# CÂU HỎI TRẮC NGHIỆM & BÀI TẬP - EXTERNAL SORTING

## I. CÂU HỎI TRẮC NGHIỆM

1. External Sorting thường được sử dụng khi nào?

* A. Khi dữ liệu vừa đủ nằm trong RAM
* B. Khi cần sắp xếp dữ liệu có cấu trúc phức tạp
* **C. Khi dữ liệu quá lớn không thể chứa hết trong bộ nhớ chính**
* D. Khi tốc độ CPU là yếu tố giới hạn

*Giải thích*: Vì external sorting dùng khi **RAM không đủ chứa toàn bộ dữ liệu**, nên cần xử lý từ từ qua file.

2. Trong thuật toán Balanced Multiway Merge Sort, số lượng run có thể merge cùng lúc là bao nhiêu nếu có M block bộ nhớ?

* A. M
* B. M + 1
* **C. M - 1**
* D. M – 2

*Giải thích*: 1 buffer cho output, M - 1 còn lại để đọc các run đầu vào.

3. Thuật toán nào tận dụng các dãy tăng liên tiếp có sẵn trong dữ liệu để tạo các run ban đầu?

* A. 2-Way Merge Sort
* B. Balanced Merge Sort
* C. Heap Sort
* **D. Natural Merge Sort**

*Giải thích*: Natural Merge tự động phát hiện các dãy tăng liên tục trong dữ liệu đầu vào.

4. Lợi ích của việc sử dụng Min-Heap trong giai đoạn merge là gì?

* A. Giảm thời gian đọc đĩa
* B. Dễ cài đặt hơn
* **C. Giảm độ phức tạp từ O(k) xuống O(log k) khi tìm phần tử nhỏ nhất**
* D. Tăng dung lượng bộ nhớ cần dùng

*Giải thích:* Giúp tìm phần tử nhỏ nhất trong K run hiệu quả hơn.

5. Pass đầu tiên trong Multiway Balanced Sort thường thực hiện việc gì?

* A. Đếm số lượng bản ghi
* **B. Tạo các run ban đầu bằng cách đọc và sắp xếp dữ liệu trong bộ nhớ**
* C. Merge toàn bộ dữ liệu
* D. Kiểm tra dữ liệu có bị lỗi hay không

*Giải thích:* Tạo các run ban đầu bằng cách đọc và sắp xếp dữ liệu trong bộ nhớ

6. Tìm các đường chạy trong mảng sau [3, 8, 12, 5, 6, 9, 15, 4, 10, 11]

* **A [3, 8, 12], [5, 6, 9, 15], [4, 10, 11]**
* B. [3, 8], [12, 5, 6], [9, 15, 4, 10, 11]
* C. [3, 8, 12, 5], [6, 9], [15, 4, 10, 11]
* D. [3], [8, 12, 5, 6], [9, 15, 4, 10], [11]

*Giải thích:* Tìm các chuỗi tăng liên tục.

7. Nhược điểm lớn nhất của Natural Merge Sort

* **A Thời gian đọc ghi dữ liệu rất lớn**
* B. Không thể xử lý dữ liệu lớn ngoài bộ nhớ
* C. Cần nhiều bộ nhớ hơn so với các thuật toán sắp xếp khác
* D. Luôn tạo số lượng pass tối đa dù dữ liệu đã gần như được sắp xếp

*Giải thích:* Nếu dữ liệu có nhiều đường chạy tự nhiên -> nhiều runs. Việc chỉ trộn hai runs cùng lúc sẽ làm tăng thao tác đọc ghi -> tăng thời gian

8.Trong Multiway Merge Sort, nếu có M buffer, bao nhiêu buffer dùng để đọc run?

* A. M
* **B. M - 1**
* C. M - 2
* D. 1

*Giải thích:* 1 buffer cần thiết để ghi output, còn lại M - 1 buffer dùng để đọc từ các run đầu vào.

9. Bạn có 6 buffer (M = 6), mỗi lần merge tối đa bao nhiêu run trong Multiway Merge Sort

* **A. 5**
* B. 6
* C. 4
* D. 3

*Giải thích:* Dùng 5 buffer để đọc 5 run, 1 buffer để ghi → tối đa 5 run mỗi lần merge.

10. Lợi ích chính của Multiway Merge so với 2-Way Merge là gì?

* A. Tạo run nhanh hơn
* B. Ít cần bộ nhớ hơn
* **C. Giảm số lần ghi và đọc từ đĩa**
* D. Không cần tạo run ban đầu

*Giải thích:* Vì merge nhiều run một lần (k-way) nên số lần truy cập đĩa (I/O) giảm đáng kể so với merge 2-way.

11. Trong Multiway Merge Sort, cấu trúc dữ liệu nào giúp chọn phần tử nhỏ nhất hiệu quả?

* A. Stack
* B. Queue
* **C. Min Heap**
* D. Linked List

*Giải thích:* Min-Heap cho phép chọn phần tử nhỏ nhất từ K run đầu vào với chi phí O(log K).

12.Điều gì xảy ra nếu số lượng run vượt quá số lượng buffer trong Multiway Merge?

* A. Tăng tốc độ sắp xếp
* **B. Chia thành nhiều đợt merge (nhiều pass)**
* C. Merge tất cả một lúc
* D. Dừng thuật toán

*Giải thích:* Không thể merge tất cả run cùng lúc → phải chia thành nhiều lượt merge (multi-pass).

13. Chia mảng [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] thành các mảng con sao cho phù hợp với MemorySize = 3

* A. **[[1, 2, 3],[4, 5, 6],[7, 8, 9],[10]]**
* B. [[1, 2, 3, 4, 5], [6, 7, 8], [9, 10]]
* C. [[1], [2], [3], [4], [5], [6], [7], [8], [9], [10]]
* D. [[1, 2, 3], [4, 5], [6, 7, 8], [9, 10, 11]]

*Giải thích:* Mỗi lần chỉ load được tối đa 3 phần tử vào RAM → chia thành từng block 3 phần tử.

14. Xây dựng Min-Heap từ mảng [7, 3, 10, 5, 1]

* A. [1, 3, 5, 7, 10]
* B. [3, 10, 5, 1, 7]
* **C. [1, 3, 10, 5, 7]**
* D. [1, 3, 10, 5, 7]

*Giải thích:* Vận dụng cấu trúc dữ liệu MinHeap và các hàm của nó

15. Merge Sort có độ phức tạp thời gian tốt nhất, trung bình và tệ nhất là gì?

* A. O(n), O(n log n), O(n²)
* **B. O(n log n), O(n log n), O(n log n)**
* C. O(n log n), O(n²), O(n²)
* D. O(log n), O(n log n), O(n²)

16. Trong Natural Merge Sort, nếu toàn bộ mảng là một chuỗi tăng, ta cần bao nhiêu lượt trộn?

* A. log n
* **B. 1**
* C. n/2
* D. 0

*Giải thích:* Nếu mảng là một run dài → chỉ cần 1 pass để hoàn tất (không cần merge nhiều lần).

17. Trong K-way Merge Sort, “K” đề cập đến?

* A. Số lần chia mảng
* B. Số phần tử trong một block
* C. Số luồng xử lý song song
* **D. Số đoạn con được trộn cùng lúc**

*Giải thích:* K là số run được merge đồng thời trong mỗi pass.

18**.** Độ phức tạp thời gian của K-way Merge bằng cách dùng Min-Heap là?

* A. O(n log n)
* **B. O(n log K)**
* C. O(K log n)
* D. O(nK)

*Giải thích:* Với n phần tử cần merge từ K run → mỗi phần tử sẽ được xử lý một lần (vào/ra heap) → **n lần thao tác heap**. Mỗi thao tác mất **O(log K)** → tổng thời gian: **O(n log K)**

## II. BÀI TẬP TỰ LUẬN (THAM KHẢO FILE BÁO CÁO)

1. **Trình bày ngắn gọn các bước của thuật toán Natural Merge Sort và Multiway Balanced Merge Sort.**

*Natural Merge Sort*

* Bước 1: Xác định các dãy con có đường chạy tăng dần (runs)
* Bước 2: Trộn lần lượt các runs cho đến khi còn một run duy nhất

*Multiway Balanced Merge Sort*

* Bước 1: Chia dữ liệu thành các k dãy con đã sắp xếp (Sorted Runs)
* Bước 2: Trộn các dãy con sử dụng cấu trúc hàng đợi ưu tiên
* Bước 3: Lặp lại quá trình trộn cho đến khi hoàn tất

1. **Giả sử bạn có 10MB bộ nhớ và một tệp dữ liệu 1GB cần sắp xếp. Hãy mô tả cách áp dụng thuật toán Multiway Balanced Sort để giải quyết bài toán này.**

*Bước 1: Chia dữ liệu thành các run ban đầu*

* RAM chỉ chứa được 10MB → ta đọc từng khối 10MB từ file, sắp xếp nó trong RAM (dùng QuickSort, HeapSort, v.v.).
* Ghi kết quả đã sắp xếp đó ra đĩa thành 1 **run**.

Tổng cộng: 1024MB / 10MB = **103 run** (lấy tròn lên)

*Bước 2: Merge các run bằng Multiway Merge Sort*

* Có 103 file đã sắp xếp (các run), cần merge lại thành 1 file cuối.
  + 1 buffer để ghi output
  + K - 1 buffer để đọc các run đầu vào
  + Với RAM = 10MB, mỗi buffer 1MB → ta có thể merge cùng lúc **K = 10 → 9 run**
* Đưa các phần tử đầu tiên của mỗi run vào cấu trúc Min Heap, chọn ra phần tử nhỏ nhất và đưa vào buffer output, tiếp tục đưa phần tử đầu tiên của run chứa phần tử nhỏ nhất

*Bước 3: Lặp lại quá trình trộn cho đến khi hoàn tất*

1. **So sánh Natural Merge Sort và Balanced Multiway Merge Sort**

|  |  |  |
| --- | --- | --- |
|  | **Ưu điểm** | **Nhược điểm** |
| **Natural Merge Sort** | * Dễ cài đặt * Có thể giảm số lần phân chia mảng nếu như có nhiều mảng con đã được sắp xếp | * Thời gian thực hiện lâu do việc thực hiện nhiều thao tác đọc, ghi * Không hiệu quả với dữ liệu ngẫu nhiên |
| **Balanced Multiway Merge Sort** | * Hiệu quả với dữ liệu lớn * Thời gian thực hiện đọc, ghi giảm nhiều | * Cài đặt phức tạp * Chọn k để không bị lỗi bộ nhớ hoặc dư thừa bộ nhớ |

1. **Viết giả mã (pseudocode) cho giai đoạn chia dữ liệu trong thuật toán Natural Merge Sort và Multiway Balanced Merge Sort.**

*Natural Merge Sort:*

**while** (not end of input):

    read next record **next\_r**

**if** (**next\_r >= r**):  **// still non-decreasing**

        append **next\_r** to **current\_run**

**else:**

        save **current\_run** as a run to disk

        start new current\_run with **next\_r**

**r = next\_r**

save **current\_run** as a run to disk

*Multiway Balanced Merge Sort.*

**while** (not end of input):

read **1** block (or as many as can **fit** in memory)

**sort** the block in memory

write sorted block to disk as a run