Week 2 Study Summary: Arrays, Binary Search, and Simple Sorts

Z5007 Programming and Data Structures · Oct 12–16, 2026 · Dr. Innocent Nyalala

1. Static Arrays, Binary Search, and Lower Bound

Contiguous memory gives $O(1)$ random reads and writes, whereas middle insertion or deletion shifts up to n elements in $O(n)$ time. On sorted arrays, binary search halves the candidate interval `[lo, hi]` each iteration in $O(\log n)$ comparisons. Using the half-open loop invariant (`while lo < hi`) finds `lower_bound(target)` cleanly without infinite loops.

2. Dynamic Arrays and Amortized Doubling

When a dynamic array of capacity C overflows, allocating a $2C$ block and copying C items costs $\Theta(C)$ for that single append. Across n appends starting from capacity 1, total copy operations sum to $1+2+4+\cdots+2^{\lfloor \log_2 n \rfloor} < 2n$, yielding $O(1)$ amortized cost per append. Shrinking by half only at 1/4 occupancy prevents oscillation thrashing at full capacity.

3. Bubble Sort vs. Selection Sort

Algorithm	Best Time	Worst Time	Swaps	Stable?
Bubble Sort (early stop)	$O(n)$	$O(n^2)$	Up to $n(n-1)/2$	Yes (use $>$)
Selection Sort	$O(n^2)$	$O(n^2)$	At most $n-1$	No

Worked Check

On $[1, 2, 3, 4, 5]$, early-stopping Bubble Sort performs 4 comparisons and 0 swaps. On $[10, 9, \ldots, 1]$, Selection Sort performs 45 comparisons and 5 swaps.