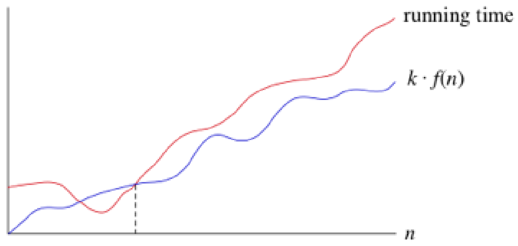
Big-Ω (Big-Omega) notation

Sometimes, we want to say that an algorithm takes *at least* a certain amount of time, without providing an upper bound. We use big-Ω notation; that's the Greek letter "omega."

If a running time is $\Omega(f(n))$, then for large enough n, the running time is at least $k \cdot f(n)$ for some constant k. Here's how to think of a running time that is $\Omega(f(n))$:



We say that the running time is "big-Ω of $f(n)$." We use big-Ω notation for **asymptotic lower bounds**, since it bounds the growth of the running time from below for large enough input sizes.

Just as $\Theta(f(n))$ automatically implies $O(f(n))$, it also automatically implies $\Omega(f(n))$. So we can say that the worst-case running time of binary search is $\Omega(\lg n)$. We can also make correct, but imprecise, statements using big-Ω notation. For example, just as if you really do have a million dollars in your pocket, you can truthfully say "I have an amount of money in my pocket, and it's at least 10 dollars," you can also say that the worst-case running time of binary search is $\Omega(1)$, because it takes at least constant time.

How helpful was this article to you?

- The article was great and really helpful.
- The article was clear and interesting but not what I was looking for.
- The article was okay but wasn't very interesting to me.
- The article was unclear or confusing.

Send feedback

Report a mistake in the article

Example:

The link in the third paragraph is broken.

Report a mistake in the article

Discuss the site

For general discussions about Khan Academy, click here.

Flag inappropriate posts

Here are posts to avoid making. If you do encounter them, flag them for attention from our Guardians.

Questions Tips & Thanks Top Recent

Ask a question...

I can't wrap my head around this: why it is correct to say that the for the binary search the running time is Ω(lgn)?

The 1st paragraph clearly says that Ω defines the *least* running time, and for the binary search the least running time is 1 guess, no matter how big is n, no?

The statement that Θ(f(n)) automatically implies Ω(f(n)) seems incomprehensible to me... What do I get wrong?

4 Votes • Comment • Flag 3 months ago by Georgiy Ivankin

When we use asymptotic notation, unless stated otherwise, we are talking about **worst-case running time**.
The worst-case running time for binary search is log(n).
Recall:
if f(n) is O(g(n)) this means that f(n) grows asymptotically no faster than g(n)
if f(n) is Ω(g(n)) this means that f(n) grows asymptotically no slower than g(n)
if f(n) is Θ(g(n)) this means that f(n) grows asymptotically at the same rate as g(n)
As a result:
if f(n) is Θ(g(n)) it is growing at the same rate as g(n). So we can... (more)

• 21 Votes • 2 Comments • Flag 3 months ago by Cameron

Show all 2 answers • Answer this question

If we know the big-Theta of a function, is there a reason that we would want to know a different big-O or big-Omega for the function? It just appears that if you have the big-Theta you would know the best big-O and big-Omega so I don't know why you would want to know these values unless they were significantly easier to find than the big-Theta. Does the big-Theta give the best big-O and big-Omega possible?

3 Votes • Comment • Flag 3 months ago by John Phipps

Typically, if we design an algorithm and perform some quick analysis on it we find a reasonable O value (it doesn't have to be tight, but it shouldn't be too loose).
Often, that's all we care about.
Sometimes, we then find a reasonable Ω value (again it doesn't have to be tight, but it shouldn't be too loose).
Then, if it is critical to have tight bounds, or if the algorithm is easy to analyze we tighten up the bounds on O and Ω until they are the same and then we have Θ.

A typical scenario is:... (more)

7 Votes • 6 Comments • Flag 3 months ago by Cameron

Show all 4 answers • Answer this question

Isn't big omega infinity?

2 Votes • Comment • Flag

3 months ago by ☺ ☻ ☺Natth4545☺ ☻ ☺ <See Bio>.

The time it takes to perform some function as n increases may go to infinity. Big omega tells you how quickly that time increases.

3 Votes • Comment • Flag 3 months ago by John Sutton

Show all 3 answers • Answer this question

I am a little bit confused with the Quiz:Asymptotic notation.

The question is finding the asymptotic relationship between n^k and c^n. k >= 1 and c > 1 are constants. I understand the values of c^n grows more rapidly than n^k. But then why is n^k = O(c^n) and not n^k = Ω(c^n)? Since n^k is always smaller than c^n, shouldn't n^k equal the group of the least possible values of c^n (Ω(c^n)) instead of the group of the maximum possible values(O(c^n))?

The hint in the quiz states "If we... (more)

2 Votes • Comment • Flag 3 days ago by In Kwon Choi

Short Answer

if f(n) is O(g(n)) this means that f(n) grows asymptotically no faster than g(n)
if f(n) is Ω(h(n)) this means that f(n) grows asymptotically no slower than h(n)
if f(n) is Θ(r(n)) this means that f(n) grows asymptotically at the same rate as r(n)

So we could interpret this as f(n) is somewhere "between" g(n) and h(n).
f(n) is no "lower" than h(n) (our "lower limit"), and f(n) is no "higher" than g(n) (our "upper limit").

Long Answer

I'll start at the basics and work up to what f(n) = O(... (more)

1 Vote • Comment • Flag 3 days ago by Cameron

Answer this question...

Hello!

abuse
- disrespectful or offensive
- an advertisement

not helpful
- low quality
- not about the video topic
- soliciting votes or seeking badges
- a homework question
- a duplicate answer
- repeatedly making the same post

wrong category
- a tip or thanks in Questions
- a question in Tips & Thanks
- an answer that should be its own question

about the site
- a question about Khan Academy (Visit our FAQ)
- a post about badges
- a technical problem with the site (Report a problem)
- a request for features

I could get the idea of the upper and lower bounds, using O and Ω, but I didn't get the following sentence:
"you can also say that the worst-case running time of binary search is Ω(1), because it takes at least constant time."
as we know that the worst case in binary search for n-size array will always be log(n)+1 tries.
and in the known case, like the worst case, O(f(n)) = Ω(f(n)) = Θ(f(n)) ; f(n) = ln(n);

and the complexity of the binary search **in worst case** will never be constant unless... (more)

1 Vote • Comment • Flag 2 months ago by Hayyan

if f(n) is Ω(g(n)) this means that f(n) grows asymptotically no slower than g(n)

log(n) is Ω(1) since log(n) grows asymptotically no slower than 1
log(n) is also Ω(log(n)) since log(n) grows asymptotically no slower than log(n)

So. saying that the worst case running time for binary search is Ω(1) , or Ω(log(n)) are both correct. However, Ω(log(n)) provides a more useful lower bound than Ω(1).

2 Votes • 2 Comments • Flag 2 months ago by Cameron

Show all 2 answers • Answer this question