

O PLANO DE 7 DIAS PARA NUNCA MAIS TRAVAR EM BIG-O.

30 minutos por dia. No fim: a cola com 20 trechos de código e as frases para dizer na entrevista. **Regra: diga a complexidade antes de tipar.**

DIA 1

$O(1)$ e $O(n)$

Acesso direto é constante. Um loop que passa por todos os itens é linear.

Faça: Resolva 3 problemas fáceis e escreva a complexidade em um comentário antes do código.

DIA 2

$O(n^2)$

Loop dentro de loop sobre os mesmos dados. 1.000 itens viram 1 milhão de passos.

Faça: Escreva a força bruta do Two Sum e diga por que ela é $O(n^2)$.

DIA 3

$O(\log n)$

Cada passo corta o problema pela metade. 1 milhão de itens em cerca de 20 passos.

Faça: Implemente a busca binária do zero, sem olhar.

DIA 4

$O(n \log n)$

O custo de ordenar. Ordenar e depois percorrer continua $O(n \log n)$.

Faça: Resolva um problema que fica fácil depois de ordenar (ex.: Merge Intervals).

DIA 5

Espaço também conta

Hash map ou array extra do tamanho da entrada custa $O(n)$ de memória.

Faça: Refaça o Two Sum com hash map e diga tempo e espaço.

DIAS 6 E 7

Fale em voz alta

Pegue 10 problemas que você já resolveu e diga a complexidade antes de rodar.

Faça: Grave 1 minuto explicando a complexidade de um problema. Ouça e refaça.

Como contar: ache os loops, veja quantas vezes cada um roda, some os que estão em sequência e multiplique os que estão um dentro do outro.

#	CÓDIGO	TEMPO	ESPAÇO
01	<code>arr[i]</code> Acesso direto por índice.	O(1)	O(1)
02	<code>map.get(k)</code> / <code>set.has(x)</code> Hash: busca sem percorrer.	O(1) em média	O(1)
03	<code>arr.push(x)</code> / <code>arr.pop()</code> Mexe só no fim do array.	O(1) amortizado	O(1)
04	<code>arr.shift()</code> / <code>arr.unshift(x)</code> Mexe no início: reindexa todos.	O(n)	O(1)
05	<code>for (const x of arr) { ... }</code> Passa uma vez por cada item.	O(n)	O(1)
06	<code>arr.includes(x)</code> / <code>arr.indexOf(x)</code> Busca linear escondida.	O(n)	O(1)
07	<code>arr.map(...)</code> / <code>arr.filter(...)</code> Cria um array novo.	O(n)	O(n)
08	<code>[...arr]</code> / <code>arr.slice()</code> Copia o array inteiro.	O(n)	O(n)
09	<code>for (a of A) {} for (b of B) {}</code> Loops em sequência somam.	O(n + m)	O(1)
10	<code>for (a of A) for (b of A) {}</code> Loops aninhados multiplicam.	O(n²)	O(1)

#	CÓDIGO	TEMPO	ESPAÇO
11	<code>for (a of A) for (b of B) {}</code> Dois tamanhos diferentes.	$O(n \cdot m)$	$O(1)$
12	<code>for (x of arr) if (arr.includes(x))</code> includes dentro do loop.	$O(n^2)$	$O(1)$
13	<code>const seen = new Set(arr)</code> Troca memória por velocidade.	$O(n)$	$O(n)$
14	contar frequência com Map Um passe + um hash.	$O(n)$	$O(n)$
15	busca binária em array ordenado Corta pela metade a cada passo.	$O(\log n)$	$O(1)$
16	<code>while (n > 1) n = Math.floor(n / 2)</code> Divide por 2 até chegar em 1.	$O(\log n)$	$O(1)$
17	<code>arr.sort((a, b) => a - b)</code> Ordenar é sempre o gargalo.	$O(n \log n)$	depende da engine
18	ordenar + dois ponteiros O sort domina o loop.	$O(n \log n)$	$O(1)$ extra
19	DFS ou BFS em grafo Visita cada vértice e aresta uma vez.	$O(V + E)$	$O(V)$
20	<code>fib(n)</code> recursivo sem memo Recalcula tudo. Com memo: $O(n)$.	$O(2^n)$	$O(n)$

FRASES PARA DIZER NA ENTREVISTA

"A força bruta é $O(n^2)$. Com hash map, cai para $O(n)$ de tempo e $O(n)$ de espaço."

"O gargalo é o sort, então o total fica $O(n \log n)$."

"Posso trocar memória por velocidade: guardo o que já vi em um Set."

"Se a entrada já vier ordenada, uso busca binária e fica $O(\log n)$."

"Antes de codar: vou confirmar o tamanho da entrada para escolher a abordagem."

MATERIAL GRATUITO

- [Big-O Cheat Sheet](#)
- [Tech Interview Handbook: study cheatsheet](#)
- [VisuAlgo \(veja os algoritmos rodando\)](#)
- [NeetCode roadmap](#)
- [MDN: Array.prototype.sort \(pt-BR\)](#)

O júnior torce para o entrevistador não perguntar. Quem passa diz a complexidade antes de tipar e já sugere a otimização.

Segue @lucasadvice