| **排序算法** | **最好时间复杂度** | **最坏时间复杂度** | **平均时间复杂度** | **空间复杂度** | **稳定性** | **备注** |
| --- | --- | --- | --- | --- | --- | --- |
| 冒泡排序 | O(n) | O(n²) | O(n²) | O(1) | 稳定 | 通过相邻交换排序，优化后可提前终止。 |
| 选择排序 | O(n²) | O(n²) | O(n²) | O(1) | 不稳定 | 每次选最小元素，可能破坏相等元素的顺序。 |
| 插入排序 | O(n) | O(n²) | O(n²) | O(1) | 稳定 | 适用于小规模或部分有序数据。 |
| 快速排序 | O(n log n) | O(n²) | O(n log n) | O(log n) | 不稳定 | 基于分治，递归栈空间平均 O(log n)，最坏 O(n)。 |
| 归并排序 | O(n log n) | O(n log n) | O(n log n) | O(n) | 稳定 | 需额外空间合并，常用于外部排序。 |
| 堆排序 | O(n log n) | O(n log n) | O(n log n) | O(1) | 不稳定 | 原地建堆，调整堆可能破坏稳定性。 |
| 希尔排序 | O(n log n) | O(n²) | O(n^1.3) | O(1) | 不稳定 | 改进的插入排序，分组间隔序列影响性能。 |
| 计数排序 | O(n + k) | O(n + k) | O(n + k) | O(n + k) | 稳定 | 适用于整数且范围较小的情况（k 为数据范围）。 |
| 基数排序 | O(d(n + k)) | O(d(n + k)) | O(d(n + k)) | O(n + k) | 稳定 | 按位排序（d 为最大位数），通常使用稳定的子排序（如计数排序）。 |
| 桶排序 | O(n + k) | O(n²) | O(n + k) | O(n + k) | 稳定 | 数据分布均匀时高效，最坏情况退化为 O(n²)（如桶内使用插入排序）。 |

‌**关键点解析：**‌

* ‌**稳定性**‌：稳定算法保持相等元素的原始顺序（如冒泡、插入、归并、计数、基数、桶排序）；不稳定算法可能改变其顺序（如选择、快排、堆排序、希尔排序）。
* ‌**时间复杂度**‌：
  + ‌**O(n²)**‌：冒泡、选择、插入排序在平均和最坏情况下表现较差。
  + ‌**O(n log n)**‌：快排、归并、堆排序在大部分情况下更高效。
  + ‌**线性或近似线性**‌：计数、基数、桶排序在特定条件下（如数据范围小）可达到线性时间复杂度。
* ‌**空间复杂度**‌：
  + ‌**O(1)**‌：冒泡、选择、插入、希尔、堆排序为原地排序。
  + ‌**O(n)**‌：归并排序需要额外空间合并。
  + ‌**O(log n)**‌：快排序递归栈空间平均较小，但最坏情况可能退化为 O(n)。
* ‌**适用场景**‌：
  + ‌**小规模数据**‌：插入、冒泡排序。
  + ‌**大规模数据**‌：快排（平均最优）、归并（稳定且适合外部排序）、堆排序（原地但常数因子大）。
  + ‌**特定数据分布**‌：计数（小范围整数）、基数（多关键字排序）、桶排序（均匀分布）。

通过此表可快速对比不同排序算法的优劣，根据实际需求（如数据规模、稳定性要求、内存限制等）选择合适的算法。