# What is the difference between ArrayList and LinkedList? When to use which one.?

ArrayList and LinkedList both implement Java's List interface but use different underlying structures: ArrayList employs a dynamic array for contiguous storage, while LinkedList uses a doubly-linked list of nodes. This leads to distinct performance profiles—ArrayList excels in random access at O(1) time via indexing, whereas LinkedList requires O(n) traversal for the same.

**Performance Comparison**

| Operation | ArrayList Time Complexity | LinkedList Time Complexity | Notes |
|---|---|---|---|
| Get (access by index) | O(1) | O(n) | ArrayList's array enables direct indexing. |
| Insert/Delete (middle) | O(n) | O(1) | LinkedList adjusts pointers without shifting. |
| Insert/Delete (ends) | Amortized O(1) | O(1) | Both efficient at boundaries. |
| Search | O(n) | O(n) | Linear scan for both. |

### When to Use ArrayList

ArrayList suits scenarios with frequent random access or reads, such as retrieving elements by index or iterating sequentially, due to its O(1) get operation and cache-friendly contiguous storage.

### When to Use LinkedList

Choose LinkedList for frequent insertions or deletions, especially at arbitrary positions or ends, leveraging its O(1) time for these via pointer adjustments without shifting elements. Ideal for queues, stacks, or dynamic lists