

Ví dụ bài toán 1 trong các đề kiểm tra về minh họa thuật toán thì trình bày đủ các bước như này (chú ý là cho dù dãy đã được chuyển đúng theo thứ tự tăng dần rồi nhưng thuật toán vẫn chạy đủ các bước)

Trong ảnh trên là bài toán về sắp xếp lựa chọn.

Mã giả:



**2. Sắp xếp nổi bọt**

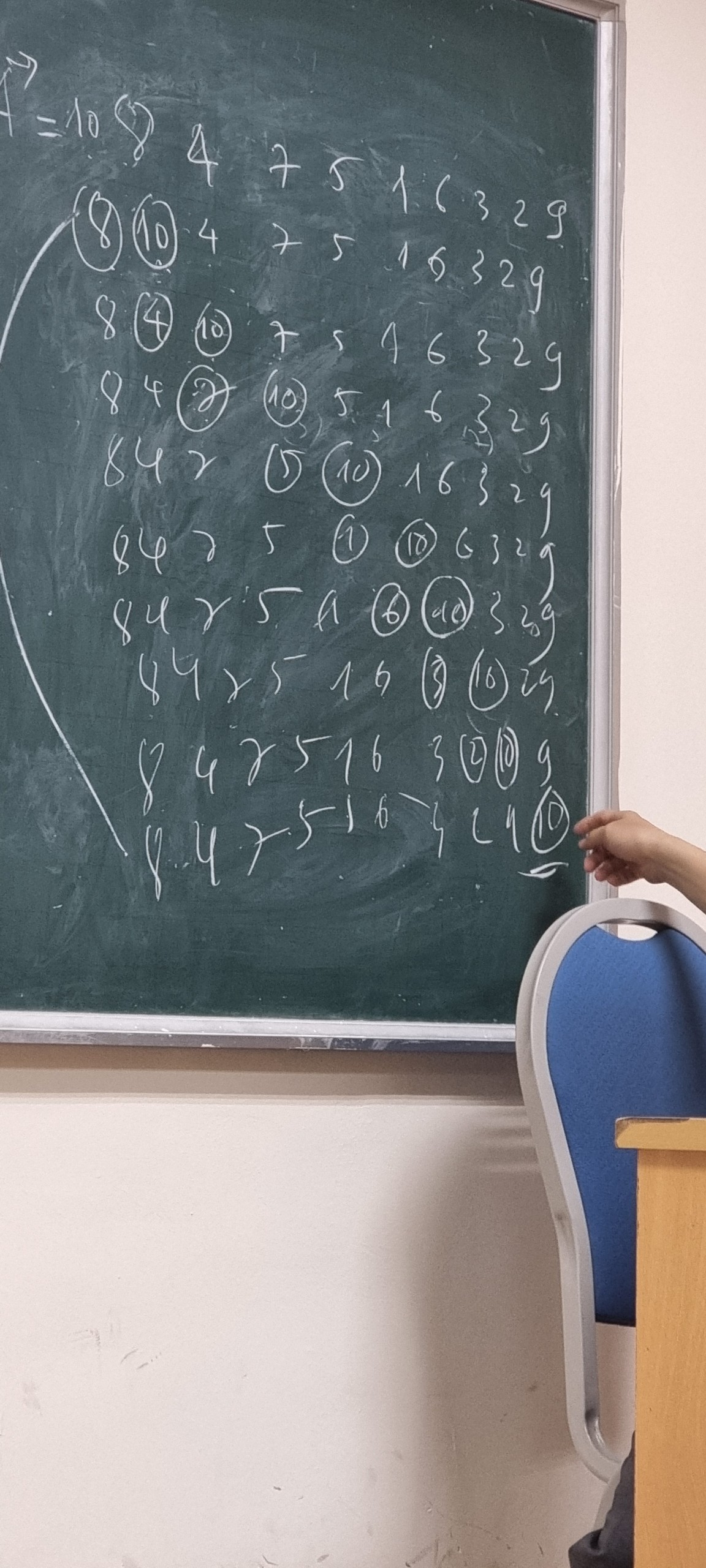
Chú ý, sau mỗi bước xét của vòng lặp, phần tử đang xét sẽ hoán vị 2 phần tử liên tiếp nhau sao cho sau khi hết mỗi vòng lặp thì phần tử lớn nhất sẽ ở cuối

Với ảnh minh họa ở dưới thì nó chỉ là 1 bước lặp

Chú ý: phải thực hiện đủ các bước lặp ngay cả khi nó đã thỏa mãn dãy tăng dần rồi.

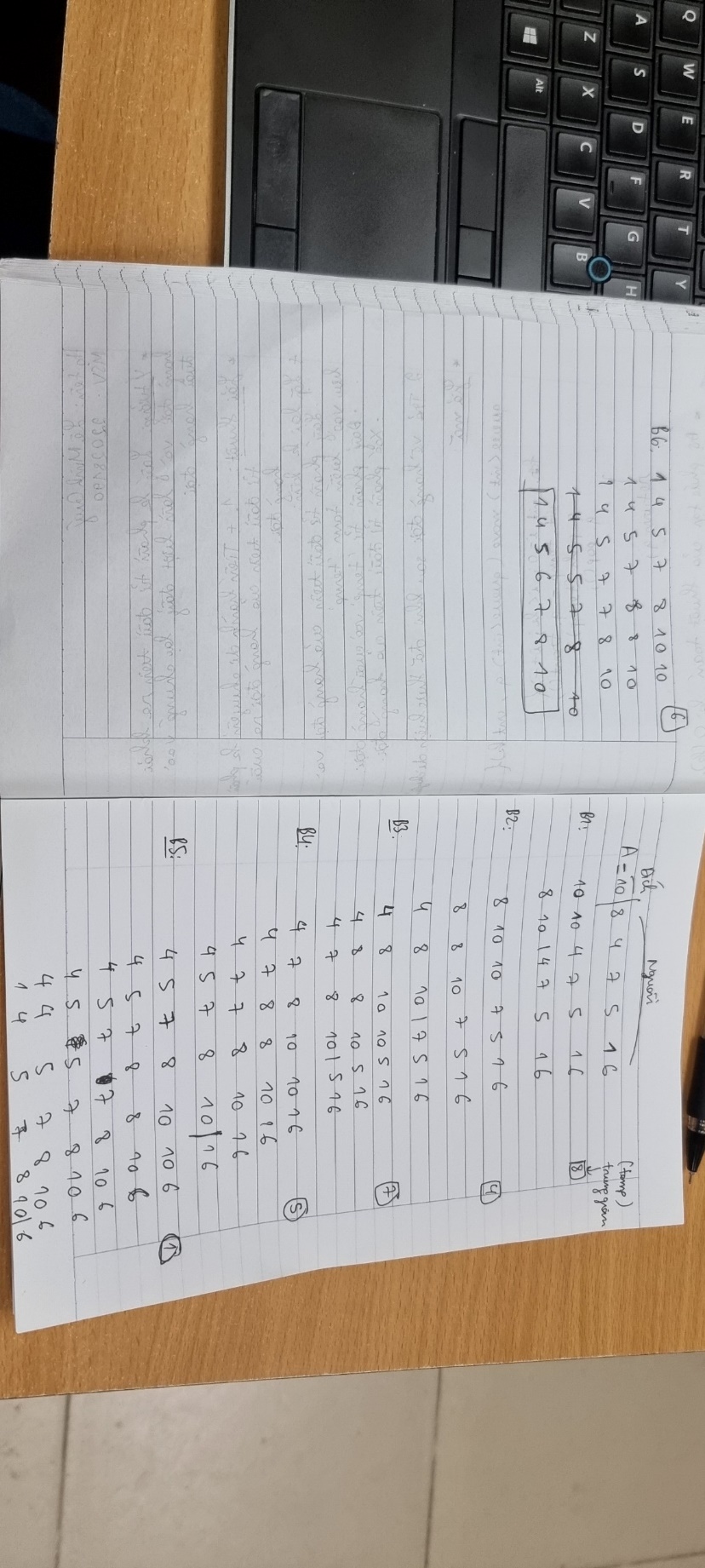
Mã giả:



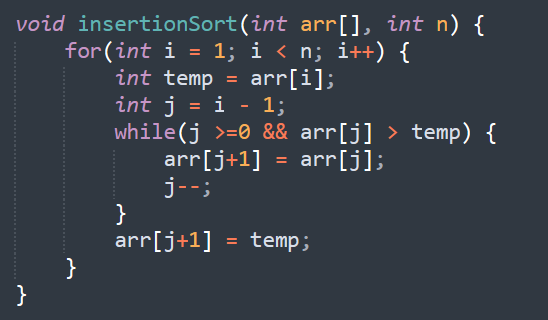
****

**3. Sắp xếp chèn**

Chú ý sau mỗi bước sẽ chọn 1 phần tử đầu tiên ở phần nguồn làm trung gian rồi so sánh nó với các phần tử đích ở đằng trước để dịch chuyển chèn vào vị trí phù hợp.



Mã giả:

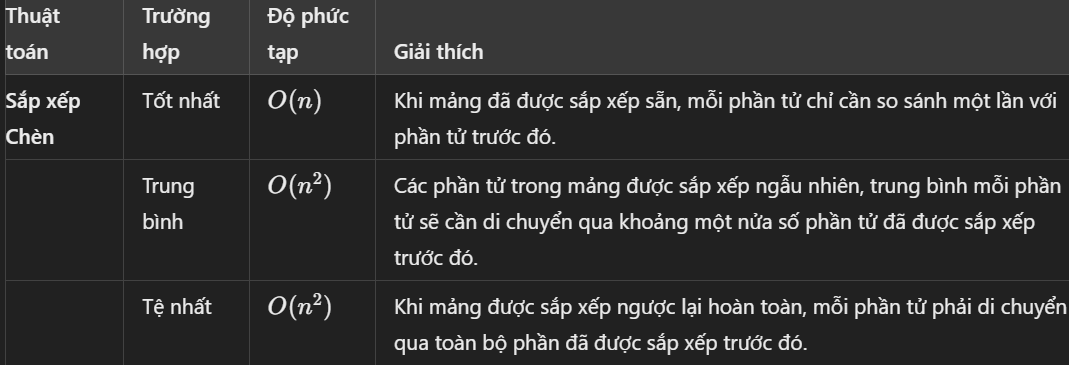


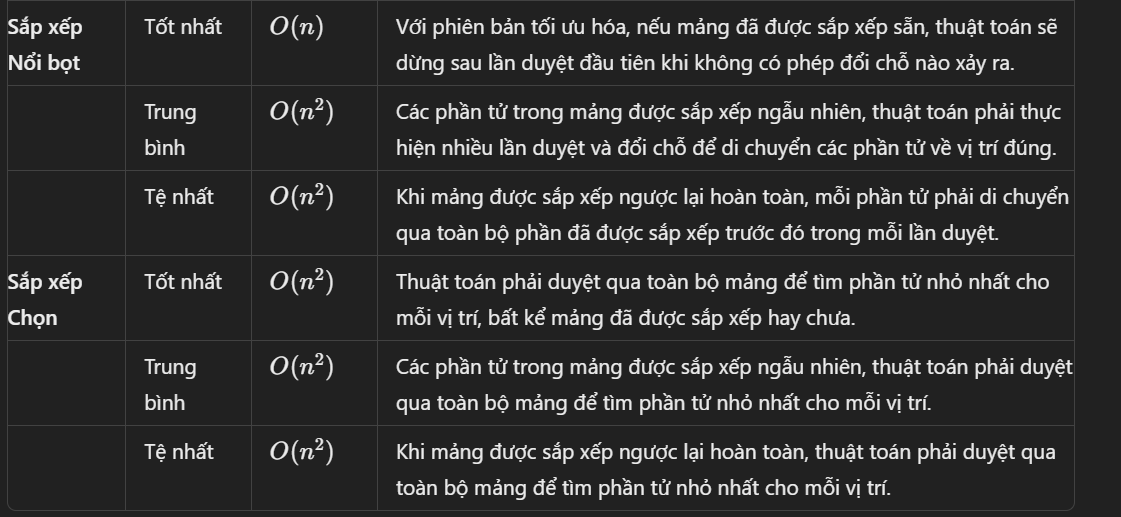
**Chú ý:**

* **Độ phức tạp của các thuật toán sắp xếp đơn giản là O(n^2). Đây là độ phức tạp trung bình**

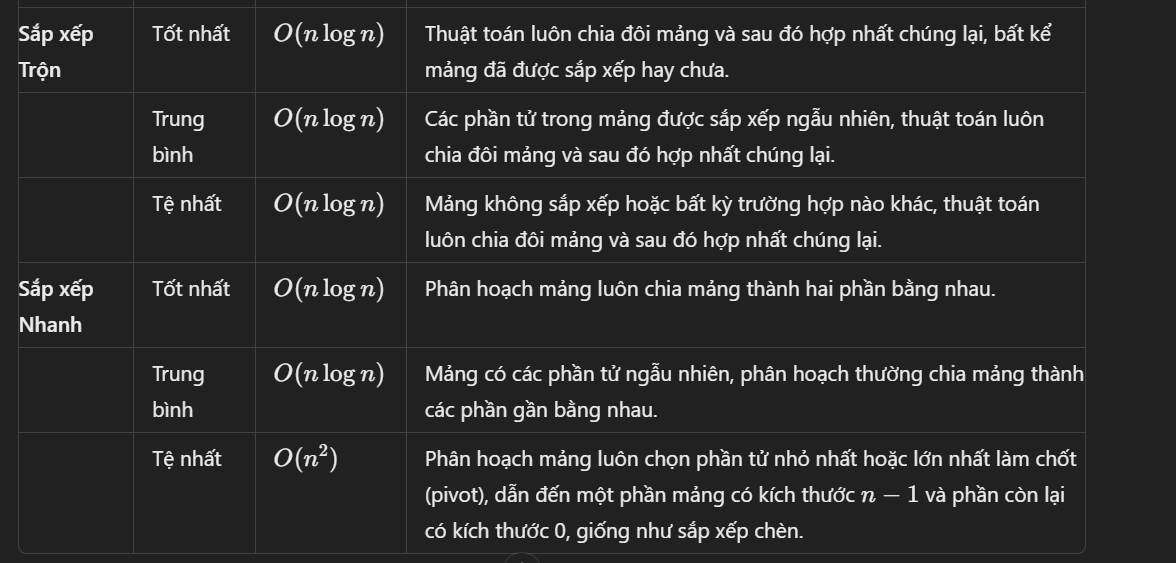
**(Thầy bảo tìm hiểu thêm về độ phức tạp tối ưu nhất hoặc tệ nhất)**

* **Chú ý khi trình bày sắp xếp thì khoanh tròn các phần tử đang đổi chỗ cho nhau để người chấm bài dễ nhận biết**

****

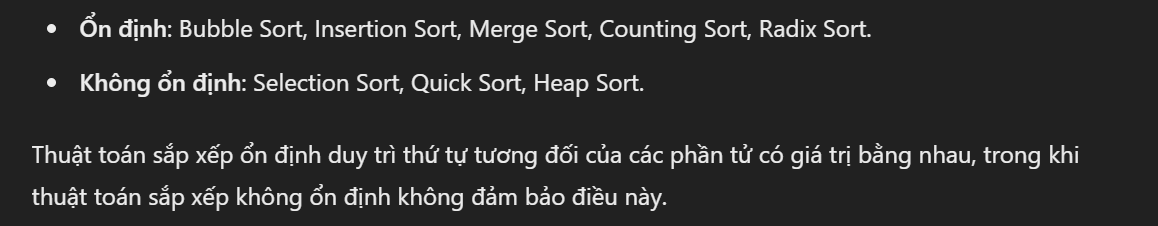
****

**Đối với QuickSort và MergeSort**

****

**QuickSort sẽ có độ phức tạp tệ nhất là O(n^2) khi mà pivot được chọn là phần tử max hoặc min của dãy.**

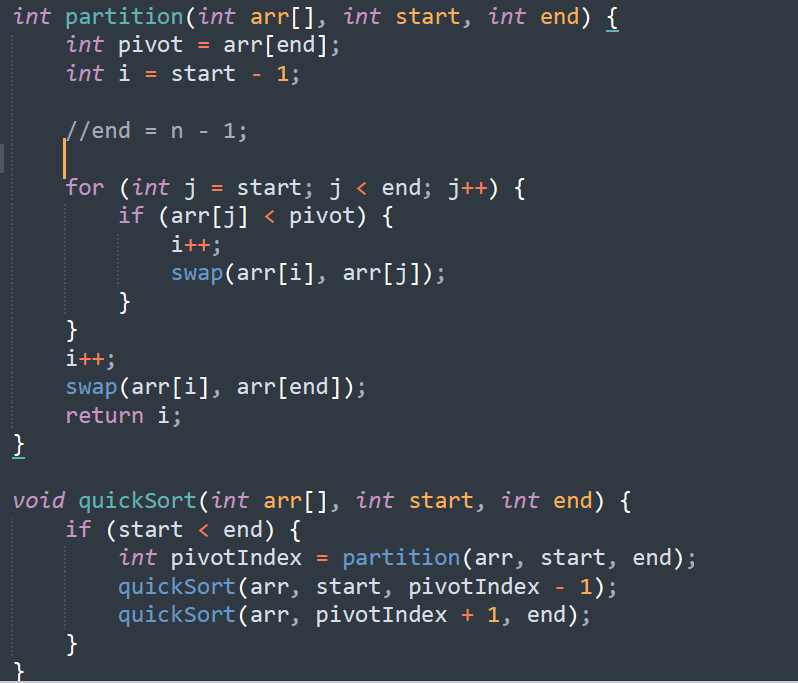
**Tính ổn định và không ổn định của thuật toán sắp xếp**

****

**4. QuickSort**

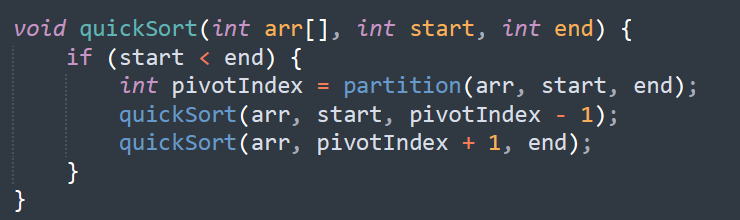
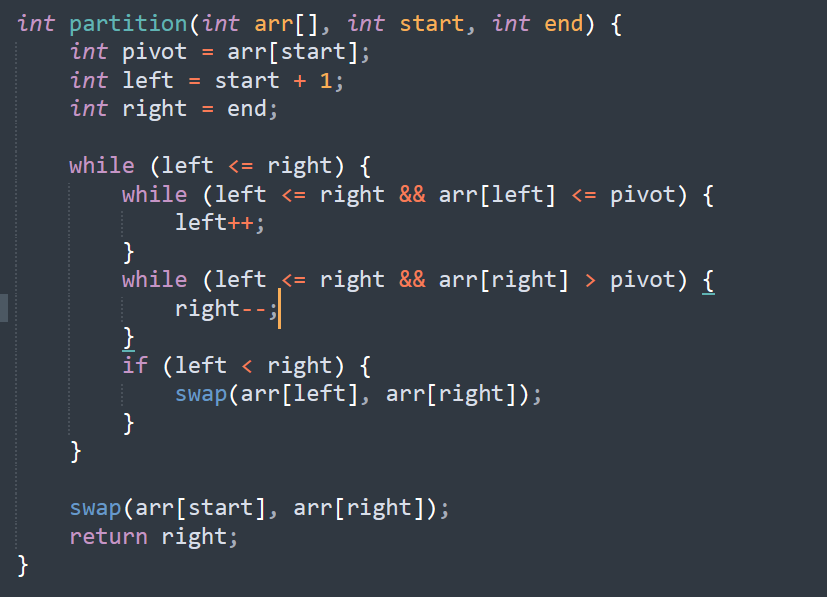
**Mã giả:**

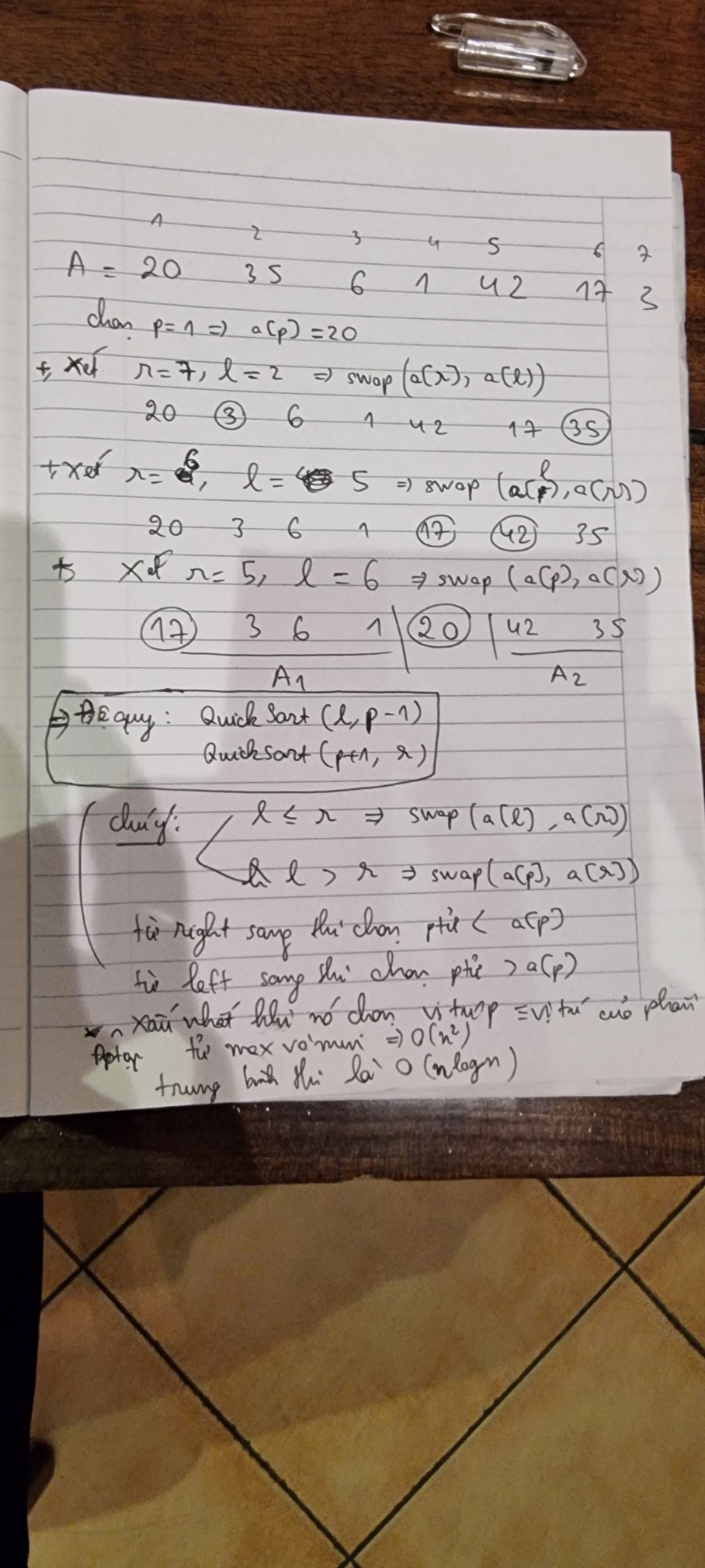
**Cách 1:**

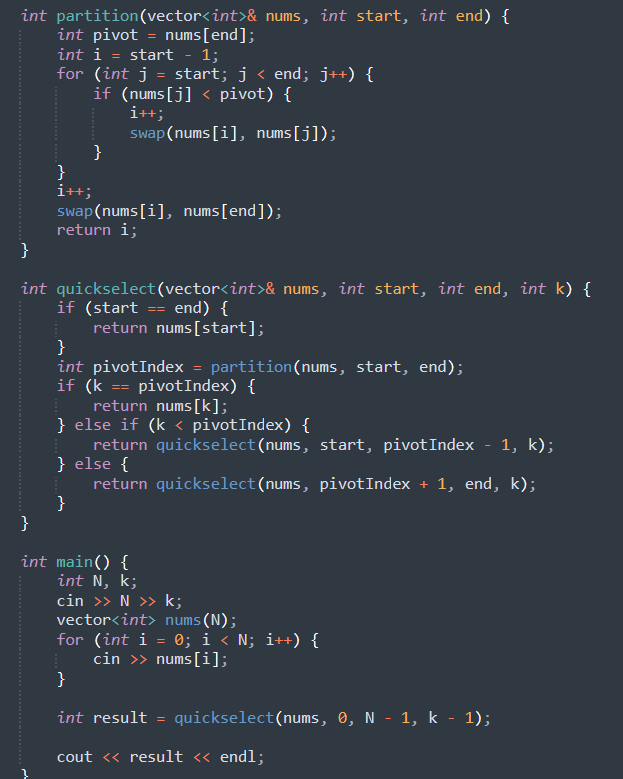
****

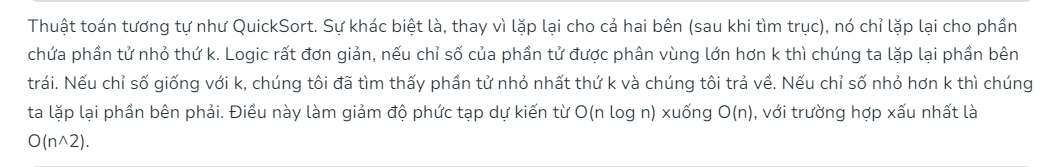
**Cách 2:**

**Mã giả:**

****

****

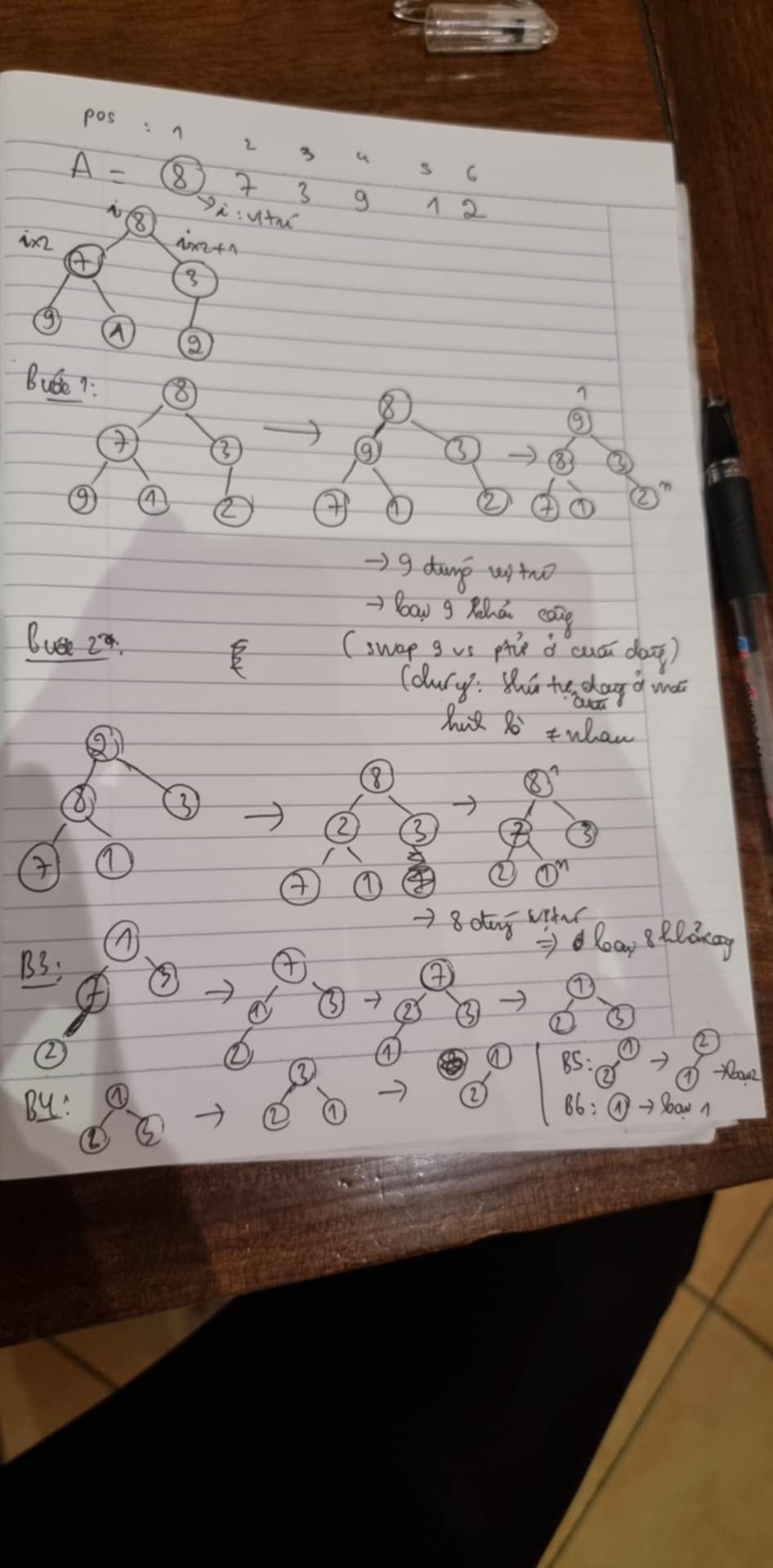
**QuickSelect: dùng cho bài toán tìm phần tử nhỏ thứ k trong dãy số chưa được sắp xếp**

****

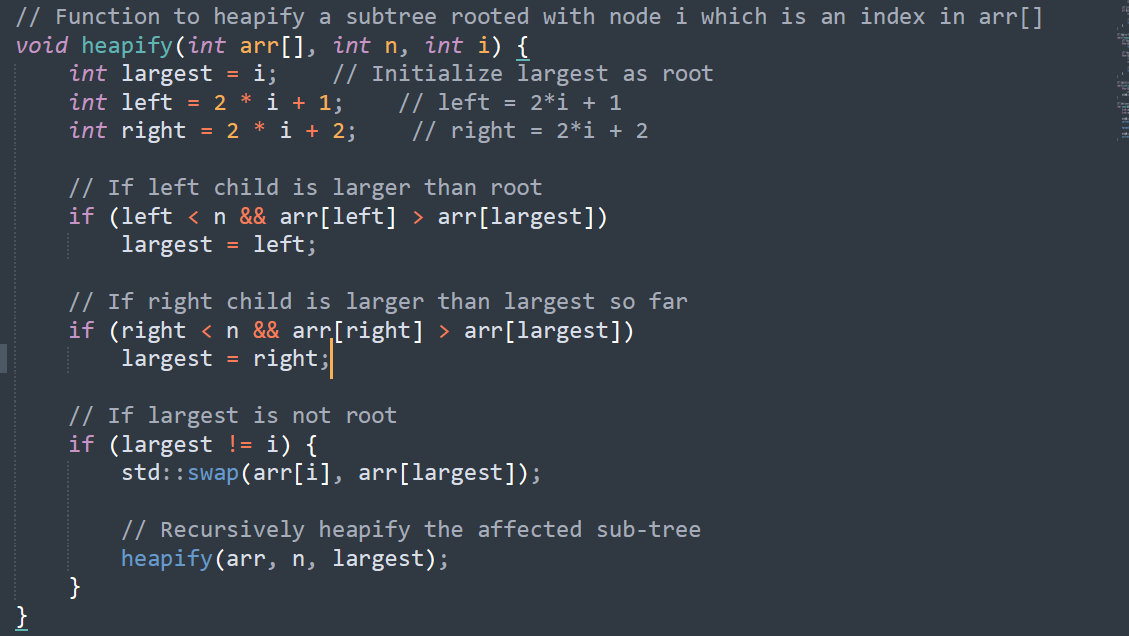
**5. HeapSort**

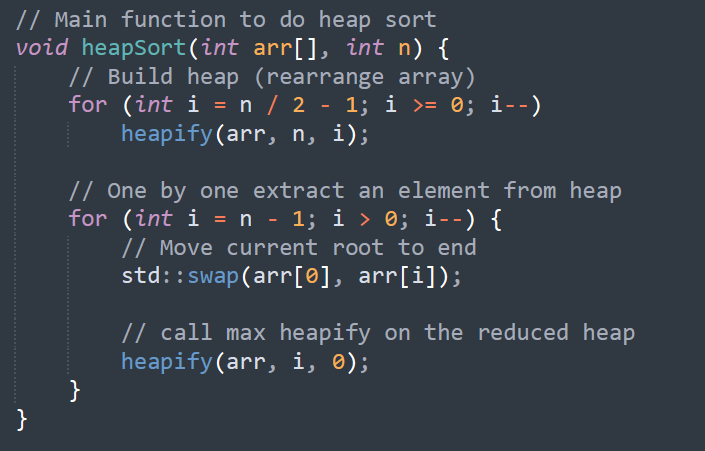
**-** Sử dụng cây thứ tự bộ phận

- Sử dụng maxheap để sắp xếp tăng dần



**Mã giả:**

****

****

Chú ý: khi bắt đầu thì chọn node ở vị trí (size /2) – 1 làm nút parent bắt đầu đầu tiên. Sau đó truy cập dần lên các parent trên bằng vị trí node parent đầu tiên – 1

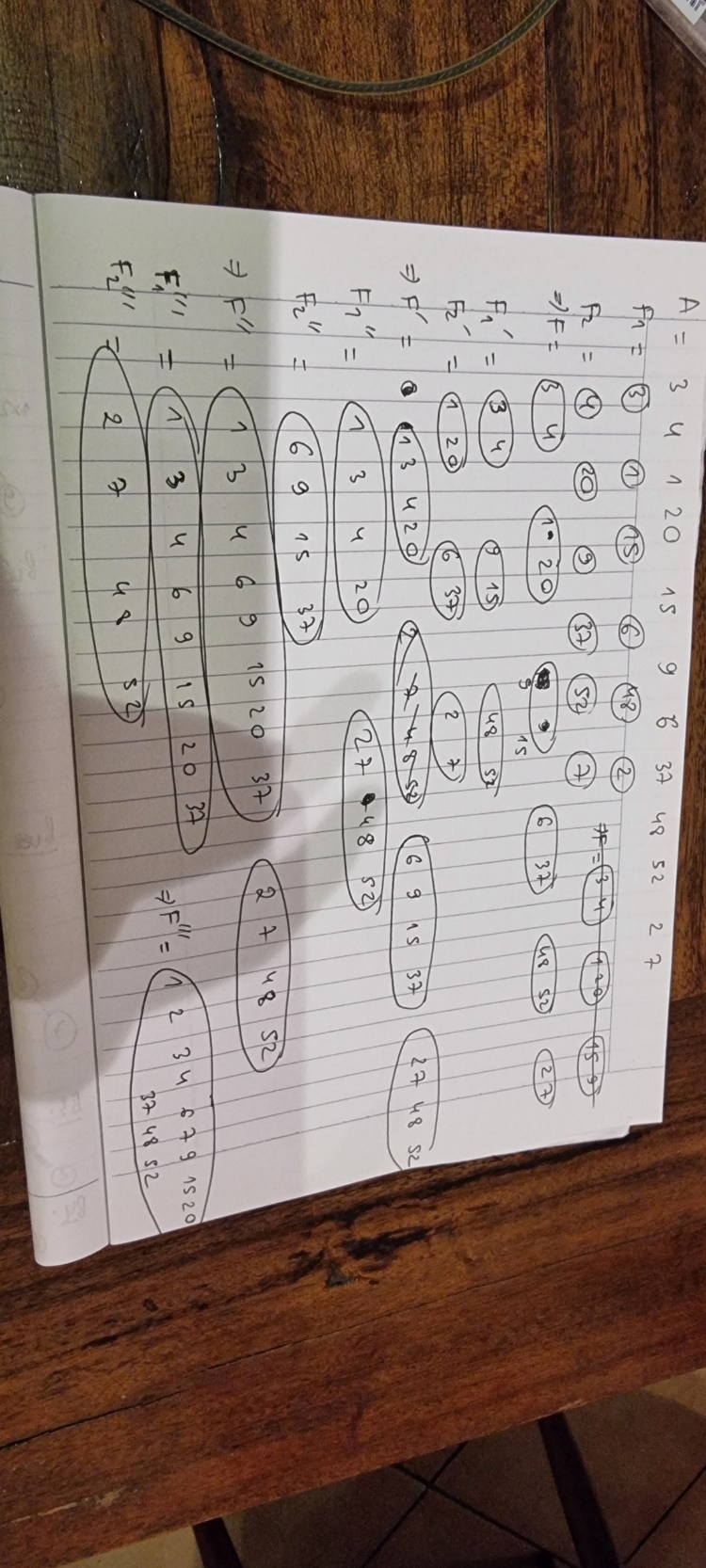
Khi nút đầu tiên đạt được max heap rồi thì swap nó với phần tử cuối cùng.

**6. MergeSort**

**- Tách thành các dãy con và sắp xếp**

**- Có 2 kiểu là sắp xếp nhị phân và sắp xếp tự nhiên:**

**+ Sắp xếp nhị phân là tách ra 2 mảng F1, F2 với các phần tử F1 là ở vị trí lẻ; Các phần tử F2 ở vị trí chẵn**

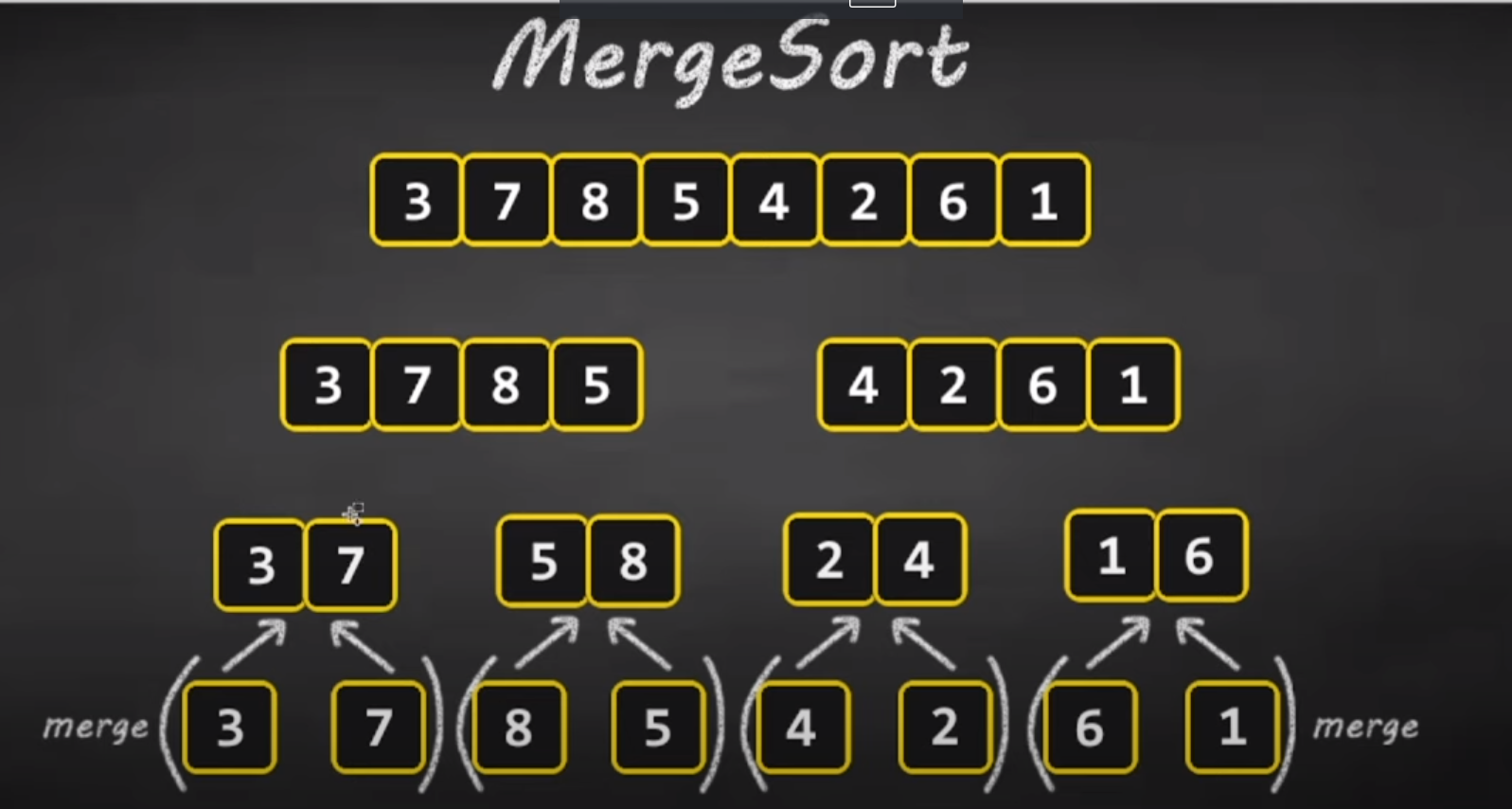
****

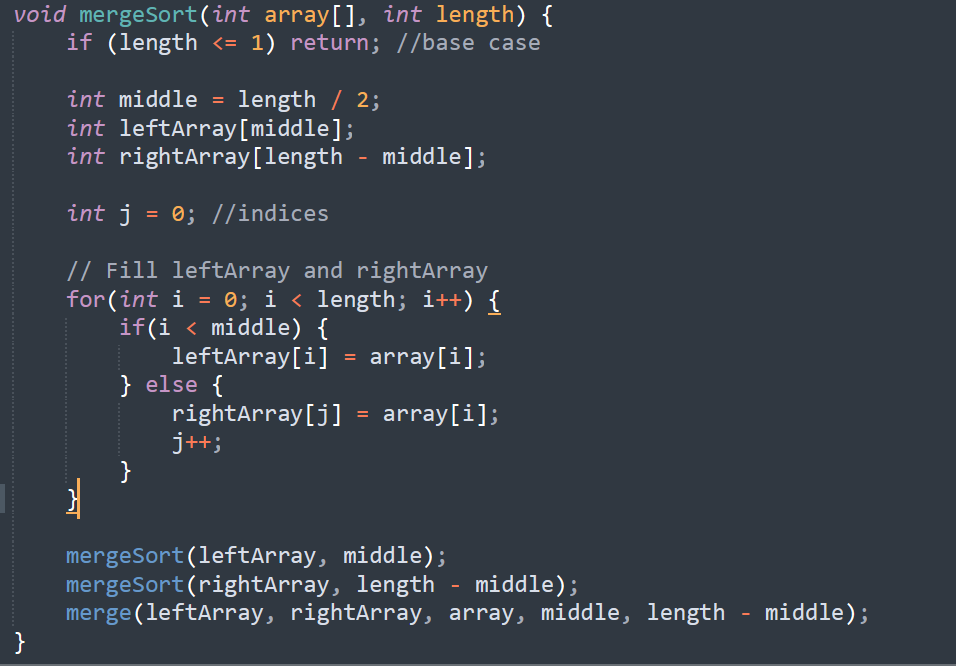
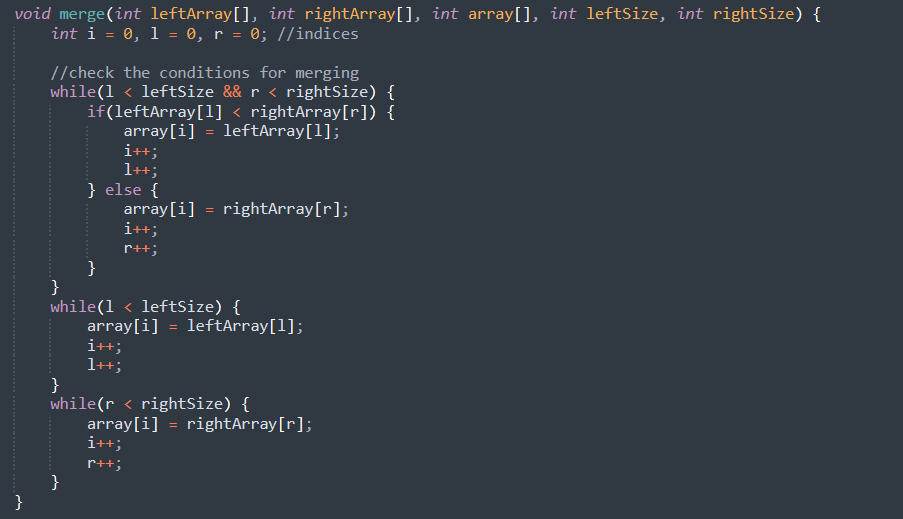
**+ Sắp xếp tự nhiên là tận dụng tối đa nếu trong mảng có các dãy con đã được sắp xếp tăng dần theo đúng thứ tự thì lấy luôn dãy con đó.**

****

**Cách 2:**

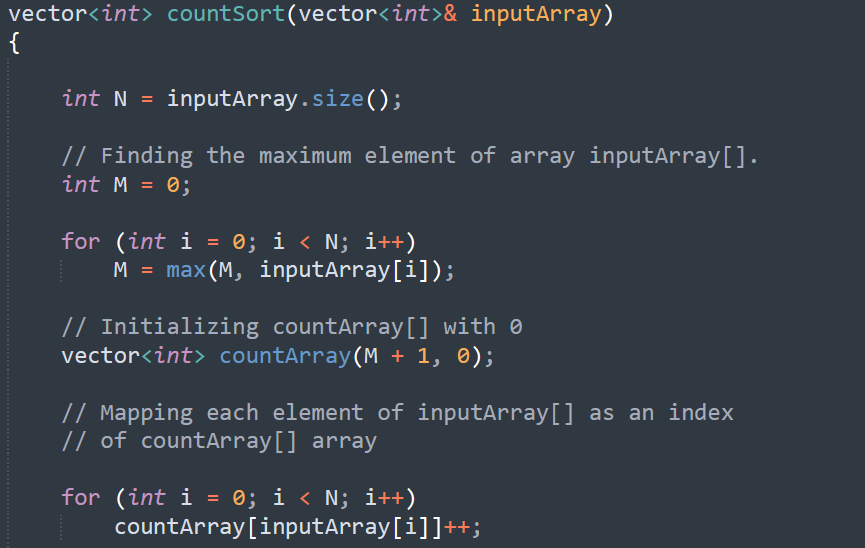
**Chia đôi mảng ban đầu thành các mảng con và cứ thế**

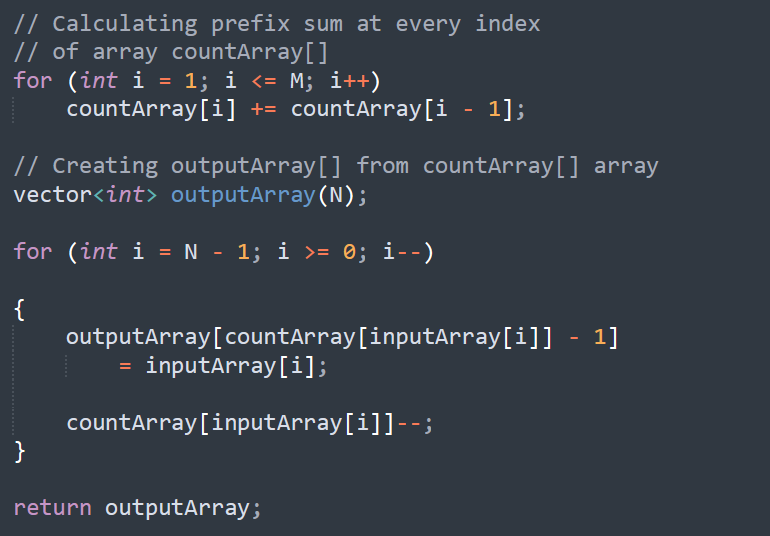
****

****

**7. CountSort**

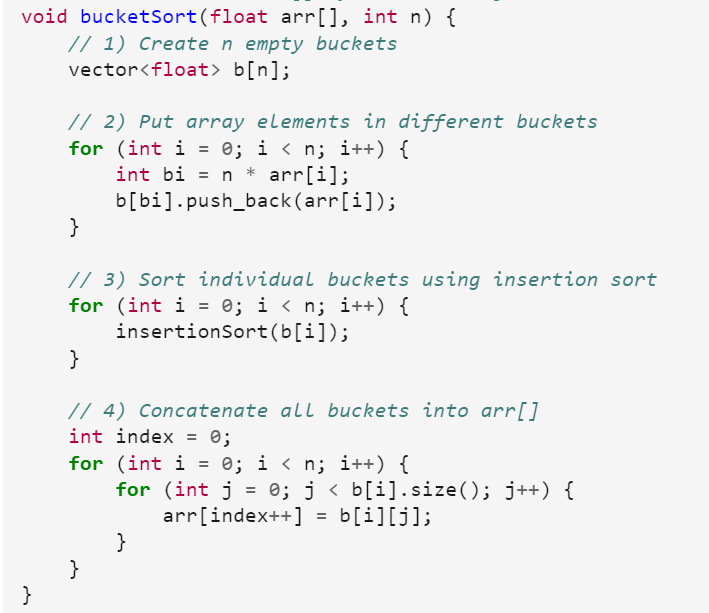
Counting Sort là thuật toán sắp xếp không dựa trên so sánh, hoạt động tốt khi phạm vi giá trị đầu vào hạn chế. Thuật toán này đặc biệt hiệu quả khi phạm vi giá trị đầu vào nhỏ so với số phần tử cần sắp xếp. Ý tưởng cơ bản đằng sau Counting Sort là đếm tần suất của từng phần tử riêng biệt trong mảng đầu vào và sử dụng thông tin đó để đặt các phần tử vào đúng vị trí đã sắp xếp của chúng.

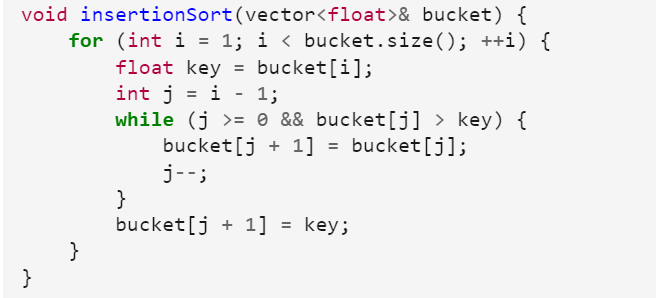
****

****

**8. Bucket Sort**

***Bucket Sort****là một kỹ thuật sắp xếp bao gồm việc chia các phần tử thành các nhóm hoặc nhóm khác nhau. Những nhóm này được hình thành bằng cách phân phối đồng đều các phần tử. Khi các phần tử được chia thành các nhóm, chúng có thể được sắp xếp bằng bất kỳ thuật toán sắp xếp nào khác. Cuối cùng, các phần tử đã được sắp xếp sẽ được tập hợp lại với nhau theo kiểu có trật tự.*

****

****

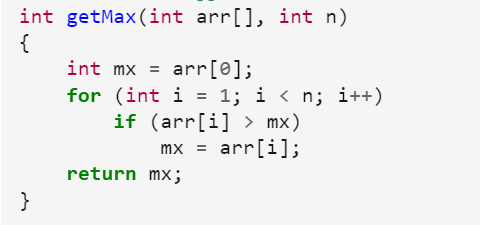
**Phân phối hẹp của dữ liệu**

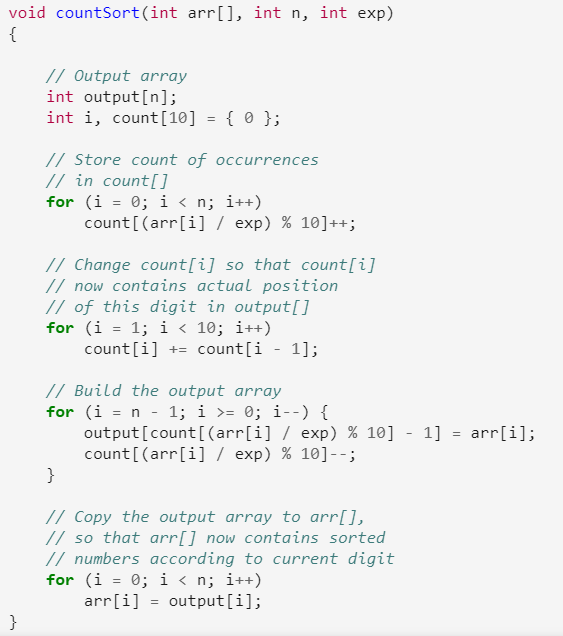
* **Đồng nhất:** Nếu các phần tử có giá trị nằm trong một khoảng hẹp, việc sử dụng Bucket Sort sẽ trở nên hiệu quả vì mỗi bucket sẽ chứa ít phần tử hơn, do đó việc sắp xếp trong mỗi bucket sẽ nhanh hơn.

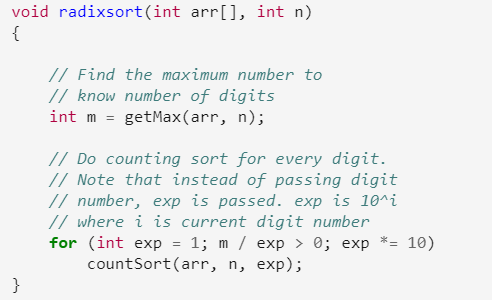
**Dữ liệu liên tục trong khoảng (0, 1)**

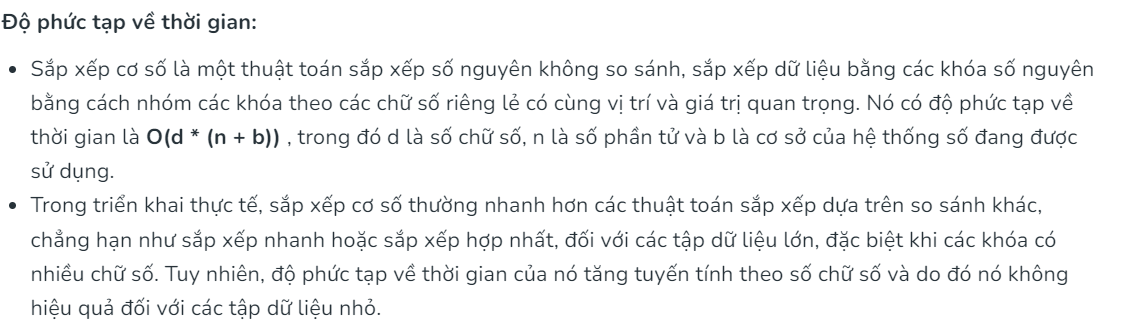
* **Số thực:** Bucket Sort rất phù hợp để sắp xếp các số thực nằm trong khoảng [0, 1). Các phần tử có thể được phân phối dễ dàng vào các bucket dựa trên giá trị của chúng. Tuy nhiên, với dữ liệu nằm ngoài khoảng này, cần chuẩn hóa trước khi phân phối vào các bucket.

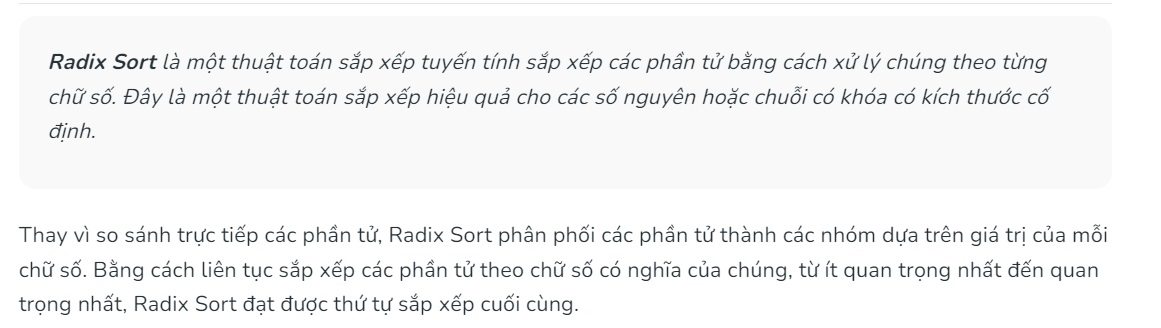
**9. Radix sort**

****

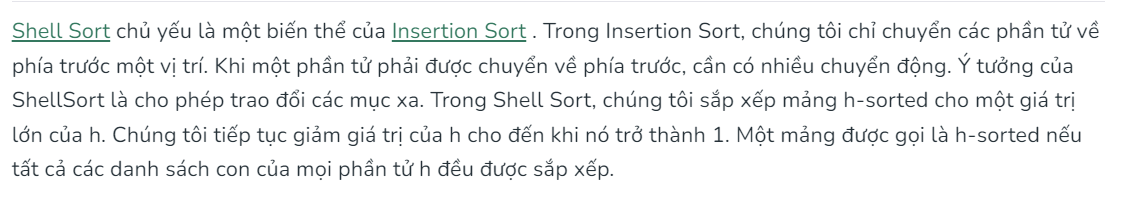
****

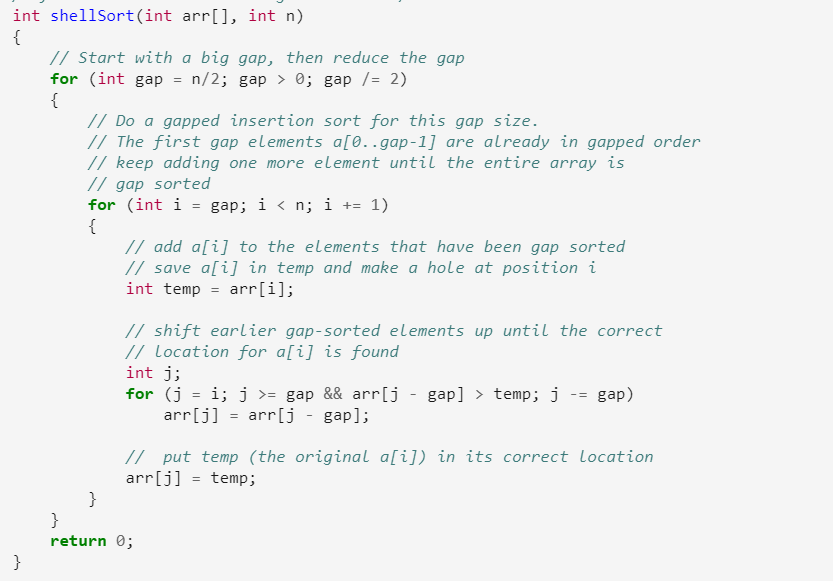
****

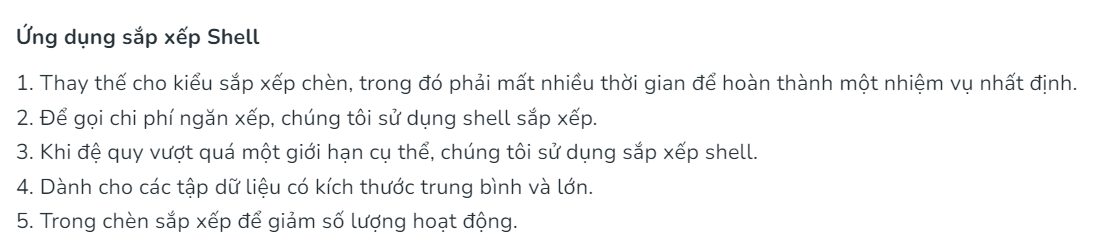
****

****

**10. Shell Sort**

****

****

****