a comparative table summarizing the time complexities and properties of all covered algorithms.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Time Complexity | Space Complexity | Best Use Case | Remarks |
| Bubble Sort | O(n²) (Average, Worst) | O(1) | Small datasets, learning purposes | Inefficient for large datasets, simple to implement |
| Selection Sort | O(n²) (Average, Worst) | O(1) | Small datasets | Similar to Bubble Sort, not ideal for large datasets |
| Insertion Sort | O(n²) (Average, Worst) | O(1) | Nearly sorted data, small datasets | Efficient when the data is nearly sorted |
| Merge Sort | O(n log n) (Average, Worst) | O(n) | Large datasets, external sorting | Stable, good for large lists ,requires extra space |
| Quick Sort | O(n log n) (Average) | O(log n) (Best) | Large datasets, efficient sorting | In-place, fast, but can degrade to O(n²) in the worst case. |
| Linear search | O(n) | O(1) | Searching in unsorted arrays or lists | Simple but slow for large datasets, no sorting need |
| Binary Search | O(log n) | O(1) | Searching in sorted arrays or lists | Fast searching in sorted data |

**Insights into Preferred Algorithms for Different Circumstances**

1. **Small Datasets**:
   * **Bubble Sort**, **Selection Sort**, and **Insertion Sort** are simple to implement and may work well on small or nearly sorted datasets. they perform poorly as the dataset grows.
2. **Large Datasets**:
   * **Merge Sort**, **Quick Sort**, and **Heap Sort** are better suited for large datasets due to their O(n log n) average time complexity. **Quick Sort** is often the fastest but **Merge Sort** guarantees a stable performance.
3. **Space Efficiency**:
   * If space efficiency is crucial, **Quick Sort** and **Heap Sort** are preferable since they sort in place. **Merge Sort**, on the other hand, requires additional space.
4. **Optimized Search**:
5. **Binary Search** is ideal for searching in sorted arrays or lists. For pathfinding in graphs with specific goals, **A**\* is the most efficient with a good heuristic