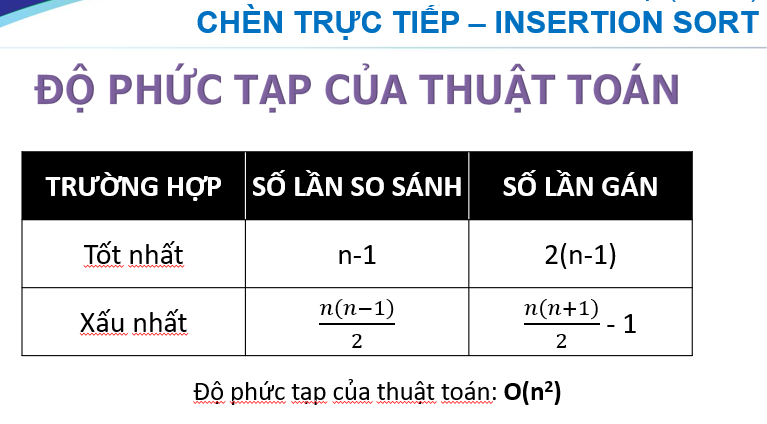
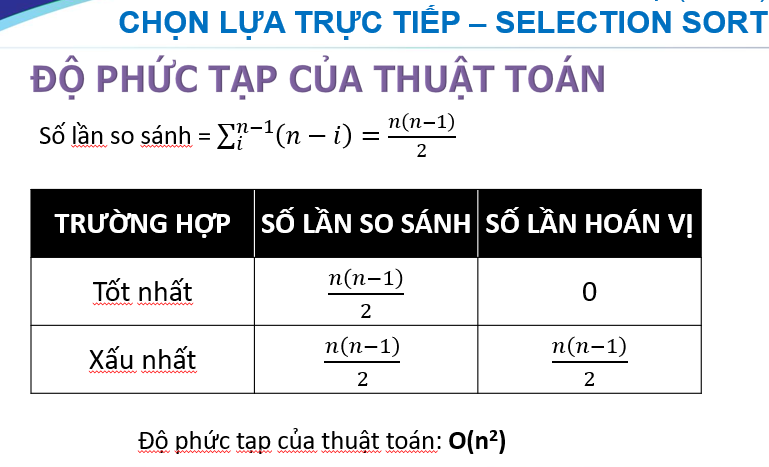
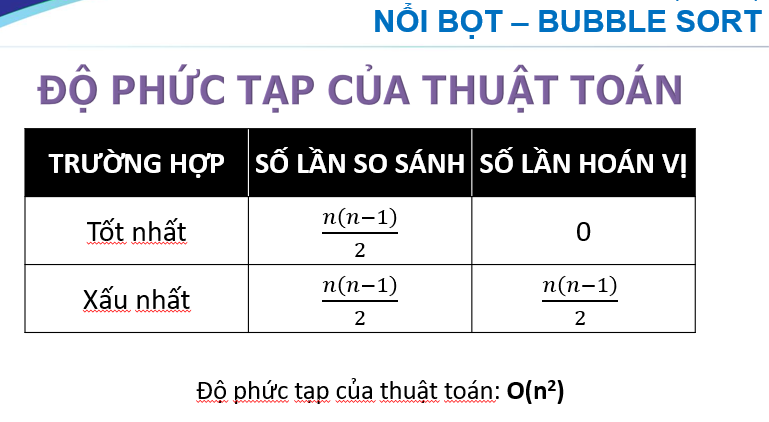
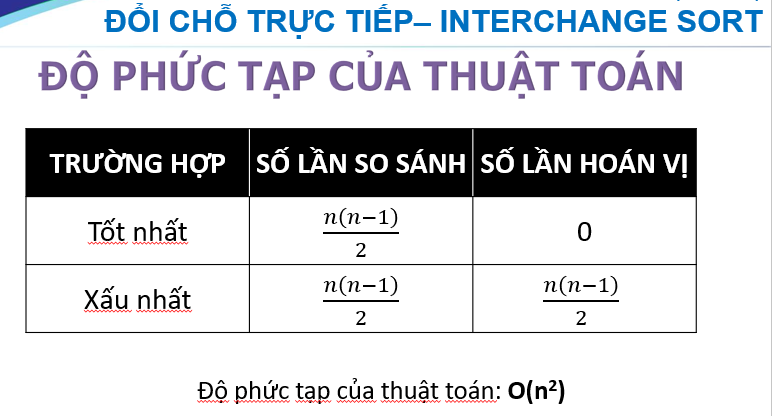
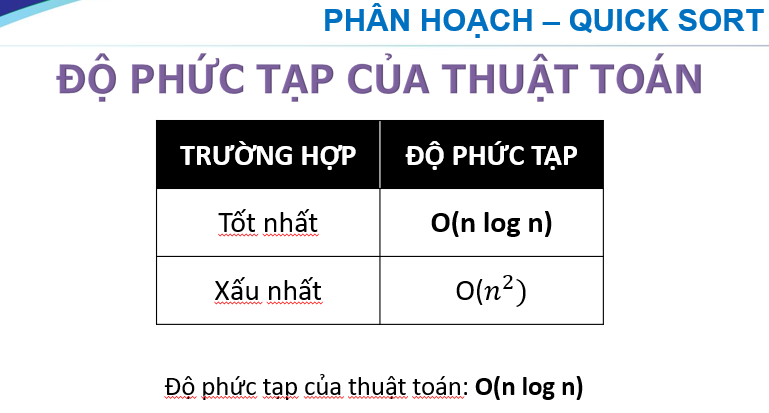
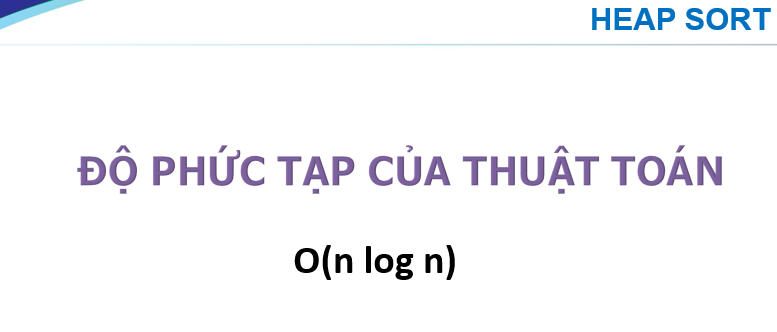
**Câu 1/ Trong các phương pháp xếp thứ tự đã học, phương pháp nào tối ưu nhất, và kém tối ưu nhất? Tại sao?**

* Các phương pháp xếp thứ tự đã học: Selection sort (Chọn lựa trực tiếp), Insertion sort (Chèn trực tiếp), Bubble sort (Nổi bọt), Interchance sort (Đổi chỗ trực tiếp), Quick sort (Phân hoạch), Heap sort



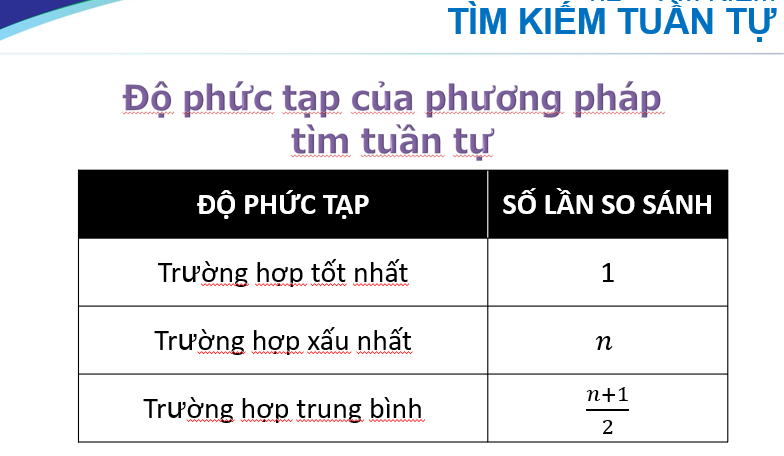
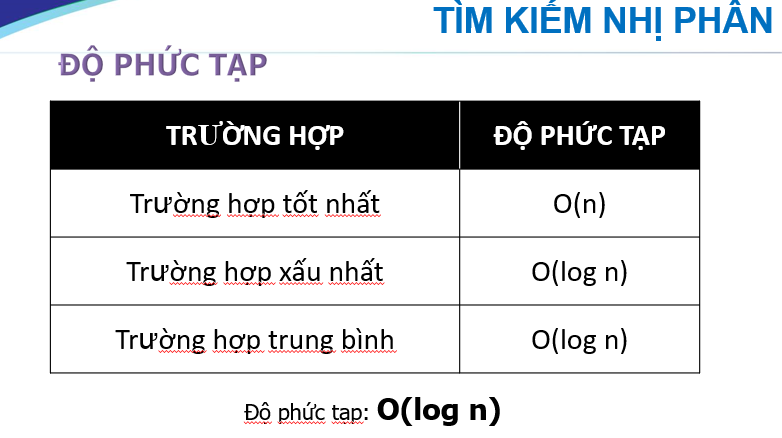
 

Dựa vào độ phức tạp của thuật toán ta thấy phương pháp Quick sort (Phân hoạch), Heap sort là tối ưu nhất với độ phức tạp là O(n log n). Còn phương pháp Selection sort (Chọn lựa trực tiếp), Insertion sort (Chèn trực tiếp), Bubble sort (Nổi bọt), Interchance sort (Đổi chỗ trực tiếp) là kém tối ưu nhất với độ phức tạp của thuật toán là O(n2).

Nhưng trên thực tế để xét về độ tối ưu nhất, kém tối ưu nhất của các phương pháp còn phụ thuộc các yếu tố như : dữ liệu đầu vào số lượng như thế nào, có sắp xếp sẵn hay không, dung lượng bộ nhớ ra sao, tốc độ xử lý CPU... Và với mỗi kiểu dữ liệu khác nhau thì lại có 1 kiểu sắp xếp chiếm ưu thế riêng. Nên không có bất kỳ 1 thuật toán nào là tối ưu nhất và kém tối ưu nhất.

**Câu 2/ Trong các 2 phương pháp tìm kiếm đã học, trường hợp nào thì cả 02 phương pháp đều như nhau? Giải thích tại sao?**

* 2 phương pháp tìm kiếm đã học : Tìm kiếm tuần tự (Sequential Search), tìm kiếm nhị phân (Binary Search).

Trong trường hợp danh sách đã được sắp xếp thứ tự thì ta dùng 1 trong 2 phương pháp trên đều được. Vì phương pháp tìm kiếm nhị phân chỉ được thực hiện trên 1 danh sách đã được xếp thứ tự còn phương pháp tìm kiếm tuần tự thì không bắt buộc.

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

* **Phương pháp nổi bọt cải tiến (Shake sort)**

Giải thuật

 Giải thuật sắp xếp ShakerSort cũng dựa trên nguyên tắc đổi chỗ trực tiếp, nhưng tìm cách khắc phục các nhược điểm của BubleSort với những ý tưởng cải tiến chính như sau :

Trong mỗi lần sắp xếp, duyệt mảng theo 2 lượt từ 2 phiá khác nhau :

+ Lượt đi: đẩy phần tử nhỏ về đầu mảng

+ Lượt về: đẩy phần tử lớn về cuối mảng

Ghi nhận lại những đoạn đã sắp xếp nhằm tiết kiệm các phép so sánh thừa.

Các bước tiến hành như sau :

Ý tưởng chính của giải thuật là xuất phát từ cuối (đầu) dãy, đổi chỗ các cặp phần tử kế cận để đưa phần tử nhỏ (lớn) hơn trong cặp phần tử đó về vị trí đúng đầu  (cuối) dãy hiện hành, sau đó sẽ không xét đến nó ở bước tiếp theo, do vậy ở lần xử lý thứ i sẽ có vị trí đầu dãy là i  . Lặp lại xử lý trên cho đến khi không còn cặp phần tử nào để xét.

* **Phương pháp Shell sort**

Shell Sort là một giải thuật sắp xếp mang lại hiệu quả cao dựa trên giải thuật sắp xếp chèn (Insertion Sort). Giải thuật này khá hiệu quả với các tập dữ liệu có kích cỡ trung bình khi mà độ phức tạp trường hợp xấu nhất và trường hợp trung bình là O(n), với n là số phần tử.

Giải thuật này tránh các trường hợp phải tráo đổi vị trí của hai phần tử xa nhau trong giải thuật sắp xếp chọn (nếu như phần tử nhỏ hơn ở vị trí bên phải khá xa so với phần tử lớn hơn bên trái).

Đầu tiên, giải thuật này sử dụng giải thuật sắp xếp chọn trên các phần tử có khoảng cách xa nhau, sau đó sắp xếp các phần tử có khoảng cách hẹp hơn. Khoảng cách này còn được gọi là khoảng (interval) – là số vị trí từ phần tử này tới phần tử khác. Khoảng này được tính dựa vào các loại công thức sau:

* Shell’s original sequence: N/2 , N/4 , …, 1
* Knuth’s increments: 1, 4, 13, …, (3k – 1) / 2
* Sedgewick’s increments: 1, 8, 23, 77, 281, 1073, 4193, 16577...4j+1+ 3·2j+ 1
* Hibbard’s increments: 1, 3, 7, 15, 31, 63, 127, 255, 511…
* Papernov & Stasevich increment: 1, 3, 5, 9, 17, 33, 65,...
* Pratt: 1, 2, 3, 4, 6, 9, 8, 12, 18, 27, 16, 24, 36, 54, 81....
* **Phương pháp sắp xếp đếm (Counting Sort)**

Counting sort là một thuật toán sắp xếp cực nhanh một mảng các phần tử mà mỗi phần tử là các số nguyên không âm; Hoặc là một danh sách các ký tự được ánh xạ về dạng số để sort theo bảng chữ cái. Counting sort là một thuật toán sắp xếp các con số nguyên không âm, không dựa vào so sánh.

Trong khi các thuật toán sắp xếp tối ưu sử dụng so sánh có độ phức tạp O(nlogn) thì Counting sort chỉ cần O(n) nếu độ dài của danh sách không quá nhỏ so với phần tử có giá trị lớn nhất.

* **Phương pháp sắp xếp theo cơ số (Radix Sort)**

Sắp xếp dựa trên cơ số là một kỹ thuật sắp xếp các phần tử bằng cách nhóm các chữ số riêng lẻ của một giá trị có cùng một vị trí. Sau đó, sắp xếp các phần tử theo thứ tự tăng hoặc giảm.

Sắp xếp cơ số thường được dùng để sắp xếp số có nhiều chữ số (số lớn)

Giả sử, chúng ta có một mảng gồm 8 phần tử. Đầu tiên, chúng ta sẽ sắp xếp các phần tử dựa trên giá trị của vị trí đơn vị. Sau đó, chúng ta sẽ sắp xếp các phần tử dựa trên giá trị của vị trí thứ mười. Quá trình này tiếp tục cho đến vị trí cuối cùng.