

# COMP 3311

# DATABASE MANAGEMENT

# SYSTEMS

## LECTURE 14 EXERCISES

## QUERY PROCESSING:

## INTRODUCTION

# EXERCISE 1

For the relation Apartment(code, year, price, area), we want to process the query

Find the apartments in a given set of areas.

There are 1,000,000 apartments uniformly distributed over 100 areas. 10 Apartment records fit in one page. The records are ordered on price. There is a hash index on code and a dense, non-clustering B<sup>+</sup>-tree index on area. Each B<sup>+</sup>-tree node can fit 20 pointers (19 values). 100 record pointers can fit on a page. Assume that each non-leaf B<sup>+</sup>-tree node holds the minimum number of values.

Is it advantageous, in terms of page I/Os, to use the B<sup>+</sup>-tree on area to answer the query if there are 5 area values?

**Cost for file scan:**  $\lceil 1,000,000 / 10 \rceil = 100,000$  page I/Os

**Query:** Find the apartments in five areas.

## EXERCISE I (cont'd)

Apartment records: 1,000,000  
 $bf_{Apartment}$ : 10  
Number of areas: 100  
# records / area: 10,000  
B<sup>+</sup>-tree node: 20 entries  
Record pointers/page: 100

### Cost to use B<sup>+</sup>-tree index

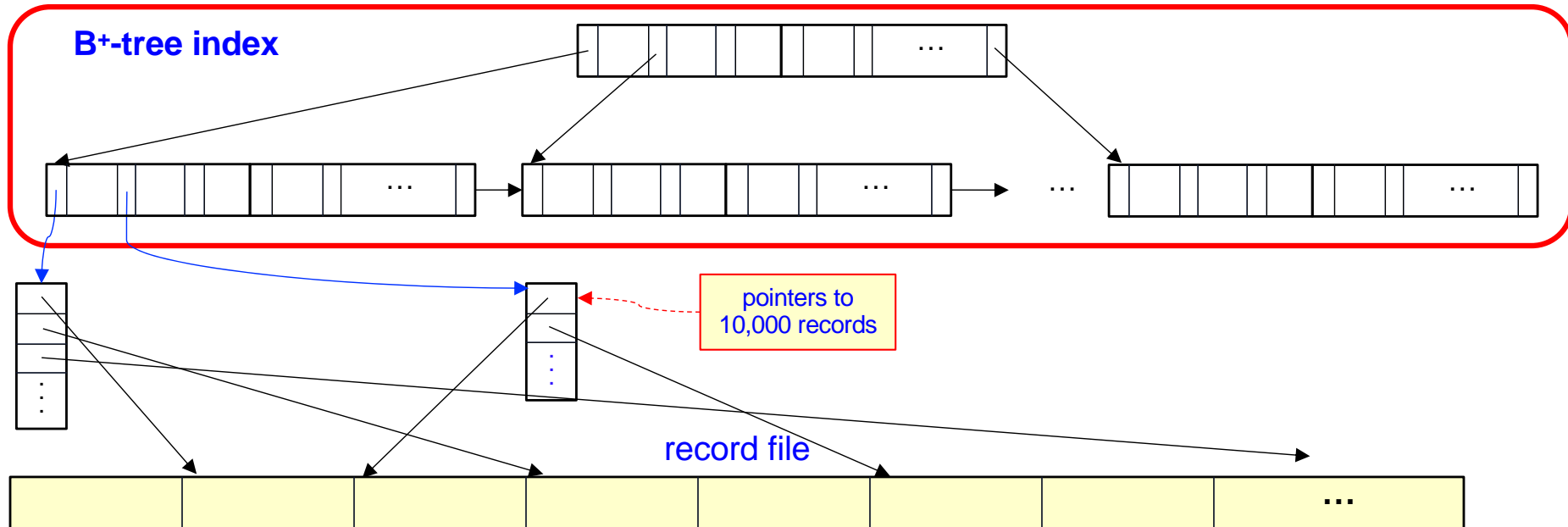
- The 1,000,000 apartments are distributed uniformly over 100 areas. Thus, there are on average  $1,000,000/100=10,000$  apartments per area.
- For the range of 5 values, on average,  $5 * 10,000 = 50,000$  records will be accessed.
- Since 10 Apartment records fit on a page,  $\lceil 50,000/10 \rceil = 5,000$  pages are needed to store these records.
- The B<sup>+</sup>-tree on area will have a height of  $\lceil \log_{\lceil 20/2 \rceil} 100 \rceil = 2$  (i.e., fan out is  $\lceil n/2 \rceil = \lceil 20/2 \rceil = 10$  since we assume *minimum node occupancy*).
- Each B<sup>+</sup>-tree leaf node will point to a (indirection) list of apartment records in a specific area.
- Since there are 10,000 apartments per area and 100 record pointers can fit on a page,  $\lceil 10,000/100 \rceil = 100$  pages are required to store the pointers for each area.

**Query:** Find the apartments in five areas.

Apartment records: 1,000,000  
 $bf_{\text{Apartment}} = 10$   
Number of areas: 100  
# records / area: 10,000  
B<sup>+</sup>-tree height: 2  
Record pointers/page: 100

**File scan cost:**  
100,000 page I/Os

## EXERCISE 1 (cont'd)



Cost to retrieve the apartment records for 5 areas using B<sup>+</sup>-tree index.

Cost to search B<sup>+</sup>-tree: # areas \* B<sup>+</sup>-tree height =  $5 * 2 = 10$  page I/Os

Cost to retrieve indirection pointers to records:  $5 * 100 = 500$  page I/Os

Cost to retrieve all records:  $5 * 10,000 = 50,000$  page I/Os

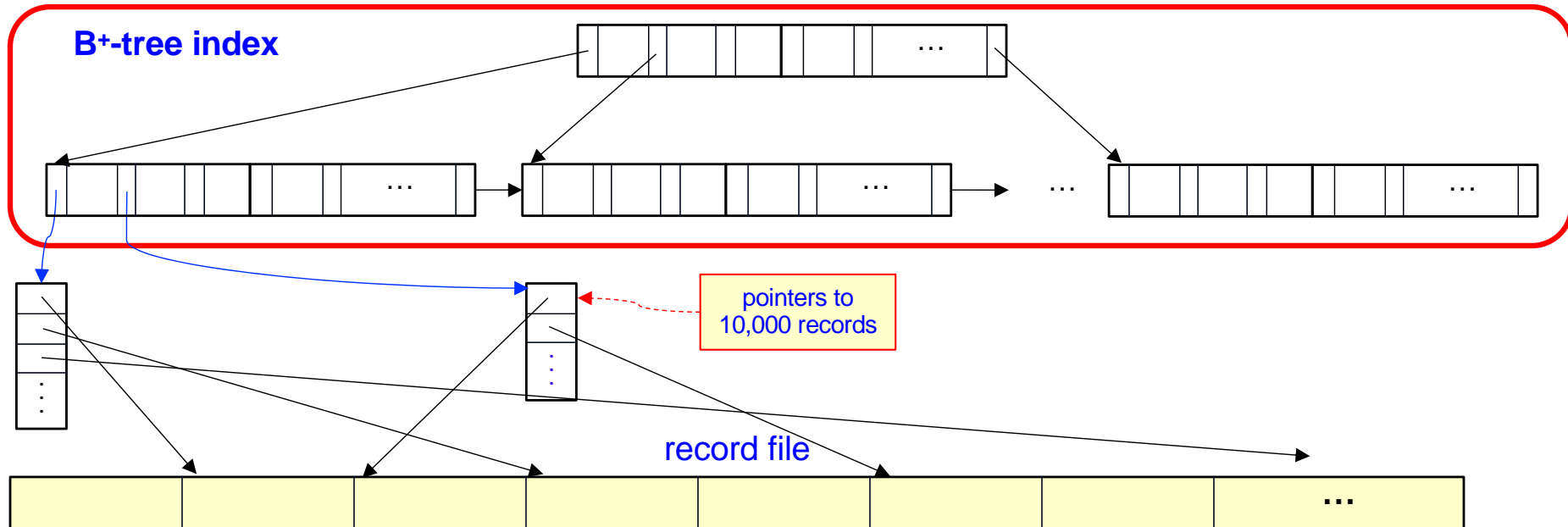
**Total cost:**  $10 + 500 + 50,000 = 50,510$  page I/Os

**Query:** Find the apartments in ten areas.

Apartment records: 1,000,000  
 $bf_{\text{Apartment}} = 10$   
Number of areas: 100  
# records / area: 10,000  
B<sup>+</sup>-tree height: 2  
Record pointers/page: 100

**File scan cost:**  
100,000 page I/Os

## EXERCISE 1 (cont'd)



Suppose the range of area is increased to 10. How does the cost change?

Cost to search B<sup>+</sup>-tree: # areas \* B<sup>+</sup>-tree height =  $10 * 2 = 20$  page I/Os

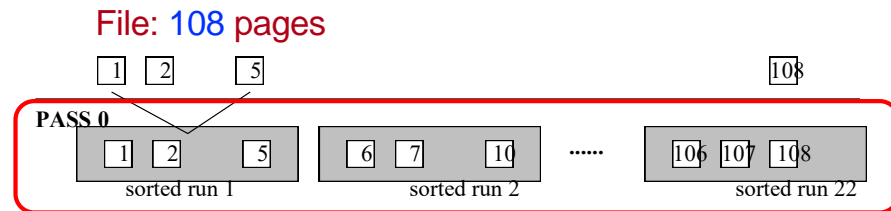
Cost to retrieve indirection pointers to records:  $10 * 100 = 1,000$  page I/Os

Cost to retrieve all records:  $10 * 10,000 = 100,000$  page I/Os

**Total cost:**  $20 + 1,000 + 100,000 = \underline{101,020}$  page I/Os

## EXERCISE 2

How many passes are needed to sort a file that has 108 pages, using only 5 pages of main memory? What is the page I/O cost?



### Number of passes

Pass 0: Sort records producing  
 $\lceil 108/5 \rceil = 22$  runs

Pass 1: 4-way merge producing  
 $\lceil 22/4 \rceil = 6$  runs

Pass 2: 4-way merge producing  
 $\lceil 6/4 \rceil = 2$  runs

Pass 3: 2-way merge producing 1  
run (final sorted file)

👉 1 sort pass

👉 3 merge passes

### Page I/O cost (sort & merge)

Each pass we read and write 108  
pages times 4 passes =

$$2 \times 108 \times 4 = \underline{864}$$

r/w      passes

## EXERCISE 3

Assume that for the relation Sailor(sailorId, sName, rating, age) each record is 50 bytes, a page can hold 80 records, there are 500 pages of records and the file is unordered. We want to process the following two queries:

- i.  $\sigma_{\text{sailorId} < 50,000}(\text{Sailor})$
- ii.  $\sigma_{\text{sailorId} = 50,000}(\text{Sailor})$

Estimate the page I/Os for each query given the information in a) and b) below.

a) Assume that we have a B<sup>+</sup>-tree index on the search key sailorId with height 5, 50 index pages, the lowest index search-key value is 1 and highest index search-key value is 100,000.

- i. Since this is a range query and the file is not ordered on sailorId, the best strategy is to do a **file scan** of all pages.

**Total page I/Os:** 500 page I/Os

- ii. Use the B<sup>+</sup>-tree index to search for sailorId=50000. The cost is that to search the B<sup>+</sup>-tree, 5 page I/Os, plus the cost to retrieve the record, 1 page I/O.

**Total page I/Os:** 6 page I/Os

How does the cost change if the file is ordered on sailorId?



## EXERCISE 3 (cont'd)

Assume that for the relation Sailor(sailorId, sName, rating, age) each record is 50 bytes, a page can hold 80 records, there are 500 pages of records and the file is unordered. We want to process the following two queries:

- i.  $\sigma_{\text{sailorId} < 50000}(\text{Sailor})$
- ii.  $\sigma_{\text{sailorId} = 50000}(\text{Sailor})$

Estimate the page I/Os for each query given the information in a) and b) below.

b) Assume that we have a hash index on the search key sailorId with 50 index pages, no overflow, the lowest index search-key value is 1 and highest index search-key value is 100,000.

- i. Since this is a range query, a **hash index is not useful**. Thus, the only option is to do a **file scan** of all pages.

**Total cost:** 500 page I/Os

- ii. Use the hash index to search for sailorId=50000. The cost is that to access the hash index, 1 page I/O, plus the cost to retrieve the record, 1 page I/O.

**Total cost:** 2 page I/Os

How does the cost change if the file is ordered on sailorId?

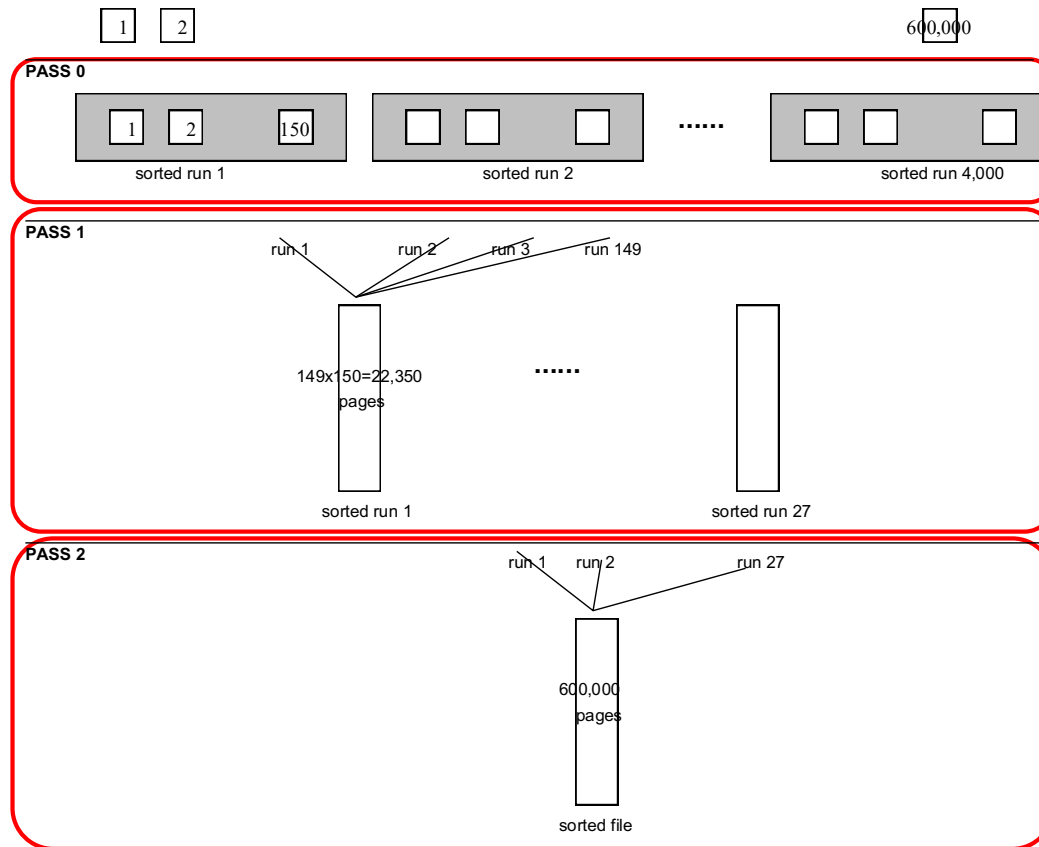




# EXERCISE 4

How many passes are needed to sort a file that has 600,000 pages, using 150 pages of main memory? What is the page I/O cost?

File: 600,000 pages



$$M=150 \quad B_f=600,000$$

## Number of passes

Pass 0: Sort records producing  $\lceil 600000/150 \rceil = 4000$  runs

Pass 1: 149-way merge producing  $\lceil 4000/149 \rceil = 27$  runs

Pass 2: 27-way merge producing  $\lceil 27/27 \rceil = 1$  run

👉 1 sort pass

👉 2 merge passes

## Page I/O cost (sort & merge)

Each pass we read and write 600,000 pages times 3 passes =  $2 \times 600000 \times 3 = 3,600,000$

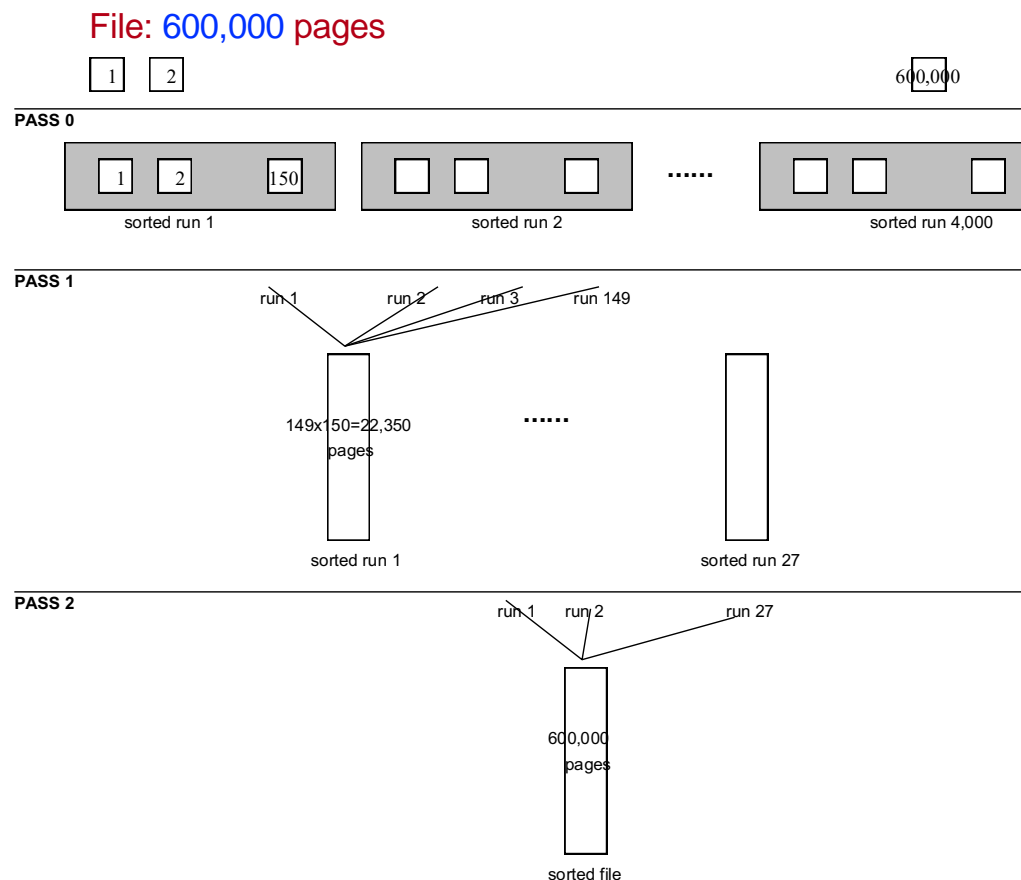
r/w

passes



## EXERCISE 4 (cont'd)

How many passes are needed to sort a file that has 600,000 pages, using 150 pages of main memory? What is the page I/O cost?



$$M=150 \quad B_r=600,000$$

**Number of merge passes**

$$\lceil \log_{M-1}(B_r/M) \rceil =$$

$$\lceil \log_{149}(600000/150) \rceil = \underline{2}$$

**I/O cost (sort & merge)**

$$2 * B_r * (\lceil \log_{M-1}(B_r/M) \rceil + 1) =$$

$$2 * 600000 * (\lceil \log_{149}(4000) \rceil + 1) =$$

$$2 * 600000 * (3) = \underline{3,600,000}$$

## EXERCISE 5

Let  $B$  be the size of a file in pages. How many memory pages  $M$  do you need to sort the file using one sorting pass and one merge pass (i.e., 2 passes total)? Show your calculations.

- At pass 0, we will create  $k = \lceil B/M \rceil$  sorted runs.
- In order to merge these sorted runs in one more pass  $k \leq M-1$ .
- By combining these two equations we get:

$$M-1 \geq \lceil B/M \rceil \text{ or } M \geq \lceil B/M \rceil + 1.$$

- An approximate solution for the above inequality is that  $M > \sqrt{B}$ .