KK12 — Algorithms for sorting and searching

Programming · Comprehensive exam study guide

STUDY DESIGN — YOU MUST KNOW

- selection sort
- quick sort
- binary search
- linear search

You must describe how each algorithm works, state any **precondition**, and compare their efficiency. Binary search's sorted-data precondition and the linear/binary comparison are classic exam questions.

Searching

Algorithm	How it works	Precondition	Efficiency
Linear search	Checks each element in turn until found or the end is reached.	None — works on unsorted data.	Slow on large lists (up to n checks).
Binary search	Repeatedly halves the list, comparing the middle element and discarding the half that can't contain the target.	Data **must be sorted** first.	Fast — about log■n checks.

Sorting

Algorithm	How it works	Notes
Selection sort	Repeatedly finds the smallest remaining element and places it next in order.	Simple but slow on large lists.
Quick sort	Picks a **pivot**, partitions items into those smaller/larger than it, then recursively sorts each part.	Much faster on large lists on average.

Efficiency message

Binary search is far faster than linear search but pays the cost of keeping data sorted. Quick sort is generally faster than selection sort on large data. 'Efficiency' here means fewer comparisons/operations as data grows.

REMEMBER

Binary search **requires sorted data** — stating this precondition is almost always worth a mark.

Key definitions to memorise

Linear search	A search that checks each element in sequence until the target is found or the list ends.
Binary search	A search that repeatedly halves a sorted list, discarding the half that cannot contain the target.

Selection sort	A sort that repeatedly selects the smallest remaining element and places it in order.
Quick sort	A sort that partitions data around a pivot and recursively sorts each partition.
Pivot	The reference element quick sort partitions the data around.

Exam technique

EXAM TIP

- Comparison questions: pair speed with precondition (binary is faster BUT needs sorted data).
- When describing an algorithm, mention the repeating step (halving / selecting smallest / partitioning).

COMMON MISTAKES

- Forgetting binary search's sorted-data precondition.
- Describing selection sort as 'swapping adjacent items' (that's bubble sort, not in this list).

Quick revision checklist

- ■ I can describe linear and binary search and state binary's precondition.
- ■ I can describe selection sort and quick sort and compare their efficiency.

Exam practice — questions & model answers

Q1. Compare linear search and binary search, including any precondition.

Compare · 4 marks

Model answer

- Linear search checks each element in turn and works on unsorted data but is slow on large lists (2).
- Binary search repeatedly halves a list (≈log■n checks) and is much faster, but requires the data to be sorted first (2).

Q2. Describe how selection sort orders a list.

Describe · 2 marks

Model answer

- It repeatedly finds the smallest remaining element (1)
- and places it into its correct position, growing the sorted part each pass (1).

Q3. A program searches a large, frequently-updated unsorted list once. Recommend a search algorithm and justify your choice.

Recommend/Justify · 2 marks

Model answer

- Linear search (1),
- because the data is unsorted and sorting it for a single search (to enable binary search) would not be worthwhile (1).