**BÀI TẬP LÀM THÊM CHƯƠNG 3**

**Bài 2:** Danh sách đặc:

a 40 70 20 60 90 10 50 30

i 0 1 2 3 4 5 6 7

**2.1. InsertionSort**

B1: i = 1; //đoạn a[0], có 1 phần tử được xem là danh sách (danh sách có 1 phần tử) đã được xếp thứ tự

40 70 20 60 90 10 50 30

i=1

B2: Thực hiện gán giá trị x = a[i] //x = 70

B3: Tìm j (j đi từ vị trí i-1 sang trái). j là vị trí phần tử ở đầu tiên mà a[j] nhỏ hơn hoặc bằng x. Do đó j + 1 là vị trí thích hợp chèn x vào. Tịnh tiến đoạn các phần tử từ a[i+j] đến a[i-1] sang phải 1 vị trí.

B4: Thực hiện gán giá trị a[j+1] = x; //j+1 là vị trí thích hợp chèn x vào

40 70 20 60 90 10 50 30

i=1

j=0 j+1=1

Vì a[j]=40 < a[i]=70 nên gán a[j+1] = a[1] = x = 70

B5: Xét vị trí tiếp theo (i++) và NẾU i<n: LẶP LẠI B2

40 70 20 60 90 10 50 30

i=2

j=0

20 40 70 60 90 10 50 30

i=3

j=1 j+1=2

20 40 60 70 90 10 50 30

i=4

j=3 j+1=4

20 40 60 70 90 10 50 30

i=5

j=0

10 20 40 60 70 90 50 30

i=6

j=2 j+1=3

10 20 40 50 60 70 90 30

i=7

j=1 j+1=2

i=7=n →dừng. Ta được dãy đã được sắp xếp thứ tự:

10 20 30 40 50 60 70 90

\*Độ phức tạp: O()

**2.2. SelectionSort**

i = 0;

B1:

* Tìm phần tử a[min\_pos] nhỏ nhất trong dãy hiện hành từ a[i] đến a[n-1]

40 70 20 60 90 10 50 30

i=0

a[5] là phần tử nhỏ nhất

* Đổi chỗ a[min\_pos] và a[i]

Đổi chỗ a[5] và a[0]

10 70 20 60 90 40 50 30

B2: i+1;

Nếu i < n-1 thì lặp lại bước 1

10 70 20 60 90 40 50 30

i=1

10 20 70 60 90 40 50 30

i=2

10 20 30 60 90 40 50 70

i=3

10 20 30 40 90 60 50 70

i=4

10 20 30 40 50 60 90 70

i=5

10 20 30 40 50 60 90 70

i=6

10 20 30 40 50 60 70 90

i=7

i = 7 = n-1 → dừng, ta được dãy đã xếp thứ tự:

10 20 30 40 50 60 70 90

\*Độ phức tạp: O()

**2.3. InterchangeSort**

B1: Bắt đầu từ phần tử đầu tiên i = 0 trong danh sách

B2: Xét phần tử tại vị trí j = i + 1

Lặp lại trong khi (j <= n - 1)

{

Nếu a[j] < a[i] thì hoán vị a[j] và a[i]

j++

}

40 70 20 60 90 10 50 30

i=0 j=1

40 70 20 60 90 10 50 30

i=0 j=2

20 70 40 60 90 10 50 30

i=0 j=3

20 70 40 60 90 10 50 30

i=0 j=4

20 70 40 60 90 10 50 30

i=0 j=5

10 70 40 60 90 20 50 30

i=0 j=6

10 70 40 60 90 20 50 30

i=0 j=7

10 70 40 60 90 20 50 30

j+1=8 > n-1 → dừng

B3: i++

Nếu i < n-1, lặp lại bước 2

10 70 40 60 90 20 50 30

i=1 j=2

10 40 70 60 90 20 50 30

i=1 j=3

10 40 70 60 90 20 50 30

i=1 j=4

10 40 70 60 90 20 50 30

i=1 j=5

10 20 70 60 90 40 50 30

i=1 j=6

10 20 70 60 90 40 50 30

i=1 j=7

10 20 70 60 90 40 50 30

i=2 j=3

10 20 60 70 90 40 50 30

i=2 j=4

10 20 60 70 90 40 50 30

i=2 j=5

10 20 40 70 90 60 50 30

i=2 j=6

10 20 40 70 90 60 50 30

i=2 j=7

10 20 30 70 90 60 50 40

i=3 j=4

10 20 30 70 90 60 50 40

i=3 j=5

10 20 30 60 90 70 50 40

i=3 j=6

10 20 30 50 90 70 60 40

i=3 j=7

10 20 30 40 90 70 60 50

i=4 j=5

10 20 30 40 70 90 60 50

i=4 j=6

10 20 30 40 60 90 70 50

i=4 j=7

10 20 30 40 50 90 70 60

i=5 j=6

10 20 30 40 50 70 90 60

i=5 j=7

10 20 30 40 50 60 90 70

i=6 j=7

10 20 30 40 50 60 70 90

j = 7 = n-1 → dừng, ta được dãy đã xếp thứ tự:

10 20 30 40 50 60 70 90

\*Độ phức tạp: O()

**2.4. BubbleSort**

B1: i=0;

B2: j = n-1; //Bắt đầu từ vị trí cuối danh sách

Lặp lại trong khi (j > i)

Nếu a[j] < a[j-1]

swap(a[j], a[j-1]);

j--;

40 70 20 60 90 10 50 30

i=0 j-1=6 j=7

40 70 20 60 90 10 30 50

i=0 j-1=5 j=6

40 70 20 60 90 10 30 50

i=0 j-1=4 j=5

40 70 20 60 10 90 30 50

i=0 j-1=3 j=4

40 70 20 10 60 90 30 50

i=0 j-1=2 j=3

40 70 10 20 60 90 30 50

i=0 j-1=1 j=2

40 10 70 20 60 90 30 50

i=0

j-1=0 j=1

10 40 70 20 60 90 30 50

i=0

B3: i++;

Nếu i < n-1: lặp lại bước 2.

10 40 70 20 60 90 30 50

i=1 j=7

10 40 70 20 60 90 30 50

i=1 j=6

10 40 70 20 60 30 90 50

i=1 j=5

10 40 70 20 30 60 90 50

i=1 j=4

10 40 70 20 30 60 90 50

i=1 j=3

10 40 20 70 30 60 90 50

i=1 j=2

10 20 40 70 30 60 90 50

i=2 j=7

10 20 40 70 30 60 50 90

i=2 j=6

10 20 40 70 30 50 60 90

i=2 j=5

10 20 40 70 30 50 60 90

i=2 j=4

10 20 40 30 70 50 60 90

i=2 j=3

10 20 30 40 70 50 60 90

i=3 j=7

10 20 30 40 70 50 60 90

i=3 j=6

10 20 30 40 70 50 60 90

i=3 j=5

10 20 30 40 50 70 60 90

i=3 j=4

10 20 30 40 50 70 60 90

i=4 j=7

10 20 30 40 50 70 60 90

i=4 j=6

10 20 30 40 50 60 70 90

i=4 j=5

10 20 30 40 50 60 70 90

i=5 j=7

10 20 30 40 50 60 70 90

i=5 j=6

10 20 30 40 50 60 70 90

i=6 j=7

10 20 30 40 50 60 70 90

i=7

i = 7 = n-1 → dừng, ta được dãy đã xếp thứ tự:

10 20 30 40 50 60 70 90

\*Độ phức tạp: O()

**2.5. QuickSort**

Vị trí đầu danh sách left = 0, vị trí cuối danh sách right = n-1

Trong khi left < right

Phân hoạch danh sách các phần tử: a[left],…,a[right] thành 2 đoạn:

+ Đoạn 1: a[left],…,j (các phần tử <=x)

+ Đoạn 2: a[(n-1)/2+1],…,a[right] (các phần tử >x)

Thực hiện tương tự các bước trên cho mỗi đoạn con 1 và đoạn con 2, cho đến khi mỗi đoạn con còn 1 phần tử.

40 70 20 60 90 10 50 30

0 1 2 3 4 5 6 7

x = a[(n-1)/2] = a[3] = 60

40 70 20 60 90 10 50 30

i=left j=right

a[i] = a[0] = 40 < x, i++

a[i] = a[1] = 70 > x, dừng i (i lúc này = 1)

a[j] = a[7] = 30 < x, dừng j (j lúc này = 7)

Hoán vị a[1] và a[7], i++, j--

40 30 20 60 90 10 50 70

0 1 2 3 4 5 6 7

a[i] = a[2] = 20 < x, i++

a[i] = a[3] = 60 > x, dừngi (lúc này i = 3)

a[j] = a[6] = 50 < x, dừng j (lúc này j = 6)

Hoán vị a[3] và a[6], i++, j--

40 30 20 50 90 10 60 70

0 1 2 3 4 5 6 7

a[i] = a[4] = 90 > x, dừng i (lúc này i = 4)

a[j] = a[5] = 10 < x, dừng j (lúc này j = 5)

Hoán vị a[4] và a[5], i++, j--

40 30 20 50 10 90 60 70

0 1 2 3 4 5 6 7

j<i, dừng

Dãy ban đầu được phân hoạch thành 2 đoạn con. Mỗi đoạn là 1 dãy con như sau:

Dãy con 1: gồm các phần tử a[0] đến a[4]: 40 30 20 50 10

Dãy con 2: gồm các phần tử a[5] đến a[7]: 90 60 70

Trên dãy con 1, ta thực hiên lặp lại việc phân hoạch với left = 0, right = 4

Trên dãy con 2, ta thực hiện lặp lại việc phân hoạch với left = 5, right = 7

Tiếp tục thực hiện phân hoạch các dãy con cho đến khi mỗi dãy còn tối đa một phần tử, khi đó danh sách ban đầu được xếp thứ tự.

\*Dãy con 1

x = 20

40 30 20 50 10

10 30 20 50 40

10 20 30 50 40

Chia thành 2 đoạn con:

10 20 | 30 50 40

10 20 | 30 40 | 50

10 | 20 | 30 40 50

10 | 20 | 30 | 40 | 50

\*Dãy con 2

x = 60

90 60 70

60 70 90

60 | 70 90

60 | 70 | 90

Ghép các đoạn con lại ta được dãy đã xếp thứ tự:

10 20 30 40 50 60 70 90

\*Độ phức tạp: O(n)

**2.6. HeapSort**

40 70 20 60 90 10 50 30

B1: Tạo Heap

Chia dãy ban đầu a[0], a[1],…, a[n-1], thành 2 phần:

Nửa dãy bên trái: a[0],…, a[(n/2)-1]: 40 70 20

Nửa dãy bên phải: a[n/2],…, a[n-1]: 60 90 10 50 30

40 70 20 60 | 90 10 50 30

0 1 2 3 4 5 6 7

So sánh phần tử cuối của dãy bên trái i=3 với 2 phần tử tại vị trí 2\*i+1 và 2\*i+2

40 70 20 60 | 90 10 50 30

3 7

Giá trị lớn nhất của a[3], a[7] là 60, dãy không đổi. Tiếp tục giảm i và so sánh với phần tử tại vị trí 2\*i+1 và 2\*i+2

40 70 20 60 | 90 10 50 30

2 5 6

Phần tử lớn nhất của a[2], a[5], a[6] là a[6] = 50, hoán vị a[6] và a[2]. Tiếp tục giảm i và so sánh

40 70 50 60 | 90 10 20 30

1 3 4

40 90 50 60 | 70 10 20 30

0 1 2

90 40 50 60 | 70 10 20 30

Xét lại tính lan truyền tại vị trí a[1], sau khi a[1] nhận gia trị mới là 40 (thay thế giá trị 90 trước đó; giá trị a[1] = 90 trước đó thỏa tính của Heap so với a[3] và a[4])

90 40 50 60 | 70 10 20 30

1 3 4

90 70 50 60 | 40 10 20 30

2 5 6

90 70 50 60 | 40 10 20 30

3 7

90 70 50 60 | 40 10 20 30

B2: Hoán vị phần tử đầu và phần tử cuối của dãy

B3: Giới hạn phần tử cuối dãy. Ta được dãy sau:

30 70 50 60 40 10 20 90

Tạo Heap ban đầu với dãy các phần tử từ a[0] đến a[6]

30 70 50 | 60 40 10 20

2 5 6

30 70 50 | 60 40 10 20

1 3 4

30 70 50 | 60 40 10 20

0 1 2

70 30 50 | 60 40 10 20

1 3 4

70 60 50 | 30 40 10 20

2 5 6

70 60 50 | 30 40 10 20

20 60 50 30 40 10 70

20 60 50 | 30 40 10

2 5

20 60 50 | 30 40 10

1 3 4

20 60 50 | 30 40 10

0 1 2

60 20 50 | 30 40 10

1 3 4

60 40 50 | 30 20 10

2 5

60 40 50 | 30 20 10

10 40 50 30 20 60

10 40 | 50 30 20

1 3 4

10 40 | 50 30 20

0 1 2

50 40 | 10 30 20

1 3 4

50 40 | 10 30 20

20 40 10 30 50

20 40 | 10 30

1 3

20 40 | 10 30

0 1 2

40 20 | 10 30

1 3

40 30 | 10 20

20 30 10 40

20 | 30 10

0 1 2

30 | 20 10

10 20 30

10 | 20

0 1

10 20

Dãy sau khi được sắp xếp: 10 20 30 40 50 60 70 90

\*Độ phức tạp: O(n)

**2.7. Tìm kiếm giá trị 90**

**Tìm kiếm tuần tự**

10 20 30 40 50 60 70 90

0 1 2 3 4 5 6 7

Bắt đầu từ i=0, nếu a[i] = 90: Tìm thấy giá trị x=90 trong dãy, return vị trí i. Ngược lại tăng i lên 1 và tiếp tục so sánh

8 lần so sánh: trường hợp xấu nhất

Độ phức tạp: O(n)

**Tìm kiếm nhị phân**

10 20 30 40 50 60 70 90

0 1 2 3 4 5 6 7

a[(0+7)/2] = a[3] = 40 < 90. Xét dãy từ a[4] đến a[7]:

50 60 70 90

4 5 6 7

a[5] = 60 < 90. Xét dãy từ a[2] đến a[3]:

70 90

6 7

a[6] = 70 < 90. Xét dãy còn lại a[7] → tìm thấy x = 90 tại vị trí i = 7

Độ phức tạp: O()