

Complexity and Big O Notation

how the number of steps an algorithm takes grows as the input grows



What complexity actually measures

- time complexity is how the number of basic steps an algorithm takes grows as the input size, called n , grows
- space complexity is how the amount of extra memory an algorithm needs grows as n grows, counted beyond the input itself
- Big O describes an upper bound on that growth, ignoring constant factors and smaller terms, so $O(2n + 5)$ is written simply as $O(n)$
- a step count is never the same as a running time in seconds, two algorithms with the same Big O can still run at different speeds on the same machine

extra

8	4	2	0	3
---	---	---	---	---

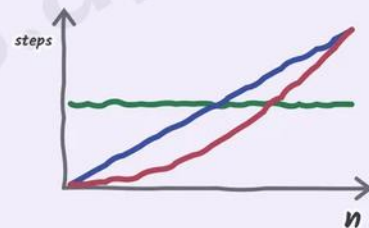
an extra array an algorithm allocates alongside its input, the size of a row like this is what space complexity counts

The common complexity classes, smallest to largest

*

Notation	Name	A concrete example
$O(1)$	constant	reading <code>arr[0]</code> , or reading an array's length field
$O(\log n)$	logarithmic	binary search on a sorted array
$O(n)$	linear	linear search, one pass to find the largest value
$O(n \log n)$	linearithmic	merge sort
$O(n^2)$	quadratic	bubble sort, selection sort, insertion sort's worst case

Three growth shapes on the same axes



flat is $O(1)$, it never rises, the straight line is $O(n)$, rising steadily, the upward curve is $O(n^2)$, rising far faster once n gets large