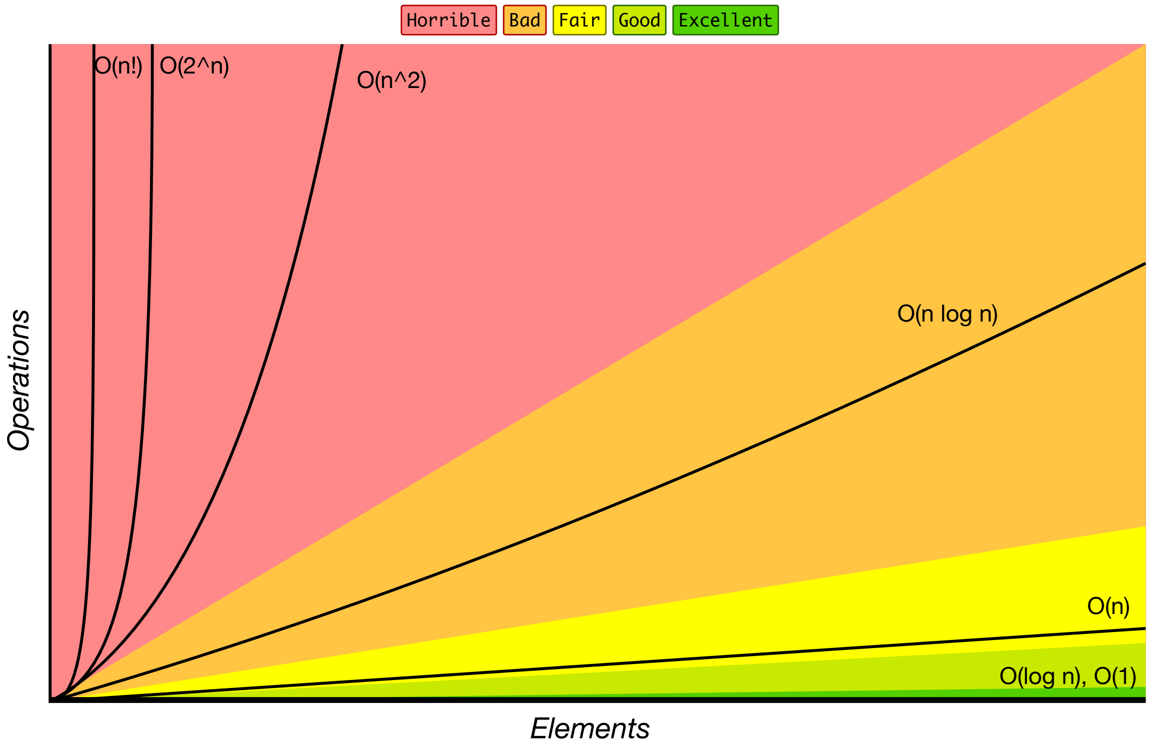
Know Thy Complexities!

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 eBay, 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

If you're trying to catch them all, you might also check out the Pokemon Go Evolution Chart.

Big-O Complexity Chart



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

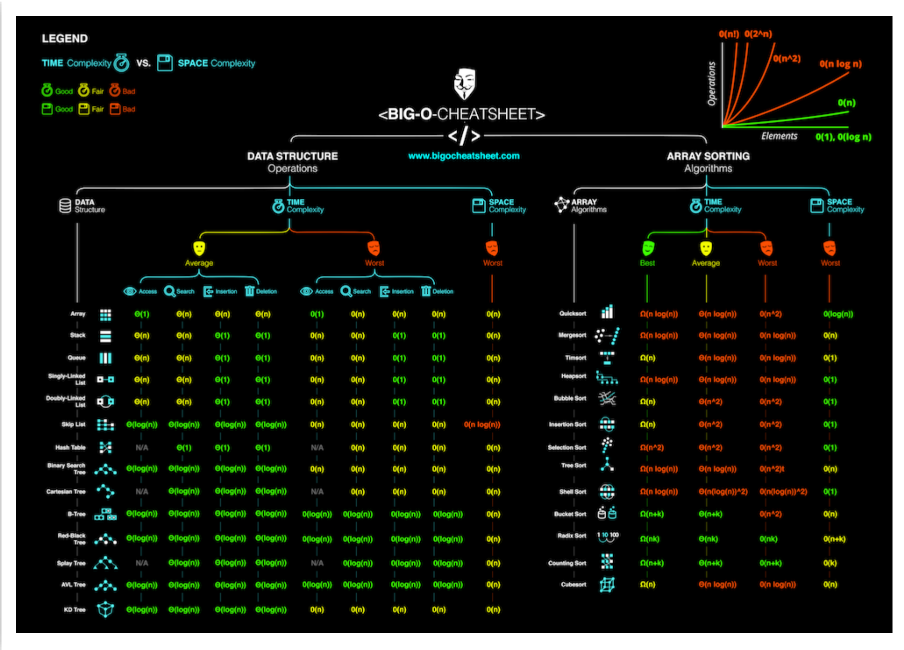
Cracking the Coding Interview: 150 Programming Questions and Solutions

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!

405 Comments Big-O Cheat Sheet Jae Hyun Lim

Recommend 224 Share Sort by Best

Join the discussion…

Michael Mitchell • 4 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.
303 • Reply • Share ›

Amanda Harlin → Michael Mitchell • 4 years ago
Yes! Please & thank you
67 • Reply • Share ›

Cam Cecil → Michael Mitchell • 4 years ago
This explanation in 'plain English' helps: http://stackoverflow.com/qu...
31 • Reply • Share ›

Richard Wheatley → Cam Cecil • 2 years ago
this is plain english.
3 • Reply • Share ›

Arjan Nieuwenhuizen → Michael Mitchell • 4 years ago

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.
15 ^ | v • Reply • Share ›

fireheron → Arjan Nieuwenhuizen • 4 years ago
Thank you Arjan. Espaecially the coursera.org one ;-)
3 ^ | v • Reply • Share ›

@hangtwentyy → fireheron • 3 years ago
also this! http://opendatastructures.org
6 ^ | v • Reply • Share ›

yth → @hangtwentyy • 3 years ago
thank you for sharing this.
1 ^ | v • Reply • Share ›

Eduardo Sánchez → Michael Mitchell • 2 years ago
There is an amazing tutorial for Big O form Derek Banas in Youtube, that guy is amazing explaining!!!

6 ^ | v • Reply • Share ›

Sudhanshu Mishra → Eduardo Sánchez • a year ago
Cool! This is a more than adequate introduction! Thanks a ton for sharing!
^ | v • Reply • Share ›

nate lipp → Michael Mitchell • 13 days ago
This is a well put together introduction
https://www.interviewcake.c...
^ | v • Reply • Share ›

CodeMunkey → Michael Mitchell • a year 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
^ | v • Reply • Share ›

Nhập Hàng Ngoại ➦ Michael Mitchell • 5 months ago
http://fashionfor.life/t-sh...

see more

^ | ˅ • Reply • Share ›

Blake Jennings • 4 years ago
i'm literally crying
93 ^ | ˅ • Reply • Share ›

friend ➦ Blake Jennings • 7 months ago
you give me a big o
2 ^ | ˅ • Reply • Share ›

Jon Renner • 4 years ago
This is god's work.
117 ^ | ˅ • Reply • Share ›

friend ➦ Jon Renner • 7 months ago
no its not
^ | ˅ • Reply • Share ›

Adam Heinermann • 4 years ago
Is there a printer-friendly version?
54 ^ | ˅ • Reply • Share ›

Thomas Feichtinger ➦ Adam Heinermann • 3 years ago
Actually copying the contents to a google doc worked pretty well!

I have made it public, have a look:
https://docs.google.com/spr...
27 ^ | ˅ • Reply • Share ›

Sudhanshu Mishra ➦ Thomas Feichtinger • a year ago
Thank you! I sometimes wish entire humanity was as benevolent as us programmers when it comes to sharing the fruits of our labour! The world would be so much better :-)
7 ^ | ˅ • Reply • Share ›

David Joo ➦ Thomas Feichtinger • a year ago
Thank you for doing that!
^ | ˅ • Reply • Share ›

Numus Software ➦ Thomas Feichtinger • 2 years ago
You have to love algorithms !!!
^ | ˅ • Reply • Share ›

ericdrowell Mod ➦ Adam Heinermann • 4 years ago
not yet, but that's a great idea!
8 ^ | ˅ • Reply • Share ›

Matt Labrum ➦ Adam Heinermann • 3 years ago
I, too, wanted a printer-friendly version for studying before an interview, and I wasn't satisfied with the solutions I found provided in the various comments here. So, I went ahead and LaTeX'ed this page to get a nice PDF.

I have uploaded the PDFs I created to my Google Drive and made them public: https://drive.google.com/fo... . In that folder are two PDFS --- one is for letter-sized paper and the other is for A4-sized paper. Assuming I didn't introduce any typos, the content of those PDFs should match the content of this page (as it appears at this moment; 17 February 2015), with the only noteworthy difference being that I moved the Graphs section to be after the Sorting section to help eliminate some extra white space.
6 ^ | v • Reply • Share ›

Yuvaraaj Sreenivasen ➦ Matt Labrum • 2 years ago
Great thanks Matt!!
^ | v • Reply • Share ›

Joe Gibson ➦ Matt Labrum • 3 years ago
Matt,

Great job on the LaTeX document. I'm preparing for a Google interview and this will be a lot of help!

Any chance you can put the .tex file on your drive as well in the same folder?
^ | v • Reply • Share ›

Matt Labrum ➦ Joe Gibson • 3 years ago
Done; the two .tex files and the .eps of the graph are now in that folder.

---Edit---
I've also put the R script and tmp.tex file I used to create the graph in that folder. After creating the .eps file with R, I did some processing on it to get the final Big-O.eps file I include in the .tex files.

For completeness, to get from the R-generated Big-O.eps file to the final Big-O.eps file, I did the following:
1. Open Big-O.eps with a text editor to ensure the text annotations have not been broken apart. I personally had to put "Operations" and "Big-O Complexity" (y-axis label and graph title) back together.
2. Process tmp.tex to get a .dvi file that contains a PSFrag'ed version of the graph.
3. dvips -j -E tmp.dvi -o Big-O.tmp.eps
4. epstool --copy --bbox Big-O.tmp.eps Big-O.eps
5. rm Big-O.tmp.eps
^ | v • Reply • Share ›

Joe Gibson ➦ Matt Labrum • 3 years ago
Thanks, you rock.
^ | v • Reply • Share ›

Gokce Toykuyu • 5 years ago
Could we add some tree algorithms and complexities? Thanks. I really like the Red-Black trees ;)
41 ^ | v • Reply • Share ›

ericdrowell Mod ➦ Gokce Toykuyu • 5 years ago
Excellent idea. I'll add a section that compares insertion, deletion, and search complexities for specific data structures
30 ^ | v • Reply • Share ›

yash bedi ➦ ericdrowell • 9 months ago
its been 4 years you haven't added that section :)
^ | v • Reply • Share ›

Elliot Géhin ➦ yash bedi • 9 months ago
It's up there Yash, bottom of the first table
1 ^ | v • Reply • Share ›

Jonathan Neufeld ➦ Gokce Toykuyu • 10 days ago
Where I come from we use trees on a regular rotation
1 ^ | v • Reply • Share ›

Valentin Stanciu • 4 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".
31 ^ | v • Reply • Share ›

Alexis Mas → Valentin Stanciu • 3 years ago
If you a list: A B C D, When you want to delete B, you can delete a node without iterating over the list.

1. B.data = C.data
2. B.next = C.next
3. delete C

If you can't copy data between nodes because its too expensive then yes, it's O(N)
6 ^ | v • Reply • Share ›

Miguel → Alexis Mas • 3 years ago
You still have to find the position in the list, which can only be done linearly.
7 ^ | v • Reply • Share ›

Guest → Miguel • 3 years ago
You still have to find the position in the list, which can only be done linearly.
3 ^ | v • Reply • Share ›

Alexis Mas → Miguel • 3 years ago
Yes of course, If you need to search the node it's O(n), otherwise you can delete it as I stated before.
1 ^ | v • Reply • Share ›

Guest → Alexis Mas • 3 years ago
No need to find the position if you can delete it as Alexis mentioned
2 ^ | v • Reply • Share ›

OmegaNemesis28 → Alexis Mas • 3 years ago
To get to B - you HAVE to iterate over the list though. You can't just manipulate B without a pointer. So unless you do book-keeping and have pointers to specific nodes you intend to delete/manipulate, LinkLists are O(n) insert and delete.
3 ^ | v • Reply • Share ›

Alexis Mas → OmegaNemesis28 • 3 years ago
Strictly speaking no, you don't. let's say you have this function.

public void delete(Node node)

That function doesn't care how did you got that node.

Did you got my point?

When you have a pointer to a node, and that node needs to be deleted you don't need to iterave over the list.
1 ^ | v • Reply • Share ›

Sam Lehman → Alexis Mas • 3 years ago
But in order to get to that pointer, you probably need to iterate through the list
2 ^ | v • Reply • Share ›

OmegaNemesis28 → Alexis Mas • 3 years ago
But that is MY point :p

You have to have the node FIRST. You have to iterate through the list before you can do that, unless you do book-keeping and happen to have said node. Reread what I said. "have pointers to specific nodes" Most of the time, you do not with LinkedLists. If you have a Linked List and want to delete index 5, you have to iterate to 5 and such. Your example was ABCD, our points are that you typically don't have the pointer to B just offhand. You have to obtain it first which will be O(n)
2 ^ | v • Reply • Share ›

Chris B → OmegaNemesis28 • 2 years ago
Search and insert/delete are different operations. Insert/delete on an unsorted linked list is O(1). The fact that you might have to first search for the element that you want to delete is not considered relevant, as that functionality is covered by the O(n) search operation, not the O(1) insert/delete operations. A real world

functionality is covered by the O(n) search operation, not the O(1) insert/delete operations. A real world example of linked list insert/delete can be found in list_del and list_add of the Linux kernel source, those functions are only 2 and 4 lines of code, so should be easy to understand: http://lxr.free-electrons.c...
2 ^ | v • Reply • Share ›

Pingu App ➦ Alexis Mas • 3 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?
2 ^ | v • Reply • Share ›

Alexis Mas ➦ Pingu App • 3 years ago
In that case you can't deleted that way, you're forced to have a pointer to the previous item.
1 ^ | v • Reply • Share ›

pvlbzn ➦ Alexis Mas • 7 months ago
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.
^ | v • Reply • Share ›

Darren Le Redgatr • 4 years ago
I came here from an idle twitter click. I have no idea what it's talking about or any of the comments. But I love the fact there are people out there this clever. Makes me think that one day humanity will come good. Cheers.
60 ^ | v • Reply • Share ›

friend ➦ Darren Le Redgatr • 7 months ago
no problem
^ | v • Reply • Share ›

qwertykeyboard • 4 years ago
It would be very helpful to have export to PDF. Thx
17 ^ | v • Reply • Share ›