**Bài Tập Lí Thuyết**

**Chương 3**

**Câu 2**: Một danh sách các phần tử được lưu trữ trong một danh sách đặc cóc các phần tử sau : 40 , 70, 20, 60, 90, 10, 50, 30.(sắp xếp giảm dần ).

2.1 : InsertionSort.

Giả sử danh sách đã được xếp thứ tự .

Chọn a[0] là giá trị phần tử đầu tiên .

So sánh a[0] với a[1], a[0] = 40 < a[1] = 70 nên chèn 70 vào trước 40

70 40 20 60 90 10 50 30.

a[2] < a[1] nên giữ nguyên vị trí .

70 40 20 60 90 10 50 30.

a[1]< a[3] <a[0] nên chèn 60 vào trước 70 và sau 40.

70 60 40 20 90 10 50 30.

a[4] > a[0] nên chèn 90 trước a[0].

90 70 60 40 20 10 50 30.

a[5] < a[4] nên giữ nguyên vị trí.

90 70 60 40 20 10 50 30.

a[3]< a[6] < a[2] nên chèn 50 vào trước 40 và sau 60.

90 70 60 50 40 20 10 30

a [5]< a[7] < a[4] nên chèn 30 và sau 40 và trước 20

90 70 60 50 40 30 20 10.

Độ phức tạp của thuật toán là O( )

2.2: SelectionSort

40 70 20 60 90 10 50 30.

i = 0;

Tìm giá trị lớn nhất từ i đến n-1;

Max-pos = a[5]

Hoán vị a[5] và a[0] ta được:

90 70 20 60 40 10 50 30.

i tăng lên 1, i = 1 .

Tìm giá trị lớn nhất từ i đến n – 1.

Maxpos = a[1] nên không hoán vị.

90 70 20 60 40 10 50 30.

i tăng lên 2, i = 2.

Tìm giá trị lớn nhất từ i đến n – 1.

Maxpos = a[3] nên hoán vị a[3] với a[2] .

90 70 60 20 40 10 50 30.

i tăng lên 3, i = 3 .

Tìm giá trị lớn nhất từ i đến n – 1.

Maxpos = a[6] nên hoán vị a[6] với a[3].

90 70 60 50 40 10 20 30.

i tăng lên 4, i = 4 .

Tìm giá trị lớn nhất từ i đến n – 1.

Maxpos = a[4] nên không hoán vị.

90 70 60 50 40 10 20 30.

i tăng lên 5, i = 5 .

Tìm giá trị lớn nhất từ i đến n – 1.

Maxpos = a[7] nên hoán vị a[7] với a[5]

90 70 60 50 40 30 20 10.

i tăng lên 6, i = 6 .

Tìm giá trị lớn nhất từ i đến n – 1.

Maxpos = a[6] nên không hoán vị.

90 70 60 50 40 30 20 10.

Tới đây i tăng lên 7 lớn hơn n-1 = 6 nên dừng.

Mảng sắp xếp được là 90 70 60 50 40 30 20 10.

Độ phức tạp của thuật toán là O( )

2.3 InterchangeSort

40 70 20 60 90 10 50 30.

Bắt đầu từ phần tử đầu tiên i = 0,

a[0] = 40.

So sánh a[0] cho a[i+1] đến a[n-1] nến nhỏ hơn thì hoán vị.

a[0] = 40 với a[1] = 70, a[0] < a[1] hoán vị a[1] cho a[0].

70 40 20 60 90 10 50 30.

Lúc này a[0] = 70 đi so sánh với a[2], a[0] = 70 > 20 = a[2] nên không thay đổi vị trí.

70 40 20 60 90 10 50 30.

So sánh a[0] với a[3], a[0] > a[3] giữ nguyên vị trí,

70 40 20 60 90 10 50 30

a[0] < a[4] hoán vị a[0] và a[4].

90 40 20 60 70 10 50 30.

a[0] = 90 > a[5] giu nguyên vị trí

90 40 20 60 70 10 50 30.

a[0] = 90 > a[6] giu nguyên vị trí

90 40 20 60 70 10 50 30.

a[0] = 90 > a[7] giu nguyên vị trí

90 40 20 60 70 10 50 30.

Lúc này i > n-1 nên lặp lại các bước trên với i = 1.

Và tương tự cho i = 2, 3… n-2.

Độ phức tạp của thuật toán là O( )

2.4 Bubble Sort.

40 70 20 60 90 10 50 30.

Bắt đầu từ cuối dãy với j = n -1 và i = 0 , so sánh các phần tử kề nhau nếu a[j] lớn hơn a[j-1] thì đổi chổ.

j = 7, a[j] bé hơn a[j-1] nên giữ nguyên vị trí.

40 70 20 60 90 10 50 30.

j =6, a[j] lớn hơn a[j-1] nên swap(a[j], a[j-1])

40 70 20 60 90 50 10 30

j =5, a[j] bé hơn a[j-1] nên giữ nguyên vị trí.

40 70 20 60 90 50 10 30.

j = 4 , a[j] lớn hơn a[j-1] nên swap(a[4],a[3])

40 70 20 90 60 50 10 30.

j = 3 , a[j] lớn hơn a[j-1] nên swap(a[3],a[2])

40 70 90 20 60 50 10 30

j = 2 , a[j] lớn hơn a[j-1] nên swap(a[2],a[1])

40 90 70 20 60 50 10 30

j = 1 , a[j] lớn hơn a[j-1] nên swap(a[1],a[0])

90 40 70 20 60 50 10 30.

j không lớn hơn i nữa nên i tăng lên i = 1 và j = 7 , tiếp tục thực hiện như các bước trên để sắp xếp mảng theo thứ tự giảm dần và kết thúc khi i > n – 1

Độ phức tạp của thuật toán là O( ).

2.5 : Quick Sort.

40 70 20 60 90 10 50 30.

Khởi tạo left = 0 và right = 7, i = 0 và j = 7;

X = ( left + right ) /2

= ( 0 + 7) /2 = 3.5

X =a[3] = 60.

i = 0 , a[0] < x, i dừng .

j = 7 ,a[j] = a[7] < x, nên j-- ;

j = 6, a[j] = a[6] < x, nên j--;

j = 5,a[j] = a[5] < x, nên j --;

j=4, a[j] = a[4] > x, dừng j.

swap ( a[0], a[4]) , i++ và j—

90 70 20 60 40 10 50 30.

i = 1 , a[i] = a[1] = 70 > x , i++

i = 2, a[i] = a[2] =20 < x, dừng i

j = 3, a[j] = a[3] = 60 > x , sai dừng j

swap(a[2],a[3]) i++, j--.

90 70 60 20 40 10 50 30

Lúc này j = 2 < i =3 dừng . Dãy ban đầu phân hoạch thành 2 dãy con là

Dãy 1 : 90 70 60

Dãy 2: 20 40 10 50 30.

Tiếp tục phân hoạch 2 dãy con trên cho đên khi mổi đoạn con có 1 phần tử.

Độ phức tạp của thuật toán là : O( n log n)

2.6 HeapSort ( thứ tự tăng dần )

40 70 20 60 90 10 50 30.

Tạo heap bằng i = n /2 = 8/2 = 4

40 70 20 60 | 90 10 50 30.

Với i = 3 ta so sánh với vị trí 2\*i + 1 = 7

So sánh a[3], a[7] tìm giá trị lớn nhất a[3] > a[7] nên giữ nguyên.

40 70 20 60 | 90 10 50 30.

Với i = 2 ta so sánh với vị trí 2\*i + 1 = 5 và 2\*i + 2= 6

So sánh a[2],a[5],a[6] là a[6] = 50 thực hiện hoán vị a[2] và a[6]

40 70 50 60 | 90 10 20 30

Với i = 1

So sánh a[1], a[3],a[4] giá trị lớn nhất là a[4] = 90 thực hiện hoán vị a[1] và a[4].

40 90 50 60 | 70 10 20 30.

Với i = 0

So sánh a[0], a[1],a[2] giá trị lớn nhất là a[1] = 90 thực hiện hoán vị a[1] và a[0]

90 40 50 60 | 70 10 20 30

Xét lại tại vị trí a[1] sau khi a[1] nhận giá trị mới là 40

So sánh a[1] , a[3], a[4] gí trị lớn nhất là a[4] hoán vị a[1] và a[4]

90 70 50 60 | 40 10 20 30.

Lúc này hoán vị a[0] với a[7]

30 70 50 60 40 10 20 90

Tiếp theo ta giới hạn phần tử cuối

30 70 50 60 40 10 20 | 90

Tao heap mới cho các phần tử đang xét từ a[0] đến a[6] và thực hiện như các bước trên và lặp lại các bước này cho đến khi danh sách trên được sắp xếp.