Bài 2:

2.3 Phương pháp sắp xếp Interchange Sort

Bắt đầu i = 0 ; j = i + 1 ;

Vị trí : a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7]

Phần tử: 40 70 20 60 90 10 50 30

i=0 j=1

a[i] < a[j] => không đổi; j++;

Phần tử: 40 70 20 60 90 10 50 30

i=0 j=2

a[i] > a[j] =>swap(a[i], a[j]); j++;

Phần tử: 20 70 40 60 90 10 50 30

i=0 j=3

a[i] < a[j] =>không đổi; j++;

Phần tử: 20 70 40 60 90 10 50 30

i=0 j=4

a[i] < a[j] =>không đổi; j++;

Phần tử: 20 70 40 60 90 10 50 30

i=0 j=5

a[i] > a[j] => swap(a[i], a[j]); j++;

Phần tử: 10 70 40 60 90 20 50 30

i=0 j=6

a[i] < a[j] =>không đổi; j++;

Phần tử: 10 70 40 60 90 20 50 30

i=0 j=7

a[i] < a[j] =>không đổi; j++;

j = 8 > 7 kết thúc vòng lặp j; i++;

Lặp lại các bước trên cho i = 1; j từ 2 đến 7. Và tương tự cho trường hợp i = 2, 3, 4,5,6

Ta được dãy sắp xếp thứ tự tăng dần:

0 1 2 3 4 5 6 7

a 10 20 30 40 50 60 70 90

\* Tính độ phức tạp

* Vì ta biết độ phức tạp của for thuộc lớp O(n)
* Mà có hai vòng lặp for lòng nhau
* Độ phức tạp của thuật toán Interchange Sort là O(n2)