**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.

Trả lời:

0 1 2 3 4 5 6 7

40 70 20 60 90 10 50 30

Gỉa sử vị trí a[0] đã có thứ tự, ta xét phần tử tiếp theo

a[1]= 70 > a[0]=40 -> chèn 70 sau 40

40 70 20 60 90 10 50 30

Xét phần tử tiếp theo a[2]=20

a[2] < a[0] -> chèn 20 trước 40

20 40 70 60 90 10 50 30

Xét phần tử tiếp theo a[3]=60

a[3] < a[2] = 70 -> chèn 60 trước 70

20 40 60 70 90 10 50 30

Xét phần tử tiếp theo a[4]=90

a[4] > a[3] = 70 -> chèn 90 sau 70

20 40 60 70 90 10 50 30

Xét phần tử tiếp theo a[5]=10

a[5] < a[0] = 20 -> chèn 10 trước 20

10 20 40 60 70 90 50 30

Xét phần tử tiếp theo a[6]=50

a[6] < a[3] = 60 -> chèn 50 trước 60

10 20 40 50 60 70 90 30

Xét phần tử tiếp theo a[7]=30

a[7] < a[2] = 40 -> chèn 30 trước 40

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.

0 1 2 3 4 5 6 7

40 70 20 60 90 10 50 30

Tìm min từ 0 -> 7, ta được 10 ( a[5]), swap (a[5] , a[0])

10 40 70 20 60 90 50 30

Tìm min từ 1 -> 7, ta được 20 ( a[3]), swap (a[3] , a[1])

10 20 40 70 60 90 50 30

Tìm min từ 2 -> 7, ta được 30 ( a[7]), swap (a[7] , a[2])

10 20 30 40 70 60 90 50

Tìm min từ 3 -> 7, ta được 40 ( a[3]), giữ nguyên vị trí)

10 20 30 40 70 60 90 50

Tìm min từ 4 -> 7, ta được 50 ( a[7]), swap (a[7] , a[4])

10 20 30 40 50 70 60 90

Tìm min từ 5 -> 7, ta được 60 ( a[6]), swap (a[6] , a[5])

10 20 30 40 50 60 70 90

Tìm min từ 6 -> 7, ta được 70 ( a[6]), giữ nguyên vị trí)

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.

0 1 2 3 4 5 6 7

40 70 20 60 90 10 50 30

Tại vị trí a[0] = 40 so sánh với a[1] = 70 thấy a[0]<a[1] không đổi

40 70 20 60 90 10 50 30

Tại vị trí a[0] = 40 so sánh với a[2] = 20 thấy a[0]>a[2] đổi vị trí

20 70 40 60 90 10 50 30

Tại vị trí a[0] = 20 so sánh với a[3] = 60 thấy a[0]<a[3] không đổi

20 70 40 60 90 10 50 30

Tại vị trí a[0] = 20 so sánh với a[4] = 90 thấy a[0]<a[4] không đổi

20 70 40 60 90 10 50 30

Tại vị trí a[0] = 20 so sánh với a[5] = 10 thấy a[0]>a[5] đổi vị trí

10 70 40 60 90 20 50 30

Tại vị trí a[0] = 10 so sánh với a[6] = 50 thấy a[0]<a[6] không đổi

10 70 40 60 90 20 50 30

Tại vị trí a[0] = 10 so sánh với a[7] = 30 thấy a[0]<a[7] không đổi

10 70 40 60 90 20 50 30

Ta được phần tử bé nhất trong mảng, tương tự lặp lại cho các phần tử sau.

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.

0 1 2 3 4 5 6 7

40 70 20 60 90 10 50 30

Duyệt từ 7:

a[7]<a[6] swap(a[7],a[6])

40 70 20 60 90 10 30 50

a[6]>a[5] không đổi chổ, xét a[5]

40 70 20 60 90 10 30 50

a[5]<a[4] swap(a[5],a[4])

40 70 20 60 10 90 30 50

a[4]<a[3] swap(a[4],a[3])

40 70 20 10 60 90 30 50

a[4]<a[3] swap(a4],a[3])

40 70 20 10 60 90 30 50

a[3]<a[2] swap(a[3],a[2])

40 70 10 20 60 90 30 50

a[2]<a[1] swap(a[2],a[1])

40 10 70 20 60 90 30 50

a[1]<a[0] swap(a[1],a[0])

10 40 70 20 60 90 30 50

Ta đã đưa được phần tử bé nhất lên đầu mảng. Tiếp theo ta xét tiếp, duyệt từ 7

a[7]>a[6] Không đổi xét a[6]

10 40 70 20 60 90 30 50

a[6]<a[5] swap(a[6],a[5])

10 40 70 20 60 30 90 50

a[5]<a[4] swap(a[5],a[4])

10 40 70 20 30 60 90 50

a[4]>a[3] không đổi, xét a[3]

10 40 70 20 30 60 90 50

a[3]<a[2] swap(a[3],a[2])

10 40 20 70 30 60 90 50

a[2]<a[1] swap(a[2],a[1])

10 20 40 70 30 60 90 50

Tương tự ta lặp lại cho các phần tử sau

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.

Left-> 0 1 2 3 4 5 6 7 <-right

40 70 20 60 90 10 50 30

Chọn x=a[(left+right)/2]=a[3]=60

40 70 20 60 90 10 50 30

i=0 j=7

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

a[i]=a[1]=70>x, dừng i

a[j]=a[7]=30<x, dừng j

swap a[i] và a[j](a[1] và a[7]), i++, j—

40 30 20 60 90 10 50 70

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

a[i]=a[3]=60=x, dừng i

a[j]=a[6]=50<x,dừng j

swap a[i] và a[j](a[3] và a[6]), i++, j—

40 30 20 50 90 10 60 70

a[i]=a[4]=90>x dừng i

a[j]=a[5]=10>x dừng j

swap a[i] và a[j](a[4] và a[5]), i++, j—

40 30 20 50 10 90 60 70

a[i]=a[5] ; a[j]=a[4] i>j dừng

40 30 20 50 10 | 90 60 70

=>Ta đã phân hoạch ra được thành 2 dãy con như sau

Lớn hơn hoặc bằng x: 90 60 70

Nhỏ hơn x: 40 30 20 50 10

Tương tự như vậy ta đệ quy cho các dãy con cho đến khi dãy con chỉ còn 1 phần tử

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.

0 1 2 3 4 5 6 7

a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7]

40 70 20 60 90 10 50 30

Chi dãy trên thành 2 đoạn, bao gồm:

Nữa dãy bên trái chứa các phần tử sau a[0],…., a[(n/2)-1]: 40, 70, 20,60

Nữa dãy bên phải chứa các phần tử sau a[n/2],…., a[n-1]: 90, 10, 50,30

40 70 20 60 | 90 10 50 30

0 1 2 3 | 4 5 6 7

Tại vị trí cuối cùng của nữa dãy con bên trái i=3, so sánh a[i] =a[3] = 60 với phần tử tại vị trí 2\*i+1 = 7

Giá trị lớn nhất của a[3], a[7] là a[3] = 60. Ta giữ nguyên vị trí

40 70 20 60 | 90 10 50 30

Tiếp tục giảm i xuống 1 giá trị (i=1), và so sánh a[2], a[5], a[6]

40 70 20 60 | 90 10 50 30

Giá trị lớn nhất của 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

Tiếp tục giảm i xuống 1 giá trị (i=1), và so sánh a[1], a[3], a[4]

40 70 50 60 | 90 10 20 30

Giá trị lớn nhất của a[1], a[3], a[4] là a[4] = 90. Thực hiện hoán vị a[1] và a[4]

40 90 50 60 | 70 10 20 30

Tiếp tục giảm i xuống 1 giá trị (i=0). So sánh a[0], a[1], a[2]

40 90 50 60 | 70 10 20 30

Giá trị lớn nhất của a[0], a[1], a[2] là a[1] = 90. Thực hiện hoán vị a[0] và a[1]

90 40 50 60 | 70 10 20 30

***Xét lại tính lan truyền tại vị trí a[1]***, sau khi a[1] nhận giá 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

Giá trị lớn nhất của a[1], a[3], a[4] là a[4] = 70. Thực hiện hoán vị a[1] và a[4]

Việc tạo Heap (Heap Max) ban đầu hoàn tất. Ta được một Heap sau:

90 70 50 60 | 40 10 20 30

**BƯỚC 2:** Hoán vị phần tử a[0] và phần tử cuối Heap đang xét. Ta có kết quả sau:

30 70 50 60 40 10 20 90

**BƯỚC 3:** Trong dãy đang xét, giới hạn phần tử cuối dãy. Ta được dãy sau:

30 70 50 60 40 10 20 | 90

Tạo Heap ban đầu lại cho dãy các phần tử đang xét từ a[0], a[1], …,a[6]

Trong trường hợp này thực chất chỉ xét lại vị trí a[0] (và sự lan truyền nếu có), các vị trí còn ại từ a[1],…, a[5] đã thỏa tính chất Heap trước đó.

**BƯỚC 4:** Sau khi dãy từ a[0], a[1], …., a[6] là một Heap, hoán vị a[0] và a[6]. Tiếp túc xét lại dãy từ a[0] đến a[5]….*Lặp lại* bước này cho đến khi danh sách được xếp thứ tự tăng dần.

Độ phức tạp của thuật toán: **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

0 1 2 3 4 5 6 7

10 20 30 40 50 60 70 90

Độ phức tạp của thuật toán tìm kếm tuần tự của giá trị 90 là 8

Độ phức tạp của thuật toán tìm kếm nhị phân của giá trị 90 là log(8)