**TRƯỜNG ĐẠI HỌC MỞ TP HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙡❄🙣**



**CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

### Bài Tập Chương 3 – Câu Hỏi

**GVHD: Thầy Lê Ngọc Hiếu**

|  |  |
| --- | --- |
| **Họ và tên** | **MSSV** |
| **Bùi Đăng Khoa** | **1654050050** |



NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

Tp. Hồ Chí Minh, ngày 29 tháng 7 Năm 2019

**Ths. Lê Ngọc Hiếu**

**CÂU HỎI**

**Câu 1: Trong các phương pháp sắp xếp thứ tự đã học, phương pháp nào tối ưu nhất? Phương pháp nào kém tối ưu nhất? Tại sao?**

- Trong các phương pháp sắp xếp thứ tự đã học phương pháp sắp xếp Quick Sort là tối ưu nhất. Vì có độ phức tạp trung bình O(n log( n )).

- Trong các phương pháp sắp xếp thứ tự đã học phương pháp sắp xếp Selection Sort là phương pháp kém tối ưu nhất. Vì giải thuật này không phù hợp với tập dữ liệu lớn khi mà độ phức tạp trường hợp xấu nhất và trường hợp trung bình là O(n2).

**Câu 2: Trong 2 phương pháp tìm kiếm đã học trường hợp nào thì cả 2 phương pháp đều như nhau? Giải thích tại sao?**

- 2 phương pháp tìm kiếm tuần tự và tìm kiếm nhị phân là như nhau nếu danh sách đã sắp xếp sẵn thứ tự và số phần tử trong danh sách cần tìm là như nhau.

**Câu 3: Ngoài các phương pháp sắp xếp thứ tự đã học, hãy tìm hiểu thêm một phương pháp xếp thứ tự khác, giới thiệu sơ và giải thích?**

- Giống như Quick sort, Merge sort là một thuật toán chia để trị.

- Thuật toán này chia mảng cần sắp xếp thành 2 nửa.

- Tiếp tục lặp lại việc này ở các nửa mảng đã chia.

- Sau cùng gộp các nửa đó thành mảng đã sắp xếp.

- Hàm Merge() được sử dụng để gộp hai nửa mảng. Hàm merge(arr, l, m, r) là tiến trình quan trọng nhất sẽ gộp hai nửa mảng thành 1 mảng sắp xếp, các nửa mảng là arr[l…m] và arr[m + 1…r] sau khi gộp sẽ thành một mảng duy nhất đã sắp xếp.

MergeSort(arr[], l, r)

If r > l

1. Tìm chỉ số nằm giữa mảng để chia mảng thành 2 nửa:

Middle m = (l + r) / 2.

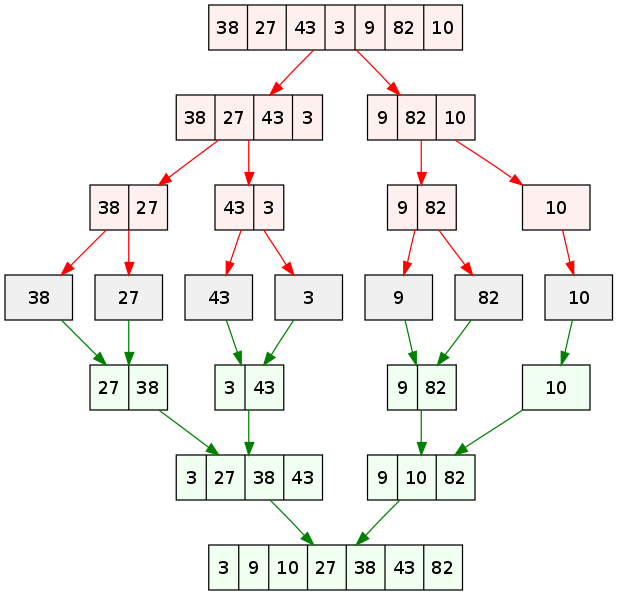
2. Gọi đệ quy hàm MergeSort cho nửa đầu tiên:

MergeSort(arr, l, m).

3. Gọi đệ quy hàm MergeSort cho nửa thứ hai:

MergeSort(arr, m + 1, r).

4. Gộp 2 nửa mảng đã sắp xếp ở (2) và (3):

MergeSort(arr, l, m, r).