

Exercise walk-through

5.7

Searching

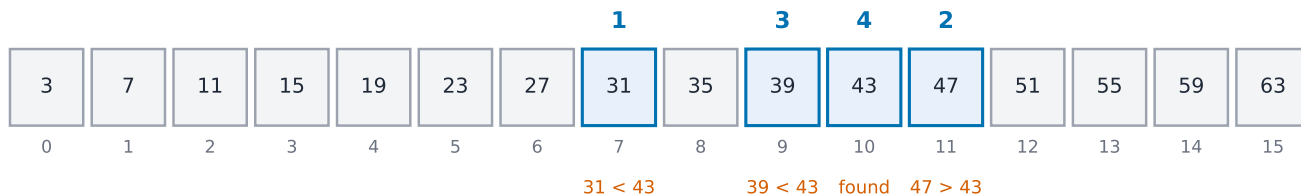
University Computer Science

Searching: halve the candidates

- 1 **Trail** Searching 43 probes indices 7 (31), 11 (47), 9 (39), 10 (43) - four probes; linear walks 11 values to the same slot.
- 2 **Miss** Searching 64 probes 7, 11, 13, 14, 15 - five probes, the worst case, then the empty range reports absent.
- 3 **Trap** Binary on unsorted data returns wrong answers silently; $lo = mid$ (not $mid + 1$) loops forever.

Binary-search probes

searching 43: numbered probes 1-4, one answer



probe order 7, 11, 9, 10 - each probe halves what remains: 16 slots resolve in at most 5 (linear pays up to 16)

The 16-slot sorted **array** with the probe trail for 43: mid at 7 (too small), 11 (too big), 9 (too small), 10 - found in 4 probes.

Check yourself

- Doubling n quadruples the count — the class is...? (Quadratic.)
- Keys 4, 17, 30, 43 under mod 13 land in...? (Bucket 4.)
- Children of **heap** index i sit at...? ($2i+1$ and $2i+2$.)
- Sixteen slots, **binary search**, worst case...? (5 probes.)

Exercises 2.1-2.3

2.1 A sorted array of 32 slots: give the worst-case binary probe count. [3]

2.2 State the update that makes binary search loop forever. [3]

2.3 Give the safe midpoint formula and the failure it prevents in fixed-width languages. [3]

Exercises 2.1-2.3

2.1 A sorted array of 32 slots: give the worst-case binary probe count. [3]

2.1

- $\lceil \log_2 33 \rceil = 6$ probes ✓.
-

2.2 State the update that makes binary search loop forever. [3]

2.2

- `lo = mid` (instead of `mid + 1`) with `hi` unchanged loops forever ✓.
-

2.3 Give the safe midpoint formula and the failure it prevents in fixed-width languages. [3]

2.3

- `lo + (hi - lo) // 2` ✓.
- Prevents `lo + hi` overflowing a fixed-width integer ✓.

Exercise 3.1

3.1 The sorted array is $[2, 6, 10, 14, 18, 22, 26, 30]$ (indices 0–7).

Search 22: give every probe index, the value read, and the range kept after each probe.

Search 5 (absent): give the probe trail and the final empty range. Give both worst-case bounds (8 slots) and the exact linear cost of each query. [8]

Exercise 3.1

3.1 The sorted array is $[2, 6, 10, 14, 18, 22, 26, 30]$ (indices 0–7).

Search 22: give every probe index, the value read, and the range kept after each probe.

Search 5 (absent): give the probe trail and the final empty range. Give both worst-case bounds (8 slots) and the exact linear cost of each query. [8]

3.1

- 22: mid 3 reads $14 < 22 \rightarrow lo=4$; mid 5 reads 22 – found: probes 3, 5 ✓.
- 5: mid 3 reads $14 > 5 \rightarrow hi=2$; mid 1 reads $6 > 5 \rightarrow hi=0$; mid 0 reads $2 < 5 \rightarrow lo=1 > hi=0$ – empty, absent ✓.
- Worst-case bounds (8 slots): $\lceil \log_2 9 \rceil = 4$ probes ✓.
- Linear: 22 sits at index 5 – 6 values; the miss 5 walks all 8 ✓.

Exercise 3.2

3.2 An unsorted array of 10^6 values must answer 10^4 queries.

Price all-linear: total comparisons. Price sort-then-binary: the $\Theta(n \log_2 n)$ sort cost (value at $n = 10^6$), plus the 10^4 queries at at most $\lceil \log_2(10^6 + 1) \rceil$ probes each (value of the ceiling). State the crossover condition on the number of queries that justifies sorting first. [7]

Exercise 3.2

3.2 An unsorted array of 10^6 values must answer 10^4 queries.

Price all-linear: total comparisons. Price sort-then-binary: the $\Theta(n \log_2 n)$ sort cost (value at $n = 10^6$), plus the 10^4 queries at at most $\lceil \log_2(10^6 + 1) \rceil$ probes each (value of the ceiling). State the crossover condition on the number of queries that justifies sorting first. [7]

3.2

- All-linear: $10^4 \times 10^6 = 10^{10}$ comparisons ✓.
- Sort: $\Theta(n \log_2 n) = 10^6 \times 19.93 \approx 1.99 \times 10^7$ comparisons ✓.
- Queries: $\lceil \log_2(10^6 + 1) \rceil = 20$ probes each – 2×10^5 total ✓.
- Sort-then-binary total $\approx 2 \times 10^7$, about 500 times less than 10^{10} ✓.
- Crossover: sorting wins once query count q satisfies $q \cdot n > n \log_2 n + q \log_2 n$ – roughly $q > \log_2 n \approx 20$ ✓.