Lê Thị Như Quỳnh - 2154050259 – IT2101 – Chương 3

**BÀI TẬP LÀM THÊM CHƯƠNG 3**

**Bài 2:** Một danh sách các phần tử được lưu trữ trong một danh sách đặc, có các phần tử sau: 40, 70, 20, 60, 90, 10, 50, 30. Yêu cầu:

**2.1 Dùng phương pháp xếp thứ tự InsertionSort, mô tả từng bước quá trình xếp thứ tự dãy số trên (không lập trình). Tín độ phức tạp của quá trình xếp thứ tự danh sách trên**

40, 70, 20, 60, 90, 10, 50, 30

Giả sử phần tử đầu tiên đã có giá trị

* i= 1, xét a[1] = 70 > a[0]= 40

=> Chèn a[1] vào bên phải a[0]

40 70 20 60 90 10 50 30

* i= 2, xét a[2]= 20 < a[0]= 40

=> Chèn a[2] vào bên trái a[0]

20 40 70 60 90 10 50 30

* i= 3, xét a[3]= 60 < a[2]= 70

=> Chèn a[3] vào bên trái a[2]

20 40 60 70 90 10 50 30

* i= 4, xét a[4]=90 > a[3]= 70

=> Chèn a[4] vào bên phải a[3]

20 40 60 70 90 10 50 30

* i=5, xét a[5]=10 < a[0]= 20

=> Chèn a[5] vào bên phải a[0]

10 20 40 60 70 90 50 30

* i=6, xét a[6]=50 < a[3]= 60

=> Chèn a[6] vào bên trái a[3]

10 20 40 50 60 70 90 30

* i=7, xét a[7]=30 < a[2]= 40

=> Chèn a[7] vào bên trái a[2]

10 20 30 40 50 60 70 90

=> Ta có dãy đã sắp xếp là: 10 20 30 40 50 60 70 90

Độ phức tạp của thuật toán: **O(n2)**

**2.2 Dùng phương pháp xếp thứ tự SelectionSort, mô tả từng bước quá trình xếp thứ tự dãy số trên (không lập trình). Tín độ phức tạp của quá trình xếp thứ tự danh sách trên.**

40 70 20 60 90 10 50 30

**B1**: i= 0, Duyệt từ 0-> 7, a[0] -> a[7]

40 70 20 60 90 10 50 30

-> min\_pos= 5

-> Đổi chỗ a[0] với a[5]

10 70 20 60 90 40 50 30

**B2**: i= 1, Duyệt từ 1-> 7, a[1] -> a[7]

10 70 20 60 90 40 50 30

-> min\_pos= 2

-> Đổi chỗ a[1] với a[2]

10 20 70 60 90 40 50 30

**B3**: i= 2, Duyệt từ 2-> 7, a[2] -> a[7]

10 20 70 60 90 40 50 30

-> min\_pos= 7

-> Đổi chỗ a[2] với a[7]

10 20 30 60 90 40 50 70

**B4**: i= 3, Duyệt từ 3-> 7, a[3] -> a[7]

10 20 30 60 90 40 50 70

-> min\_pos= 5

-> Đổi chỗ a[3] với a[5]

10 20 30 40 90 60 50 70

**B5**: i= 4, Duyệt từ 4-> 7, a[4] -> a[7]

10 20 30 40 90 60 50 70

-> min\_pos= 6

-> Đổi chỗ a[4] với a[6]

10 20 30 40 50 60 90 70

**B6**: i= 5, Duyệt từ 5-> 7, a[5] -> a[7]

10 20 30 40 50 60 90 70

-> min\_pos= 5

-> Đổi chỗ a[5] với a[5]

10 20 30 40 50 60 90 70

**B7**: i= 6, Duyệt từ 6-> 7, a[6] -> a[7]

10 20 30 40 50 60 90 70

-> min\_pos= 7

-> Đổi chỗ a[6] với a[7]

10 20 30 40 50 60 70 90

=> Ta có dãy đã sắp xếp là : 10 20 30 40 50 60 70 90

Độ phức tạp của thuật toán: **O(n2)**

**2.3 Dùng phương pháp xếp thứ tự InterchangeSort, mô tả từng bước quá trình xếp thứ tự dãy số trên (không lập trình). Tín độ phức tạp của quá trình xếp thứ tự danh sách trên**.

40 70 20 60 90 10 50 30

**B1:** i= 0

* j= i+ 1= 1, a[0]= 40 < a[1]= 70 => không swap

40 70 20 60 90 10 50 30

j= 2, a[0]= 40 > a[2]= 20 => swap

20 70 40 60 90 10 50 30

* j= 3, a[0]= 20 < a[3]= 60 => swap

20 70 40 60 90 10 50 30

* j= 4, a[0]= 20 < a[4]= 90 => không swap

20 70 40 60 90 10 50 30

* j= 5, a[0]= 20 > a[5]= 10 => swap

10 70 40 60 90 20 50 30

* j= 6, a[0]= 10 < a[6]= 50 => không swap

10 70 40 60 90 20 50 30

* j= 7, a[0]= 10 < a[7]= 30 => không swap

10 70 40 60 90 20 50 30

**B2:** i= 1

* j= 2, a[1]= 70 > a[2]= 40 => swap

10 40 70 60 90 20 50 30

* j= 3, a[1]= 40 < a[3]= 60 => không swap

10 40 70 60 90 20 50 30

* j= 4, a[1]= 40 < a[4]= 90 => không swap

10 40 70 60 90 20 50 30

* j= 5, a[1]= 40 > a[5]= 20 => swap

10 20 70 60 90 40 50 30

* j= 6, a[1]= 20 < a[6]= 50 => không swap

10 20 70 60 90 40 50 30

* j= 7, a[1]= 20 < a[7]= 30 => không swap

10 20 70 60 90 40 50 30

**B3:** i= 2

* j= 3, a[2]= 70 > a[3]= 60 => swap

10 20 60 70 90 40 50 30

* j= 4, a[2]= 60 < a[4]= 90 => không swap

10 20 60 70 90 40 50 30

* j= 5, a[2]= 60 > a[5]= 40 => swap

10 20 40 70 90 60 50 30

j= 6, a[2]= 40 < a[6]= 50 => không swap

10 20 40 70 90 60 50 30

* j= 7, a[2]= 40 > a[7]= 30 => swap

10 20 30 70 90 60 50 40

**B4:** i= 3

* j= 4, a[3]= 70 < a[4]= 90 => không swap

10 20 30 70 90 60 50 40

* j= 5, a[3]= 70 > a[5]= 60 => swap

10 20 30 60 90 70 50 40

* j= 6, a[3]= 60 > a[6]= 50 => swap

10 20 30 50 90 70 60 40

* j= 7, a[3]= 50 > a[7]= 40 => swap

10 20 30 40 90 70 60 50

**B5:** i= 4

* j= 5, a[4]= 90 > a[5]= 70 => swap

10 20 30 40 70 90 60 50

* j= 6, a[4]= 70 > a[6]= 60 => swap

10 20 30 40 60 90 70 50

* j= 7, a[4]= 60 > a[7]= 50 => swap

10 20 30 40 50 90 70 60

**B6:** i= 5

* j= 6, a[5]= 90 > a[6]= 70 => swap

10 20 30 40 50 70 90 60

* j= 7, a[5]= 70 > a[7]= 60 => swap

10 20 30 40 50 60 90 70

**B7:** i= 6

* j= 7, a[6]= 90 > a[7]= 70 => swap

10 20 30 40 50 60 70 90

=>Ta có dãy đã sắp xếp là : 10 20 30 40 50 60 70 90

Độ phức tạp của thuật toán: **O(n2)**

**2.4 Dùng phương pháp xếp thứ tự BubbleSort, mô tả từng bước quá trình xếp thứ tự dãy số trên (không lập trình). Tín độ phức tạp của quá trình xếp thứ tự danh sách trên.**

40 70 20 60 90 10 50 30

**B1:** i= 0

* j= 7, xét a[7] =30 < a[6]= 50 => swap

40 70 20 60 90 10 30 50

* j= 6, xét a[6]=30 > a[5]= 10 => không swap

40 70 20 60 90 10 30 50

* j= 5, xét a[5]= 10 < a[4]= 90 => swap

40 70 20 60 10 90 30 50

* j= 4, xét a[4]= 10 < a[3]= 60 => swap

40 70 20 10 60 90 30 50

* j= 3, xét a[3]= 10 < a[2]= 20 => swap

40 70 10 20 60 90 30 50

* j= 2, xét a[2]= 10 < a[1]= 70 => swap

40 10 70 20 60 90 30 50

* j=1, xét a[1]= 10 < a[0] = 40 => swap

10 40 70 20 60 90 30 50

**B2:** i= 1

* j= 7, xét a[7] = 50 > a[6]= 30 => không swap

10 40 70 20 60 90 30 50

* j= 6, xét a[6]= 30 < a[5]= 90 => swap

10 40 70 20 60 30 90 50

* j= 5, xét a[5]= 30 < a[4]= 60 => swap

10 40 70 20 30 60 90 50

* j= 4, xét a[4]= 30 > a[3]= 20 => không swap

10 40 70 20 30 60 90 50

* j= 3, xét a[3]= 20 < a[2]= 70 => swap

10 40 20 70 30 60 90 50

* j= 2, xét a[2]= 20 < a[1]=40 => swap

10 20 40 70 30 60 90 50

**B3:** i= 2

* j= 7, xét a[7] =50 > a[6]= 90 => swap

10 20 40 70 30 60 50 90

* j= 6, xét a[6]=50 < a[5]= 60 => swap

10 20 40 70 30 50 60 90

* j= 5, xét a[5]= 50 > a[4]= 30 => không swap

10 20 40 70 30 50 60 90

* j= 4, xét a[4]= 30 < a[3]= 70 => swap

10 20 40 30 70 50 60 90

* j= 3, xét a[3]= 30 < a[2]= 40 => swap

10 20 30 40 70 50 60 90

**B4:** i= 3

* j= 7, xét a[7] =90 > a[6]= 60 => không swap

10 20 30 40 70 50 60 90

* j= 6, xét a[6]=60 > a[5]= 50 => không swap

10 20 30 40 70 50 60 90

* j= 5, xét a[5]= 50 < a[4]= 70 => swap

10 20 30 40 50 70 60 90

* j= 4, xét a[4]= 50 > a[3]= 40 => không swap

10 20 30 40 50 70 60 90

**B5:** i= 4

* j= 7, xét a[7] =90 > a[6]= 60 => không swap

10 20 30 40 50 70 60 90

* j= 6, xét a[6]=60 < a[5]= 70 => swap

10 20 30 40 50 60 70 90

* j= 5, xét a[5]= 60 > a[4]= 50 => không swap

10 20 30 40 50 60 70 90

**B6:** i= 5

* j= 7, xét a[7] =90 > a[6]= 70 => không swap

10 20 30 40 50 60 70 90

* j= 6, xét a[6]=70 > a[5]= 60 => không swap

10 20 30 40 50 60 70 90

**B7:** i= 5

* j= 7, xét a[7] =90 > a[6]= 70 => không swap

10 20 30 40 50 60 70 90

=> Ta có dãy đã sắp xếp: 10 20 30 40 50 60 70 90

Độ phức tạp của thuật toán: **O(n2)**

**2.5 Dùng phương pháp xếp thứ tự QuickSort, mô tả từng bước quá trình xếp thứ tự dãy số trên (không lập trình). Tín độ phức tạp của quá trình xếp thứ tự danh sách trên.**

40 70 20 60 90 10 50 30

**B1:** l(left) = 0, r(right)= 7

Chọn x= a[( l+ r) \2]= a[3]= 60

i= 0, a[0]= 40 < x= 60 => i++

i= 1, a[1]= 70 > x= 60 => dừng

j= 7, a[7]= 30 < x= 60 => dừng

* swap a[1] với a[7]

40 30 20 60 90 10 50 70

* i++, j-- => i= 2, j= 6

**B2:**

i= 2, a[2]= 20 < x= 60 => i++

i= 3, a[3]= 60 < x= 60 => dừng

j= 6, a[6]= 50 < x= 60 => dừng

* swap a[3] với a[6]

40 30 20 50 90 10 60 70

* i++, j-- => i= 4, j= 5

**B3:**

i= 4, a[4]= 90 > x= 60 => dừng

j= 5, a[5]= 10 < x= 60 => dừng

* swap a[4] với a[5]

40 30 20 50 10 90 60 70

* i++, j-- => i= 5, j= 4 => dừng( i> j)

Ta có 2 dãy: 40 30 20 50 (1)

10 90 60 70 (2)

1. 40 30 20 50

B1: l= 0, r= 3

Chọn x= a[1]= 30

i= 0, a[0]= 40 > x= 30 => dừng

j= 3, a[3]= 50 > x= 60 => j- -

j= 2, a[2]= 20 < x= 30 => dừng

* swap a[0] với a[2]

20 30 40 50 (1+)

* i++, j-- => i= 1, j= 1 ( dừng)

(2) 10 90 60 70

B1: l= 0, r= 4

Chọn x= a[1]= 90

i= 0, a[0]= 10 < x= 90 => i++

i= 1, a[1]= 90 < x= 90 => dừng

j= 4, a[4]= 70 < x= 90 => dừng

* swap a[1] với a[4]

10 70 60 90

* i++, j-- => i= 2, j= 3

B2:

i= 2, a[1]= 60 < x= 90 => i++

i= 3, a[3]= 90 < x= 90 => dừng

j= 3, a[3]= 90 > x= 90 => dừng

* swap a[3] với a[3]

10 70 60 90 (2+)

* dừng

Từ (1+) và (2+) ta có dãy: 20 30 40 50 10 70 60 90

**B1:** l(left) = 0, r(right)= 7

Chọn x= a[( l+ r) \2]= a[3]= 50

i= 0, a[0]= 20 < x= 50 => i++

i= 1, a[1]= 30 < x= 50 => i++

i= 2, a[2]= 40 < x= 50 => i++

i= 3, a[3]= 50 < x= 50 => dừng

j= 7, a[7]= 90 > x= 50 => j- -

j= 6, a[6]= 60 > x= 50 => j- -

j= 5, a[7]= 70 > x= 50 => j- -

j= 4, a[4]= 10 < x= 50 =>dừng

* swap a[3] với a[4]

20 30 40 10 50 70 60 90

* i++, j-- => i= 4, j= 3( dừng)

Ta có 2 dãy: 20 30 40 10 (1.1)

50 70 60 90 (2.1)

* 1. 20 30 40 10

B1: l= 0, r= 3

Chọn x= a[1]= 30

i= 0, a[0]= 20 < x= 30 => i++

i= 1, a[1]= 30 < x= 30 => dừng

j= 3, a[3]= 10 < x= 30 => dừng

* swap a[1] với a[3]

20 10 40 30 (1.1+)

* i++, j-- => i= 2, j= 2 ( dừng)

(2.1) 50 70 60 90

B1: l= 0, r= 3

Chọn x= a[1]= 70

i= 0, a[0]= 50 < x= 70 => i++

i= 1, a[1]= 70 < x= 70 => dừng

j= 3, a[3]= 90 > x= 70 => j- -

j= 2, a[2]= 60 < x= 70 => dừng

* swap a[1] với a[2]

50 60 70 90 (2.1+)

* i++, j-- => i= 2, j= 1 ( dừng)

Từ (1.1+) ta có 2 dãy: (1.1.1) 20 10

(1.1.2) 40 30

(1.1.1) 20 10

B1: l=0, r= 1=> x= a[0]= 20

i= 0, a[0]= 20 < x= 20 => dừng

j= 1, a[1]= 10 < x= 20 => dừng

* swap a[0] với a[1]

10 20 (1.1.1+)

* i++, j-- => i= 1, j= 0 ( dừng)

(1.1.2) 40 30

B1: l=0, r= 1=> x= a[0]= 40

i= 0, a[0]= 40 < x= 40 => dừng

j= 1, a[1]= 30 < x= 30 => dừng

* swap a[0] với a[1]

30 40 (1.1.2+)

* i++, j-- => i= 1, j= 0 ( dừng)

=>Từ (1.1.1+) và (1.1.2+) ta có dãy 10 20 30 40 (1.1.3)

=>Từ (1.1.3) và (2.1+) ta có dãy : 10 20 30 40 50 60 70 90

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

**2.6 Dùng phương pháp xếp thứ tự HeapSort, mô tả từng bước quá trình xếp thứ tự dãy số trên (không lập trình). Tín độ phức tạp của quá trình xếp thứ tự danh sách trên**

40 70 20 60 90 10 50 30

Dãy 1: a[0] -> a[3]: 40 70 20 60

Dãy 2: a[4]-> a[7]: 90 10 50 30

40 70 20 60 90 10 50 30

* Xét i=3, a[3] = 60

a[7] = 30

* Không swap => dừng

40 70 20 60 90 10 50 30

Xét i= 2, a[2] = 20

a[5] = 10

a[6] = 50

* Swap a[2] với a[6]

40 70 50 60 90 10 20 30

Xét i= 1, a[1]= 70

a[3]= 60

a[4]= 90

* Swap a[1] với a[4]

40 90 20 60 70 10 50 30

Xét i= 0, a[0]= 40

a[1]= 90

a[2]= 20

* Swap a[0] với a[1]

90 40 20 60 70 10 50 30

Đổi: 30 40 20 60 70 10 50 90

**B2:** 30 40 20 60 70 10 50

Dãy 1: a[0]-> a[2]: 30 40 20

Dãy 2: a[3]-> a[6]: 60 70 10 50

30 40 20 60 70 10 50

Xét i=2, a[2]= 20

a[5]= 10

a[6]= 50

* Swap a[2] với a[6]

30 40 50 60 70 10 20

Xét i= 1, a[1]= 40

a[3]= 60

a[4]= 70

* Swap a[1] với a[4]

30 70 50 60 40 10 20

Xét i= 0, a[0]= 30

a[1]= 70

a[2]= 50

* Swap a[0] với a[1]

70 30 50 60 40 10 20

Đổi: 20 30 50 60 40 10 70

**B3:** 20 30 50 60 40 10

Dãy 1: a[0]-> a[2]: 20 30 50

Dãy 2: a[3]->a[5]: 60 40 10

20 30 50 60 40 10

Xét i=2, a[2]= 50

a[5]= 10

* Không swap, dừng

20 30 50 60 40 10

Xét i=1, a[1]= 30

a[3]= 60

a[4]= 40

* Swap a[1] với a[3]

20 60 50 30 40 10

Xét i=0, a[0]= 20

a[1]= 60

a[2]= 50

* Swap a[0] với a[1]

60 20 50 30 40 10

Đổi: 10 20 50 30 40 60

**B4:** 10 20 50 30 40

Dãy 1: a[0]-> a[2]: 10 20

Dãy 2: a[3]-> a[4]: 50 30 40

10 20 50 30 40

Xét i= 1, a[1]= 20

a[3]= 30

a[4]= 40

* Swap a[1] với a[4]

10 40 50 30 20

Xét i= 0, a[0]= 10

a[1]= 40

a[2]= 50

* Swap a[0] với a[2]

50 40 10 30 20

Đổi: 20 40 10 30 50

**B5:** 20 40 10 30

Dãy 1: a[0]->a[1]: 20 40

Dãy 2: a[2]->a[3]: 10 30

20 40 10 30

Xét i= 1, a[1]= 40

A[3]= 30

* Không swap, dừng

20 40 10 30

Xét i=0, a[0]= 20

a[1]= 40

a[2]= 10

* Swap a[0] với a[1]

40 20 10 30

Đổi: 30 20 10 40

**B6:** 30 20 10

Dãy 1: a[0]: 30

Dãy 2: a[1]-> a[2]: 20 10

30 20 10

Xét i= 0, a[0]= 30

a[1]= 20

a[2]= 10

* Không swap, dừng

30 20 10

Đổi: 10 20 30

**B7:** 10 20

Dãy 1: a[0]= 10

Dãy 2: a[1]= 20

10 20

Xét i=0, a[0] = 10

a[1] = 20

* Không swap , dừng

10 20

Đổi: 10 20

* Ta có dãy đã sắp xếp : 10 20 30 40 50 60 70 90

Độ phức tạp : O(n log n)

**2.7 Sau khi xếp thú tự danh sách trên. Yêu cầu: Tính độ phức tạp của quá trình tìm kiếm giá trị 90 trong danh sách trên cho cả hai thuật toán tìm kiếm tuần tự và tìm kiếm nhị phân**

- Tìm kiếm tuần tự: O(n)

- Tìm kiếm nhị phân: O(log n)