tamanho da lista(n)	Binary Search(0(log n))	Interpolation Search (O(log log n) - melhor caso)	Jump Search (O(√n))	Exponential Search (O(log n))
10	3	3	3	3
100	7	7	10	7
1	10	10	32	10
10	14	14	100	14
100	17	17	316	17
1,000,000	20	20	1,000	20,000
10,000,000	23	23	3,162	23

tamanho da lista	algoritmo	Tempo de Execução (ms)	Comparações
1000	shell sort	3.2	1,53
1000	merge sort	2.1	1,200
1000	selection sort	10.5	500,000
10000	shell sort	18.4	15,300
10000	merge sort	12.8	12,000

Algoritmo	Complexidade de Tempo	Complexidade de Espaço	Observações
Binary Search	O(logn)	O(1)	Funciona apenas em arrays ordenados. Divide a busca pela metade a cada iteração.
Interpolation Search	gn) no melhor caso, O(n) no pi	O(1)	Útil para dados uniformemente distribuídos, mas pode ser lento em dados não uniformes.
Jump Search	O(√n)	O(1)	Divide o array em blocos, fazendo saltos de tamanho √n. Após o salto, realiza busca linear no bloco.
Exponential Search	O(logn)	O(1)	Rápido para encontrar o intervalo, mas requer que o array esteja ordenado. Utiliza Binary Search no intervalo encontrado.
			ALGORITMO DE BUSCA

Algoritmo	Complexidade de Tempo	Complexidade de Espaço	Observações
Shell Sort	O(n3/2) no caso médio, O(n2) no pior caso	O(1)	Usa gaps para comparações, melhorando a eficiência para listas parcialmente ordenadas.
Merge Sort	O(nlogn) em todos os casos	O(n)	Divide e conquista. Efetivamente lida com listas grandes, mas requer espaço adicional para armazenar subarrays.
Selection Sort	O(n2) em todos os casos	O(1)	Simples de implementar, mas ineficiente para listas grandes devido ao grande número de comparações.
Quick Sort	O(nlogn) no melhor/médio caso, O(n2) no pior caso	O(logn)	Divide e conquista. O caso pior pode ser evitado com pivôs escolhidos aleatoriamente ou pelo método de mediana de três.
Bucket Sort	O(n+k)	O(n+k)	Útil para dados uniformemente distribuídos em um intervalo pequeno. Requer k buckets adicionais, onde k é o número de buckets.
Radix Sort	O(nk), onde k é o número de dígitos	O(n+k)	Especializado para inteiros. Rápido para listas com inteiros de tamanho fixo, mas não é comparativo.
ALGORITMOS DE ORDENAÇÃO			

Coluna 1	Coluna 2	Coluna 3
Merge Sort	Estável	Durante a fusão, elementos iguais são copiados na ordem em que aparecem nos subarrays originais.
Bucket Sort	Estável	Desde que cada bucket seja ordenado usando um algoritmo estável, os elementos permanecem na ordem relativa em que foram distribuídos nos buckets.
Radix Sort	Estável	Ordena dígito por dígito (ou unidade por unidade) preservando a ordem relativa em cada etapa.