Big-O Cheat Sheet

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 Yahoo, eBay, LinkedIn, and Google, 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

Legend

Excellent Good Fair Bad Horrible

Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	-	O(1)	O(1)	O(1)	-	O(n)	O(n)	O(n)	O(n)
Binary									

Binary Search Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	-	O(log(n))	O(log(n))	O(log(n))	-	O(n)	O(n)	O(n)	O(n)
B-Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	-	O(log(n))	O(log(n))	O(log(n))	-	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)

Array Sorting Algorithms

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

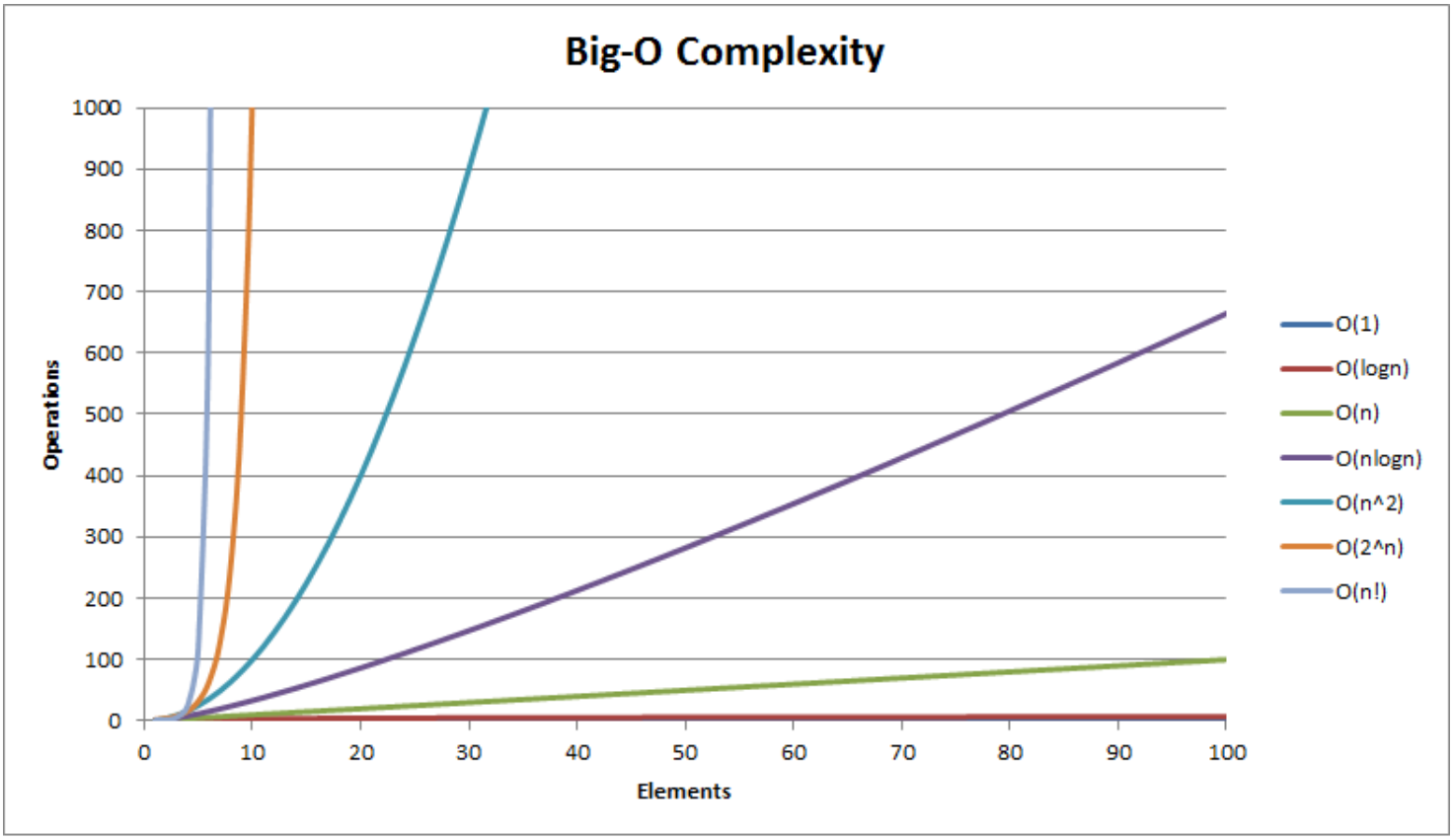
Graph Operations

Node / Edge Management	Storage	Add Vertex	Add Edge	Remove Vertex	Remove Edge	Query
Adjacency list	O(|V|+|E|)	O(1)	O(1)	O(|V| + |E|)	O(|E|)	O(|V|)
Incidence list	O(|V|+|E|)	O(1)	O(1)	O(|E|)	O(|E|)	O(|E|)
Adjacency matrix	O(|V|^2)	O(|V|^2)	O(1)	O(|V|^2)	O(1)	O(1)
Incidence matrix	O(|V| · |E|)	O(|V| · |E|)	O(|V| · |E|)	O(|V| · |E|)	O(|V| · |E|)	O(|E|)

Heap Operations

Type	Time Complexity						
	Heapify	Find Max	Extract Max	Increase Key	Insert	Delete	Merge
Linked List (sorted)	-	O(1)	O(1)	O(n)	O(n)	O(1)	O(m+n)
Linked List (unsorted)	-	O(n)	O(n)	O(1)	O(1)	O(1)	O(1)
Binary Heap	O(n)	O(1)	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(m+n)
Binomial Heap	-	O(1)	O(log(n))	O(log(n))	O(1)	O(log(n))	O(log(n))
Fibonacci Heap	-	O(1)	O(log(n))	O(1)	O(1)	O(log(n))	O(1)

Big-O Complexity Chart



Recommended Reading

- 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)

Contributors

1. Eric Rowell, founder of Coder Lifestyle
2. Quentin Pleple
3. Michael Abed
4. Nick Dizazzo
5. Adam Forsyth
6. David Dorfman
7. Jay Engineer
8. Jennifer Hamon
9. Josh Davis
10. Nodir Turakulov
11. Bart Massey
12. Vinnie Magro
13. Miguel Amigot
14. Drew Bailey
15. Aneel Nazareth
16. Rahul Chowdhury
17. Robert Burke
18. steven41292
19. Brandon Amos

20. Mike Davis
21. Casper Van Gheluwe
22. Joel Friedly
23. Oleg
24. Renfred Harper
25. Piper Chester
26. Eric Lefevre-Ardant
27. Jonathan McElroy
28. Si Pham
29. mcverry
30. Max Hoffmann
31. Alejandro Ramirez
32. Damon Davison
33. Alvin Wan
34. Alan Briolat
35. Drew Hannay
36. Andrew Rasmussen
37. Dennis Tsang
38. Bahador Saket

Edit these tables!

Comments

288 Comments Big-O Cheat Sheet 1 Login

Recommend 106 Share Sort by Best

Join the discussion…

Michael Mitchell • 2 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.

248 • Reply • Share ›

Amanda Harlin Michael Mitchell • 2 years ago

Yes! Please & thank you

57 • Reply • Share ›

Cam Cecil Michael Mitchell • 2 years ago

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

23 • Reply • Share ›

Arjan Nieuwenhuizen Michael Mitchell • 2 years ago

Here are the links that I know of.

#1) http://aduni.org/courses/algor...
#2) http://ocw.mit.edu/courses/ele...
#3) https://www.udacity.com/course...

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

Sincerely,
Arjan

p.s.
https://www.coursera.org/cours...
This course has just begun on coursera (dated 1 July 2013), and looks very good.
11 • Reply • Share ›

fireheron ➚ Arjan Nieuwenhuizen • 2 years ago
Thank you Arjan. Espaecially the coursera.org one ;-)
3 • Reply • Share ›

michaelfloering ➚ fireheron • 7 months ago
also this! http://opendatastructures.org
4 • Reply • Share ›

yth ➚ michaelfloering • 6 months ago
thank you for sharing this.
1 • Reply • Share ›

Blake Jennings • 2 years ago
i'm literally crying
77 • Reply • Share ›

Jon Renner • 2 years ago
This is god's work.
100 • Reply • Share ›

Adam Heinermann • 2 years ago
Is there a printer-friendly version?
51 • Reply • Share ›

Thomas Feichtinger ➚ Adam Heinermann • a year ago
Actually copying the contents to a google doc worked pretty well!

I have made it public, have a look:
https://docs.google.com/spread...
24 • Reply • Share ›

ericdrowell Mod ➚ Adam Heinermann • 2 years ago
not yet, but that's a great idea!
8 • Reply • Share ›

Matt Labrum ➚ Adam Heinermann • 7 months 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/folde... . 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.

2 • Reply • Share ›

Yuvaraaj Sreenivasen Matt Labrum • 4 months ago

Great thanks Matt!!

• Reply • Share ›

Joe Gibson Matt Labrum • 7 months 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?

• Reply • Share ›

Matt Labrum Joe Gibson • 7 months 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

• Reply • Share ›

Joe Gibson Matt Labrum • 7 months ago

Thanks, you rock.

• Reply • Share ›

Gokce Toykuyu • 3 years ago

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

34 • Reply • Share ›

ericdrowell Mod Gokce Toykuyu • 3 years ago

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

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

30 • Reply • Share ›

Darren Le Redgatr • 2 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.

55 • Reply • Share ›

Valentin Stanciu • 2 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".

25 • Reply • Share ›

Alexis Mas ↗ Valentin Stanciu • a year 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)

5 • Reply • Share ›

Miguel ↗ Alexis Mas • 9 months ago

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

5 • Reply • Share ›

Guest ↗ Miguel • 7 months ago

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

3 • Reply • Share ›

Alexis Mas ↗ Miguel • 7 months 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 • Reply • Share ›

Guest ↗ Alexis Mas • 7 months ago

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

2 • Reply • Share ›

OmegaNemesis28 ↗ Alexis Mas • 7 months 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.

2 • Reply • Share ›

Alexis Mas ➚ OmegaNemesis28 • 7 months 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 • Reply • Share ›

OmegaNemesis28 ➚ Alexis Mas • 7 months 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 • Reply • Share ›

SamLehman617 ➚ Alexis Mas • 6 months ago

But in order to get to that pointer, you probably need to iterate through the list

• Reply • Share ›

Pingu App ➚ Alexis Mas • 7 months 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?

1 • Reply • Share ›

Alexis Mas ➚ Pingu App • 7 months ago

In that case you can't deleted that way, you're forced to have a pointer to the previous item.

1 • Reply • Share ›

qwertykeyboard • 2 years ago

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

17 • Reply • Share ›

Gene ➚ qwertykeyboard • 2 years ago

You could convert the document yourself using Pandoc: http://johnmacfarlane.net/pand...
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 • Share ›

Ashutosh → Gene • 2 years ago

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

4 • Reply • Share ›

Juan Carlos Alvarez → Gene • 2 years ago

big oh. haha funny.

1 • Reply • Share ›

Guest • 2 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 • Share ›

sigmaalgebra • 2 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 • Share ›

Jon Renner • 2 years ago

Anyway I can get a PDF version without taking screenshots myself?

8 • Reply • Share ›

Attila Oláh → Jon Renner • 8 months ago

Print → Destination: Change → Select "Save as PDF" (in Chrome).

2 • Reply • Share ›

Antoine Grondin • 2 years ago

I think DFS and BFS, under Search, would be more appropriate listed as Graph instead of Tree.

8 • Reply • Share ›

ericdrowell Mod → Antoine Grondin • 2 years ago

Fixed! Thanks

4 • Reply • Share ›

Quentin Pleplé → Antoine Grondin • 2 years ago

Agreed

1 • Reply • Share ›

Ankush Gupta • 2 years ago

Awesome resource! You should add Dijkstra using a Fibonacci Heap!

7 • Reply • Share ›

tempire • 2 years ago

This chart seems misleading. Big O is worst case, not average case; ~ is average case. O(...) shouldn't be used in the average case columns.

16 • Reply • Share ›

guest tempire • 2 years ago

I think big O is just an upper bound. It could be used for all (best, worst and average) cases. Am I wrong?

20 • Reply • Share ›

Luis guest • 2 years ago

You are right.

6 • Reply • Share ›

Oleksandr Luis • 2 years ago

@Luis That is WRONG. **@tempire** is correct. Big O cannot be used for lower, average, and upper bound.. Big O (Omicron) is the Worst Case Scenario. It is the upper bound for for the algorithm. For instance in a linear search algorithm, worst case is when the list is completed out of order, i.e. the list sorted but backwards. Omega is the lower bound. This is almost pointless to have, for instance you would rather have Big O then Omega because it is exactly the same as say "it will take more than five dollars to get to N.Y. vs. Its will always take, at most, 135 dollars to get to New York." The first bit of information from Omega is essentially useless, the third however gives you the constraint. Theta is the upper and lower bound together. This is the most beneficially piece of information to have about an algorithm but unfortunately it is usually very hard to find, but we have done this. You can usually find that average for an algorithms efficiency by testing it in average case and worst cases together. Simply this is a computational exercise to extract the empirical data. There is another problem I do not like is the color scheme is sometimes wrong.. O(n) is better the O(log(n))? In what way? 1024 vs 10 increments that a sort algorithm has to perform for instance? All in all this is good information but in its current state, to the

- Page styling via Bootstrap
- Comments via Disqus
- Algorithm detail via Wikipedia
- Table source hosted on Github
- Mashup via @ericdrowell