## Chương 3

## Câu Hỏi

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

* Phương pháp tối ưu nhất: Quick Sort

Tốc độ xử lý nhanh, không lo lắng về đầu vào kể cả trường hợp xấu nhất.

* Phương pháp kém tối ưu nhất: Bubble Sort

Hoạt động chậm nhất , xét từng cặp kề nhau lặp lại nhiều lần đến khi không còn phần tử nào để xét, không đủ nhanh cho dữ liệu lớn.

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

Bubble Sort và Interchange Sort.

So sánh các cặp liền kề nhau, nếu phần tử sau nào nhỏ hơn phần tử trước thì hoán vị hai phần tử cho nhau.

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

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

 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.

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.

*void ShakeSort(int a[], int N ) {*

*int   i, j;*

*int   left, right, k;*

*left = 0; right = n-1; k = n-1;*

*while (left < right) {*

*for (j = right; j > left; j--) {*

*if (a[j]< a[j-1]) { // sai vị trí thì đổi chỗ*

*Hoanvi(a[j],a[j-1]);*

*k = j;*

*}*

*}*

*left = k;*

*for (j = left; j < right; j++)  {*

*if (a[j]> a[j+1]) {  // sai vị trí thì đổi chỗ*

*Hoanvi(a[j],a[j-1]);*

*k = j;*

*}*

*}*

*right = k;*

*}*

*}*