write three more examples like in this photo and also write two more examples with solutions

Sample Question

- 10,000,000 tuples of 160 bytes each = 1.6GB file.
  - Block size 16K (2^{14}) blocks, each holding 100 tuples
  - Entire file takes 100,000 blocks
  - 100MB available main memory

How many phases and how many I/Os would this need if we sort the file using Multiway Merge Sort?

Exercise 1

Sample Question:

- **Dataset:** 20,000,000 tuples of 200 bytes each = **4GB file**
- **Block Details:**
  - Block size **32K** (2^{15}), each block holds **160 tuples**
  - Entire file takes **125,000 blocks**
  - **200MB** available main memory

How many phases and how many I/Os would be needed if we sort the file using Multiway Merge Sort?

Exercise 2

Sample Question:

- **Dataset:** 50,000,000 tuples of 128 bytes each = **6.4GB file**
- **Block Details:**
  - Block size **8K** (2^{13}), each block holds **64 tuples**
  - Entire file takes **781,250 blocks**
  - **500MB** available main memory

How many phases and how many I/Os would be needed if we sort the file using Multiway Merge Sort?

Exercise 3

Sample Question:

- **Dataset:** 5,000,000 tuples of 256 bytes each = **1.2GB file**
- **Block Details:**
  - Block size **64K** (2^{16}), each block holds **250 tuples**
  - Entire file takes **20,000 blocks**
  - **50MB** available main memory

How many phases and how many I/Os would be needed if we sort the file using Multiway Merge Sort?

Exercises with Solutions

Exercise 4 (With Solution)

- **Dataset:** 12,000,000 tuples of 180 bytes each = **2.16GB file**
- **Block Details:**
  - Block size **16K** (2^{14}), each block holds **89 tuples**
  - Entire file takes **134,831 blocks**
  - **150MB** available main memory

Solution:

1. **Determine how many blocks fit into memory:**
   - Memory available = **150MB**
   - Block size = **16KB**
   - Number of blocks that fit in memory:
   $$\frac{150MB}{16KB} = \frac{150 \times 1024}{16} = 9600 \text{ blocks}$$
2. **Number of initial sorted runs:**
   - Total blocks = **134,831**
   - Each run sorts at most **9600** blocks
   - Number of runs:
   $$\frac{134,831}{9600} \approx 14.04 \Rightarrow 15 \text{ runs (round up)}$$
3. **Number of merge phases:**
   - In each merge phase, we merge at most **9600 runs**
   - Since we have **15 initial runs**, they can all be merged in **one phase**
4. **Total I/O Cost:**
   - **Reading the file initially: 134,831 I/Os**
   - **Writing out sorted runs: 134,831 I/Os**
   - **Final merge phase (reading and writing): 2 × 134,831 = 269,662 I/Os**
   - **Total I/O operations:**
   $$134,831 + 134,831 + 269,662 = 539,324 \text{ I/Os}$$

Exercise 5 (With Solution)

- **Dataset:** 30,000,000 tuples of 100 bytes each = **3GB file**
- **Block Details:**
  - Block size **8K** (2^{13}), each block holds **80 tuples**
  - Entire file takes **375,000 blocks**
  - **250MB** available main memory

Solution:

1. **Determine how many blocks fit into memory:**
   - Memory available = **250MB**
   - Block size = **8KB**
   - Number of blocks that fit in memory:

$$\frac{250MB}{8KB} = \frac{250 \times 1024}{8} = 32,000 \text{ blocks}$$

2. **Number of initial sorted runs:**
   - Total blocks = **375,000**
   - Each run sorts at most **32,000** blocks
   - Number of runs:

$$\frac{375,000}{32,000} \approx 11.72 \Rightarrow 12 \text{ runs (round up)}$$

3. **Number of merge phases:**
   - In each merge phase, we merge at most **32,000 runs**
   - Since we have **12 initial runs**, they can all be merged in **one phase**
4. **Total I/O Cost:**
   - **Reading the file initially: 375,000 I/Os**
   - **Writing out sorted runs: 375,000 I/Os**
   - **Final merge phase (reading and writing): 2 × 375,000 = 750,000 I/Os**

Total I/O operations: 375,000 + 375,000 + 750,000 = 1,500,000 I/Os

---

Sorting Example

Setup:

- 10^7 records of 100 bytes = 10^9 bytes file.
  - Stored on disk 1, with 4Kb blocks, each holding 40 records + header information.
  - Entire file takes 250,000 blocks ≈ 1000 cylinders.
- 50Mb available main memory = 12,800 blocks ≈ 1/20th of file.
- Sort by primary key field.

M= 50MB

B = 4KB

R = 100 Bytes

$M^2/RB = (50*2^{20})^2 / (100 * 4*2^{10}) \sim 6.7$ billion records

ie. we can increase the number of records to 6.7 billion and still sort in 2 phases.

Sorting Example

Setup:

- **Dataset:** 5×10^7 records of 150 bytes each = **7.5GB file**
- **Storage Details:**
  - Stored on disk, with **8KB blocks**, each holding **53 records + header information**
  - Entire file takes **1,000,000 blocks**
- **Memory Details:**
  - **100MB** available main memory = **12,500 blocks ≈ 1/80th of the file**
- **Sorting field:** Primary key

Given Parameters:

- $M = 100MB$
- $B = 8KB$
- $R = 150$ Bytes

Compute Sorting Limit:

$$M^2/RB = (100 \times 2^{20})^2/(150 \times 8 \times 2^{10})$$

$$\approx 10.2 \text{ billion records}$$

➡️ **We can increase the number of records to 10.2 billion and still sort in 2 phases.**

Sorting Example

Setup:

- **Dataset:** 2×10^8 records of 200 bytes each = **40GB file**
- **Storage Details:**
  - Stored on disk, with **16KB blocks**, each holding **80 records + header information**
  - Entire file takes **2,500,000 blocks**
- **Memory Details:**
  - **500MB** available main memory = **31,250 blocks ≈ 1/80th of the file**
- **Sorting field:** Primary key

Given Parameters:

- $M = 500MB$
- $B = 16KB$
- $R = 200$ Bytes

Compute Sorting Limit:

$$M^2/RB = (500 \times 2^{20})^2/(200 \times 16 \times 2^{10})$$

$$\approx 83 \text{ billion records}$$

➡️ **We can increase the number of records to 83 billion and still sort in 2 phases.**

Sorting Example

Setup:

- **Dataset:** 10^6 records of 250 bytes each = **250MB file**
- **Storage Details:**
  - Stored on disk, with **4KB blocks**, each holding **16 records + header information**
  - Entire file takes **62,500 blocks**
- **Memory Details:**
  - **20MB** available main memory = **5,000 blocks ≈ 1/12th of the file**
- **Sorting field:** Primary key

Given Parameters:

- $M = 20MB$
- $B = 4KB$
- $R = 250$ Bytes

Compute Sorting Limit:

$$M^2/RB = (20 \times 2^{20})^2/(250 \times 4 \times 2^{10})$$

$$\approx 1.6 \text{ billion records}$$

➡️ **We can increase the number of records to 1.6 billion and still sort in 2 phases.**

Example 4 (With Solution)

Sorting Example

Setup:

- **Dataset:** 3×10^7 records of 120 bytes each = **3.6GB file**
- **Storage Details:**
  - Stored on disk, with **8KB blocks**, each holding **66 records + header information**
  - Entire file takes **450,000 blocks**
- **Memory Details:**
  - **75MB** available main memory = **9,375 blocks ≈ 1/48th of the file**
- **Sorting field:** Primary key

Given Parameters:

- $M = 75MB$
- $B = 8KB$
- $R = 120$ Bytes

Compute Sorting Limit:

$$M^2/RB = (75 \times 2^{20})^2/(120 \times 8 \times 2^{10})$$

$$\approx 6.4 \text{ billion records}$$

Steps:

1. **Initial Sorted Runs:**
   - Memory holds **9,375 blocks**
   - Initial sorted runs = $\frac{450{,}000}{9{,}375} \approx 48$ runs
2. **Number of Merge Phases:**
   - Each merge step can merge up to **9,375 runs**
   - Since we only have **48 runs**, everything can be merged in **one phase**

Final Result:

✅ **Sorting will complete in 2 phases with a total I/O cost of**

$$2 \times 450,000 + 450,000 = 1,350,000 \text{ I/Os}$$

Example 5 (With Solution)

Sorting Example

Setup:

- **Dataset:** 5×10^8 records of 90 bytes each = **45GB file**
- **Storage Details:**
  - Stored on disk, with **16KB blocks**, each holding **177 records + header information**
  - Entire file takes **2,500,000 blocks**
- **Memory Details:**
  - **200MB** available main memory = **12,500 blocks ≈ 1/200th of the file**
- **Sorting field:** Primary key

Given Parameters:

- $M = 200MB$
- $B = 16KB$
- $R = 90$ Bytes

Compute Sorting Limit:

$$M^2/RB = (200 \times 2^{20})^2/(90 \times 16 \times 2^{10})$$
$$\approx 44 \text{ billion records}$$

Steps:

1. **Initial Sorted Runs:**
   - Memory holds **12,500 blocks**
   - Initial sorted runs = $\frac{2{,}500{,}000}{12{,}500} = 200$ runs
2. **Number of Merge Phases:**
   - Each merge step can merge up to **12,500 runs**
   - Since we only have **200 runs**, everything can be merged in **one phase**

Final Result:

✅ **Sorting will complete in 2 phases with a total I/O cost of**

$$2 \times 2,500,000 + 2,500,000 = 7,500,000 \text{ I/Os}$$