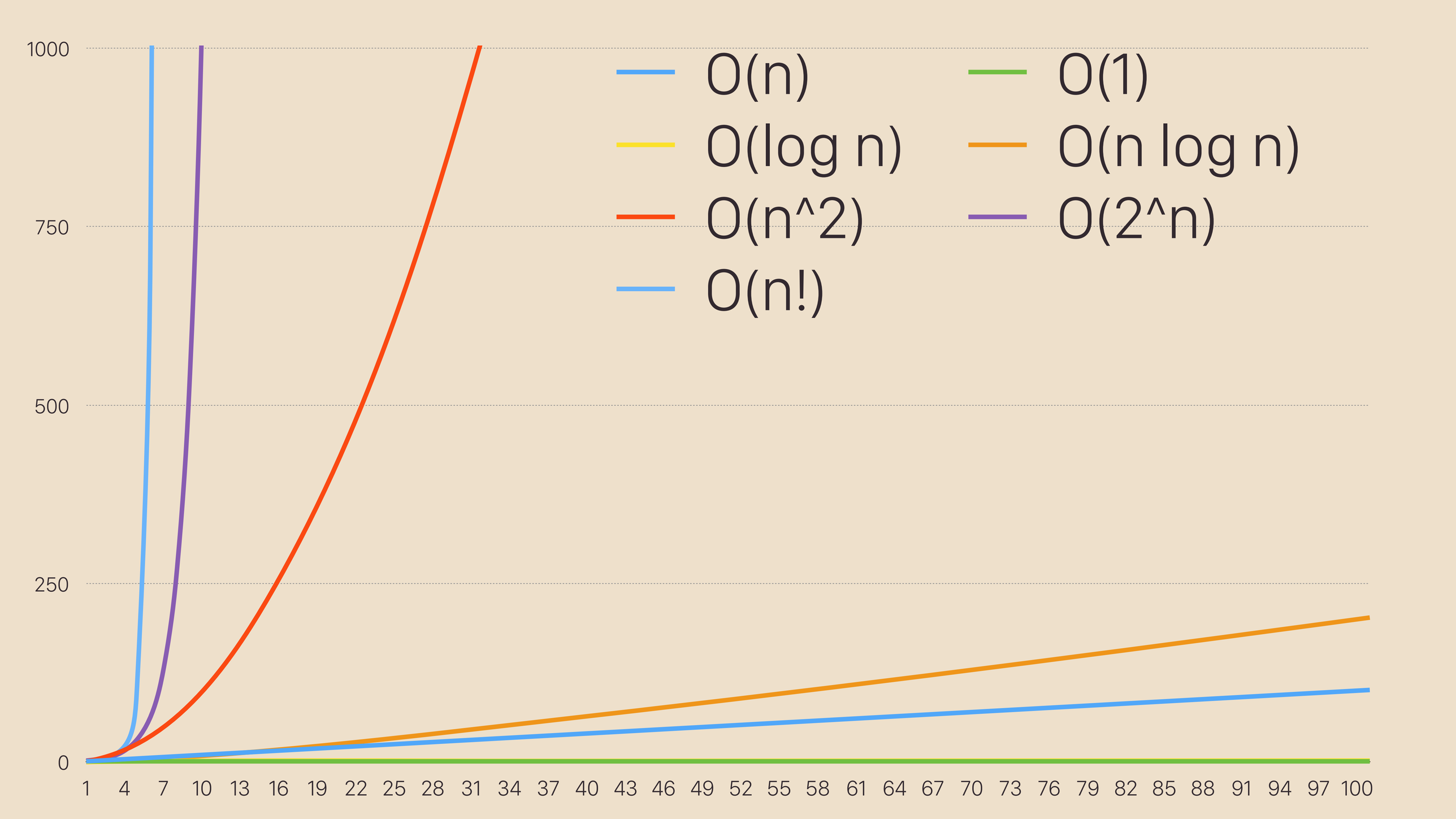
Algorithms

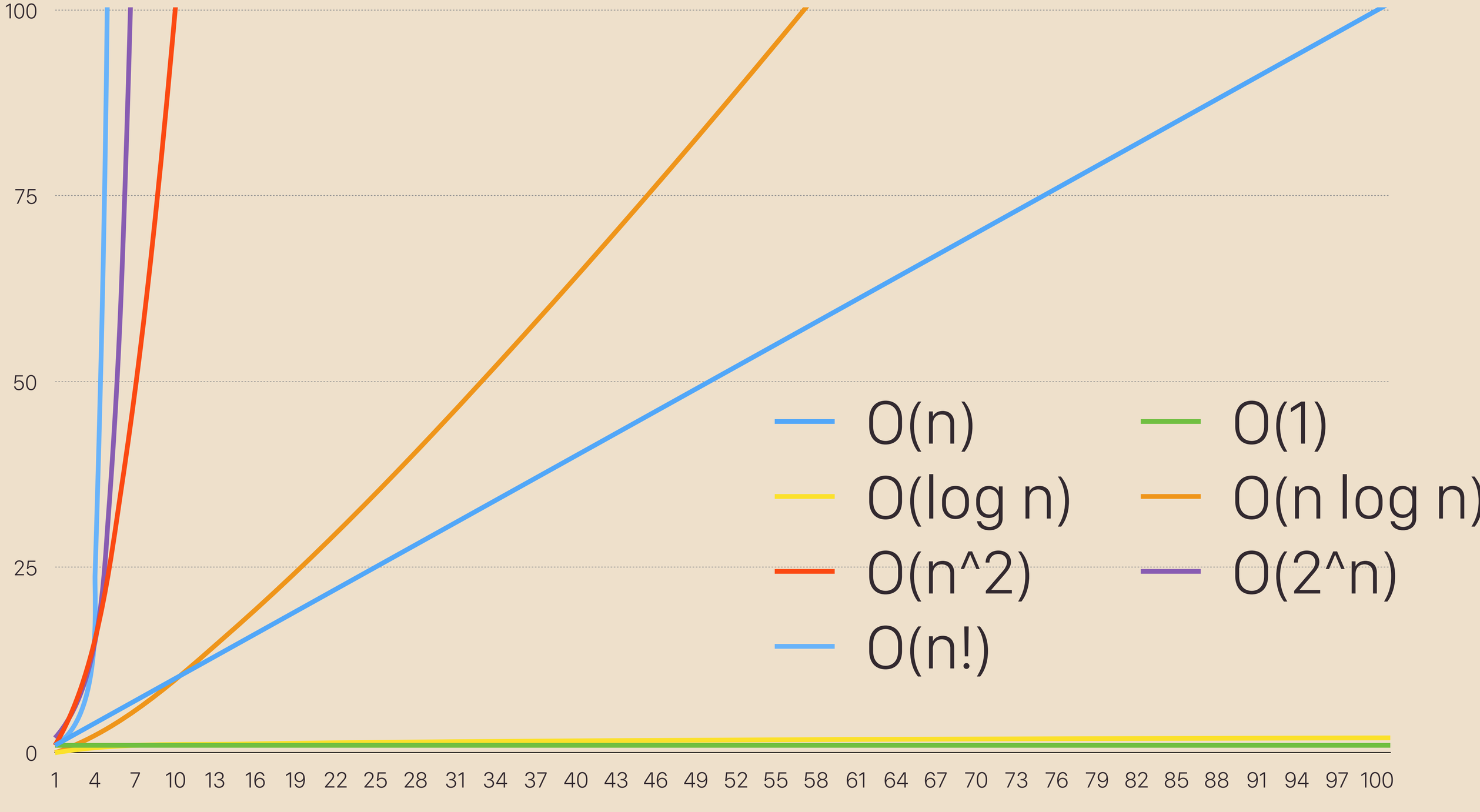
A series of step-by-step instructions

Big O


10
8
6
4
2
0
0
1
2
3
4
5
6
7
8
9
10
$O(n)$


10
8
6
4
2
0
0
1
2
3
4
5
6
7
8
9
10
$\mathcal{O}(\log n)$


1000
750
500
250
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100
O(n)
O(log n)
O(n^2)
O(n!)
O(1)
O(n log n)
O(2^n)


100
75
50
25
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100
O(n)
O(log n)
O(n^2)
O(n!)
O(1)
O(n log n)
O(2^n)

When do you update a user record?

When do you update a user record?

$$\mathcal{O}(n \log n)$$

If you understood everything

If you understood
everything

I'm really $\mathcal{O}(n)$ it today

How do you debug an app?

How do you debug an app?

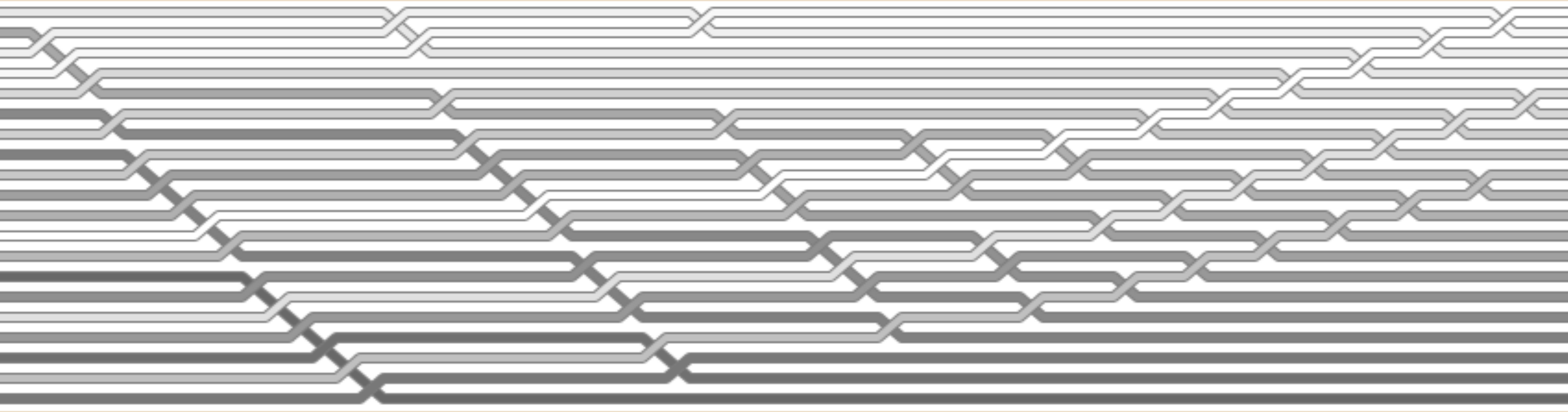
Start $\mathcal{O}(\log n)$

What happens when you buy something?

What happens when you buy something?

You $\mathcal{O}(n)$ it

Big Ω



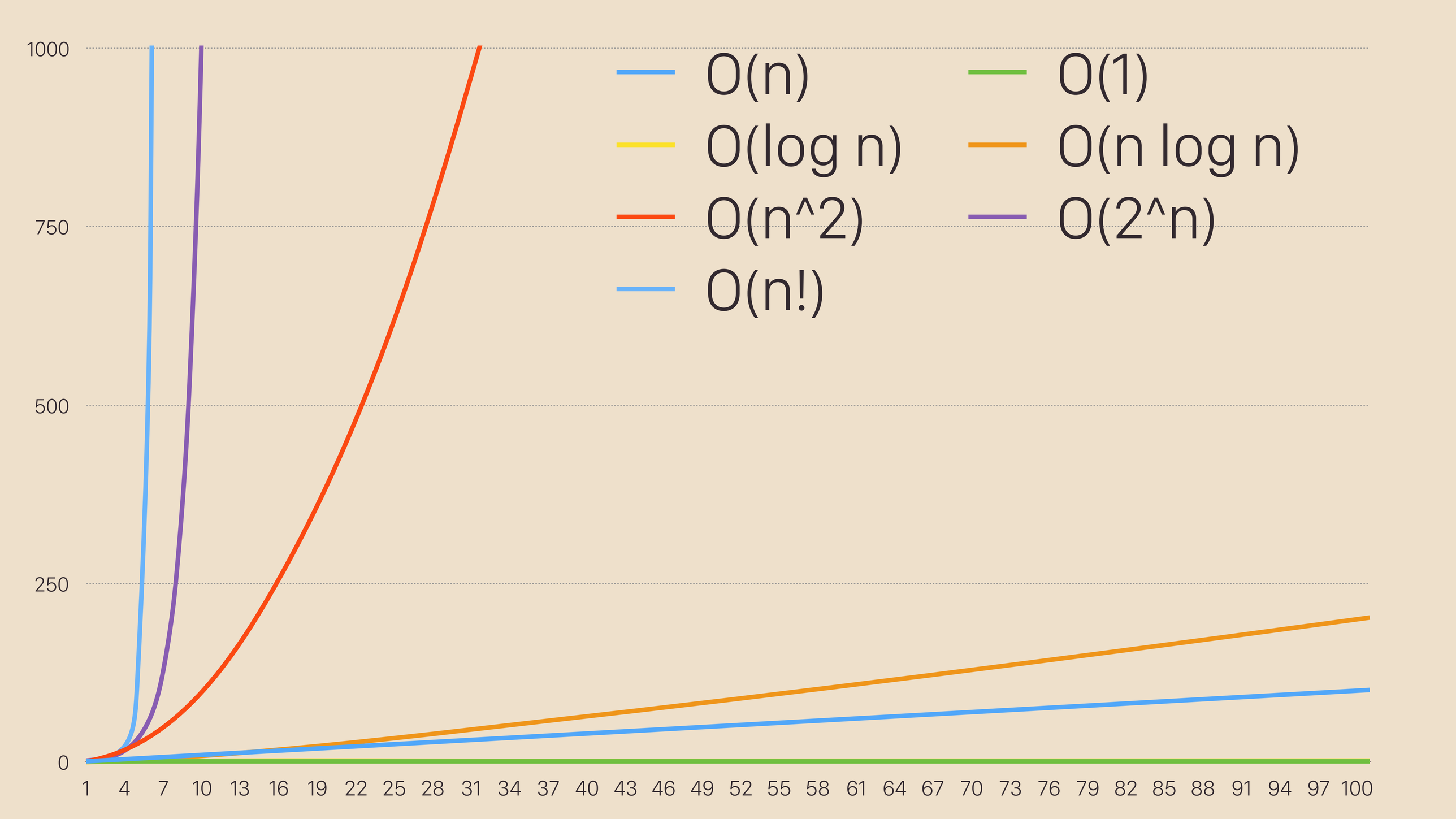
Bubble Sort

$$O(n^2)$$

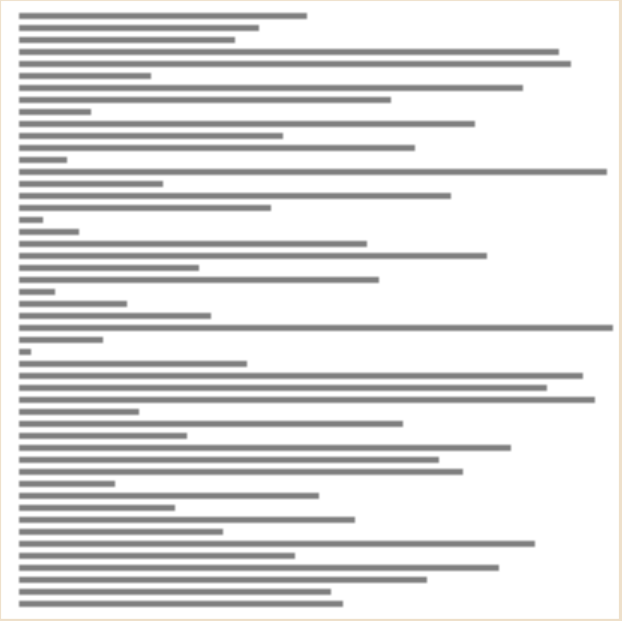
$$\mathcal{O}(\texttt{set.length}^2)$$

$$O(n^2)$$

$$\Omega(n)$$

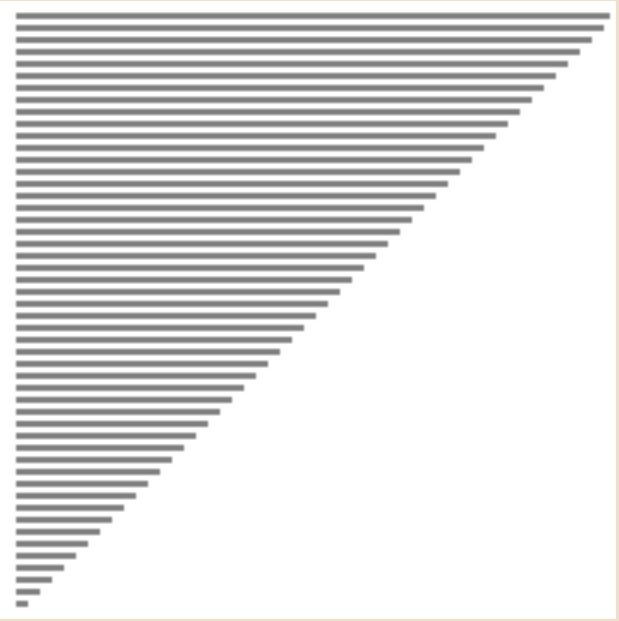

1000
750
500
250
0
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100
O(n)
O(log n)
O(n^2)
O(n!)
O(1)
O(n log n)
O(2^n)

Random



Nearly Sorted

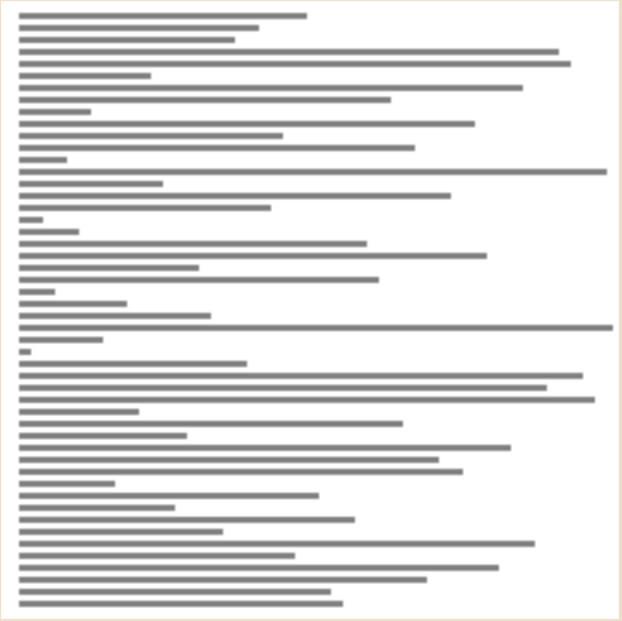
Reversed



Few Unique

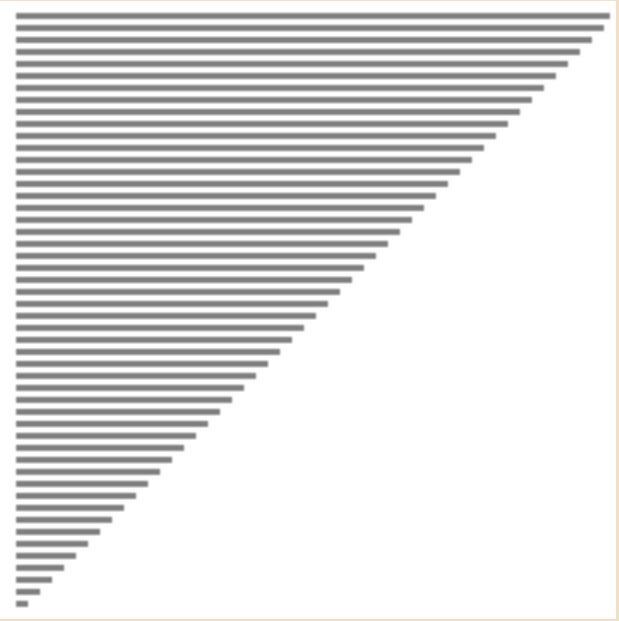
https://www.toptal.com/developers/sorting-algorithms/bubble-sort

Random



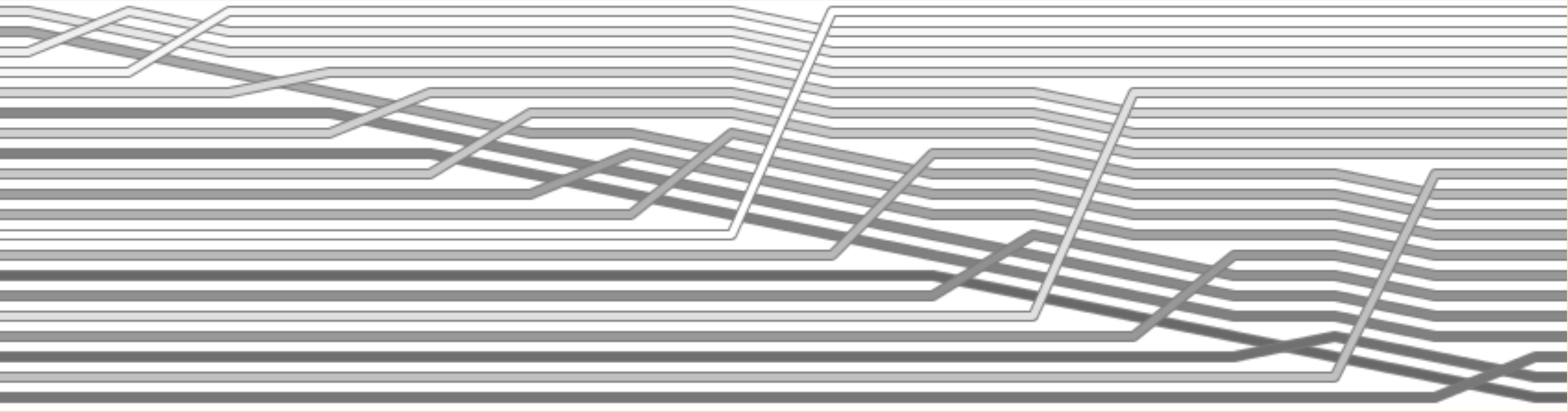
Nearly Sorted

Reversed

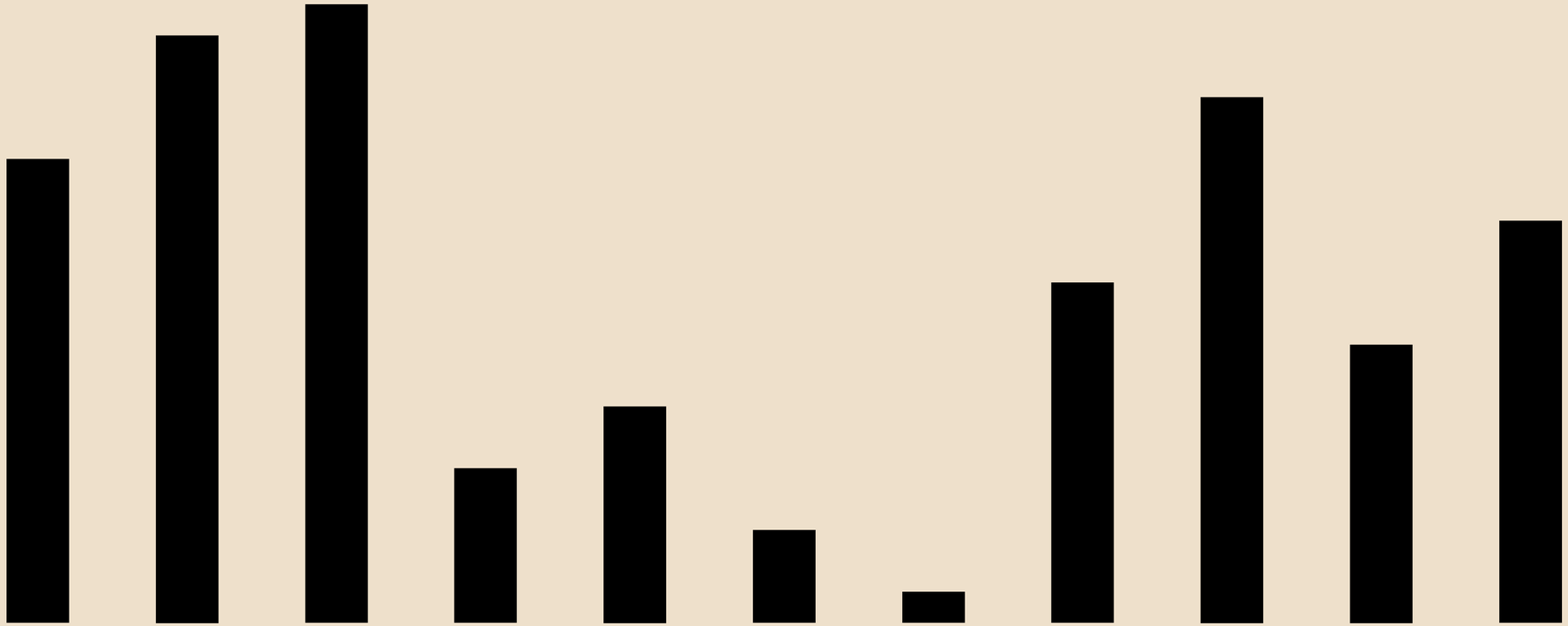


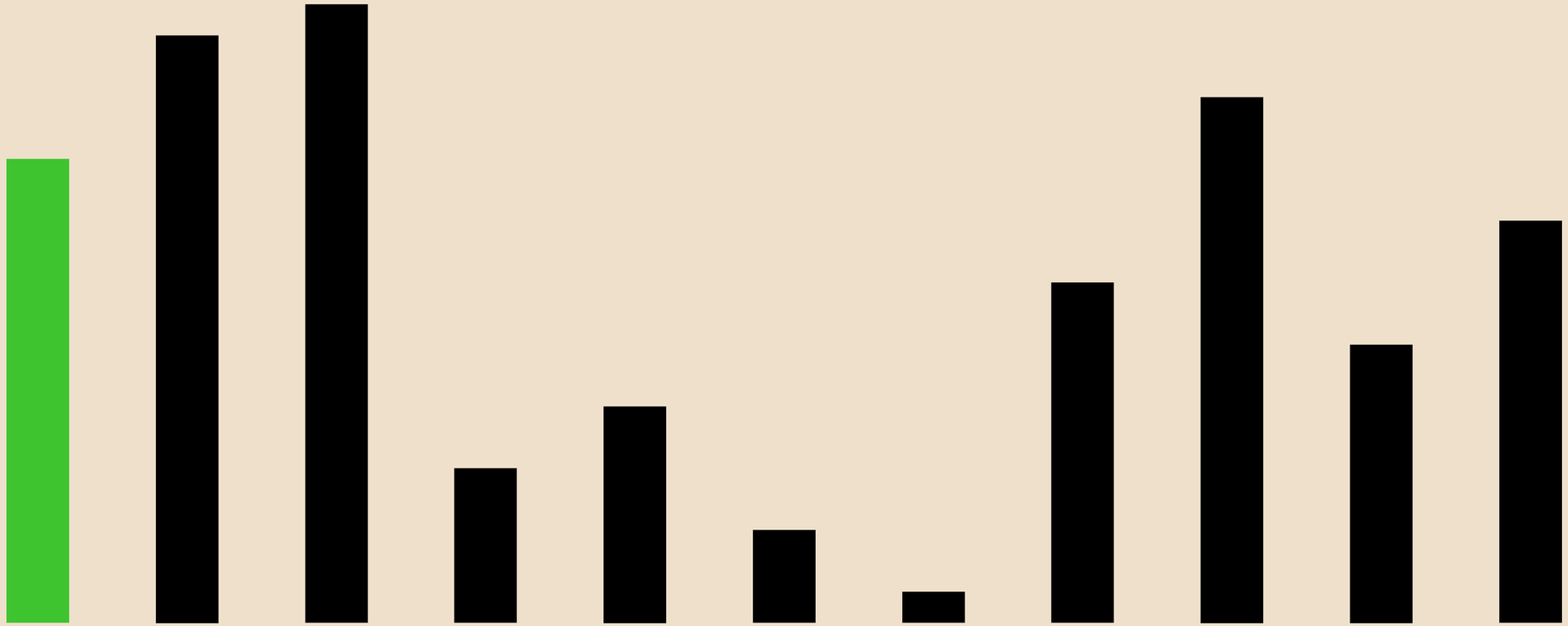
Few Unique

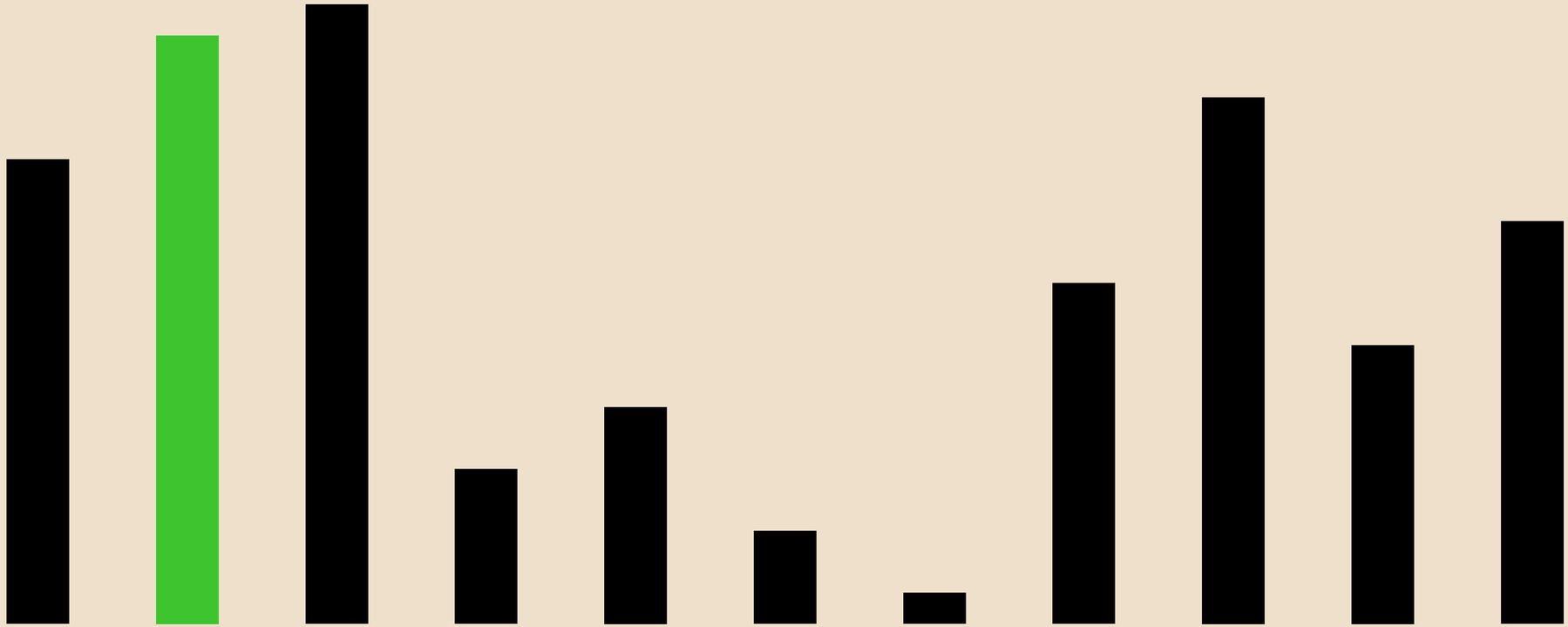
https://www.toptal.com/developers/sorting-algorithms/bubble-sort

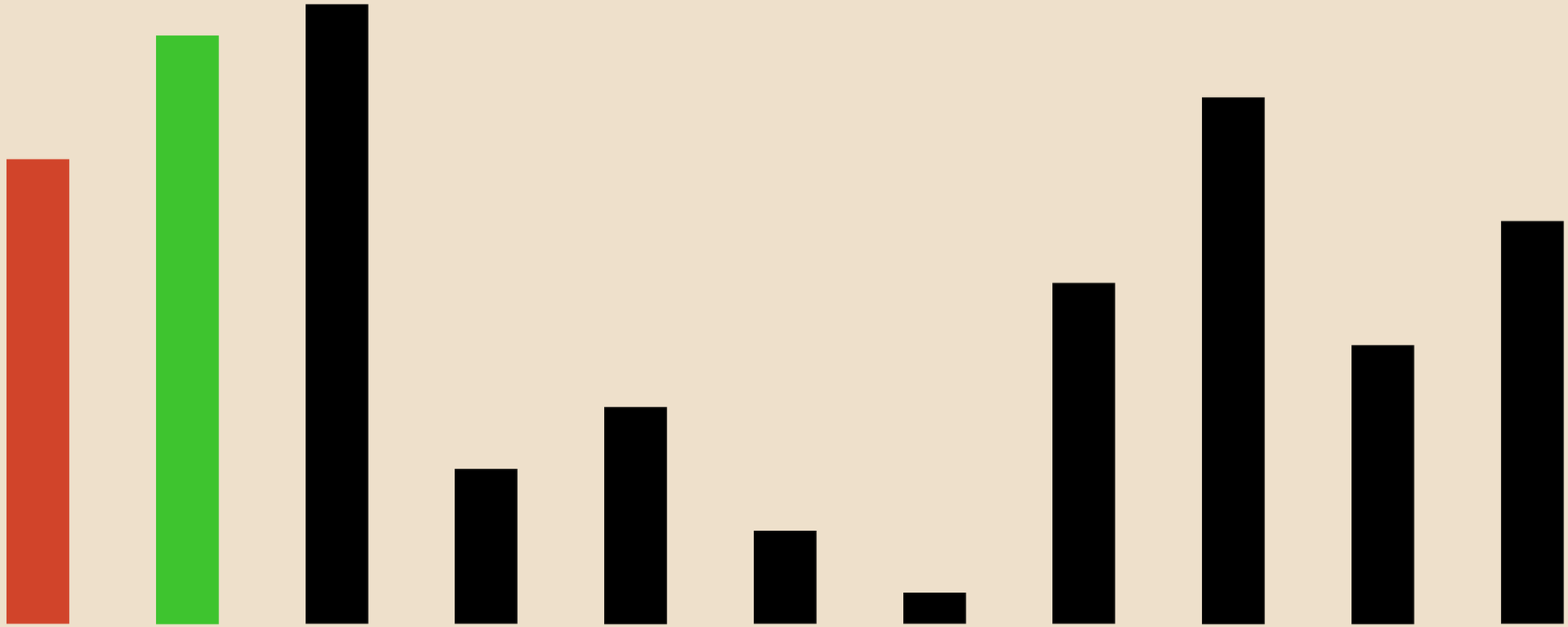


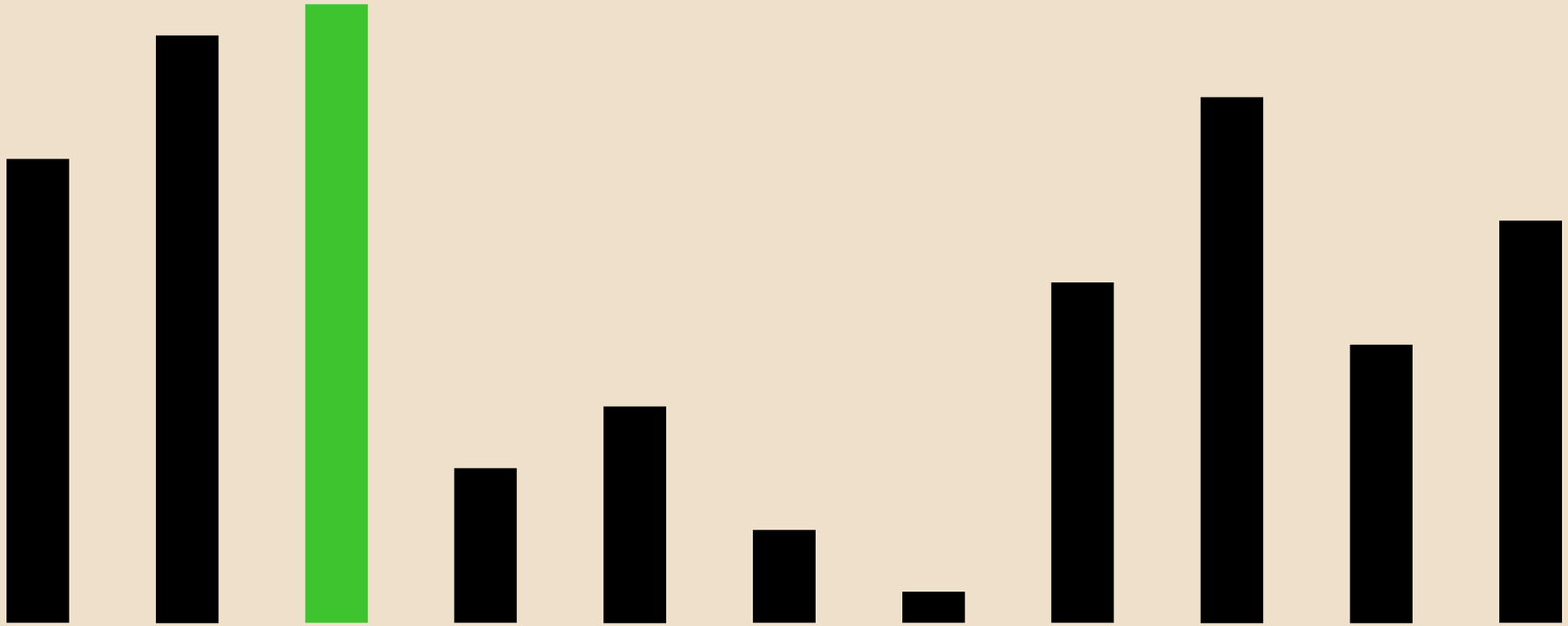
Insertion Sort

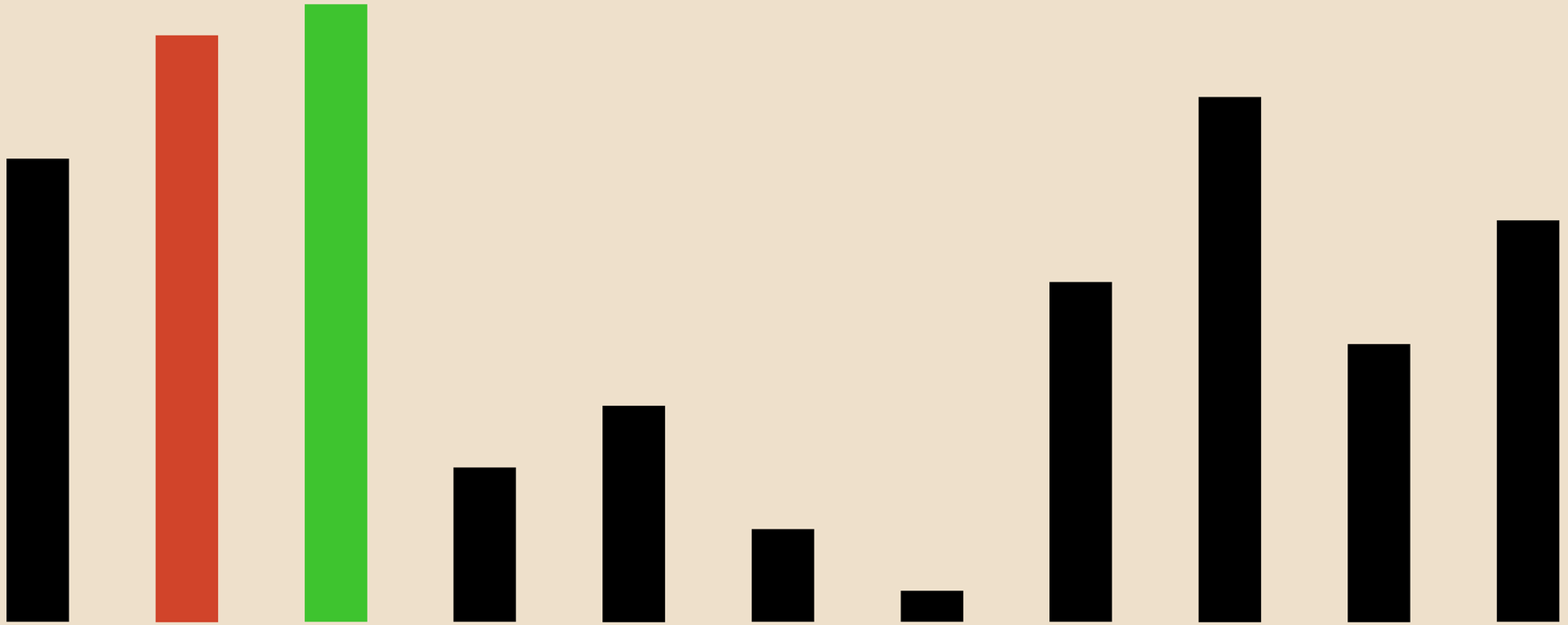


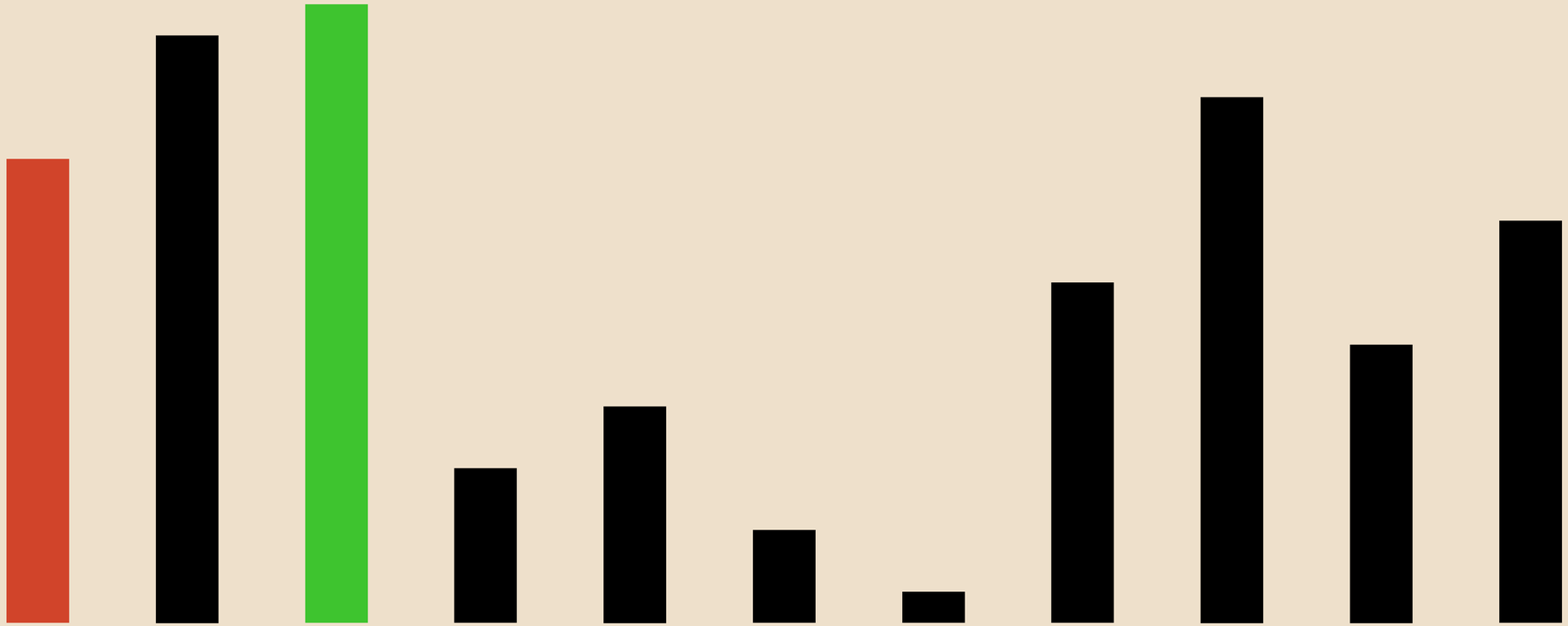


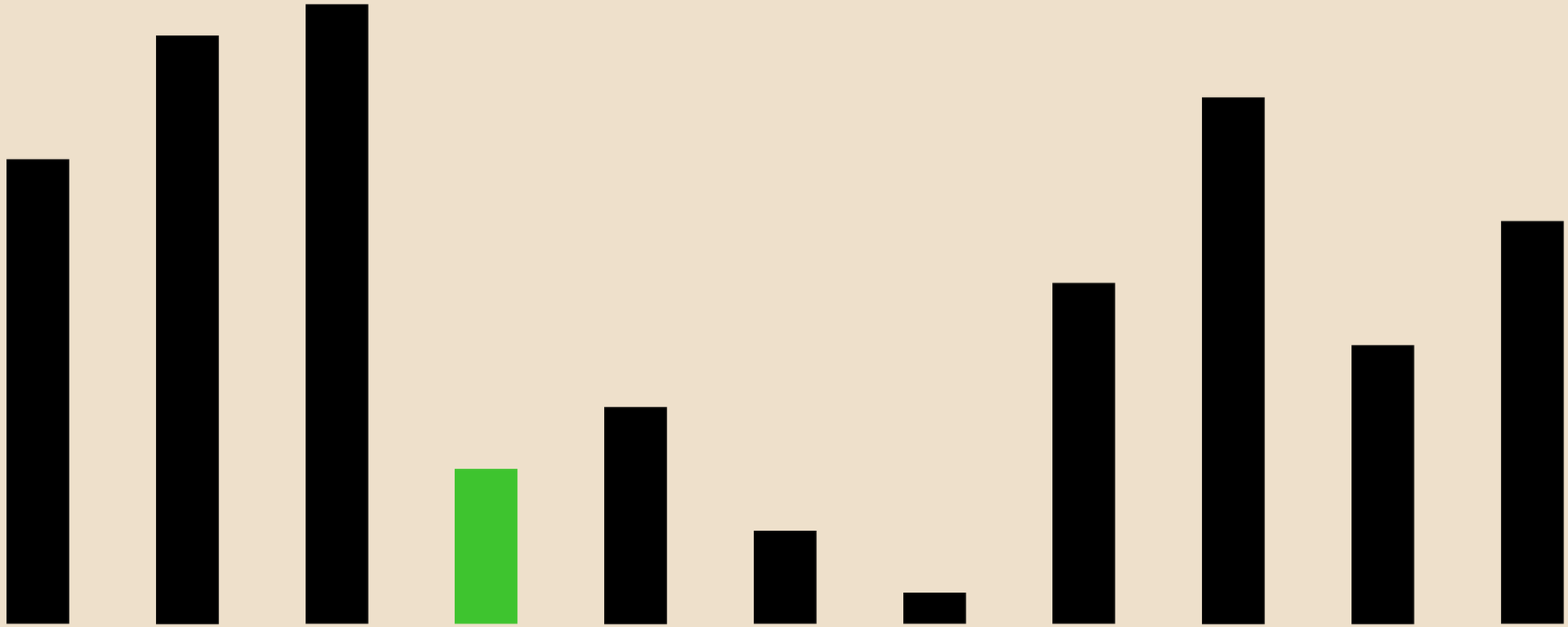


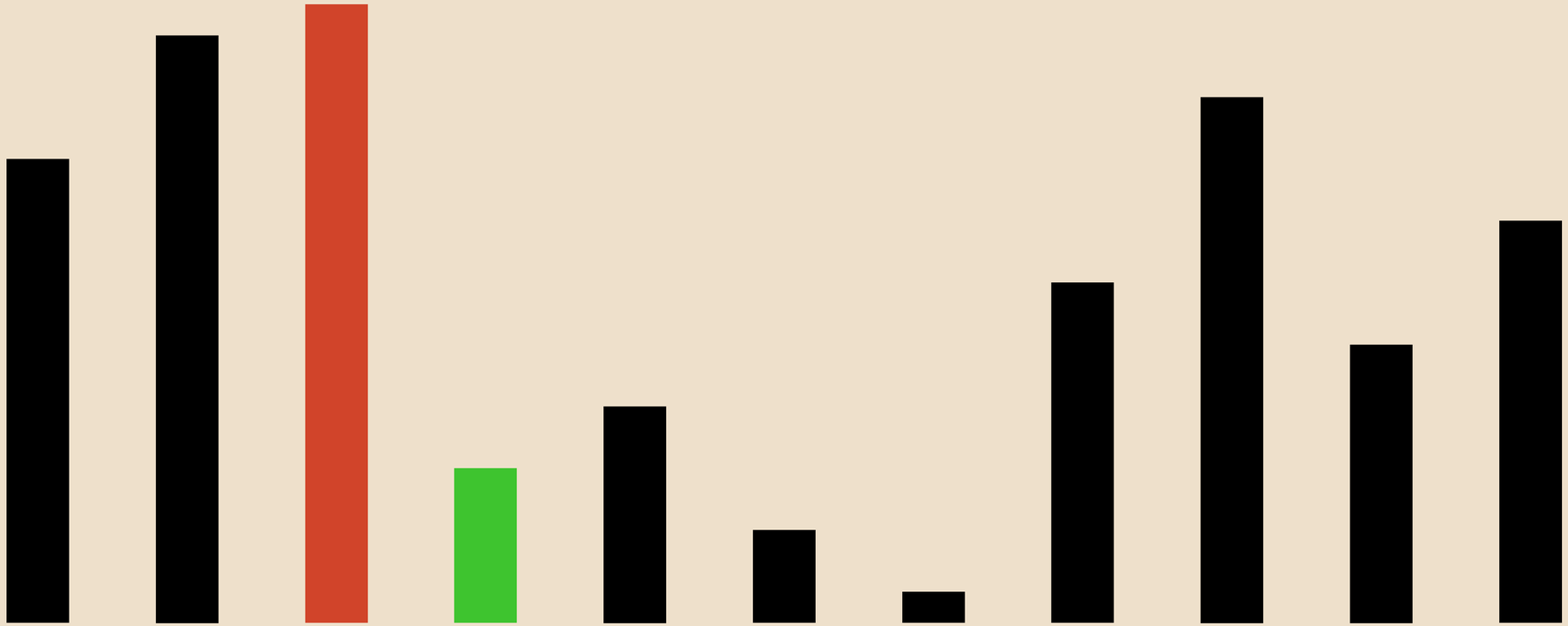


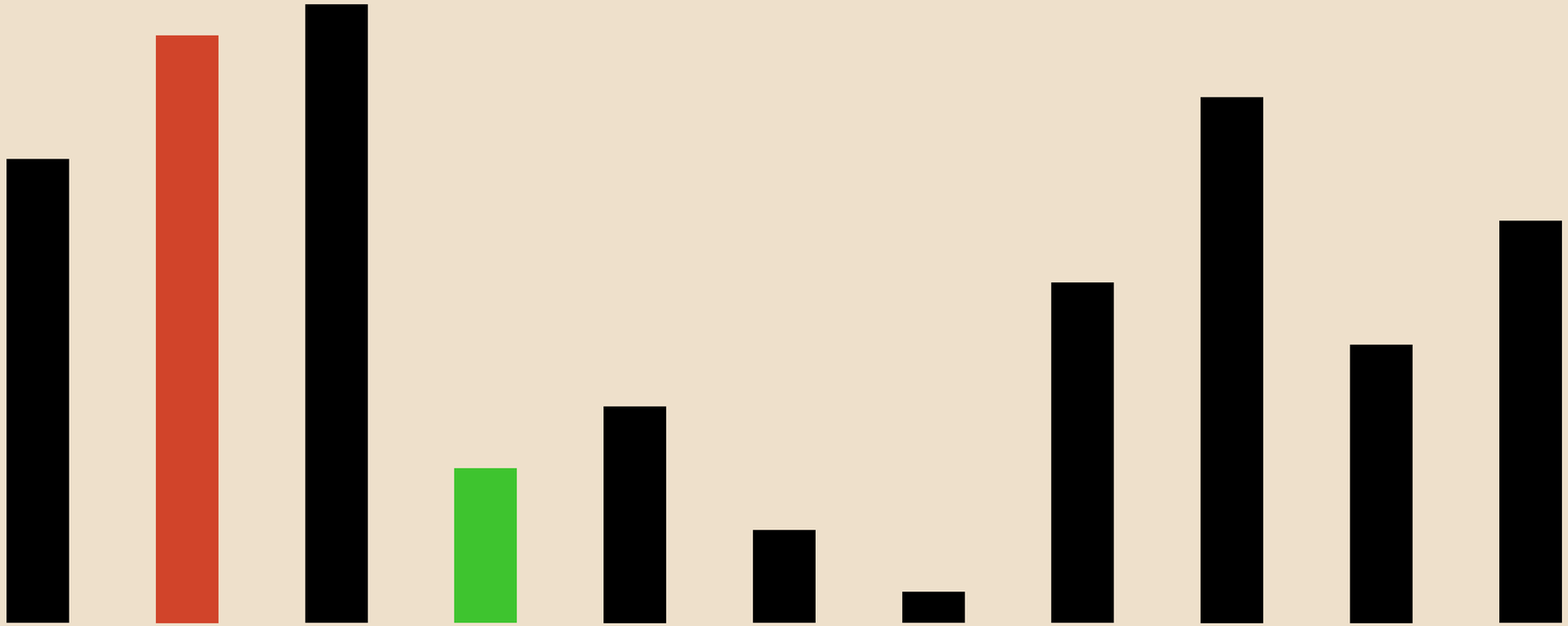


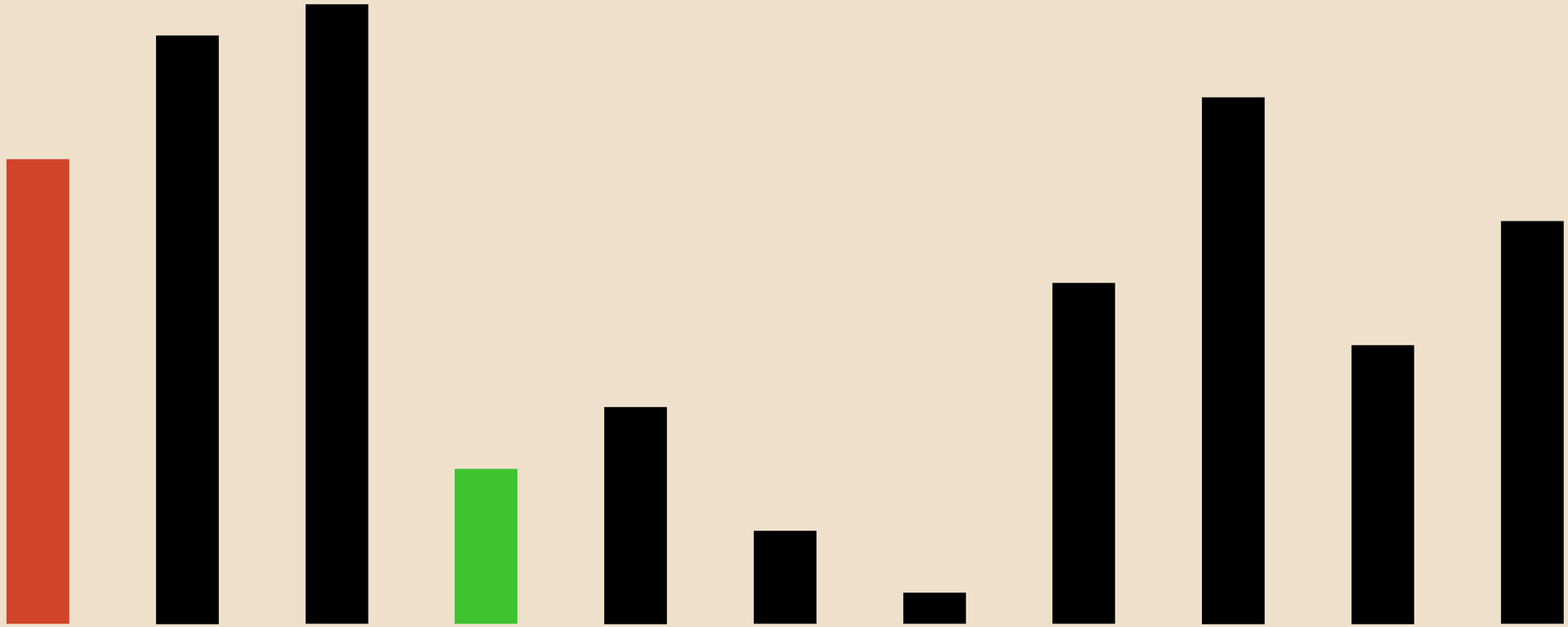


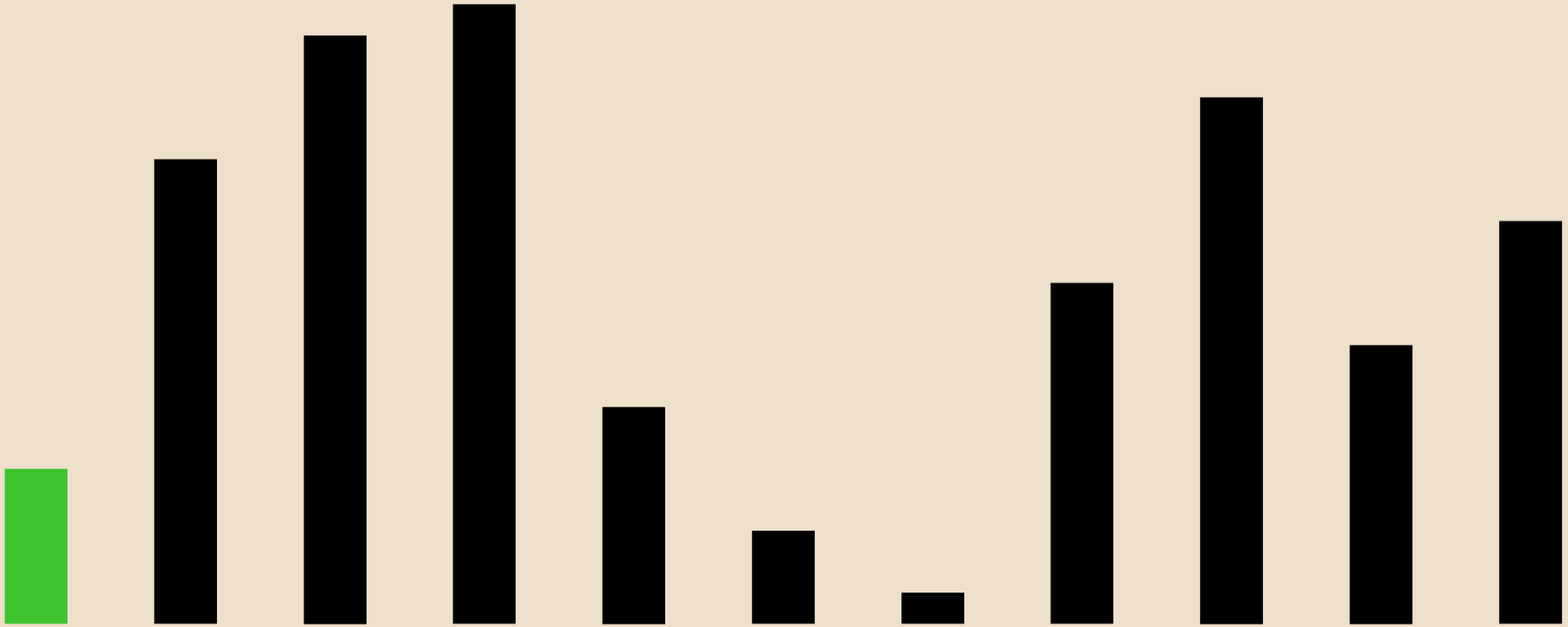


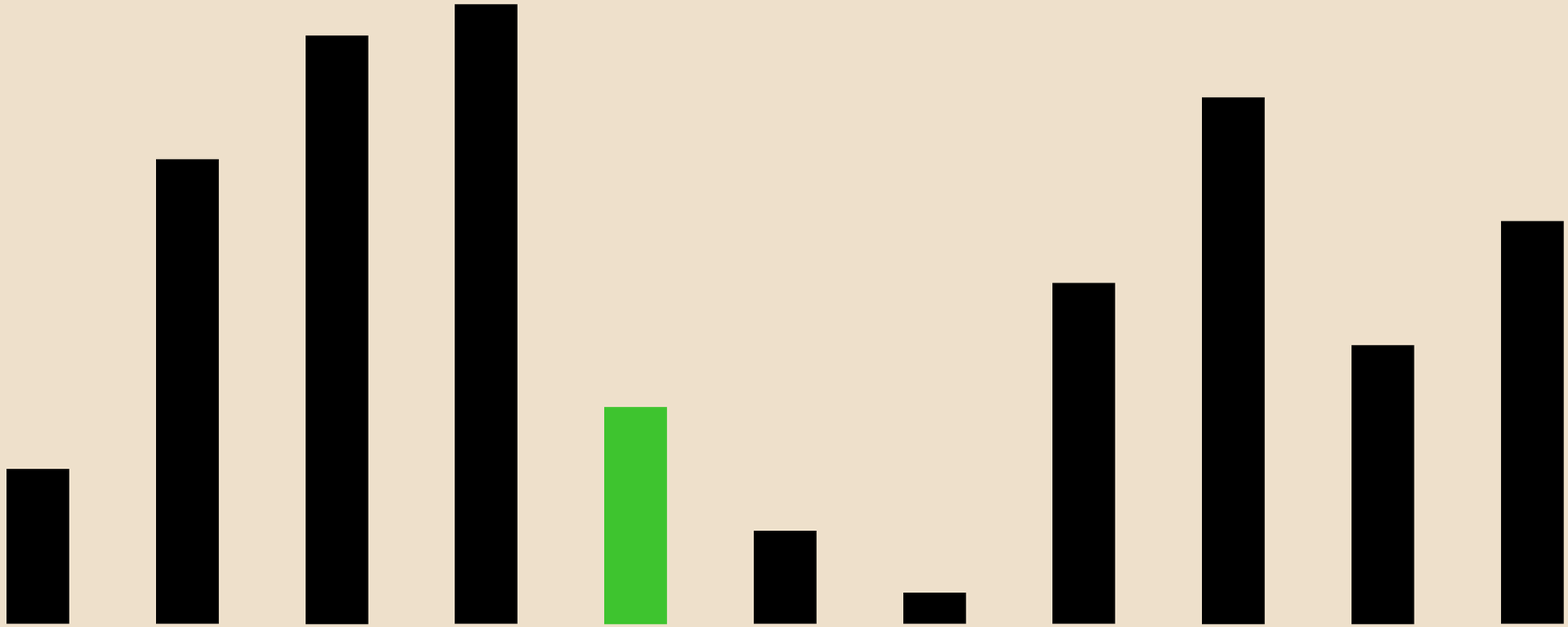


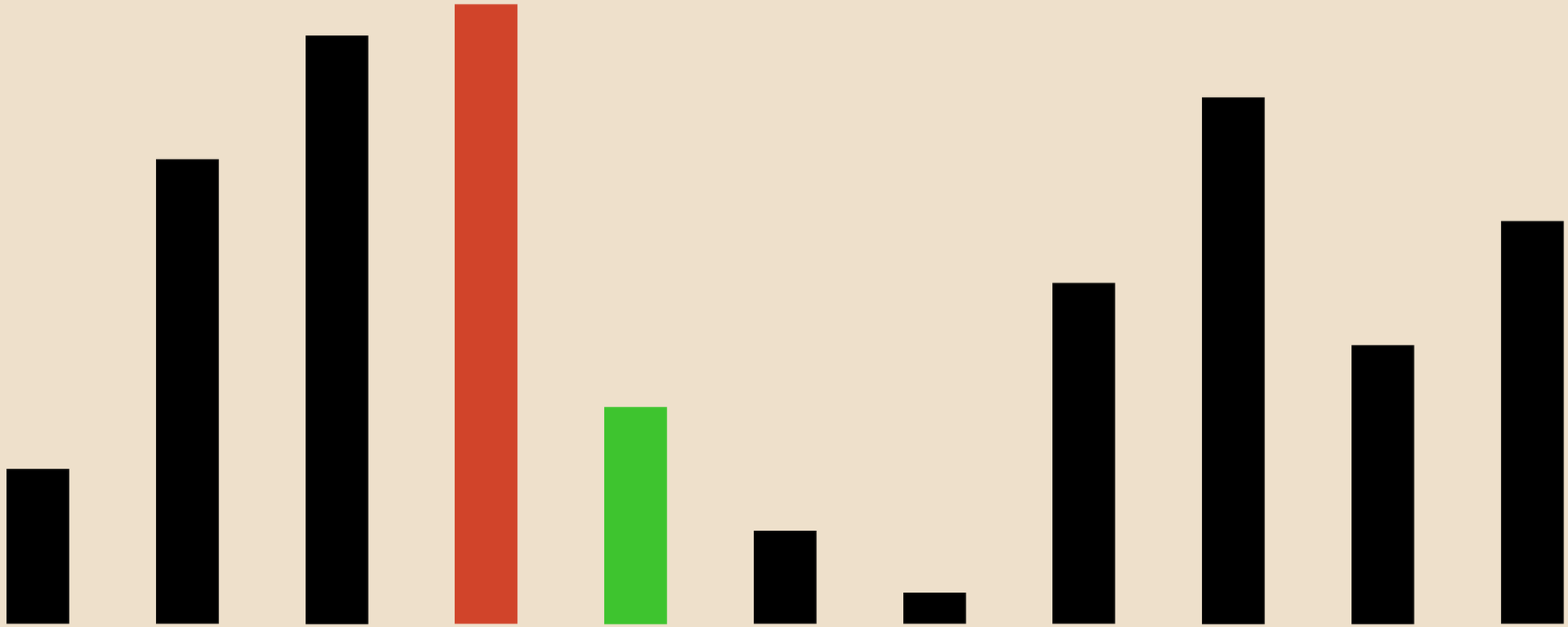


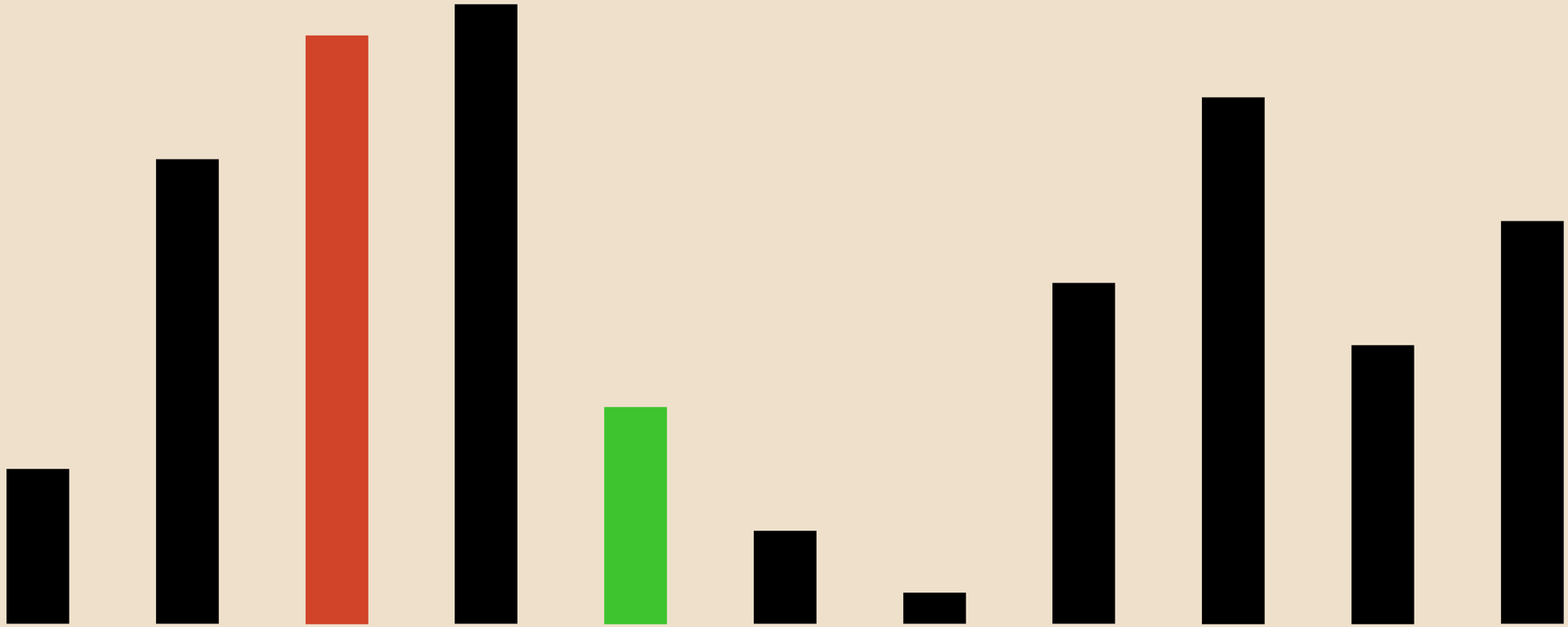


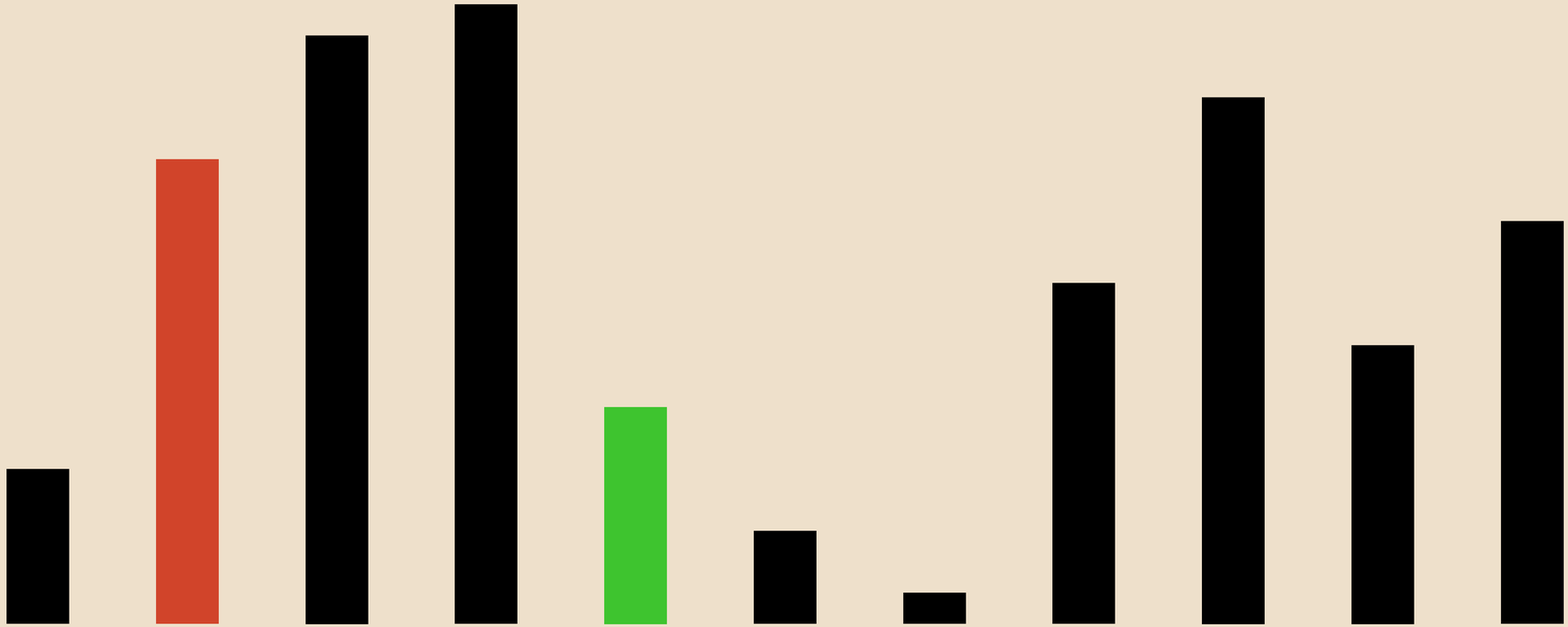


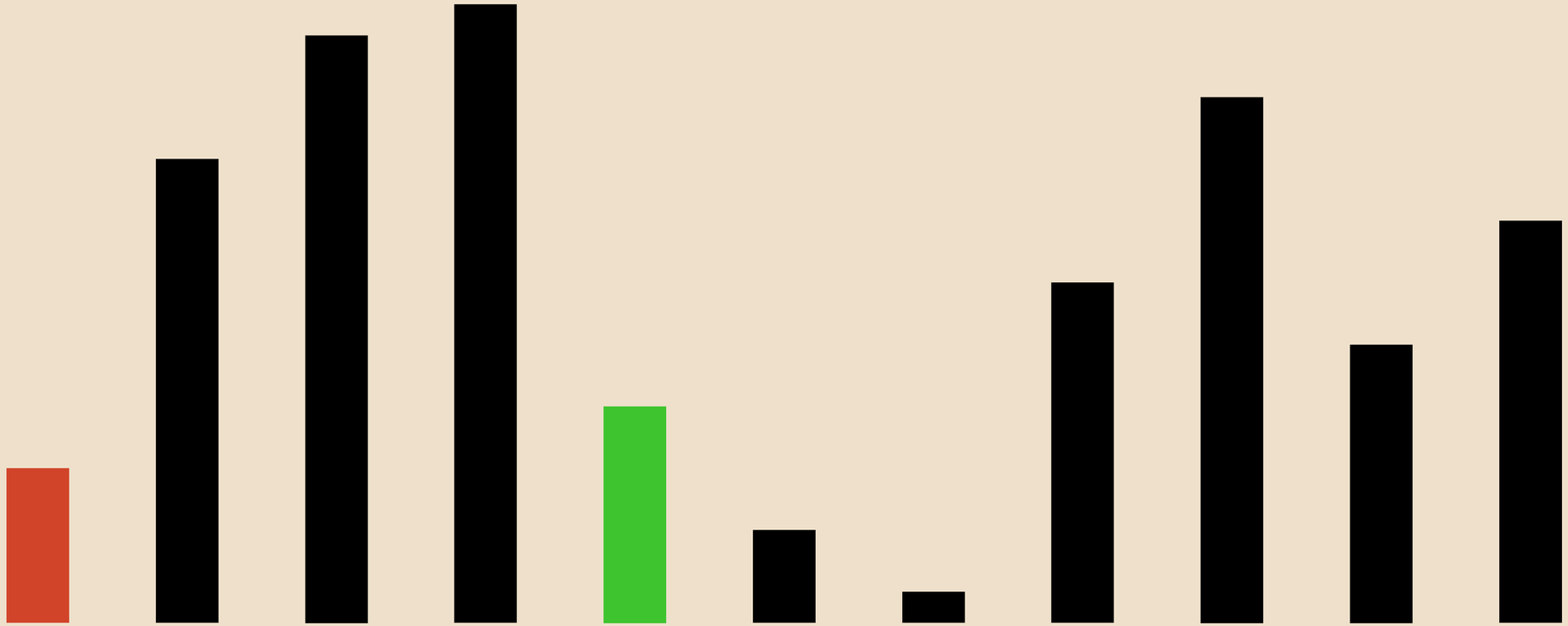


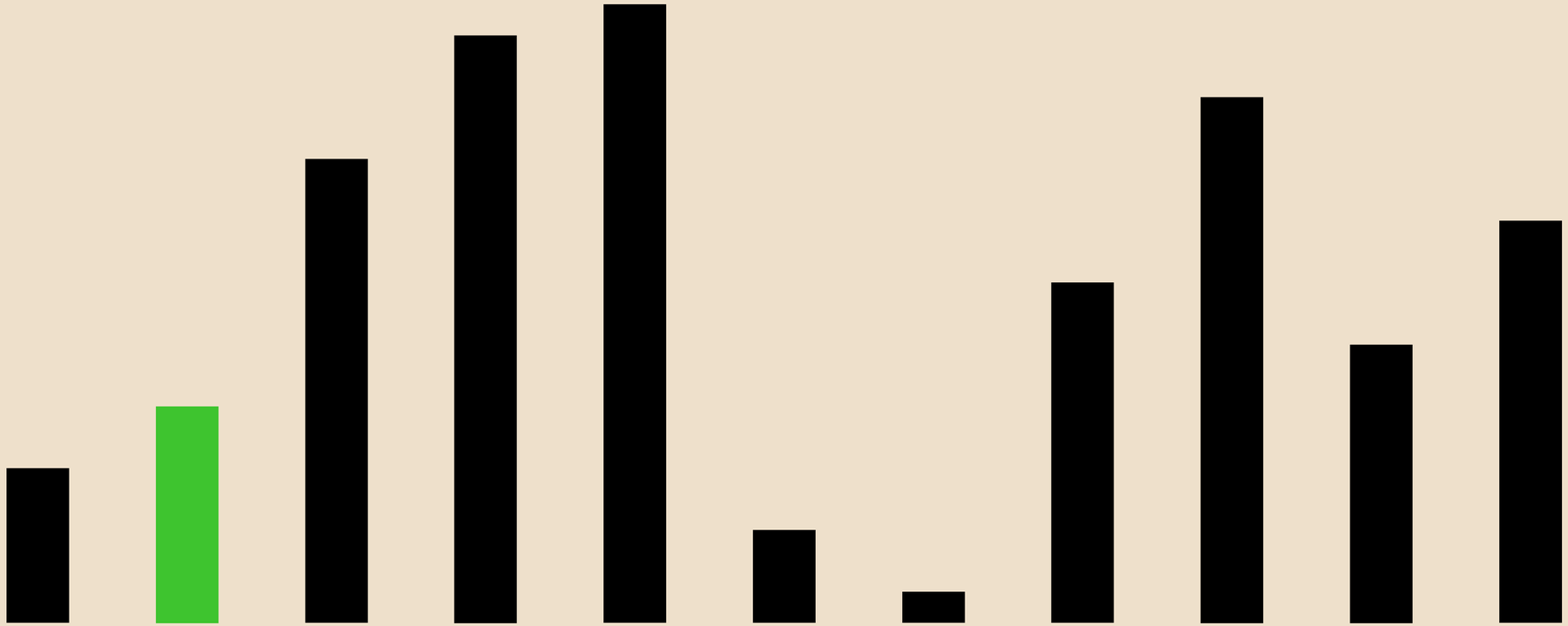


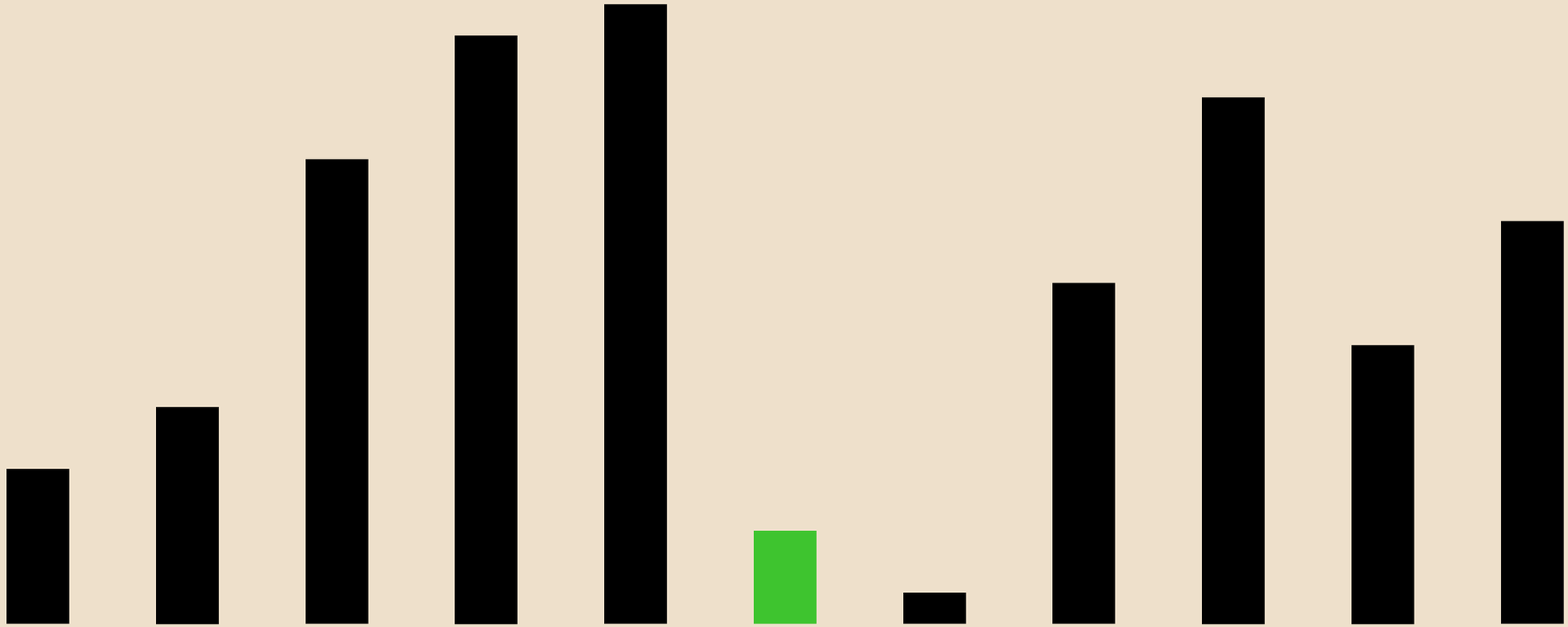


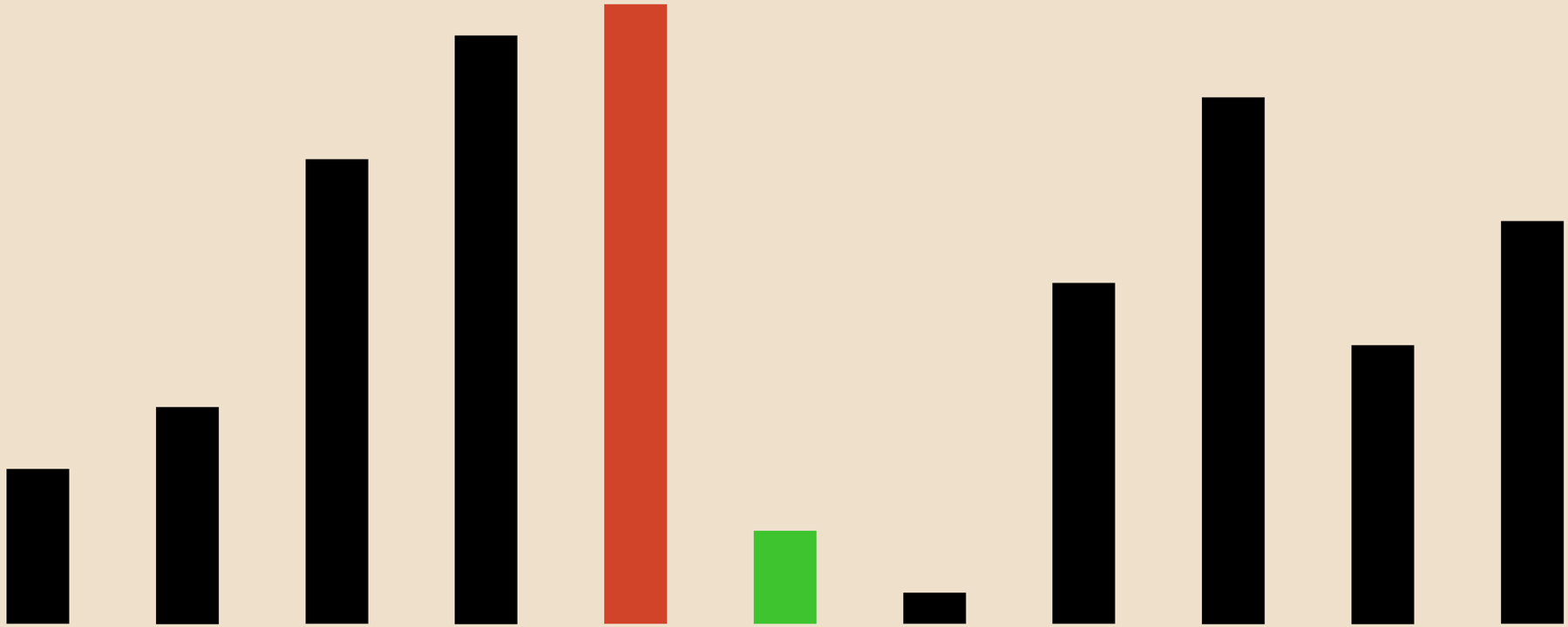


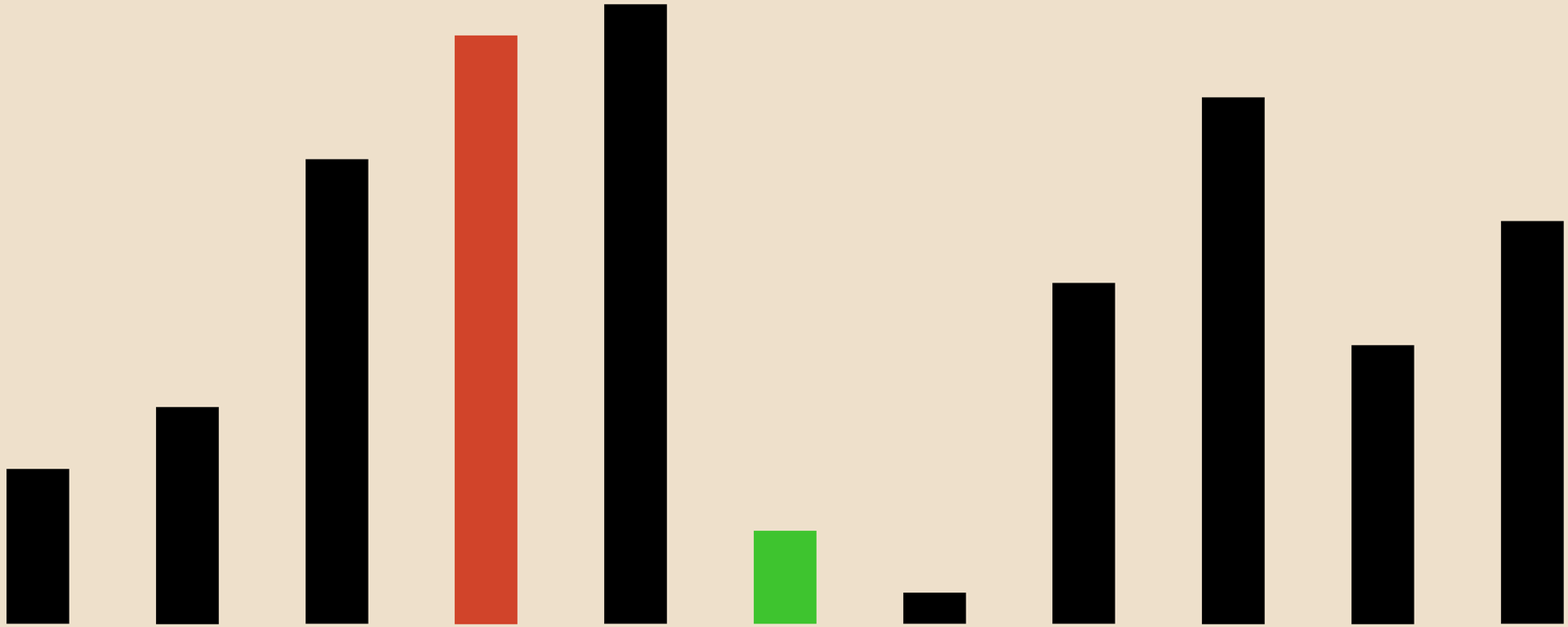


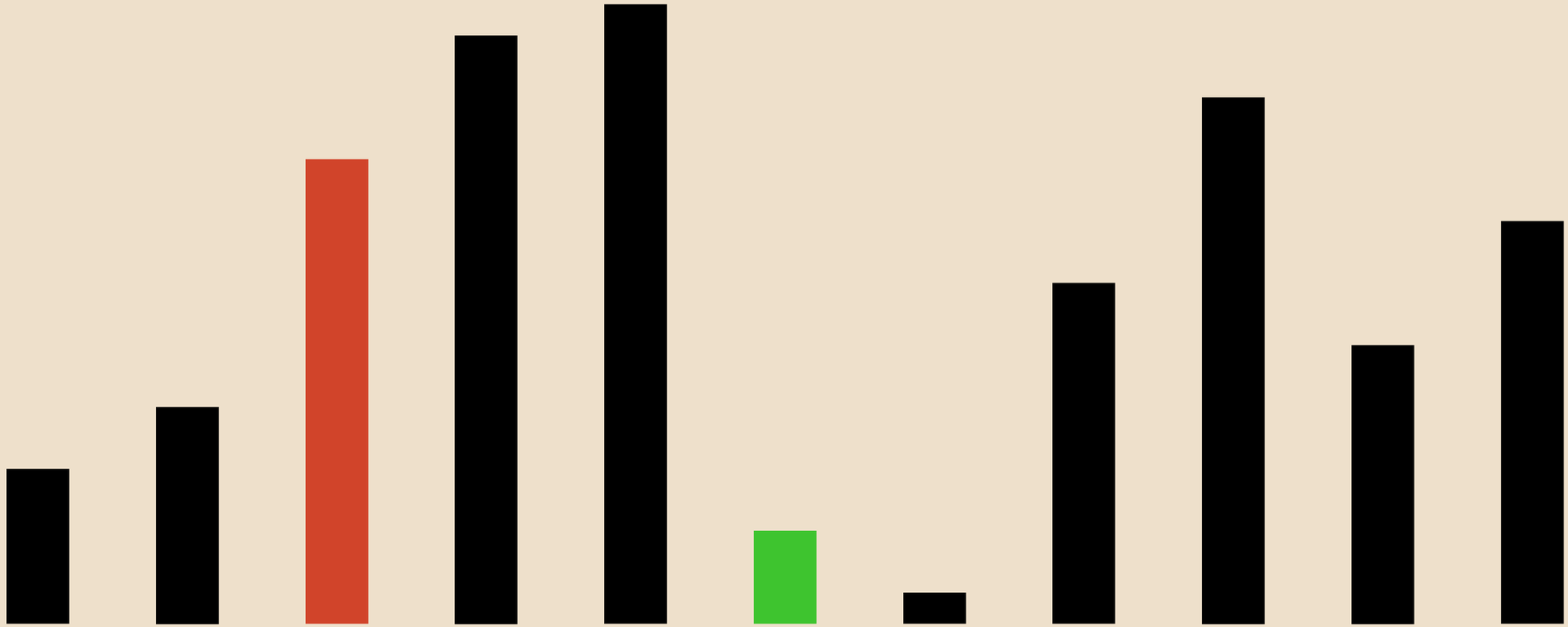


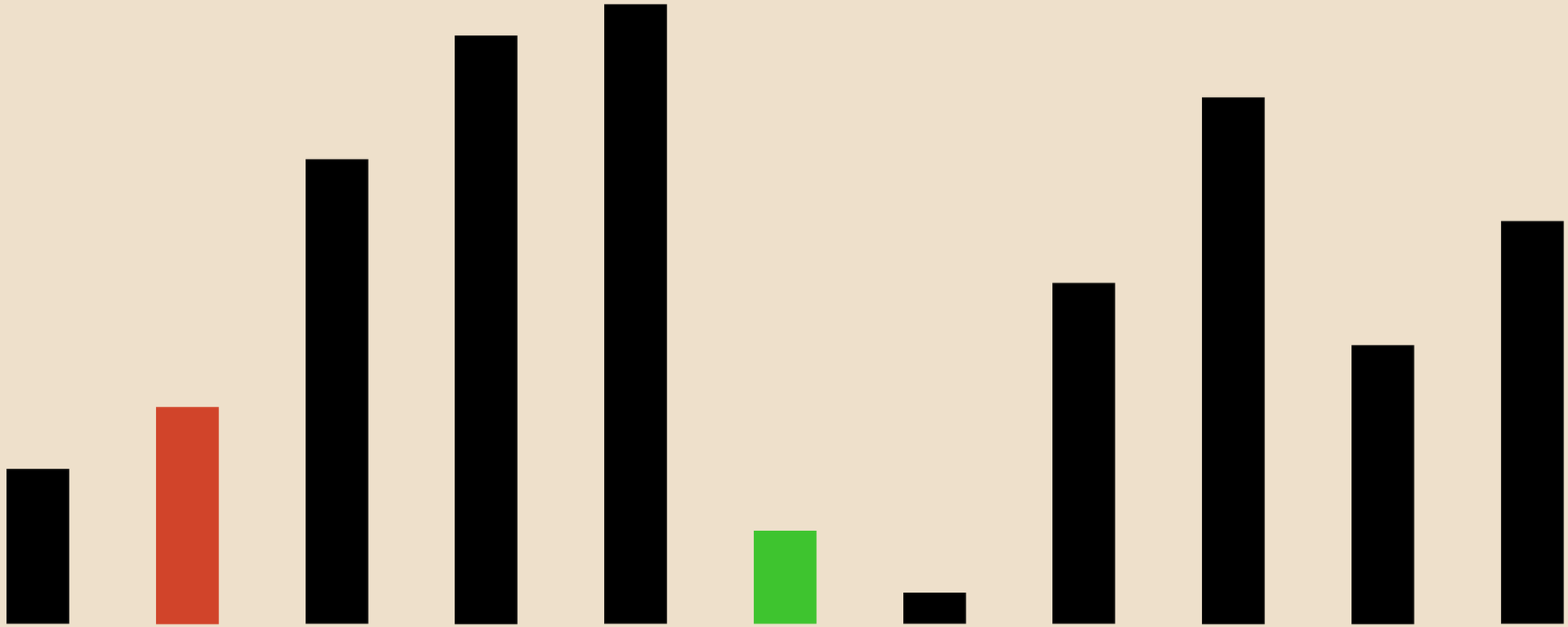


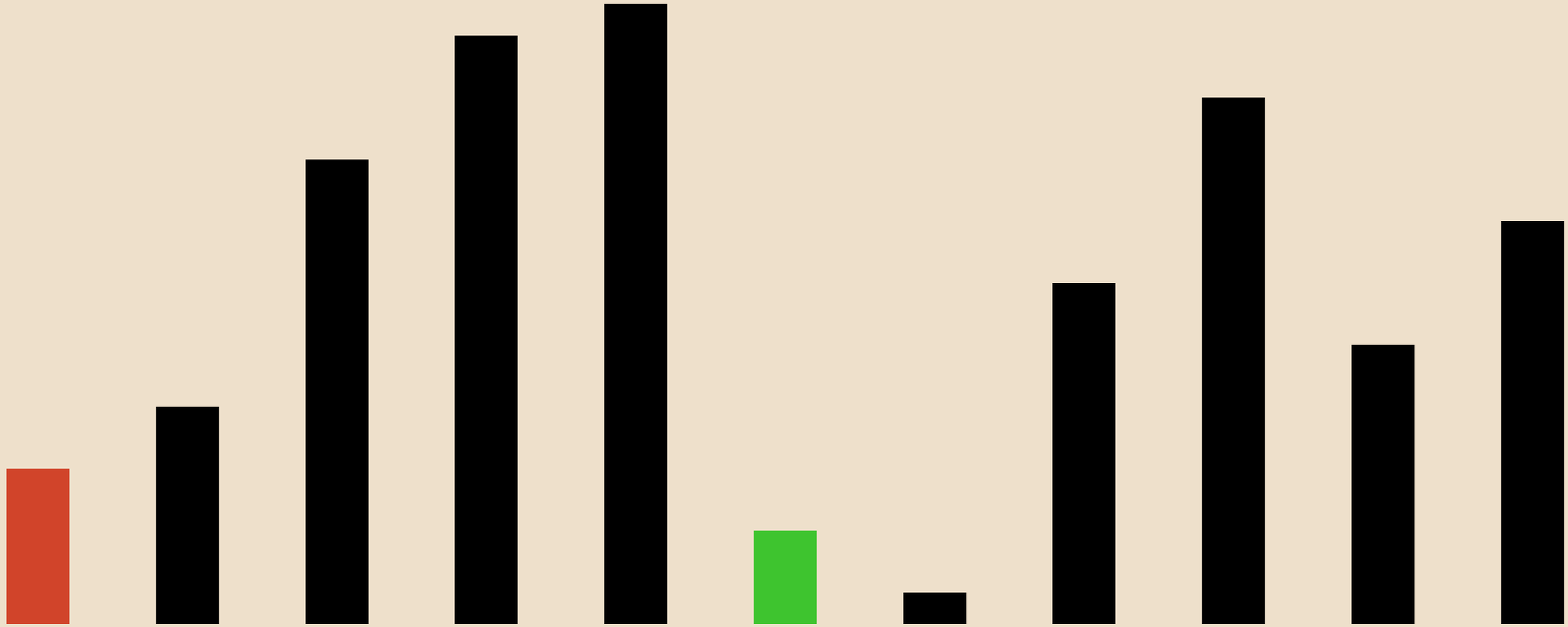


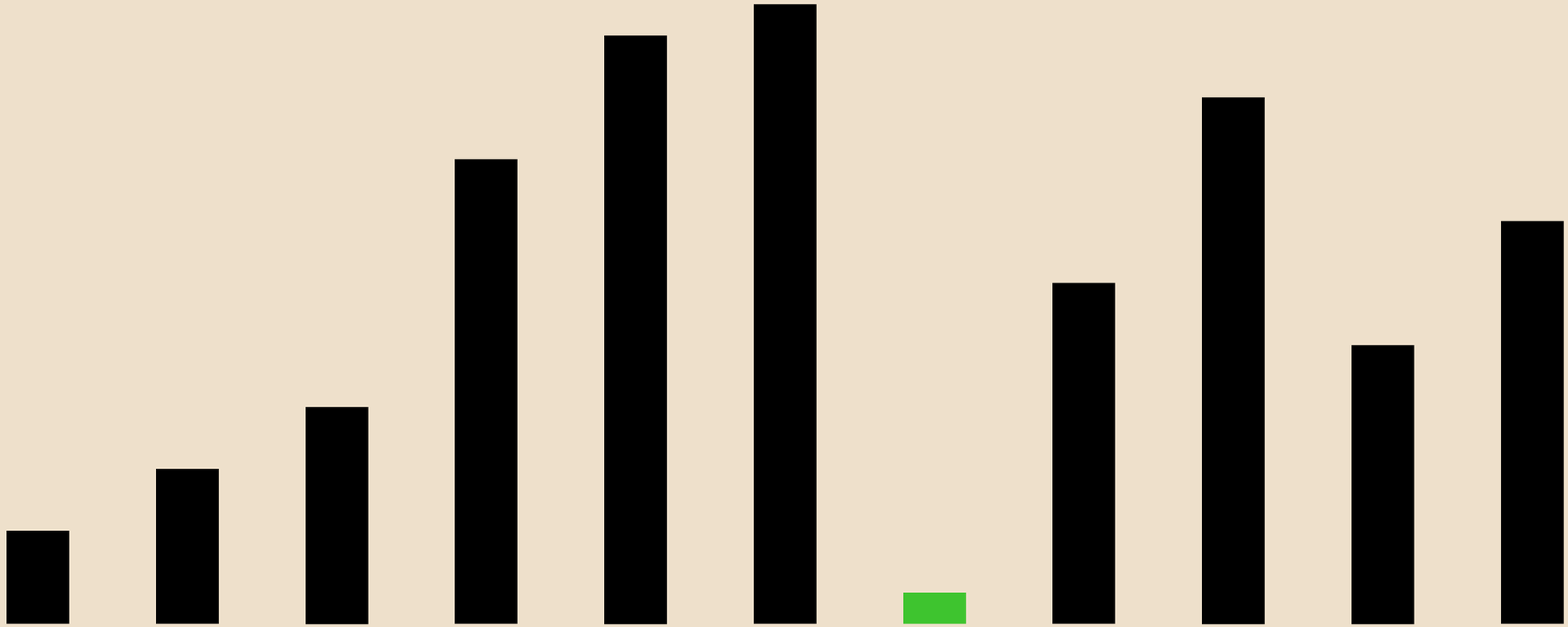


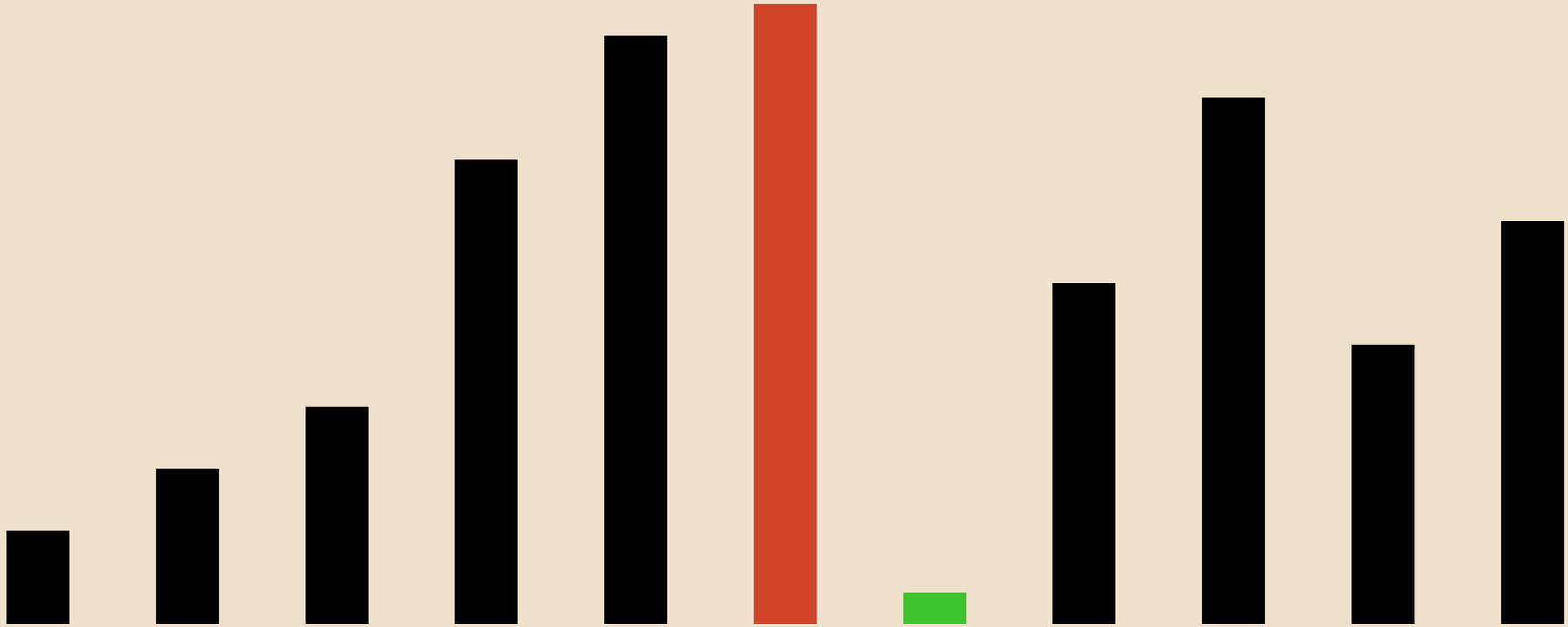


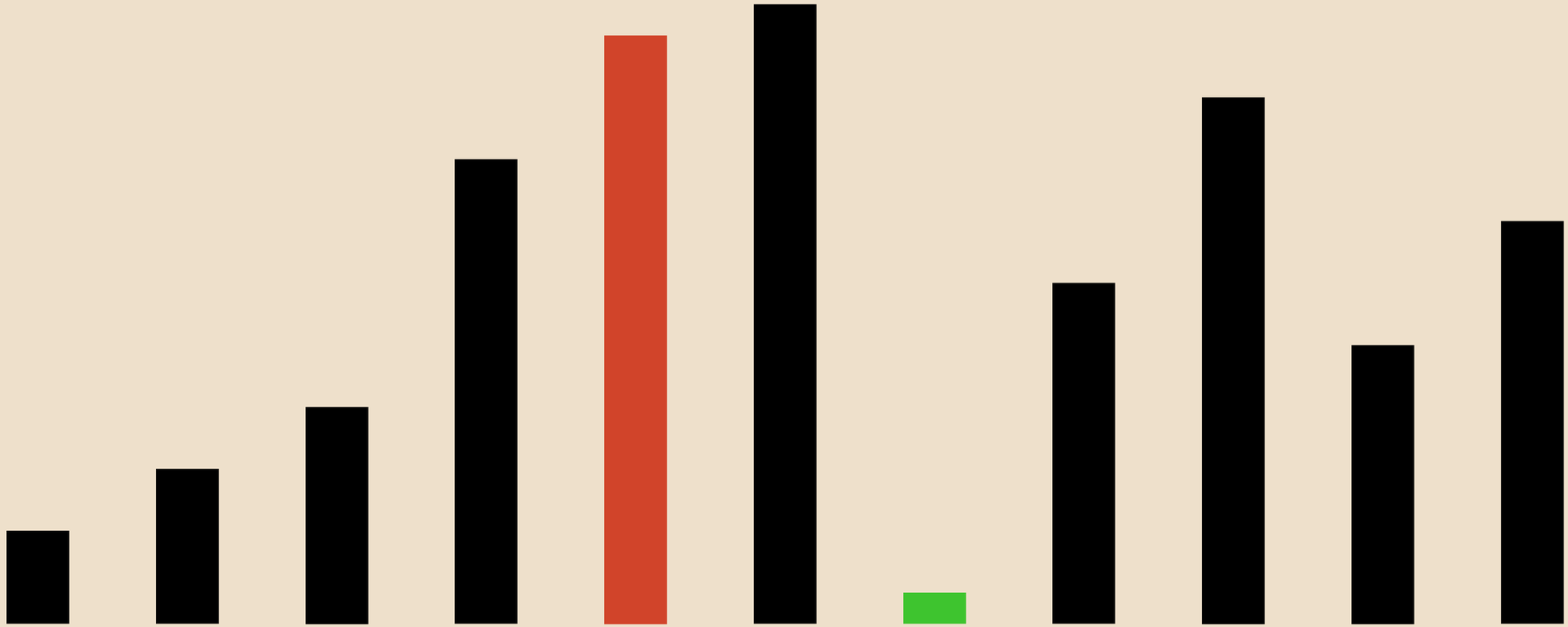


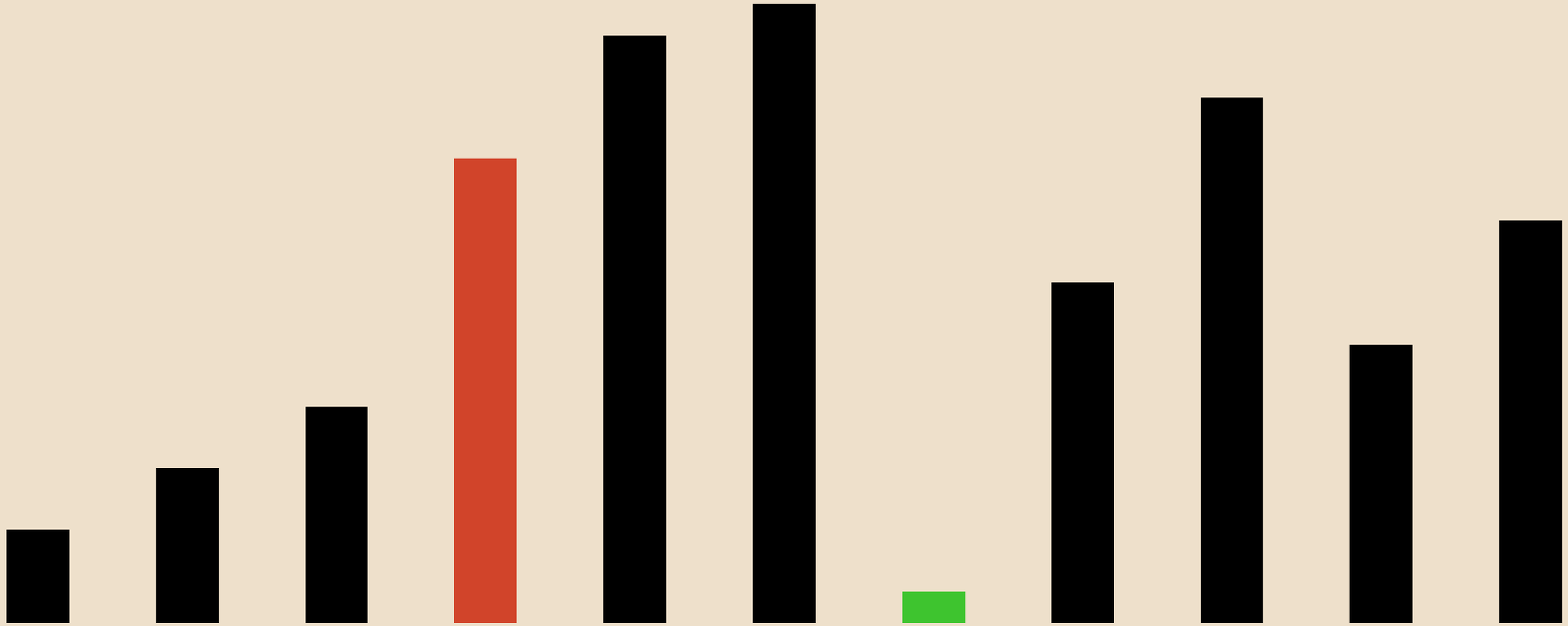


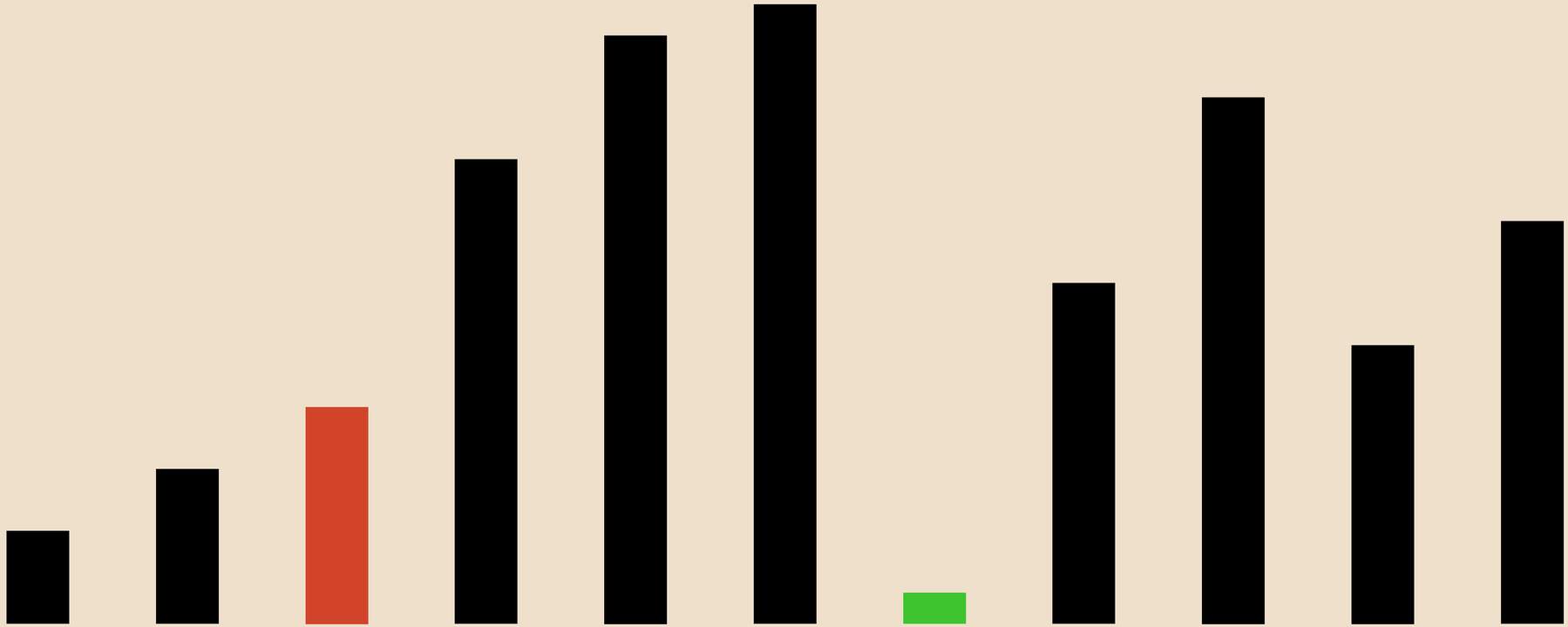


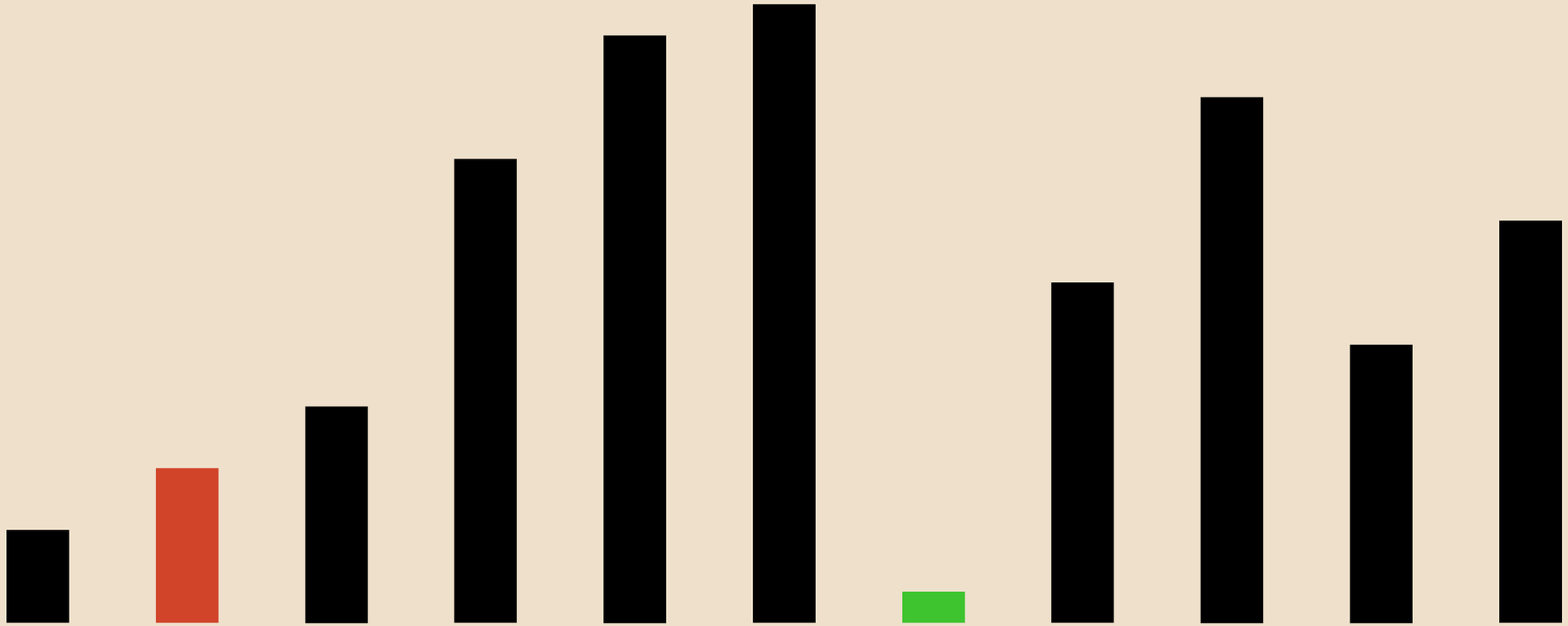


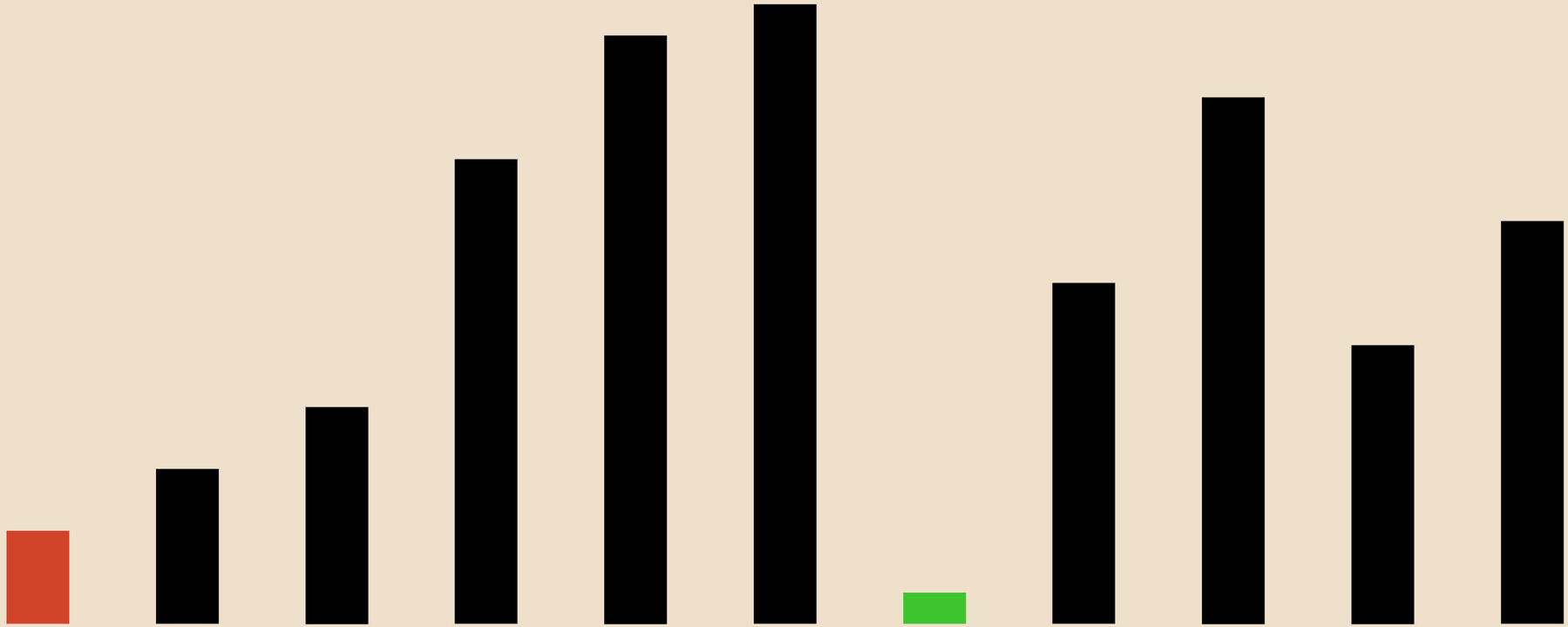


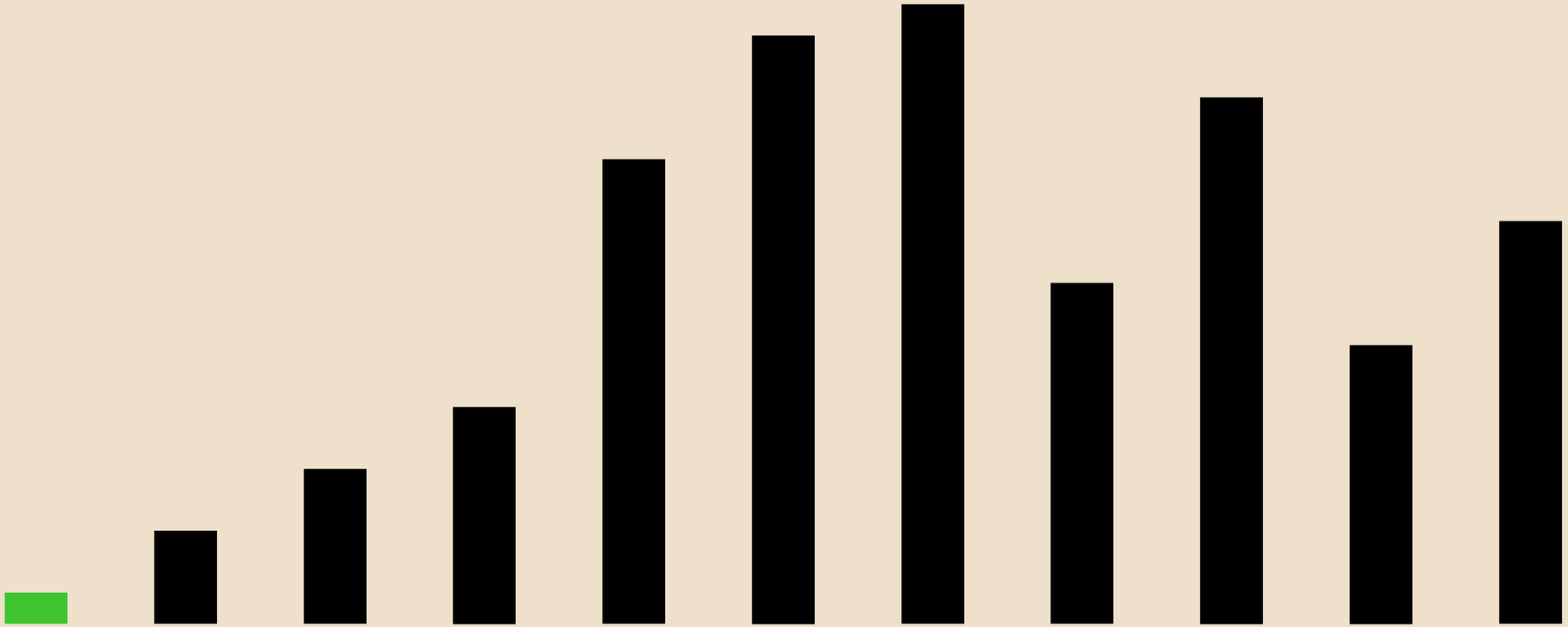


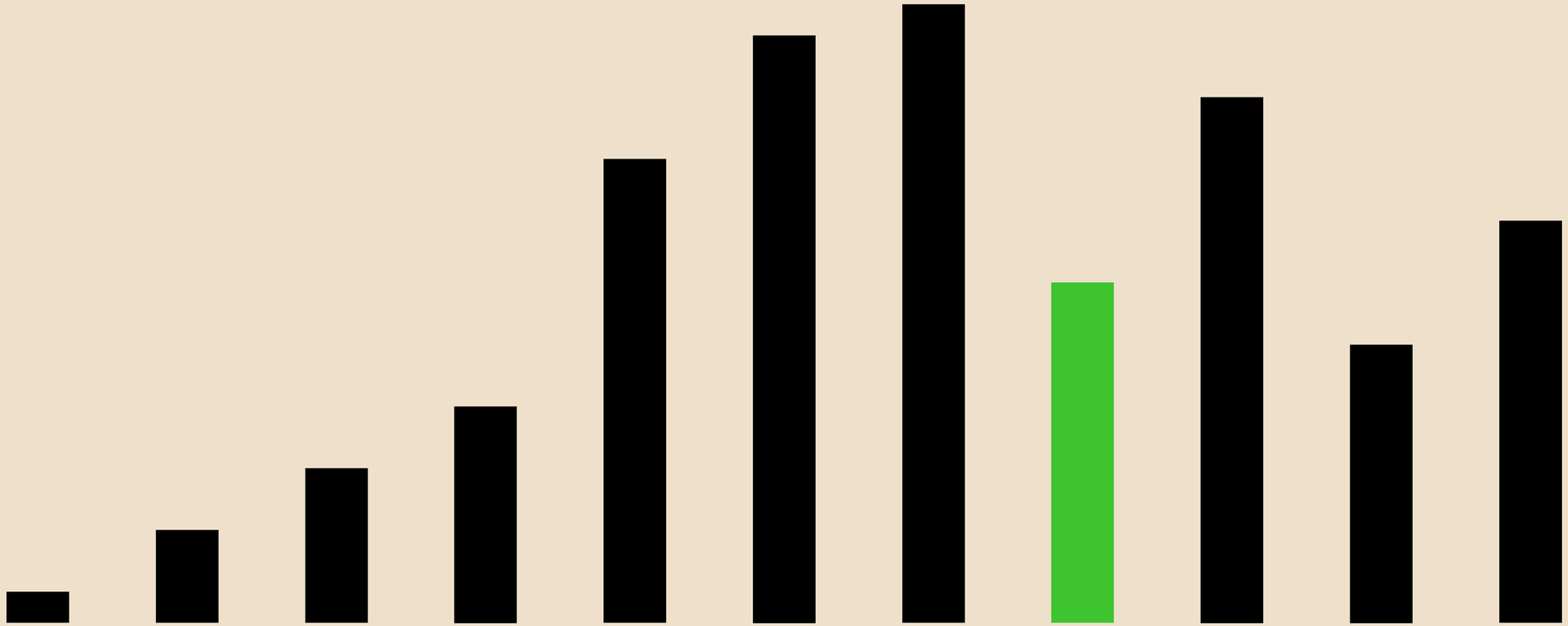


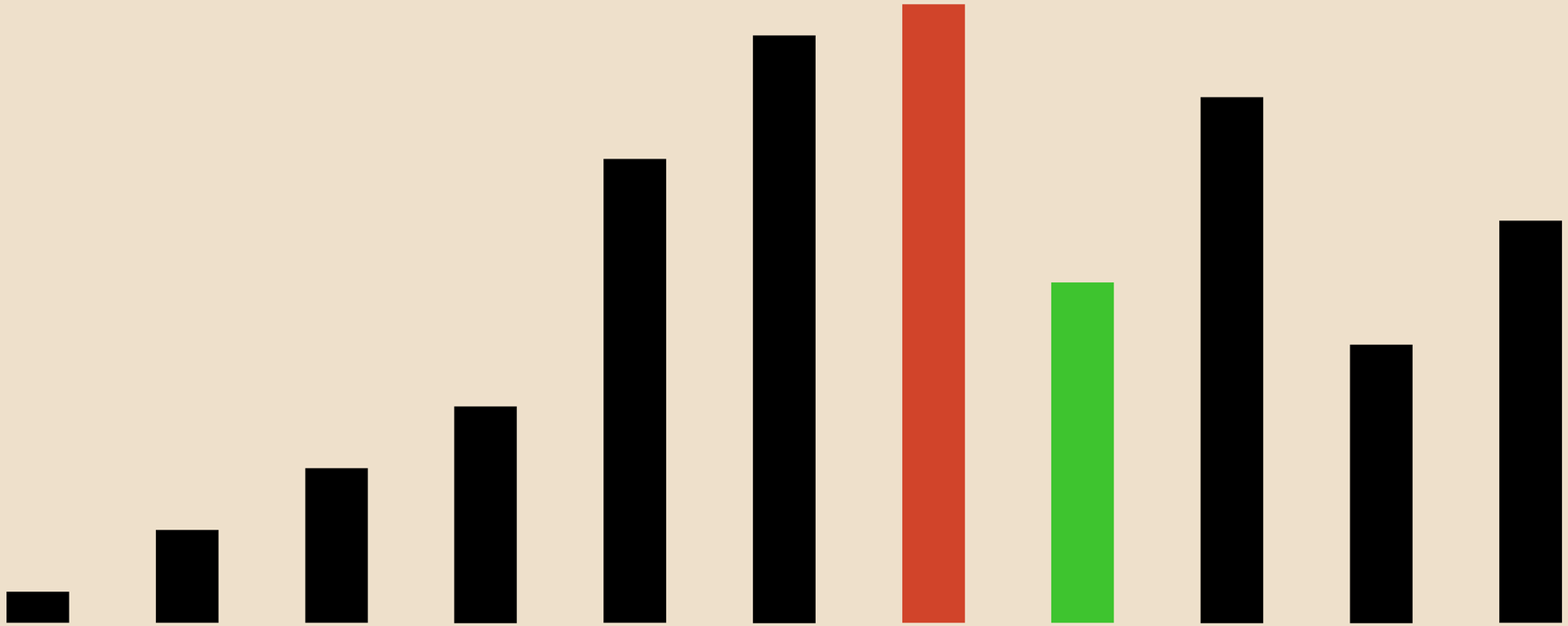


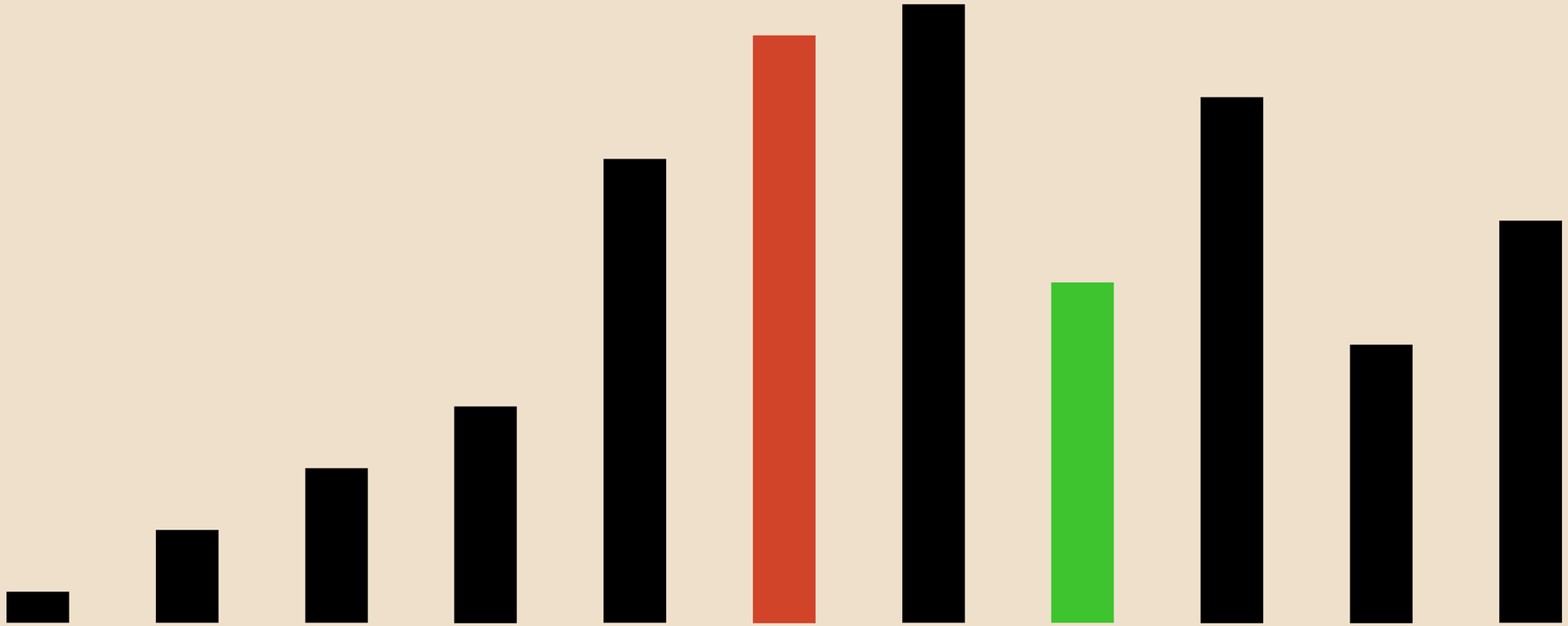


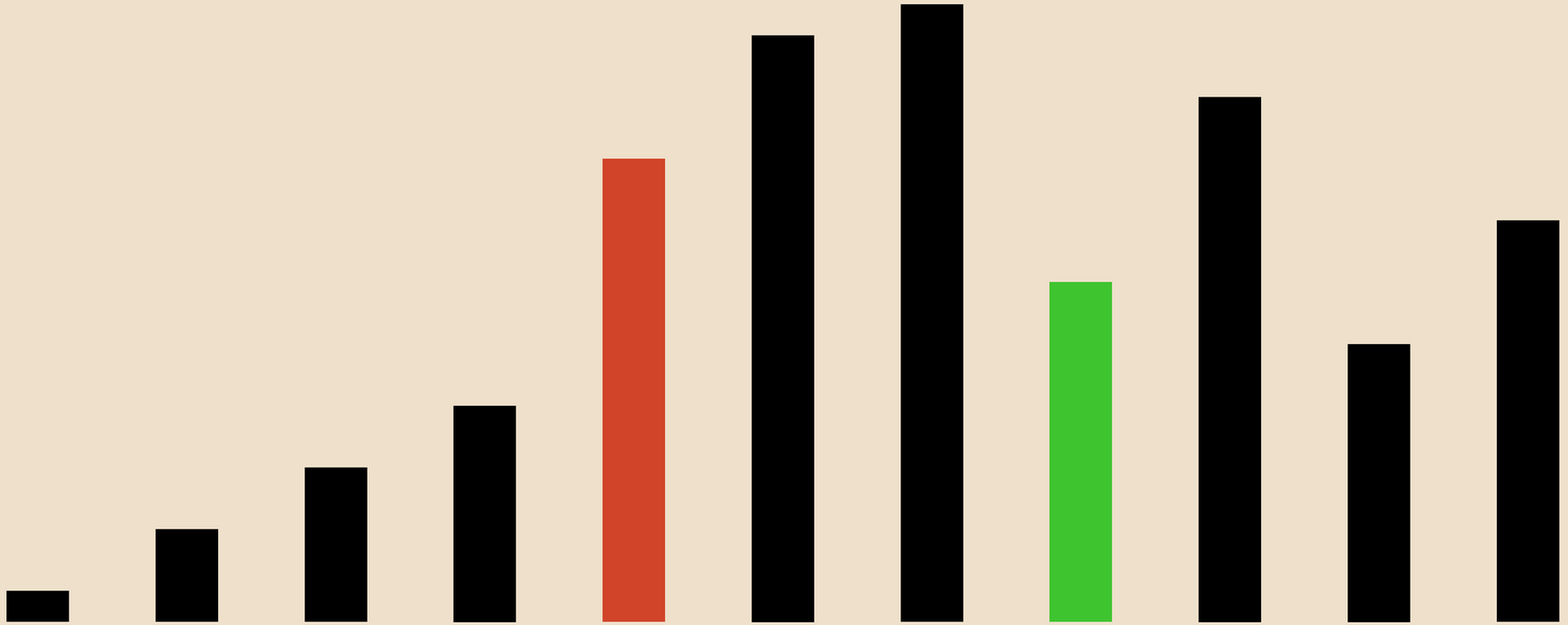


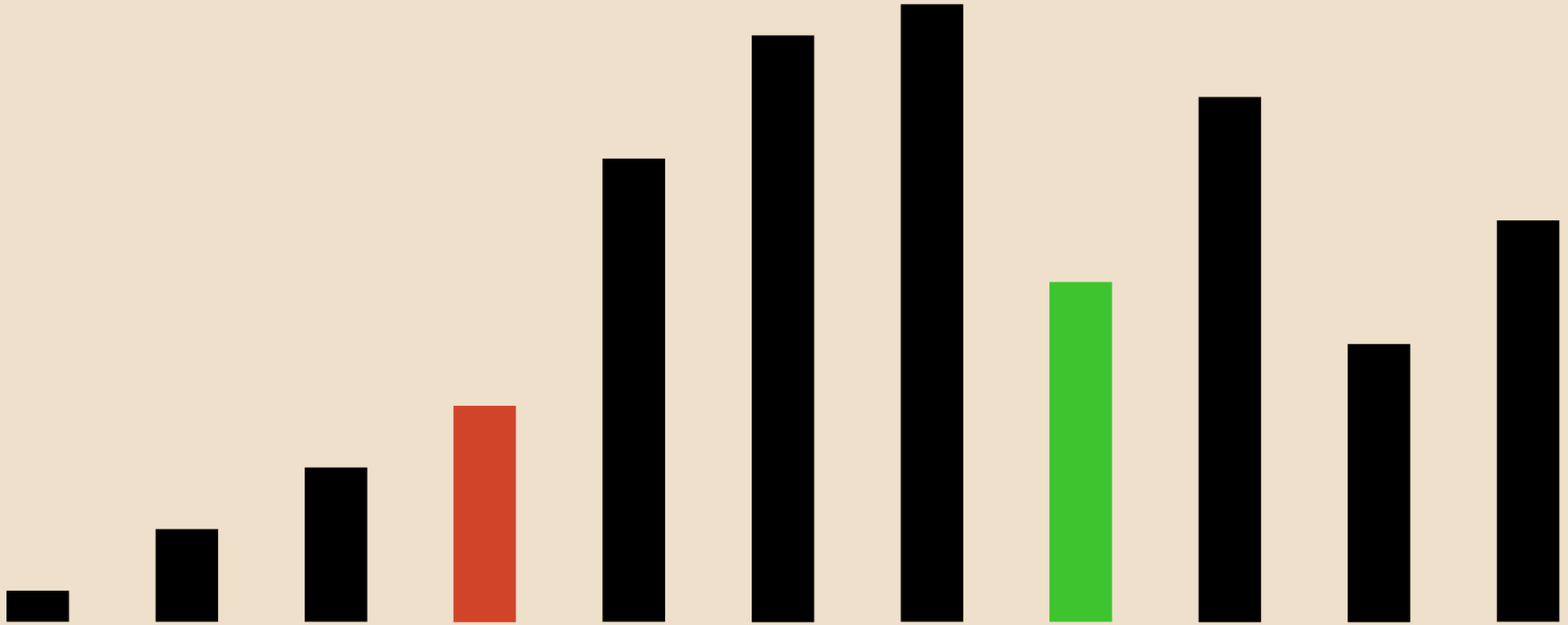


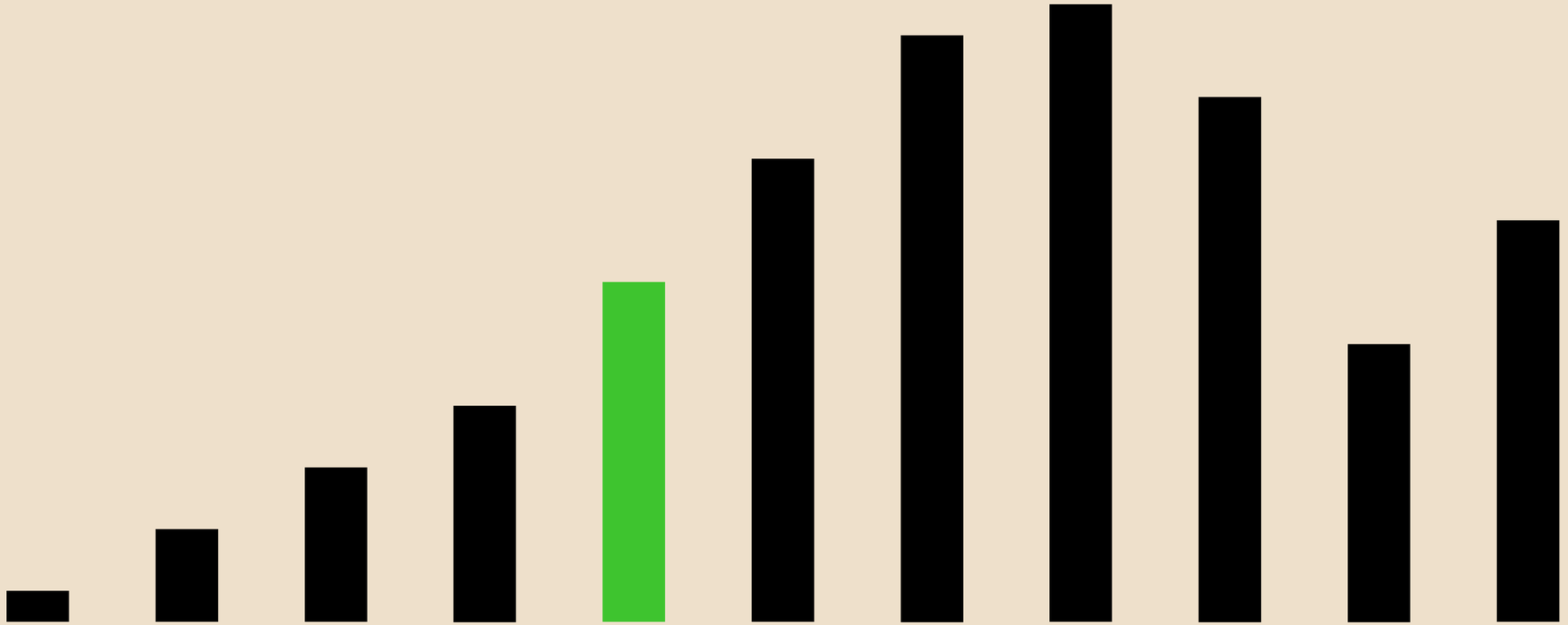


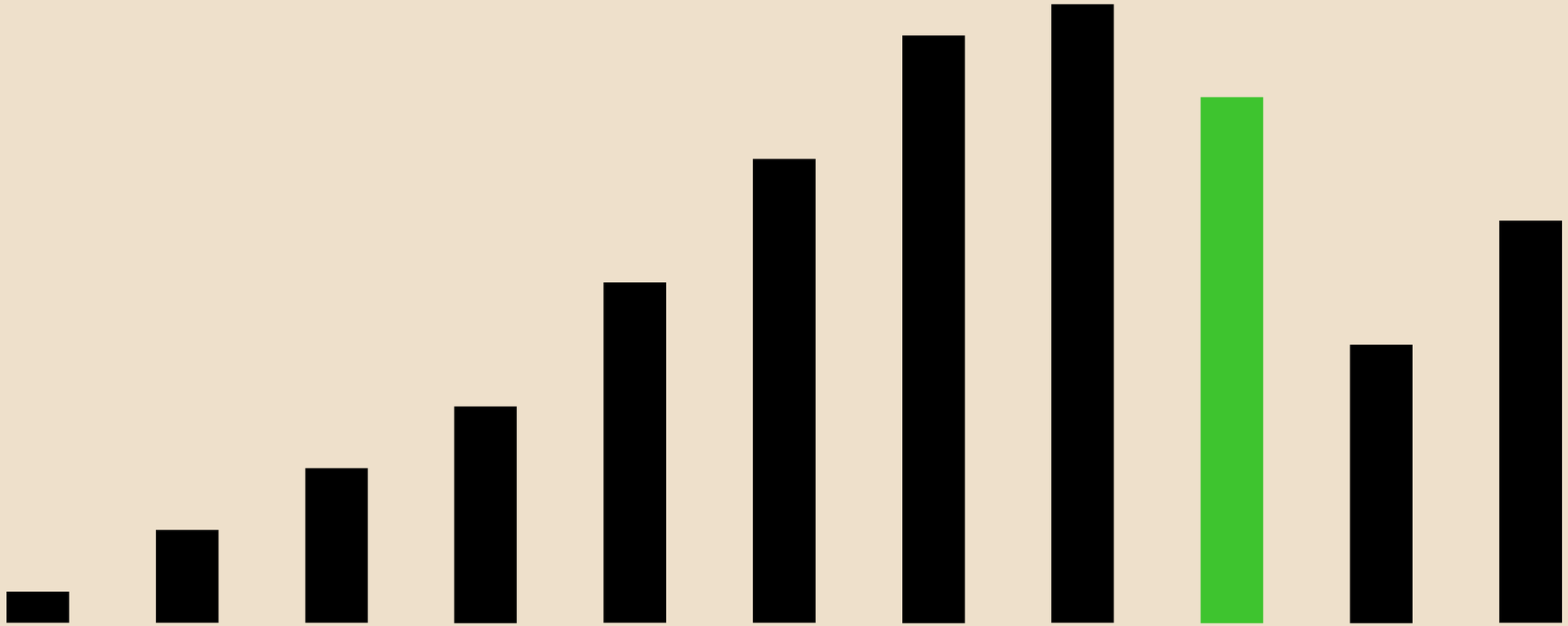


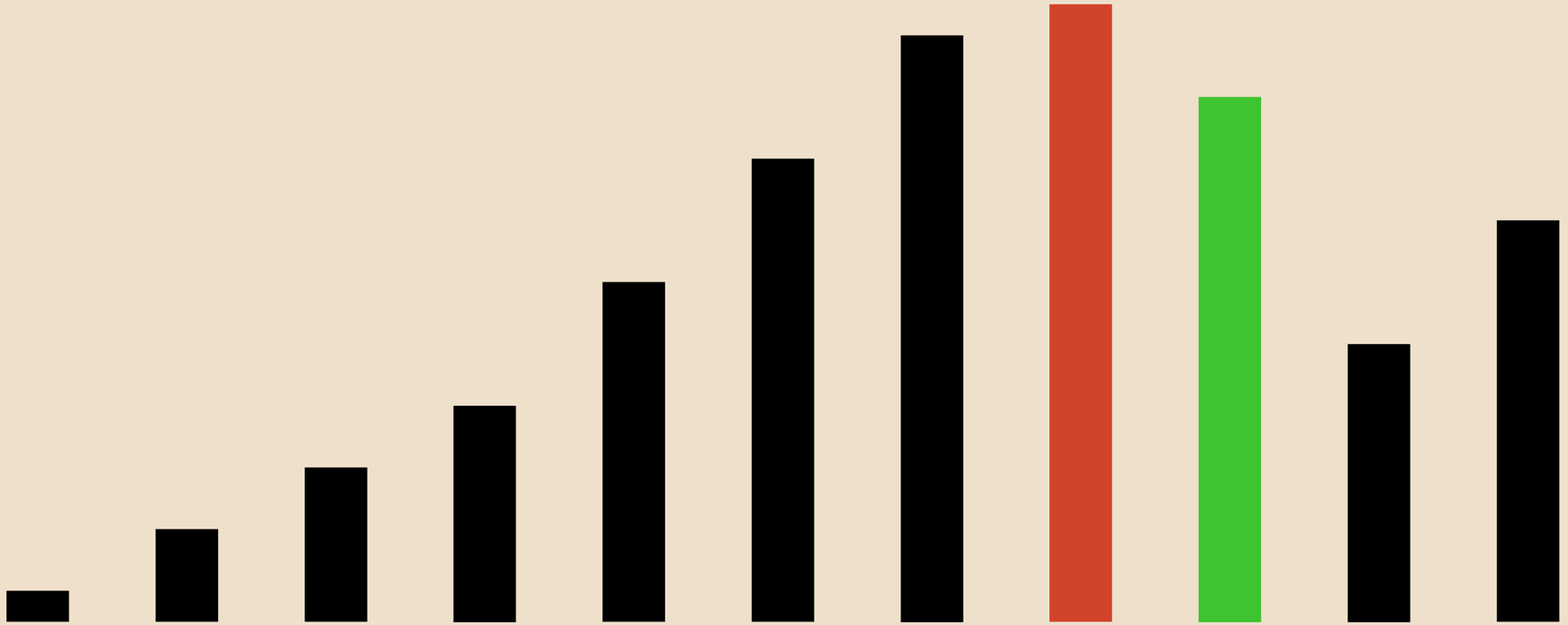


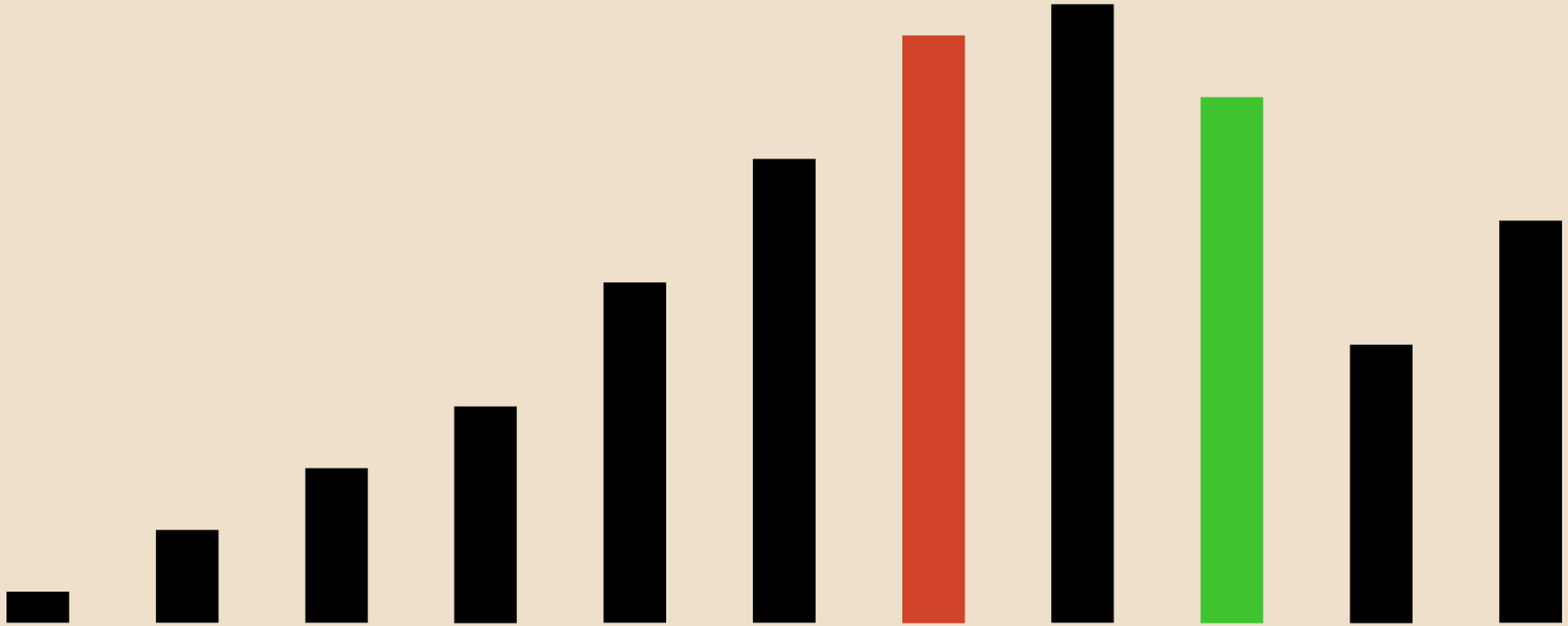


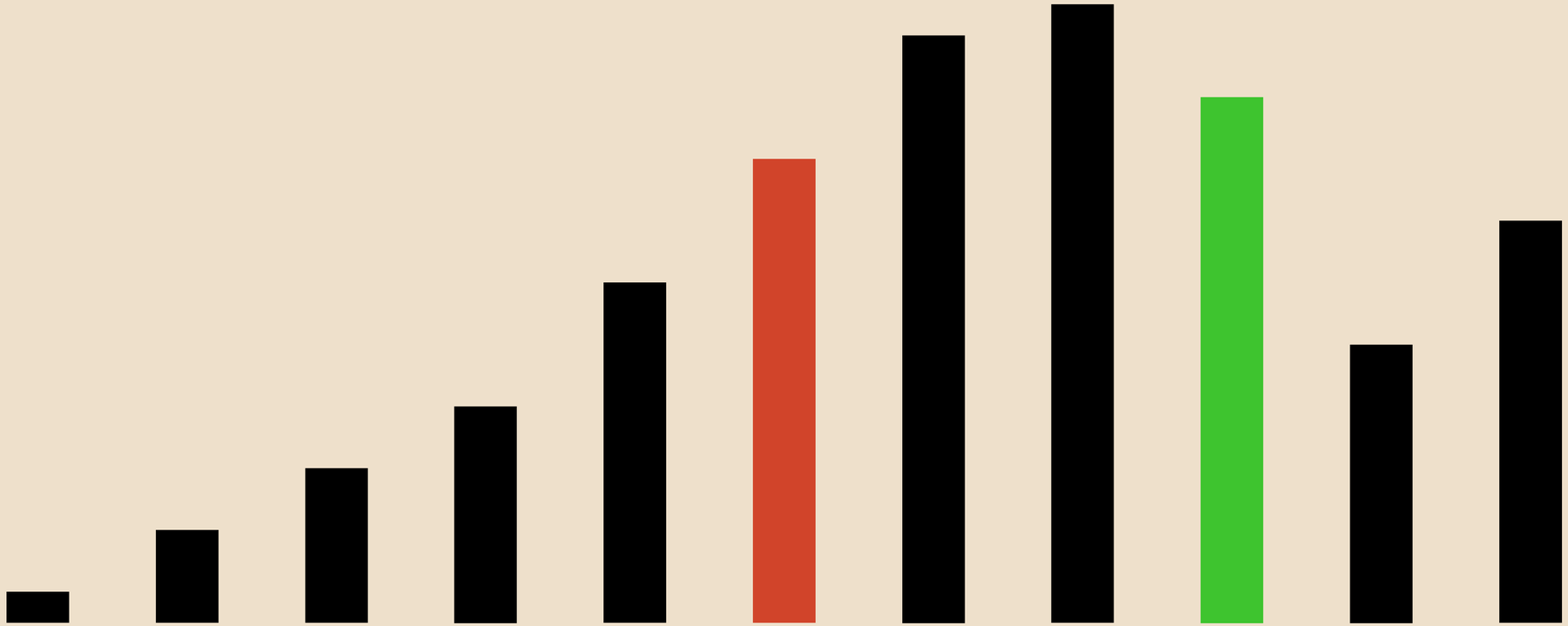


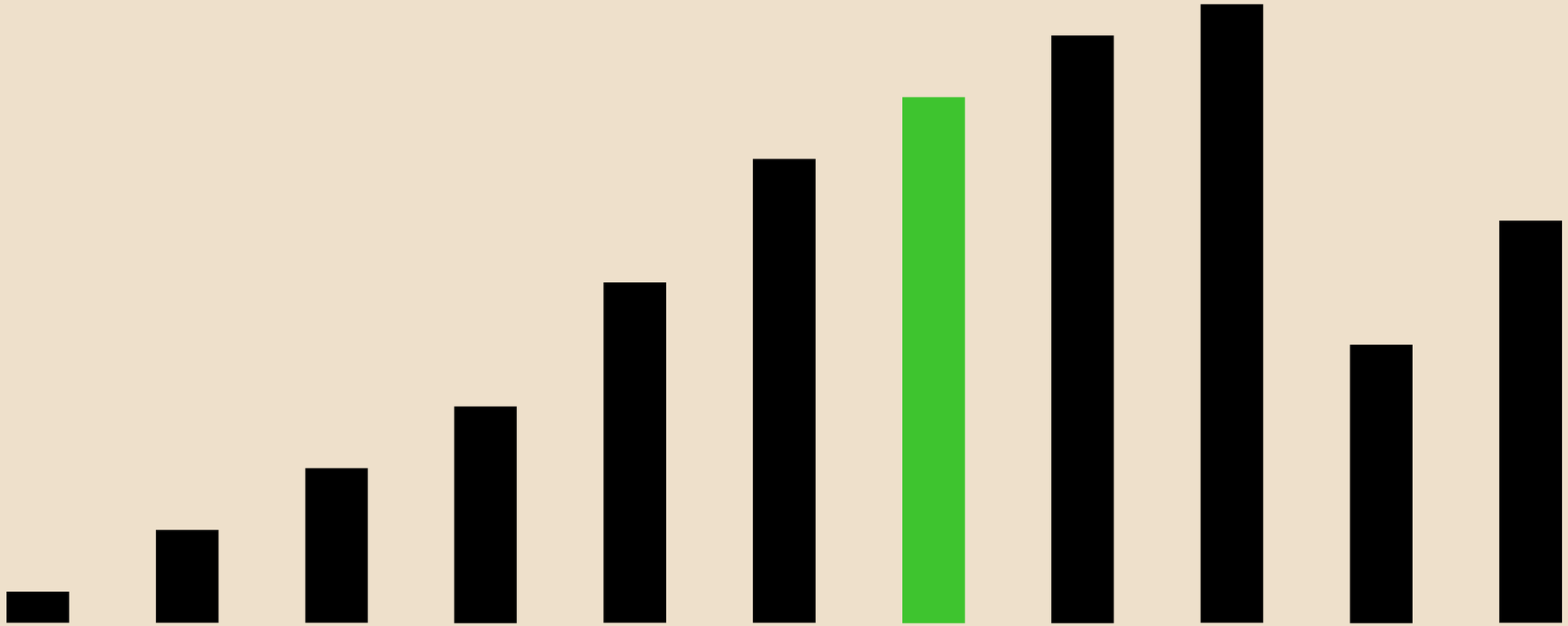


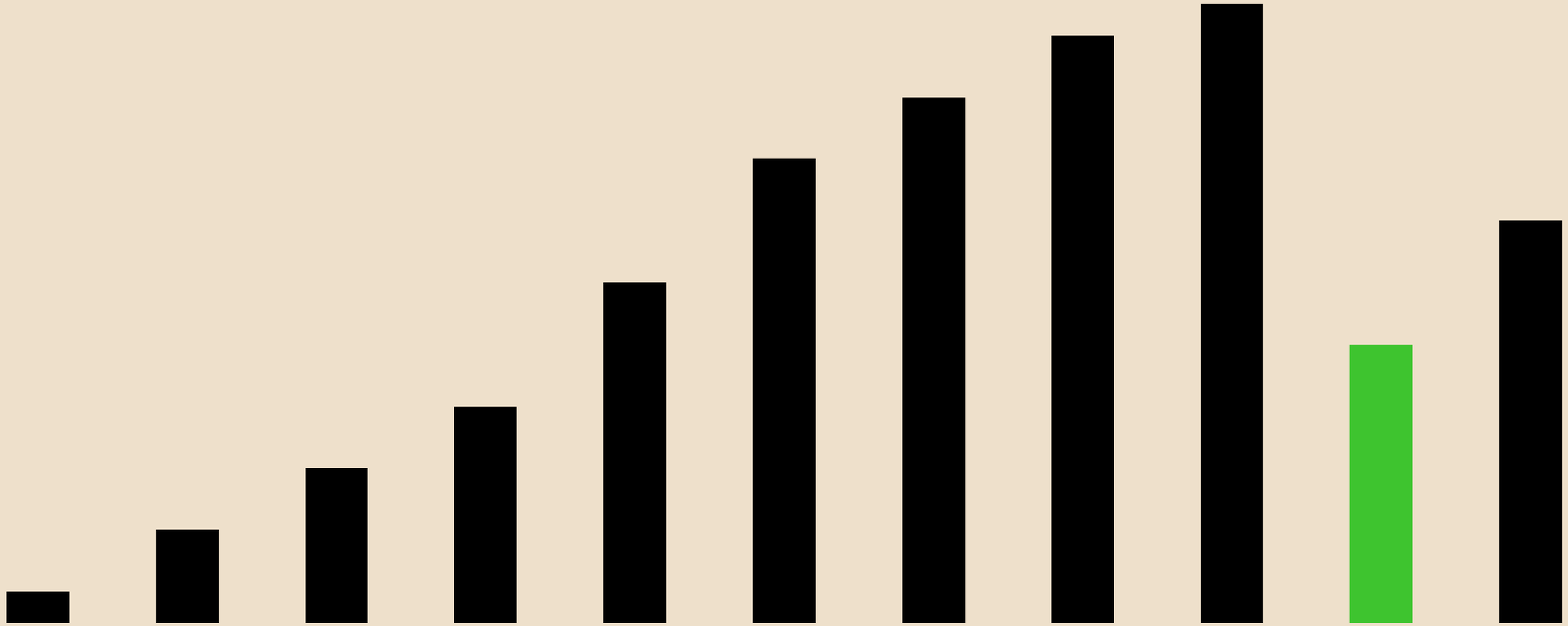


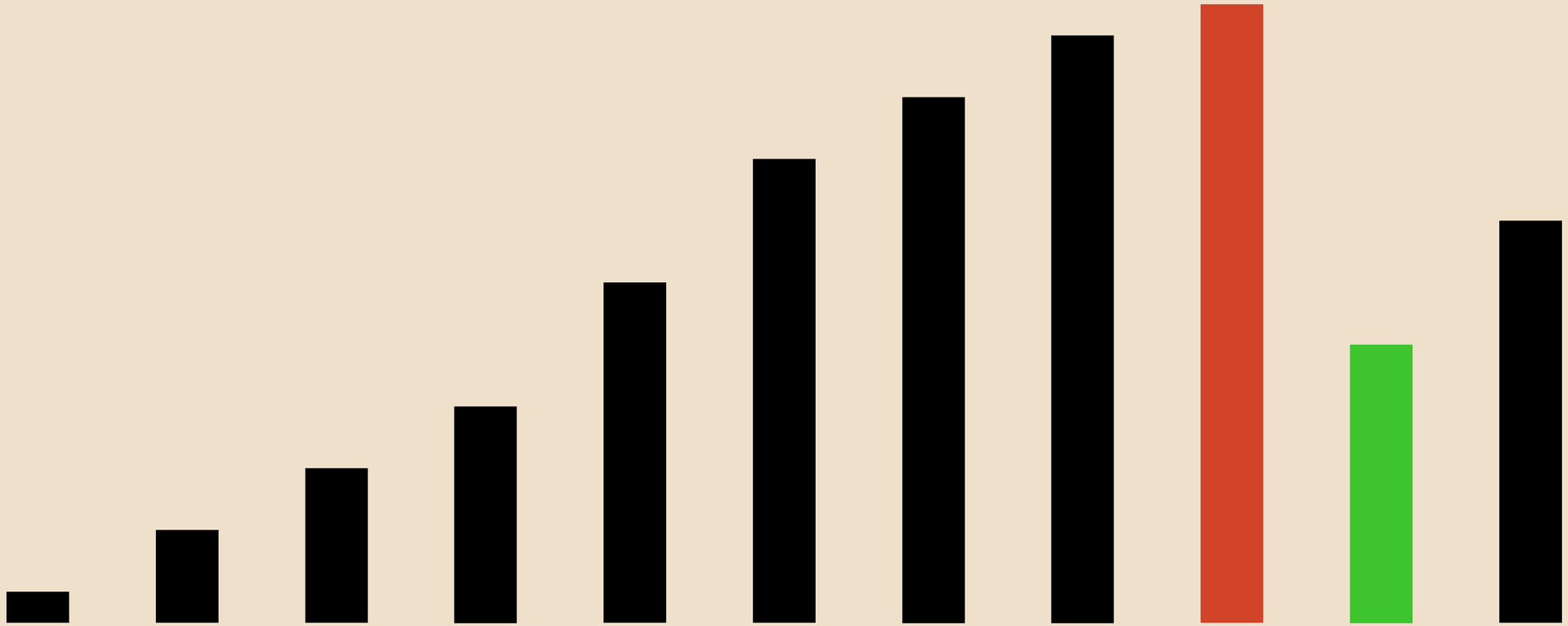


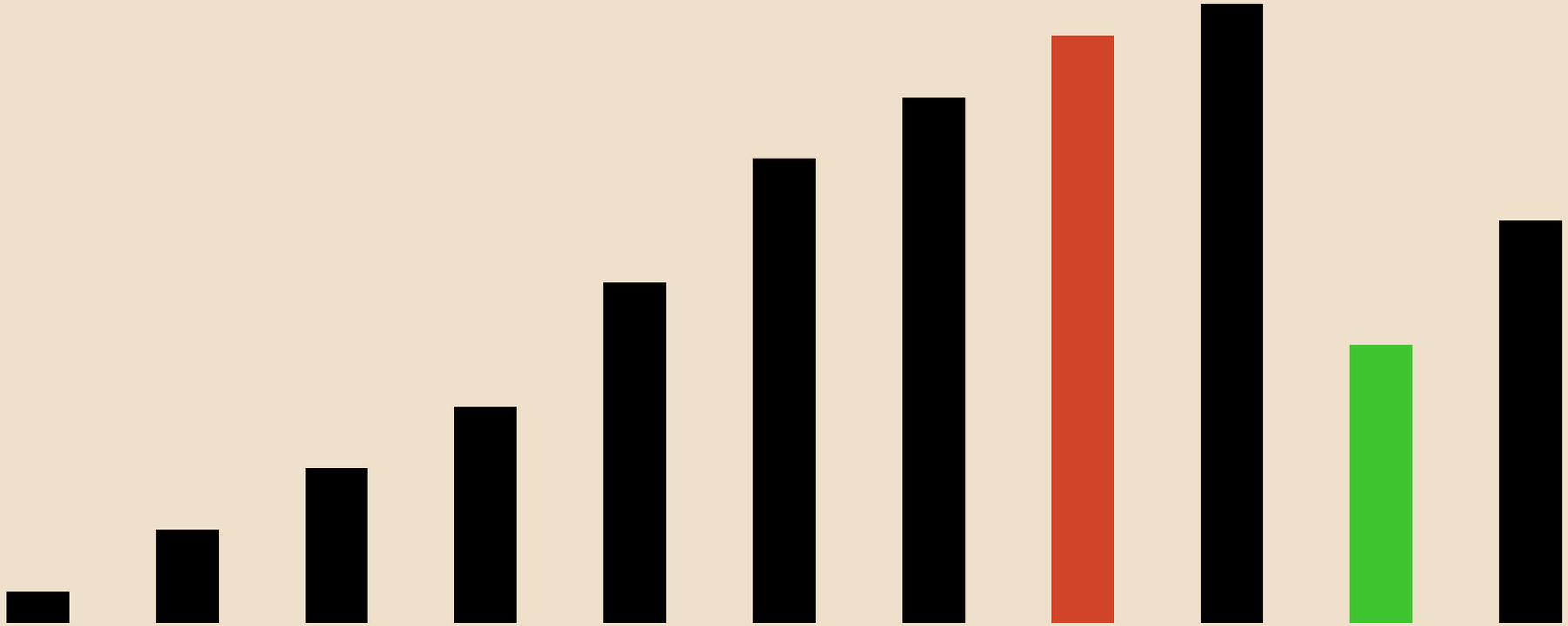


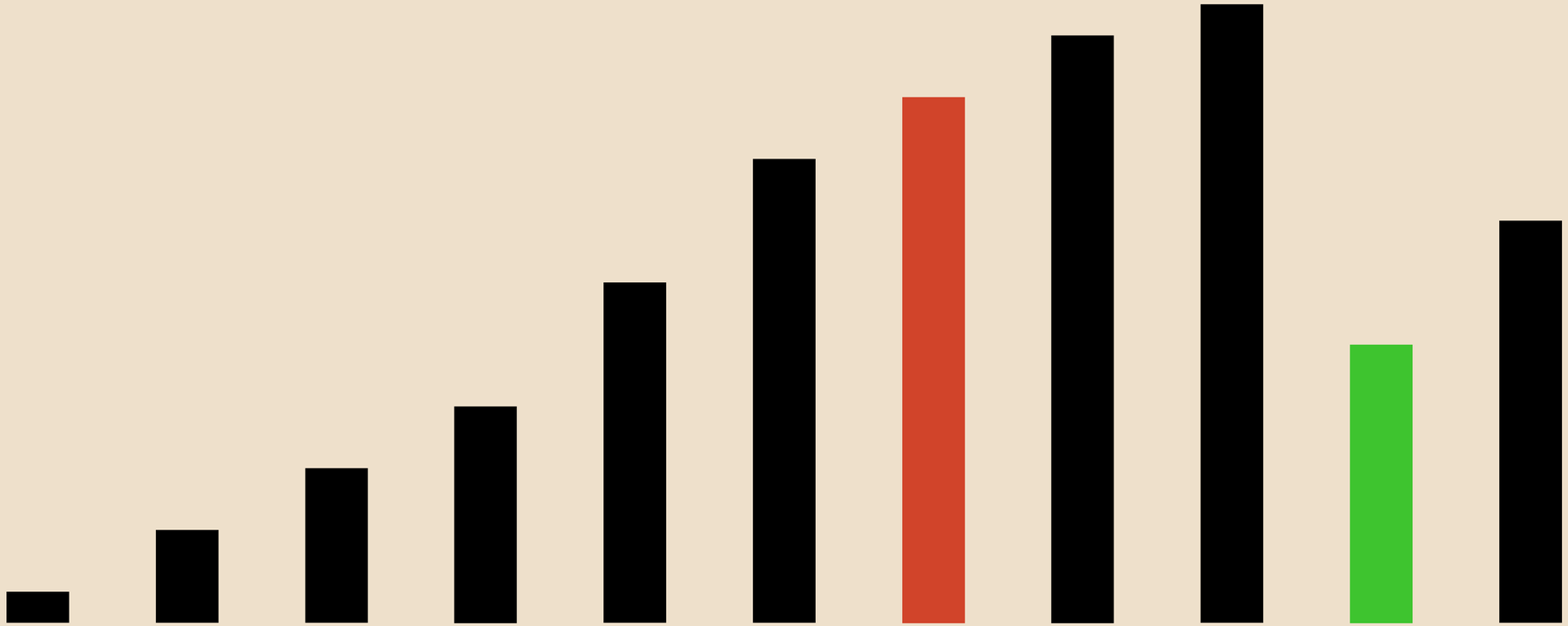


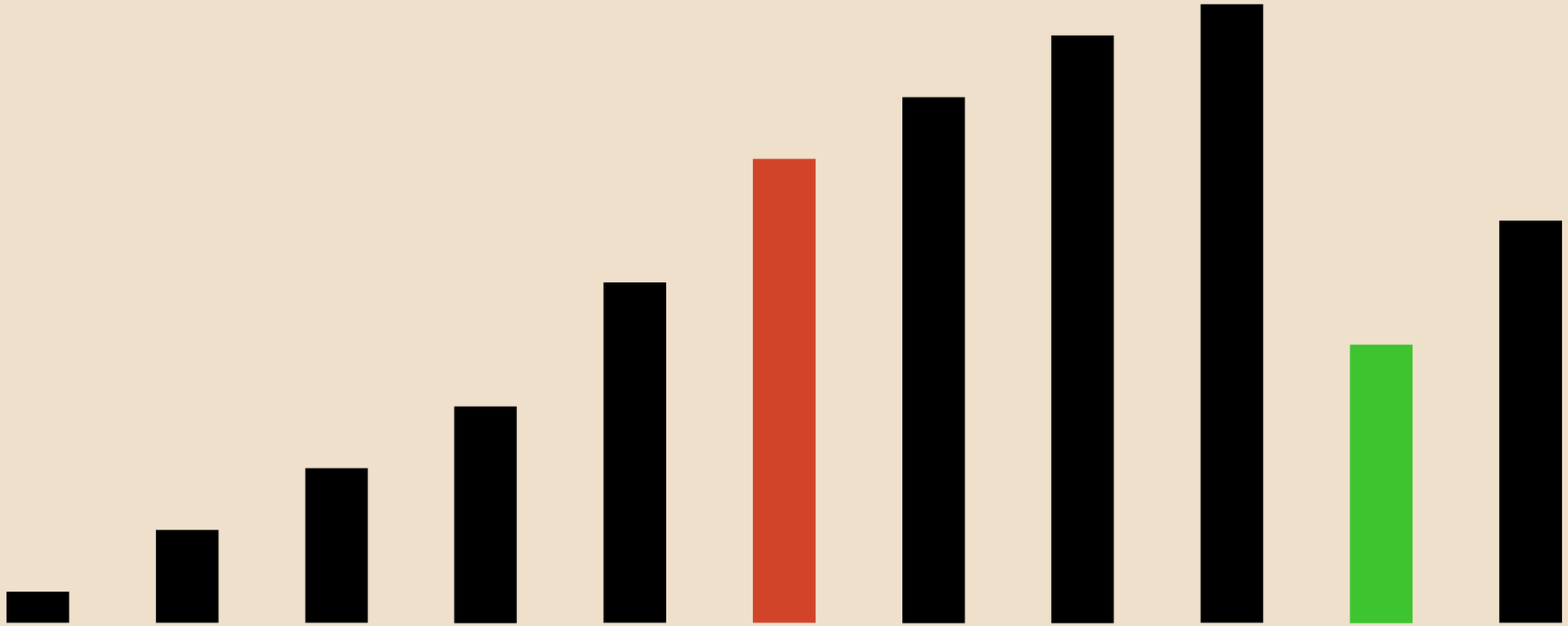


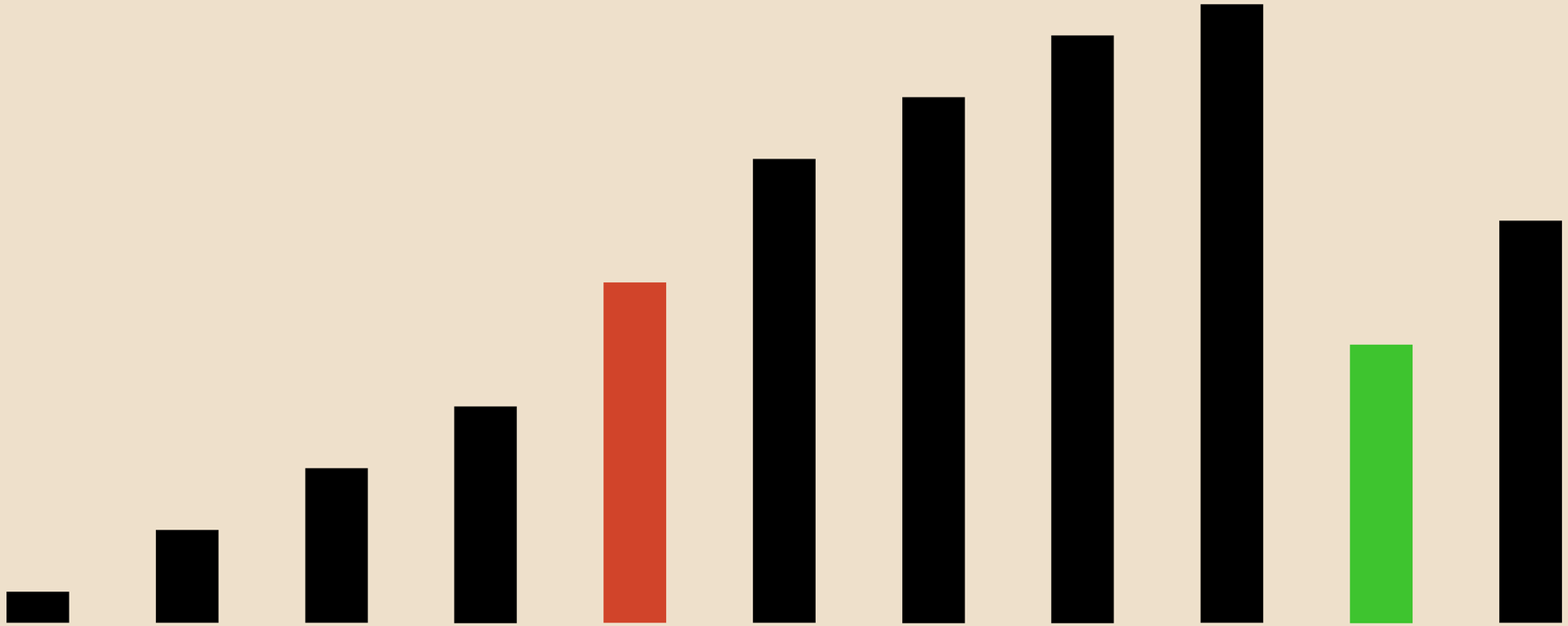


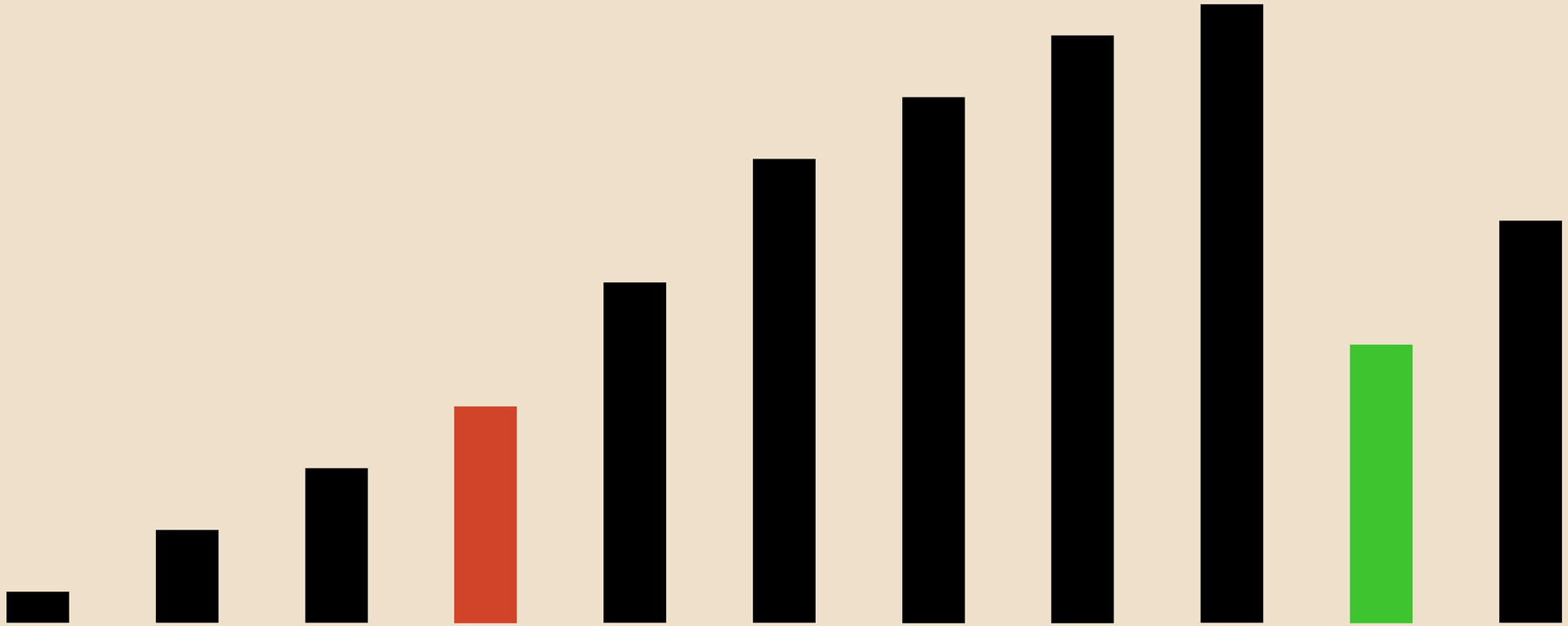


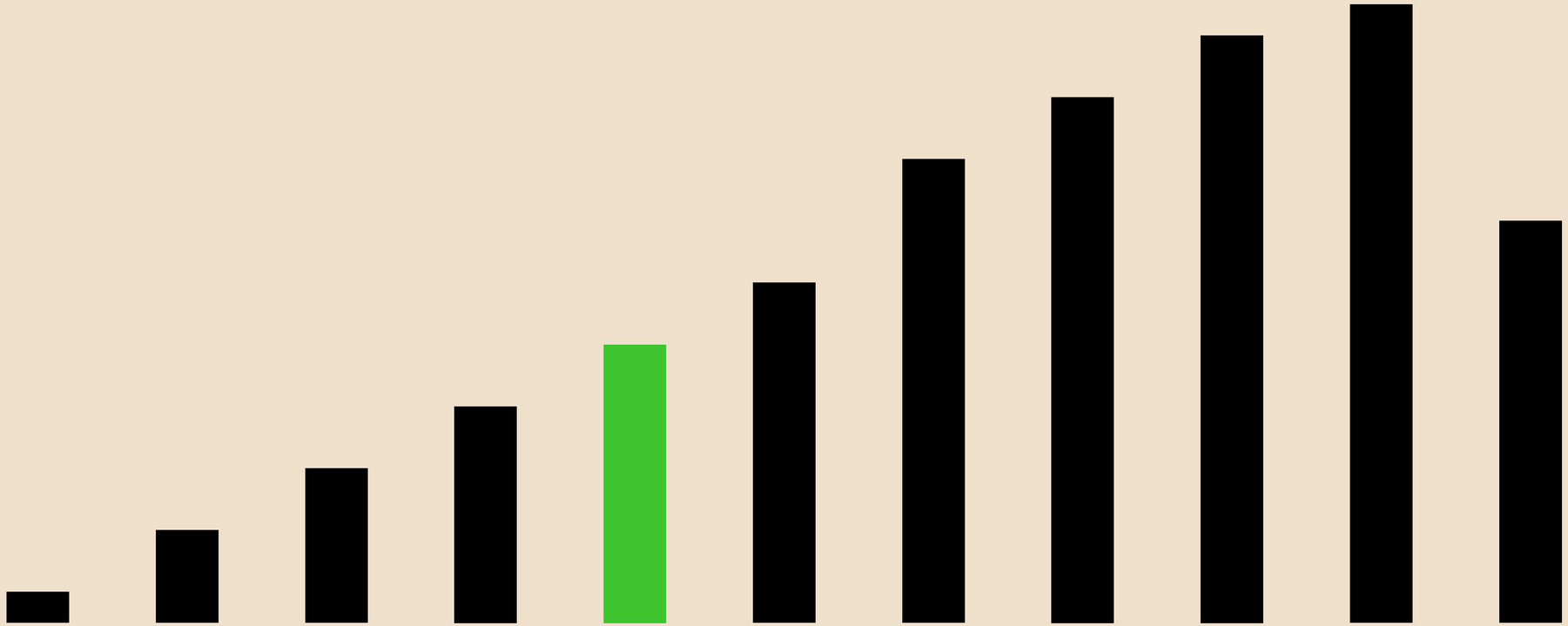


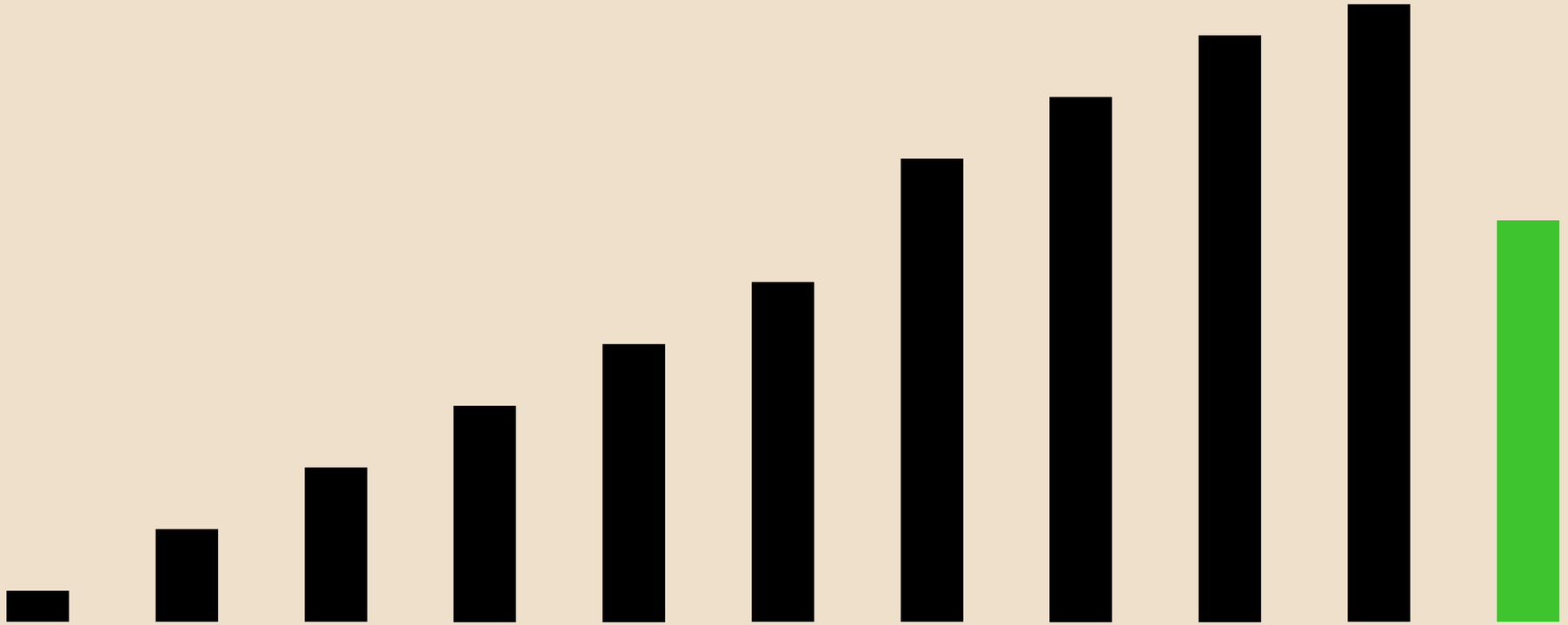


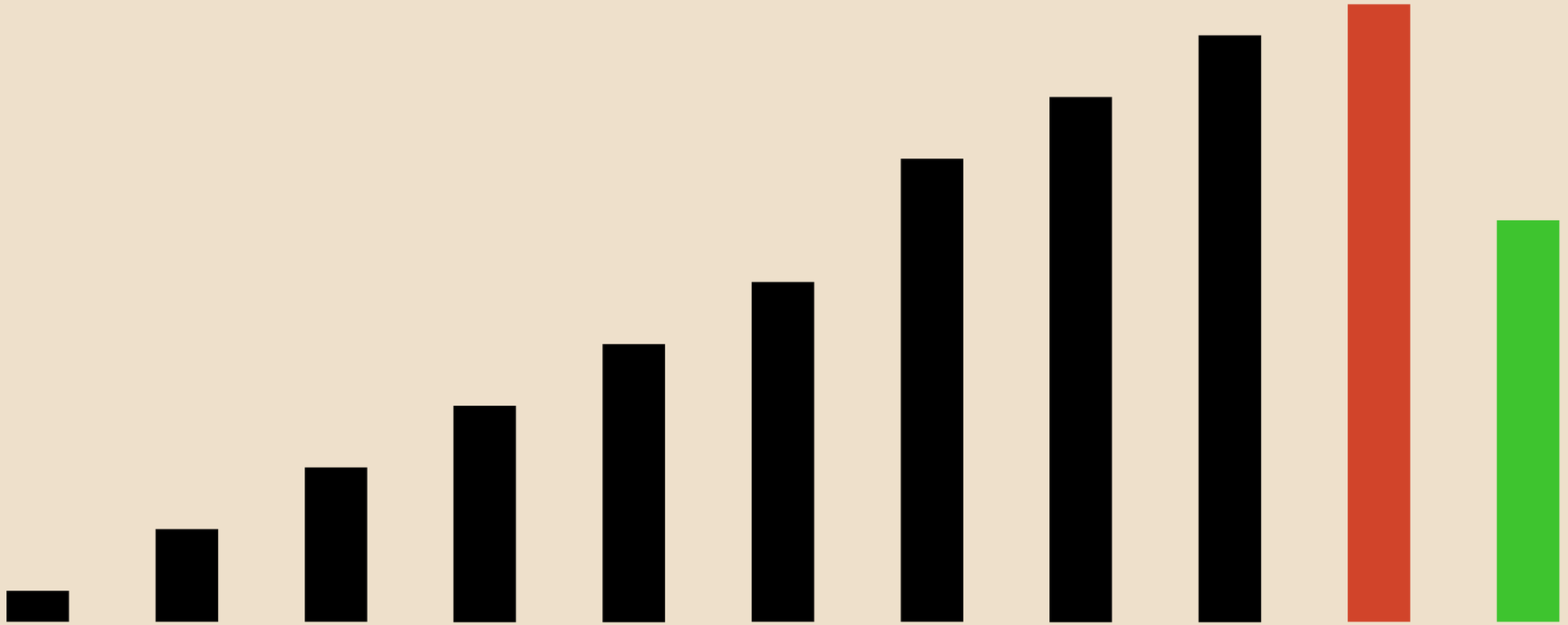


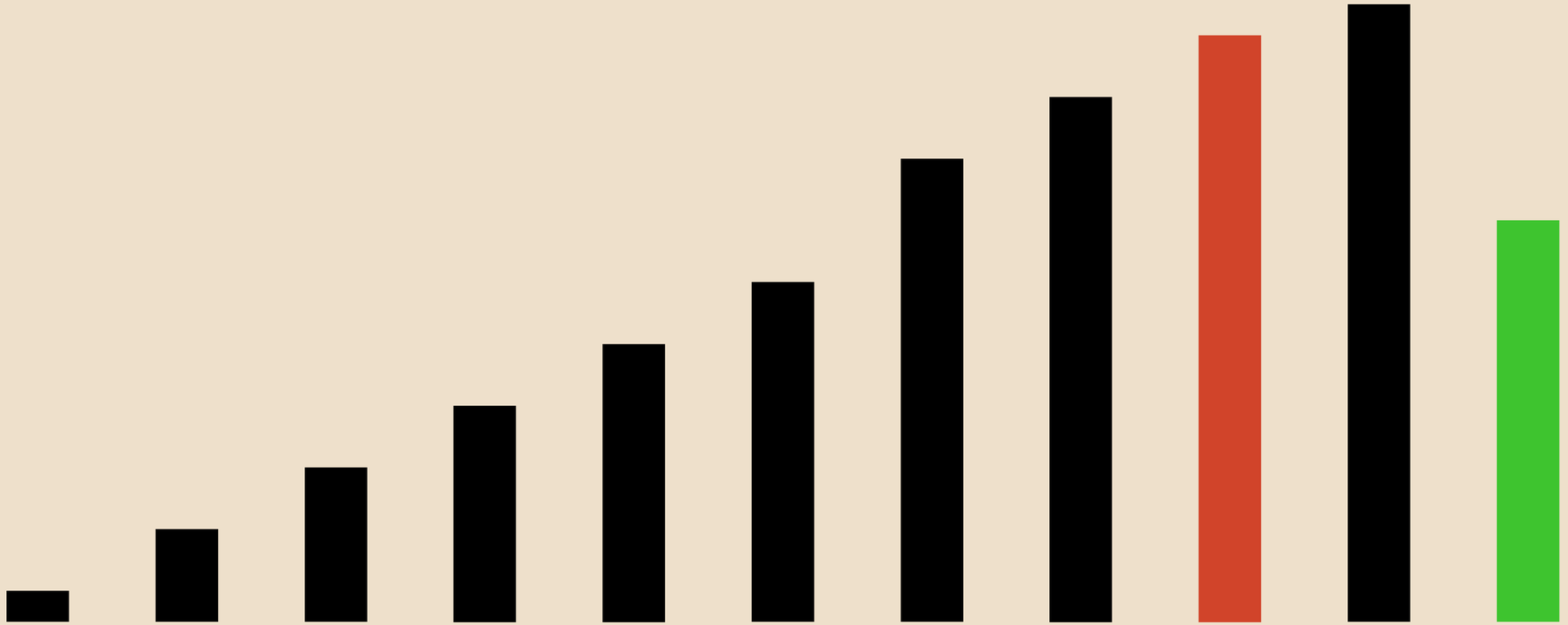


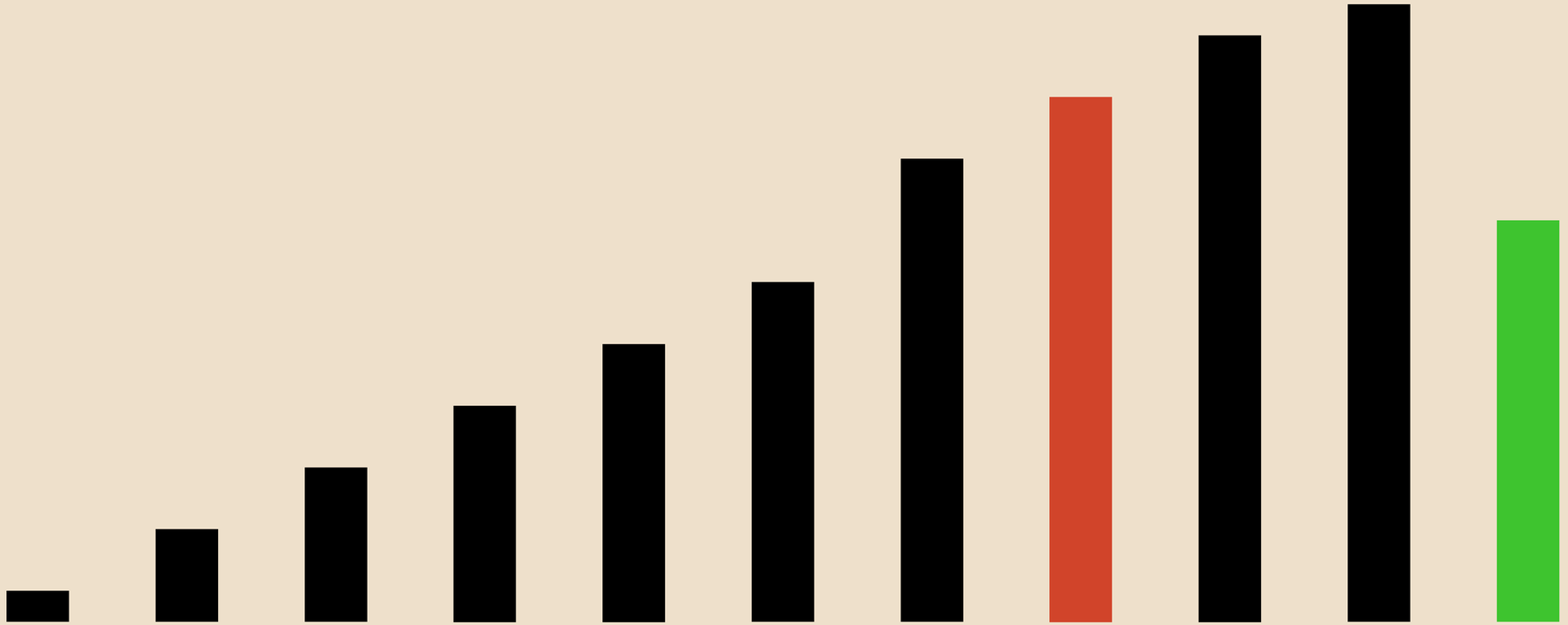


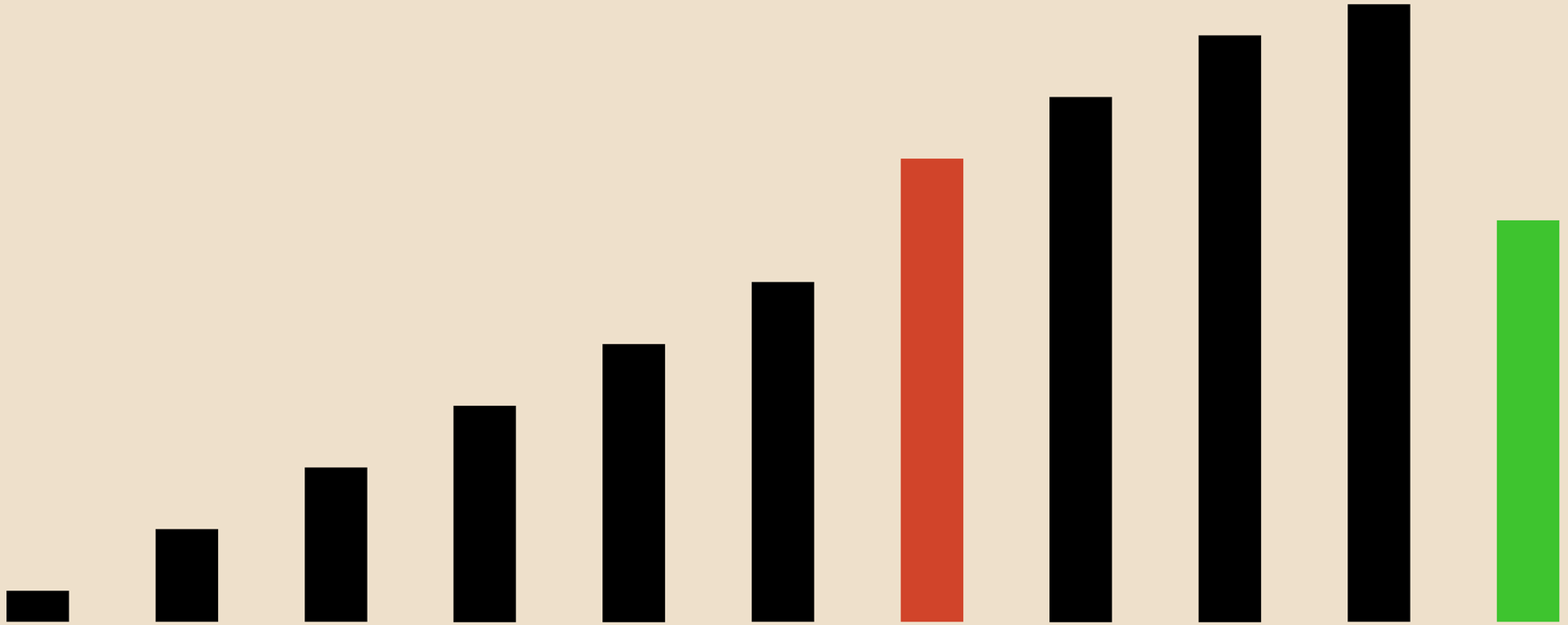


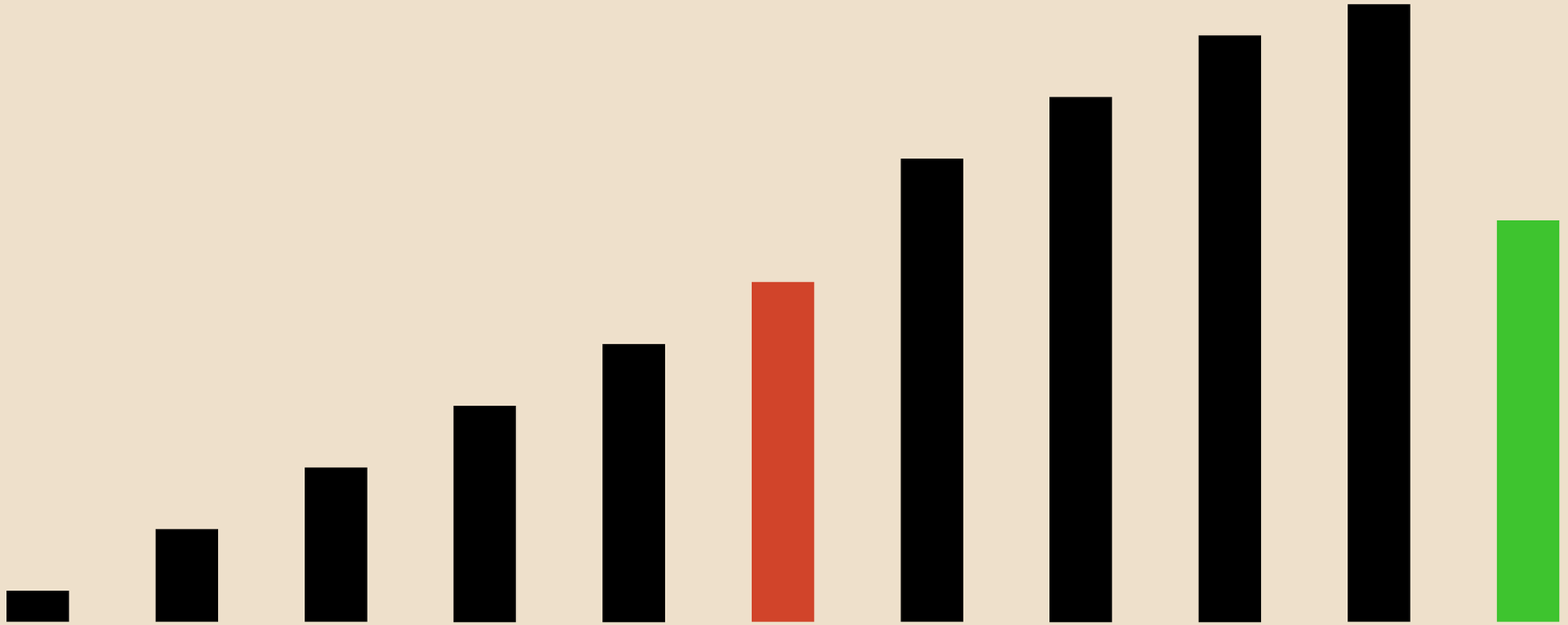


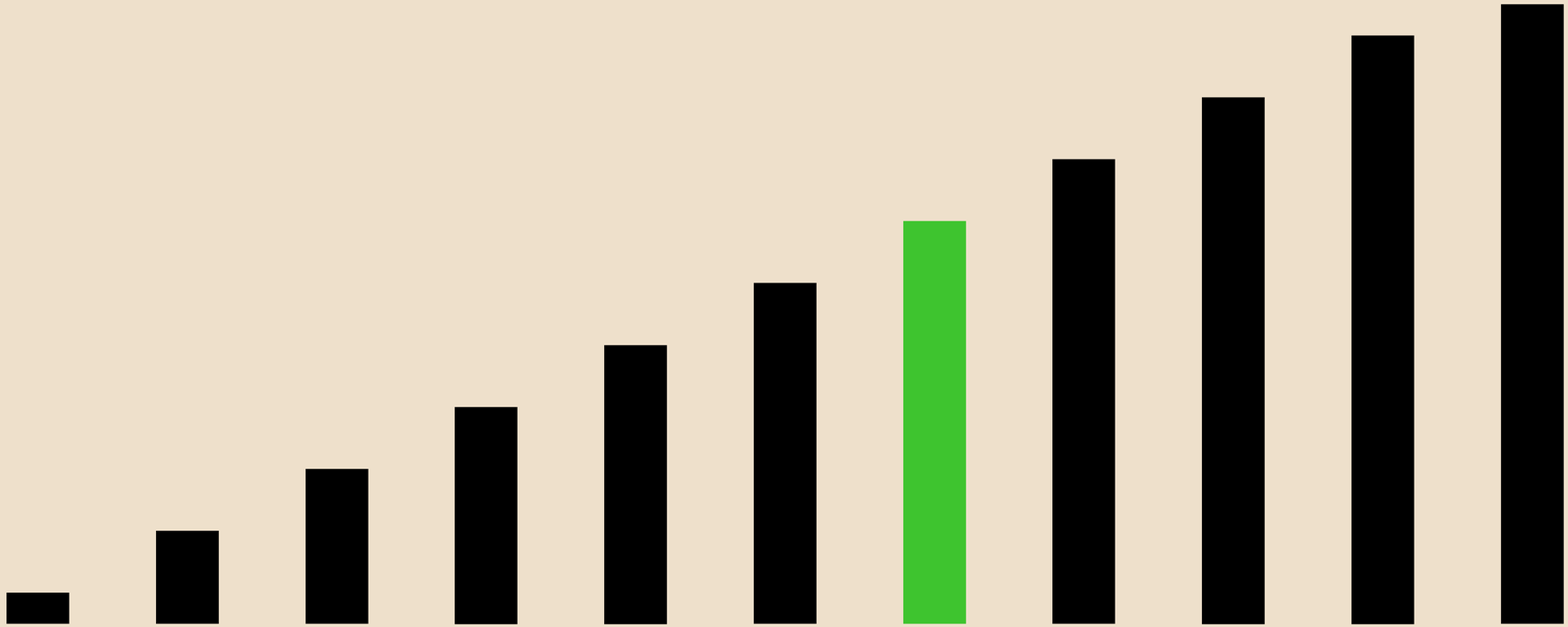


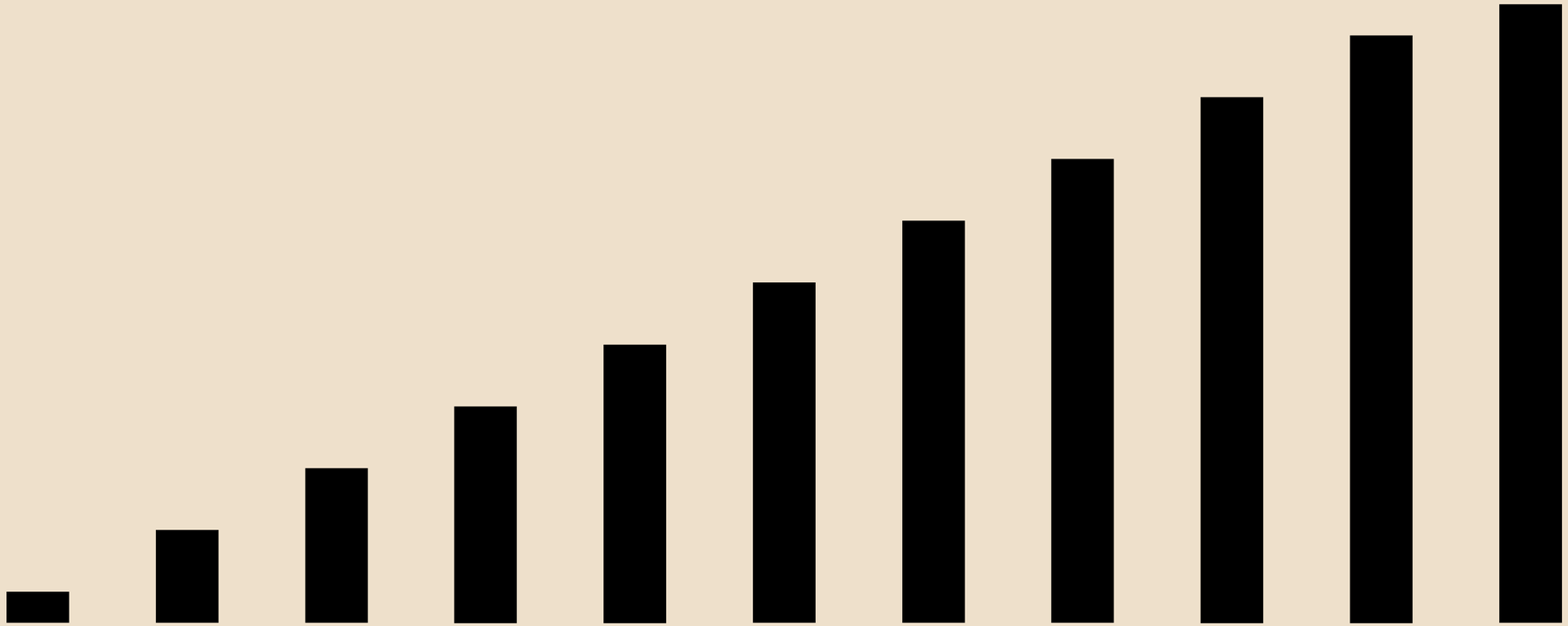








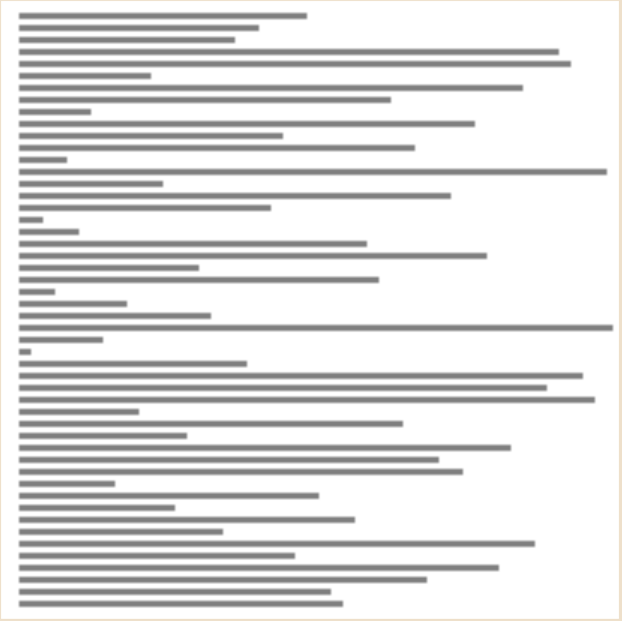




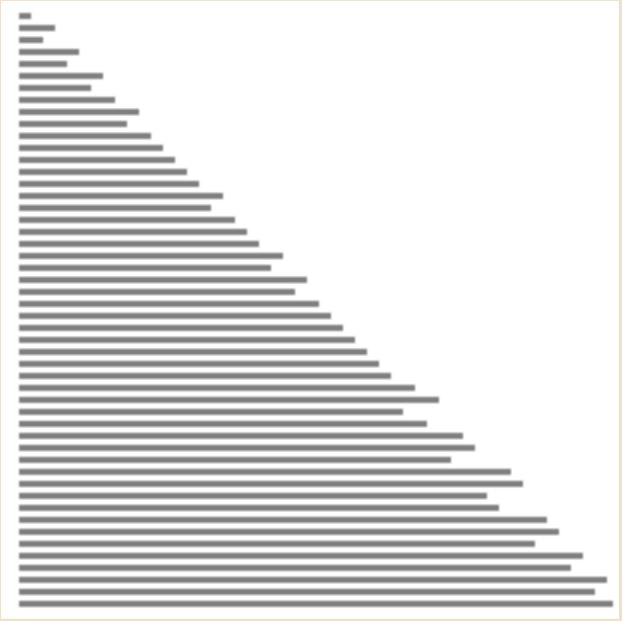
$$O(n^2)$$

$$\Omega(n)$$

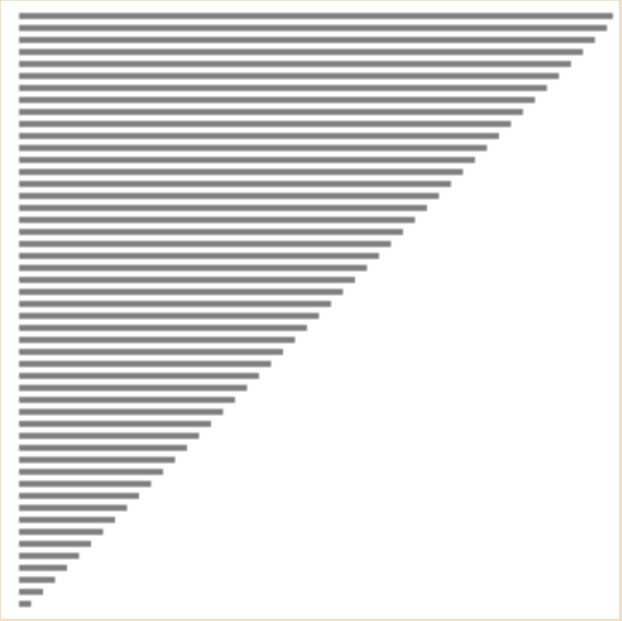
Random



Nearly Sorted



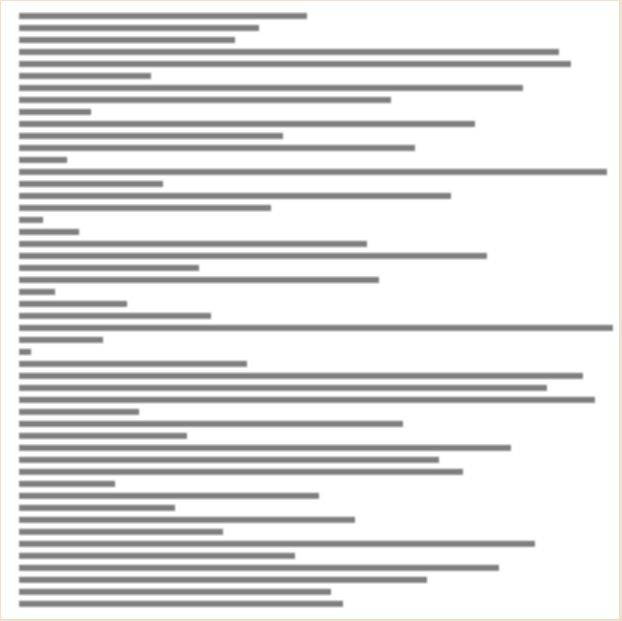
Reversed



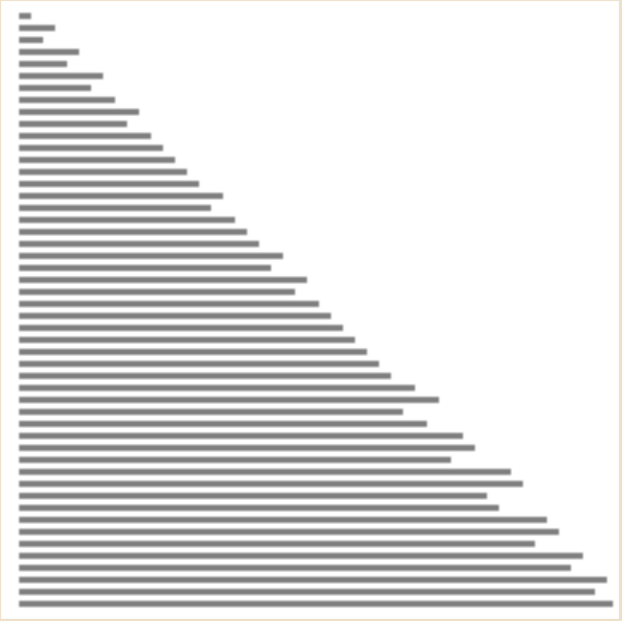
Few Unique

https://www.toptal.com/developers/sorting-algorithms/insertion-sort

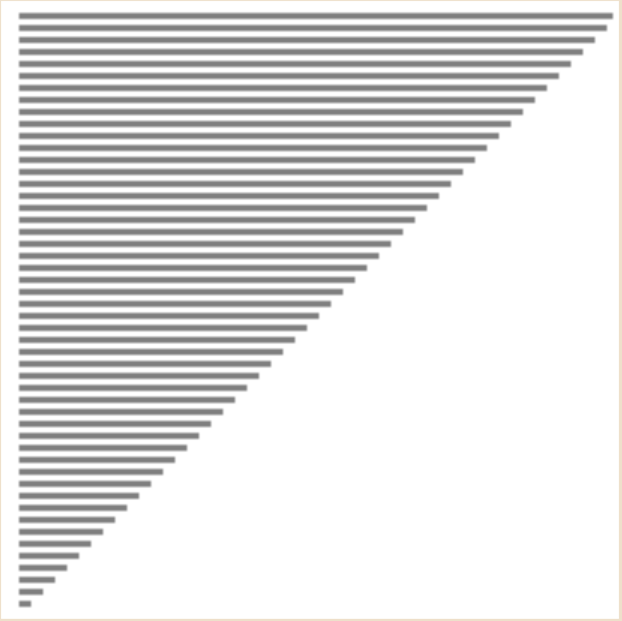
Random



Nearly Sorted

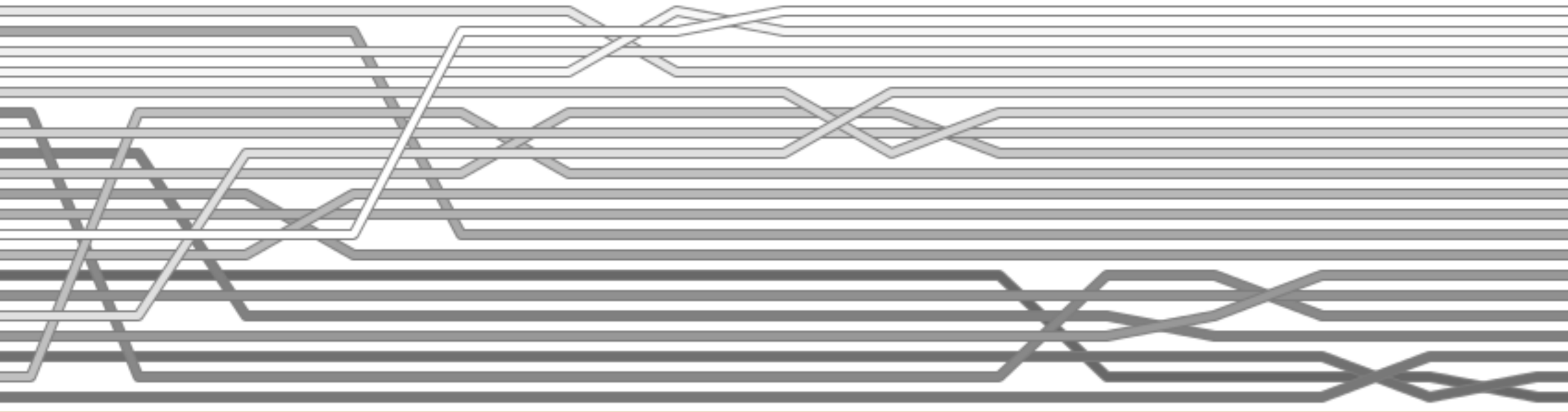


Reversed



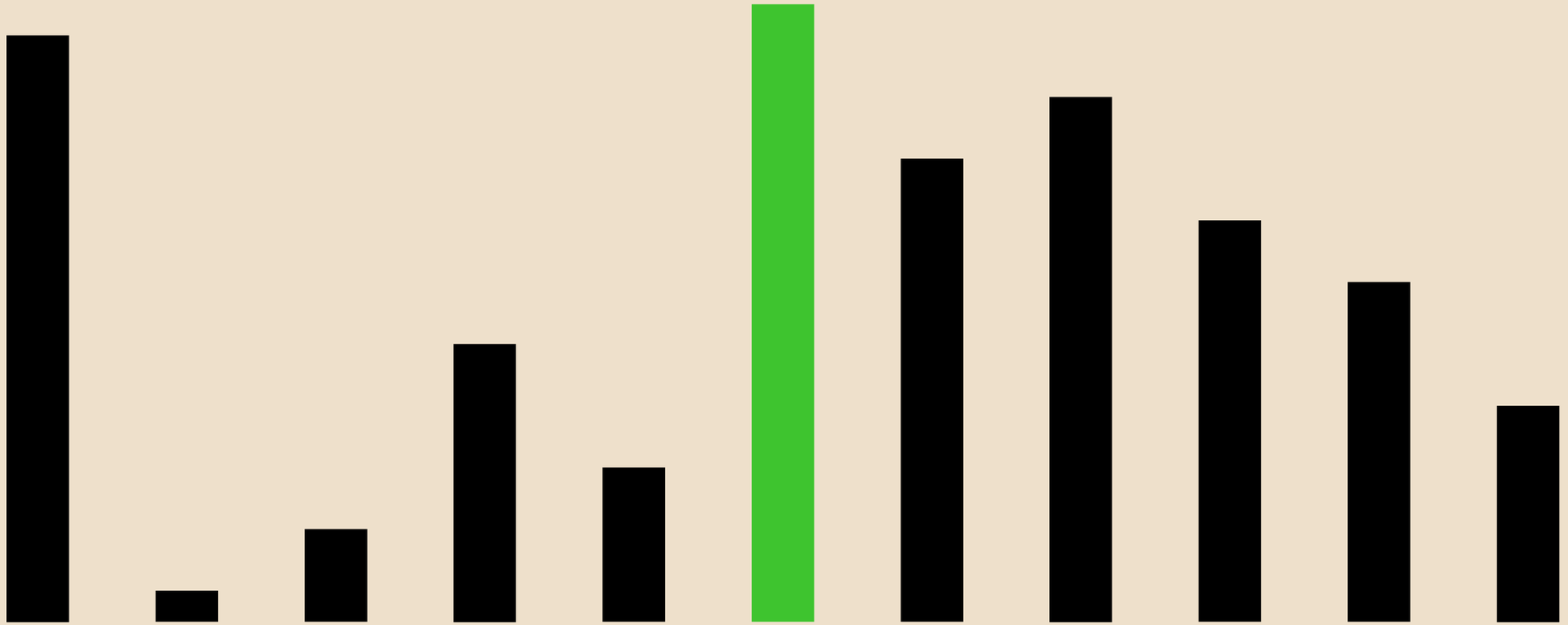
Few Unique

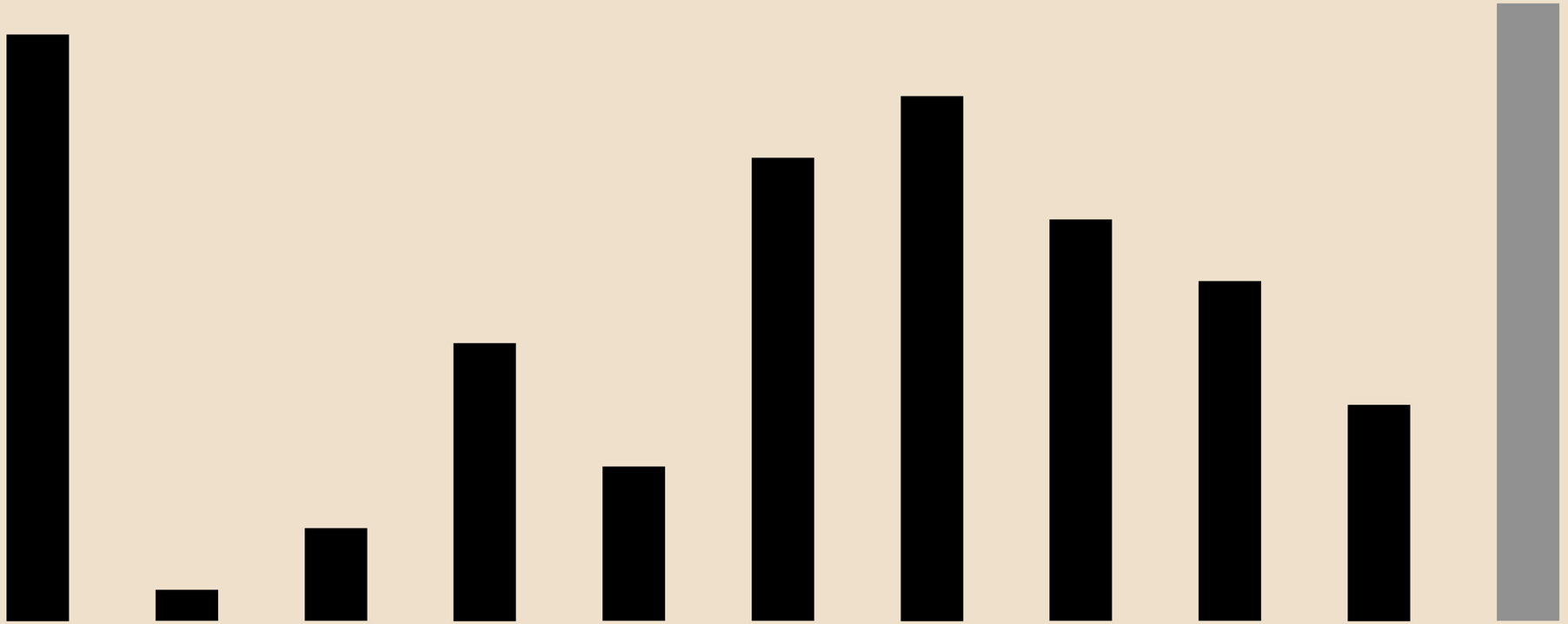
https://www.toptal.com/developers/sorting-algorithms/insertion-sort



Quicksort

$$O(n^2)$$

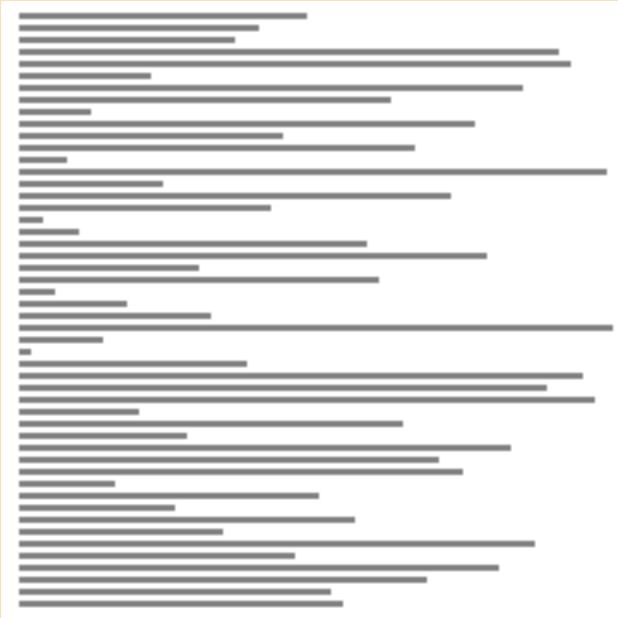




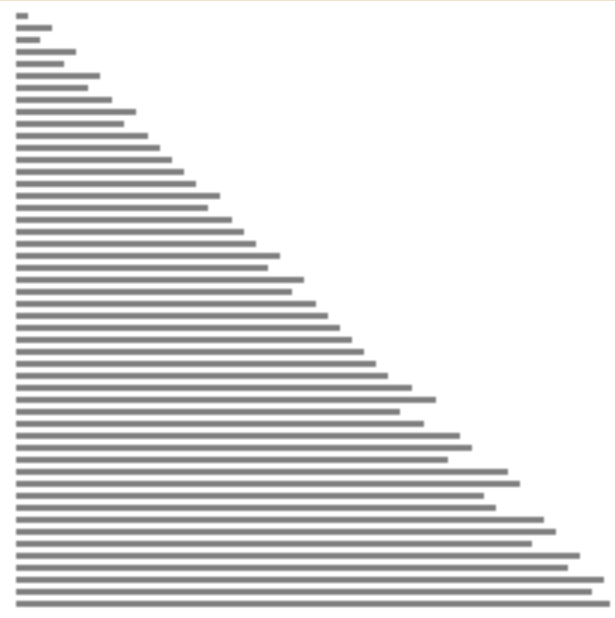
$$\mathcal{O}(n \log n)$$

$$\Omega(n \log n)$$

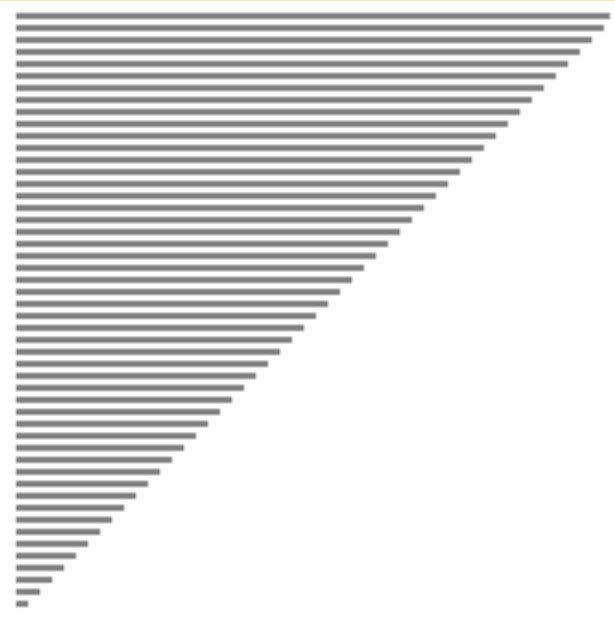
Random



Nearly Sorted



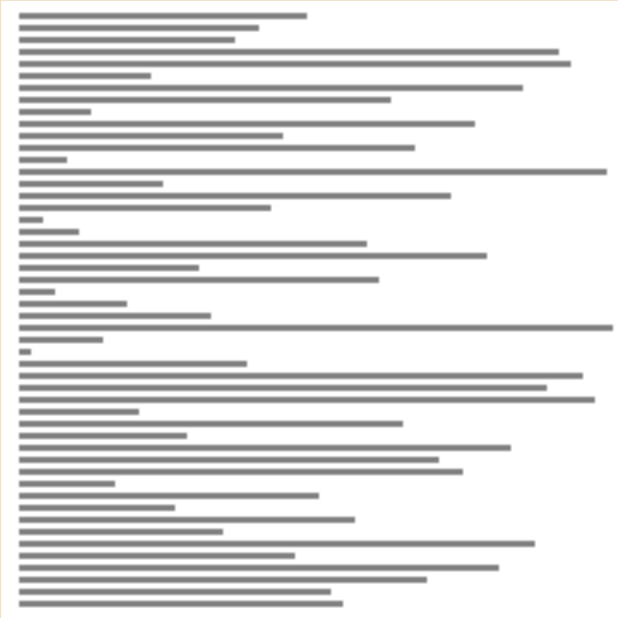
Reversed



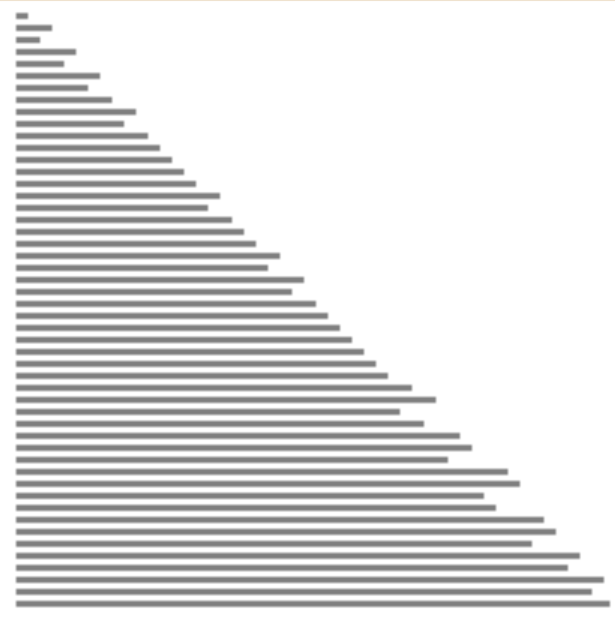
Few Unique

https://www.toptal.com/developers/sorting-algorithms/quick-sort

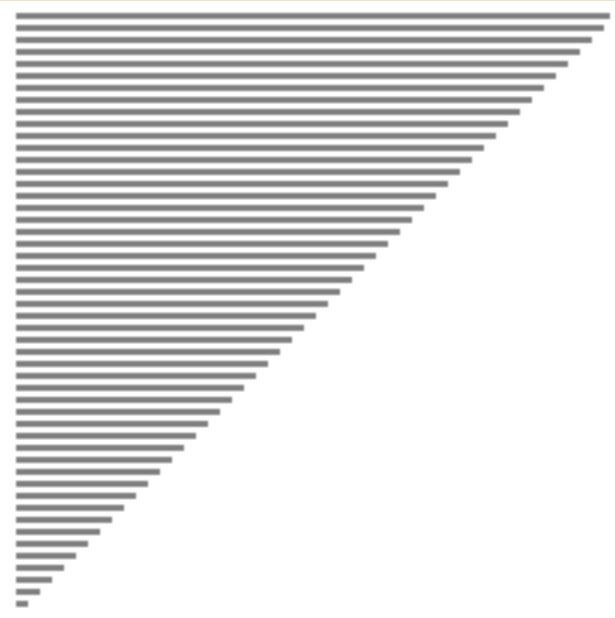
Random



Nearly Sorted

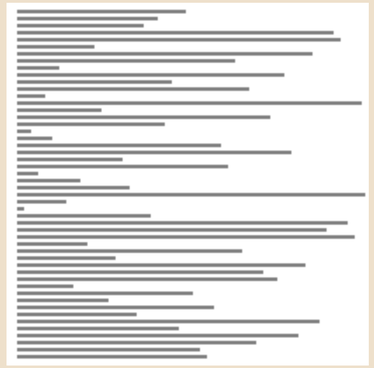
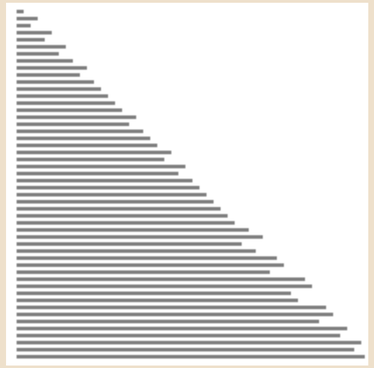
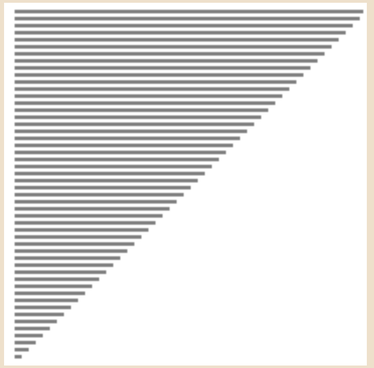
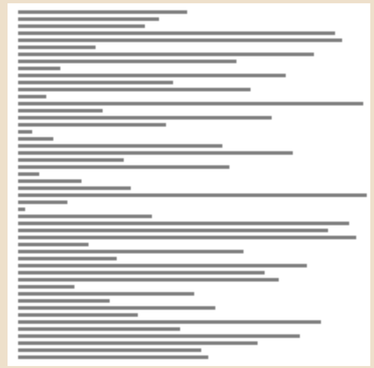
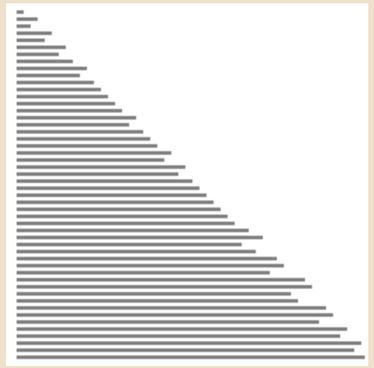
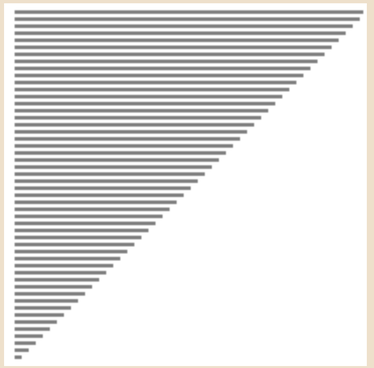
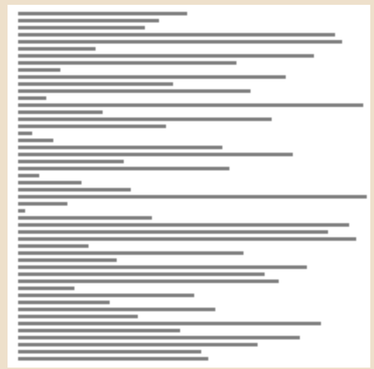
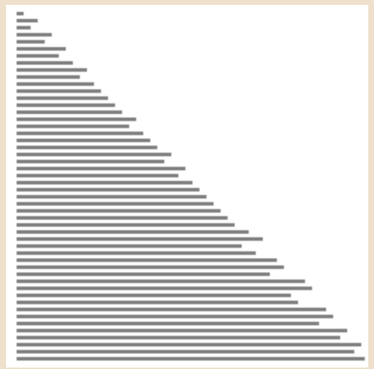
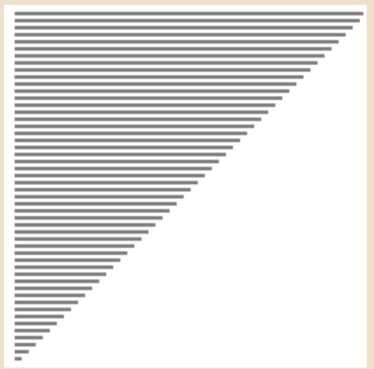


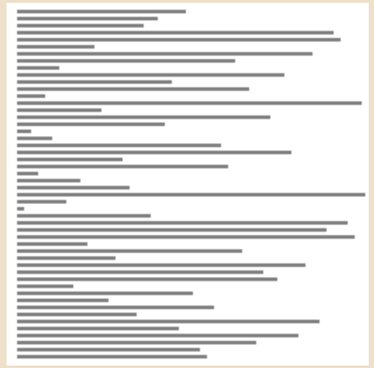
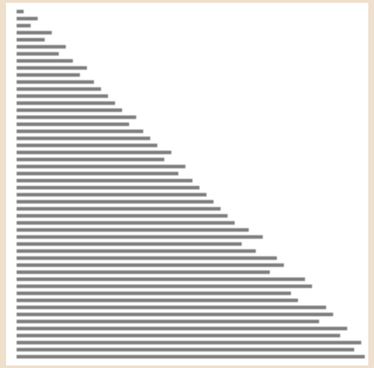
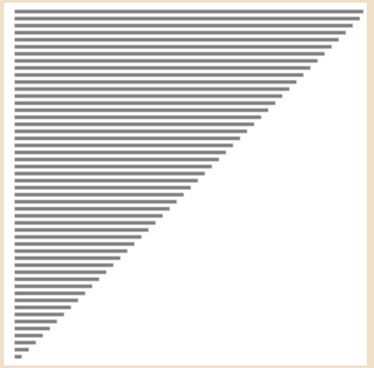
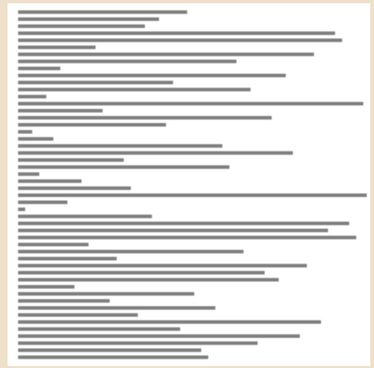
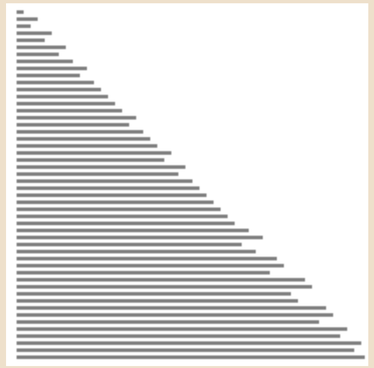
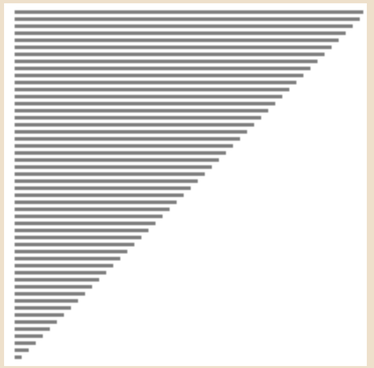
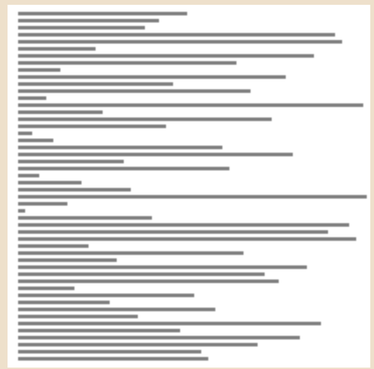
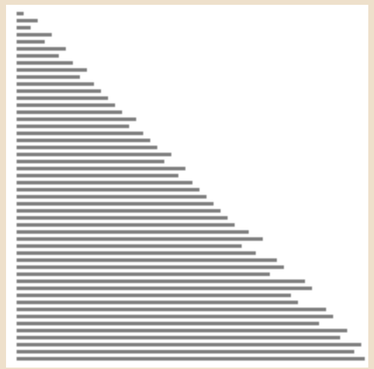
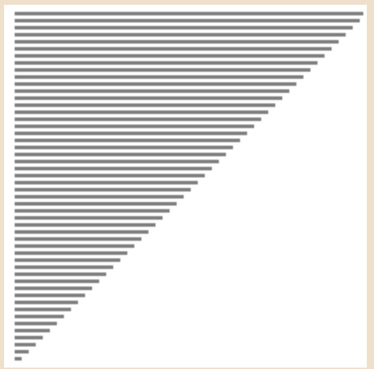
Reversed



Few Unique

https://www.toptal.com/developers/sorting-algorithms/quick-sort

	Random	Nearly Sorted	Reversed	Few Unique
Bubble				
Insertion				
Quick				

	Random	Nearly Sorted	Reversed	Few Unique
Bubble				
Insertion				
Quick				

Gracias.

Bibliography

- Pollice, G., Selkow, S., Heineman, G.T. (2008). *Algorithms in a Nutshell*. O'Reilly Media, Inc.

- Bhargava, A. (2016). *Grokking Algorithms*. Manning Publications.

- du Sautoy, M. (Presenter), Overton, P. (Producer). (2015, September 24). *The Secret Rules of Modern Living: Algorithms* [Television broadcast]. London, UK: BBC Four.

- Yukihiro, M. (1996). Array.sort!. Ruby [Software]. Retrieved 2017, March 27 from https://github.com/ruby/ruby/blob/47563655037ed453607de33b86fcc094878769ac/array.c#L2431-L2513.

Images

- Toptal. (2010). [Sorting Algorithms Animations]. Retrieved March 26, 2017, from https://www.toptal.com/developers/sorting-algorithms
- Cortesi, A. (2010). [sorting algorithm visualization]. Retrieved March 26, 2017, from http://sortvis.org/
- http://chocolate.wikia.com/wiki/Chocolate_Chip_Cookie
- https://www.babble.com/best-recipes/anatomy-of-a-chocolate-chip-cookie/
- http://thequotablekitchen.com/thick-chewy-chocolate-chip-cookies/
- http://www.beachwoodreporter.com/tv/what_i_watched_last_night_55.php
- http://asegrad.tufts.edu/academics/explore-graduate-programs/computer-science

Funography

- https://www.youtube.com/watch?v=kPRA0W1kECg
- https://www.youtube.com/watch?v=gOKVwRIyWdg
- Big O and Big Omega graphics: https://texblog.org/2014/06/24/big-o-and-related-notations-in-latex/