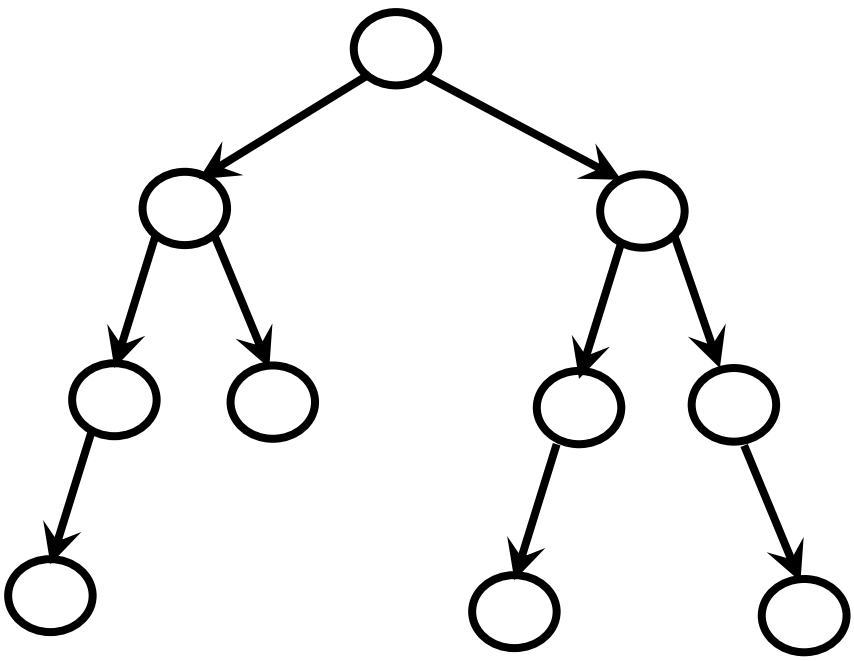
Chương 7
CẤU TRÚC DỮ LIỆU
CÂY NHỊ PHÂN

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

1. HÌNH ẢNH CÂY NHỊ PHÂN



TS. Nguyễn Tấn Trần Minh Khang

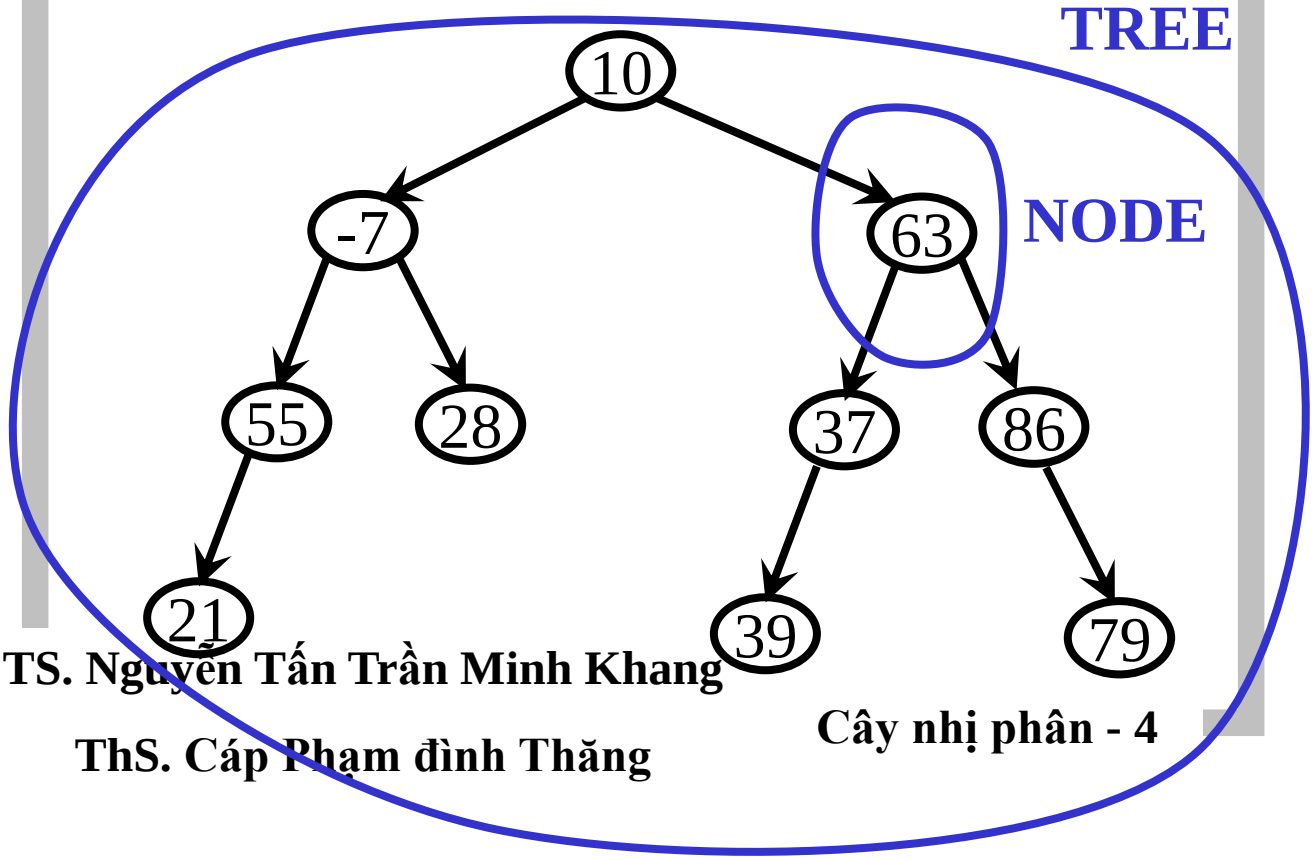
ThS. Cáp Phạm đình Thăng

2. KHÁI NIỆM CÂY NHỊ PHÂN

- Cây nhị phân là một cây thỏa điều kiện: mọi node trong cây có tối đa 2 node con.

3. CẤU TRÚC DỮ LIỆU CỦA CÂY NHỊ PHÂN

```
1. struct node
2. {
3.     KDL info;
4.     struct node *pLeft;
5.     struct node *pRight;
6. };
7. typedef struct node NODE;
8. typedef NODE *TREE;
```



3. CẤU TRÚC DỮ LIỆU CỦA CÂY NHỊ PHÂN

- Ví dụ 1: Hãy khai báo cấu trúc dữ liệu cho cây nhị phân các số nguyên.
- Cấu trúc dữ liệu.

```
1. struct node
2. {
3.     int info;
4.     struct node *pLeft;
5.     struct node *pRight;
6. };
7. typedef struct node NODE;
8. typedef NODE *TREE;
```

3. CẤU TRÚC DỮ LIỆU CỦA CÂY NHỊ PHÂN

- Ví dụ 2: Hãy khai báo cấu trúc dữ liệu cho cây nhị phân các phân số.
- Cấu trúc dữ liệu.

```
10. struct phanso
11. {
12.     int tu;
13.     int mau;
14. };
15. typedef struct phanso PHANSO;
16. struct node
17. {
18.     PHANSO info;
19.     struct node *pLeft;
20.     struct node *pRight;
21. };
22. typedef struct node NODE;
23. typedef NODE *TREE;
```

3. CẤU TRÚC DỮ LIỆU CỦA CÂY NHỊ PHÂN

- Ví dụ 3: Hãy khai báo cấu trúc dữ liệu cho cây nhị phân các số thực.
- Cấu trúc dữ liệu.

```
1. struct node
2. {
3.     float info;
4.     struct node *pLeft;
5.     struct node *pRight;
6. };
7. typedef struct node NODE;
8. typedef NODE *TREE;
```

3. CẤU TRÚC DỮ LIỆU CỦA CÂY NHỊ PHÂN

- Ví dụ 4: Hãy khai báo CTDL cho CNP tọa độ các điểm trong mặt phẳng Oxy.
- Cấu trúc dữ liệu.

```
10. struct diem
11. {
12.     float x;
13.     float y;
14. };
15. typedef struct diem DIEM;
16. struct node
17. {
18.     DIEM info;
19.     struct node *pLeft;
20.     struct node *pRight;
21. };
22. typedef struct node NODE;
23. typedef NODE *TREE;
```

4. KHỞI TẠO CÂY NHỊ PHÂN

- Khái niệm: Khởi tạo cây nhị phân là tạo ra một cây nhị phân rỗng không chứa node nào hết.
- Định nghĩa hàm

```
1. void Init(TREE &t)
2. {
3.     t = NULL;
4. }
```

5. TẠO NODE CHO CÂY NHỊ PHÂN

- Khái niệm: Tạo node cho cây nhị phân là xin cấp phát bộ nhớ có kích thước bằng kích thước của KDL NODE để chứa thông tin biết trước.
- **Định nghĩa hàm**

```
10. NODE* GetNode(KDL x)
11. {
12.     NODE *p = new NODE;
13.     if(p==NULL)
14.         return NULL;
15.     p->info = x;
16.     p->pLeft = NULL;
17.     p->pRight= NULL;
18.     return p;
19. }
```

5. TẠO NODE CHO CÂY PHÂN

- Ví dụ 1: Định nghĩa hàm tạo node cho cây nhị phân các số nguyên.
- **Định nghĩa hàm**

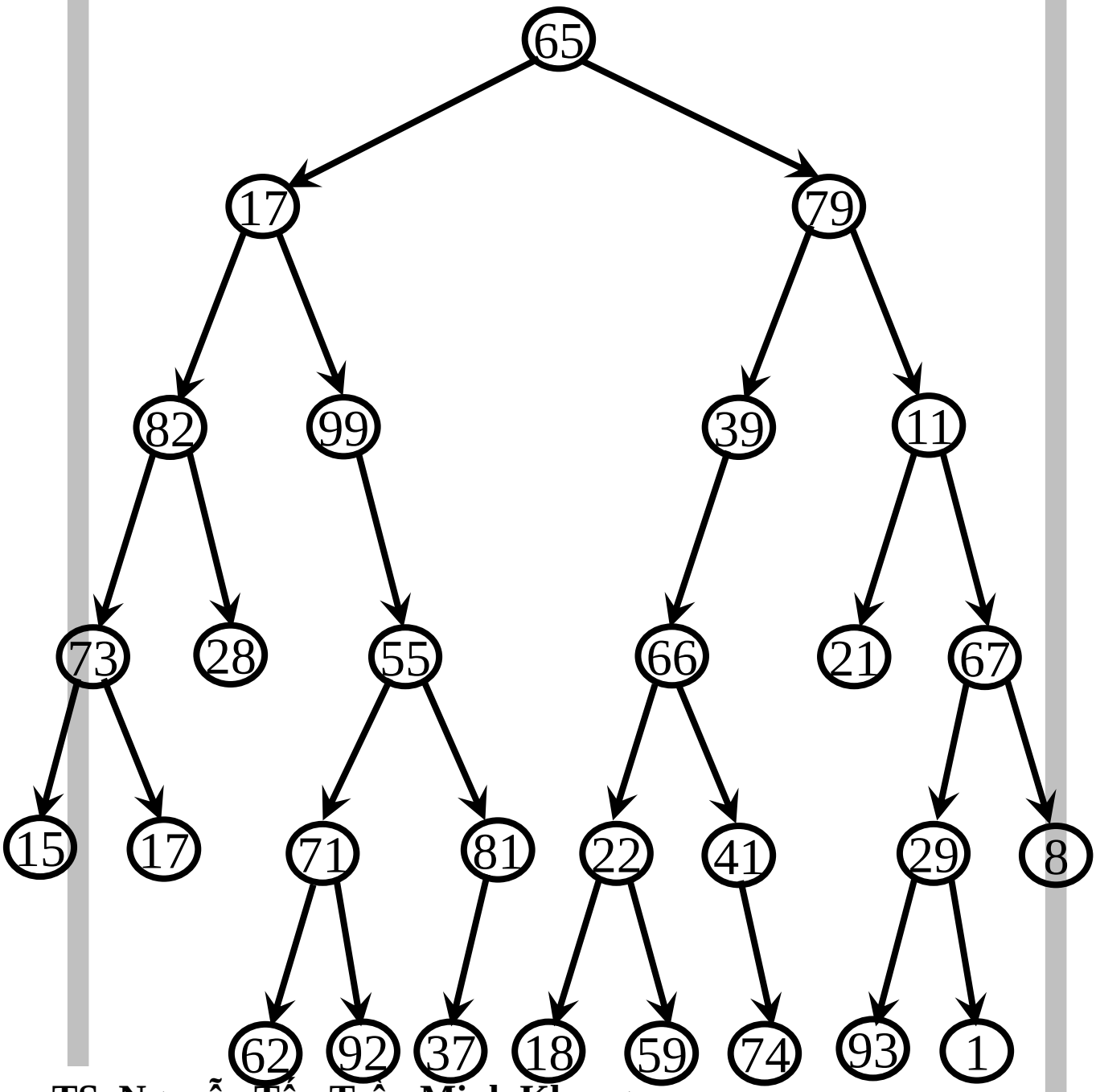
```
10. NODE* GetNode(int x)
11. {
12.     NODE *p = new NODE;
13.     if(p==NULL)
14.         return NULL;
15.     p->info = x;
16.     p->pLeft = NULL;
17.     p->pRight= NULL;
18.     return p;
19. }
```

5. TẠO NODE CHO CÂY PHÂN

- Ví dụ 2: Định nghĩa hàm tạo node cho cây nhị phân các phân số.
- **Định nghĩa hàm**

```
10. NODE* GetNode(PHANSO x)
11. {
12.     NODE *p = new NODE;
13.     if(p==NULL)
14.         return NULL;
15.     p->info = x;
16.     p->pLeft = NULL;
17.     p->pRight= NULL;
18.     return p;
19. }
```

6. NHÌN CÂY NHỊ PHÂN DƯỚI CON MẮT ĐỆ QUY



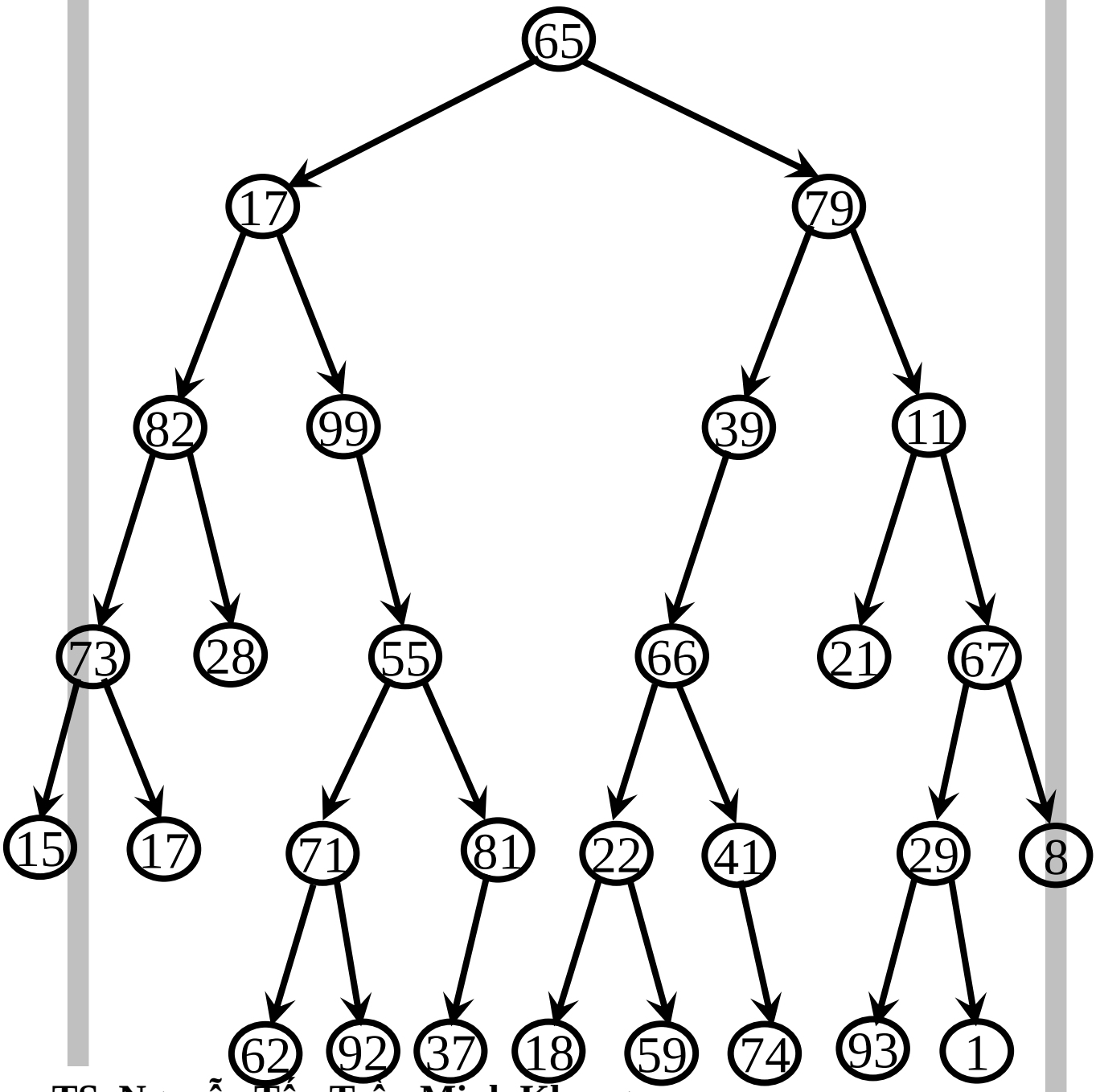
7. DUYỆT CÂY NHỊ PHÂN

- Khái niệm: Duyệt cây nhị phân là thăm qua tất cả các node trong cây mỗi node một lần.
- Các phương pháp duyệt cây:
  - Phương pháp **LNR** (Left Node Right).
  - Phương pháp **RNL** (Right Node Left).
  - Phương pháp **NLR** (Node Left Right).
  - Phương pháp **NRL** (Node Right Left).
  - Phương pháp **LRN** (Left Right Node).
  - Phương pháp **RLN** (Right Left Node).

7. DUYỆT CÂY NHỊ PHÂN

- Duyệt cây theo phương pháp **LNR** (Left Node Right) là: duyệt cây con trái trước, sau đó duyệt tới node gốc trong cây và cuối cùng là duyệt cây con phải. Cách thức duyệt cây con trái và duyệt cây con phải cũng giống như cách thức duyệt cây cha.

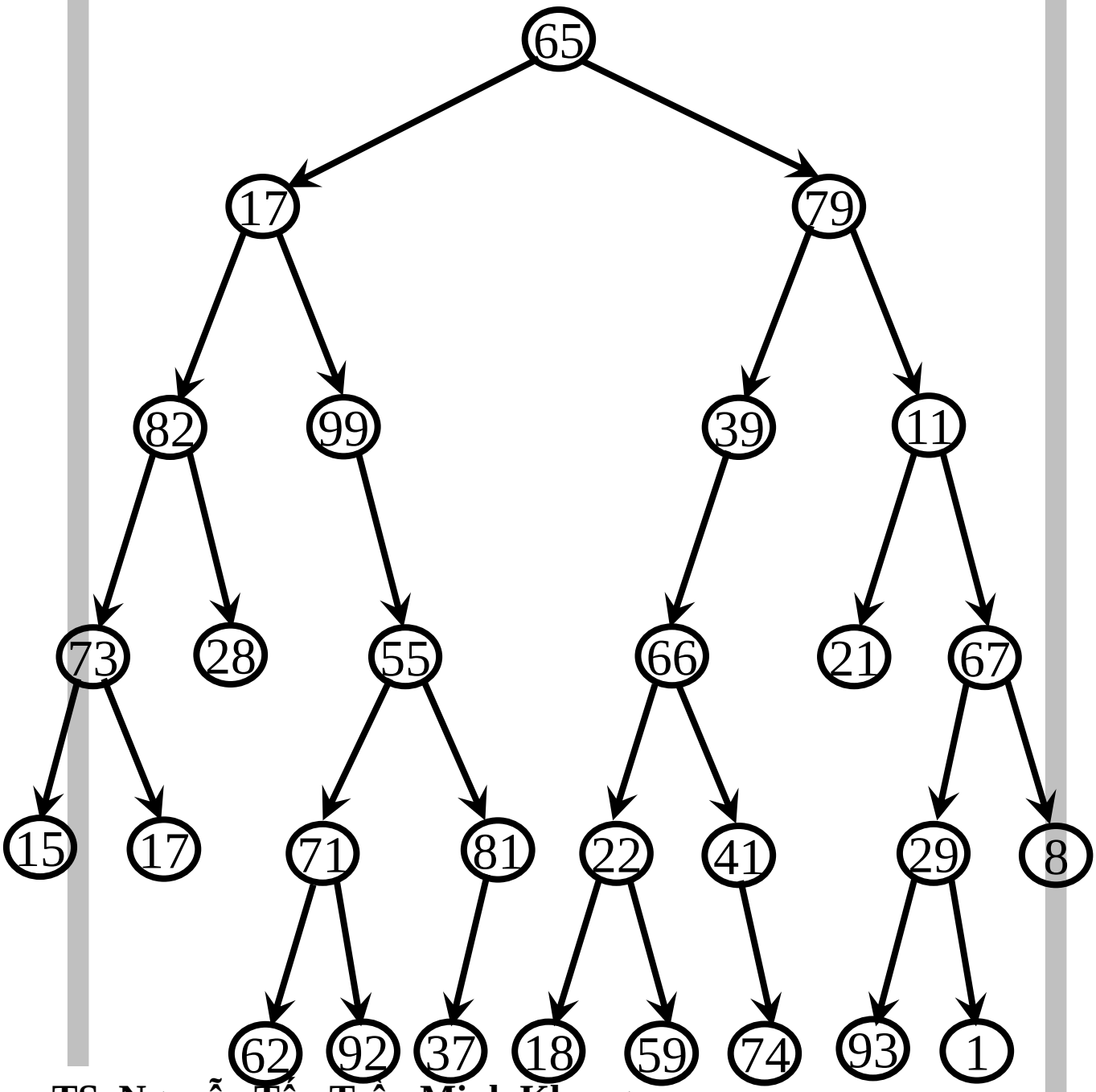
LNR


65
17
79
82
99
39
11
73
28
55
66
21
67
15
17
71
81
22
41
29
8
62
92
37
18
59
74
93
1

7. DUYỆT CÂY NHỊ PHÂN

- Duyệt cây theo phương pháp **RNL** (Right Node Left) là: duyệt cây con phải trước, sau đó duyệt tới node gốc trong cây và cuối cùng là duyệt cây con trái. Cách thức duyệt cây con phải và duyệt cây con trái cũng giống như cách thức duyệt cây cha.

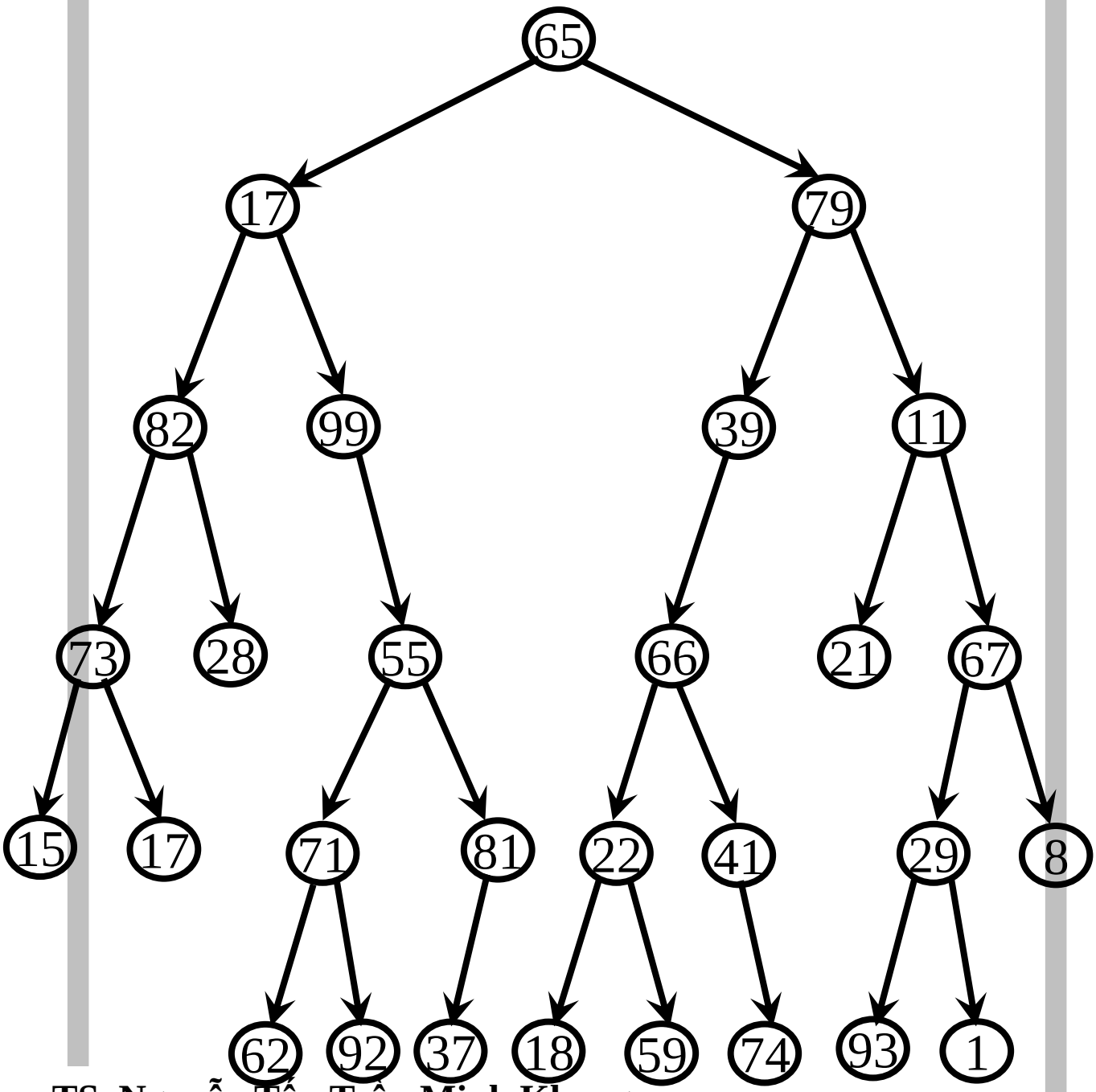
RNL


65
17
79
82
99
39
11
73
28
55
66
21
67
15
17
71
81
22
41
29
8
62
92
37
18
59
74
93
1

7. DUYỆT CÂY NHỊ PHÂN

- Duyệt cây thep phương pháp **NLR** (Node Left Right) là: duyệt node gốc trong cây trước, sau đó duyệt tới cây con trái và cuối cùng là duyệt cây con phải. Cách thức duyệt cây con trái và duyệt cây con phải cũng giống như cách thức duyệt cây cha.

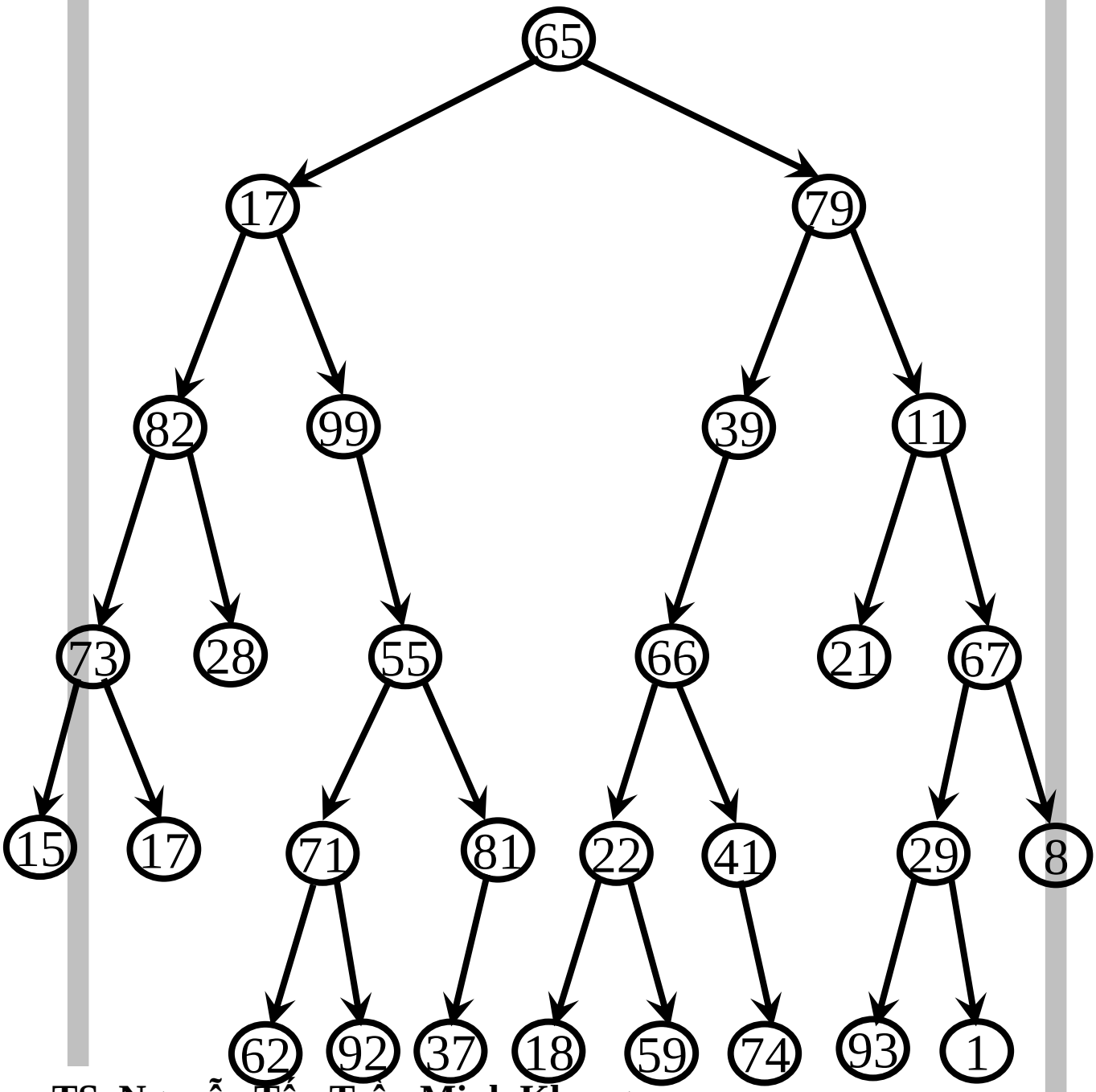
NLR


65
17
79
82
99
39
11
73
28
55
66
21
67
15
17
71
81
22
41
29
8
62
92
37
18
59
74
93
1

7. DUYỆT CÂY NHỊ PHÂN

- Duyệt cây thep phương pháp **NRL** (Node Right Left) là: duyệt node gốc trong cây trước, sau đó duyệt tới cây con phải và cuối cùng là duyệt cây con trái. Cách thức duyệt cây con phải và duyệt cây con trái cũng giống như cách thức duyệt cây cha.

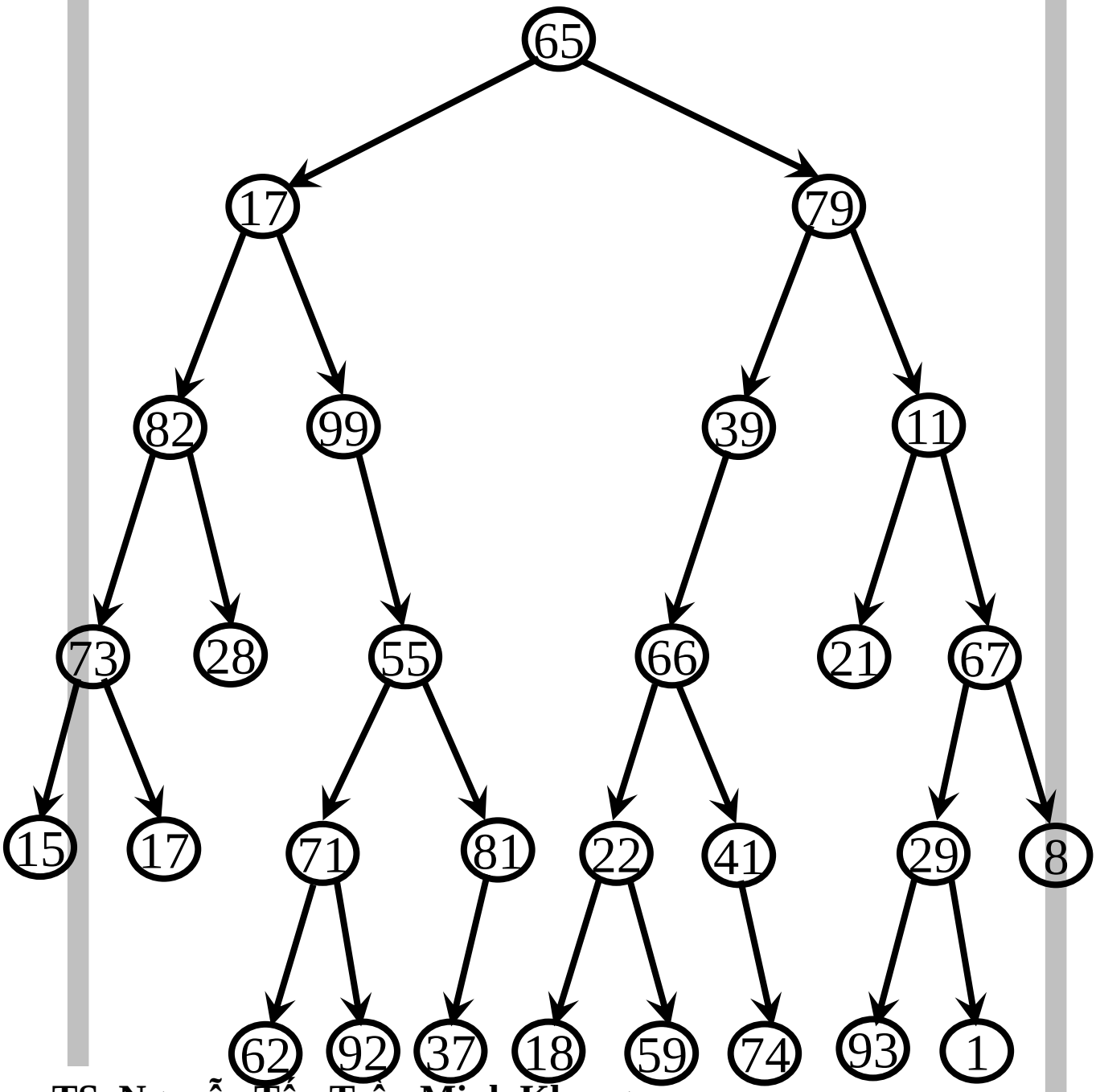
NRL


65
17
79
82
99
39
11
73
28
55
66
21
67
15
17
71
81
22
41
29
8
62
92
37
18
59
74
93
1

7. DUYỆT CÂY NHỊ PHÂN

- Duyệt cây thep phương pháp **LRN** (Left Right Node) là: duyệt cây con trái trước, sau đó duyệt tới cây con phải và cuối cùng duyệt tới node gốc trong cây. Cách thức duyệt cây con trái và duyệt cây con phải cũng giống như cách thức duyệt cây cha.

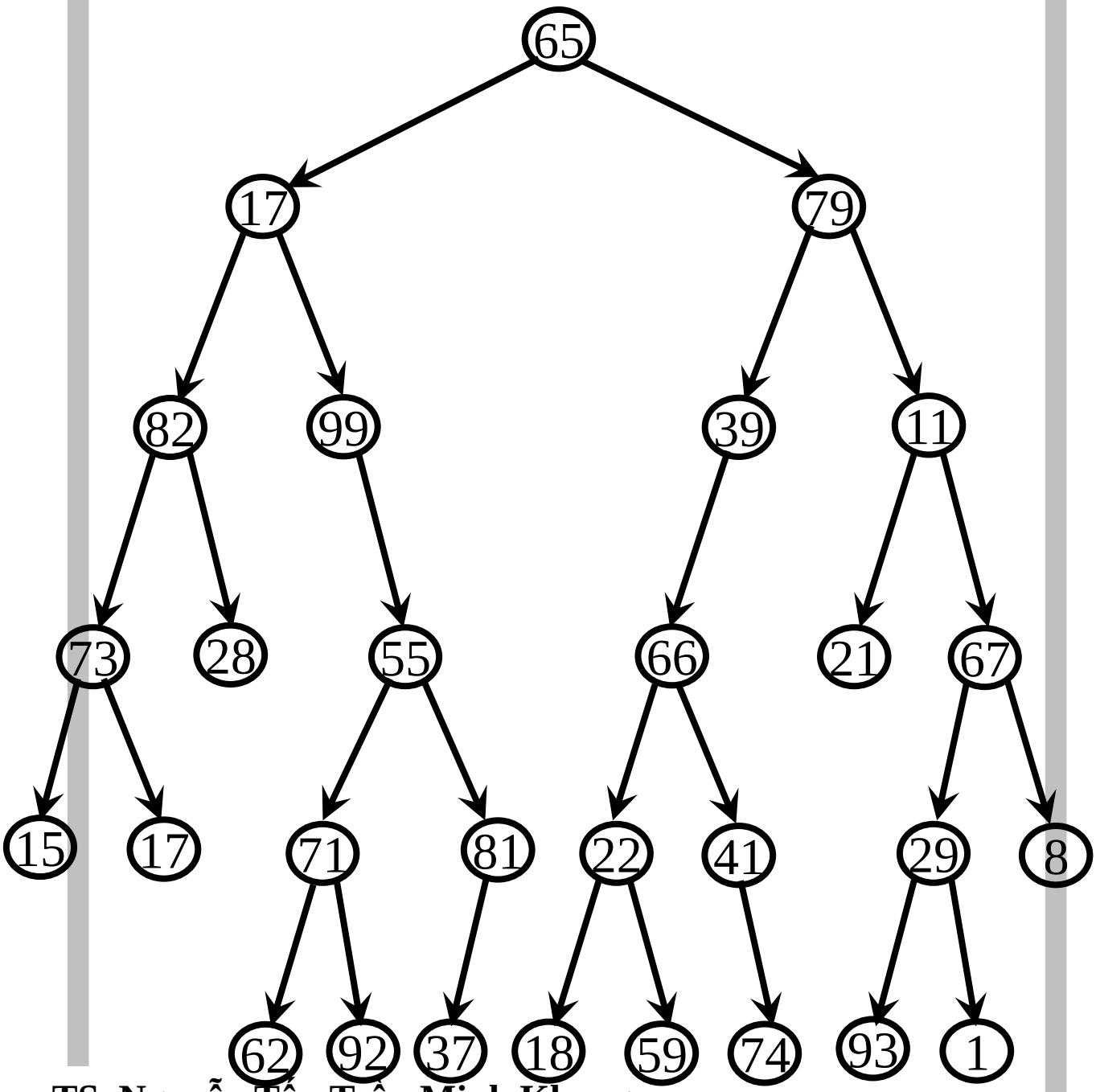
LRN


65
17
79
82
99
39
11
73
28
55
66
21
67
15
17
71
81
22
41
29
8
62
92
37
18
59
74
93
1

7. DUYỆT CÂY NHỊ PHÂN

- Duyệt cây thep phương pháp **RLN** (Right Left Node) là: duyệt cây con phải trước, sau đó duyệt tới cây con trái và cuối cùng duyệt tới node gốc trong cây. Cách thức duyệt cây con phải và duyệt cây con trái cũng giống như cách thức duyệt cây cha.

RLN


65
17
79
82
99
39
11
73
28
55
66
21
67
15
17
71
81
22
41
29
8
62
92
37
18
59
74
93
1

8. HÀM CÀI ĐẶT DUYỆT CÂY

- Bài toán: Hãy định nghĩa tất cả các hàm duyệt và xuất cây nhị phân các số nguyên bằng 6 phương pháp.
- Cấu trúc dữ liệu

```
1. struct node
2. {
3.     int info;
4.     struct node *pLeft;
5.     struct node *pRight;
6. };
7. typedef struct node NODE;
8. typedef NODE* TREE;
```

8. HÀM CÀI ĐẶT DUYỆT CÂY

- Định nghĩa hàm duyệt và xuất cây theo phương pháp LNR

```
1. void Xuat(TREE t)
2. {
3.     if(t==NULL)
4.         return;
5.     Xuat(t->pLeft);
6.     printf("%4d", t->info);
7.     Xuat(t->pRight);
8. }
```

8. HÀM CÀI ĐẶT DUYỆT CÂY

- Định nghĩa hàm duyệt và xuất cây theo phương pháp RNL

```
1. void Xuat(TREE t)
2. {
3.     if(t==NULL)
4.         return;
5.     Xuat(t->pRight);
6.     printf("%4d", t->info);
7.     Xuat(t->pLeft);
8. }
```

8. HÀM CÀI ĐẶT DUYỆT CÂY

- Định nghĩa hàm duyệt và xuất cây theo phương pháp NLR.

```
1. void Xuat(TREE t)
2. {
3.     if(t==NULL)
4.         return;
5.     printf("%4d", t->info);
6.     Xuat(t->pLeft);
7.     Xuat(t->pRight);
8. }
```

8. HÀM CÀI ĐẶT DUYỆT CÂY

- Định nghĩa hàm duyệt và xuất cây theo phương pháp NRL.

```
1. void Xuat(TREE t)
2. {
3.     if(t==NULL)
4.         return;
5.     printf("%4d", t->info);
6.     Xuat(t->pRight);
7.     Xuat(t->pLeft);
8. }
```

8. HÀM CÀI ĐẶT DUYỆT CÂY

- Định nghĩa hàm duyệt và xuất cây theo phương pháp LRN.

```
1. void Xuat(TREE t)
2. {
3.     if(t==NULL)
4.         return;
5.     Xuat(t->pLeft);
6.     Xuat(t->pRight);
7.     printf("%4d", t->info);
8. }
```

8. HÀM CÀI ĐẶT DUYỆT CÂY

- Định nghĩa hàm duyệt và xuất cây theo phương pháp RLN.

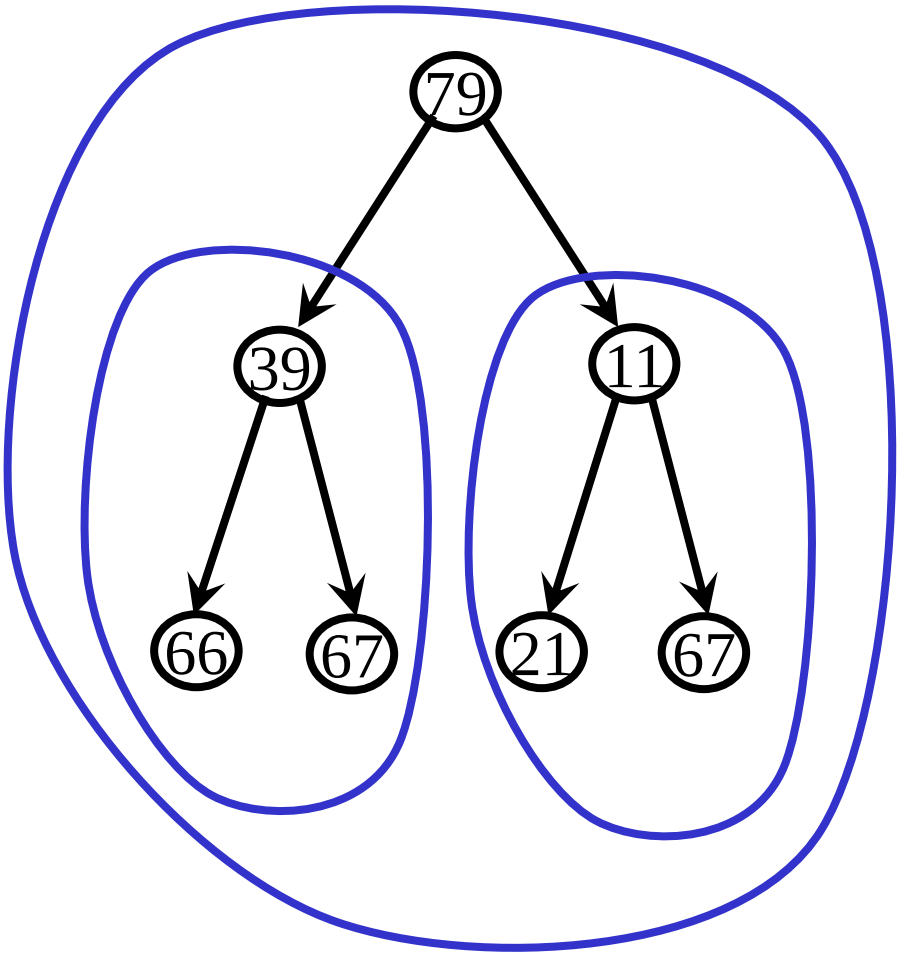
```
1. void Xuat(TREE t)
2. {
3.     if(t==NULL)
4.         return;
5.     Xuat(t->pRight);
6.     Xuat(t->pLeft);
7.     printf("%4d", t->info);
8. }
```

Bài 01: Đếm số nút trong cây nhị phân các số thực.

- Cấu trúc dữ liệu

```
1. struct node
2. {
3.     float info;
4.     struct node *pLeft;
5.     struct node *pRight;
6. };
7. typedef struct node NODE;
8. typedef NODE* TREE;
```

Bài 01: Đếm số nút trong cây nhị phận các số thực.



Bài 01: Đếm số nút trong cây nhị phân các số thực.

- Định nghĩa hàm

```
1. int DemNode(TREE t)
2. {
3.     if(t==NULL)
4.         return 0;
5.     int a = DemNode(t->pLeft);
6.     int b = DemNode(t->pRight);
7.     return (a+b+1);
8. }
```

Bài 02: Đếm số nút có đầy đủ 2 cây con trong cây nhị các số nguyên.

- Cấu trúc dữ liệu

```
1. struct node
2. {
3.     int info;
4.     struct node *pLeft;
5.     struct node *pRight;
6. };
7. typedef struct node NODE;
8. typedef NODE* TREE;
```

Bài 02: Đếm số nút có đầy đủ 2 cây con trong cây nhị các số nguyên.

- Định nghĩa hàm

```
11. int DemHaiCon(TREE t)
12. {
13.     if(t==NULL)
14.         return 0;
15.     int a=DemHaiCon(t->pLeft);
16.     int b=DemHaiCon(t->pRight);
17.     if(t->pLeft!=NULL &&
18.        t->pRight!=NULL)
19.         return a+b+1;
20.     return (a+b);
21. }
```

Bài 02: Đếm số nút có đầy đủ 2 cây con trong cây nhị các số nguyên.

- Một cách định nghĩa hàm khác

```
11. int DemHaiCon(TREE t)
12. {
13.     if(!t)
14.         return 0;
15.     int a=DemHaiCon(t->pLeft);
16.     int b=DemHaiCon(t->pRight);
17.     if(t->pLeft && t->pRight)
18.         return a+b+1;
19.     return (a+b);
20. }
```

Bài 03: Đếm số nút lá trong cây nhị các phân số.

- Cấu trúc dữ liệu

```
11. struct phanso
12. {
13.     int tu;
14.     int mau;
15. };
16. typedef struct phanso PHANSO;
17. struct node
18. {
19.     PHANSO info;
20.     struct node *pLeft;
21.     struct node *pRight;
22. };
23. typedef struct node NODE;
24. typedef NODE* TREE;
```

Bài 03: Đếm số nút có lá trong cây nhị các phân số.

- Định nghĩa hàm

```
11. int DemLa(TREE t)
12. {
13.     if(t==NULL)
14.         return 0;
15.     int a=DemLa(t->pLeft);
16.     int b=DemLa(t->pRight);
17.     if(t->pLeft==NULL &&
18.        t->pRight==NULL)
19.         return a+b+1;
20.     return (a+b);
21. }
```

Bài 03: Đếm số nút có lá trong cây nhị các phân số.

- Định nghĩa hàm

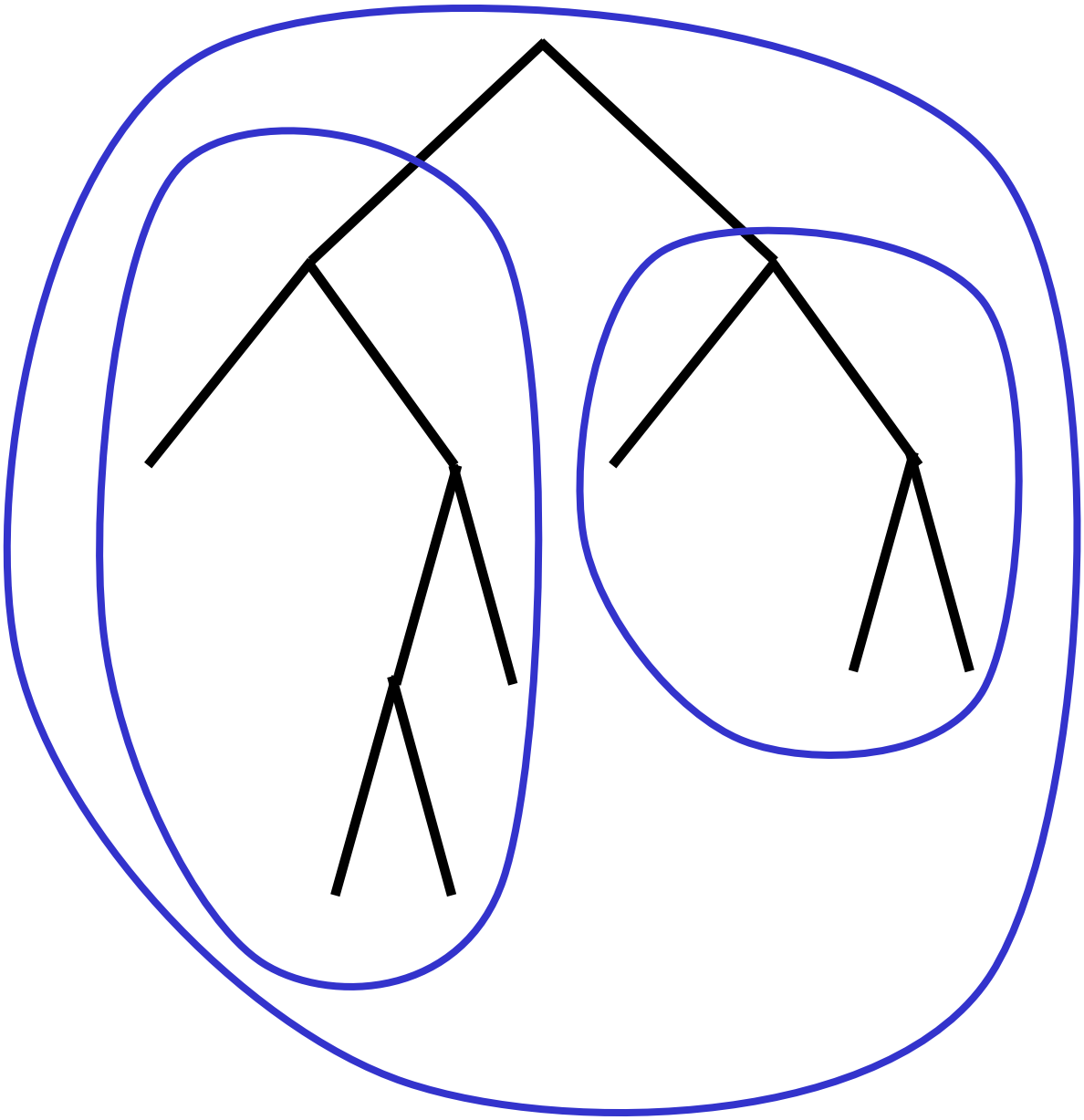
```
11. int DemLa(TREE t)
12. {
13.     if(!t)
14.         return 0;
15.     int a=DemLa(t->pLeft);
16.     int b=DemLa(t->pRight);
17.     if(!t->pLeft && !t->pRight)
18.         return a+b+1;
19.     return (a+b);
20. }
```

Bài 04: Tính chiều cao của cây nhị phân các số nguyên.

- Cấu trúc dữ liệu

```
1. struct node
2. {
3.     int info;
4.     struct node *pLeft;
5.     struct node *pRight;
6. };
7. typedef struct node NODE;
8. typedef NODE* TREE;
```

Bài 04: Tính chiều cao của cây nhị phân các số nguyên.



TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

Bài 04: Tính chiều cao của cây nhị phân các số nguyên.

- Định nghĩa hàm

```
11. int ChieuCao(TREE t)
12. {
13.     if(t==NULL)
14.         return 0;
15.     int a=ChieuCao(t->pLeft);
16.     int b=ChieuCao(t->pRight);
17.     if(a>b)
18.         return a+1;
19.     return (b+1);
20. }
```

Bài 05: Định nghĩa hàm tìm địa chỉ một node trong cây nhị phân tìm kiếm các số nguyên có khóa bằng x.

- Cấu trúc dữ liệu

```
1. struct node
2. {
3.     int info;
4.     struct node *pLeft;
5.     struct node *pRight;
6. };
7. typedef struct node NODE;
8. typedef NODE* TREE;
```

Bài 05: Định nghĩa hàm tìm địa chỉ một node trong cây nhị phân tìm kiếm các số nguyên có khóa bằng x.

- Định nghĩa hàm

```
11. NODE* TimKiem(TREE t,int x)
12. {
13.     if(t==NULL)
14.         return NULL;
15.     if(t->info==x)
16.         return t;
17.     if(x < t->info)
18.         return TimKiem(t->pLeft,x);
19.     return TimKiem(t->pRight,x);
20. }
```