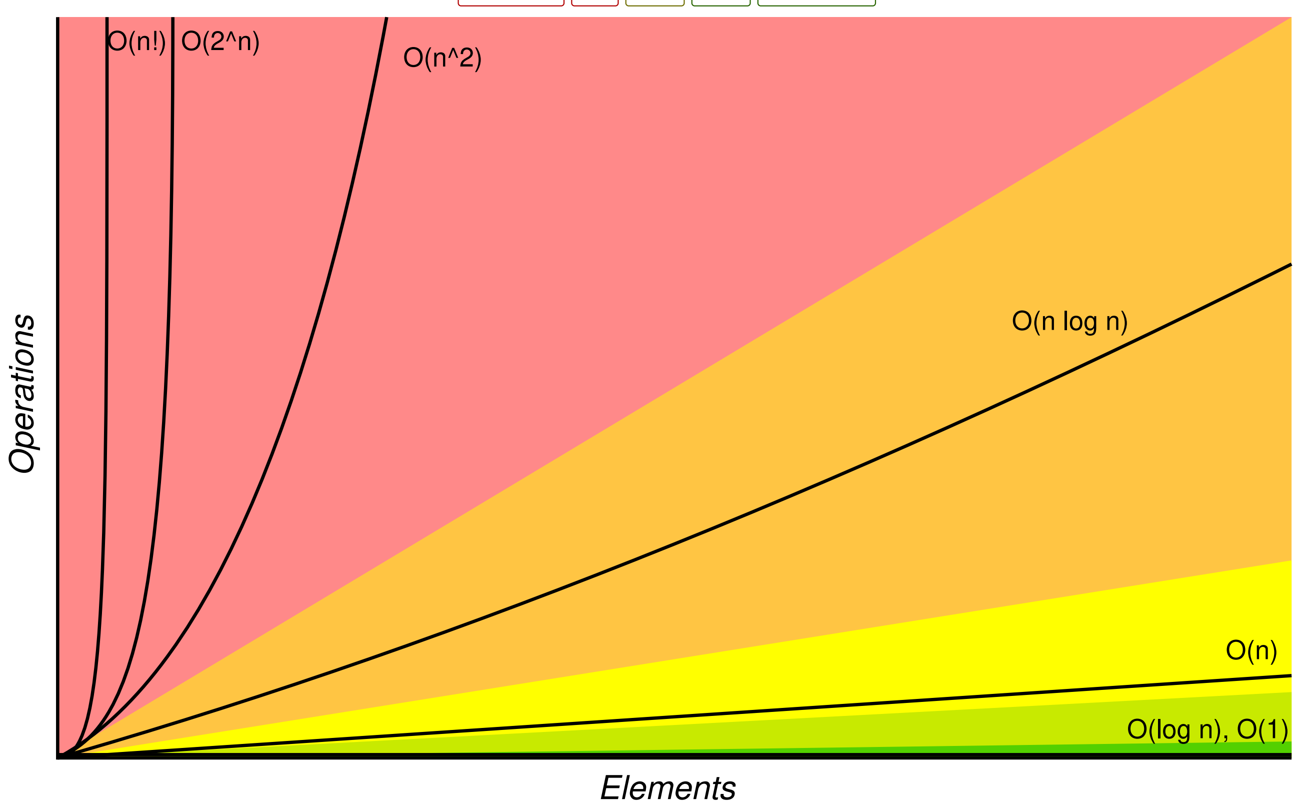
Big-O Cheat Sheet Download PDF

Know Thy Complexities!

Hi there! This webpage covers the space and time Big-O complexities of common algorithms used in Computer Science. When preparing for technical interviews in the past, I found myself spending hours crawling the internet putting together the best, average, and worst case complexities for search and sorting algorithms so that I wouldn't be stumped when asked about them. Over the last few years, I've interviewed at several Silicon Valley startups, and also some bigger companies, like Google, Facebook, Yahoo, LinkedIn, and Uber, and each time that I prepared for an interview, I thought to myself "Why hasn't someone created a nice Big-O cheat sheet?". So, to save all of you fine folks a ton of time, I went ahead and created one. Enjoy! - Eric

Big-O Complexity Chart

Horrible Bad Fair Good Excellent



Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Learn More

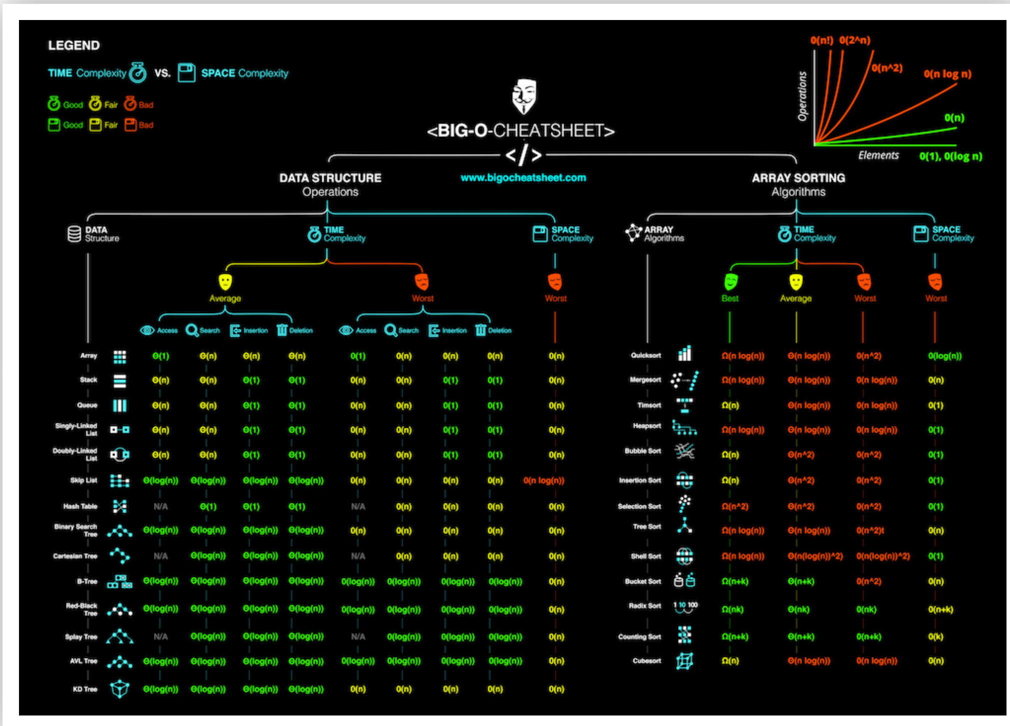
[Cracking the Coding Interview: 150 Programming Questions and Solutions]

[Introduction to Algorithms, 3rd Edition]

[Data Structures and Algorithms in Java (2nd Edition)]

[High Performance JavaScript (Build Faster Web Application Interfaces)]

Get the Official Big-O Cheat Sheet Poster



Contributors

Eric Rowell | Quentin Pleple | Michael Abed | Nick Dizazzo | Adam Forsyth | Felix Zhu | Jay Engineer
Josh Davis | Nodir Turakulov | Jennifer Hamon | David Dorfman | Bart Massey | Ray Pereda | Si Pham
Mike Davis | mcverry | Max Hoffmann | Bahador Saket | Damon Davison | Alvin Wan | Alan Briolat
Drew Hannay | Andrew Rasmussen | Dennis Tsang | Vinnie Magro | Adam Arold | Alejandro Ramirez
Aneel Nazareth | Rahul Chowdhury | Jonathan McElroy | steven41292 | Brandon Amos | Joel Friedly
Casper Van Gheluwe | Eric Lefevre-Ardant | Oleg | Renfred Harper | Piper Chester | Miguel Amigot | Apurva K
Matthew Daronco | Yun-Cheng Lin | Clay Tyler | Orhan Can Ozalp | Ayman Singh | David Morton
Aurelien Ooms | Sebastian Paaske Torholm | Koushik Krishnan | Drew Bailey | Robert Burke

Make this Page Better

[Edit these tables!]

551 Comments

1 Login ▾

G

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

368 Share

Best Newest Oldest

Michael Mitchell

11 years ago

This is great. Maybe you could include some resources (links to khan academy, mooc etc) that would explain each of these concepts for people trying to learn them.

416 1 Reply Share ›

Amanda Harlin ➦ Michael Mitchell

11 years ago

Yes! Please & thank you

91 1 Reply Share ›

A

Asim Ahmad ➦ Amanda Harlin

7 years ago

Can you Explain the Above Algorithm.??

0 13 Reply Share ›

A

Anonymous ➦ Asim Ahmad

6 years ago

Mr.
you can learn these algorithms easily in google by searching
Don't always ask or wait for someone to post things for you go out and search on internet
You will find everything you want to learn

If you are a beginner in Data structures and algorithms then visit mycodeschool youtube channel and learn there
if you want more then email me at rise.d1105@gmail.com I will help you as much as I can

54 17 Reply Share ›

ichiraku demigod ➦ Anonymous

9 months ago

this offer still goin? i got finals soon

1 0 Reply Share ›

Biribiri ➦ ichiraku demigod

5 months ago

Lmao

1 0 Reply Share ›

S

Safin ➦ Anonymous
3 years ago edited

@Anonymous Your email isnt working.

0 0 Reply Share ›

Careerdrill ➦ Anonymous
4 years ago

https://www.careerdrill.com/

0 3 Reply Share ›

Trey Huffine ➦ Asim Ahmad
4 years ago

https://skilled.dev provides a detailed explanation

2 1 Reply Share ›

Cam Cecil ➦ Michael Mitchell
11 years ago

This explanation in 'plain English' helps: http://stackoverflow.com/qu...

39 1 Reply Share ›

Richard Wheatley ➦ Cam Cecil
9 years ago

this is plain english.

15 3 Reply Share ›

T

Tomás Gemes | xygrr ➦ Richard Wheatley
3 years ago

No, this is spooky english to some you know

1 0 Reply Share ›

Arjan Nieuwenhuizen ➦ Michael Mitchell
11 years ago edited

Here are the links that I know of.

#1) http://aduni.org/courses/al...
#2) http://ocw.mit.edu/courses/...
#3) https://www.udacity.com/cou...

probably as good or maybe better # 2, but I have not had a chance to look at it.
http://ocw.mit.edu/courses/...

Sincerely,
Arjan

p.s.
https://www.coursera.org/co...
This course has just begun on coursera (dated 1 July 2013), and looks very good.

23 0 Reply Share ›

fireheron ↪ Arjan Nieuwenhuizen
11 years ago

Thank you Arjan. Espaecially the coursera.org one ;-)

5 0 Reply Share ›

@hangtwentyy ↪ fireheron
10 years ago

also this! http://opendatastructures.org

9 0 Reply Share ›

Y

yth ↪ @hangtwentyy
9 years ago

thank you for sharing this.

2 0 Reply Share ›

Eduardo Sánchez ↪ Michael Mitchell
9 years ago

There is an amazing tutorial for Big O form Derek Banas in Youtube, that guy is amazing explaining!!!

see more

13 1 Reply Share ›

Sudhanshu Mishra ↪ Eduardo Sánchez
8 years ago

Cool! This is a more than adequate introduction! Thanks a ton for sharing!

3 0 Reply Share ›

M

Mohammed Hameed ↪ Eduardo Sánchez
5 years ago

Thanks...

0 0 Reply Share ›

CodeMunkey ↪ Michael Mitchell
8 years ago

Not sure if this helps, but here's a more visual learner for some of these algorithms - if you're interested.
http://visualgo.net

6 0 Reply Share ›

Divyendra Patil Michael Mitchell
7 years ago

www.codenza.us

2 0 Reply Share ›

Ahmed KHABER Michael Mitchell
3 years ago

https://classroom.udacity.c...

1 0 Reply Share ›

Trey Huffine Michael Mitchell
4 years ago

https://skilled.dev/ for a detailed explanation on Big O

1 0 Reply Share ›

nate lipp Michael Mitchell
7 years ago

This is a well put together introduction
https://www.interviewcake.c...

1 0 Reply Share ›

S

Sam Michael Mitchell
2 years ago

Here's another great article on the most common sorting algorithms in python:

https://educashare.com/most...

0 0 Reply Share ›

P

Pranati B Michael Mitchell
2 years ago

This post is really great. If you are looking for a more detailed explanation with code, you can check out this link.
https://botbark.com/2023/01...

0 0 Reply Share ›

Abby Jones Michael Mitchell
5 years ago

Fabulous idea!

0 0 Reply Share ›

Jeshika Morneau Michael Mitchell
6 years ago

Or you could have supplied them in your comment instead.

0 0 Reply Share ›

N

Nhập Hàng Ngoại ➦ Michael Mitchell

7 years ago

http://fashionfor.life/t-sh...

see more

0 16 Reply Share ›

Blake Jennings

11 years ago

i'm literally crying

102 0 Reply Share ›

F

friend ➦ Blake Jennings

7 years ago

you give me a big o

6 0 Reply Share ›

Gokce Toykuyu

12 years ago

Could we add some tree algorithms and complexities? Thanks. I really like the Red-Black trees ;)

91 0 Reply Share ›

ericdrowell Mod ➦ Gokce Toykuyu

12 years ago

Excellent idea. I'll add a section that compares insertion, deletion, and search complexities for specific data structures

31 0 Reply Share ›

Y

yash bedi ➦ ericdrowell

8 years ago

its been 4 years you haven't added that section :)

0 1 Reply Share ›

Elliot Géhin ➦ yash bedi

8 years ago

It's up there Yash, bottom of the first table

1 0 Reply Share ›

A

Ahsan Sukamuljo ➜ yash bedi

4 years ago

fuck off asshole

0 7 Reply Share ›

Valentin Stanciu

11 years ago

1. Deletion/insertion in a single linked list is implementation dependent. For the question of "Here's a pointer to an element, how much does it take to delete it?", single-linked lists take O(N) since you have to search for the element that points to the element being deleted. Double-linked lists solve this problem.
2. Hashes come in a million varieties. However with a good distribution function they are O(logN) worst case. Using a double hashing algorithm, you end up with a worst case of O(loglogN).
3. For trees, the table should probably also contain heaps and the complexities for the operation "Get Minimum".

63 0 Reply Share ›

Avatar This comment was deleted.

Miguel ➜ Guest

10 years ago

You still have to find the position in the list, which can only be done linearly.

8 0 Reply Share ›

G

Guest ➜ Miguel

10 years ago edited

You still have to find the position in the list, which can only be done linearly.

3 0 Reply Share ›

Avatar This comment was deleted.

G

Guest ➜ Guest

10 years ago

No need to find the position if you can delete it as Alexis mentioned

2 1 Reply Share ›

P

Pingu App ➜ Guest

10 years ago

What if B is the last element in the list?
How would B's predecesor know that its next field should point to NULL and not to a futurely invalid memory address?

3 0 Reply Share ›

Avatar This comment was deleted.

O

Oscar Martinez Sanchez ➜ Guest

4 years ago

that's confusing, so if u want to delete the last element of the list, to maintain the delete method working u shoudl receive two params, node to delete and previous node?

0 0 Reply Share ›

pvlbzn Guest
8 years ago edited

And you will introduce the side effect which will be hell to debug. Consider:
Singly linked list { A:1, B:2, C:3, D:4 } where is X:Y, y is a value, function `delete` which works as you described, function `get` which returns pointer to the node by index.

```
// Take needed node C
node_t* node = get(list, 2)
print(node->value) // prints 3

// Delete B
delete(list, 1)

// Try to access C again
print(node->value) // well, enjoy your O(1)
```

Don't.

2 1 Reply Share ›

Avatar This comment was deleted.

Avatar This comment was deleted.