**REPORT BIG ASSIGNMENT**

**Chương 1**

**Họ tên:** Nguyễn Anh Tuấn

**MSSV:** 20200400

**Requirement**:

Viết chương trình mô phỏng trực quan các thuật toán sắp xếp. Yêu cầu xuất ra được độ phức tạp, thời gian chạy của mỗi thuật toán

Các thuật toán gồm:

1. Quick\_sort

2. Merge\_sort

3. Natural\_merge\_sort

4. Heap\_sort

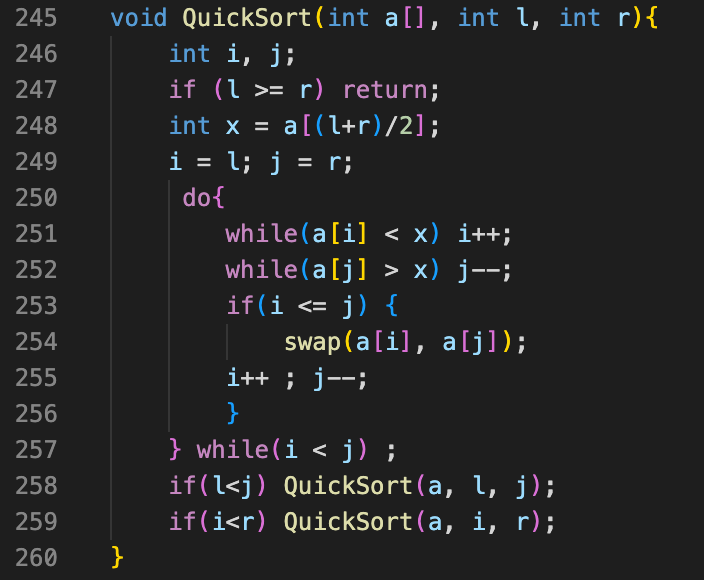
5. Radix\_sort

**Bài làm:**

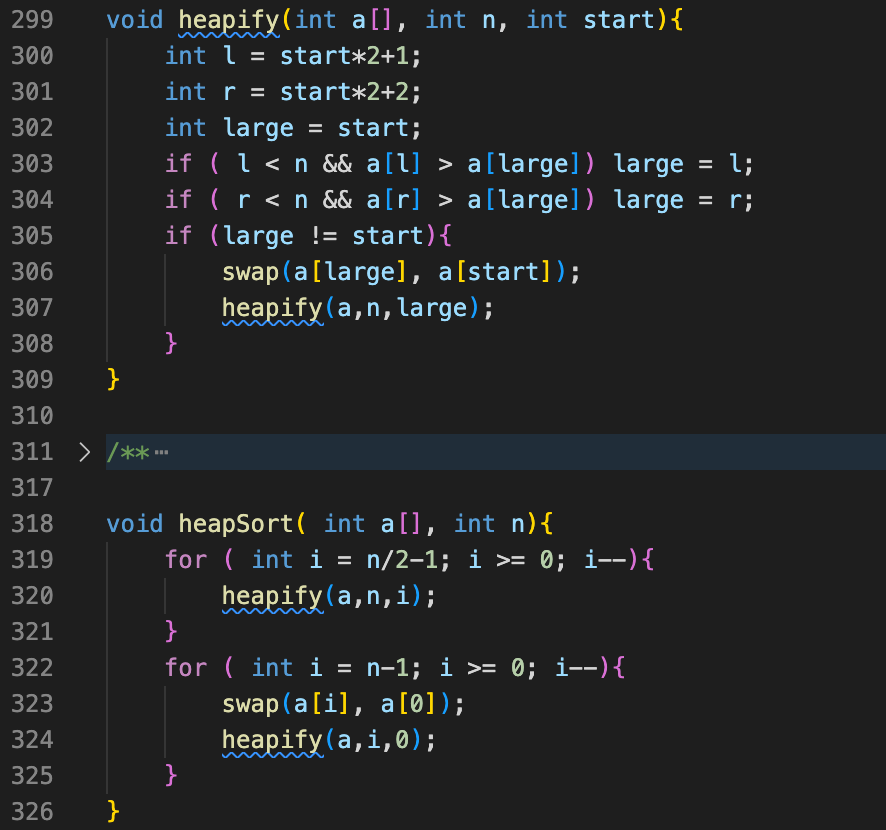
**Code:**

Các thuật toán sắp xếp theo yêu cầu:

* QuickSort

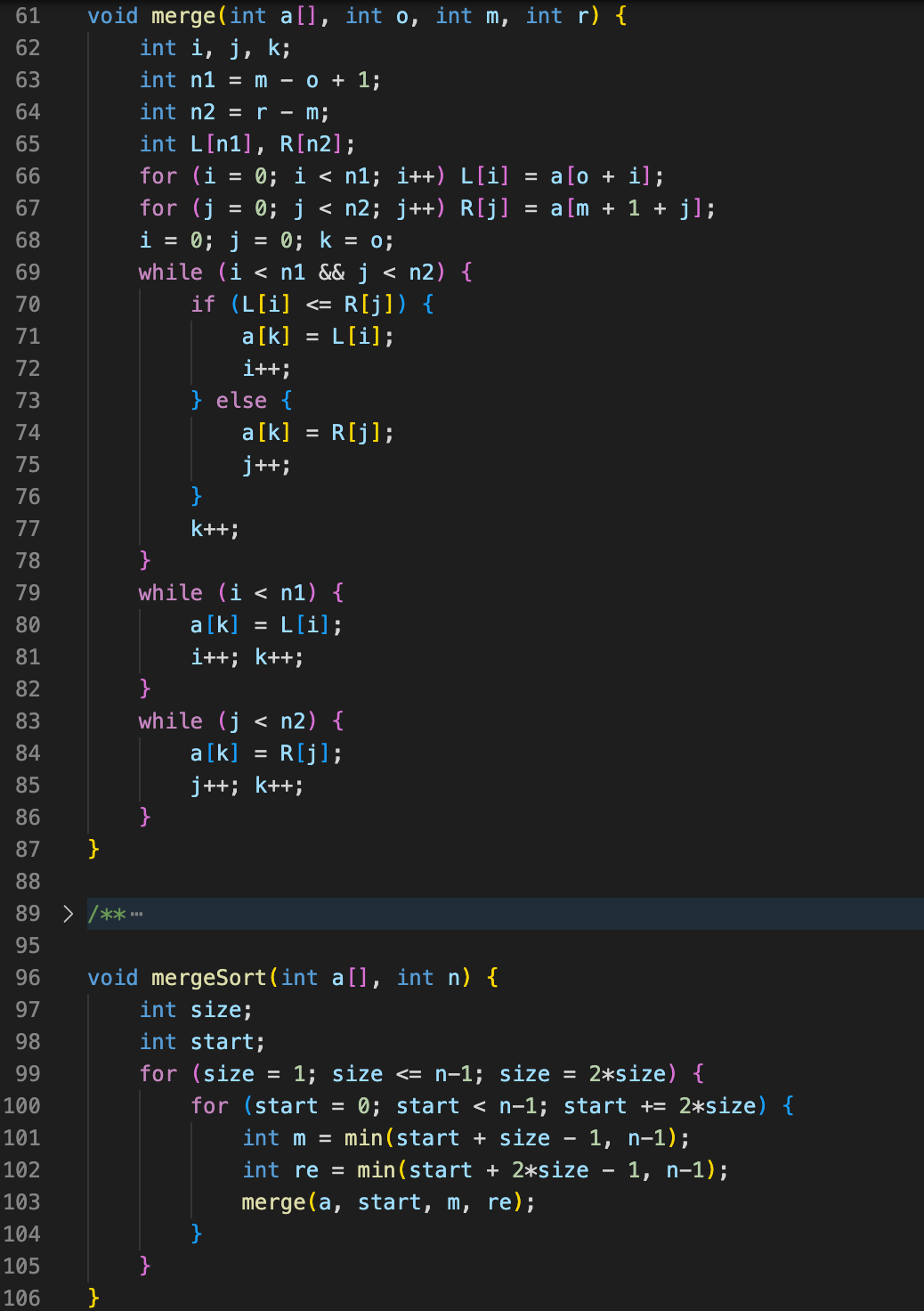


* HeapSort

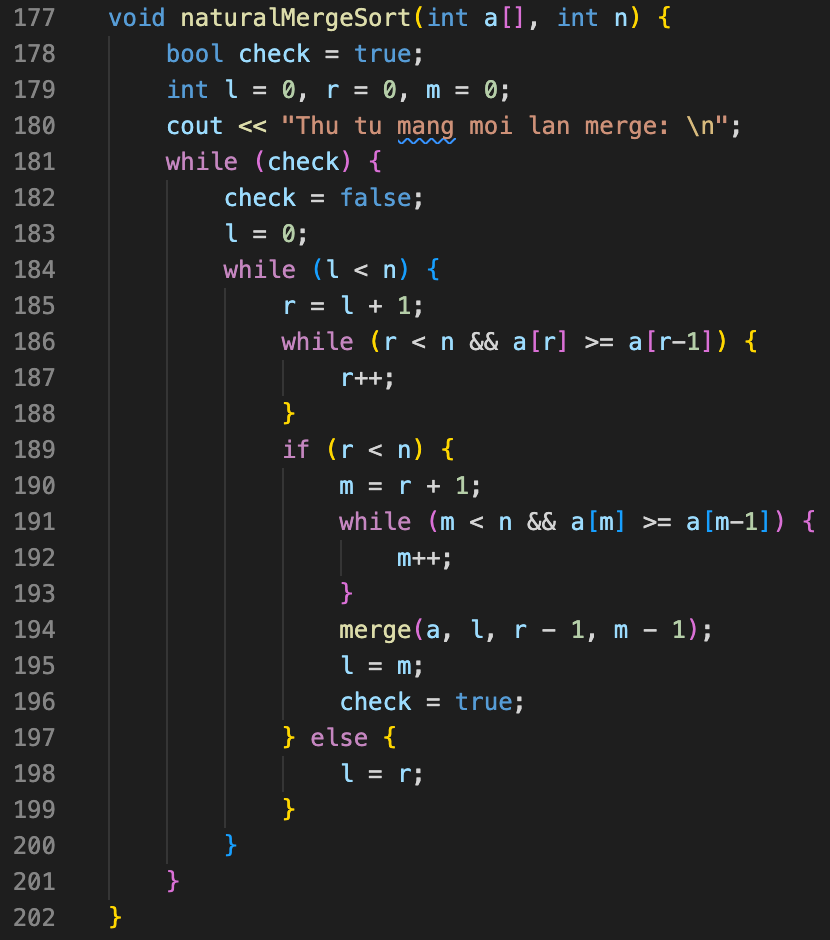


­

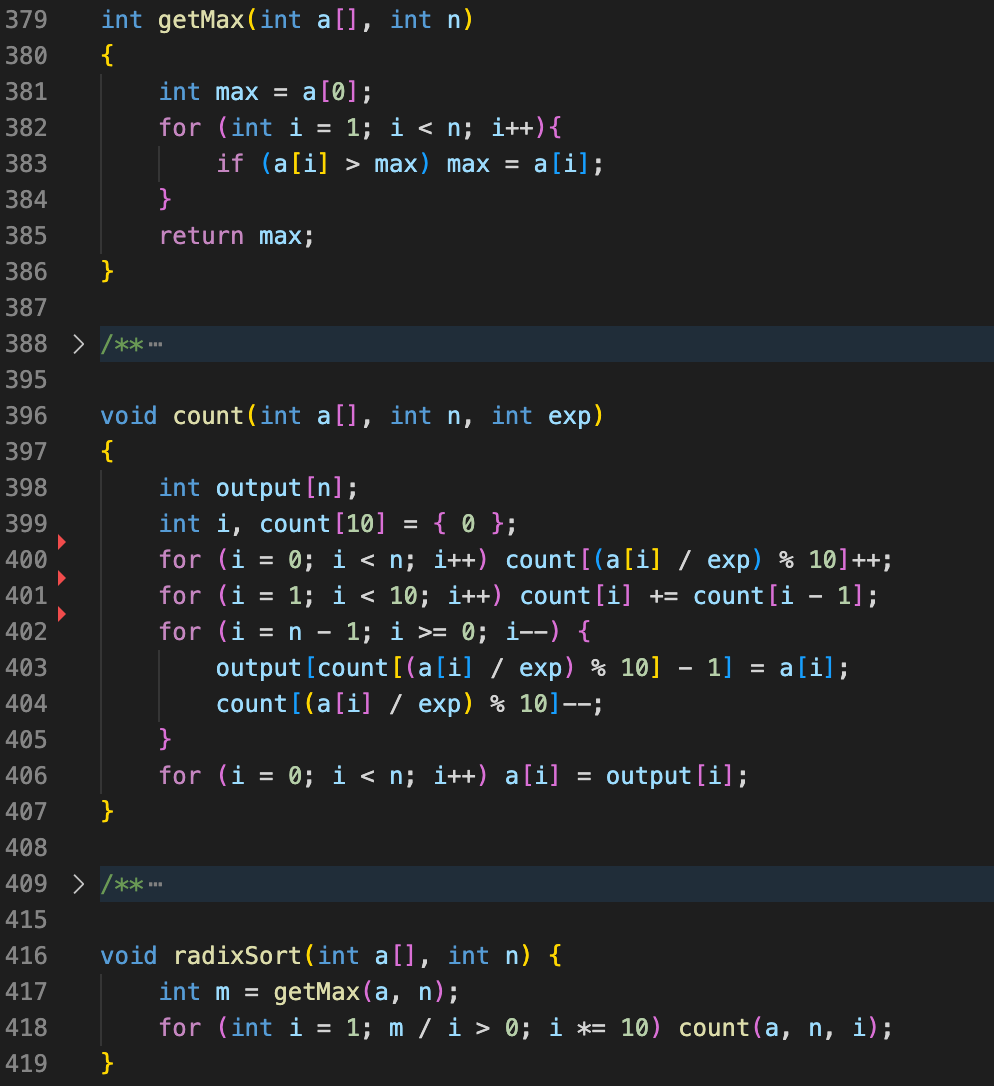
* MergeSort



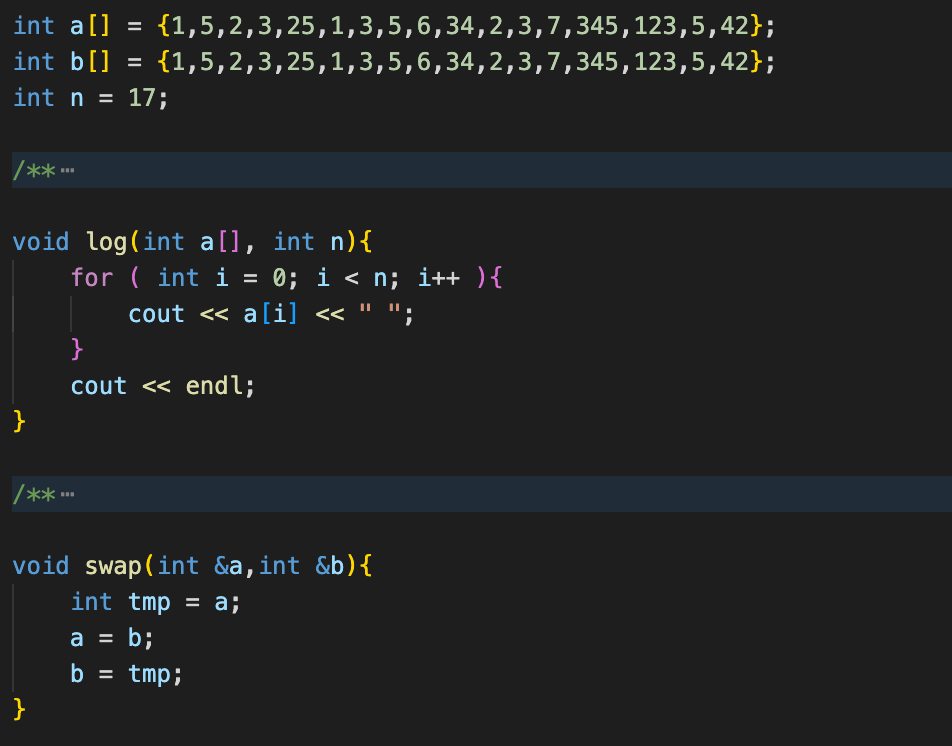
* Natural Merge Sort



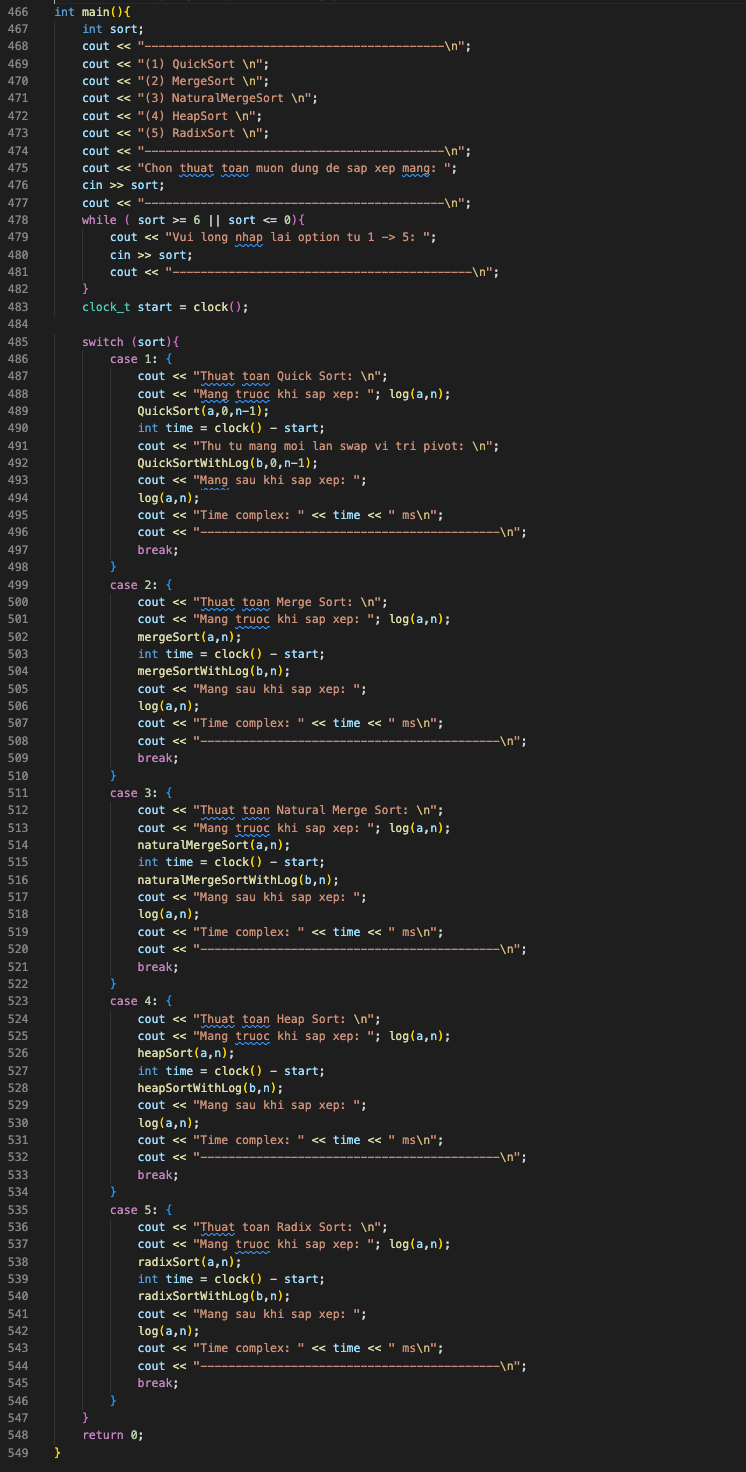
* Radix Sort

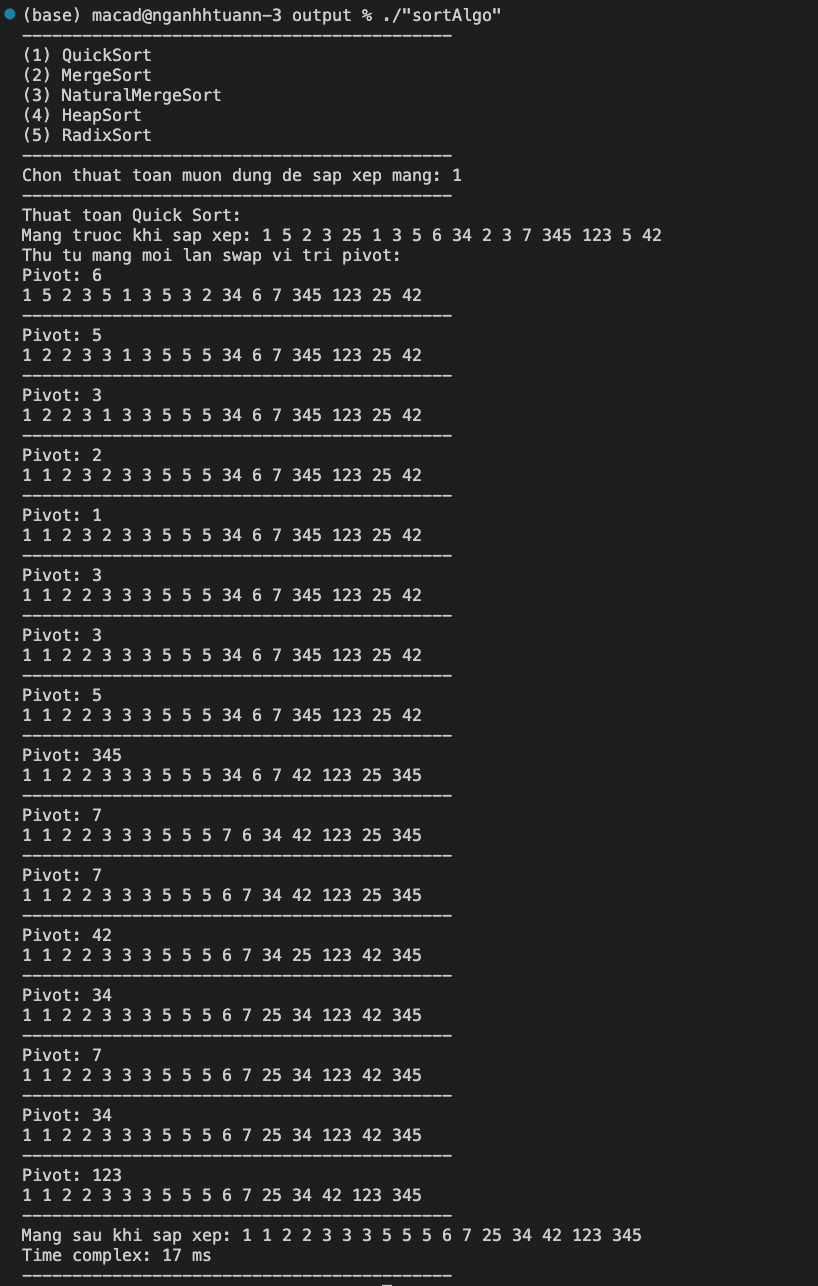
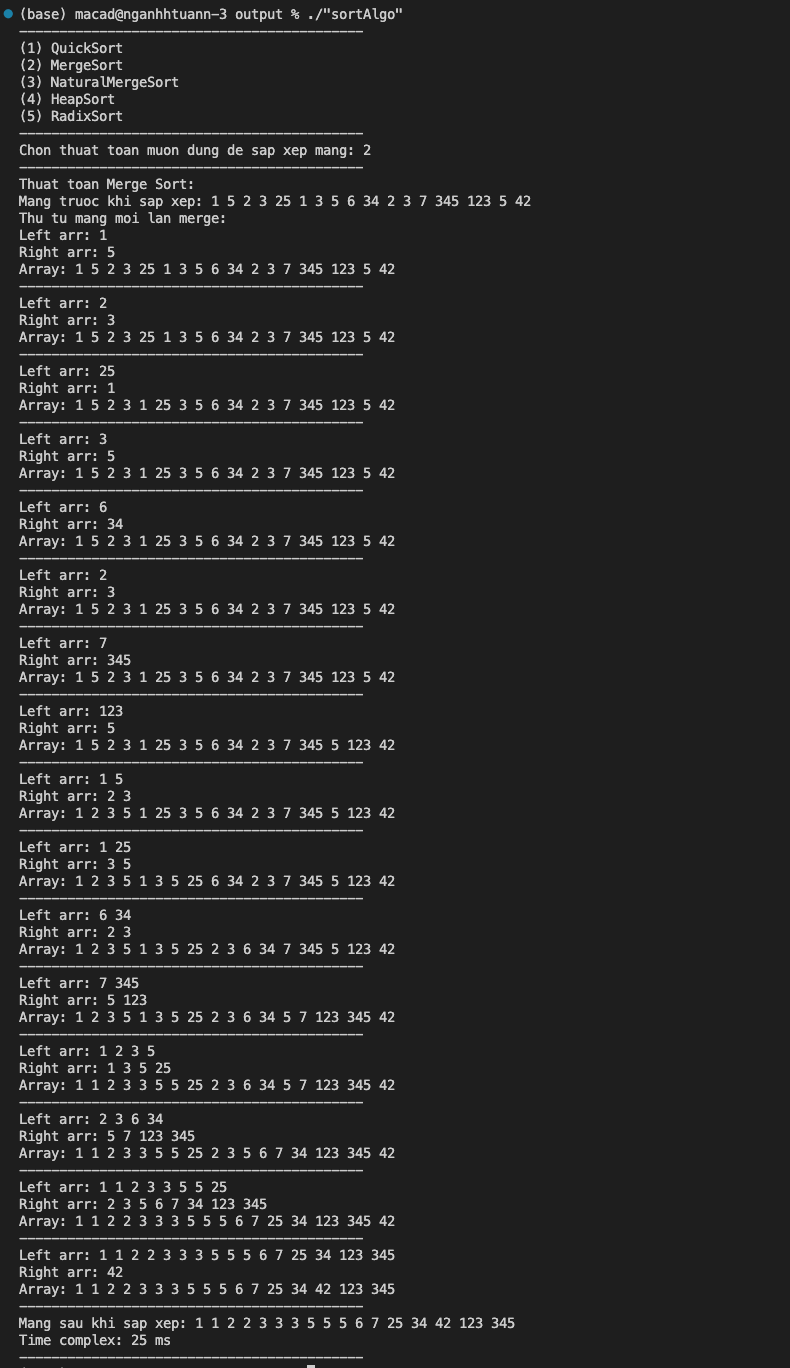
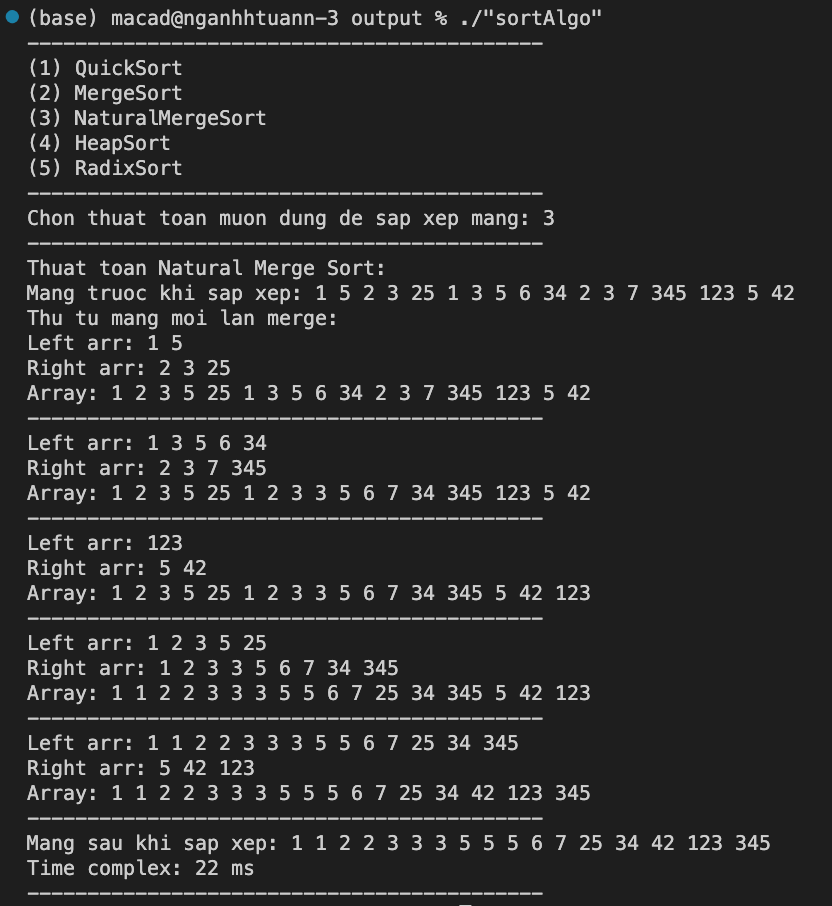
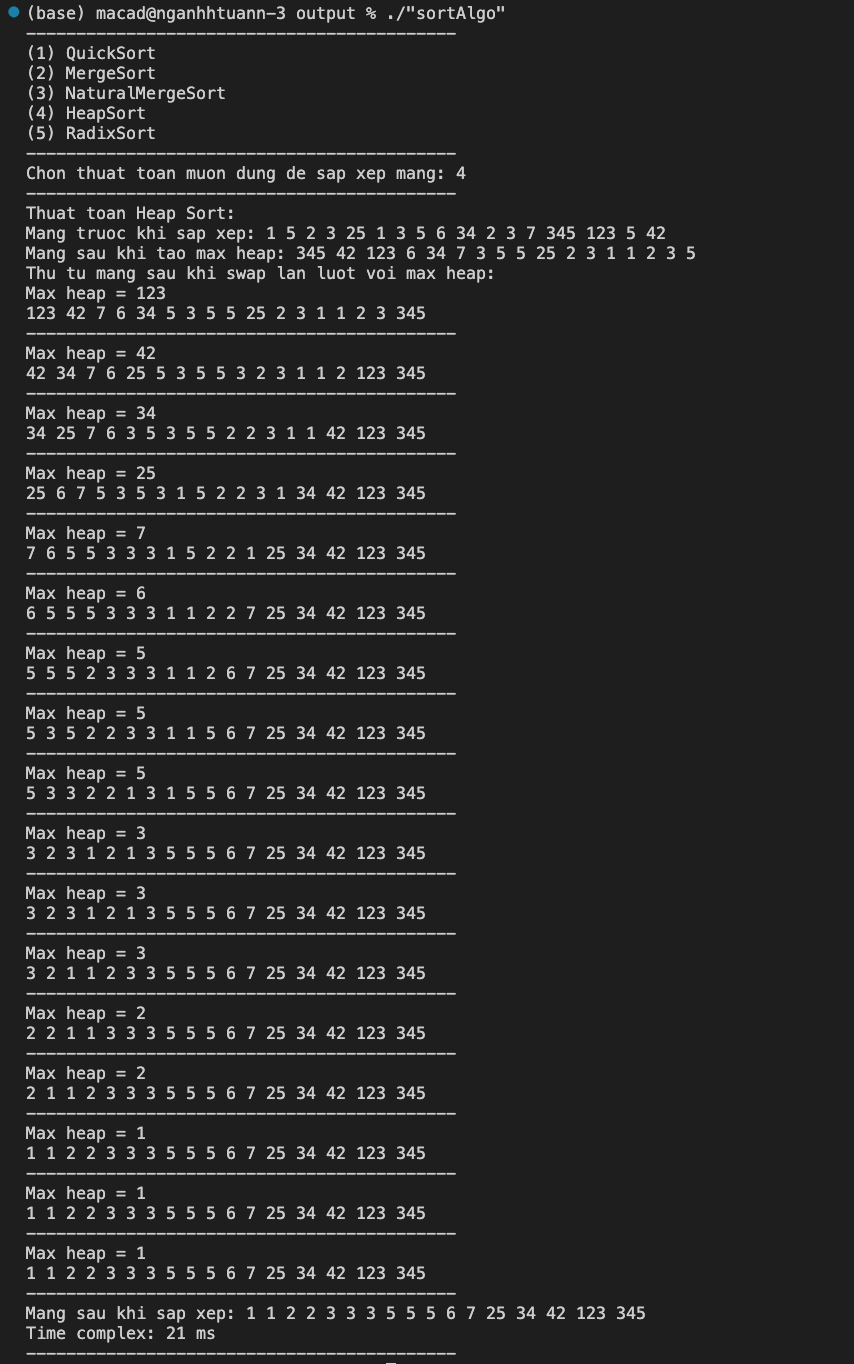
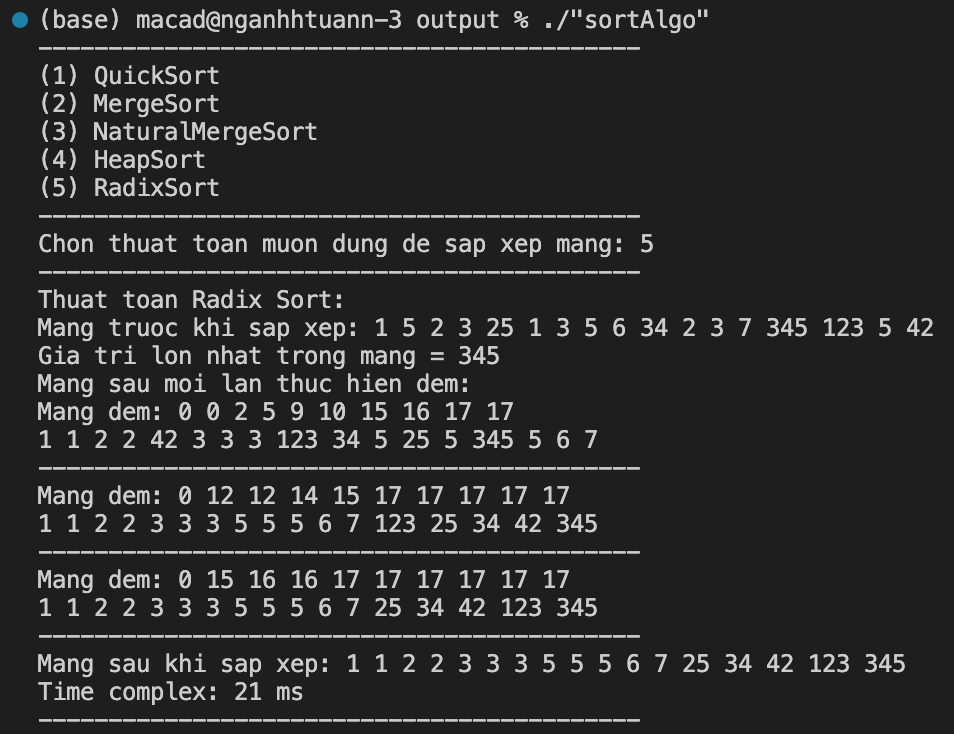


Các hàm bổ trợ:



**Kết quả:**

****

* QuickSort:****
* Merge Sort****
* Natural Merge Sort:
* Heap Sort:****
* Radix sort:****

**Nhận xét:**

Việc viết chương trình mô phỏng trực quan các thuật toán để tìm hiểu và dễ dàng trogn việc mường tượng mảng sau mỗi step sẽ trở thành như thế nào. Ngoài ra yêu cầu xuất ra độ phúc tạp thời gian cũng rất hữu ích để so sanh hiệu năng của các thuật toán với nhau với input đầu vào khác nhau

Trong đó độ phức tạp thuật toán sẽ được nhận xét như sau:

* QuickSort: trung bình O(nlogn), trường hợp xấu nhất O(n^2)
* MergeSort: trung bình O(nlogn) nhưng yêu cầu bộ nhớ hơn để chứa các mảng phụ, hiệu suất tốt trong mọi trường hợp
* Natural Merge Sort: trung bình O(nlogn) như merge sort giảm thiểu độ phức tạp ở các run
* HeapSort: trung bình O(nlogn) sử dụng tốt trong mọi trường hợp
* RadixSort: O(num(n+k)) trong đó num là số chữ số của số lớn nhất trong mảng, n là số lượng phần tử và k là số lượng giá trị khác nhau mà mỗi chữ số có thể có, thuật toán hiệu quả nhất khi các phần tử trong danh sách có cùng độ dài num.