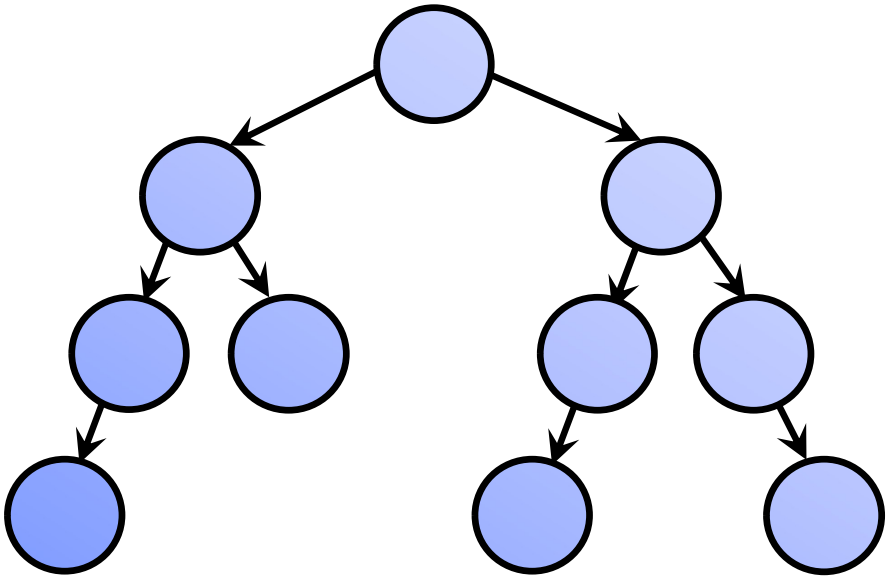
CÂY NHỊ PHÂN – BINARY TREE

1. Hồ Thái Ngọc
2. ThS. Võ Duy Nguyên
3. TS. Nguyễn Tấn Trần Minh Khang

HÌNH ẢNH CÂY NHỊ PHÂN

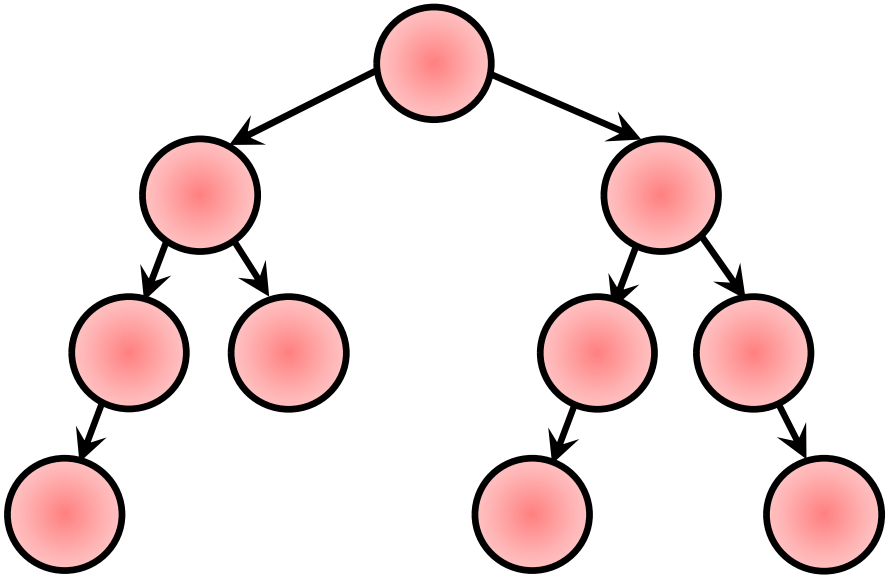
Hình ảnh cây nhị phân



KHÁI NIỆM CÂY NHỊ PHÂN

Khái niệm cây nhị phân

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



Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

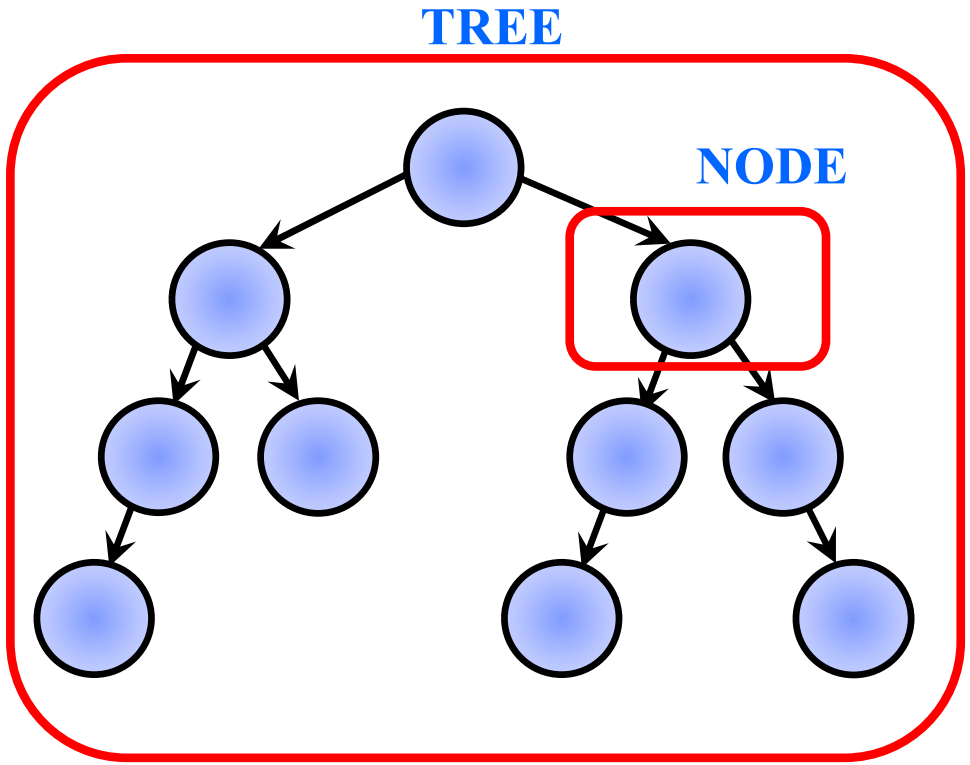
TS. Nguyễn Tấn Trần Minh Khang

CẤU TRÚC DỮ LIỆU CỦA CÂY NHỊ PHÂN

Cấu trúc dữ liệu của cây nhị phân

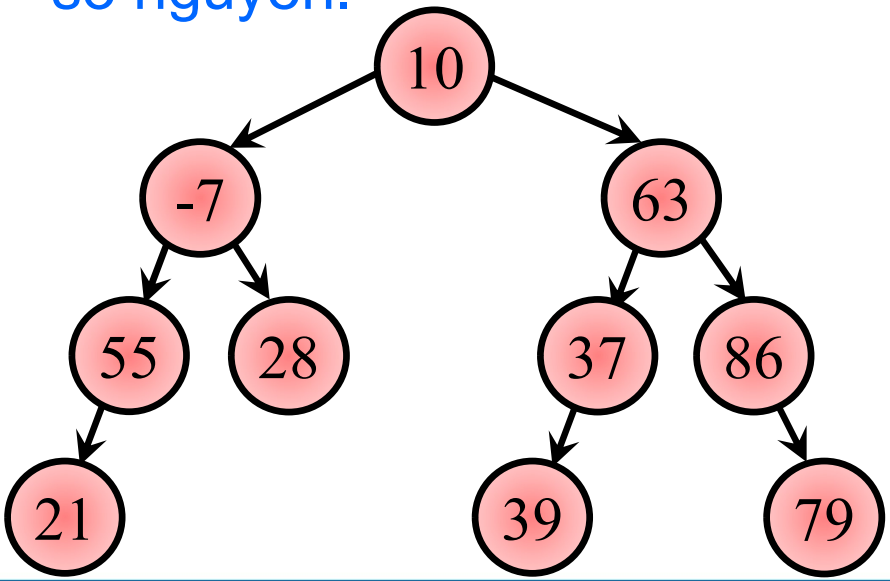
— Cấu trúc dữ liệu

```
11.struct node
12.{
13.|    KDL info;
14.|    struct node* pLeft;
15.|    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```



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.

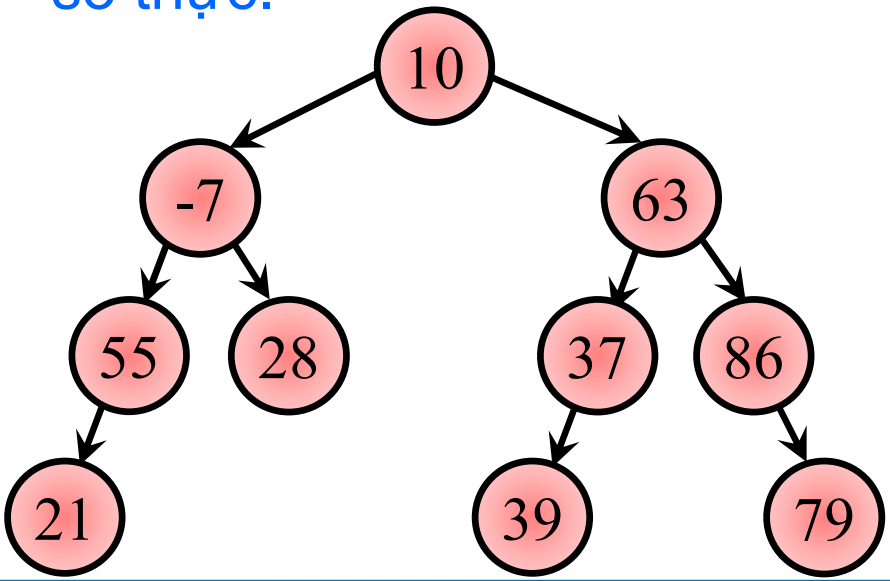


— Cấu trúc dữ liệu

```
11.struct node
12.{
13.    int info;
14.    struct node* pLeft;
15.    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

Cấu trúc dữ liệu của cây nhị phân

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



— Cấu trúc dữ liệu

```
11.struct node
12.{
13.    float info;
14.    struct node* pLeft;
15.    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

Cấu trúc dữ liệu của cây nhị phân

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

— Cấu trúc dữ liệu

```
11.struct phanso
12.{
13.|    int tu;
14.|    int mau;
15.};
16.typedef struct phanso PHANSO;
```

— Cấu trúc dữ liệu

```
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;
```

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

KHỞI TẠO CÂY NHỊ PHÂN

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

```
11.void Init(TREE& t)
12.{
13.    t = NULL;
14.}
```

KIỂM TRA CÂY NHỊ PHÂN RỖNG

Kiểm tra cây nhị phân rỗng

- Khái niệm: Kiểm tra cây nhị phân rỗng là hàm trả về giá trị 1 khi cây rỗng. Trong tình huống cây không rỗng thì hàm sẽ trả về giá trị 0.

```
11.int IsEmpty(TREE t)
12.{
13.    if(t==NULL)
14.        return 1;
15.    return 0;
16.}
```

Kiểm tra cây nhị phân rỗng

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

TẠO NODE CHO CÂY NHỊ PHÂN

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.

Tạo node cho cây nhị phân trừu tượng

```
11.NODE* GetNode(KDL x)
12.{
13.    NODE* p=new NODE;
14.    if(p==NULL)
15.        return NULL;
16.    p->info = x;
17.    p->pLeft=NULL;
18.    p->pRight=NULL;
19.    return p;
20.}
```

Tạo node cho cây nhị phân

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

```
11.NODE* GetNode(      )
12.{
13.|    NODE* p=new NODE;
14.|    if(p==NULL)
15.|        return NULL;
16.|    p->info = x;
17.|    p->pLeft=NULL;
18.|    p->pRight=NULL;
19.|    return p;
20.}
```

Tạo node cho cây nhị phân số nguyên

Tạo node cho cây nhị phân

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

```
11.NODE* GetNode(int x)
12.{
13.    NODE* p=new NODE;
14.    if(p==NULL)
15.        return NULL;
16.    p->info = x;
17.    p->pLeft=NULL;
18.    p->pRight=NULL;
19.    return p;
20.}
```

Tạo node cho cây nhị phân số nguyên

Tạo node cho cây nhị phân

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

```
11.NODE* GetNode(          )
12.{
13.|    NODE* p=new NODE;
14.|    if(p==NULL)
15.|        return NULL;
16.|    p->info = x;
17.|    p->pLeft=NULL;
18.|    p->pRight=NULL;
19.|    return p;
20.}
```

Tạo node cho cây nhị phân số thực

Tạo node cho cây nhị phân

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

```
11.NODE* GetNode(float x)
12.{
13.    NODE* p=new NODE;
14.    if(p==NULL)
15.        return NULL;
16.    p->info = x;
17.    p->pLeft=NULL;
18.    p->pRight=NULL;
19.    return p;
20.}
```

Tạo node cho cây nhị phân số thực

Tạo node cho cây nhị phân

– Ví dụ 3: Định nghĩa hàm tạo node cho cây nhị phân các phân số.

```
11.NODE* GetNode(          )
12.{
13.    NODE* p=new NODE;
14.    if(p==NULL)
15.        return NULL;
16.    p->info = x;
17.    p->pLeft=NULL;
18.    p->pRight=NULL;
19.    return p;
20.}
```

Tạo node cho cây nhị phân các phân số

Tạo node cho cây nhị phân

– Ví dụ 3: Định nghĩa hàm tạo node cho cây nhị phân các phân số.

```
11.NODE* GetNode(PHANSO x)
12.{
13.    NODE* p=new NODE;
14.    if(p==NULL)
15.        return NULL;
16.    p->info = x;
17.    p->pLeft=NULL;
18.    p->pRight=NULL;
19.    return p;
20.}
```

Tạo node cho cây nhị phân các phân số

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

THÊM MỘT GIÁ TRỊ VÀO TRONG CÂY NHỊ PHÂN (NGẪU NHIÊN)

Thêm một giá trị vào trong cây

- Khái niệm: Thêm một giá trị vào trong cây nhị phân là thêm thông tin vào cây một cách ngẫu nhiên.
- Giá trị trả về: Hàm thêm một giá trị vào trong cây nhị phân trả về một trong 2 giá trị -1, 1 với ý nghĩa như sau:
  + Giá trị $+1$: Thêm thành công.
  + Giá trị -1: Không đủ bộ nhớ.

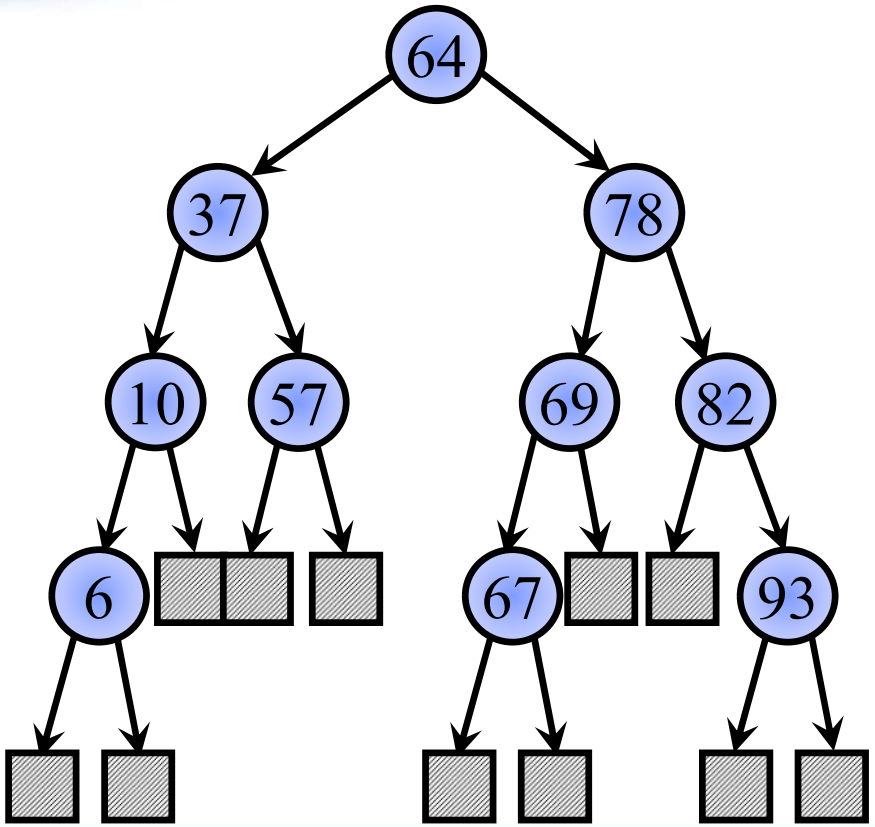
Thêm một giá trị vào trong cây

- Bài toán: Hãy định nghĩa hàm thêm một giá trị vào trong cây nhị phân các số nguyên.

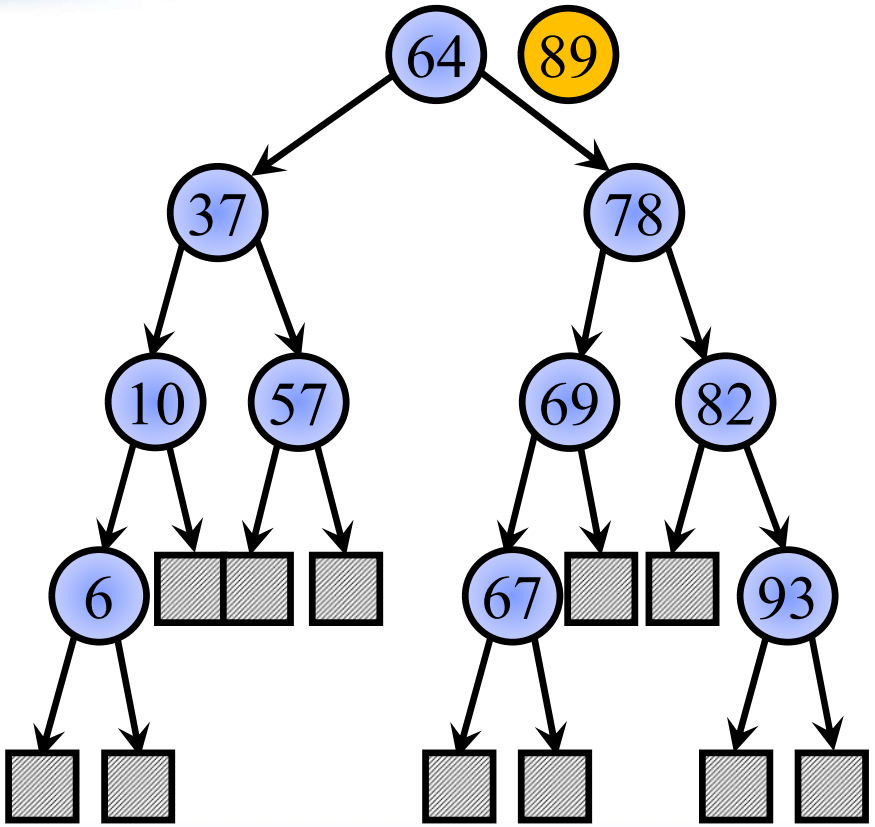
- Cấu trúc dữ liệu

```
11.struct node
12.{
13.    int info;
14.    struct node* pLeft;
15.    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

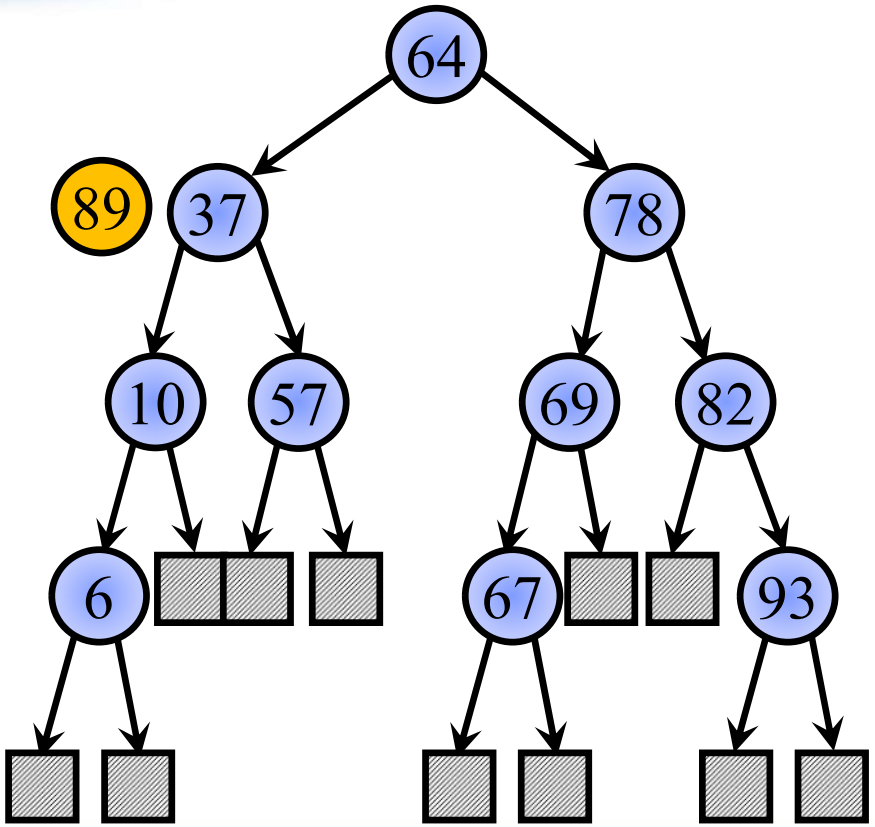
Thêm một giá trị vào trong cây



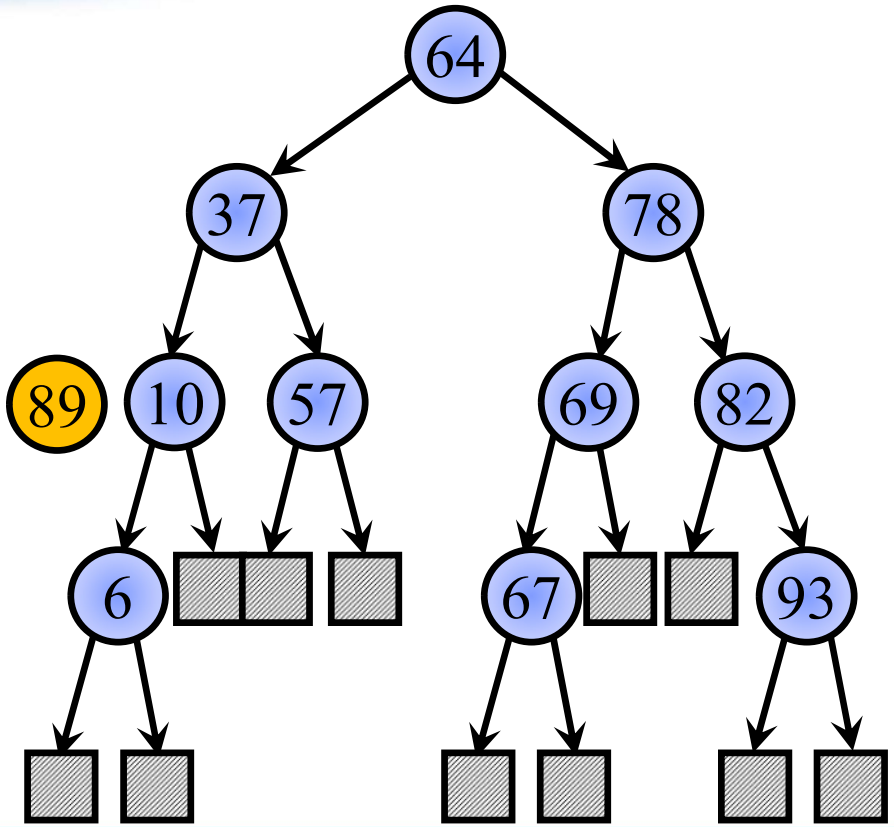
Thêm một giá trị vào trong cây



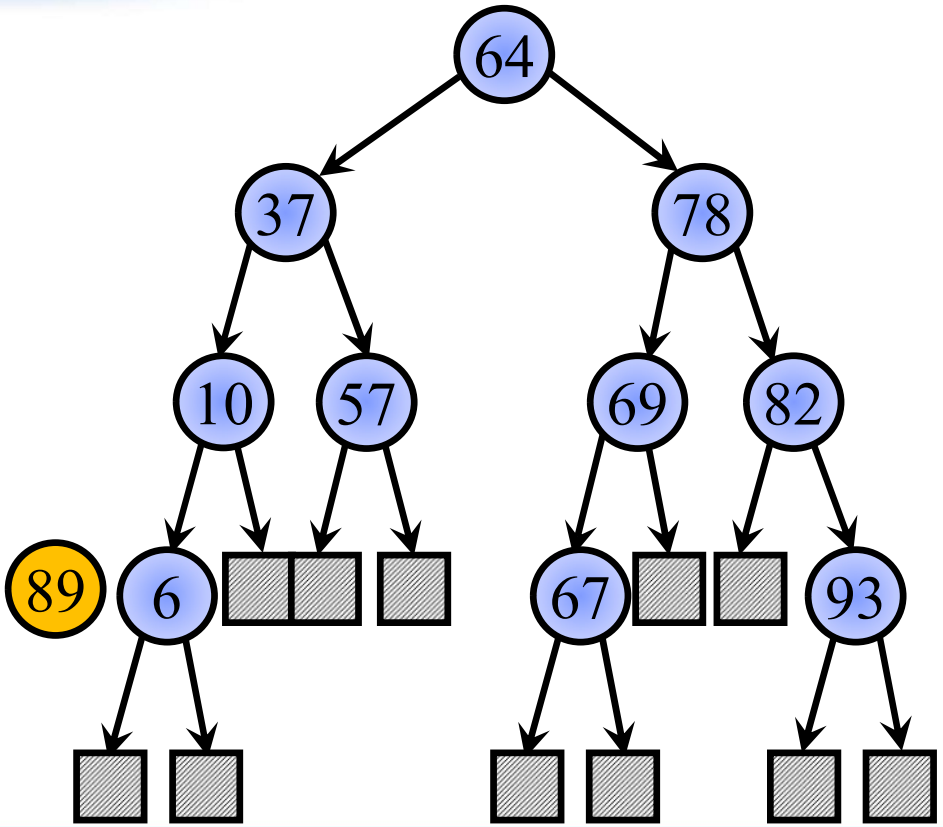
Thêm một giá trị vào trong cây



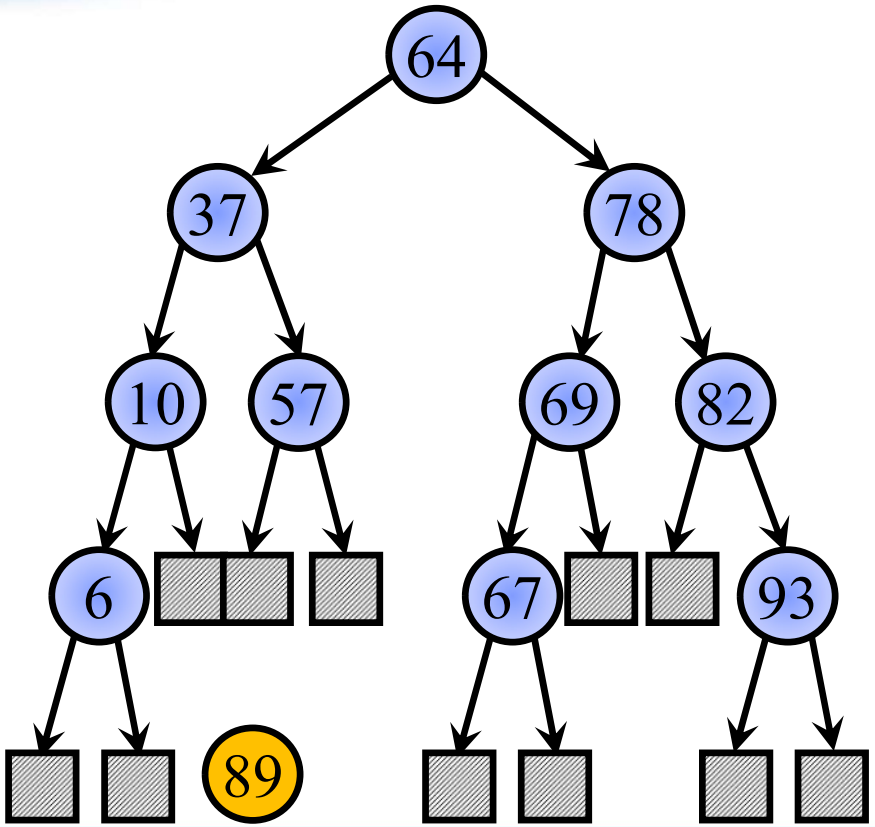
Thêm một giá trị vào trong cây



Thêm một giá trị vào trong cây



Thêm một giá trị vào trong cây



```
11. int InsertNode(TREE &t,int x)
12. {
13.     if(t!=NULL)
14.     {
15.         int rValue =  rand();
16.         if(rValue %2 == 0)
17.             return InsertNode(t->pRight,x);
18.         return InsertNode(t->pLeft,x);
19.     }
20.     t = GetNode(x);
21.     if(t==NULL)
22.         return -1;
23.     return 1;
24. }
```

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

NHẬP CÂY NHỊ PHÂN CÁC SỐ NGUYÊN TỪ FILE

Nhập cây nhị phân các số nguyên từ file

- Bài toán: Hãy định nghĩa hàm nhập cây nhị phân các số nguyên từ file.

- Cấu trúc dữ liệu

```
11.struct node
12.{
13.    int info;
14.    struct node* pLeft;
15.    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

```
11.int Nhap(TREE  t, string filename)
12.{
13      ifstream fi(fil
25.}
```

Nhập cây nhị phân số nguyên từ file

```
11.int Nhap(TREE& t, string filename)
12.{
13      ifstream fi(fil
25.}
```

Nhập cây nhị phân số nguyên từ file

```
11. int Nhap(TREE& t, string filename)
12. {
13.     ifstream fi(filename);
25. }
```

Nhập cây nhị phân số nguyên từ file

```
11. int Nhap(TREE& t, string filename)
12. {
13.     ifstream fi(filename);
25. }
```

Ý1: fi là đối tượng thuộc lớp ifstream.

Nhập cây nhị phân số nguyên từ file

```
11. int Nhap(TREE& t, string filename)
12. {
13.     ifstream fi(filename);
25. }
```

Ý2: Dòng lệnh số 13 khai báo đối tượng `fi` với đối số có tên `filename` và có kiểu `string`.

Nhập cây nhị phân số nguyên từ file

```
11. int Nhap(TREE& t, string filename)
12. {
13.     ifstream fi(filename);
...
25. }
```

Ý3: Khi chương trình thực hiện tới dòng lệnh số 13. Đối tượng `fi` gọi thực hiện phương thức thiết lập với tham số có kiểu `string`.

Nhập cây nhị phân số nguyên từ file

```
11. int Nhap(TREE& t, string filename)
12. {
13.     ifstream fi(filename);
14.     if (fi.fail())
15.         return 0;
16.     int n,x;
17.     fi >> n;
18.     Init(t);
19.     for (int i = 1; i <= n; i++)
20.     {
21.         fi >> x;
22.         InsertNode(t, x);
23.     }
24.     return 1;
25. }
```

Nhập cây nhị phân số nguyên từ file

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

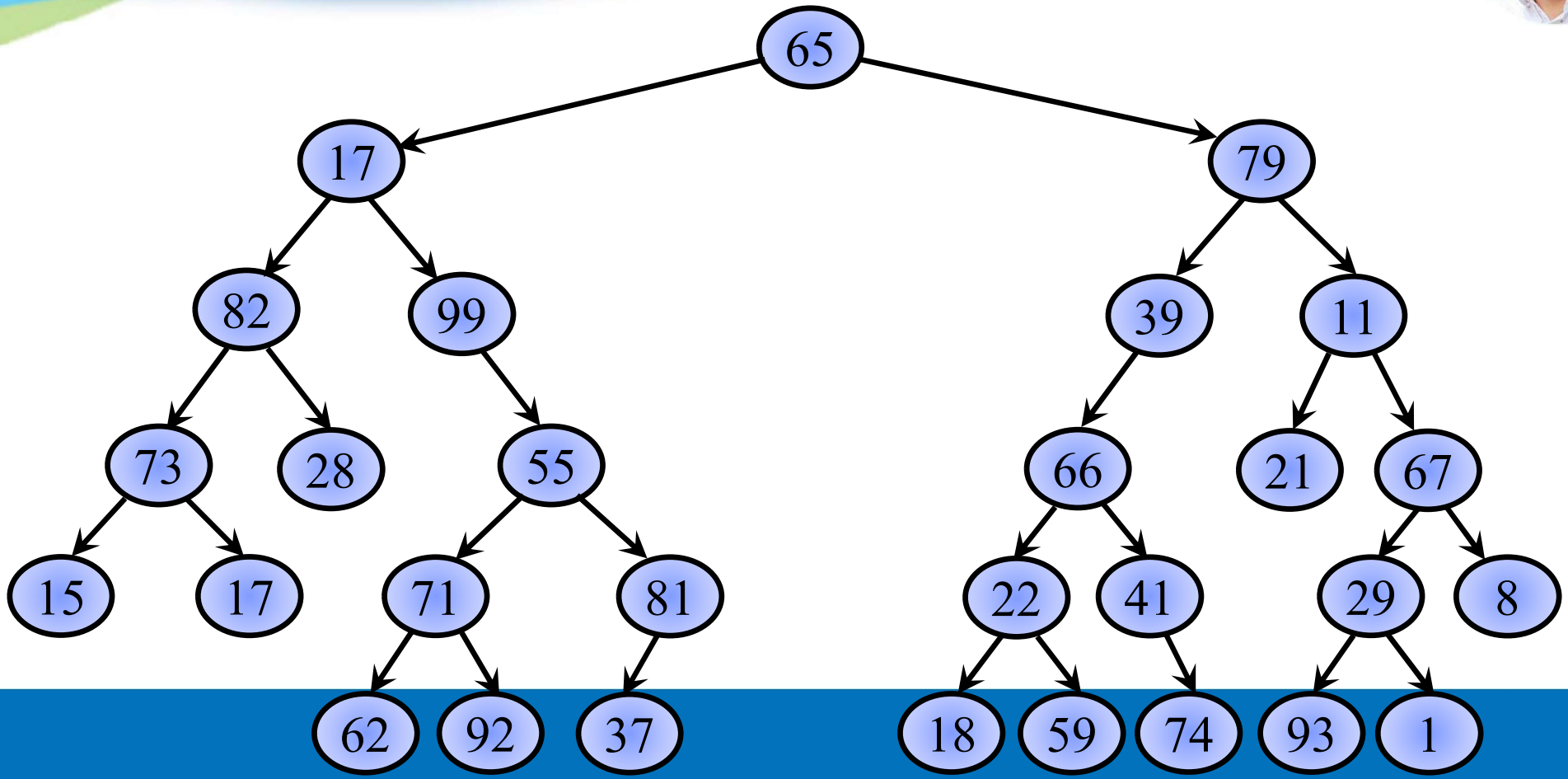
Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

NHÌN CÂY NHỊ PHÂN DƯỚI CON MẮT ĐỆ QUY

Nhìn cây nhị phân dưới con mắt đệ quy



Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

CÁC PHƯƠNG PHÁP DUYỆT CÂY

Các phương pháp duyệt cây

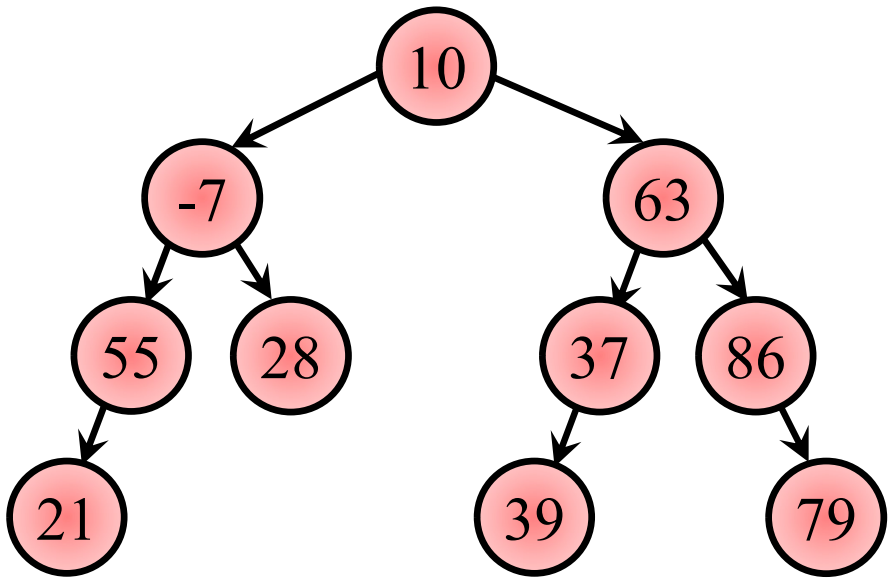
- Khái niệm: Duyệt cây nhị phân là thăm qua tất cả các node trong cây mỗi node một lần.
- Các phương pháp duyệt cây:
  + Phương pháp duyệt cây theo thứ tự giữa (node ở giữa).
  + Phương pháp duyệt cây theo thứ tự trước (node ở trước).
  + Phương pháp duyệt cây theo thứ tự sau (node ở sau).

Các phương pháp duyệt cây

- Phương pháp duyệt cây theo thứ tự giữa (node ở giữa).
  - + Phương pháp LNR (Left Node Right).
  - + Phương pháp RNL (Right Node Left).
- Phương pháp duyệt cây theo thứ tự trước (node ở trước).
  - + Phương pháp NLR (Node Left Right).
  - + Phương pháp NRL (Node Right Left).
- Phương pháp duyệt cây theo thứ tự sau (node ở sau).
  - + Phương pháp LRN (Left Right Node).
  - + Phương pháp RLN (Right Left Node).

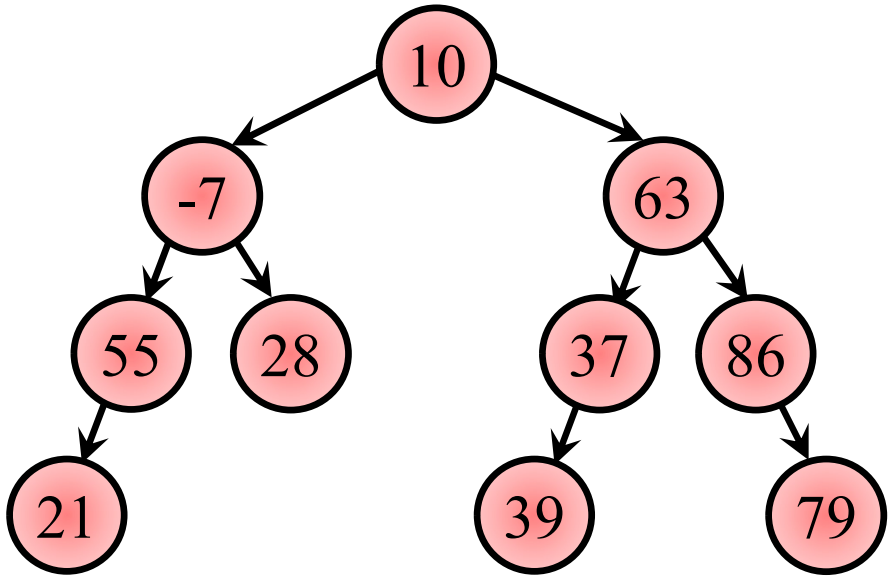
PHƯƠNG PHÁP 1: LEFT NODE RIGHT

Phương pháp left node right



- 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.

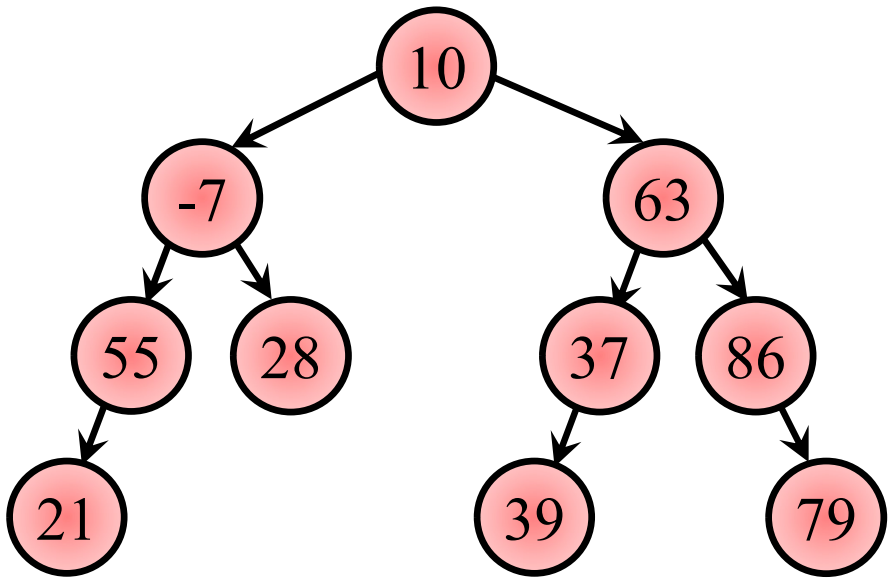
Phương pháp left node right



- 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.
- Kết quả duyệt cây bên trái: 21, 55, -7, 28, 10, 39, 37, 63, 86, 79.

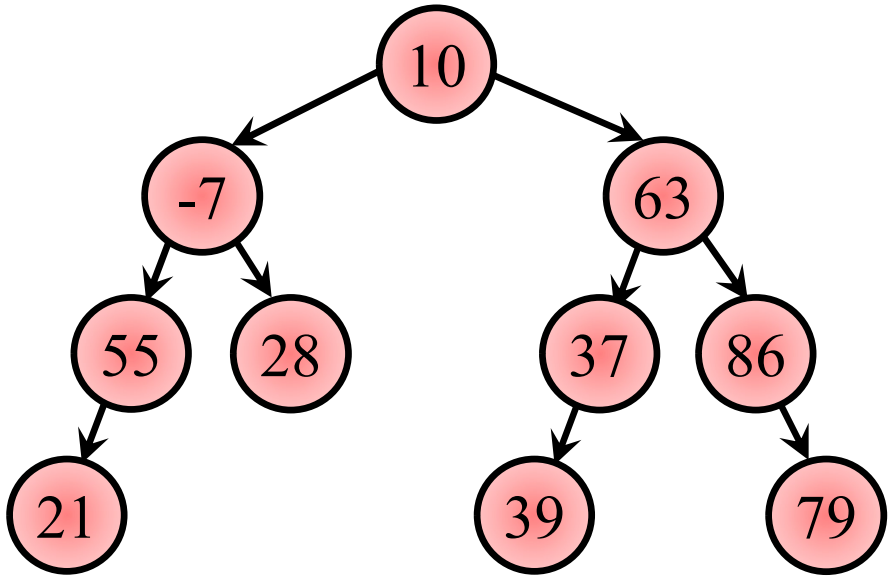
PHƯƠNG PHÁP 2: RIGHT NODE LEFT

Phương pháp right node left



- 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.

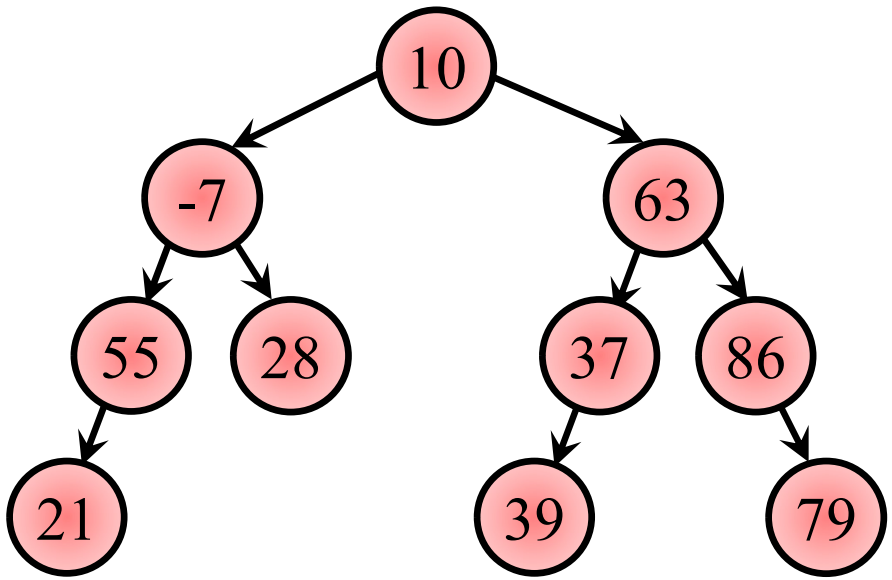
Phương pháp right node left



- 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.
- Kết quả duyệt cây bên trái: 79, 86, 63, 37, 39, 10, 28, -7, 55, 21.

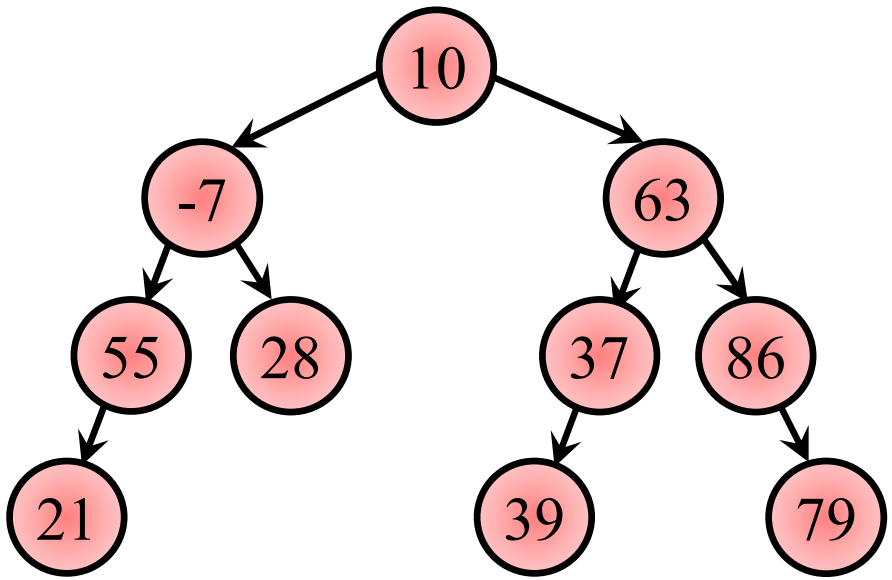
PHƯƠNG PHÁP 3: NODE LEFT RIGHT

Phương pháp node left right



- 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.

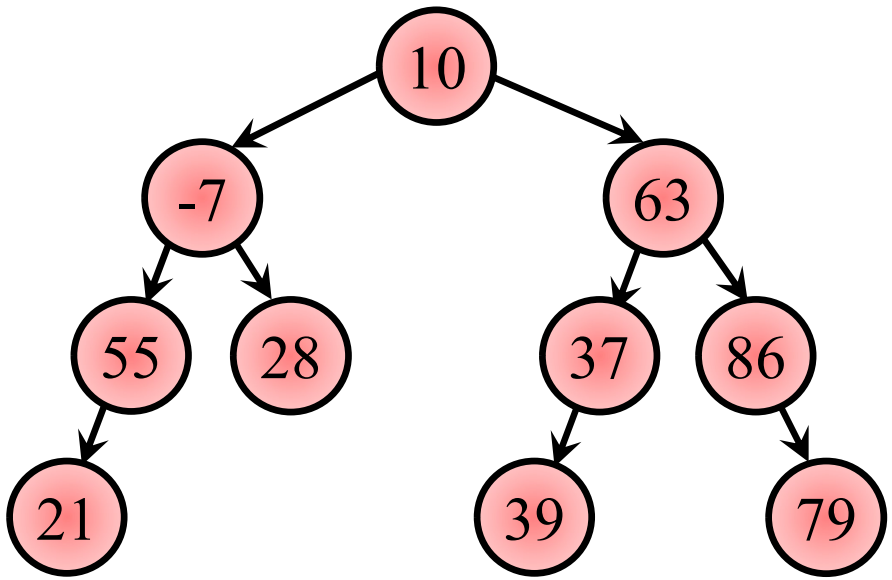
Phương pháp node left right



- 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.
- Kết quả duyệt cây bên trái: 10, -7, 55, 21, 28, 63, 37, 39, 86, 79.

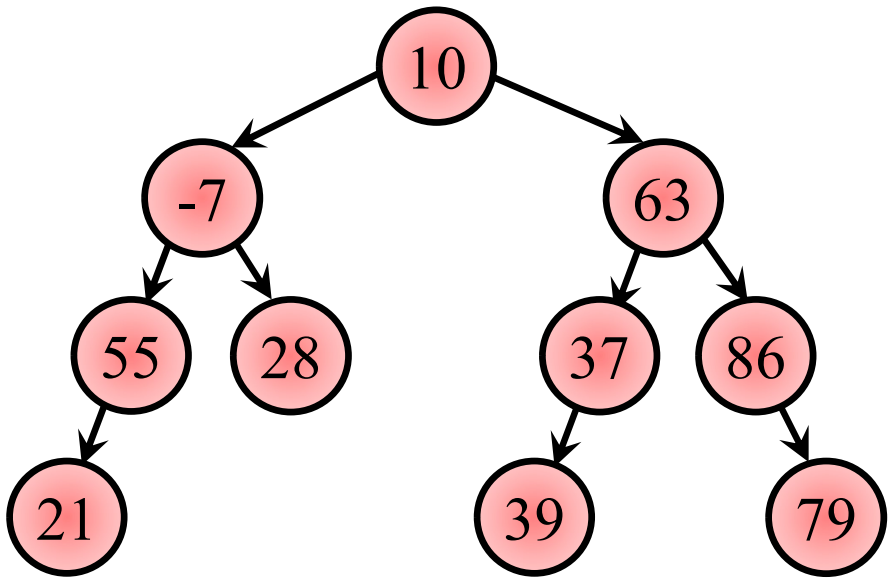
PHƯƠNG PHÁP 4: NODE RIGHT LEFT

Phương pháp node right left



- 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.

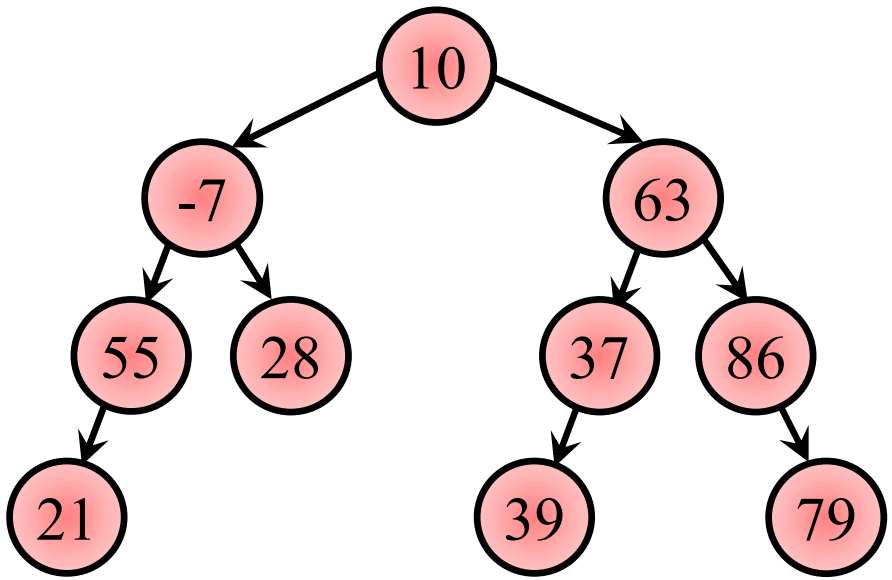
Phương pháp node right left



- 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.
- Kết quả duyệt cây bên trái: 10, 63, 86, 79, 37, 39, -7, 28, 55, 21.

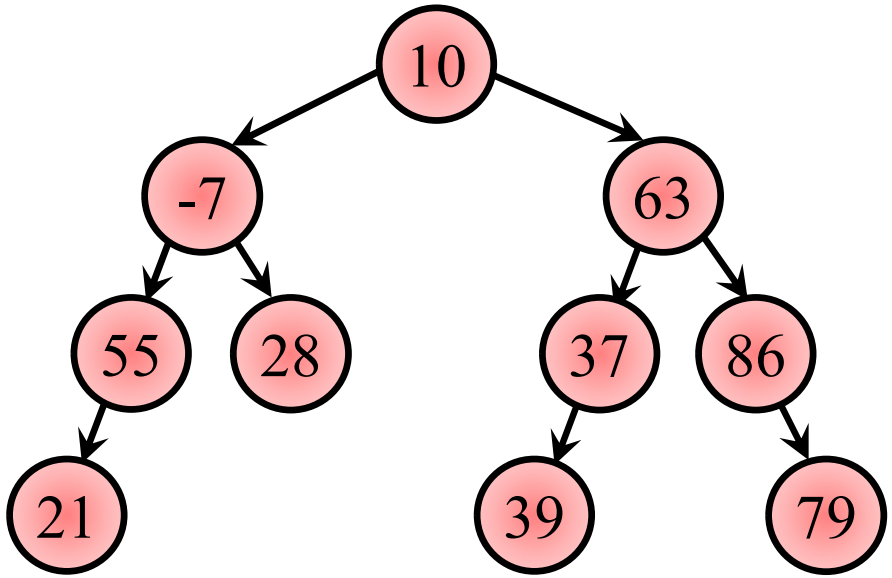
PHƯƠNG PHÁP 5: LEFT RIGHT NODE

Phương pháp left right node



- 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.

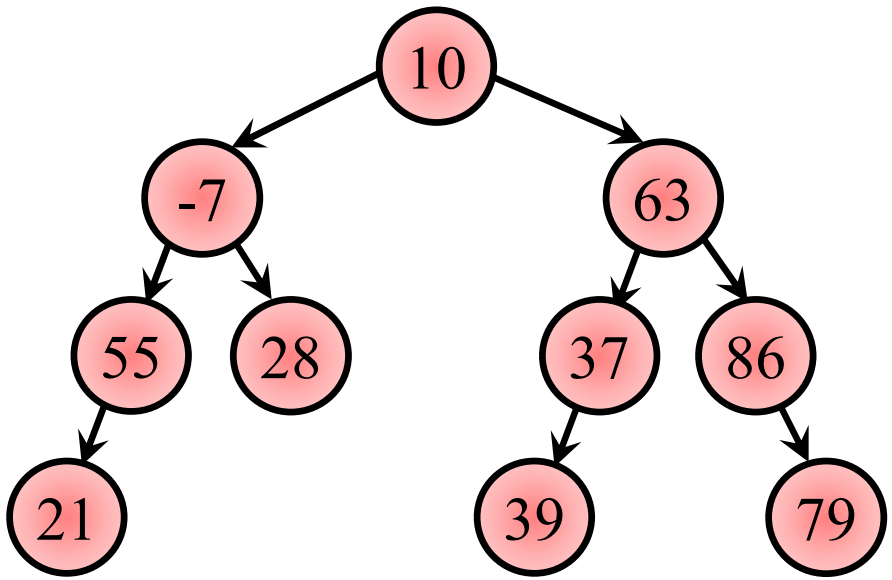
Phương pháp left right node



- 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.
- Kết quả duyệt cây bên trái: 21, 55, 28, -7, 39, 37, 79, 86, 63, 10.

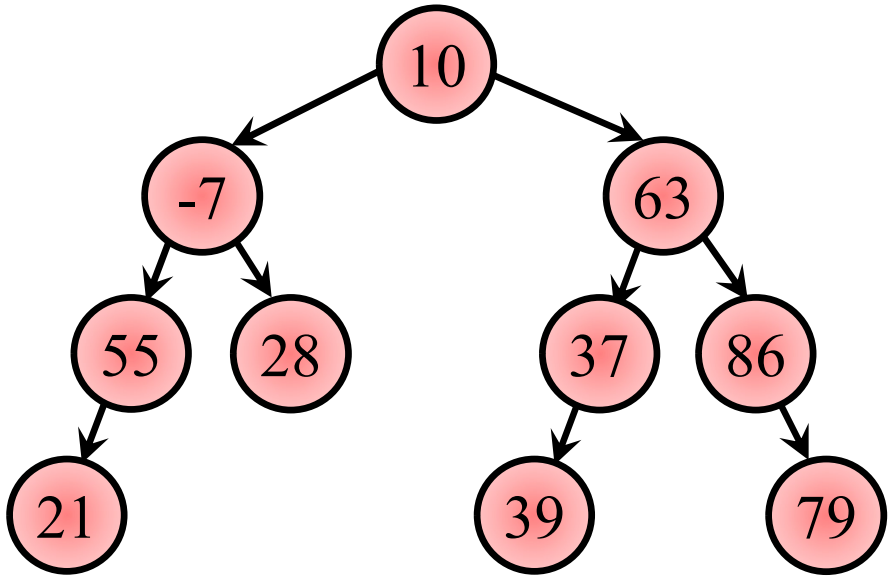
PHƯƠNG PHÁP 6: RIGHT LEFT NODE

Phương pháp right left node



- 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.

Phương pháp right left node



- 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.
- Kết quả duyệt cây bên trái: 79, 86, 39, 37, 63, 28, 21, 55, -7, 10.

Cảm ơn quí vị đã lắng nghe

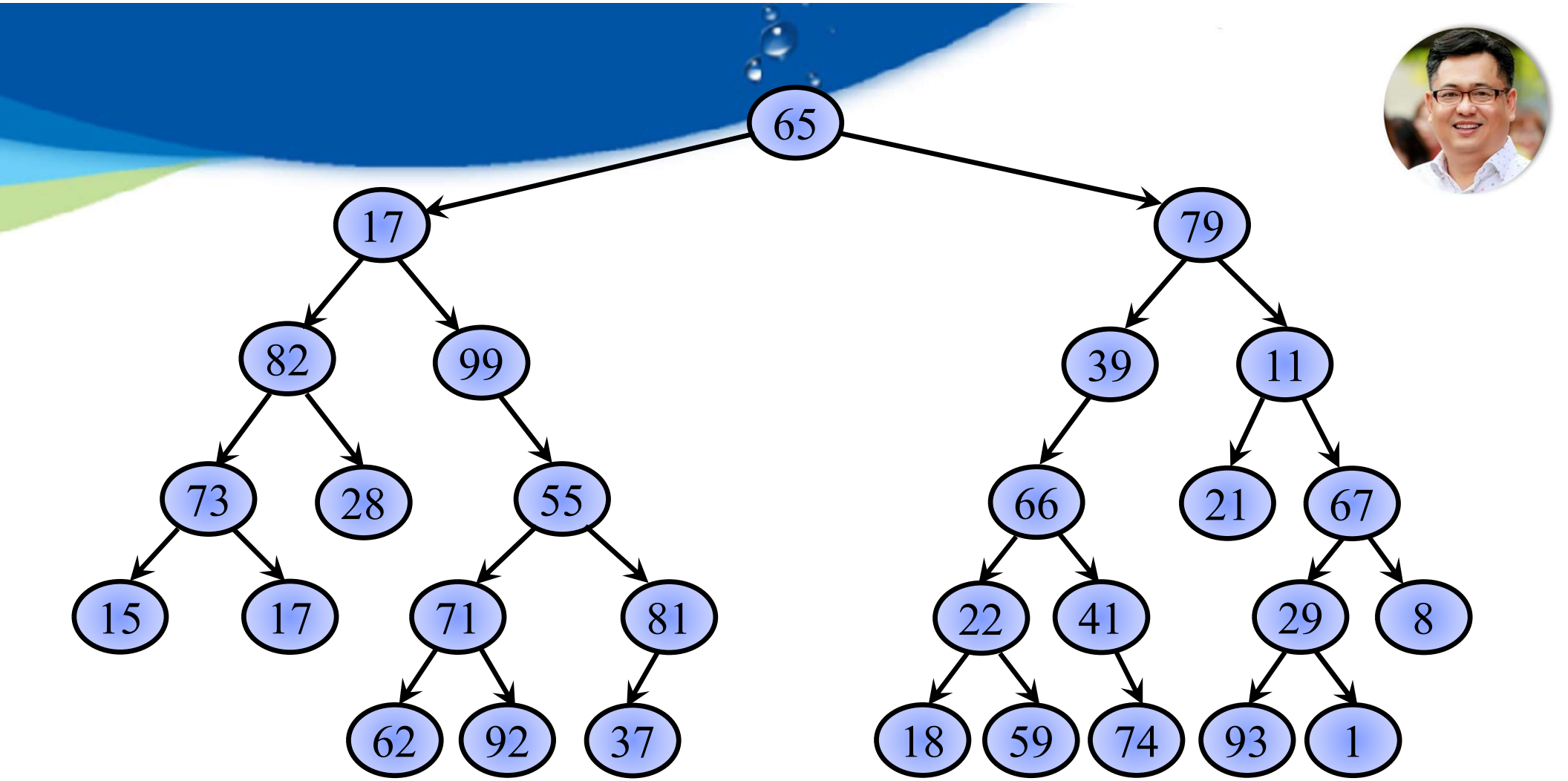
Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

CHẠY TỪNG BƯỚC DUYỆT CÂY


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

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

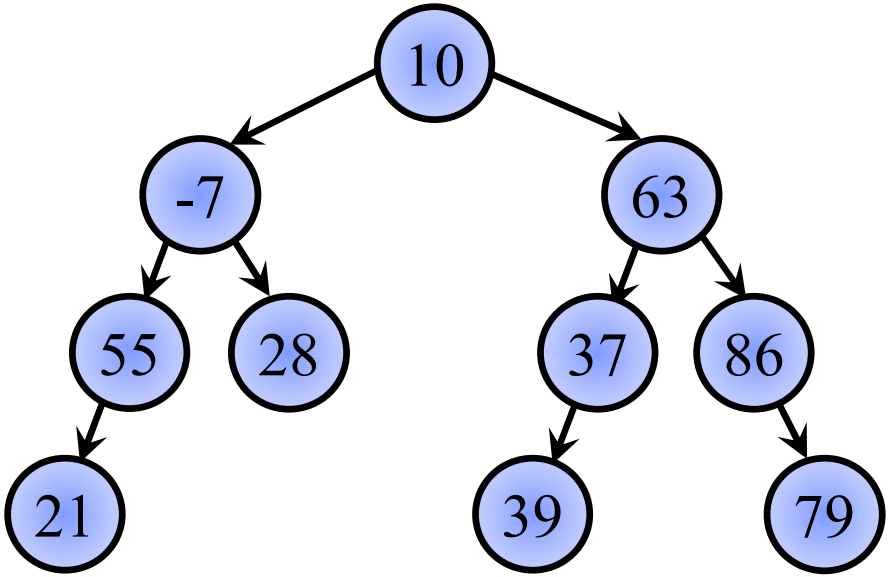
ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

XUẤT CÂY NHỊ PHÂN CÁC SỐ NGUYÊN

Xuất cây nhị phân các số nguyên

- Bài toán: Hãy định nghĩa tất cả các hàm duyệt và xuất cây nhị phân các số nguyên bằng 6 phương pháp.
  - + LNR – Left Node Right
  - + RNL – Right Node Left
  - + NLR – Node Left Right
  - + NRL – Node Right Left
  - + LRN – Left Right Node
  - + RNL – Node Right Left



Xuất cây nhị phân các số nguyên

- Bài toán: Hãy định nghĩa tất cả các hàm duyệt và xuất cây nhị phân các số nguyên bằng 6 phương pháp.
  - + LNR – Left Node Right
  - + RNL – Right Node Left
  - + NLR – Node Left Right
  - + NRL – Node Right Left
  - + LRN – Left Right Node
  - + RNL – Node Right Left

- Cấu trúc dữ liệu

```
11.struct node
12.{
13.    int info;
14.    struct node* pLeft;
15.    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

Xuất cây nhị phân các số nguyên

— Phương pháp LNR

```
11.void Xuat(TREE t)
12.{
13.|    if(t==NULL)
14.|        return;
15.|    Xuat(t->pLeft);
16.|    cout << t->info;
17.|    Xuat(t->pRight);
18.}
```

— Cấu trúc dữ liệu

```
11.struct node
12.{
13.|    PHANSO info;
14.|    struct node* pLeft;
15.|    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

Xuất cây nhị phân các số nguyên

— Phương pháp RNL

```
11.void Xuat(TREE t)
12.{
13.|    if(t==NULL)
14.|        return;
15.|    Xuat(t->pRight);
16.|    cout << t->info;
17.|    Xuat(t->pLeft);
18.}
```

— Cấu trúc dữ liệu

```
11.struct node
12.{
13.|    PHANSO info;
14.|    struct node* pLeft;
15.|    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

Xuất cây nhị phân các số nguyên

— Phương pháp NLR

```
11.void Xuat(TREE t)
12.{
13.|    if(t==NULL)
14.|        return;
15.|    cout << t->info;
16.|    Xuat(t->pLeft);
17.|    Xuat(t->pRight);
18.}
```

— Cấu trúc dữ liệu

```
11.struct node
12.{
13.|    PHANSO info;
14.|    struct node* pLeft;
15.|    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

Xuất cây nhị phân các số nguyên

— Phương pháp NRL

```
11.void Xuat(TREE t)
12.{
13.|    if(t==NULL)
14.|        return;
15.|    cout << t->info;
16.|    Xuat(t->pRight);
17.|    Xuat(t->pLeft);
18.}
```

— Cấu trúc dữ liệu

```
11.struct node
12.{
13.|    PHANSO info;
14.|    struct node* pLeft;
15.|    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

Xuất cây nhị phân các số nguyên

— Phương pháp LRN

```
11.void Xuat(TREE t)
12.{
13.|    if(t==NULL)
14.|        return;
15.|    Xuat(t->pLeft);
16.|    Xuat(t->pRight);
17.|    cout << t->info;
18.}
```

— Cấu trúc dữ liệu

```
11.struct node
12.{
13.|    PHANSO info;
14.|    struct node* pLeft;
15.|    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

Xuất cây nhị phân các số nguyên

— Phương pháp RLN

```
11.void Xuat(TREE t)
12.{
13.|    if(t==NULL)
14.|        return;
15.|    Xuat(t->pRight);
16.|    Xuat(t->pLeft);
17.|    cout << t->info;
18.}
```

— Cấu trúc dữ liệu

```
11.struct node
12.{
13.|    PHANSO info;
14.|    struct node* pLeft;
15.|    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

ĐẾM SỐ LƯỢNG NODE TRONG CÂY NHỊ PHÂN CÁC SỐ NGUYÊN

Đếm số lượng node

— Cấu trúc dữ liệu

```
11.struct node
12.{
13.    int info;
14.    struct node* pLeft;
15.    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

Đếm số lượng node

— Đếm số lượng node

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

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

XUẤT CÂY NHỊ PHÂN CÁC SỐ NGUYÊN RA FILE

Xuất cây nhị phân các số nguyên ra file

- Bài toán: Hãy định nghĩa hàm xuất cây nhị phân các số nguyên ra file.
- Cấu trúc dữ liệu

```
11.struct node
12.{
13.|    int info;
14.|    struct node* pLeft;
15.|    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

Xuất cây nhị phân các số nguyên ra file

— Đếm số lượng node

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

Xuất cây nhị phân các số nguyên ra file

```
11.int Xuat(TREE t, string filename)
12.{
13.    ofstream fo(filename);
14.    if (fo.fail()==true)
15.        return 0;
16.    fo << setw(6) << DemNode(t) << endl;
17.    return Xuat(t, fo);
18.}
```

Xuất cây nhị phân các số nguyên ra file

```
11.int Xuat(TREE t, ofstream & fo)
12.{
13.    if (t == NULL)
14.        return 0;
15.    Xuat(t->pLeft, fo);
16.    fo << setw(6) << t->info;
17.    Xuat(t->pRight, fo);
18.}
```

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

DỰ ÁN CÂY NHỊ PHÂN

Dự án cây nhị phân

- Viết chương trình thực hiện các yêu cầu sau:
  - + Nhập cây nhị phân các số nguyên từ các tập tin: intdata01.inp; intdata02.inp; …; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
  - + Đếm số lượng node trong cây.
  - + Xuất cây nhị phân các số nguyên ra các tập tin: intdata01.out; intdata02.out; …; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

Dự án cây nhị phân

- Định dạng tập tin `intdataxx.inp` và `intdataxx.out`
  - + Dòng đầu tiên: số phần tử của cây nhị phân các số nguyên (n).
  - + Dòng tiếp theo: lưu n số nguyên tương ứng với các giá trị trong cây nhị phân các số nguyên.

Kiến trúc chương trình

```
11.#include <iostream>
12.#include <fstream>
13.#include <iomanip>
14.#include <string>
15.using namespace std;
```

Khai báo sử dụng thư viện

Kiến trúc chương trình

```
11.struct node
12.{
13.    int info;
14.    struct node* pLeft;
15.    struct node* pRight;
16.};
17.typedef struct node NODE;
18.typedef NODE* TREE;
```

Khai báo cấu trúc dữ liệu

Kiến trúc chương trình

```
11.void Init(TREE&);
12.NODE* GetNode(int);
13.int InsertNode(TREE&, int);
14.int Nhap(TREE&, string);
15.void Xuat(TREE);
16.int Xuat(TREE, string);
17.int Xuat(TREE, ofstream&);
18.int DemNode(TREE);
```

Khai báo hàm

Định nghĩa hàm main

```
11. int main()
12. {
13.     TREE t;
14.     for (int i = 1; i <= 13; i++)
15.     {
16.         string filename = "intdata";
17.         if (i < 10)
18.             filename += '0';
19.         filename += to_string(i);
20.         string filenameinp = filename;
21.         filenameinp += ".inp";
22.         if (Nhap(t, filenameinp) == 1)
23.         {
24.             string filenameout = filename;
25.             filenameout += ".out";
26.             Xuat(t, filenameout);
27.             cout << "\n" << filenameinp;
28.             cout << "\n" << filenameout;
29.         }
30.         else
31.             cout << "\n Khong mo duoc file " << filename << "\n";
32.     }
33.     cout << "\n\n\n";
34.     return 1;
35. }
```

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang