**Introduction to Data Structures**

Data structures are essential for **organizing and storing data efficiently**, **enabling faster access and modification(Get/Search/Insert/Delete)**. They play a crucial role in software development by **optimizing performance and scalability**.

**Types of Data Structures**

**Linear Structures**: Arrays, Linked Lists, Stacks, Queues. **Non-Linear**: Trees, Graphs, Hash Tables.

**Sequential Storage**: Arrays, Matrices, Stacks, Queues. **Linked Storage**: Linked Lists, Trees, Graphs.

**Big-O notation O(n)** the running time of an algorithm **scales** as the input size increases.

|  |  |  |
| --- | --- | --- |
| **Big-O Notation** | **Complexity Type** | **Example Algorithm** |
| O(1) | Constant Time | Accessing an array element arr[i] |
| O(log n) | Logarithmic Time | Binary Search |
| O(n) | Linear Time | Iterating through an array |
| O(n²) | Quadratic Time | Bubble Sort, Selection Sort |

O(n log n)/O(2ⁿ)/O(n!) even slower!

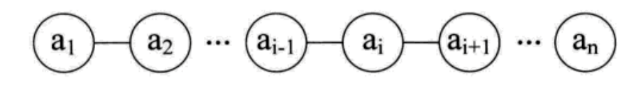
**2. Hash Tables**

A **Hash Table** (or Hash Map) stores key-value pairs using a **hash function** to determine the index in an array. It allows **O(1)** average-time complexity for searches, insertions, and deletions.

**3. Linked Lists**

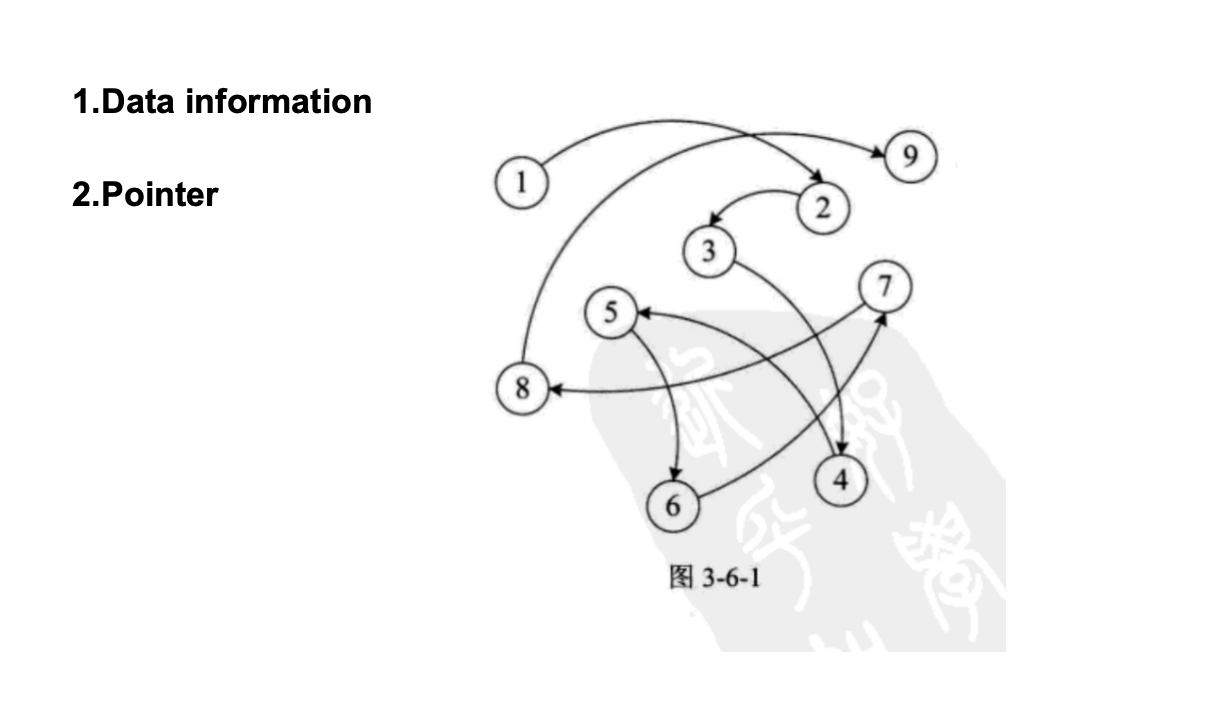
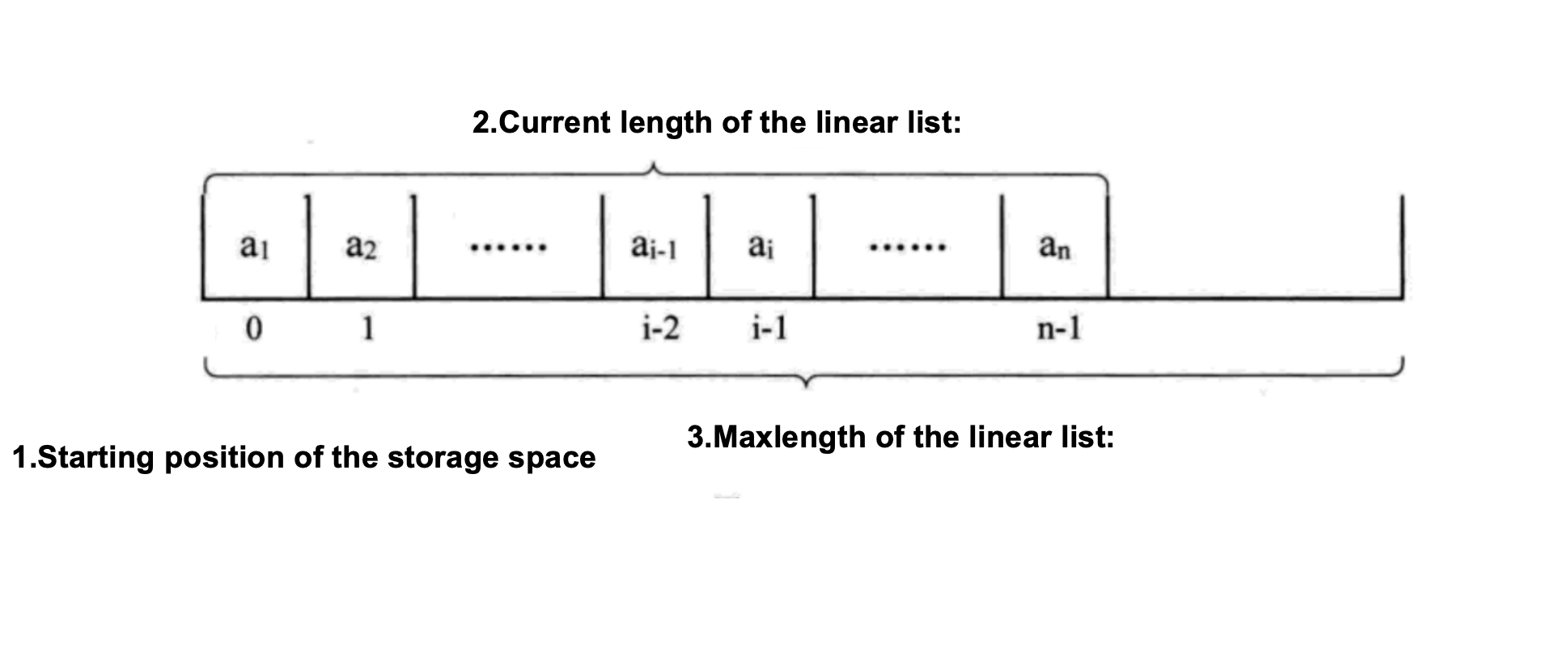
Linear Structures (Arraylist vs. Linked Lists)- Yi

Key: Elements have a one-to-one logical relationship.

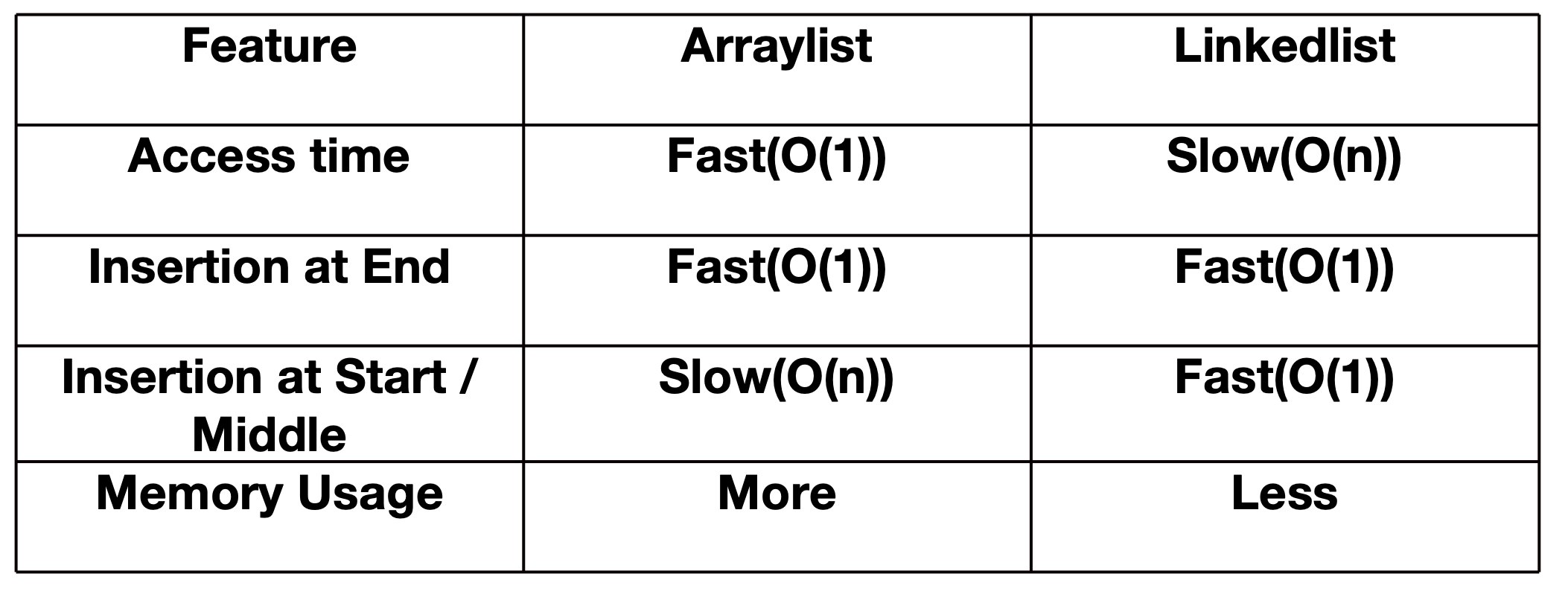


Two physical structures:

Arraylist： Linkedlist：



The contrast between the two structures



Conclusion:

Linear structures like arrays and linked lists each have unique strengths and weaknesses. Arrays excel in scenarios requiring rapid index-based access and frequent data scanning due to their direct memory access.