

Definition

Algorithm is a step-by-step set of instructions for completing a task.

Search Algorithms

Linear Search: Iterate across the array from left to right, searching for a specified element.

Worst case $O(n)$: the target element is the last element of the array or it doesn't exist at all. **Best case $\Omega(1)$:** the target element is the first element of the array.

Binary Search: Divide and conquer, reducing the search area by half each time, trying to find a target number. First, the array should be sorted.

Worst case $O(\log n)$: the target element will be found at the end of the last division or it won't be found at all. **Best case $\Omega(1)$:** the target element is the mid-point of the full array.

Sorting Algorithms

Bubble Sort: Swapping pairs of two elements: higher valued elements towards the right and the lower valued elements towards the left.

Worst case $O(n^2)$: the array is in the reversed order. **Best case $\Omega(1)$:** the array is already perfectly sorted and we don't need to make any swaps on the first run.

Selection Sort: Find the smallest unsorted element and add it to the end of the sorted list.

Worst case $O(n^2)$: iterate over each of the n elements of the array (to find the smallest unsorted element) → it's at the very end of the array and repeat this process n times, since only one element gets sorted on each pass. **Best case $\Omega(n^2)$:** exactly the same!

Merge Sort: Sort smaller arrays and then merge them in sorted order.

The best and the worst cases are the same: $n \log n$.

Big O notation

Big-O notation is a simplified analysis of an algorithm's efficiency. It attempts to answer two questions:

Q1: how much memory is needed?

Q2: how much time does it take to complete?

Recursion

A **recursive function** is one that, as part of its execution, invokes itself.

Every recursive function has two cases that could apply, given any input:

- The *base case* → terminate the recursive process
- The *recursive case* → the recursion occurs

Example of recursion: the factorial function

The **factorial** is found by multiplying n by all the whole numbers less than it.

Example:

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

In C:

```
int fact(int n)
{
    if (n == 1)
        return 1;
    else
        return n * fact(n-1);
}
```

Scenario A: The function calls itself in the 'else' clause → recursive case

Scenario B: The function terminates in the 'if' clause → base case



By dmytronoks

Published 5th August, 2022.

Last updated 4th August, 2022.

Page 1 of 1.

Sponsored by CrosswordCheats.com

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>