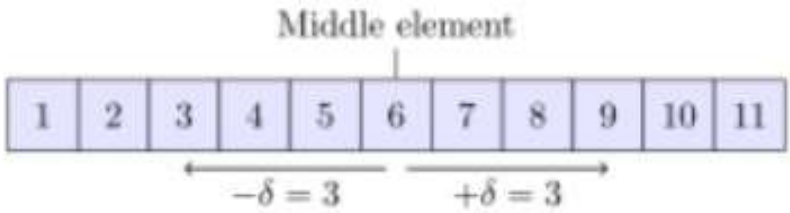
Cập nhật tháng 8 năm 2024

[Bài đọc] Thuật toán tìm kiếm và sắp xếp

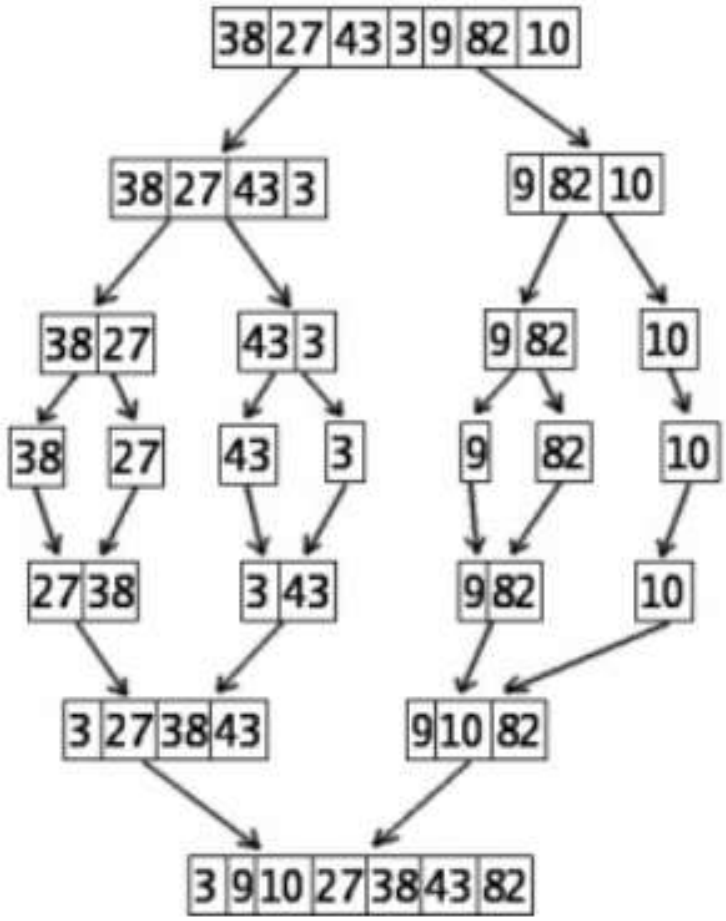
1. **Thuật toán là gì?**
- Thuật toán là **tập hợp các bước cụ thể** để giải quyết một bài toán. Trong lập trình, thuật toán giúp **xử lý dữ liệu một cách hiệu quả**, chẳng hạn như tìm kiếm, sắp xếp, tính toán,…

2. **Tìm kiếm (Searching)**



- Mục tiêu:
  - Tìm xem một **giá trị cụ thể có nằm trong mảng/danh sách** hay không
- Một số thuật toán tìm kiếm phổ biến
  - **Tìm kiếm tuyến tính (Linear Search):**
    - Duyệt qua từng phần tử từ đầu đến cuối
    - Dễ viết, nhưng chậm với mảng lớn
    - Ví dụ: Tìm số 5 trong mảng **[2, 4, 5, 8]**, kiểm tra lần lượt từng phần tử
  - **Tìm kiếm nhị phân (Binary Search):**
    - **Yêu cầu mảng đã sắp xếp**
    - So sánh với phần tử ở giữa → chia đôi vùng tìm → tiếp tục
    - Rất nhanh (độ phức tạp O(log n))
    - Ví dụ: Tìm số 7 trong **[1, 3, 5, 7, 9]** → so sánh 5 → 7 > 5 → tìm trong nửa sau

3. **Sắp xếp (Sorting)**



- Mục tiêu:
    - Tìm số 7 trong **[1, 3, 5, 7, 9]** → so sánh 5 → 7 > 5 → tìm trong nửa sau
- Một số thuật toán sắp xếp cơ bản:
    - **Bubble Sort (Sắp xếp nổi bọt):**
        - So sánh từng cặp liền kề và hoán đổi nếu sai thứ tự
        - Đơn giản nhưng chậm với dữ liệu lớn
    - **Selection Sort (Sắp xếp chọn):**
        - Mỗi vòng tìm **giá trị nhỏ nhất còn lại** và đưa về đầu.
        - Dễ hiểu, hiệu suất không cao
    - **Insertion Sort (Sắp xếp chèn):**
        - Giống như cách bạn xếp bài trên tay
        - Tốt với mảng nhỏ hoặc gần như đã sắp xếp
    - **Merge Sort / Quick Sort (Sắp xếp nâng cao):**
        - Dùng chia để trị, tốc độ nhanh (O(n log n))
        - Thường dùng trong thực tế và thư viện Java (**Arrays.sort()**)

4. **Lý do cần học thuật toán này**

- Giúp chương trình **chạy nhanh và hiệu quả hơn**
- **Cơ sở để học cấu trúc dữ liệu nâng cao**
- Áp dụng nhiều trong **thi tuyển, phỏng vấn, lập trình thực tế**