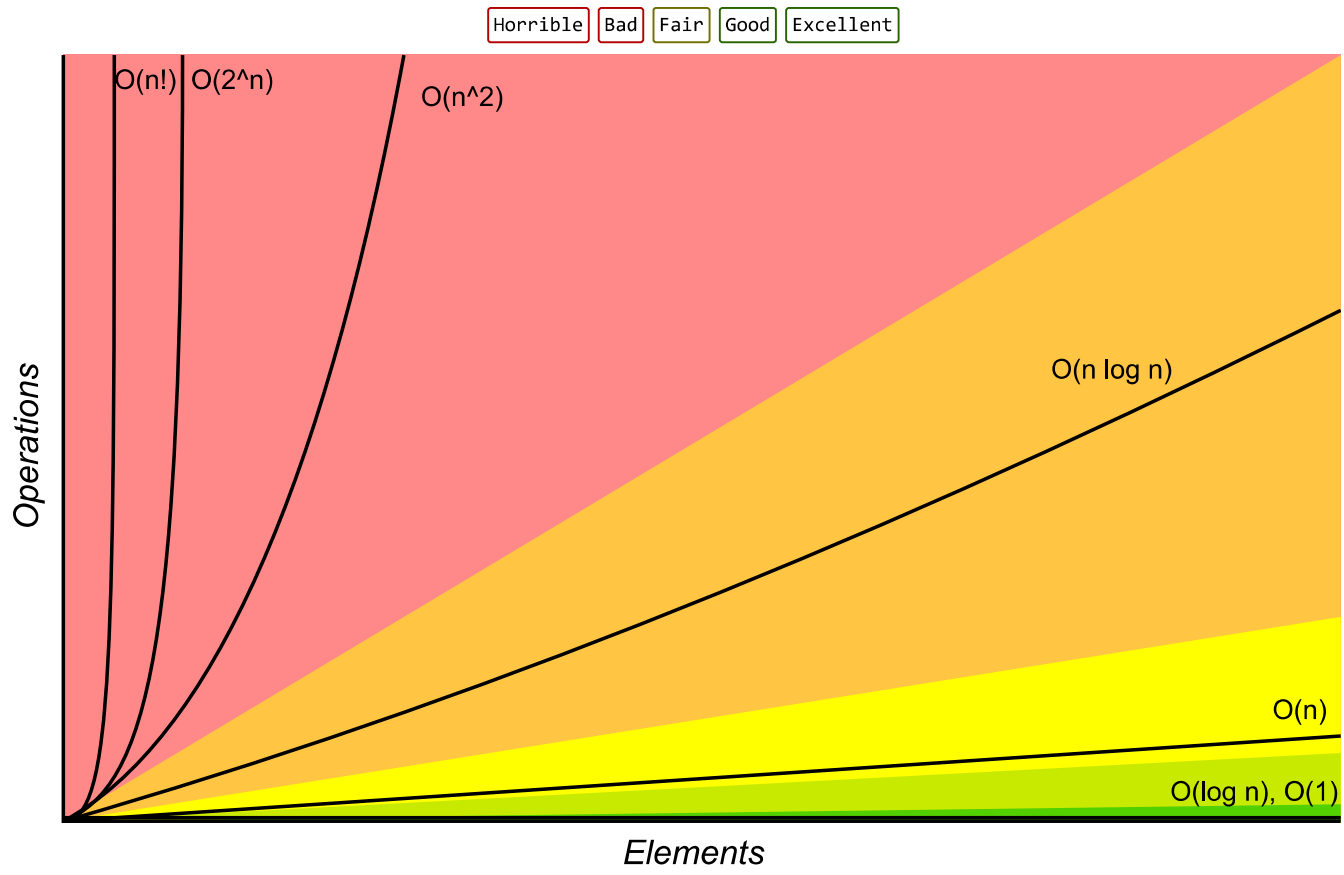
Big-O Cheat Sheet Download PDF

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

Check out El Grapho, a graph data visualization library that supports millions of nodes and edges

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

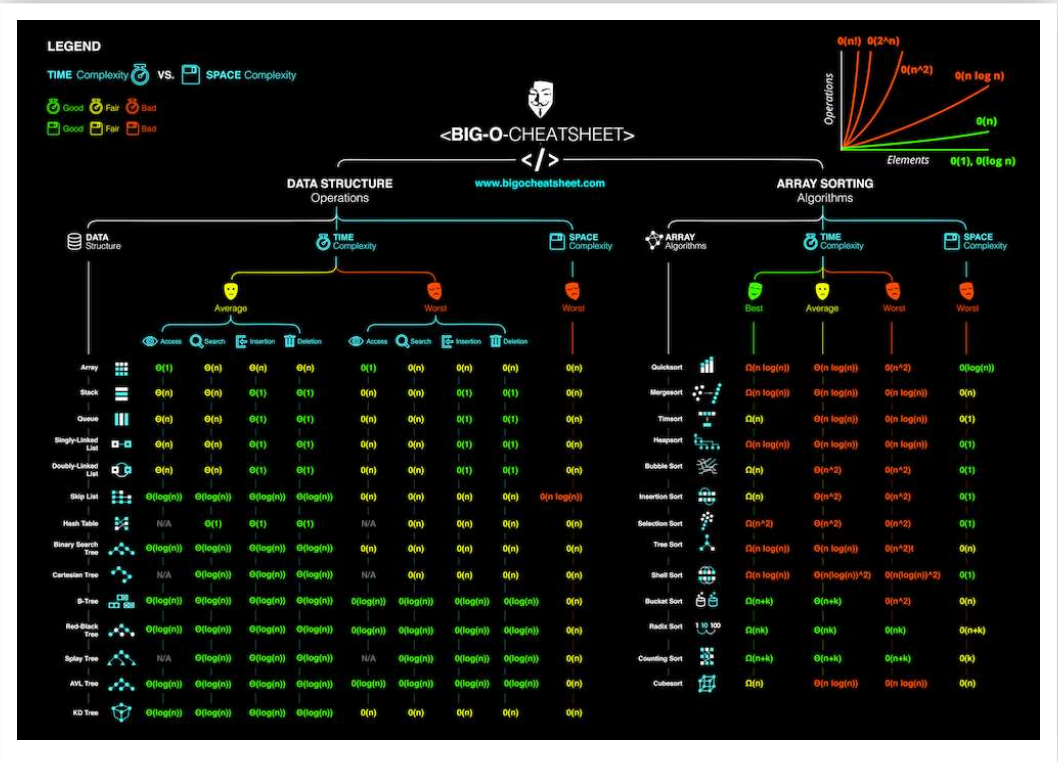
Cracking the Coding Interview: 150 Programming Questions and Solutions

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!

460 Comments Big-O Cheat Sheet Login

Recommend 290 Sort by Best

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

Michael Mitchell • 6 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.

349 • Reply •

Amanda Harlin ➚ Michael Mitchell • 6 years ago

Yes! Please & thank you

78 • Reply •

Asim Ahmad ➚ Amanda Harlin • a year ago

Can you Explain the Above Algorithm.??

• Reply •

Show more replies

Cam Cecil ➚ Michael Mitchell • 6 years ago

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

34 • Reply •

Richard Wheatley ➚ Cam Cecil • 3 years ago

this is plain english.

11 • Reply •

Show more replies

Blake Jennings • 6 years ago

i'm literally crying

97 • Reply •

friend ➚ Blake Jennings • 2 years ago

you give me a big o

4 • Reply •

Gokce Toykuyu • 6 years ago

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

83 • Reply •

ericdrowell Mod ➚ Gokce Toykuyu • 6 years ago

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

31 • Reply •

yash bedi ➚ ericdrowell • 2 years ago

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

• Reply •

Show more replies

Show more replies

Jonathan Neufeld ➚ Gokce Toykuyu • 2 years ago

Where I come from we use trees on a regular rotation

3 ^ | v • Reply •

Valentin Stanciu • 6 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".

61 ^ | v • Reply •

Alexis Mas ➚ Valentin Stanciu • 5 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 •

Miguel ➚ Alexis Mas • 4 years ago

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

7 ^ | v • Reply •

Show more replies

Show more replies

Adam Heinermann • 6 years ago

Is there a printer-friendly version?

55 ^ | v • Reply •

Thomas Feichtinger ➚ Adam Heinermann • 5 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...

31 ^ | v • Reply •

Sudhanshu Mishra ➚ Thomas Feichtinger • 3 years 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 :-)

15 ^ | v • Reply •

Show more replies

Matt Labrum ➚ Adam Heinermann • 4 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.

9 ^ | v • Reply •

Yuvaraaj Sreenivasen ➚ Matt Labrum • 4 years ago

Great thanks Matt!!

^ | v • Reply •

Show more replies

Show more replies

Jon Renner • 6 years ago

This is god's work.

119 ^ | v • Reply •

friend ➚ Jon Renner • 2 years ago

no its not

1 ^ | v • Reply •

Reehz Ree • 4 years ago

great work.. thank you for making it simple

51 ^ | v • Reply •

kapil bindal • 5 years ago

what are the time complexities of searching, deletion and inserting an element in heap tree??

39 ^ | v • Reply •

TheGreat • 4 years ago

This is handy, thanks.

29 ^ | v • Reply •

Darren Le Redgatr • 6 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 •

friend ➚ Darren Le Redgatr • 2 years ago

no problem

^ | v • Reply •

Sheep • 3 years ago

Hi the worst space complexity of quicksort is O(n) ? O(logn) is the best case

19 • Reply •

Suman Kumar Hansada • 5 years ago

Being an computer science student, I really appreciate your work. Algorithms are really one of the difficult topics.
I would like to know how you have created this simple webpage. Have you used any software ? Please name them..

18 • Reply •

Anup Sawant • 6 years ago

Space complexity of Quicksort is O(n) ?. I always thought its O(1).

18 • Reply •

Thierry A. ➚ Anup Sawant • 6 years ago

The space complexity is O(n) for the "naive" version but O(log(n)) for the Sedgewick's version. There is a version of Quicksort which uses an in-place partition algorithm and can achieve the complete sort using O(log n) space (not counting the input) on average (for the call stack). (source: http://en.wikipedia.org/wik...

1 • Reply •

qwertykeyboard • 6 years ago

It would be very helpful to have export to PDF. Thx

17 • Reply •

Gene ➚ qwertykeyboard • 5 years ago

You could convert the document yourself using Pandoc: http://johnmacfarlane.net/p...
It might take you a long time to get it working, but Pandoc is an amazing one stop shop for file conversion, and cross platform compatible.
If I understand big oh notation correctly I might say "I estimate your learning rate for learning Pandoc will be O(1). ".

2 • Reply •

Ashutosh ➚ Gene • 5 years ago

proved O(n), n=number of format conversions to learn :)

4 • Reply •

Show more replies

Tomas Kazlauskas • 9 months ago

It is quite interesting.

14 • Reply •

rcgldr • 9 months ago

Although a hash table has big O space complexity O(n), there's a load factor considered as a constant, which doesn't show up in big O space complexity. For efficiency, the "load factor" for a hash table should be kept at .70 to .75 or less, so the space requirement is 1/.75 ~ 1.33 n to 1/.70 ~ 1.43 n hash table entries. For comparison, space complexity for merge sort is typically n or n/2, and could be a size related to the number of objects, number of pointers, or number of indexes.

13 • Reply •

Guest • 6 years ago

Finals are already over... This should have been shared a week ago! Would have saved me like

45 minutes of using Wikipedia.
11 • Reply •

Jon Renner • 6 years ago
Anyway I can get a PDF version without taking screenshots myself?
9 • Reply •

Attila Oláh → Jon Renner • 4 years ago
Print → Destination: Change → Select "Save as PDF" (in Chrome).
3 • Reply •

Ankush Gupta • 6 years ago
Awesome resource! You should add Dijkstra using a Fibonacci Heap!
9 • Reply •

sigmaalgebra • 6 years ago
You omitted an in-place, guaranteed O(n log(n)) array sort,
e.g., heap sort. You omitted radix sort that can be faster

than any of the algorithms you mentioned.

Might mention SAT and related problems in NP-complete

where the best known algorithm for
a problem of size n has O(2^n).

Might include an actual, precise definition of O().
9 • Reply •

Jackson Westeen • 6 years ago
Awesome! Also, SplayTrees?
8 • Reply •

robbyoconnor → Jackson Westeen • 6 years ago
amortized O(logn) performance.on all operations
• Reply •

Antoine Grondin • 6 years ago
I think DFS and BFS, under Search, would be more appropriate listed as Graph instead of Tree.
8 • Reply •

ericdrowell Mod → Antoine Grondin • 6 years ago
Fixed! Thanks
4 • Reply •

Quentin Pleplé → Antoine Grondin • 6 years ago
Agreed
1 • Reply •