Big Omega (Ω) and Big O (O)

6 min

Sometimes, a program may have a different

Preview: Docs Loading link description

runtime

for the best case and worst case. For instance, a program could have a best case runtime of Θ(1) and a worst case of Θ(N). We use a different notation when this is the case. We use big Omega or Ω to describe the best case and big O or O to describe the worst case. Take a look at the following

Preview: Docs Loading link description

pseudocode

that returns True if 12 is in the list and False otherwise:

```
Function with input that is a list of size N:
  For each value in the list:
    If value is equal to 12:
      Return True
  Return False
```

How many times will the loop iterate? Let's take a list of size 1000. If the first value in the list was 12, then the loop would only iterate once. However, if 12 wasn't in the list at all, the loop would iterate 1000 times. If the input was a list of size N, the loop could iterate anywhere from 1 to N times depending on where 12 is in the list (or if it's in the list at all). Thus, in the best case, it has a constant runtime and in the worst case it has a linear runtime.

There are many ways we could describe the runtime of this program:

- This program has a best case runtime of Θ(1).
- This program has a worst case runtime of Θ(N).
- This program has a runtime of Ω(1).
- This program has a runtime O(N).

You may be tempted to say the following:

- This program has a runtime of Θ(N).

However, this is not true because the program does not have a linear runtime in every case, only the worst case.

In fact, when describing runtime, people typically discuss the worst case because you should always prepare for the worst case scenario! **Often times, in technical interviews, they will only ask you for the big O of a program.**

Great! Now you know the different types of asymptotic notation and when to use which one! Now, let's delve into more complex program runtimes!

Instructions

Play the video to go through an example of finding the big Omega and big O runtime of a function.

size = N

```
function(List):
    For value in List:
        IF value is 12:
            Return True
    Return False
```

N	# OF Iterations: If first value is 12	# OF Iterations: If 12 is not in List
10	1	10
100	1	100
1000	1	1000
N	1	N
	Ω(1)	O(N)