

Algorithmic Complexity

Big- Θ Notation

We compute the big- Θ of an algorithm by counting the number of iterations the algorithm *always* takes with an input of n . For instance, the loop in the pseudo code below will always iterate N times for a list size of N . The runtime can be described as $\Theta(N)$.

```
for each item in list:  
    print item
```

Asymptotic Notation

Asymptotic Notation is used to describe the running time of an algorithm – how much time an algorithm takes with a given input, n . There are three different notations: big O , big Theta (Θ), and big Omega (Ω). big- Θ is used when the running time is the same for all cases, big- O for the worst case running time, and big- Ω for the best case running time.

Adding Runtimes

When an algorithm consists of many parts, we describe its runtime based on the slowest part of the program.

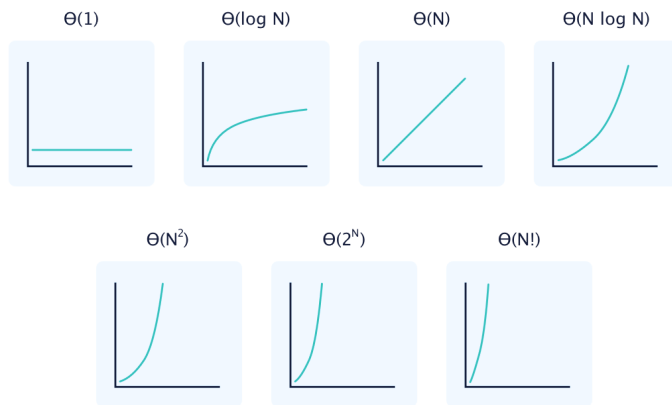
An algorithm with three parts has running times of $\Theta(2N) + \Theta(\log N) + \Theta(1)$. We only care about the slowest part, so we would quantify the runtime to be $\Theta(N)$. We would also drop the coefficient of 2 since when N gets really large, the multiplier 2 will have a small effect.

Algorithmic Common Runtimes

The common algorithmic runtimes from fastest to slowest are:

- constant: $\Theta(1)$
- logarithmic: $\Theta(\log N)$
- linear: $\Theta(N)$
- polynomial: $\Theta(N^2)$
- exponential: $\Theta(2^N)$
- factorial: $\Theta(N!)$

Common Runtimes



Big-O Notation

The Big-O notation describes the worst-case running time of a program. We compute the Big-O of an algorithm by counting how many iterations an algorithm will take in the worst-case scenario with an input of N . We typically consult the Big-O because we must always plan for the worst case. For example, $O(\log n)$ describes the Big-O of a binary search algorithm.

Big- Ω Notation

Big- Ω (Omega) describes the best running time of a program. We compute the big- Ω by counting how many iterations an algorithm will take in the best-case scenario based on an input of N . For example, a Bubble Sort algorithm has a running time of $\Omega(N)$ because in the best case scenario the list is already sorted, and the bubble sort will terminate after the first iteration.