**Câu 2.4 - Chương 3:** 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: phương pháp xếp thứ tự BubbleSort, mô tả từng bước quá trình xếp thứ tự dãy số trên (không lập trình). Tính độ phức tạp của quá trình xếp thứ tự danh sách trên.

**Giải:**

**Phần tử:**  40 70 20 60 90 10 50 30

**Vị trí:**  0 1 2 3 4 5 6 7

* **j = 7**, vì a[j] ( a[7] ) bé hơn a[j-1] ( a[6] )nên đổi chỗ giá trị giữa a[7] và a[6].

Phần tử: 40 70 20 60 90 10 **30 50**

Vị trí: 0 1 2 3 4 5 6 7

* **j = 6**, vì a[6] lớn hơn a[5] nên không đổi chỗ.

Phần tử: 40 70 20 60 90 **10 30** 50

Vị trí: 0 1 2 3 4 5 6 7

* **j = 5**, vì a[5] bé hơn a[4] nên đổi chỗ giá trị giữa a[5] và a[4].

Phần tử: 40 70 20 60 **10 90** 30 50

Vị trí: 0 1 2 3 4 5 6 7

* **j = 4**, vì a[4] bé hơn a[3] nên đổi chỗ giá trị giữa a[4] và a[3].

Phần tử: 40 70 20 **10** **60** 9030 50

Vị trí: 0 1 2 3 4 5 6 7

* **j = 3**, vì a[3] lớn hơn a[2] nên đổi chỗ giá trị giữa a[3] và a[2].

Phần tử: 40 70 **10 20** 60 9030 50

Vị trí: 0 1 2 3 4 5 6 7

* **j = 2**, vì a[2] bé hơn a[1] nên đổi chỗ giá trị giữa a[2] và a[1].

Phần tử: 40 **10 70** 20 60 9030 50

Vị trí: 0 1 2 3 4 5 6 7

* **j = 1**, vì a[1] bé hơn a[0] nên đổi chỗ giá trị giữa a[1] và a[0].

Phần tử: **10 40** 70 20 60 9030 50

Vị trí: 0 1 2 3 4 5 6 7

* **j = 7**, vì a[7] lớn hơn a[6] nên không đổi chỗ.

Phần tử: 10 40 70 20 60 90 **30 50**

Vị trí: 0 1 2 3 4 5 6 7

* **j = 6**, vì a[6] bé hơn a[5] nên đổi chỗ giá trị giữa a[6] và a[5].

Phần tử: 10 40 70 20 60 **30 90** 50

Vị trí: 0 1 2 3 4 5 6 7

* **j = 5**, vì a[5] bé hơn a[4] nên đổi chỗ giá trị giữa a[5] và a[4].

Phần tử: 10 40 70 20 **30** **60** 90 50

Vị trí: 0 1 2 3 4 5 6 7

* **j = 4**, vì a[4] lớn hơn a[3] nên không đổi chỗ.

Phần tử: 10 40 70 **20 30** 6090 50

Vị trí: 0 1 2 3 4 5 6 7

* **j = 3**, vì a[3] bé hơn a[2] nên đổi chỗ giá trị giữa a[3] và a[2].

Phần tử: 10 40 **20 70** 30 6090 50

Vị trí: 0 1 2 3 4 5 6 7

* **j = 2**, vì a[2] bé hơn a[1] nên đổi chỗ giá trị giữa a[2] và a[1].

Phần tử: 10 **20** **40** 70 30 6090 50

Vị trí: 0 1 2 3 4 5 6 7

* **j = 1**, vì a[1] lớn hơn a[0] nên không đổi chỗ.

Phần tử: **10 20** 40 70 30 6090 50

Vị trí: 0 1 2 3 4 5 6 7

* **j = 7**, vì a[7] bé hơn a[6] nên đổi chỗ giá trị giữa a[7] và a[6].

Phần tử: 10 20 40 70 30 60 **50 90**

Vị trí: 0 1 2 3 4 5 6 7

* **j = 6**, vì a[6] bé hơn a[5] nên đổi chỗ giá trị giữa a[6] và a[5].

Phần tử: 10 20 40 70 30 **50 60** 90

Vị trí: 0 1 2 3 4 5 6 7

* **j = 5**, vì a[5] lớn hơn a[4] nên không đổi chỗ.

Phần tử: 10 20 40 70 **30 50** 6090

Vị trí: 0 1 2 3 4 5 6 7

* **j = 4**, vì a[4] bé hơn a[3] nên đổi chỗ giá trị giữa a[4] và a[3].

Phần tử: 10 20 40 **30** **70** 50 6090

Vị trí: 0 1 2 3 4 5 6 7

* **j = 3**, vì a[3] bé hơn a[2] nên đổi chỗ giá trị giữa a[3] và a[2].

Phần tử: 10 20 **30** **40** 7050 6090

Vị trí: 0 1 2 3 4 5 6 7

* **j = 2**, vì a[2] lớn hơn a[1] nên không đổi chỗ.

Phần tử: 10 **20** **30** 40 7050 6090

Vị trí: 0 1 2 3 4 5 6 7

* **j = 7**, vì a[7] lớn hơn a[6] nên không đổi chỗ.

Phần tử: 10 20 30 40 7050 **60 90**

Vị trí: 0 1 2 3 4 5 6 7

* **j = 6**, vì a[6] lớn hơn a[5] nên không đổi chỗ.

Phần tử: 10 20 30 40 70 **50 60** 90

Vị trí: 0 1 2 3 4 5 6 7

* **j = 5**, vì a[5] bé hơn a[4] nên đổi chỗ giá trị giữa a[5] và a[4].

Phần tử: 10 20 30 40 **50 70** 6090

Vị trí: 0 1 2 3 4 5 6 7

* **j = 4**, vì a[4] bé hơn a[3] nên đổi chỗ giá trị giữa a[4] và a[3].

Phần tử: 10 20 30 **40** **50** 706090

Vị trí: 0 1 2 3 4 5 6 7

* **j = 7**, vì a[7] lớn hơn a[6] nên không đổi chỗ.

Phần tử: 10 20 30 40 5070 **60 90**

Vị trí: 0 1 2 3 4 5 6 7

* **j = 6**, vì a[6] bé hơn a[5] nên đổi chỗ giá trị giữa a[6] và a[5].

Phần tử: 10 20 30 40 50 **60 70** 90

Vị trí: 0 1 2 3 4 5 6 7

* **j = 5**, vì a[5] lớn hơn a[4] nên không đổi chỗ.

Phần tử: 10 20 30 40 **50 60** 7090

Vị trí: 0 1 2 3 4 5 6 7

- Độ phức tạp của thuật toán: O()

**Câu 2.7 - Chương 4:** Viết thủ tục duyệt cây theo thứ tự LNR (dùng stack):

**Giải:**

- Gọi thực thi: duyetLNR(root);

void duyetLNR(Node \*p)

{

if (p != NULL)

{

duyetLNR(p->left);

cout << p->info<<” “;

duyetLNR(p->right);

}

}