

Exercise walk-through

# 13.2

## File organisation and access

---

A-Level Computer Science

# Three ways to lay records out

## File organisation

is the layout; **file access** is how you reach a record.

### serial

records in the *order added*. Append is fast; search is slow. Logs.

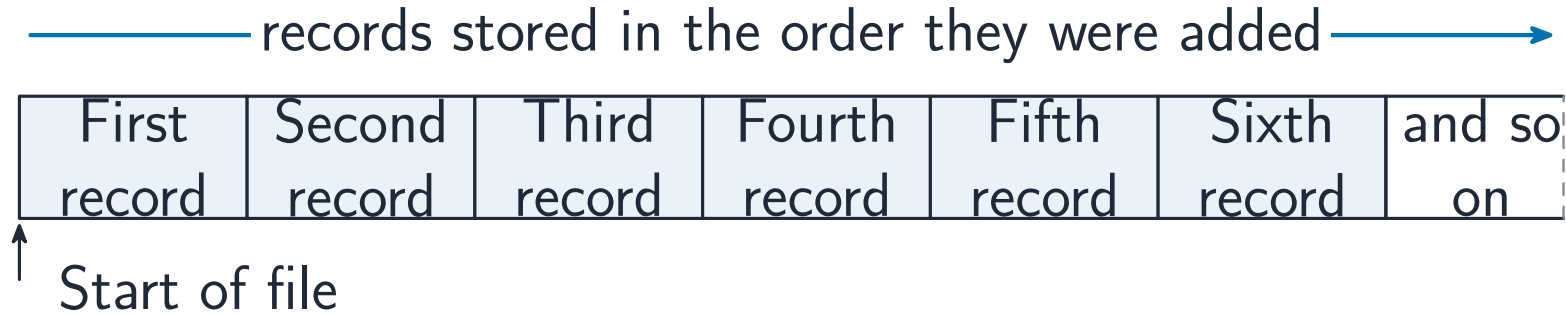
### sequential

sorted on a *key*. Search can stop early; insert must shift.

### random

position from the key (often a hash). Direct lookup is fast.

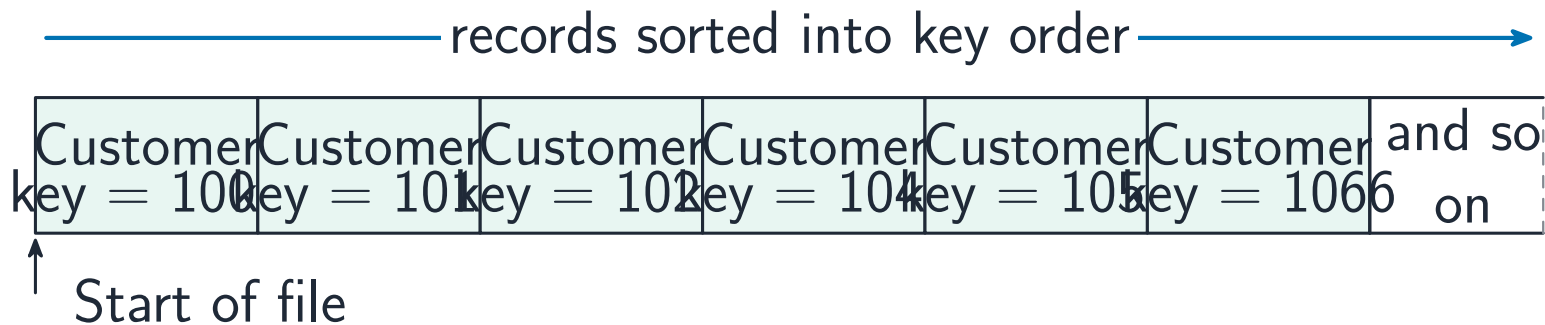
## Serial – records in arrival order



New records go on the **end**. Simple to write, but any search reads from the start.

A **serial file** appends each record as it arrives, so writing is fast and finding one means reading from the start.

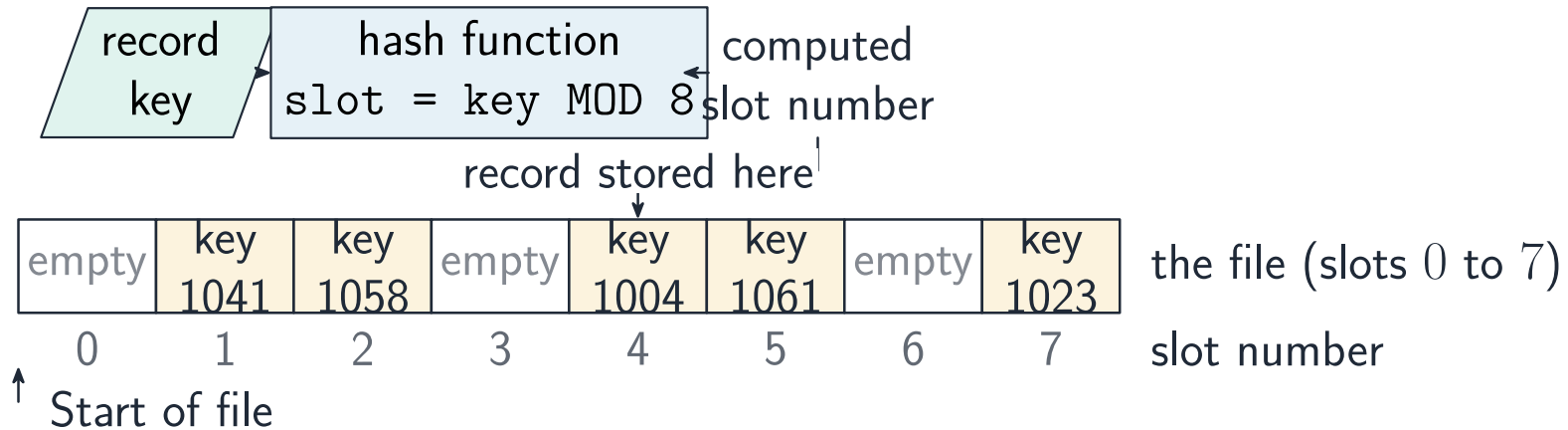
## Sequential – ordered by a key



Records are kept in **key order**, so a search can stop early – but an insertion means rewriting the file.

A **sequential file** keeps records **sorted by a key**, so a search can stop early – but every insertion has to make room.

## Random – the key computes the address



A **hash** of the key gives the record's position, so one read finds it – **direct access**, at the cost of managing collisions.

# Sequential access vs direct access

- **sequential access** – read from the start, one record after another
- **direct access** – jump straight to a known position

Serial and sequential files **need** sequential access. A random file **allows** direct access.

Match the layout to the dominant operation: single-key lookups favour random; in-order reports favour sequential.

serial / sequential

walk from the start

random

hash, then jump

## Exercises 2.1-2.3

---

2.1 State the difference between a **serial** file and a **sequential** file. [2]

---

2.2 State which file organisation gives the **fastest single-key lookup**. [1]

---

2.3 State what a **hash function** does. [1]

## Exercises 2.1-2.3

---

**2.1** State the difference between a **serial** file and a **sequential** file. [2]

**2.1**

- a **serial** file keeps records in the **order they were added** (unsorted)
- a **sequential** file keeps them **sorted by a key field**

---

**2.2** State which file organisation gives the **fastest single-key lookup**. [1]

**2.2**

- a **random (direct-access)** file

---

**2.3** State what a **hash function** does. [1]

**2.3**

- it turns a **record key** into an **address/slot** where the record is stored

## Exercises 2.4.3.1

---

2.4 State what a **collision** is.

[1]

---

3.1 A file uses  $\text{address} \leftarrow \text{key} \bmod 10$ . Give the slot number for the keys 45, 23 and 30.

[3]

## Exercises 2.4-3.1

---

2.4 State what a **collision** is.

[1]

2.4

- when **two different keys** hash to the **same address/slot**

---

3.1 A file uses  $\text{address} \leftarrow \text{key} \bmod 10$ . Give the slot number for the keys 45, 23 and 30.

[3]

3.1

- $45 \bmod 10 = 5$
- $23 \bmod 10 = 3$
- $30 \bmod 10 = 0$

## Exercises 3.2-3.3

---

**3.2** Keys 45 and 35 both hash to slot 5. Explain how **linear probing** stores the second one. [2]

---

**3.3** State **one** advantage and **one** disadvantage of a **random (direct-access)** file. [2]

## Exercises 3.2-3.3

---

**3.2** Keys 45 and 35 both hash to slot 5. Explain how **linear probing** stores the second one. [2]

### 3.2

- $35 \text{ MOD } 10 = 5$ , but slot 5 is **taken** by 45
- linear probing tries the **next slot** (slot 6), and stores 35 there (the first free slot, wrapping if needed)

---

**3.3** State **one** advantage and **one** disadvantage of a **random (direct-access)** file. [2]

### 3.3

- advantage: very **fast direct lookup** by key
- disadvantage: reading in **key order** is harder, and collisions must be handled (space may be wasted)

## Exercise 3.4

---

**3.4** A bank must look up an account **quickly** by its account number. State the best file organisation and **justify** your choice. [2]

## Exercise 3.4

---

**3.4** A bank must look up an account **quickly** by its account number. State the best file organisation and **justify** your choice. [2]

**3.4**

- a **random (direct-access)** file using a **hash** of the account number
- because it gives near-instant lookup of a single record without searching the whole file