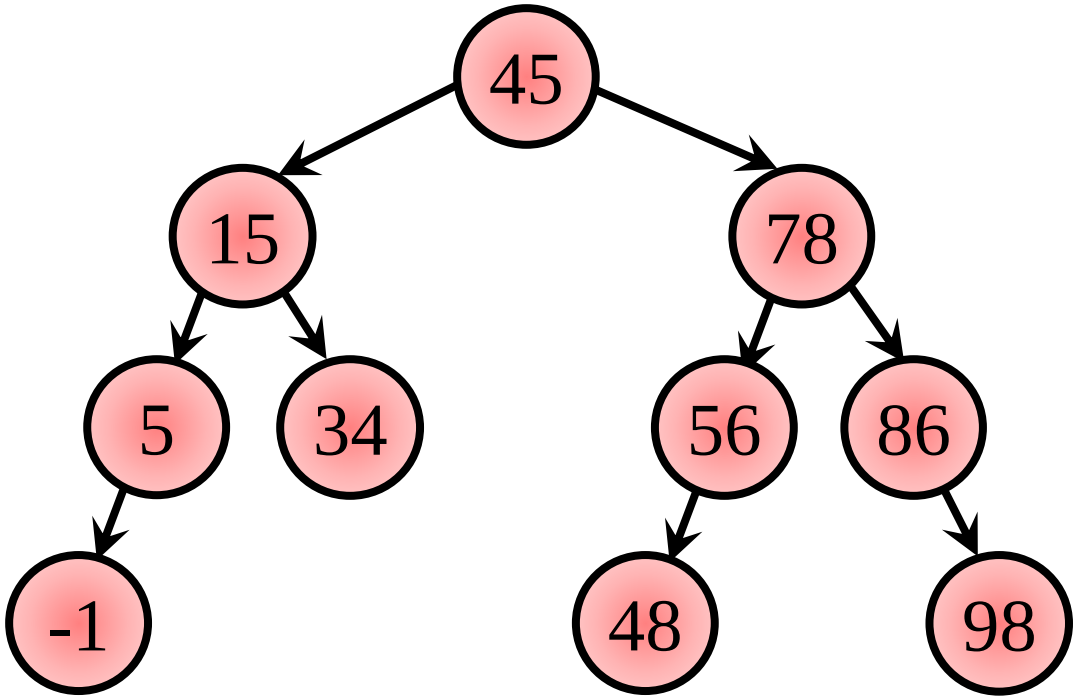
BINARY SEARCH TREE – PHẦN 01

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

HÌNH ẢNH CÂY NHỊ PHÂN TÌM KIẾM

Hình ảnh cây nhị phân tìm kiếm



KHÁI NIỆM CÂY NHỊ PHÂN TÌM KIẾM

Khái niệm cây nhị phân tìm kiếm

– Cây nhị phân tìm kiếm là:

+ Một cây nhị phân.

+ Mỗi nút p của cây đều thỏa:

- Tất cả các nút thuộc cây con trái ($\mathrm{p} \to \mathrm{pLeft}$) đều có giá trị nhỏ hơn giá trị của p

$$\text{"}\forall \mathrm{q} \in \mathrm{p} \to \mathrm{pLeft}: \mathrm{q} \to \mathrm{Data} < \mathrm{p} \to \mathrm{Data}\text{"}$$

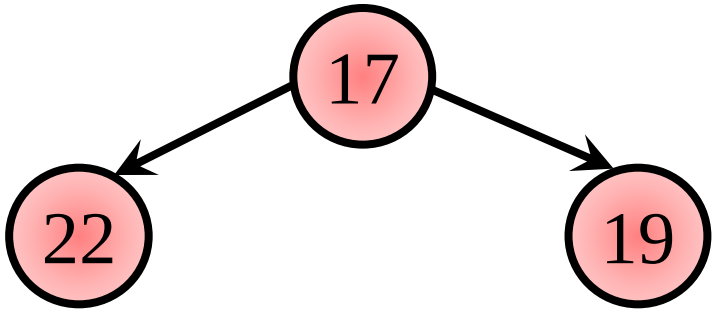
- Tất cả các nút thuộc cây con phải ($\mathrm{p} \to \mathrm{p}Right$) đều có giá trị lớn hơn giá trị của p.

$$\text{"}\forall \mathrm{q} \in \mathrm{p} \to \mathrm{p}Right: \mathrm{q} \to \mathrm{Data} > \mathrm{p} \to \mathrm{Data}\text{"}$$

VÍ DỤ CÂY NHỊ PHÂN TÌM KIẾM

Ví dụ cây nhị phân tìm kiếm

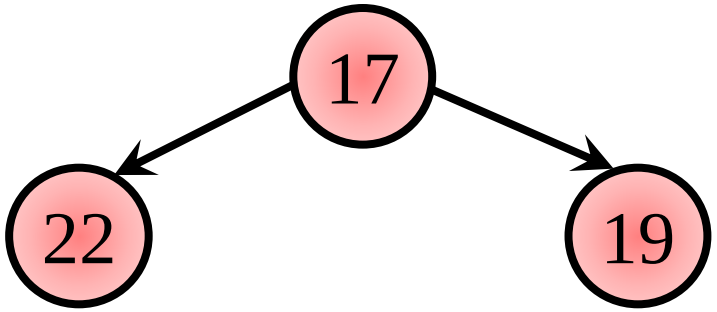
– Ví dụ 1: Cây nhị phân sau có phải là cây nhị phân tìm kiếm không?



– Trả lời:

Ví dụ cây nhị phân tìm kiếm

– Ví dụ 1: Cây nhị phân sau có phải là cây nhị phân tìm kiếm không?

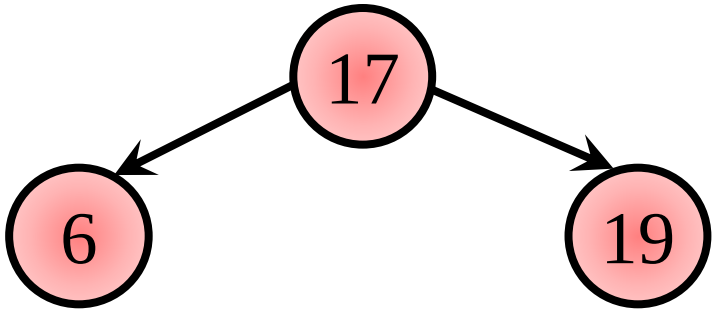


– Trả lời:

+ Đây không là một cây nhị phân tìm kiếm.

Ví dụ cây nhị phân tìm kiếm

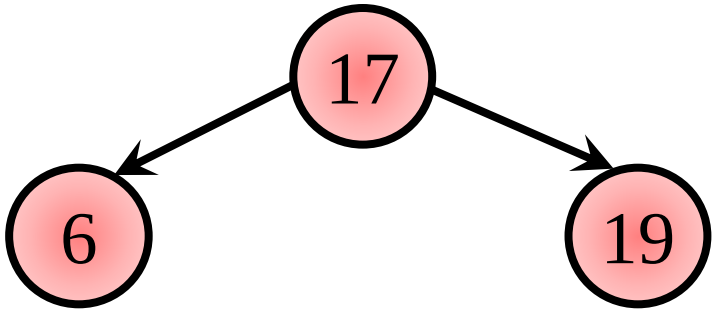
– Ví dụ 2: Cây nhị phân sau có phải là cây nhị phân tìm kiếm không?



– Trả lời:

Ví dụ cây nhị phân tìm kiếm

– Ví dụ 2: Cây nhị phân sau có phải là cây nhị phân tìm kiếm không?

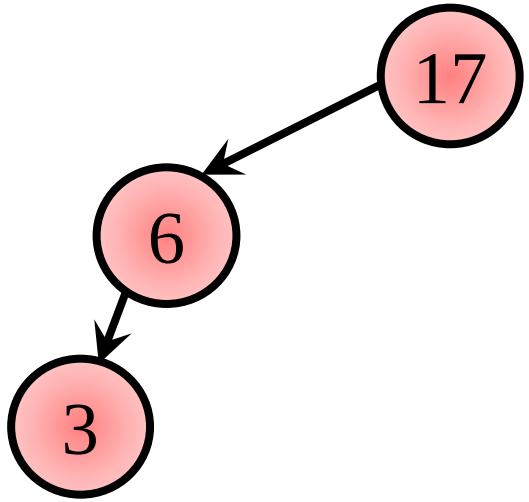


– Trả lời:

+ Đây là một cây nhị phân tìm kiếm.

Ví dụ cây nhị phân tìm kiếm

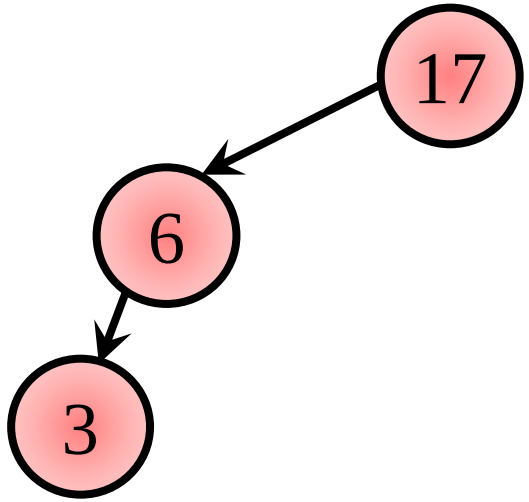
– Ví dụ 3: Cây nhị phân sau có phải là cây nhị phân tìm kiếm không?



– Trả lời:

Ví dụ cây nhị phân tìm kiếm

– Ví dụ 3: Cây nhị phân sau có phải là cây nhị phân tìm kiếm không?

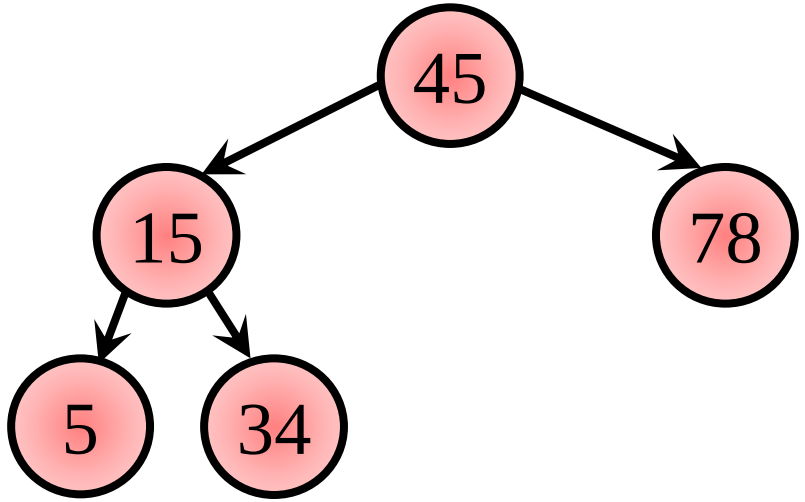


– Trả lời:

+ Đây là một cây nhị phân tìm kiếm.

Ví dụ cây nhị phân tìm kiếm

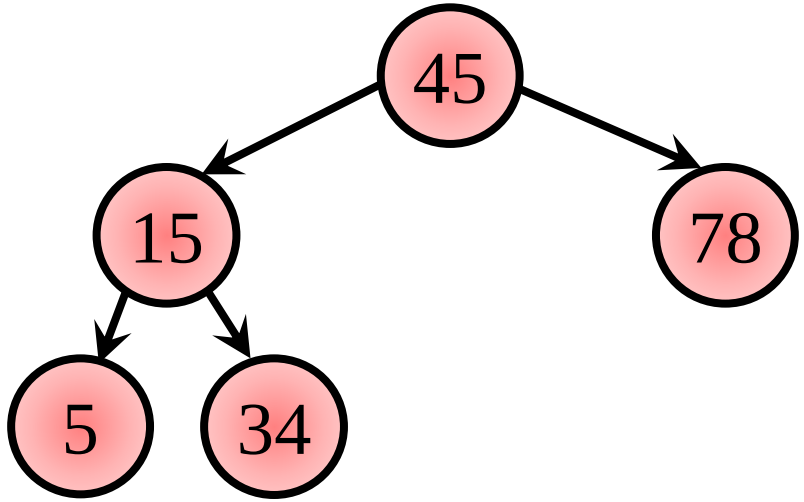
– Ví dụ 4: Cây nhị phân sau có phải là cây nhị phân tìm kiếm không?



– Trả lời:

Ví dụ cây nhị phân tìm kiếm

– Ví dụ 4: Cây nhị phân sau có phải là cây nhị phân tìm kiếm không?



– Trả lời:

+ Đây là một cây nhị phân tìm kiếm.

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

BINARY SEARCH TREE – PHẦN 02

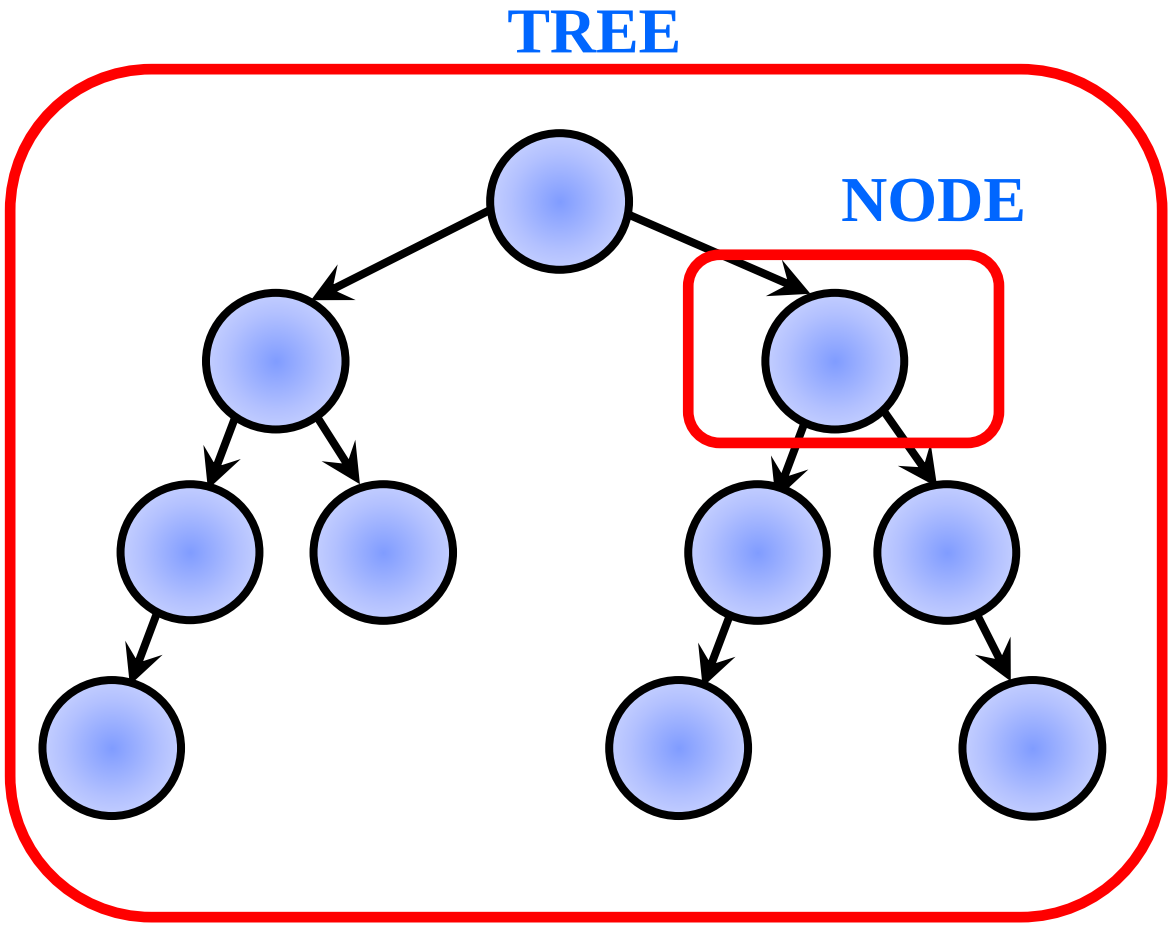
1. Hồ Thái Ngọc
2. ThS. Võ Duy Nguyên
3. TS. Nguyễn Tấn Trần Minh Khang

CẤU TRÚC DỮ LIỆU BST

Cấu trúc dữ liệu BST

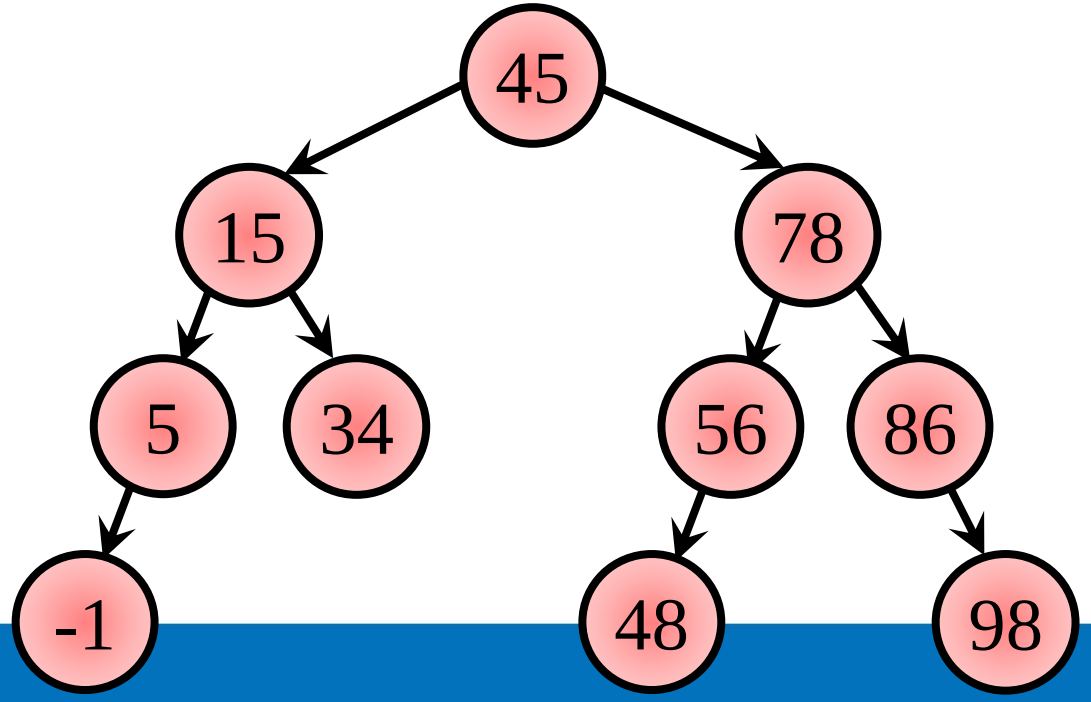
– Cấu trúc dữ liệu

```
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 BST

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

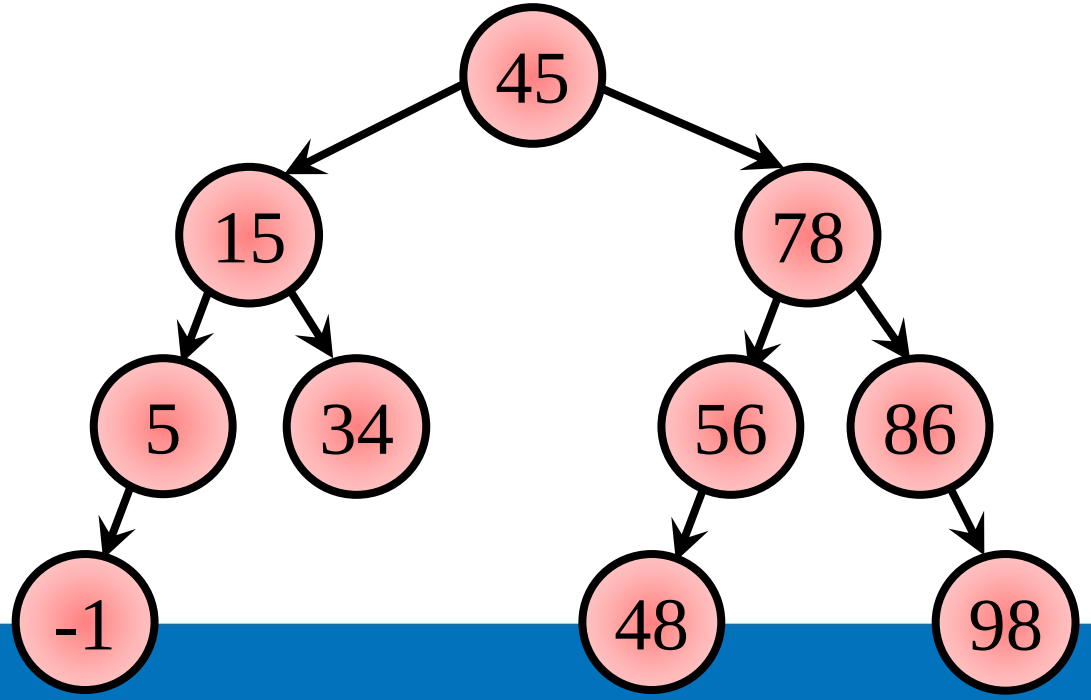


- Cấu trúc dữ liệu

```
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 BST

– Ví dụ 2: Hãy khai báo cấu trúc dữ liệu cho cây nhị phân tìm kiếm 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 BST

- Ví dụ 3: Hãy khai báo cấu trúc dữ liệu cho cây nhị phân tìm kiếm 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

BINARY SEARCH TREE – PHẦN 03

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

KHỞI TẠO CÂY NHỊ PHÂN TÌM KIẾM

Khởi tạo cây nhị phân tìm kiếm

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

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

KIỂM TRA CÂY NHỊ PHÂN TÌM KIẾM RỖNG

Kiểm tra cây nhị phân tìm kiếm rỗng

- Khái niệm: Kiểm tra cây nhị phân tìm kiếm 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 tìm kiếm rỗng

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

BINARY SEARCH TREE – PHẦN 04

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

TẠO NODE CHO CÂY NHỊ PHÂN TÌM KIẾM

Tạo node cho cây nhị phân

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

Tạo node cho BST 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 tìm kiếm 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 BST các số nguyên

Tạo node cho cây nhị phân

- Ví dụ 1: Định nghĩa hàm tạo node cho cây nhị phân tìm kiếm 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 BST các số nguyên

Tạo node cho cây nhị phân

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

Tạo node cho cây nhị phân

- Ví dụ 2: Định nghĩa hàm tạo node cho cây nhị phân tìm kiếm 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 BST các số thực

Tạo node cho cây nhị phân

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

Tạo node cho cây nhị phân

– Ví dụ 3: Định nghĩa hàm tạo node cho cây nhị phân tìm kiếm 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 BST 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

BINARY SEARCH TREE – PHẦN 05

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

THÊM MỘT GIÁ TRỊ VÀO TRONG CÂY NHỊ PHÂN TÌM KIẾM

Thêm một giá trị vào trong BST

- Khái niệm: Thêm một giá trị vào trong cây nhị phân tìm kiếm là thêm thông tin vào cây.
- Giá trị trả về: Hàm thêm một giá trị vào trong cây nhị phân tìm kiếm trả về một trong 3 giá trị -1, 1 với ý nghĩa như sau:
  - Giá trị $+1$: Thêm thành công.
  - Giá trị $+0$: Giá trị cần thêm trùng với một giá trị có sẵn trong cây.
  - Giá trị -1: Không đủ bộ nhớ.

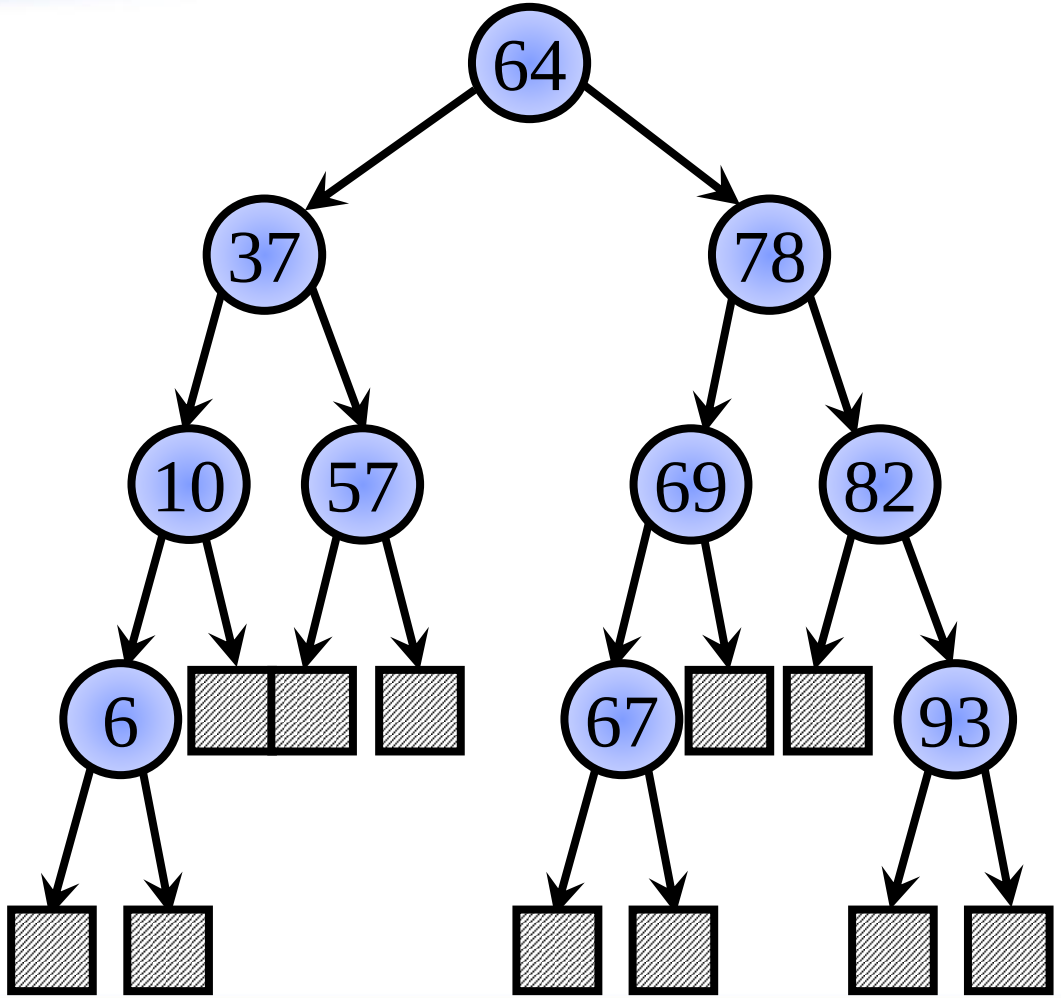
Thêm một giá trị vào trong BST

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

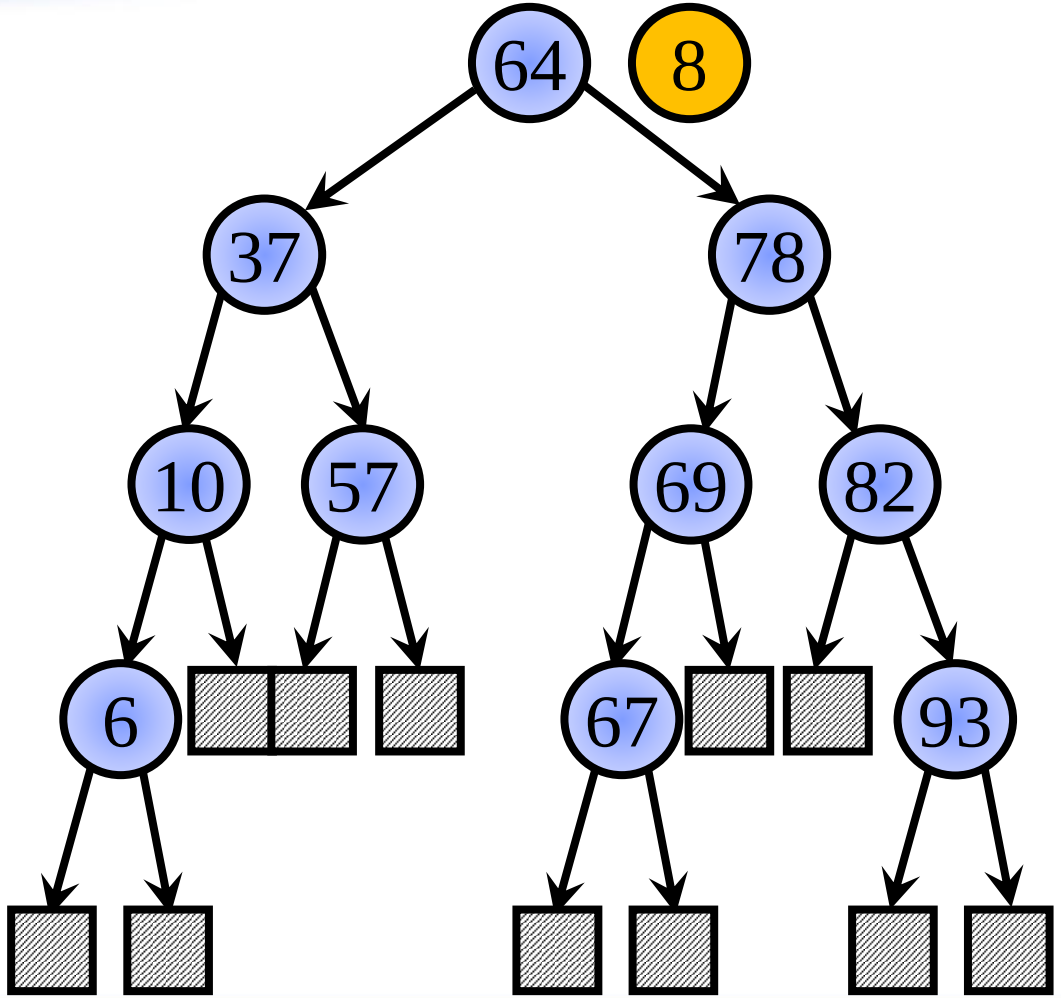
- Cấu trúc dữ liệu

```
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;
```

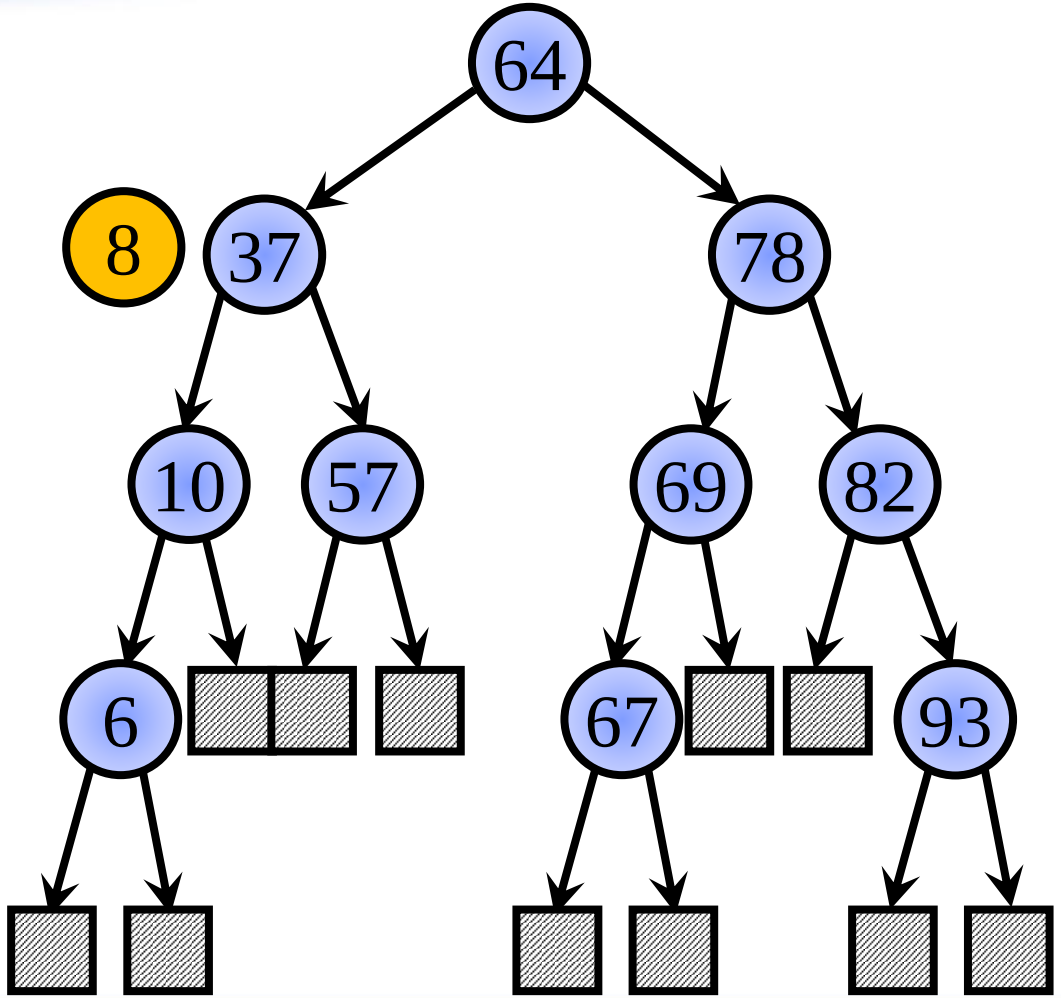
Thêm một giá trị vào trong BST



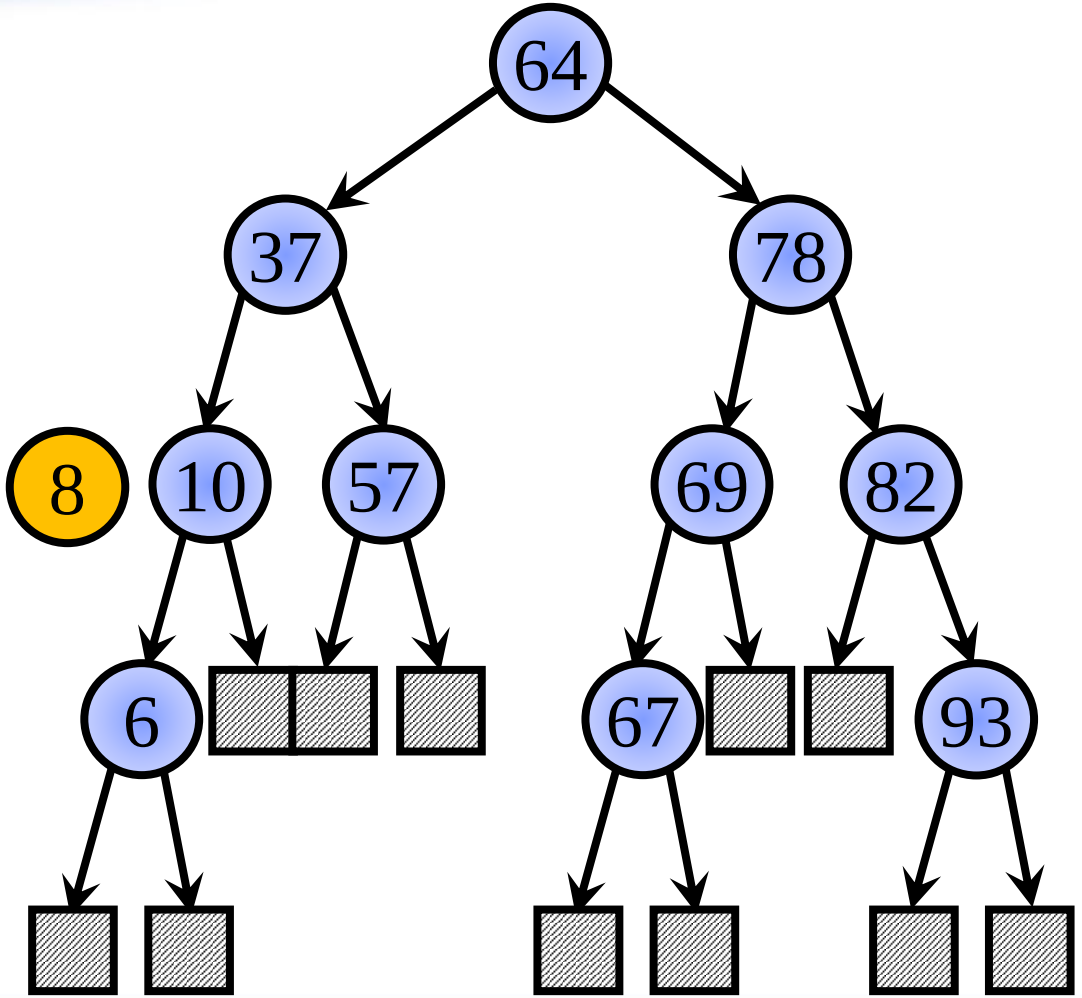
Thêm một giá trị vào trong BST



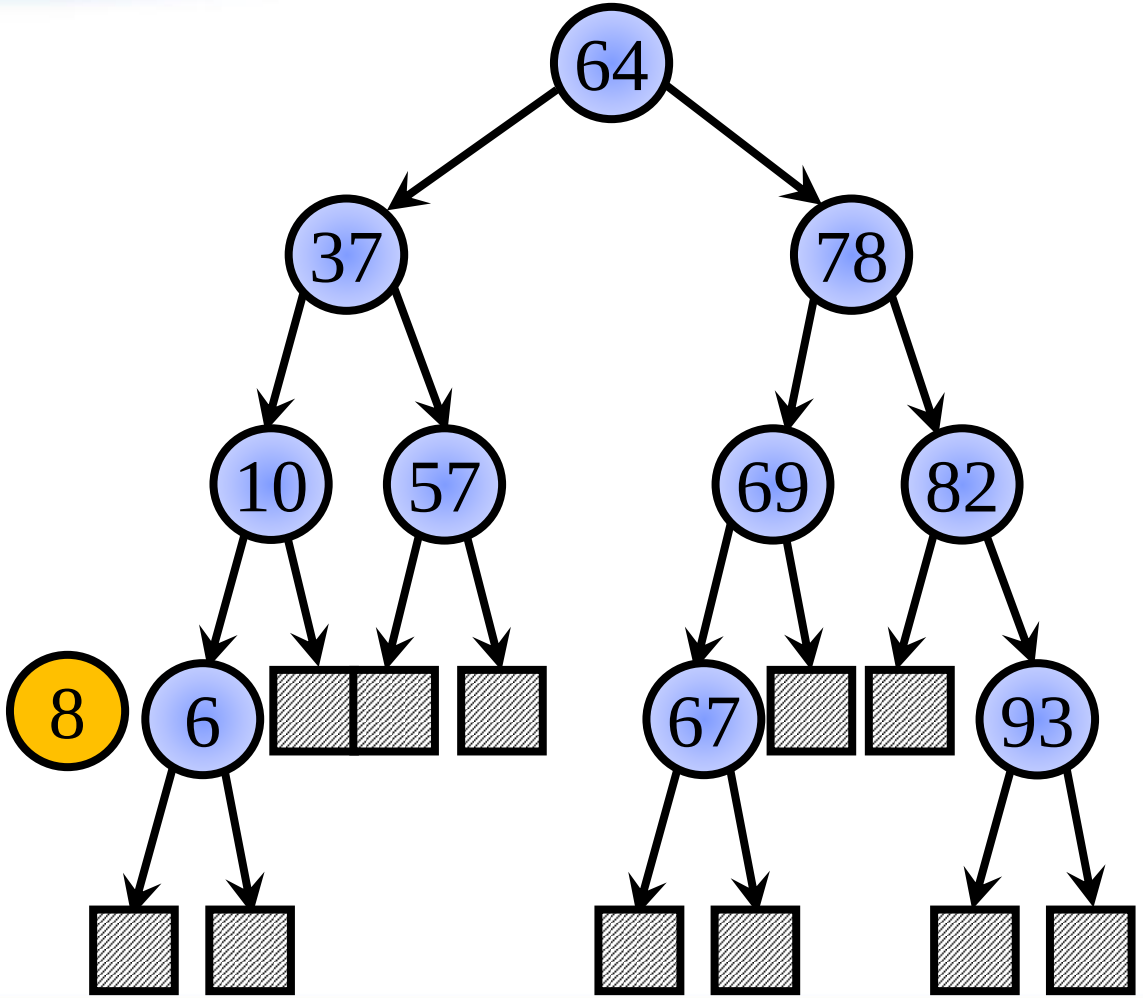
Thêm một giá trị vào trong cây



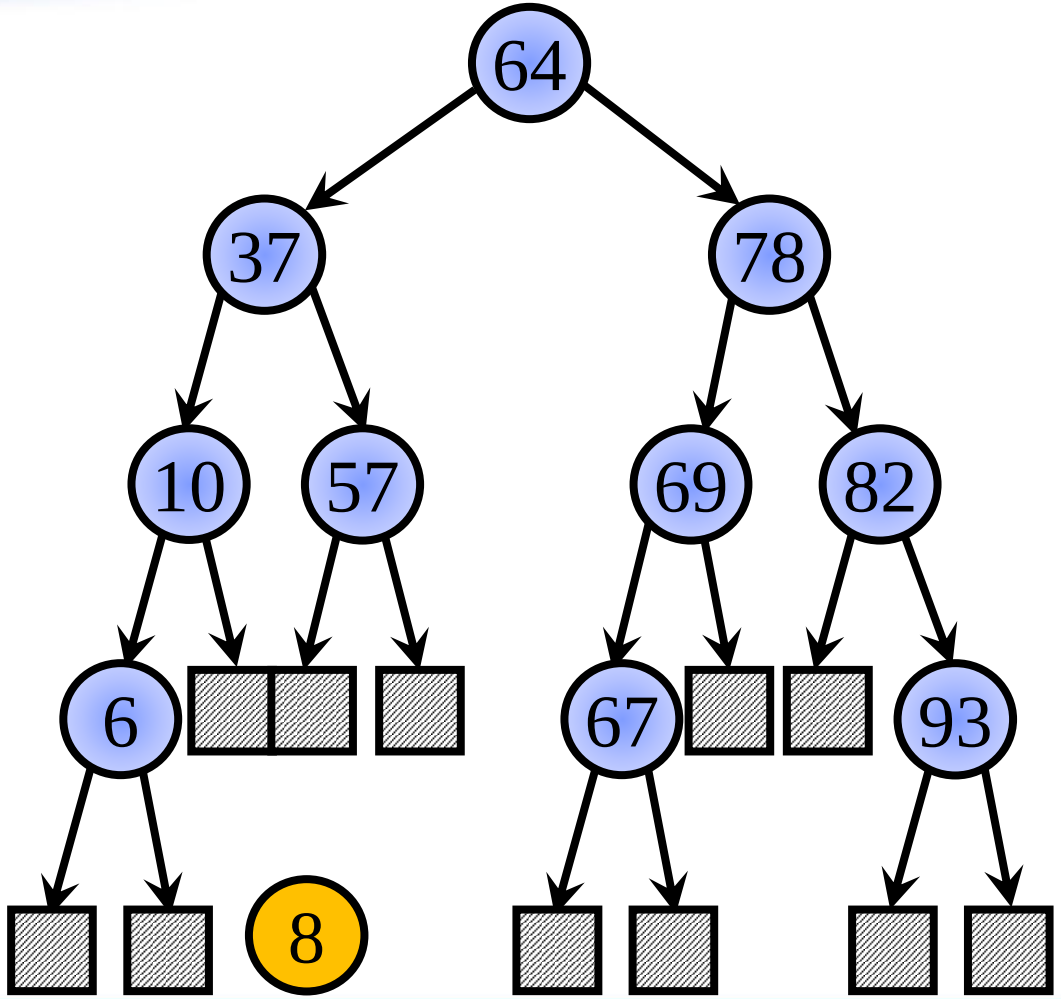
Thêm một giá trị vào trong cây



Thêm một giá trị vào trong cây

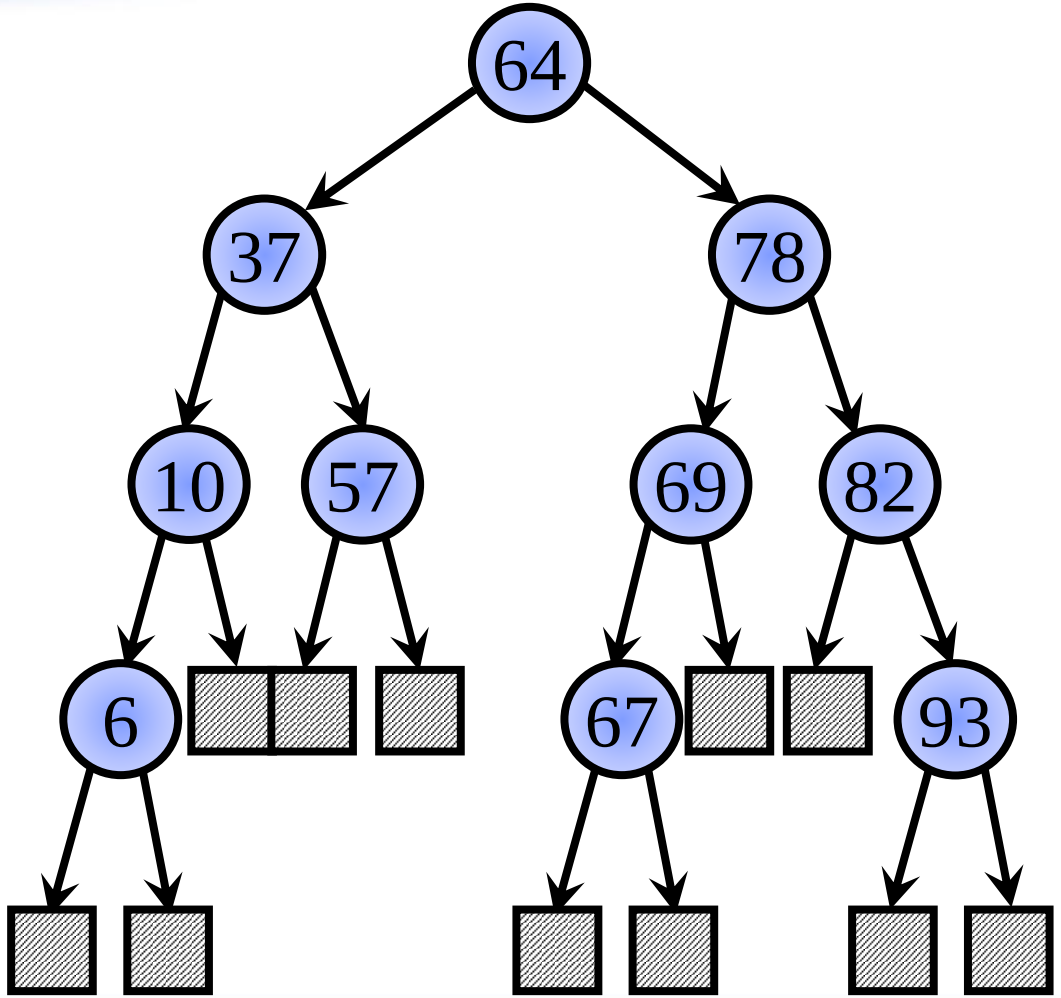


Thêm một giá trị vào trong cây

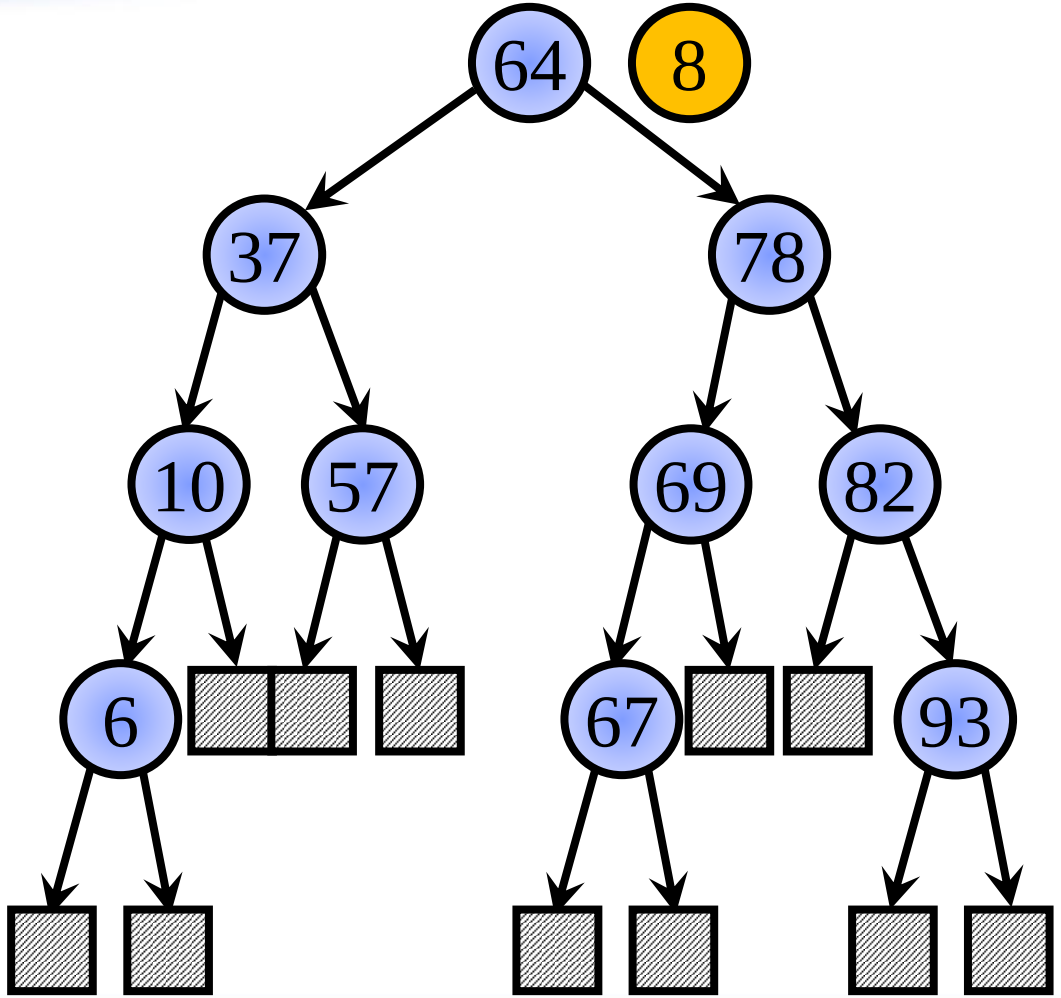


```
11. int InsertNode(TREE &t,int x)
12. {
13.     if(          )
14.     {
20.     }
25. }
```

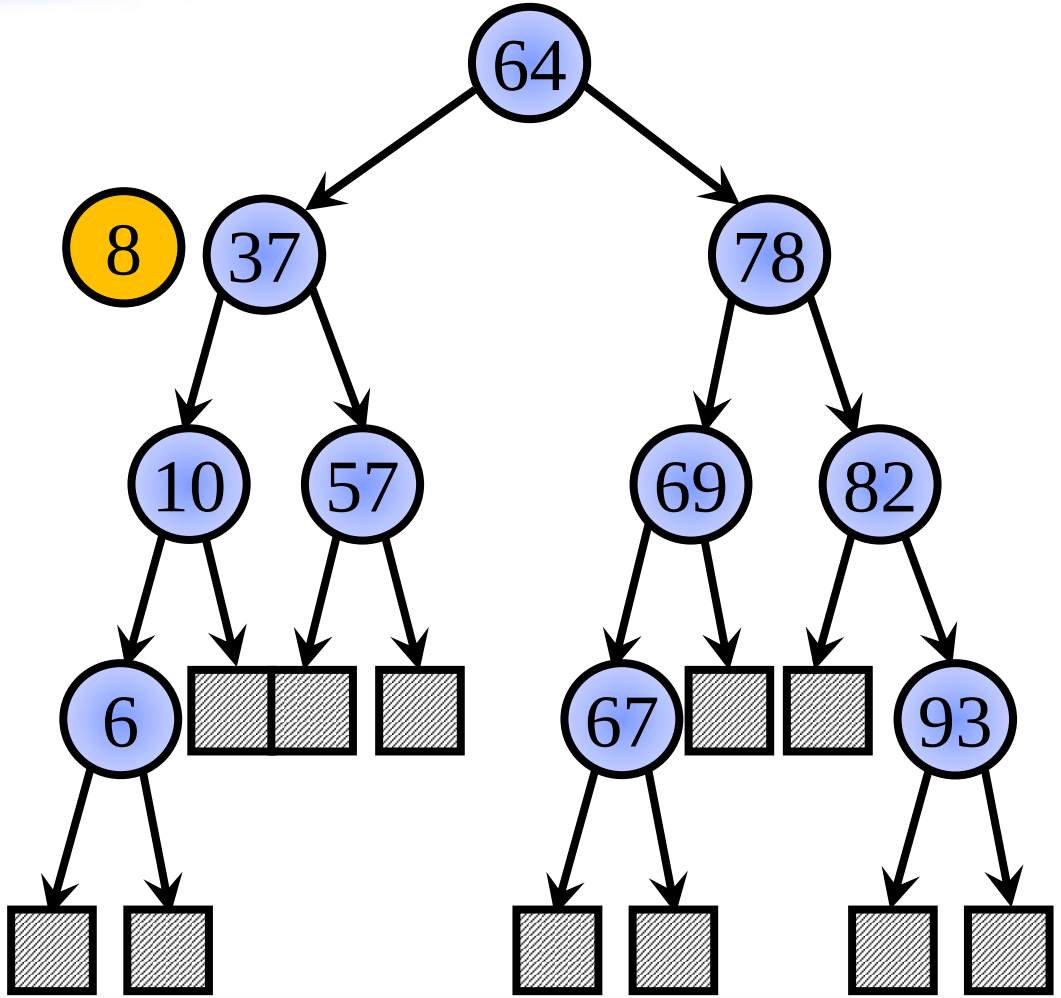
Thêm một giá trị vào trong BST



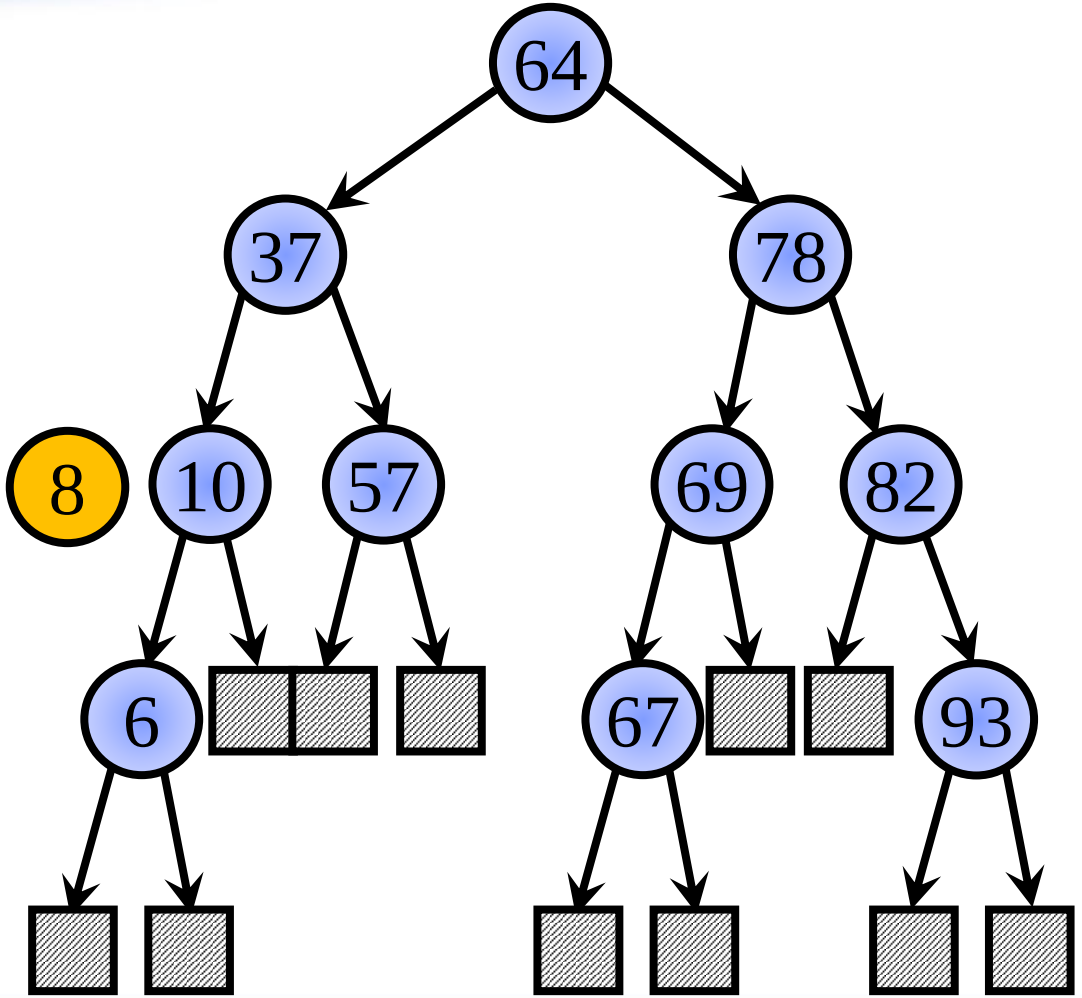
Thêm một giá trị vào trong BST



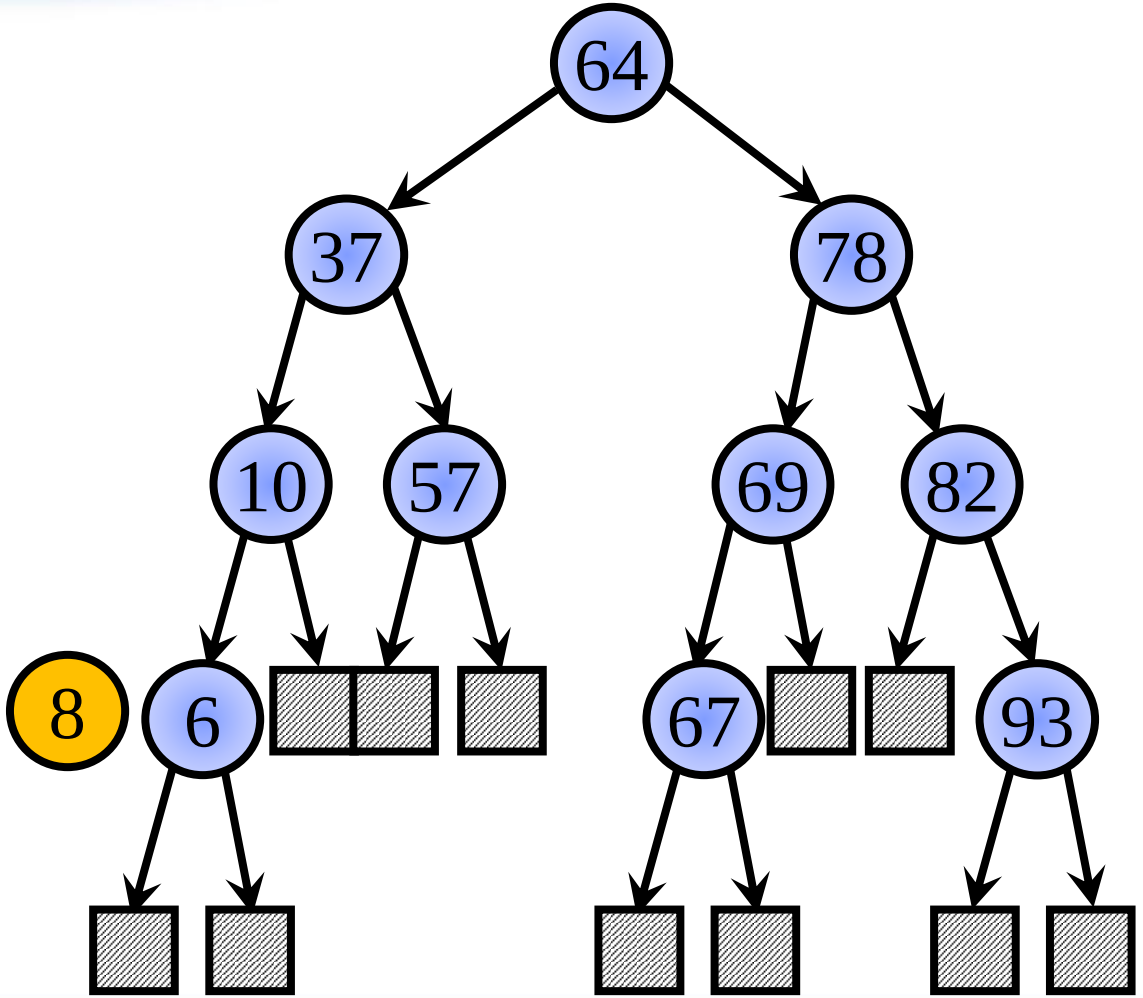
Thêm một giá trị vào trong cây



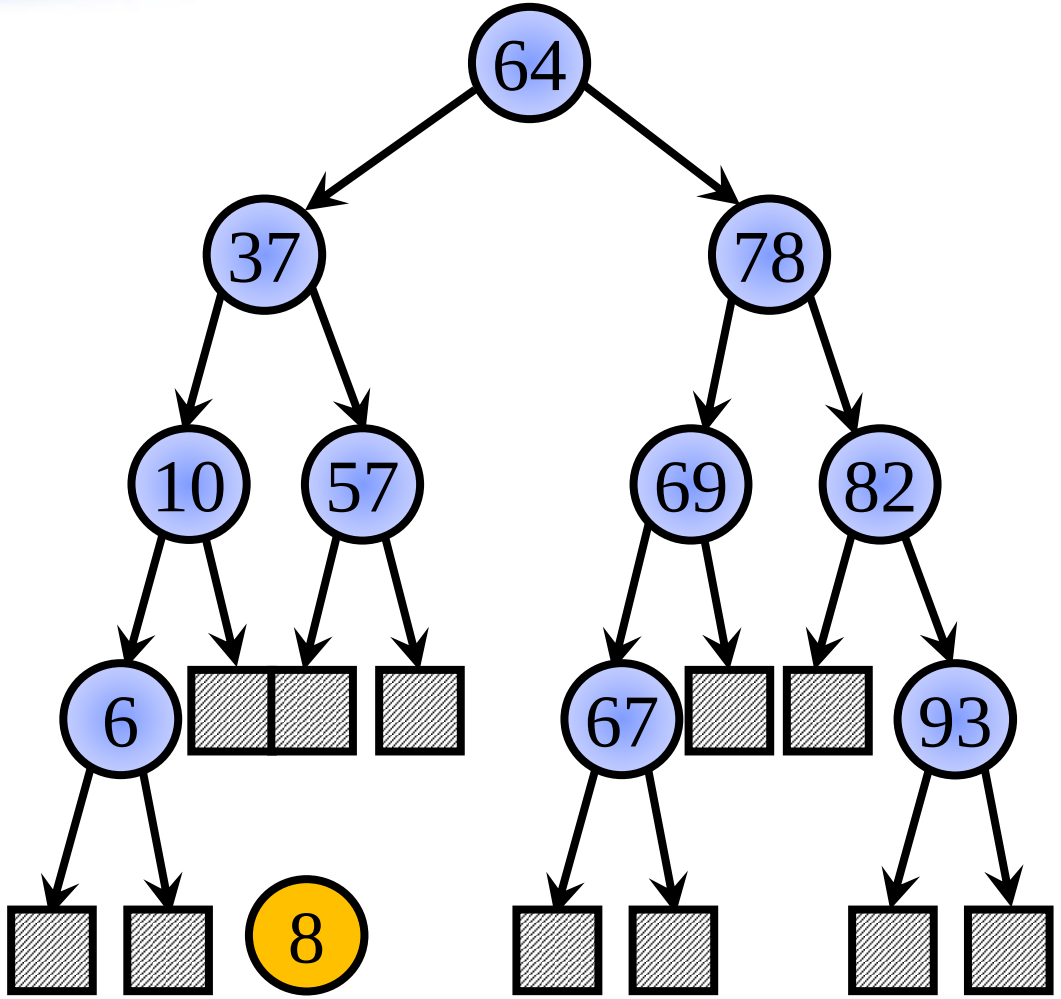
Thêm một giá trị vào trong cây



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.     {
20.     }
25. }
```

```
11. int InsertNode(TREE &t,int x)
12. {
13.     if(t!=NULL)
14.     {
15.         if(x > t->info)
20.     }
25. }
```

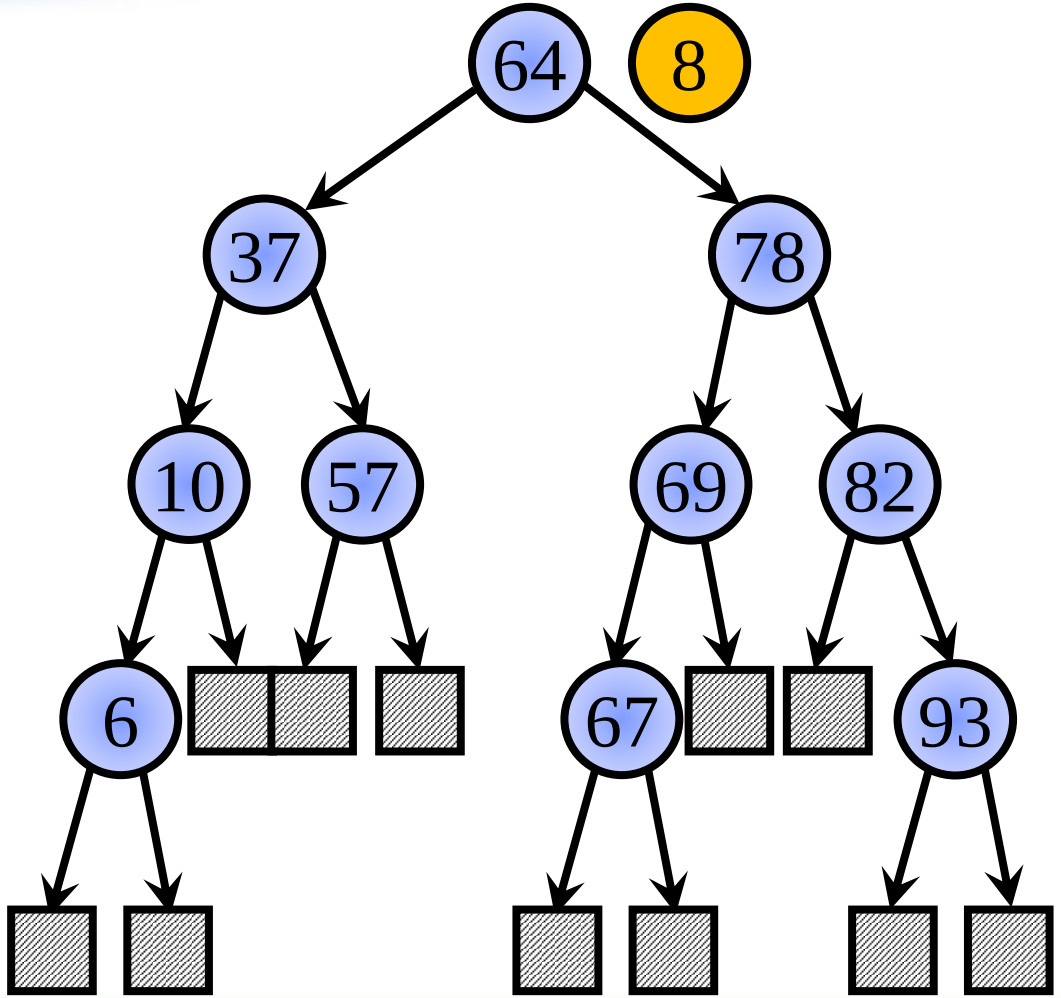
```
11. int InsertNode(TREE &t,int x)
12. {
13.     if(t!=NULL)
14.     {
15.         if(x > t->info)
16.             return InsertNode(t->pRight,x);
20.     }
25. }
```

```
11. int InsertNode(TREE &t,int x)
12. {
13.     if(t!=NULL)
14.     {
15.         if(x > t->info)
16.             return InsertNode(t->pRight,x);
17.         if(x < t->info)
20.     }
25. }
```

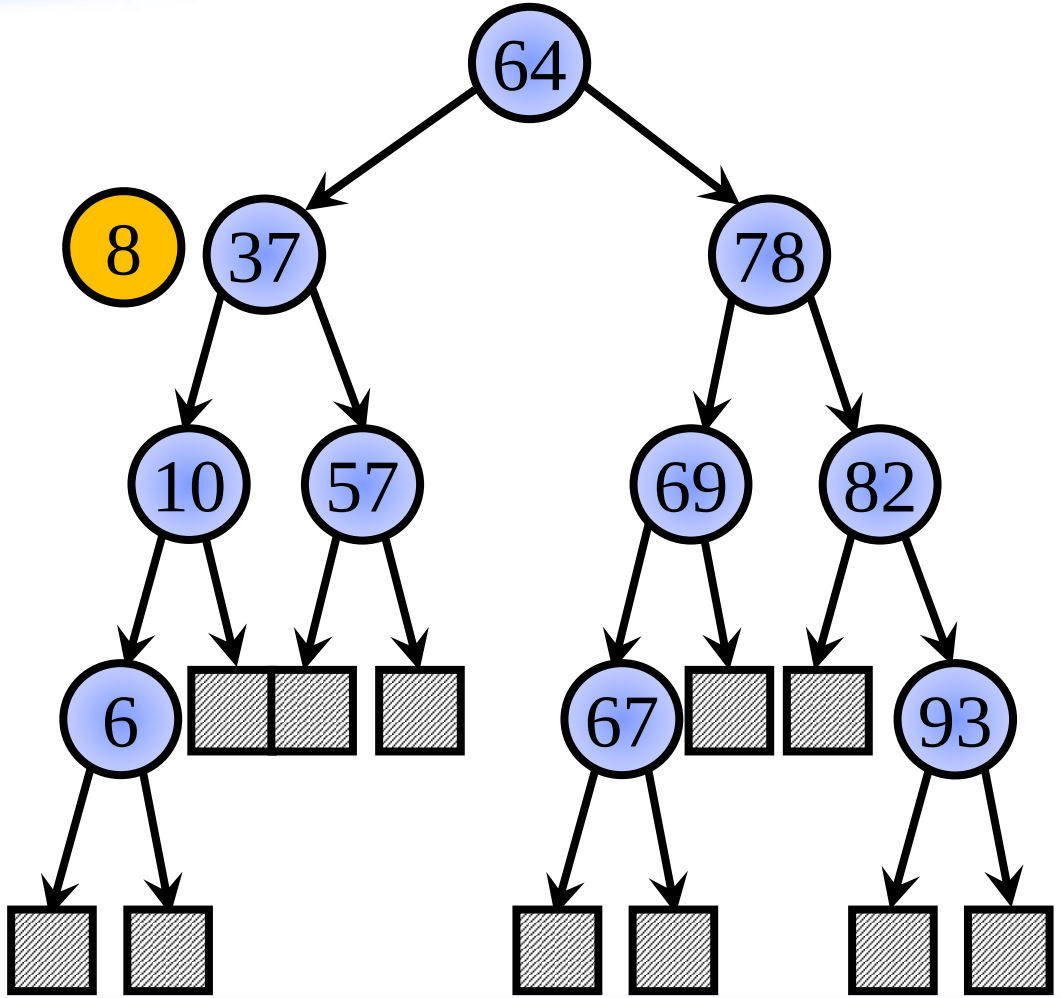
```
11. int InsertNode(TREE &t,int x)
12. {
13.     if(t!=NULL)
14.     {
15.         if(x > t->info)
16.             return InsertNode(t->pRight,x);
17.         if(x < t->info)
18.             return InsertNode(t->pLeft,x);
20.     }
25. }
```

```
11. int InsertNode(TREE &t,int x)
12. {
13.     if(t!=NULL)
14.     {
15.         if(x > t->info)
16.             return InsertNode(t->pRight,x);
17.         if(x < t->info)
18.             return InsertNode(t->pLeft,x);
19.         return 0;
20.     }
25. }
```

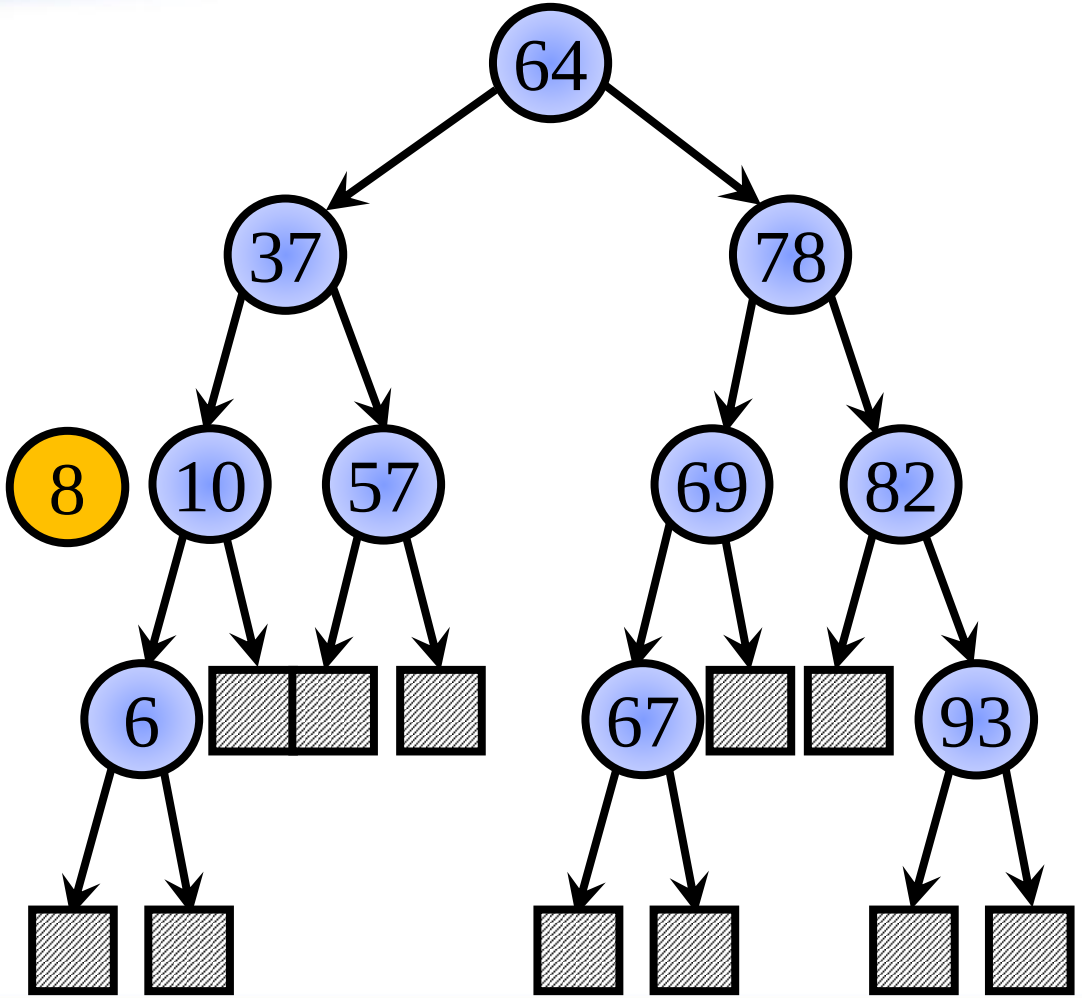
Thêm một giá trị vào trong cây



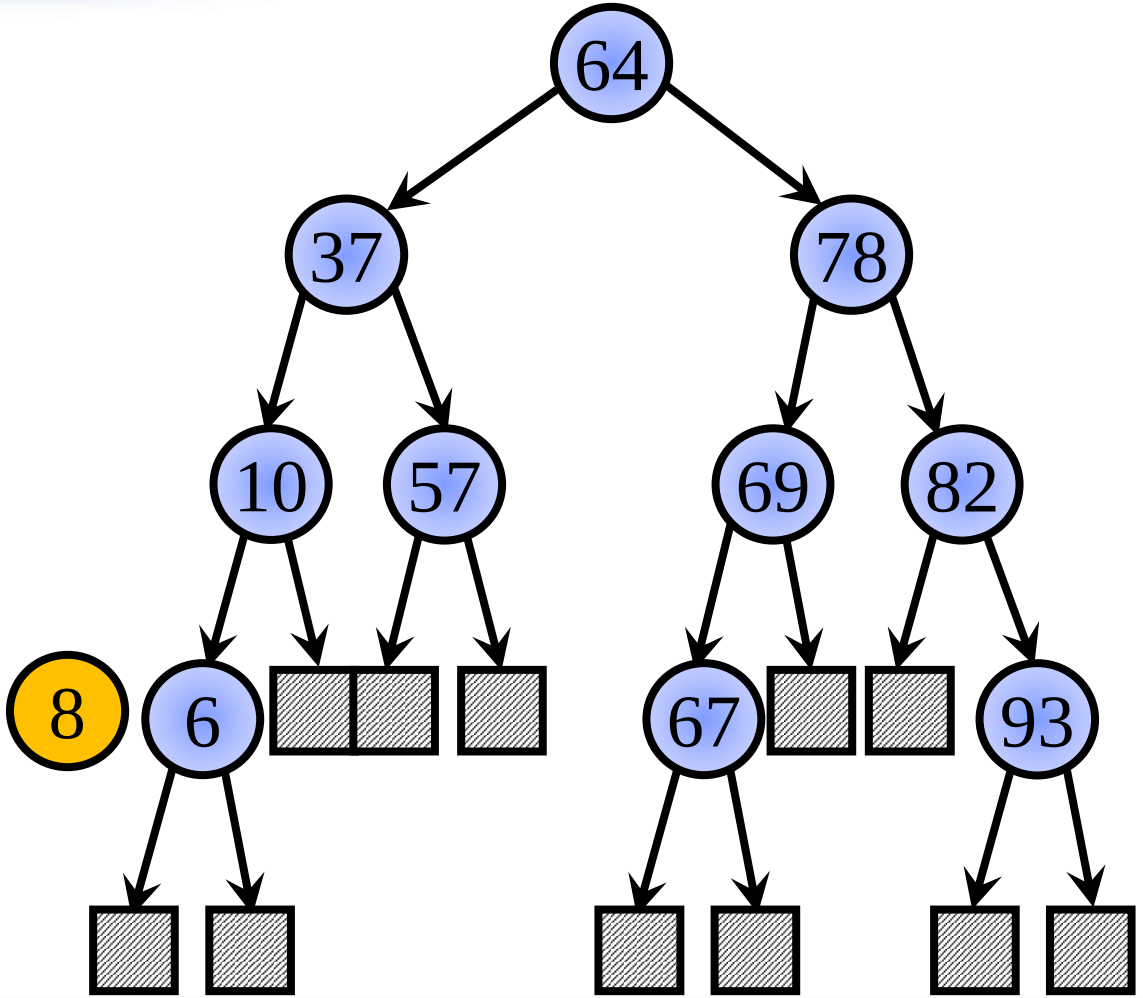
Thêm một giá trị vào trong cây



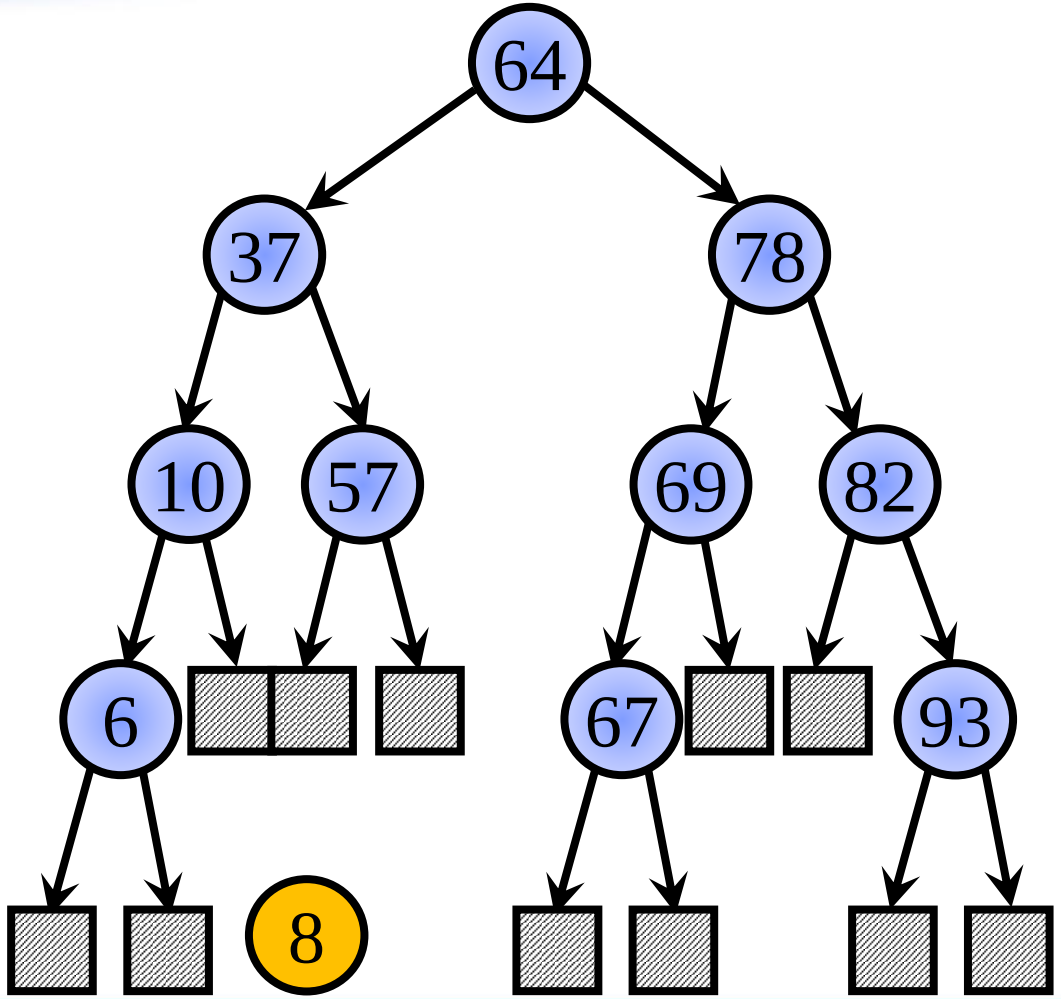
Thêm một giá trị vào trong cây



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.         if(x > t->info)
16.             return InsertNode(t->pRight,x);
17.         if(x < t->info)
18.             return InsertNode(t->pLeft,x);
19.         return 0;
20.     }
21.     t = GetNode(x);
22.     if(t==NULL)
23.         return -1;
24.     return 1;
25. }
```

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

BINARY SEARCH TREE – PHẦN 06

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

NHẬP CÂY NHỊ PHÂN TÌM KIẾM CÁC SỐ NGUYÊN TỪ FILE

Nhập BST các số nguyên từ file

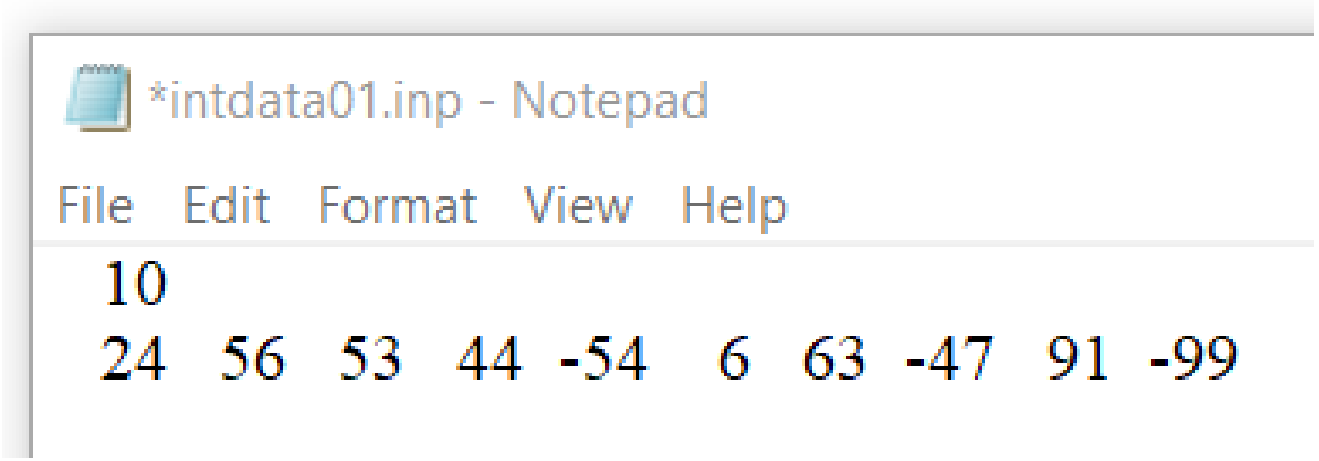
- Bài toán: Hãy định nghĩa hàm nhập cây nhị phân tìm kiếm các số nguyên từ file.

- 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;
```

Nhập BST các số nguyên từ file

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



```
int Nhap(TREE  t, string filename)
{
}
```

Nhập BST các số nguyên từ file

```
int Nhap(TREE& t, string filename)
{
}
```

Nhập BST các số nguyên từ file

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

Nhập BST các 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 BST các số nguyên từ file

```
11. int Nhap(TREE& t, string filename)
12. {
13.     ifstream fi(filename);
24.     return 1;
25. }
```

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

Nhập BST các 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 BST các số nguyên từ file

```
11. int Nhap(TREE& t, string filename)
12. {
13.     ifstream fi(filename);
14.     if (fi.fail()==true)
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 BST các số nguyên từ file

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

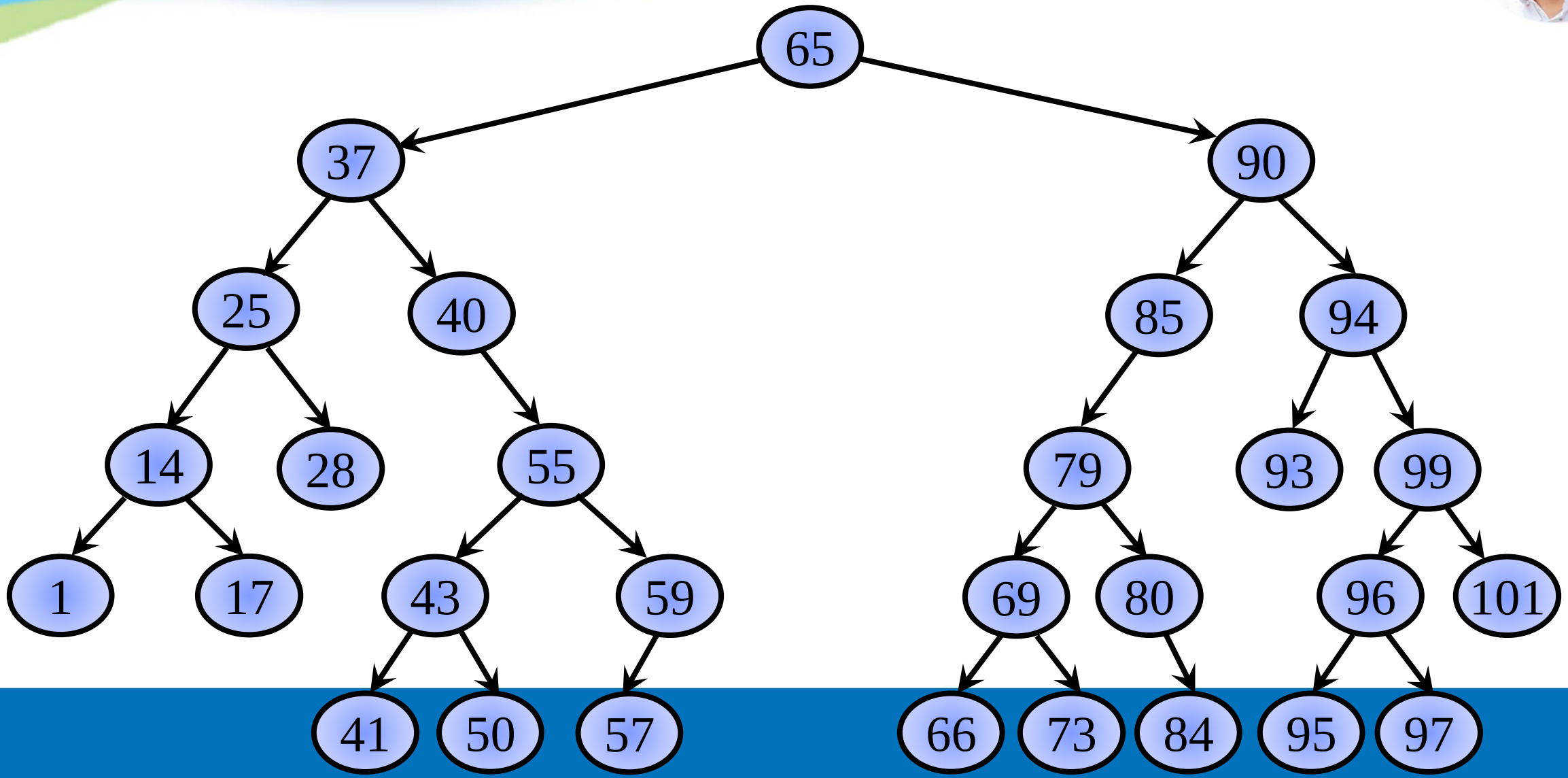
TS. Nguyễn Tấn Trần Minh Khang

BINARY SEARCH TREE – PHẦN 07

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

NHÌN CÂY NHỊ PHÂN TÌM KIẾM DƯỚI CON MẮT ĐỆ QUY

Nhìn BST 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

BINARY SEARCH TREE – PHẦN 08

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

CÁC PHƯƠNG PHÁP DUYỆT CÂY TÌM KIẾM

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).

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

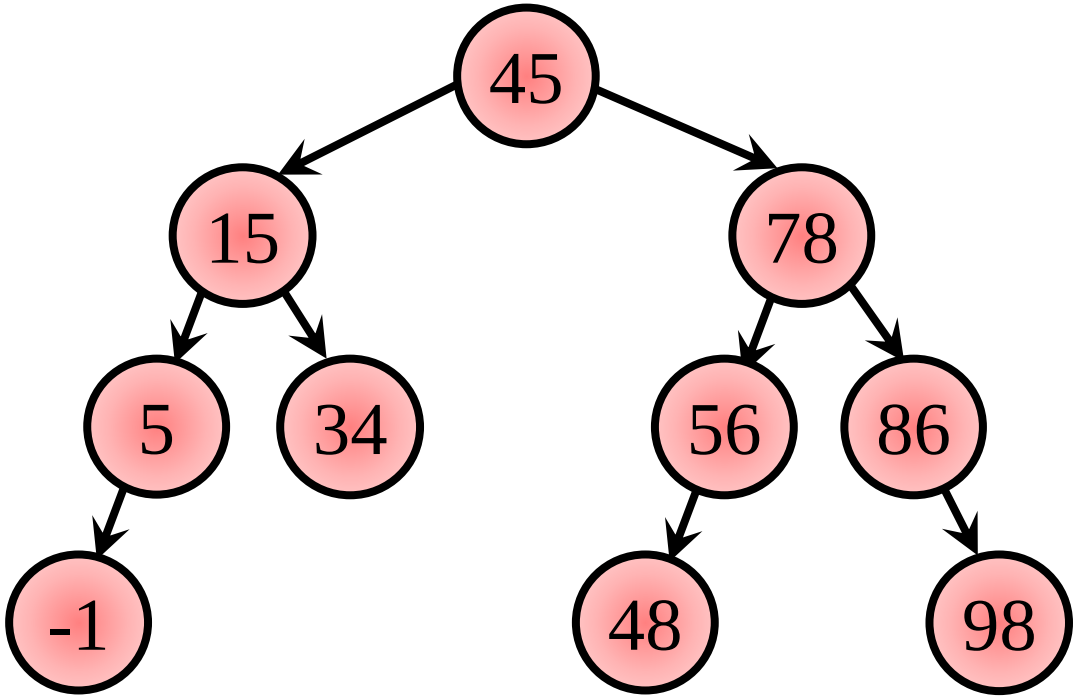
Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

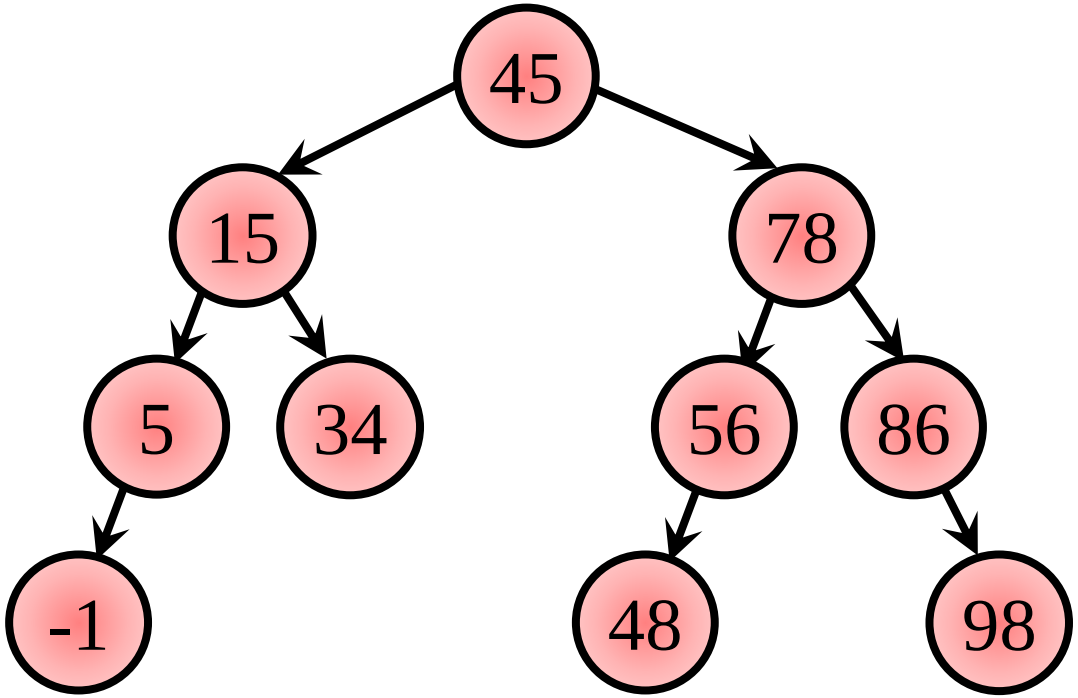
PHƯƠNG PHÁP 1: LEFT NODE RIGHT

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: -1, 5, 15, 34, 45, 48, 56, 78, 86, 98.

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

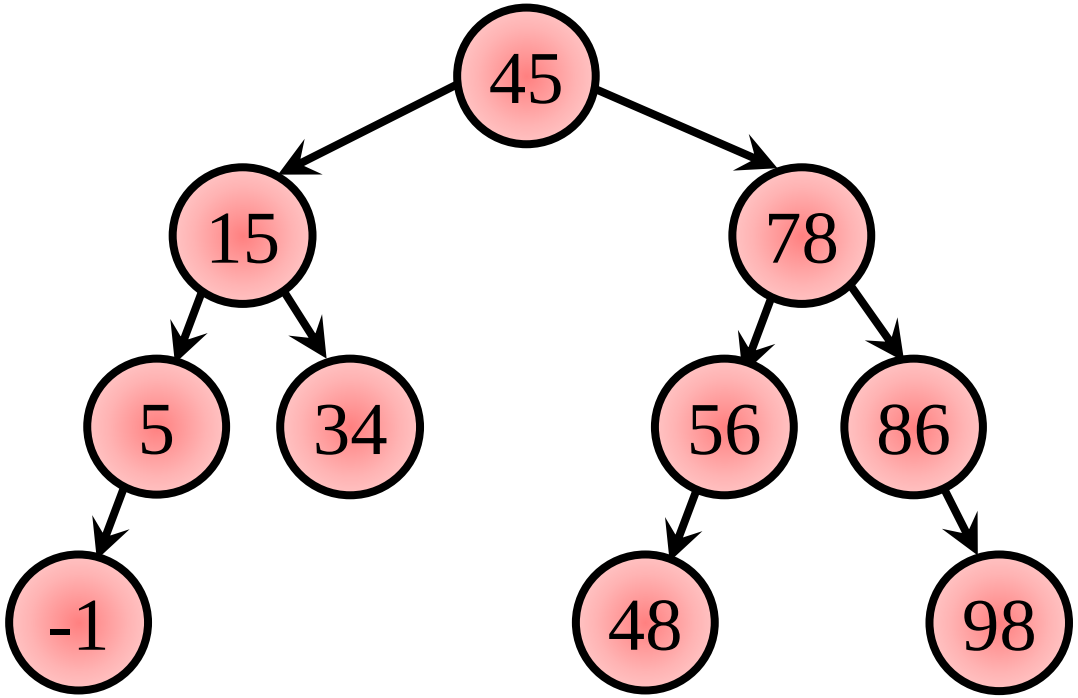
TS. Nguyễn Tấn Trần Minh Khang

BINARY SEARCH TREE – PHẦN 09

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

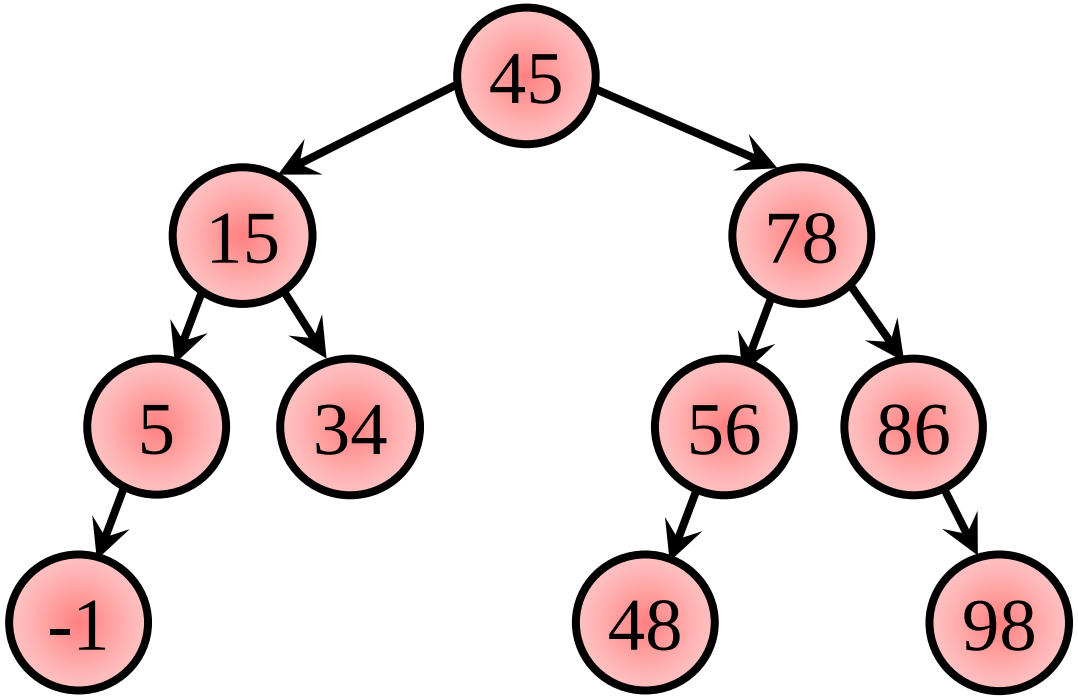
PHƯƠNG PHÁP 2: RIGHT NODE LEFT

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: 98, 86, 78, 56, 48, 45, 34, 15, 5, -1.

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

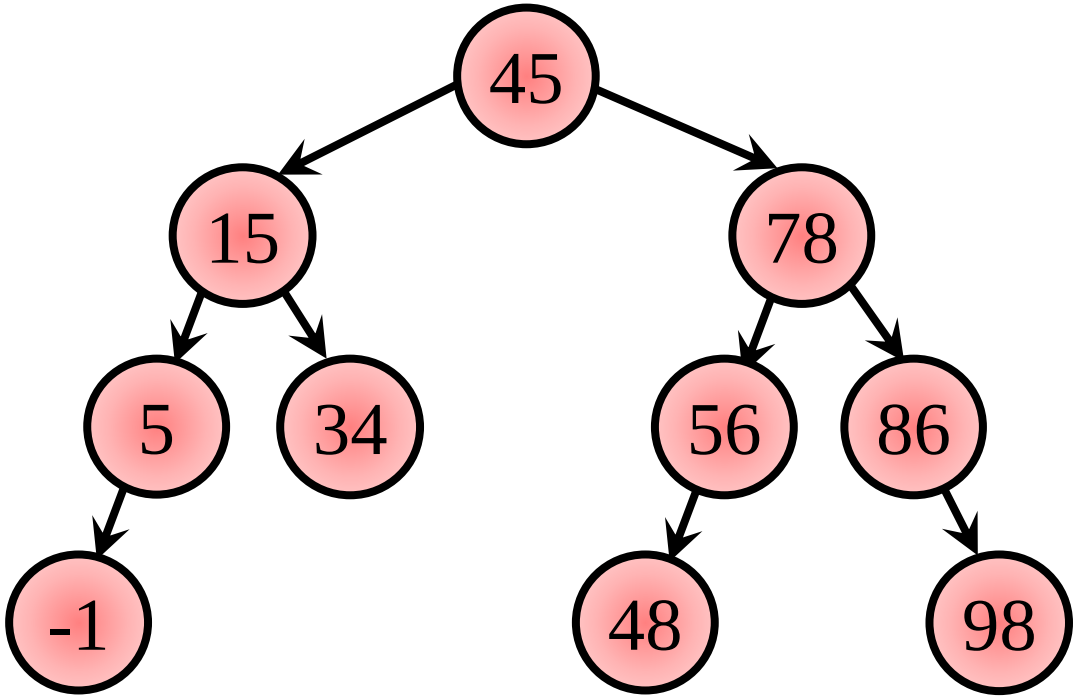
TS. Nguyễn Tấn Trần Minh Khang

BINARY SEARCH TREE – PHẦN 10

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

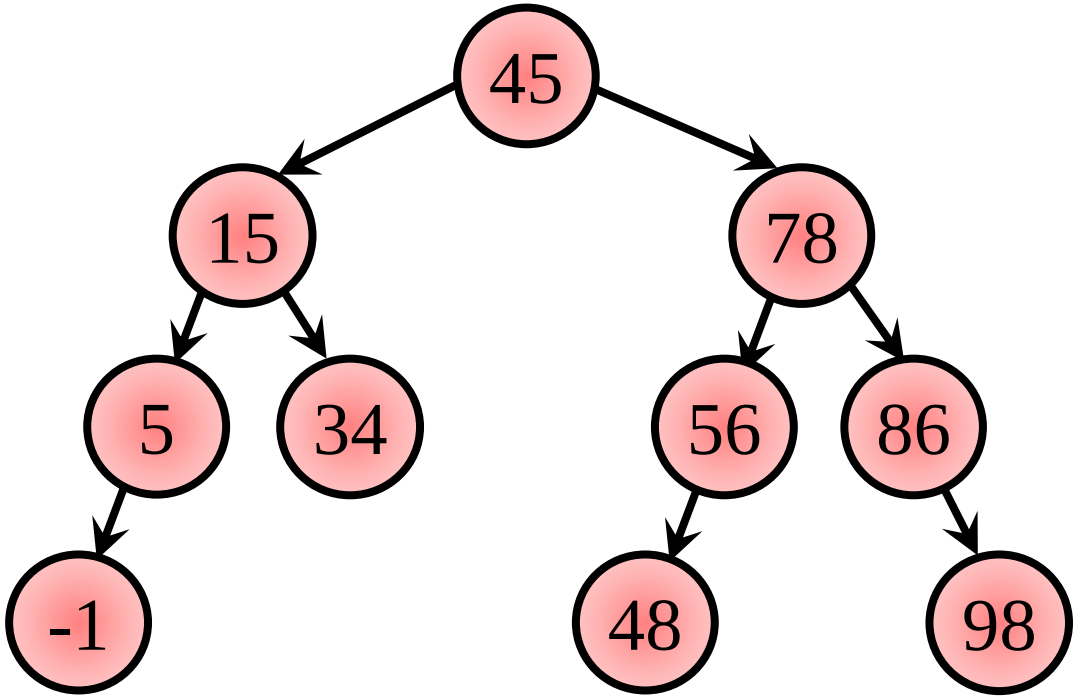
PHƯƠNG PHÁP 3: NODE LEFT RIGHT

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

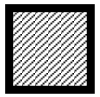


- Duyệt cây thep phương pháp NLR (Node Left Right) là: duyệt node gốc trong cây trước, sau đó duyệt tới cây con trái và cuối cùng là duyệt cây con phải.
- Cách thức duyệt cây con trái và duyệt cây con phải cũng giống như cách thức duyệt cây cha.
- Kết quả duyệt cây bên trái: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.

Phương pháp node left right

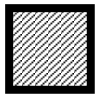
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.

Phương pháp node left right



- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Cây ban đầu rỗng

Phương pháp node left right



- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 45

Phương pháp node left right

– Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.

– Thêm 45

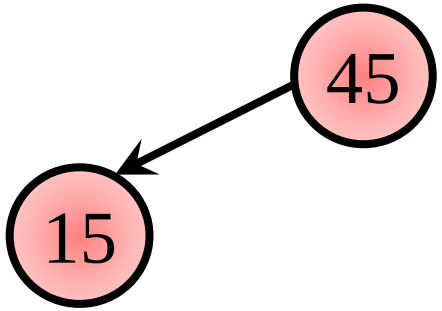
Phương pháp node left right

45

- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 15

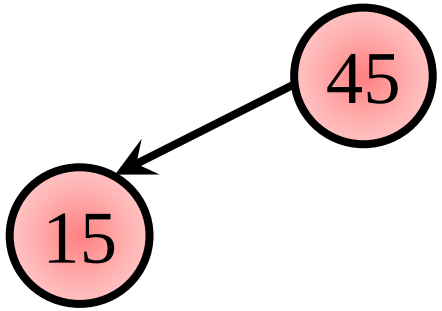
15

Phương pháp node left right



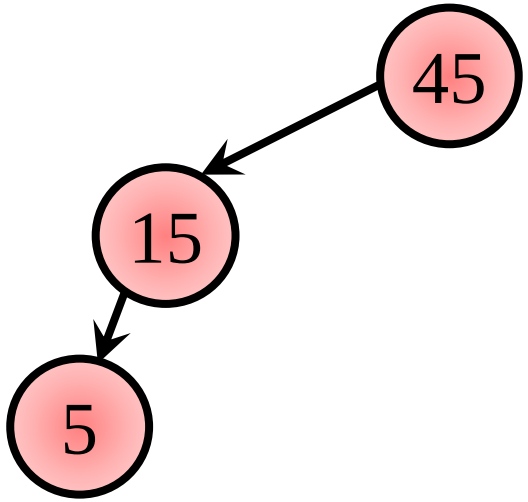
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 15

Phương pháp node left right



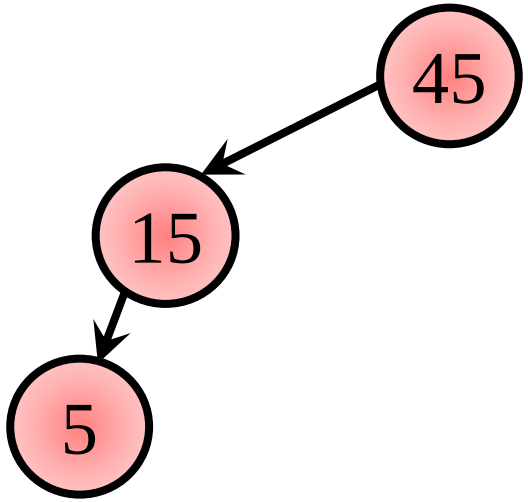
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 5

Phương pháp node left right



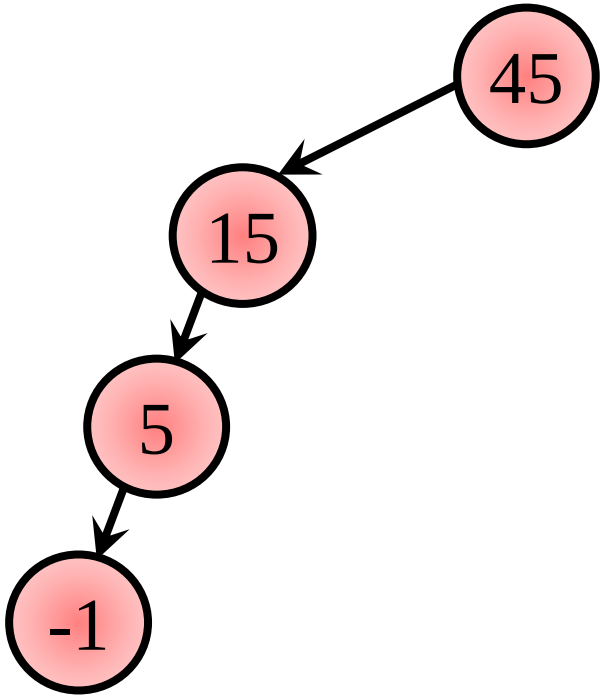
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 5

Phương pháp node left right



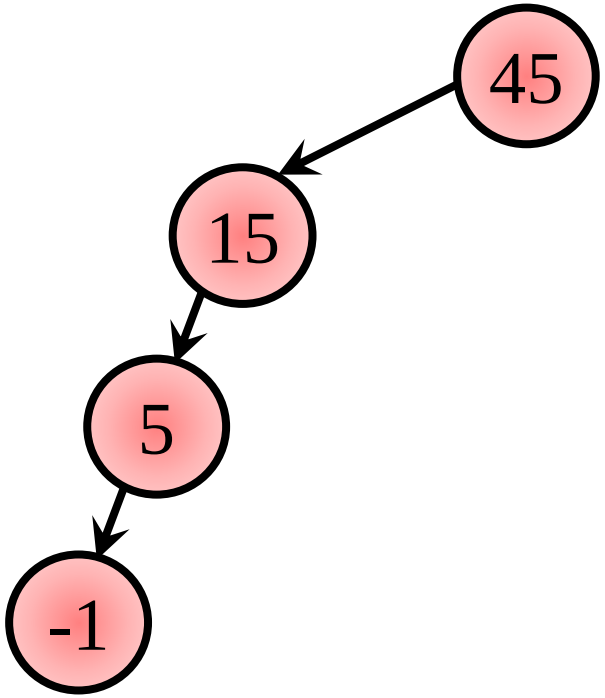
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm -1

Phương pháp node left right



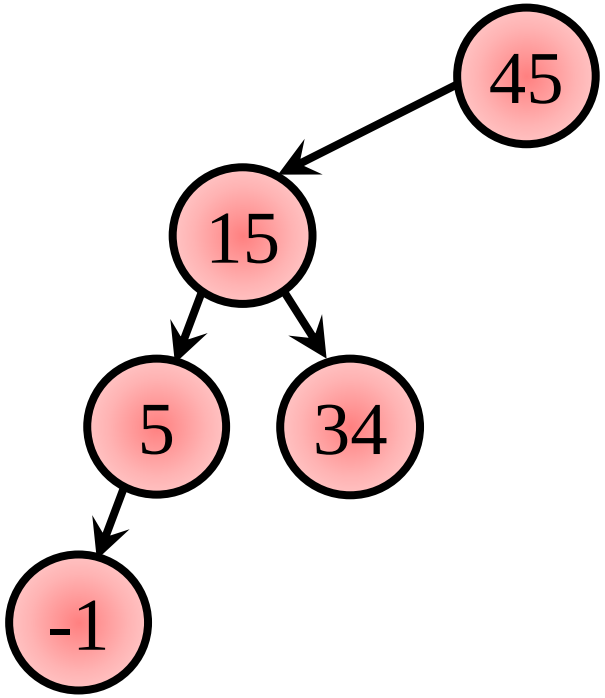
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm -1

Phương pháp node left right



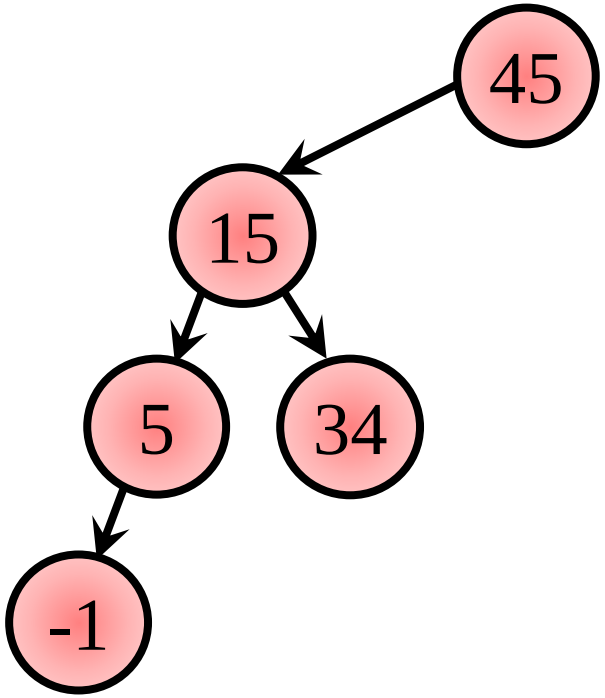
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 34

Phương pháp node left right



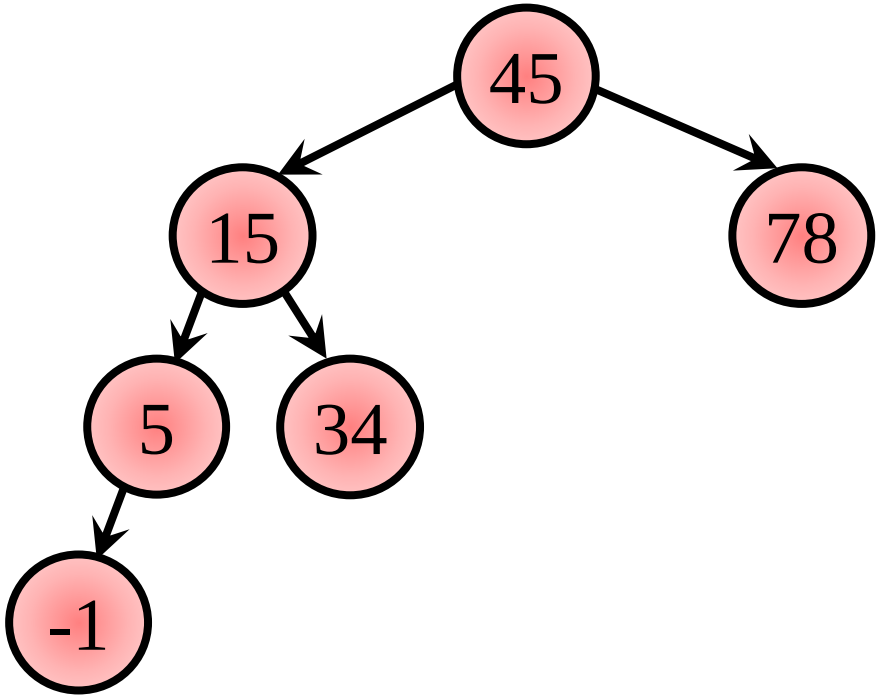
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 34

Phương pháp node left right



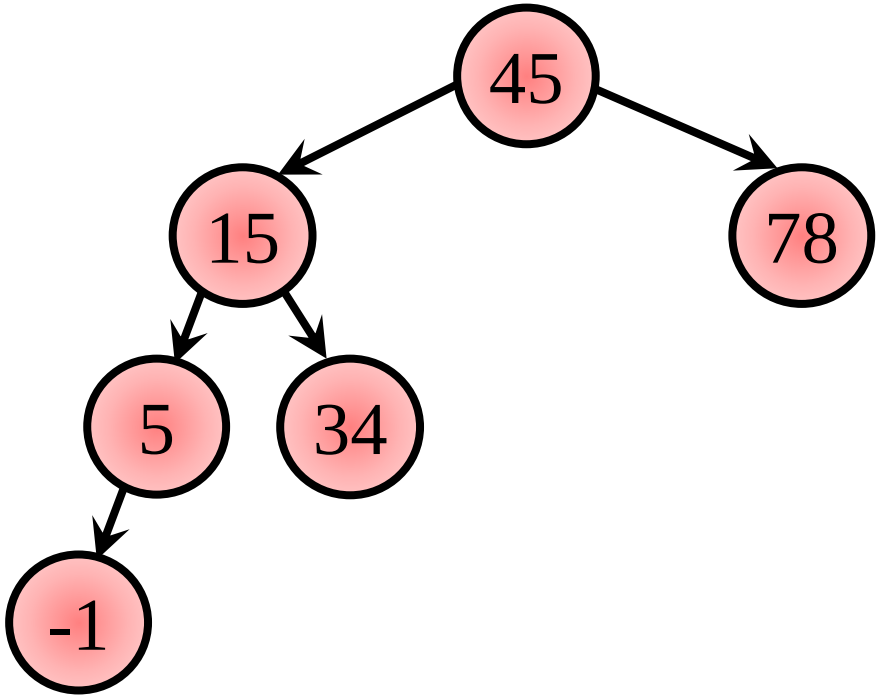
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 78

Phương pháp node left right



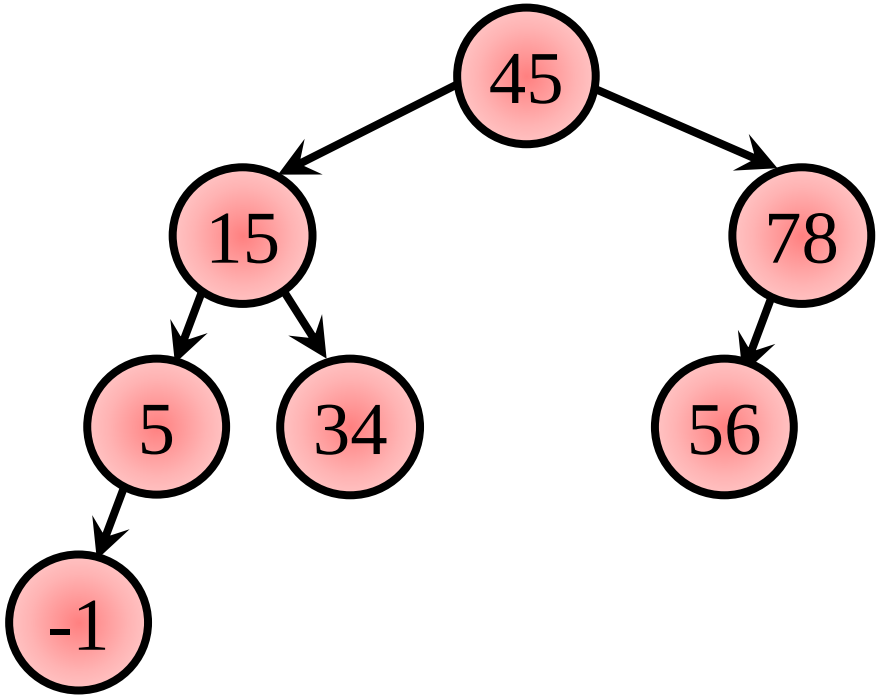
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 78

Phương pháp node left right



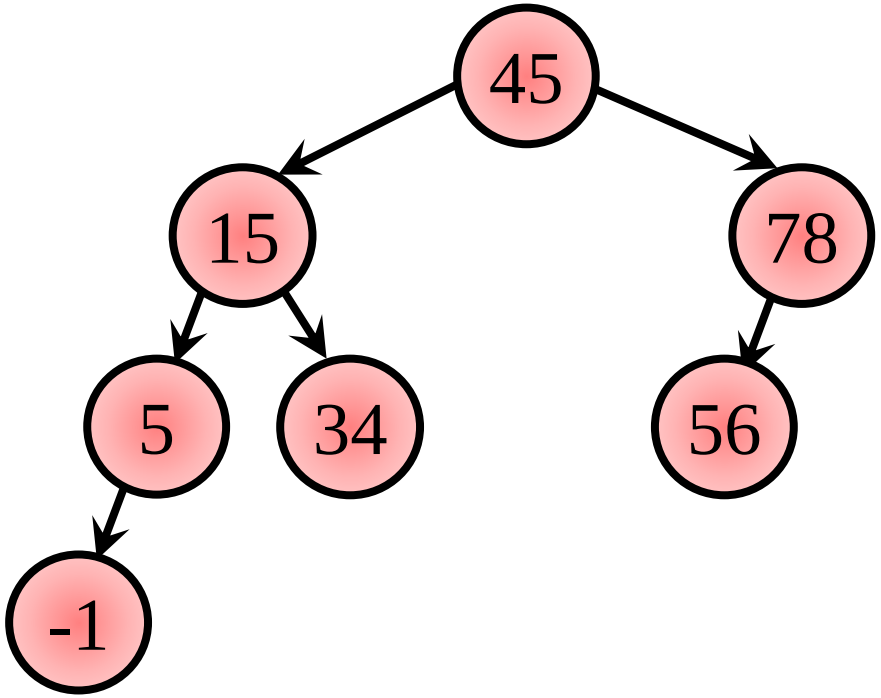
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 56

Phương pháp node left right



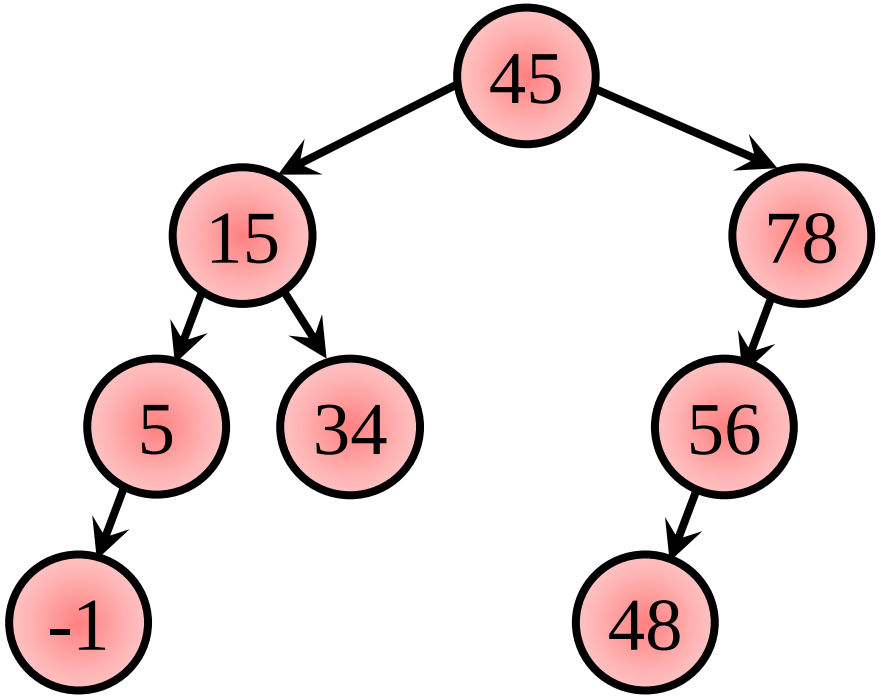
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 56

Phương pháp node left right



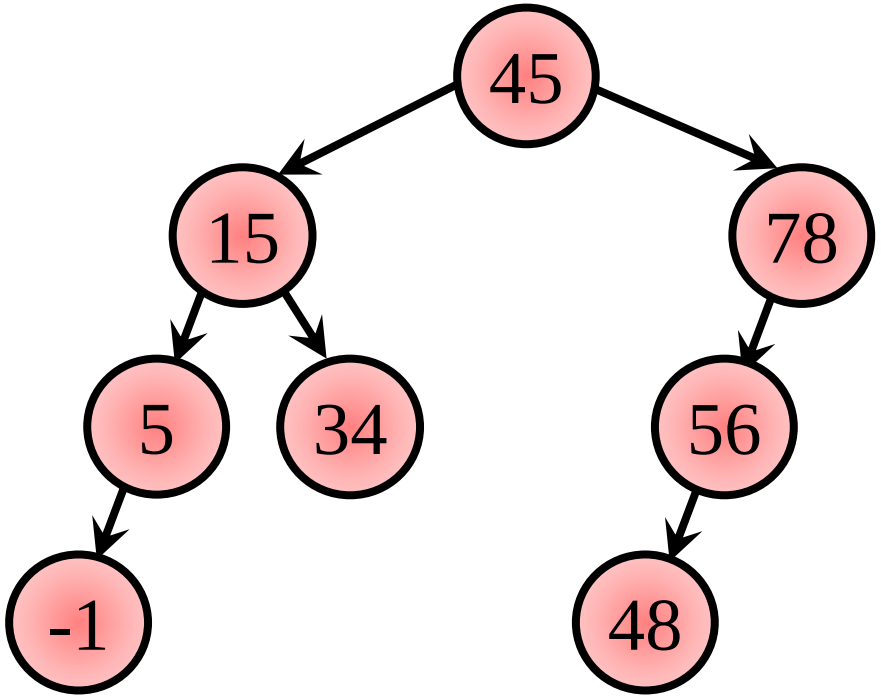
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 48

Phương pháp node left right

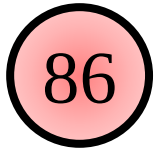


- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 48

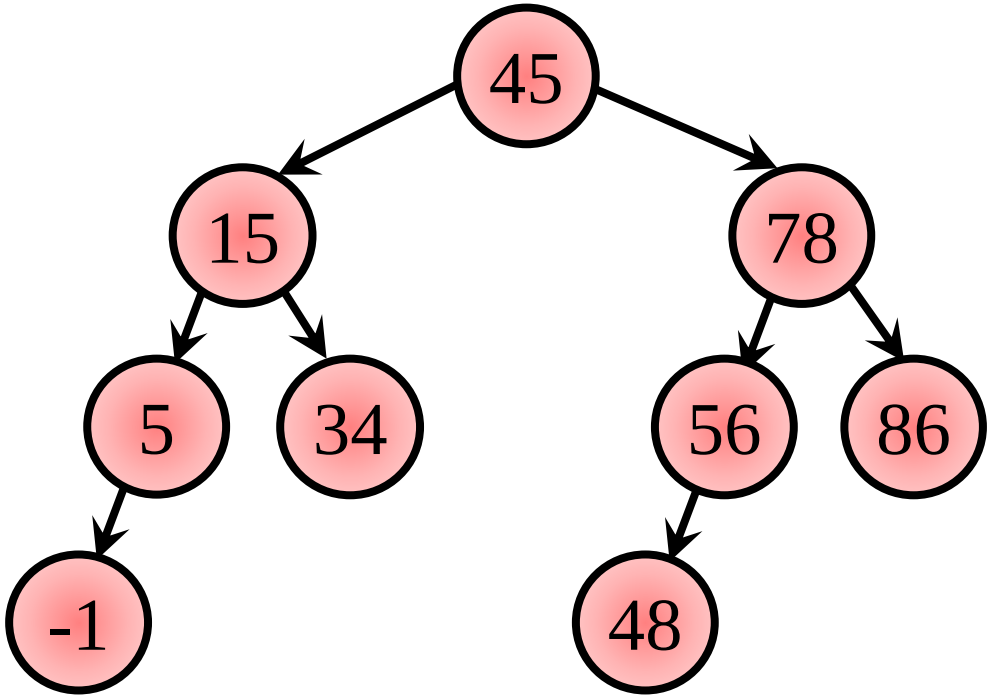
Phương pháp node left right



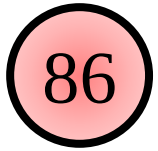
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 86



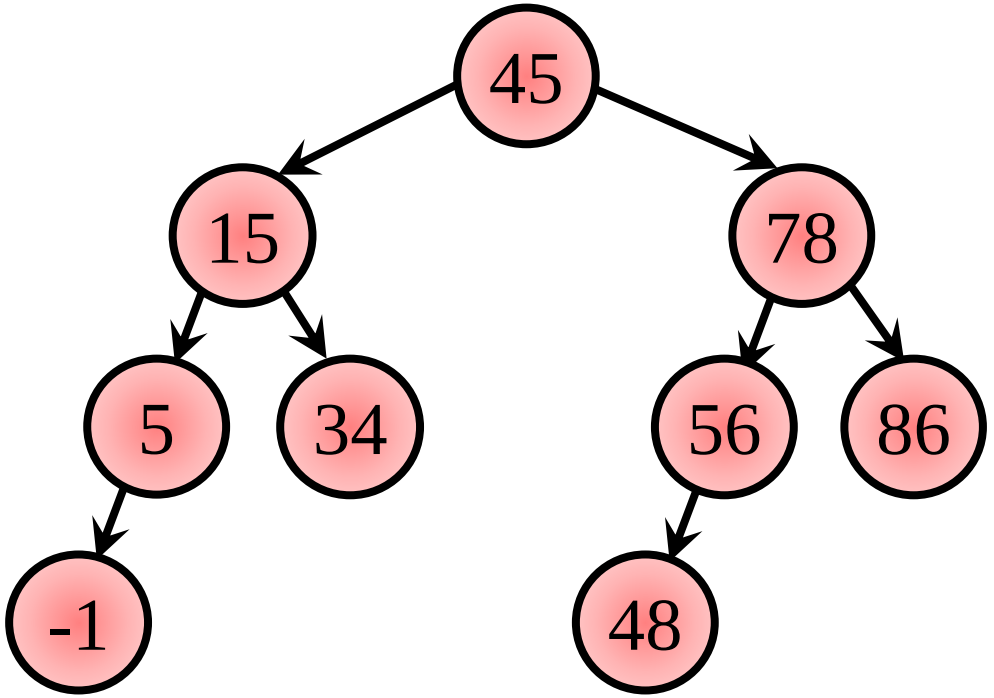
Phương pháp node left right



- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 86

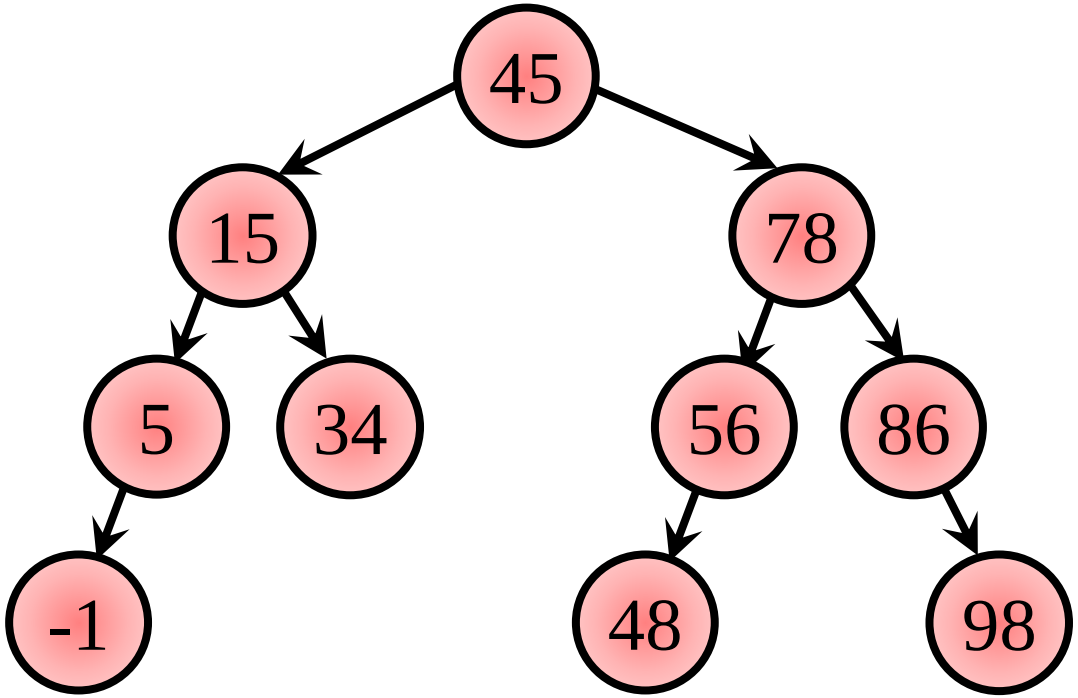


Phương pháp node left right



- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 98

Phương pháp node left right



- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 15, 5, -1, 34, 78, 56, 48, 86, 98.
- Thêm 98

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

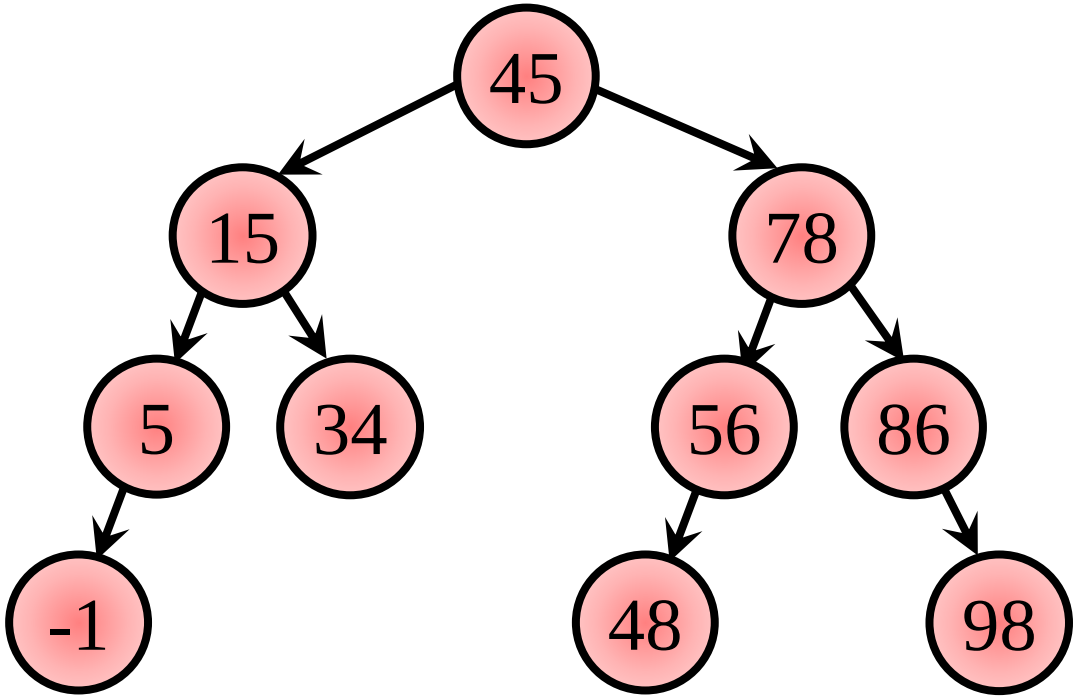
TS. Nguyễn Tấn Trần Minh Khang

BINARY SEARCH TREE – PHẦN 11

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

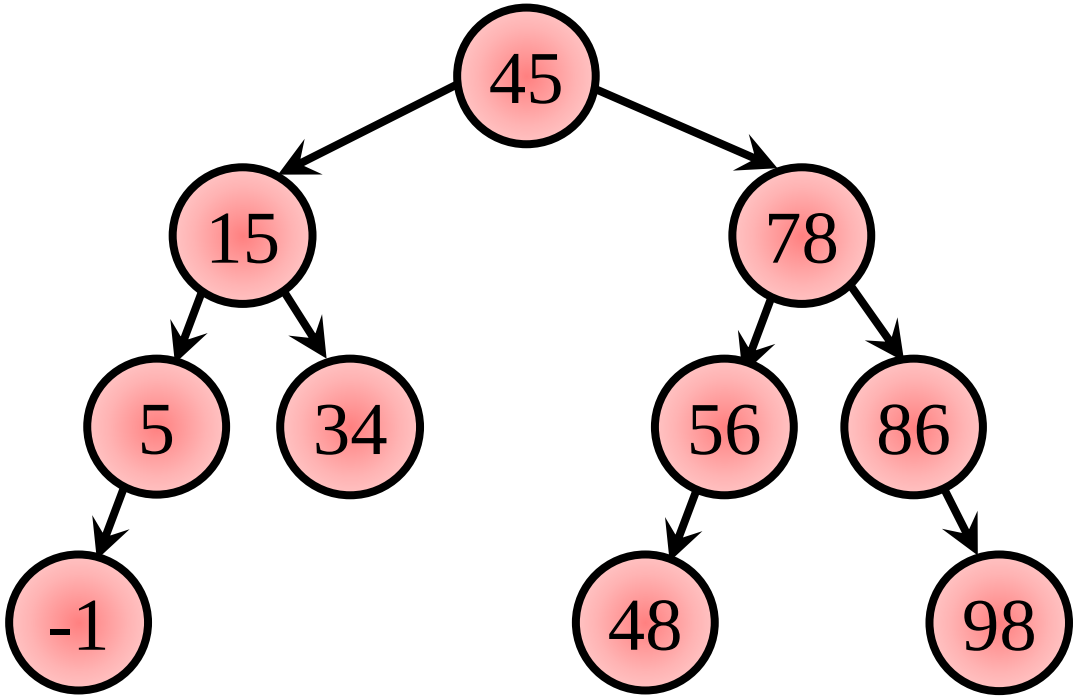
PHƯƠNG PHÁP 4: NODE RIGHT LEFT

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

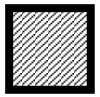


- Duyệt cây thep phương pháp NRL (Node Right Left) là: duyệt node gốc trong cây trước, sau đó duyệt tới cây con phải và cuối cùng là duyệt cây con trái.
- Cách thức duyệt cây con phải và duyệt cây con trái cũng giống như cách thức duyệt cây cha.
- Kết quả duyệt cây bên trái: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.

Phương pháp node right left

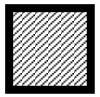
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.

Phương pháp node right left



- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Cây ban đầu rỗng

Phương pháp node right left



- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 45

Phương pháp node right left

– Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.

– Thêm 45

45

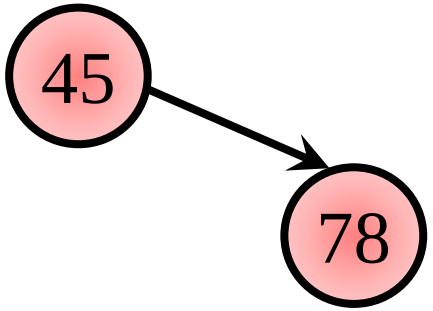
45

Phương pháp node right left

– Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.

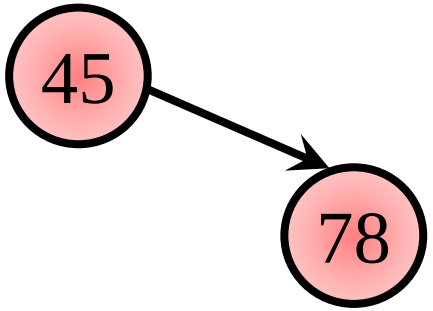
– Thêm 78

Phương pháp node right left

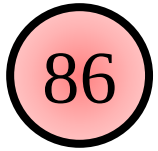


- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 78

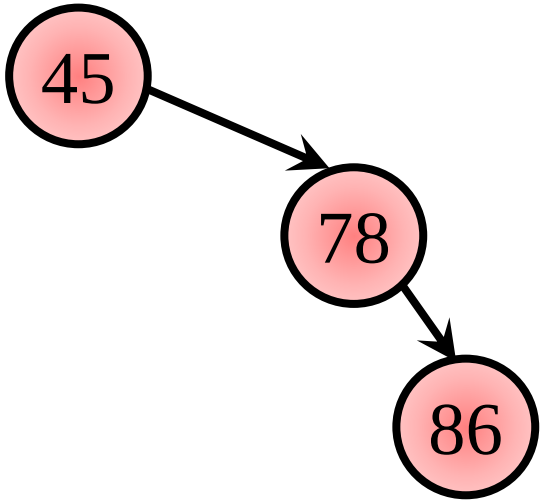
Phương pháp node right left



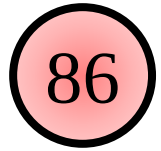
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 86



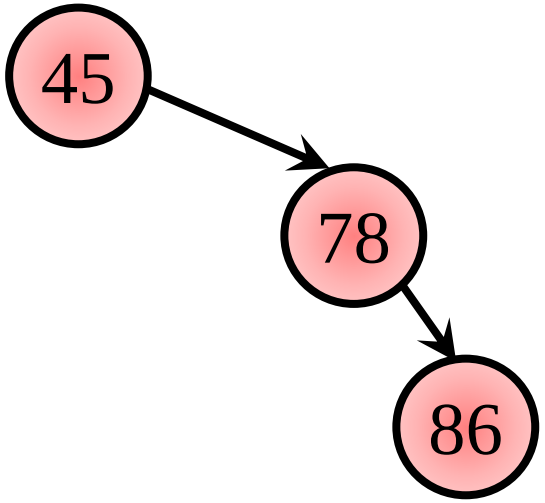
Phương pháp node right left



- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 86

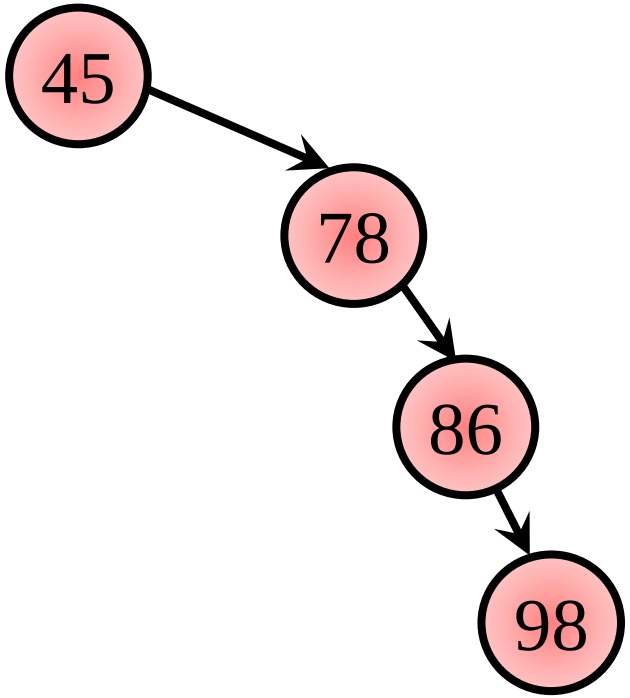


Phương pháp node right left



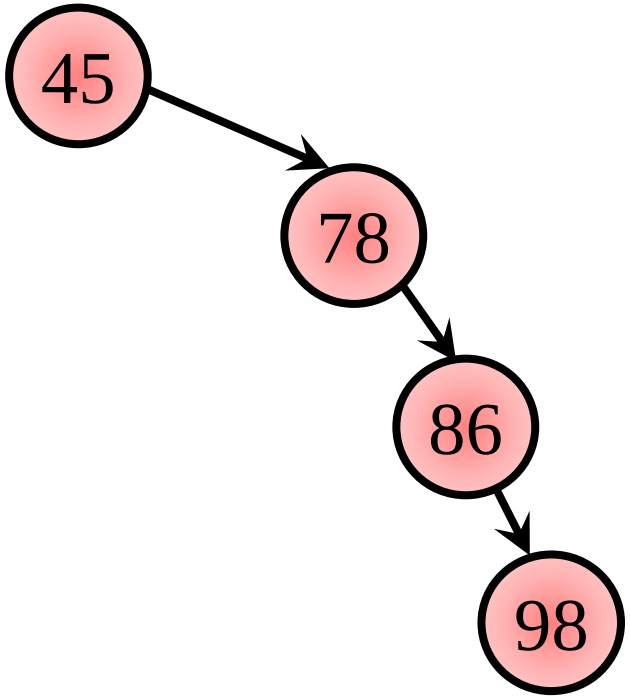
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 98

Phương pháp node right left



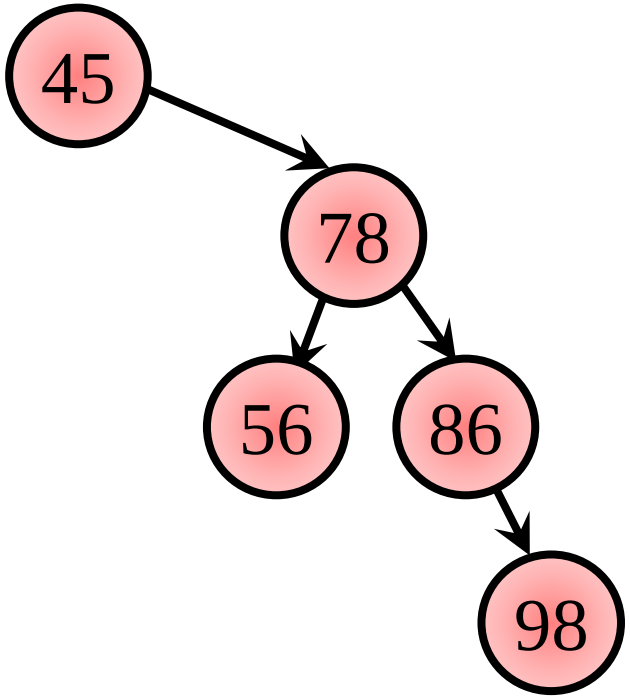
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 98

Phương pháp node right left



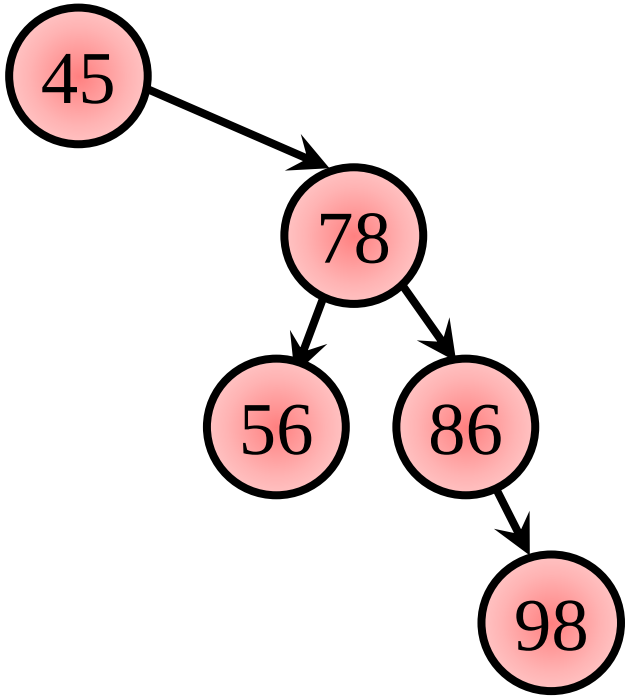
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 56

Phương pháp node right left



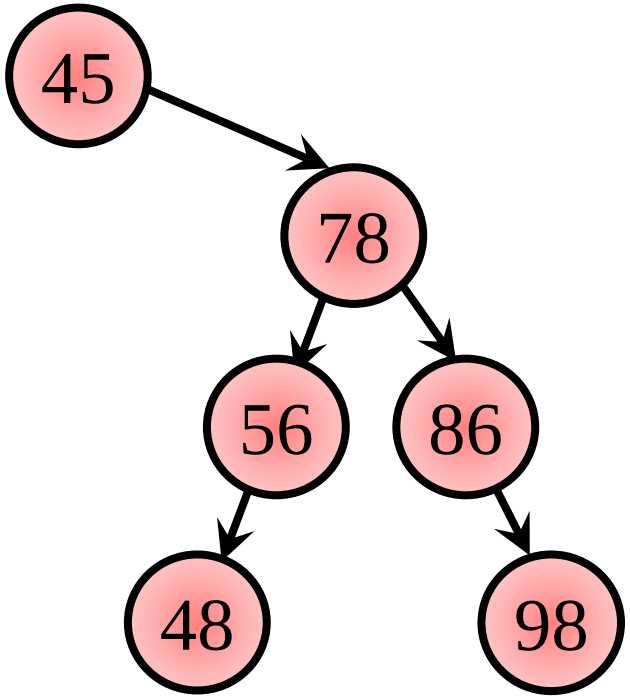
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 56

Phương pháp node right left



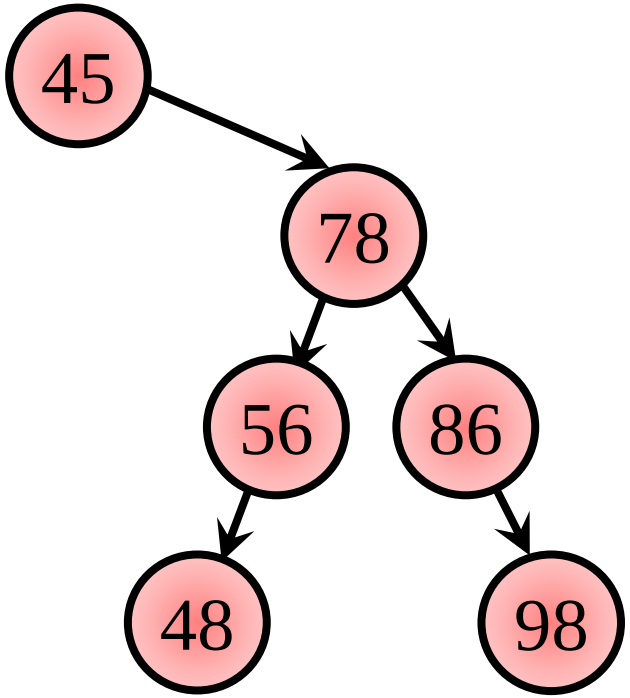
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 48

Phương pháp node right left



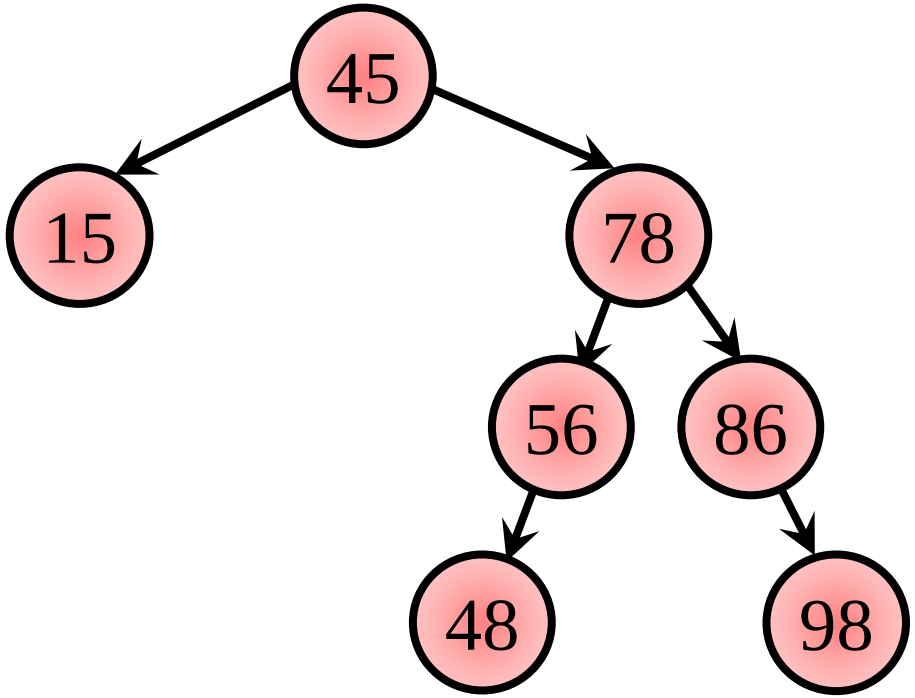
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 48

Phương pháp node right left



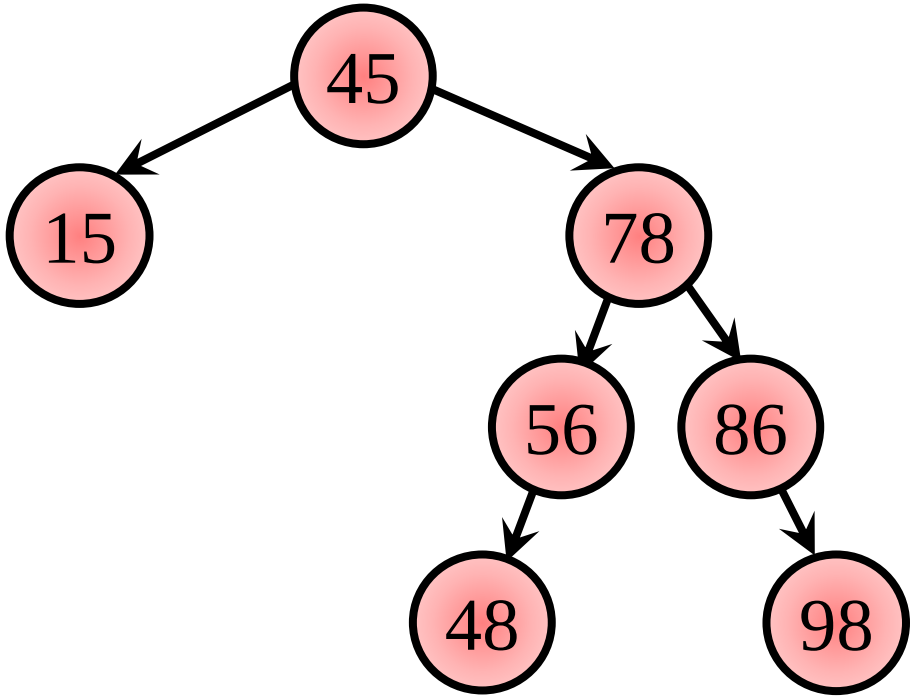
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 15

Phương pháp node right left



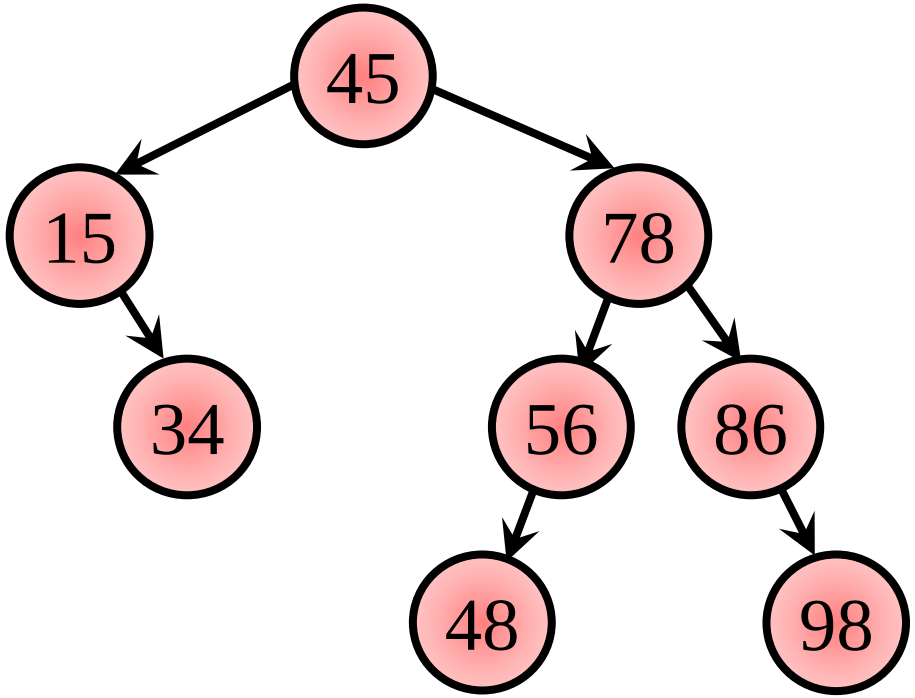
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 15

Phương pháp node right left



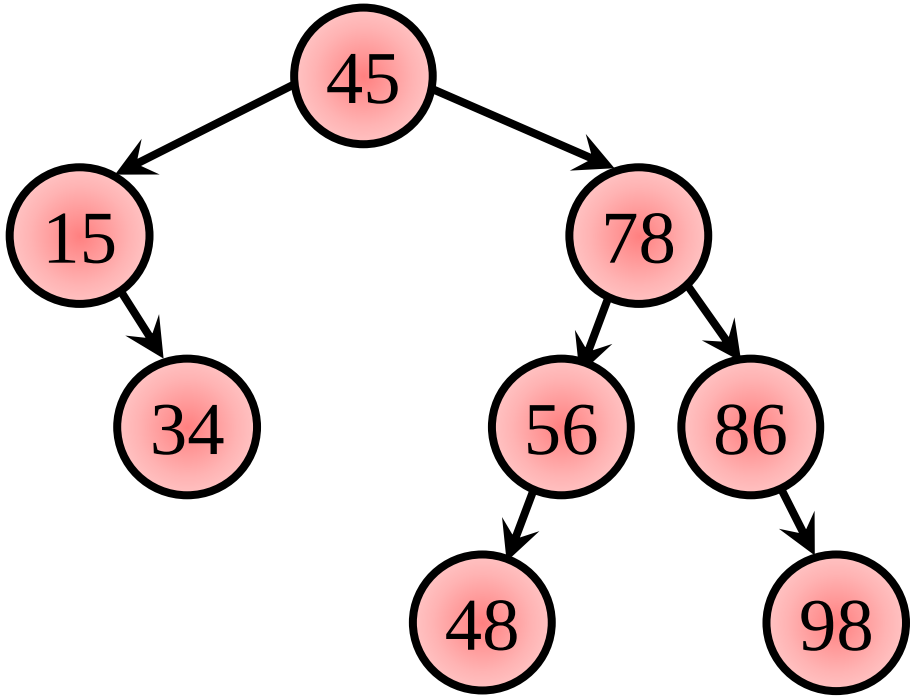
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 34

Phương pháp node right left



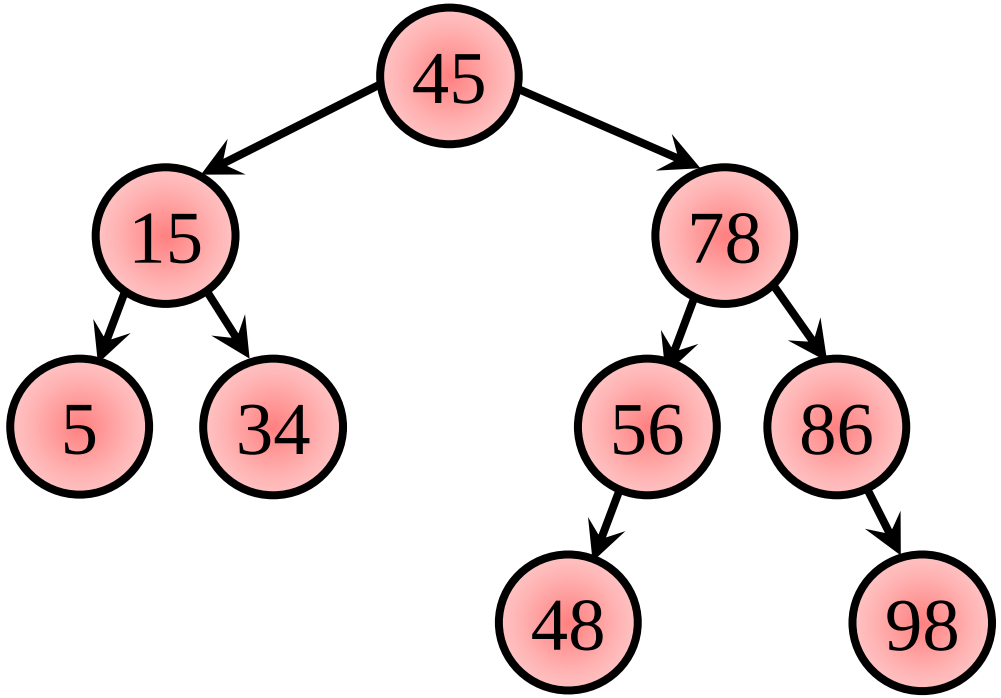
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 34

Phương pháp node right left



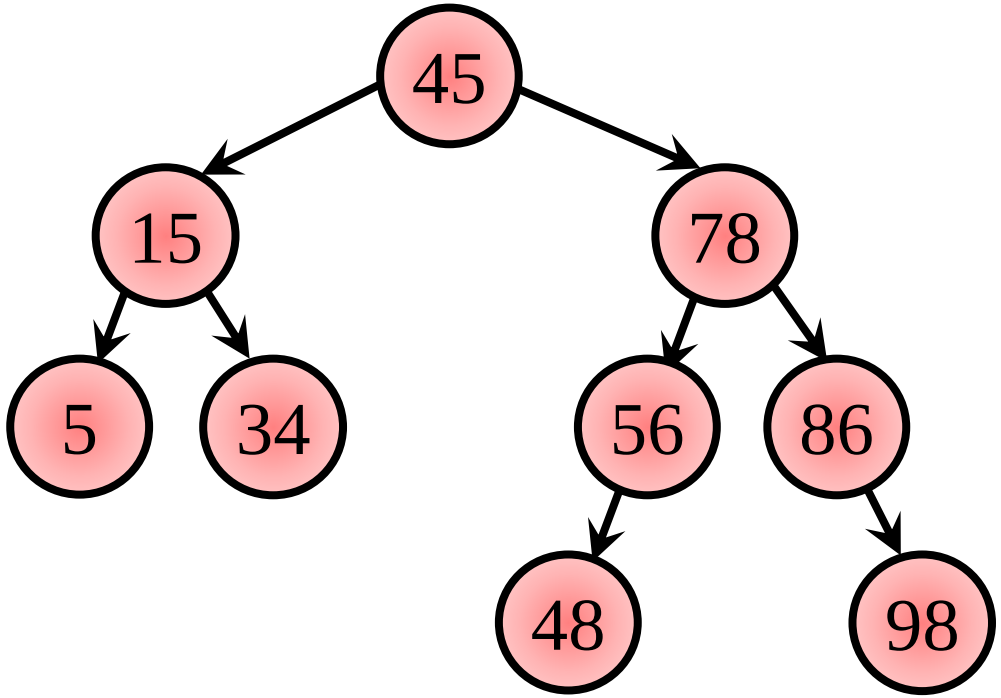
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 5

Phương pháp node right left



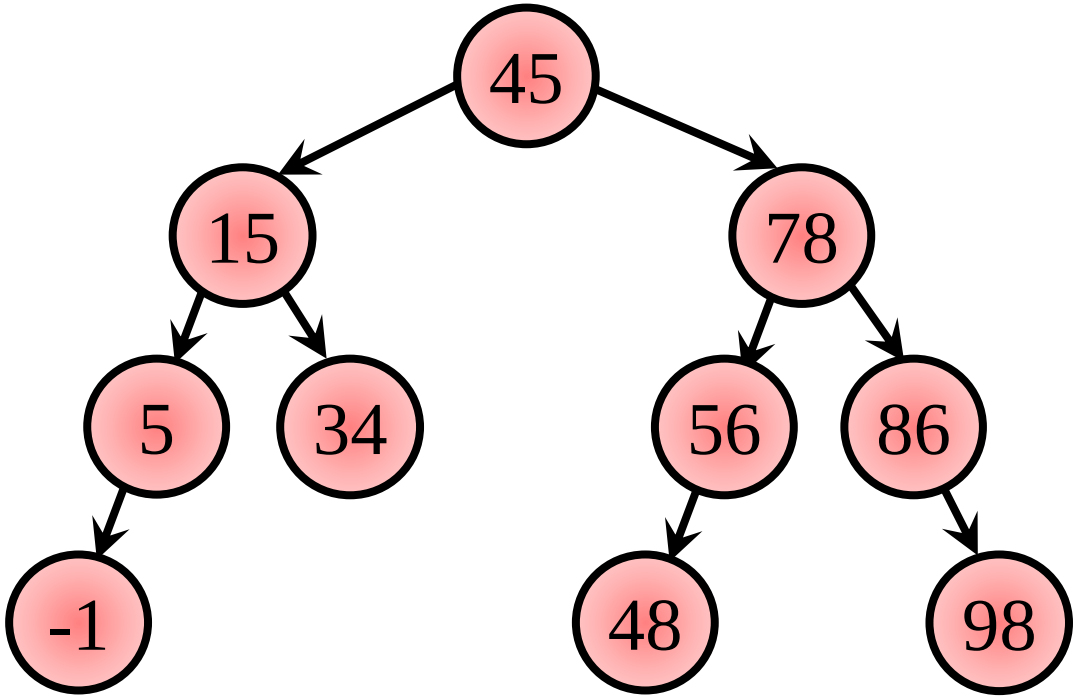
- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm 5

Phương pháp node right left



- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm -1

Phương pháp node right left



- Tạo cây nhị phân tìm kiếm bằng cách thêm lần lượt các giá trị sau biết rằng ban đầu cây rỗng: 45, 78, 86, 98, 56, 48, 15, 34, 5, -1.
- Thêm -1

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

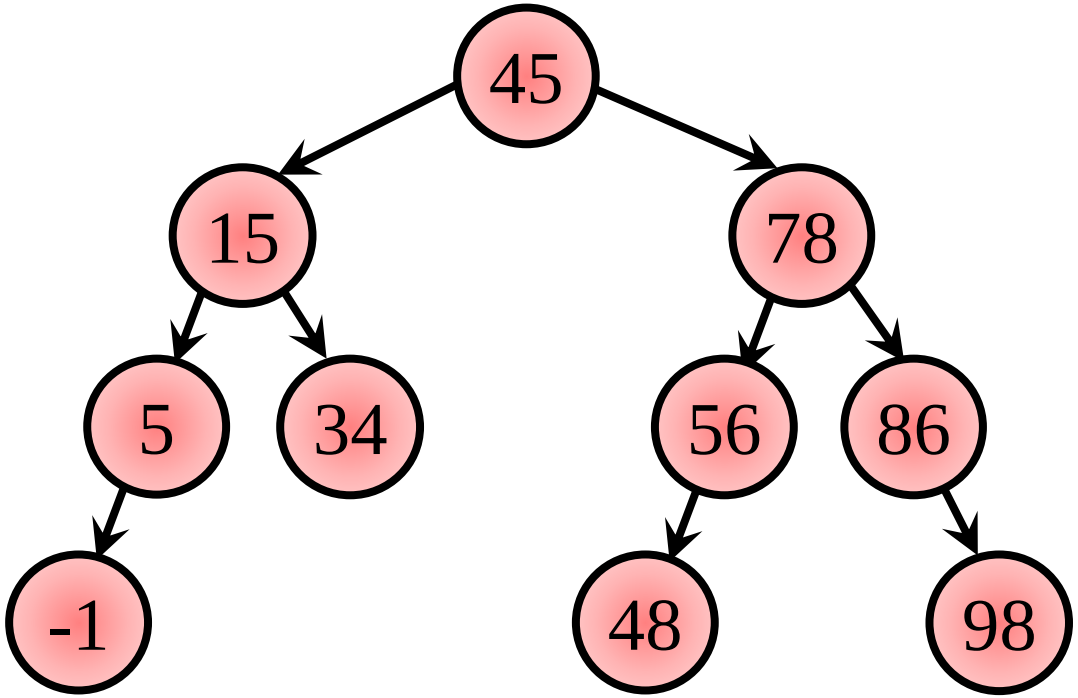
TS. Nguyễn Tấn Trần Minh Khang

BINARY SEARCH TREE – PHẦN 12

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

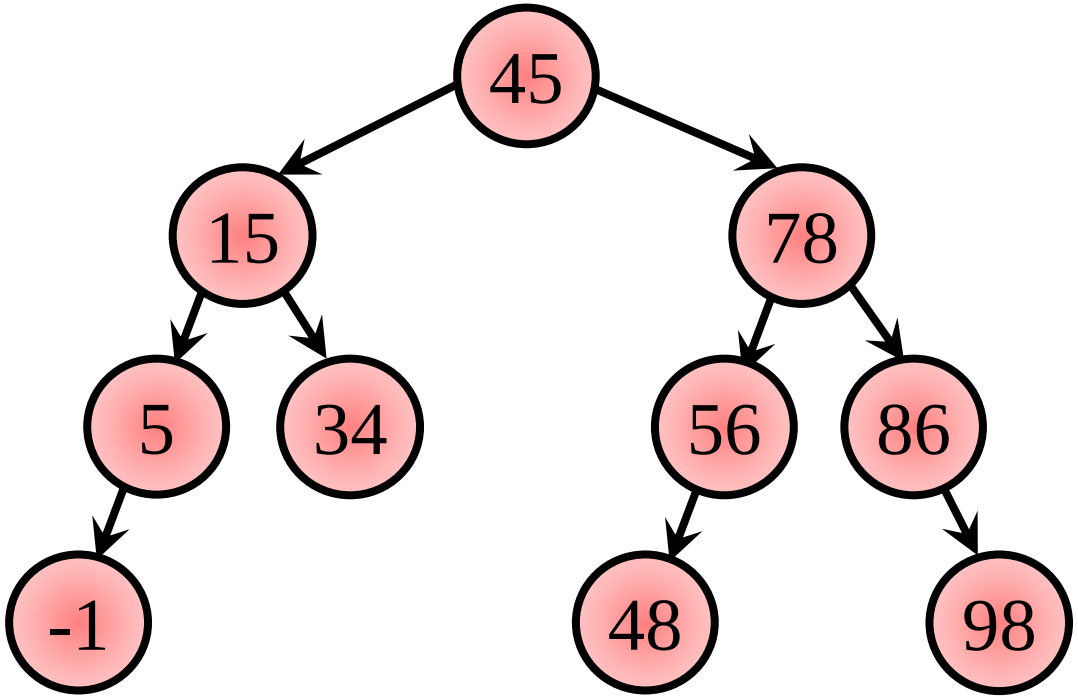
PHƯƠNG PHÁP 5: LEFT RIGHT NODE

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: -1, 5, 34, 15, 48, 56, 98, 86, 78, 45.

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

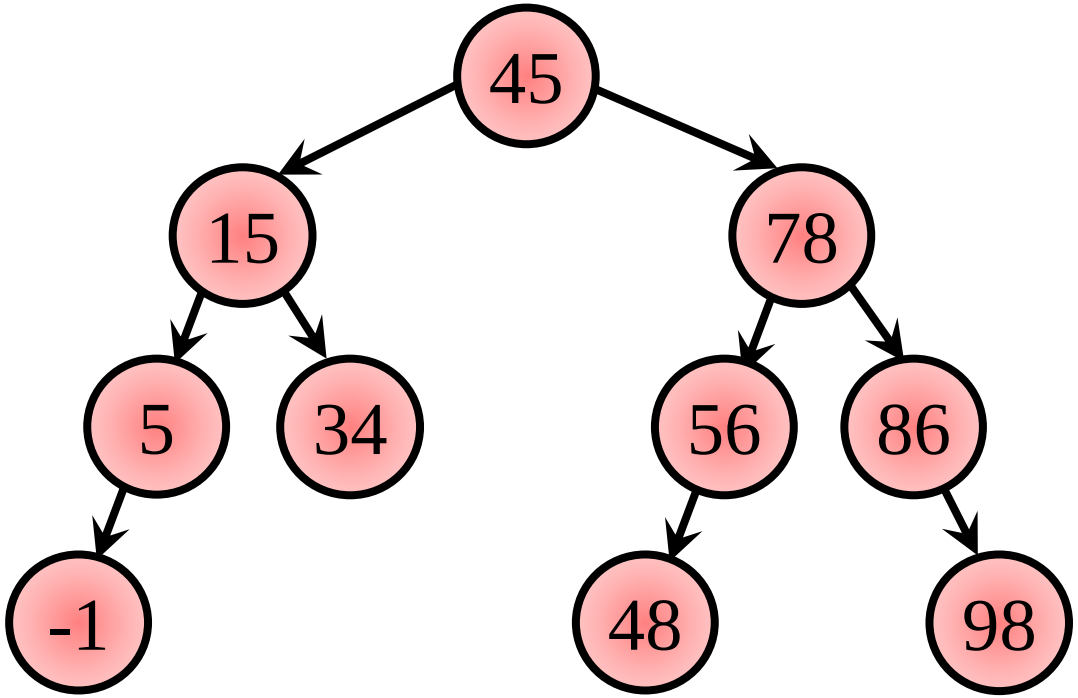
TS. Nguyễn Tấn Trần Minh Khang

BINARY SEARCH TREE – PHẦN 13

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

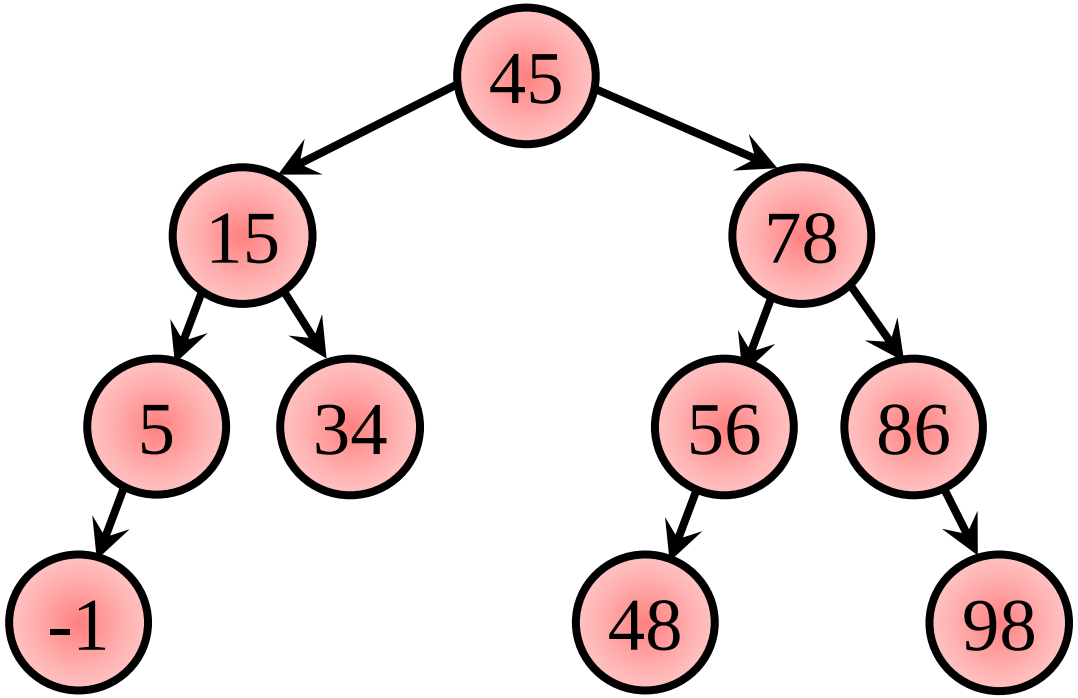
PHƯƠNG PHÁP 6: RIGHT LEFT NODE

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: 98, 86, 48, 56, 78, 34, -1, 5, 15, 45.

Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

BINARY SEARCH TREE – PHẦN 14

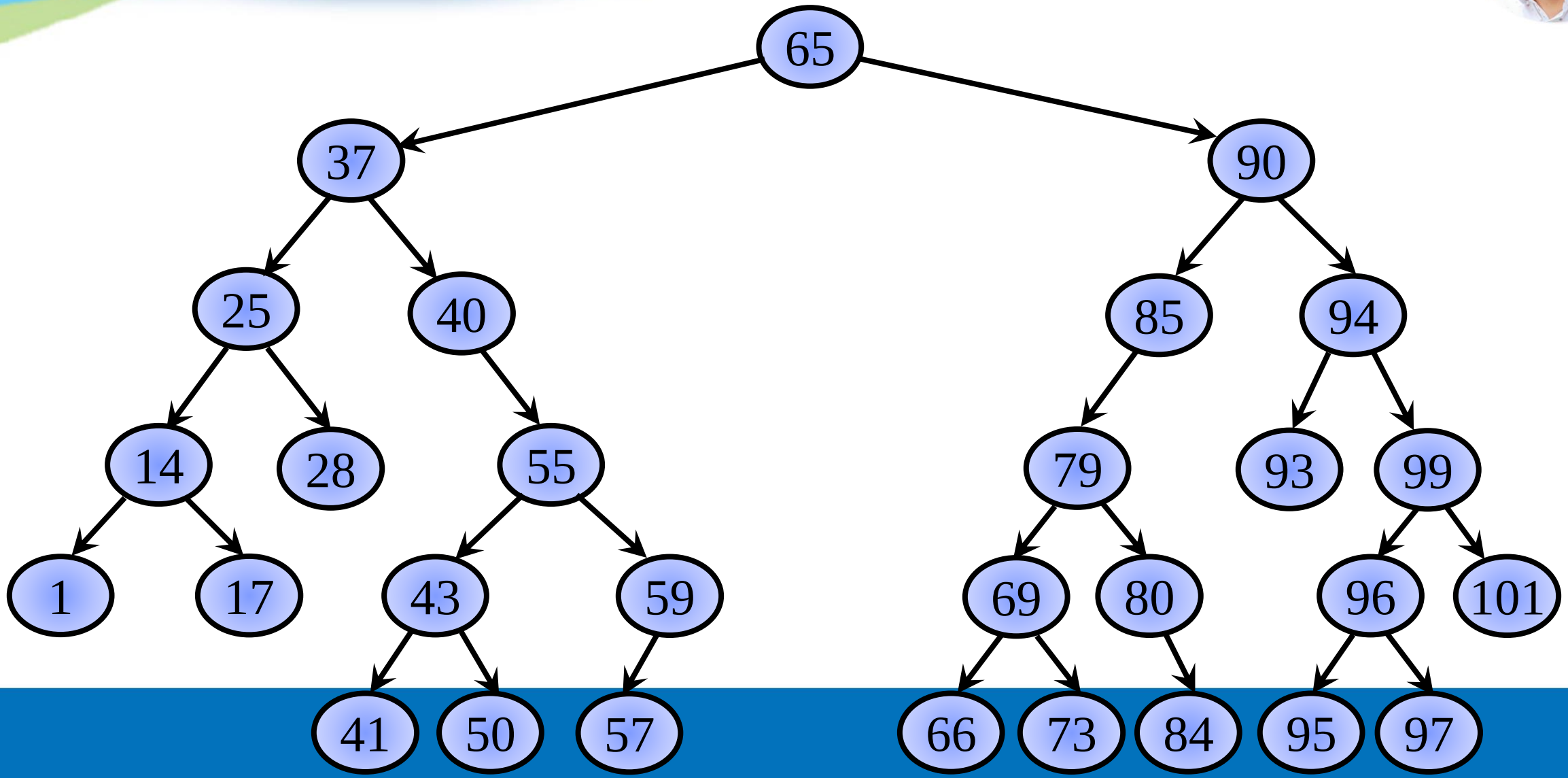
1. Hồ Thái Ngọc
2. ThS. Võ Duy Nguyên
3. TS. Nguyễn Tấn Trần Minh Khang

DUYỆT CÂY NHỊ PHÂN CỤ THỂ

Duyệt cây nhị phân tìm kiếm cụ thể

- Hãy duyệt cây nhị phân cụ thể dưới đây và cho biết kết quả duyệt cây theo 6 phương pháp:
  + 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).

Duyệt cây nhị phân tìm kiếm



Cảm ơn quí vị đã lắng nghe

Nhóm tác giả

Hồ Thái Ngọc

ThS. Võ Duy Nguyên

TS. Nguyễn Tấn Trần Minh Khang

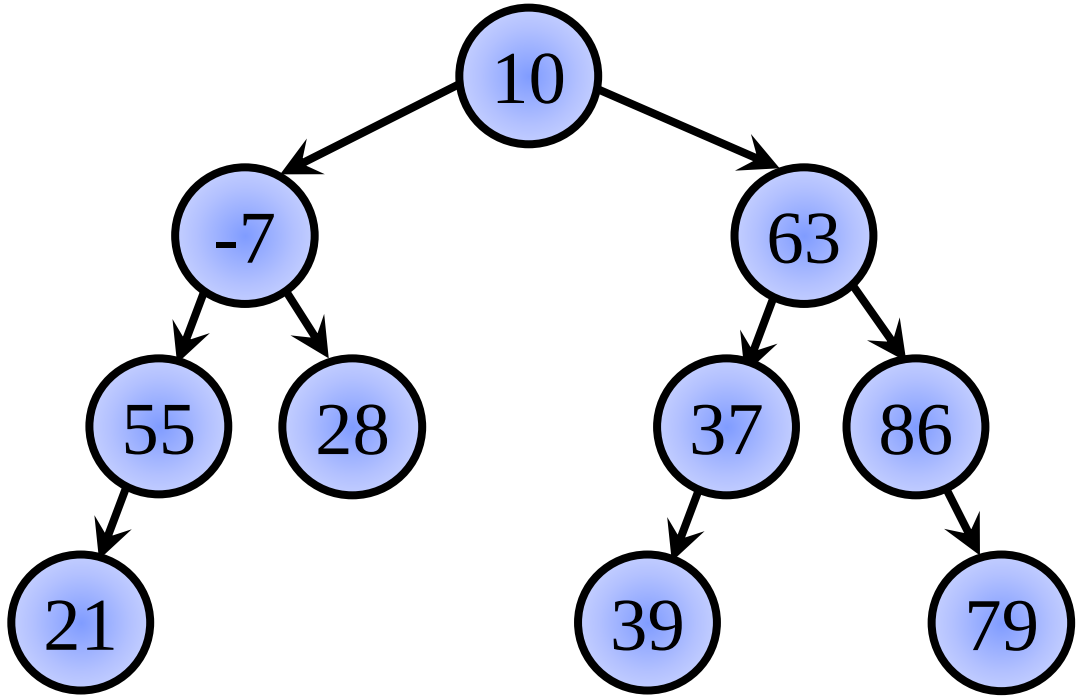
BINARY SEARCH TREE – PHẦN 15

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

XUẤT CÂY NHỊ PHÂN TÌM KIẾM CÁC SỐ NGUYÊN

Xuất BST các số nguyên

- Bài toán: Hãy định nghĩa tất cả các hàm duyệt và xuất cây nhị phân tìm kiếm 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
  + RLN – Right Left Node



Xuất BST các số nguyên

- Bài toán: Hãy định nghĩa tất cả các hàm duyệt và xuất cây nhị phân tìm kiếm 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 BST 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 BST 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 BST 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 BST 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 BST 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 BST 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

BINARY SEARCH TREE – PHẦN 16

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

ĐẾM SỐ LƯỢNG NODE TRONG BST CÁC SỐ NGUYÊN

Đếm số lượng node trong BST

– 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 trong BST

— 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

BINARY SEARCH TREE – PHẦN 17

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

XUẤT BST CÁC SỐ NGUYÊN RA FILE

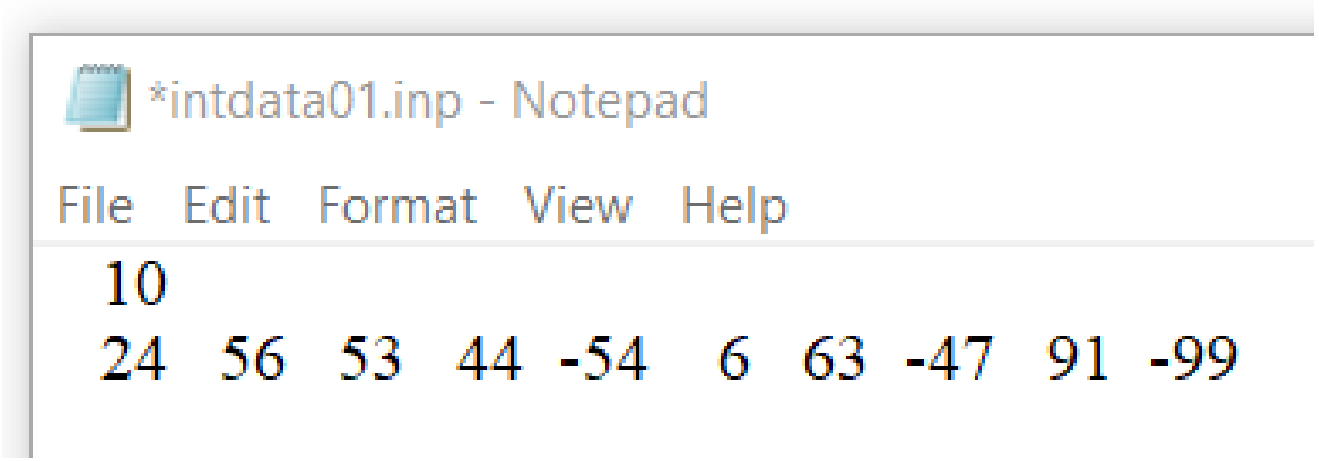
Xuất BST các số nguyên ra file

- Bài toán: Hãy định nghĩa hàm xuất cây nhị phân tìm kiếm 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 BST các số nguyên ra file

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



Xuất BST 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 BST 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 BST 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

BINARY SEARCH TREE – PHẦN 18

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

DỰ ÁN CÂY NHỊ PHÂN TÌM KIẾM

Dự án cây nhị phân tìm kiếm

- Viết chương trình thực hiện các yêu cầu sau:
  - + Nhập cây nhị phân các số nguyên từ các tập tin: intbst01.inp; intbst02.inp; …; intbst09.inp; intbst10.inp; intbst11.inp; intbst12.inp; intbst13.inp;
  - + Đếm số lượng node trong cây.
  - + Xuất cây nhị phân các số nguyên ra các tập tin: intbst01.out; intbst02.out; …; intbst09.out; intbst10.out; intbst11.out; intbst12.out; intbst13.out;

Dự án cây nhị phân tìm kiếm

- Định dạng tập tin `intbstxx.inp` và `intbstxx.out`
  - + Dòng đầu tiên: số phần tử của BST các số nguyên (n).
  - + Dòng tiếp theo: lưu n số nguyên tương ứng với các giá trị trong cây nhị phân tìm kiếm 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 = "intbst";
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.             cout << "\n So luong node: " << DemNode(t);
25.             string filenameout = filename;
26.             filenameout += ".out";
27.             Xuat(t, filenameout);
28.             cout << "\n" << filenameinp << "\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