# Sorting Algorithms Analysis

# Definitions

**Bubble Sort:**  
Bubble Sort is a simple comparison-based algorithm. It repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process is repeated until the list is sorted. The largest elements gradually 'bubble' to the top with each pass.

**Quick Sort:**  
Quick Sort is a divide-and-conquer algorithm. It selects a 'pivot' element and partitions the array such that elements less than the pivot go to the left and elements greater go to the right. It then recursively sorts the sub-arrays.

# 2. Time and Space Complexities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Best Case | Average Case | Worst Case | Space Complexity |
| Bubble Sort | O(n) (optimized) | O(n²) | O(n²) | O(1) |
| Quick Sort | O(n log n) | O(n log n) | O(n²) (rare) | O(log n) |

# 3. When and Where to Use

**Bubble Sort:**  
- Use for educational purposes and very small datasets.  
- Easy to implement but very inefficient for large inputs.

**Quick Sort:**  
- Suitable for large datasets.  
- Preferred in performance-critical applications.  
- Used in standard libraries due to its efficiency.

# 4. Why Quick Sort is Preferred over Bubble Sort

Quick Sort is generally much faster than Bubble Sort, especially for larger datasets. While Bubble Sort compares adjacent items and requires multiple full passes through the array, Quick Sort divides the array and solves smaller problems efficiently through recursion. It leverages a smarter strategy of partitioning which reduces the total number of comparisons and swaps required.  
  
Additionally, Quick Sort has an average-case time complexity of O(n log n), while Bubble Sort remains at O(n²). This makes Quick Sort significantly more scalable and practical in real-world scenarios.

In short: Bubble Sort is simple but slow. Quick Sort is more complex but far more efficient.