|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SEARCHES** | **Best** | **Average** | **Worst** |  |  |  | **Other notes** |
| **Binary Search** | 1 |  |  |  |  |  | Use on Sorted arrays or binary search trees |
| **Linear Search** | 1 | *n* | *n* |  |  |  | Use on any array or linked list |
| **Depth-first (DFS)** | 1 | *n = |V|+|E|* | |  |  |  | Use on a graph of |V| vertices and |E| edges |
| **Breadth-first (BFS)** | 1 | *n = |V|+|E|* | |  |  |  | Use on a graph of |V| vertices and |E| edges |
| **SORTS** | **Best** | **Average** | **Worst** | **Memory** | **Stable** | **Method** | **Other notes** |
| [**Binary tree sort**](http://en.wikipedia.org/wiki/Binary_tree_sort) | *n* | 20 ! | 20 ! | 15 ! *n* | Yes | Insertion | When using a [self-balancing binary search tree](http://en.wikipedia.org/wiki/Self-balancing_binary_search_tree) |
| [**Timsort**](http://en.wikipedia.org/wiki/Timsort) | *n* | 20 ! | 20 ! | *n* | Yes | Insertion & Merging | *n* comparisons when the data is already sorted or reverse sorted. |
| [**Heapsort**](http://en.wikipedia.org/wiki/Heapsort) |  | 20 ! |  | 00 !1 | No | Selection |  |
| [**Merge sort**](http://en.wikipedia.org/wiki/Merge_sort) |  |  |  | 50 !worst is *n* | Yes | Merging | Highly parallelizable. Used to sort this table in Firefox. |
| **In-place Merge Sort** | – | –--2 |  | 1 | Yes | Merging | Can be implemented as a stable sort based on stable in-place merging. |
| [**Quicksort**](http://en.wikipedia.org/wiki/Quicksort) | 20 ! *n* log *n* | 20 ! *n* log *n* | *n2* | 20 !log *n* | ??? | Partitioning | Can be implemented as a stable sort depending on how the pivot is handled. Most implementations are unstable, as stable in-place partitioning is more complex. [Naïve](http://en.wikipedia.org/wiki/Naïve_algorithm) variants use space array to store the partition. |
| [**Bubble sort**](http://en.wikipedia.org/wiki/Bubble_sort) | 15 !*n* | *n2* | *n2* | 00 !1 | Yes | Exchanging | Tiny code size |
| [**Insertion sort**](http://en.wikipedia.org/wiki/Insertion_sort) | *n* | 25 ! | 25 ! | 00 !1 | Yes | Insertion | Average case is also , where *d* is the number of [inversions](http://en.wikipedia.org/wiki/Permutation_groups#Transpositions.2C_simple_transpositions.2C_inversions_and_sorting) |
| [**Selection sort**](http://en.wikipedia.org/wiki/Selection_sort) | 25 ! *n2* | 25 ! *n2* | 25 ! *n2* | 00 !1 | No | Selection | Stable with  extra space, for example using lists. Used to sort this table in Safari. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SEARCHES** | **Best** | **Average** | **Worst** |  |  |  | **Other notes** |
| **Binary Search** | 1 |  |  |  |  |  | Use on Sorted arrays or binary search trees |
| **Linear Search** | 1 | *n* | *n* |  |  |  | Use on any array or linked list |
| **Depth-first (DFS)** | 1 | *n = |V|+|E|* | |  |  |  | Use on a graph of |V| vertices and |E| edges |
| **Breadth-first (BFS)** | 1 | *n = |V|+|E|* | |  |  |  | Use on a graph of |V| vertices and |E| edges |
| **SORTS** | **Best** | **Average** | **Worst** | **Memory** | **Stable** | **Method** | **Other notes** |
| [**Binary tree sort**](http://en.wikipedia.org/wiki/Binary_tree_sort) | *n* | 20 ! | 20 ! | 15 ! *n* | Yes | Insertion | When using a [self-balancing binary search tree](http://en.wikipedia.org/wiki/Self-balancing_binary_search_tree) |
| [**Timsort**](http://en.wikipedia.org/wiki/Timsort) | *n* | 20 ! | 20 ! | *n* | Yes | Insertion & Merging | *n* comparisons when the data is already sorted or reverse sorted. |
| [**Heapsort**](http://en.wikipedia.org/wiki/Heapsort) |  | 20 ! |  | 00 !1 | No | Selection |  |
| [**Merge sort**](http://en.wikipedia.org/wiki/Merge_sort) |  |  |  | 50 !worst is *n* | Yes | Merging | Highly parallelizable. Used to sort this table in Firefox. |
| **In-place Merge Sort** | – | –--2 |  | 1 | Yes | Merging | Can be implemented as a stable sort based on stable in-place merging. |
| [**Quicksort**](http://en.wikipedia.org/wiki/Quicksort) | 20 ! *n* log *n* | 20 ! *n* log *n* | *n2* | 20 !log *n* | ??? | Partitioning | Can be implemented as a stable sort depending on how the pivot is handled. Most implementations are unstable, as stable in-place partitioning is more complex. [Naïve](http://en.wikipedia.org/wiki/Naïve_algorithm) variants use space array to store the partition. |
| [**Bubble sort**](http://en.wikipedia.org/wiki/Bubble_sort) | 15 !*n* | *n2* | *n2* | 00 !1 | Yes | Exchanging | Tiny code size |
| [**Insertion sort**](http://en.wikipedia.org/wiki/Insertion_sort) | *n* | 25 ! | 25 ! | 00 !1 | Yes | Insertion | Average case is also , where *d* is the number of [inversions](http://en.wikipedia.org/wiki/Permutation_groups#Transpositions.2C_simple_transpositions.2C_inversions_and_sorting) |
| [**Selection sort**](http://en.wikipedia.org/wiki/Selection_sort) | 25 ! *n2* | 25 ! *n2* | 25 ! *n2* | 00 !1 | No | Selection | Stable with  extra space, for example using lists. Used to sort this table in Safari. |

