**Câu 1:**

- Độ phức tạp của 6 phương pháp sắp xếp:

|  |  |
| --- | --- |
| THUẬT TOÁN SẮP XẾP | ĐỘ PHỨC TẠP |
| Selection Sort | O(n2) |
| Insertion Sort | O(n2) |
| Interchange Sort | O(n2) |
| Bubble Sort | O(n2) |
| Quick Sort | O(n log n) |
| Heap Sort | O(n log n) |

- Thời gian sắp xếp của dãy dữ liệu ngẫu nhiên:

Quick Sort > Heap Sort > Insertion Sort > Selection Sort > Interchange Sort> Bubble Sort

Quick Sort là thuật toán sắp xếp tối ưu nhất. *Heap Sort* và *Quick Sort* đều có độ phức tạp là O(n log n) nhưng thức tế thời gian thực hiện và thuật toán của Quick Sort tối ưu hơn so với *Heap Sort*. Còn phương pháp kém tối ưu nhất là phương pháp *Selection Sort* vì với phương pháp này, ta phải thực hiện lặp lại việc xét từng phần tử trong danh sách để chọn phần tử bé nhất đưa lên đầu danh sách. Việc lặp lại này làm cho thời gian thực hiện kéo dài và lâu hơn so với những phương pháp khác có độ phức tạp là O(n2).

**Câu 2:**

Trong hai phương pháp tìm kiếm đã học, cả hai phương pháp đều như nhau trong trường hợp danh sách chưa được sắp xếp thứ tự. Vì khi đó, nếu ta sử dụng phương pháp tìm kiếm nhị phân thì ta phải dùng thuật toán sắp xếp để sắp xếp lại danh sách rồi mới thực hiện tìm kiếm nhị phân. Do vậy, thời gian thực hiện đối với tìm kiếm nhị phân sẽ kéo dài ra và có thể tương đương với thời gian thực hiện tìm kiếm tuần tự.

**Câu 3:**

Ngoài các phương pháp xếp thứ tự đã học, ta còn có những phương pháp sắp xếp thứ tự khác trong đó có phương pháp Merge Sort:

* Ý tưởng của thuật toán Merge sort:

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:

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):

          merge(arr, l, m, r)

* Cách hàm merge hoạt động khi gộp hai mảng con :

Với trường hợp khi 2 mảng con chỉ có 1 phần tử, ta chỉ việc xem phần tử nào nhỏ hơn và đẩy lên đầu, phần tử còn lại đặt phía sau. Do vậy, các mảng con cần gộp lại có tính chất luôn được sắp tăng dần.

Ví dụ:

