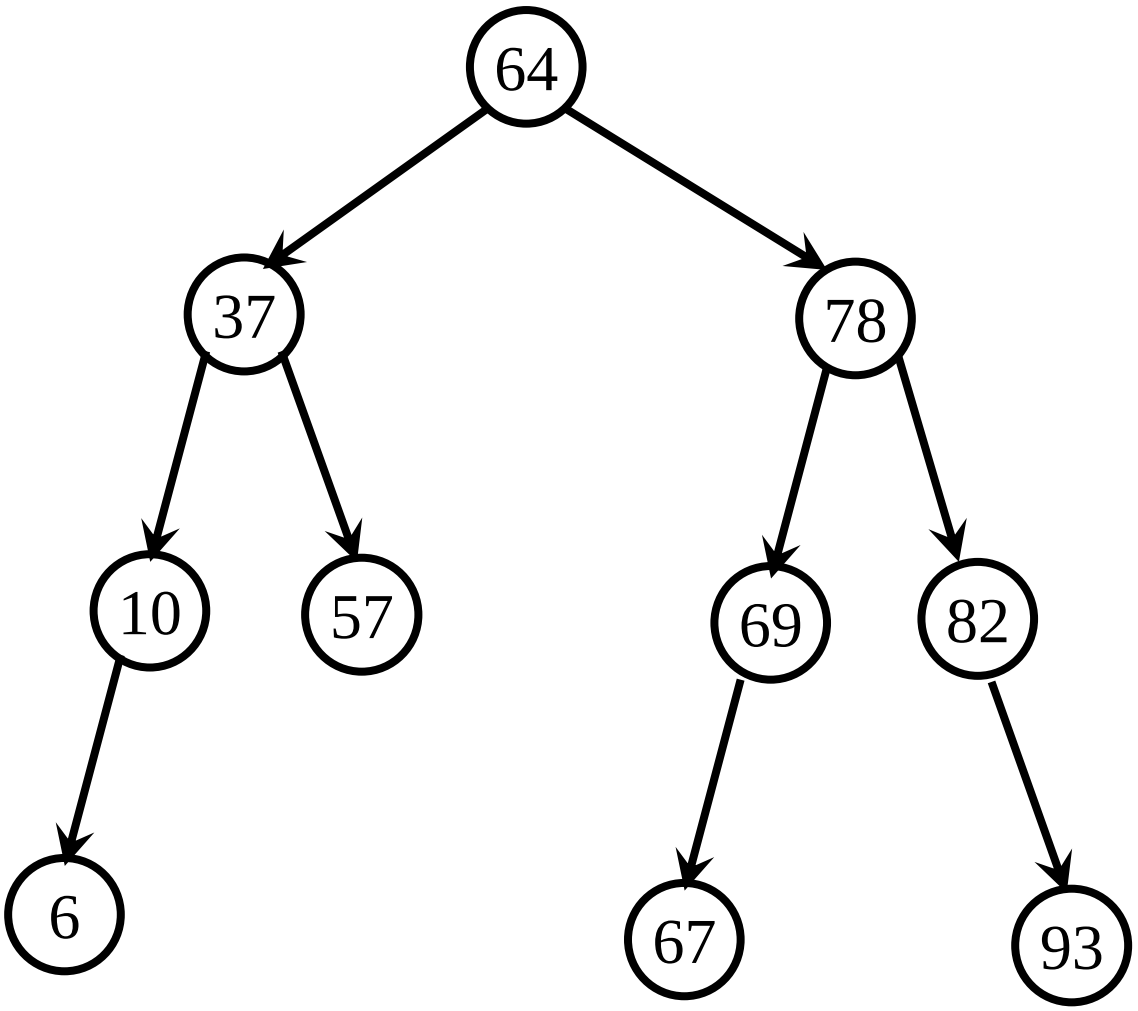
Chương 7
CÂY NHỊ PHÂN TÌM KIẾM
BINARY SEARCH TREE-BST

TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

0. HÌNH VẼ



1. KHÁI NIỆM

- Cây nhị phân tìm kiếm là cây nhị phân thoả điều kiện sau: **Mọi** node trong cây đều có khoá lớn hơn tất cả các khoá thuộc cây con trái và nhỏ hơn tất cả các khoá thuộc cây con phải.
- Cây nhị phân tìm kiếm là cây nhị phân hỗ trợ việc tìm kiếm dựa trên khoá của từng node trong cây.

Click to see Figure 1-2

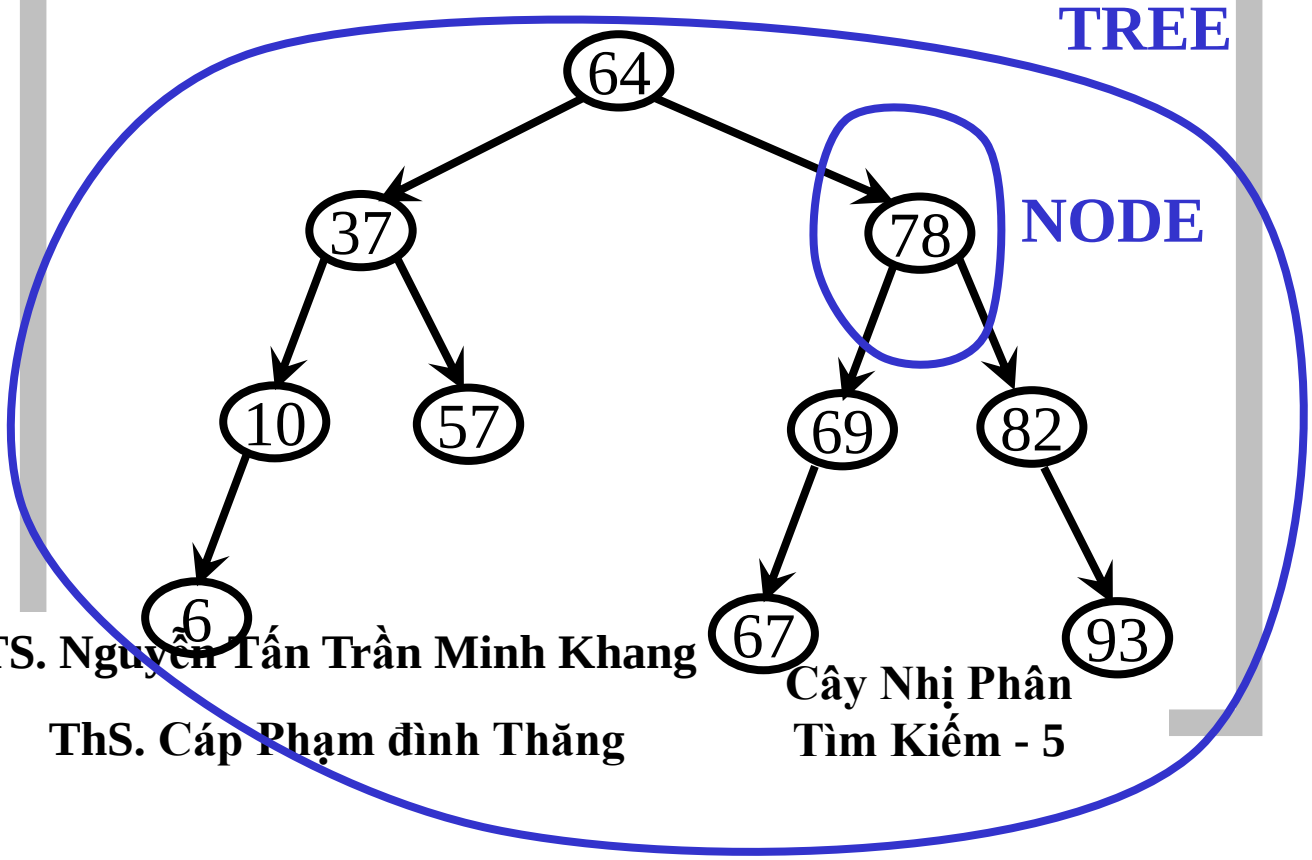
2. CẤU TRÚC DỮ LIỆU CỦA CÂY PHÂN TÌM KIẾM

```
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;
```

- KDL là kiểu dữ liệu của đối tượng được lưu trong node của cây nhị phân tìm kiếm.

2. CẤU TRÚC DỮ LIỆU CỦA CÂY PHÂN TÌM KIẾM

```
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. KHỞI TẠO CÂY PHÂN TÌM KIẾM

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

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

4. KIỂM TRA CÂY NHỊ PHÂN TÌM KIẾM RỖNG

- Khái niệm: Kiểm tra cây nhị phân rỗng là một hàm sẽ trả về giá trị 1 nếu cây nhị phân rỗng. Ngược lại, nếu cây nhị phân ko rỗng thì hàm trả về giá trị 0.
- Định nghĩa hàm

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

5. TẠO NODE CHO CÂY NHỊ PHÂN TÌM KIẾM

- 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.
- Định nghĩa hàm

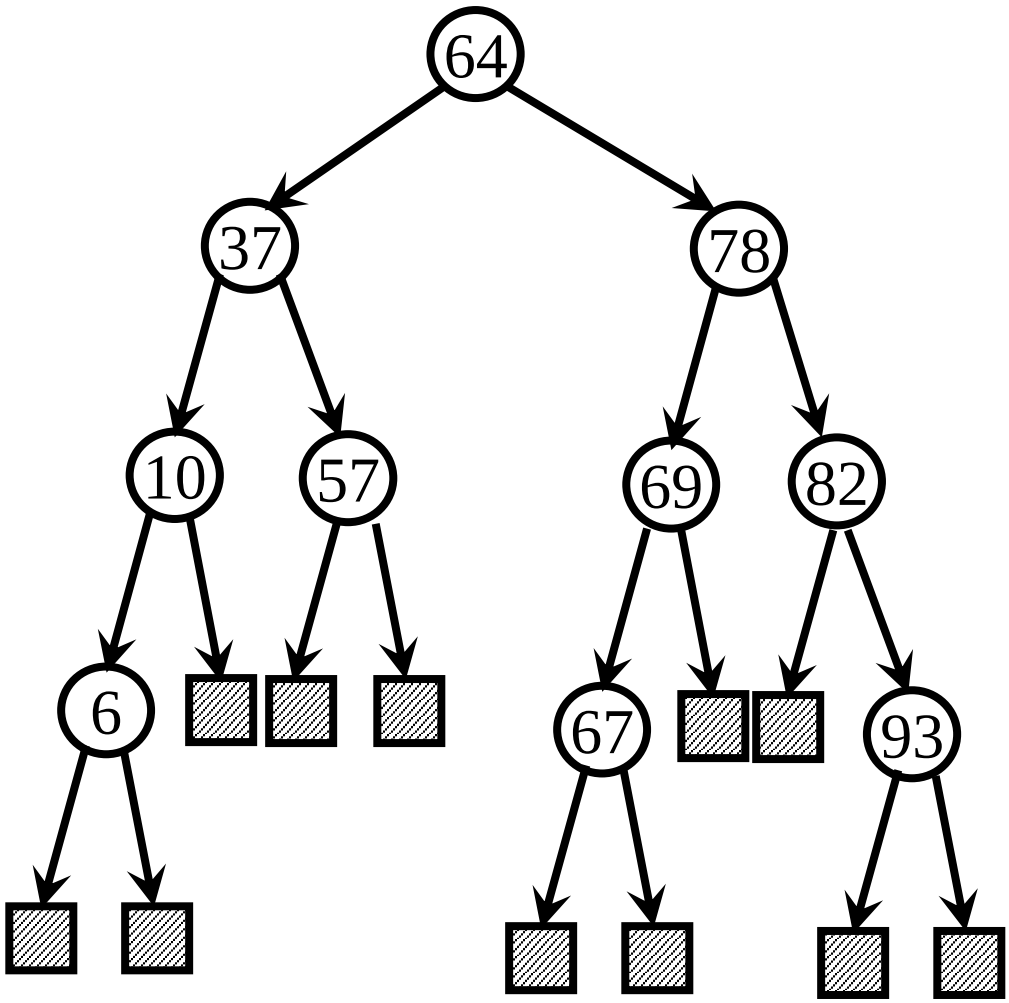
```
11. NODE* GetNode(KDL x)
12. {
13.     NODE *p = new NODE;
14.     if(p==NULL)
15.         return NULL;
16.     p->info=x;
17.     p->pLeft = NULL;
18.     p->pRight= NULL;
19.     return p;
20. }
```

- Ghi chú: KDL là kiểu dữ liệu của thông tin chứa tại node trong cây.

6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM

- Khái niệm: Thêm một giá trị vào trong cây nhị phân tìm kiếm là thêm thông tin vào cây sao cho tính chất của cây nhị phân tìm kiếm không bị vi phạm.
- Giá trị trả về: Hàm thêm một giá trị vào trong cây nhị phân tìm kiếm trả về một trong 3 giá trị -1, 0, 1 với ý nghĩa như sau:
  - Giá trị 1: Thêm thành công.
  - Giá trị 0: Trùng với khoá một node đã có sẵn trong cây.
  - Giá trị -1: Không đủ bộ nhớ.

6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM



TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM

- Vấn đề 1: Hãy định nghĩa hàm thêm một node (thông tin) vào nhị phân tìm kiếm các số nguyên.
- Cấu trúc dữ liệu:

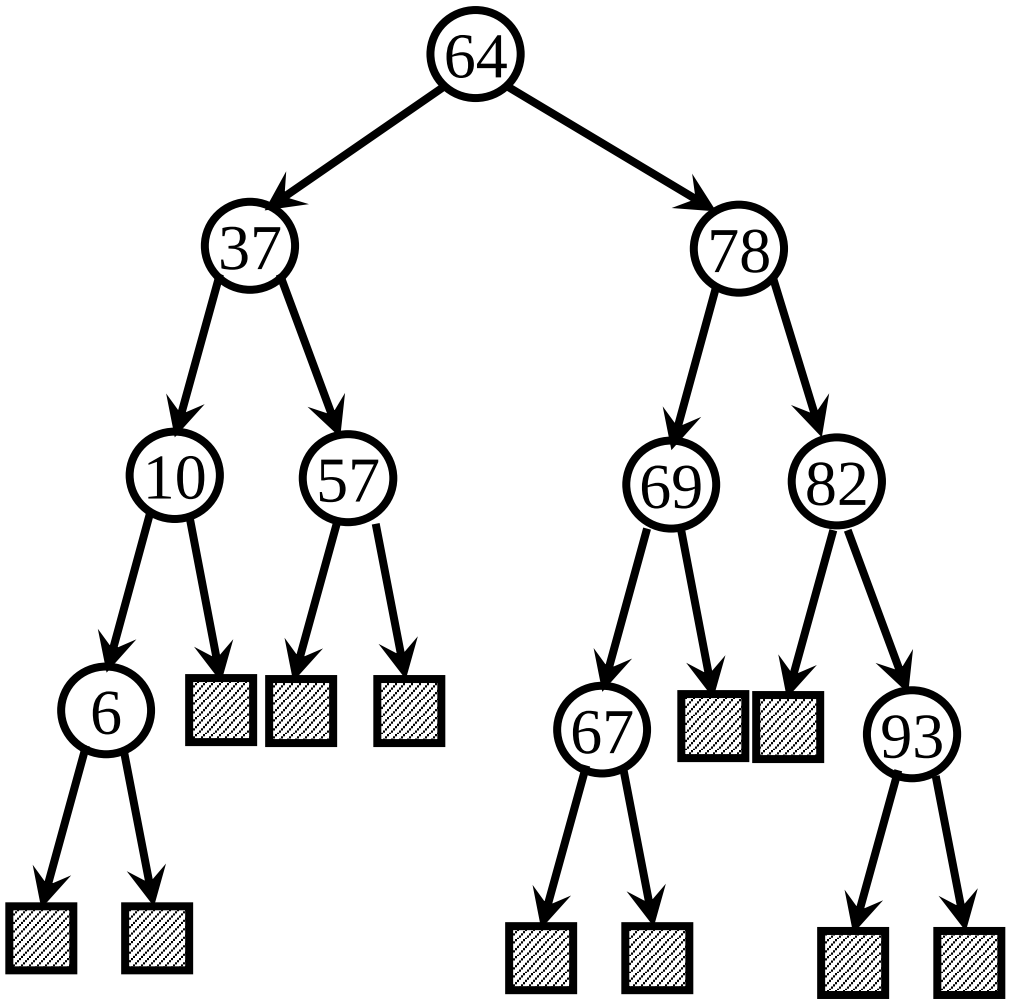
```
1. struct node
2. {
3.     int info;
4.     struct node *pLeft;
5.     struct node *pRight;
6. };
7. typedef struct node NODE;
8. typedef NODE *TREE;
```

6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM

- Định nghĩa hàm

```
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. }
```

6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM



TS. Nguyễn Tấn Trần Minh Khang

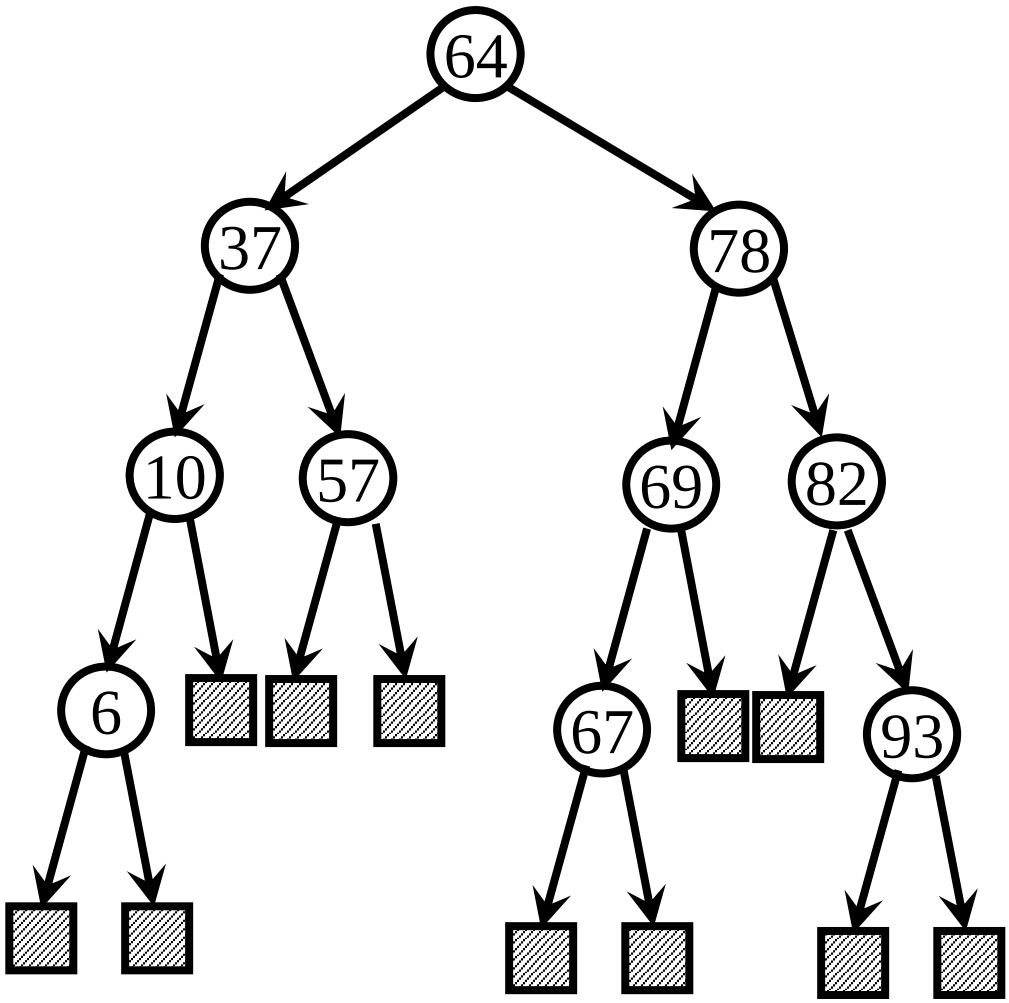
ThS. Cáp Phạm đình Thăng

6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM

- Vấn đề 1: Thêm một node vào cây nhị phân tìm kiếm các số nguyên.
- Định nghĩa hàm

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

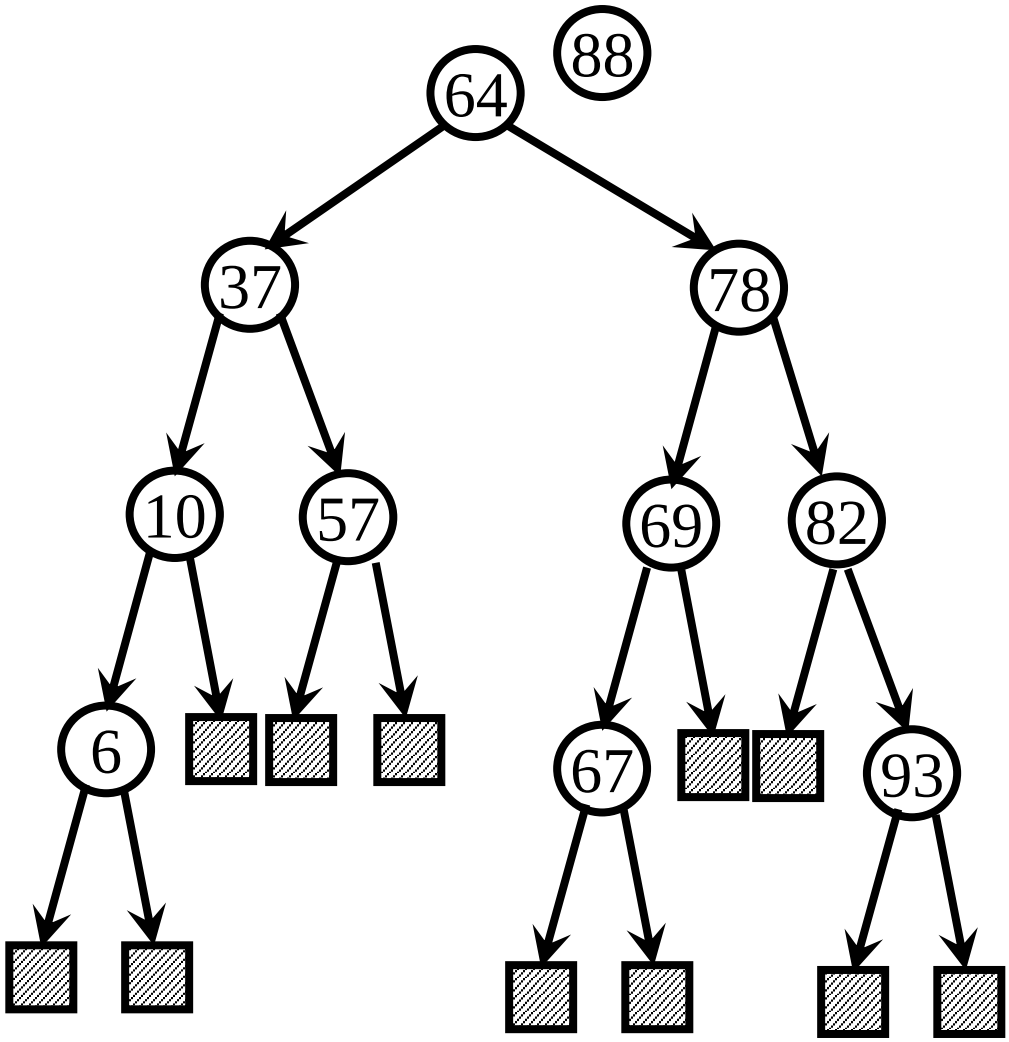
6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM



TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

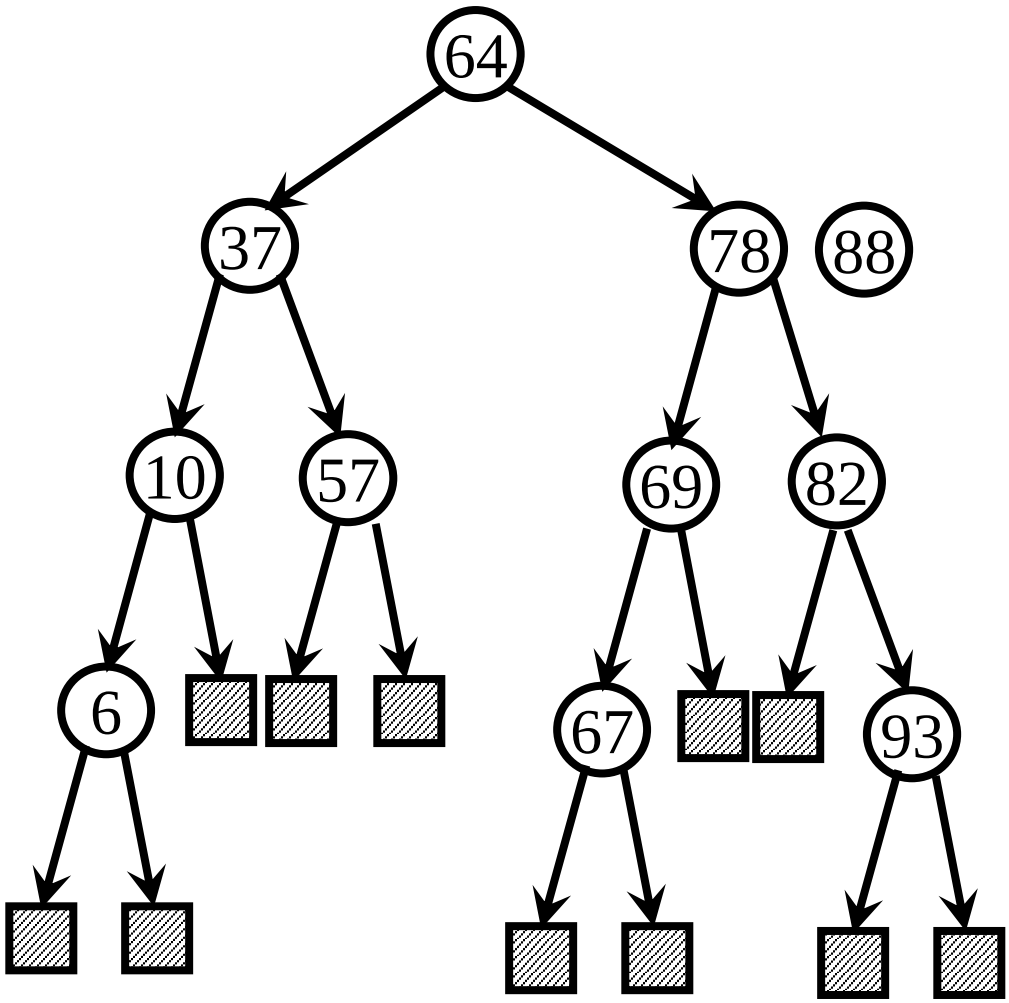
6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM



TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

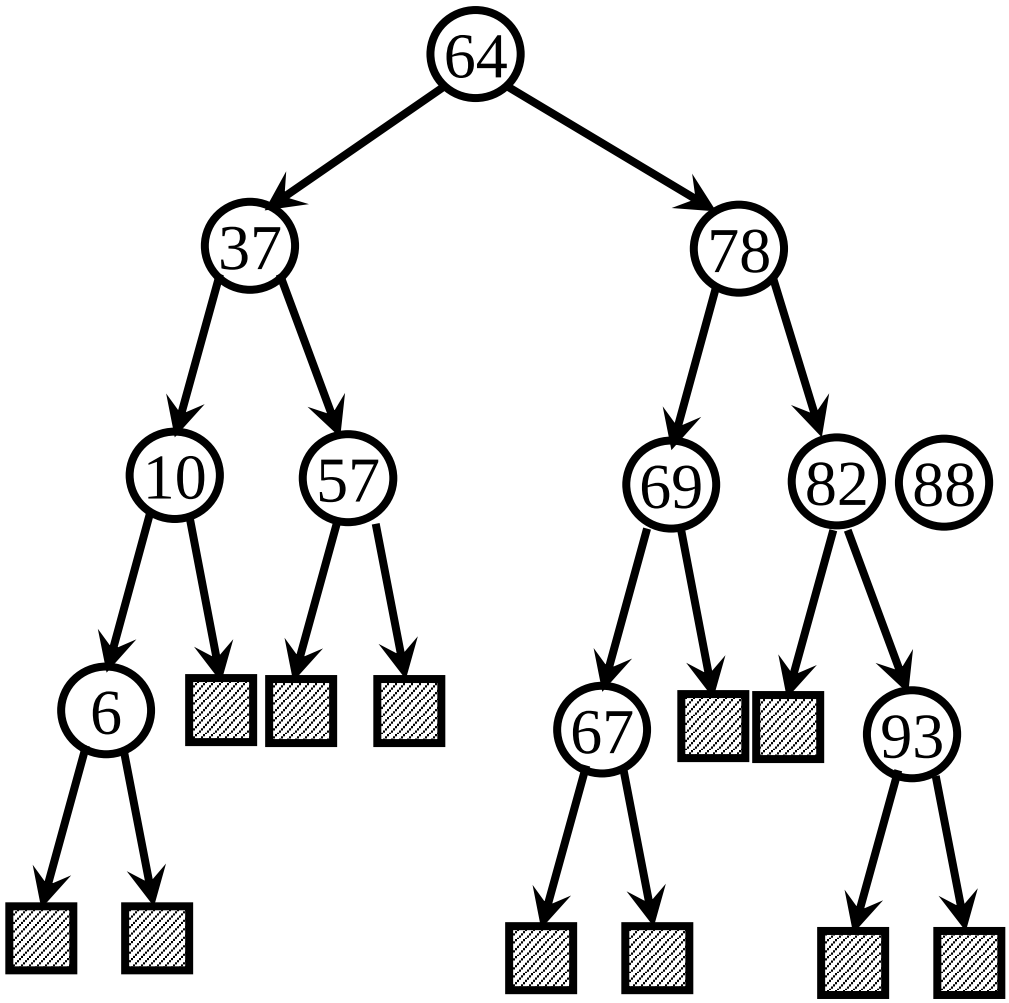
6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM



TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

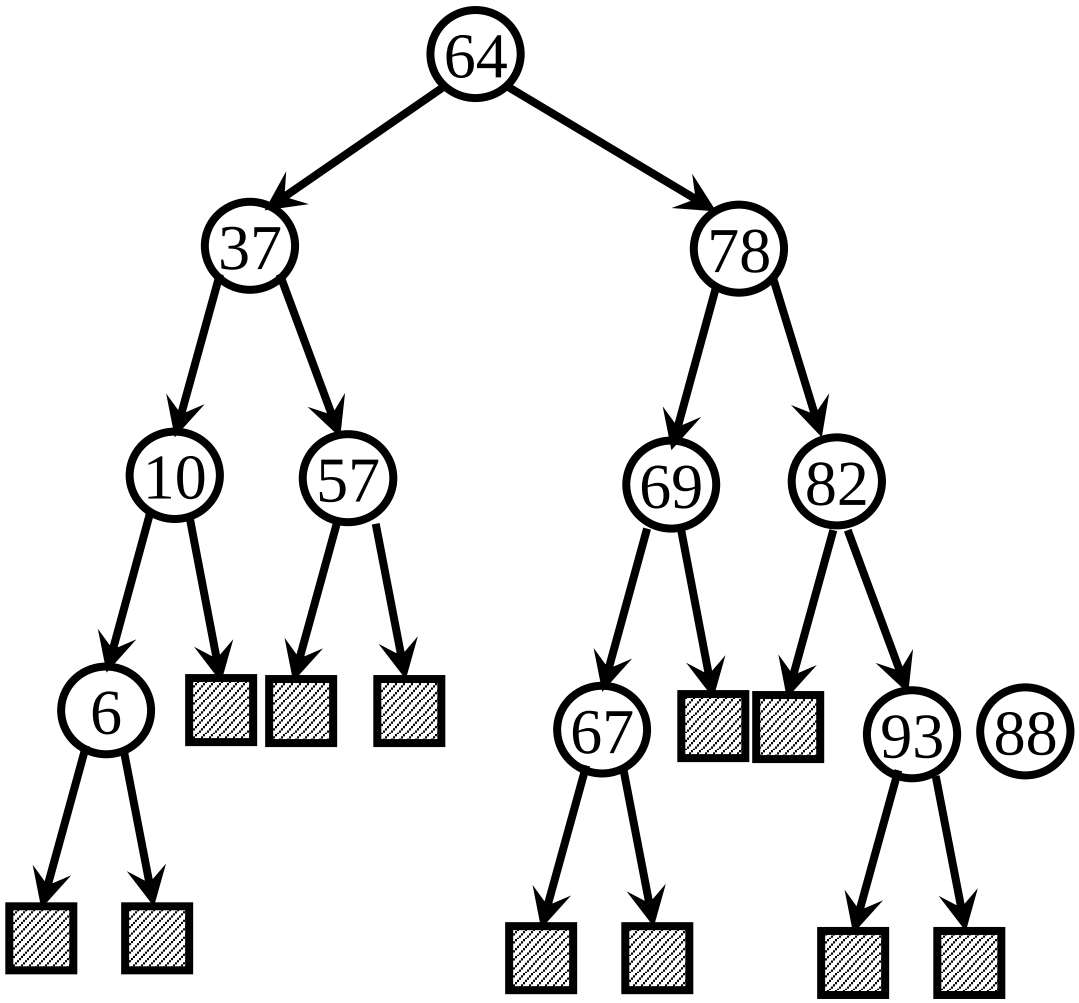
6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM



TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

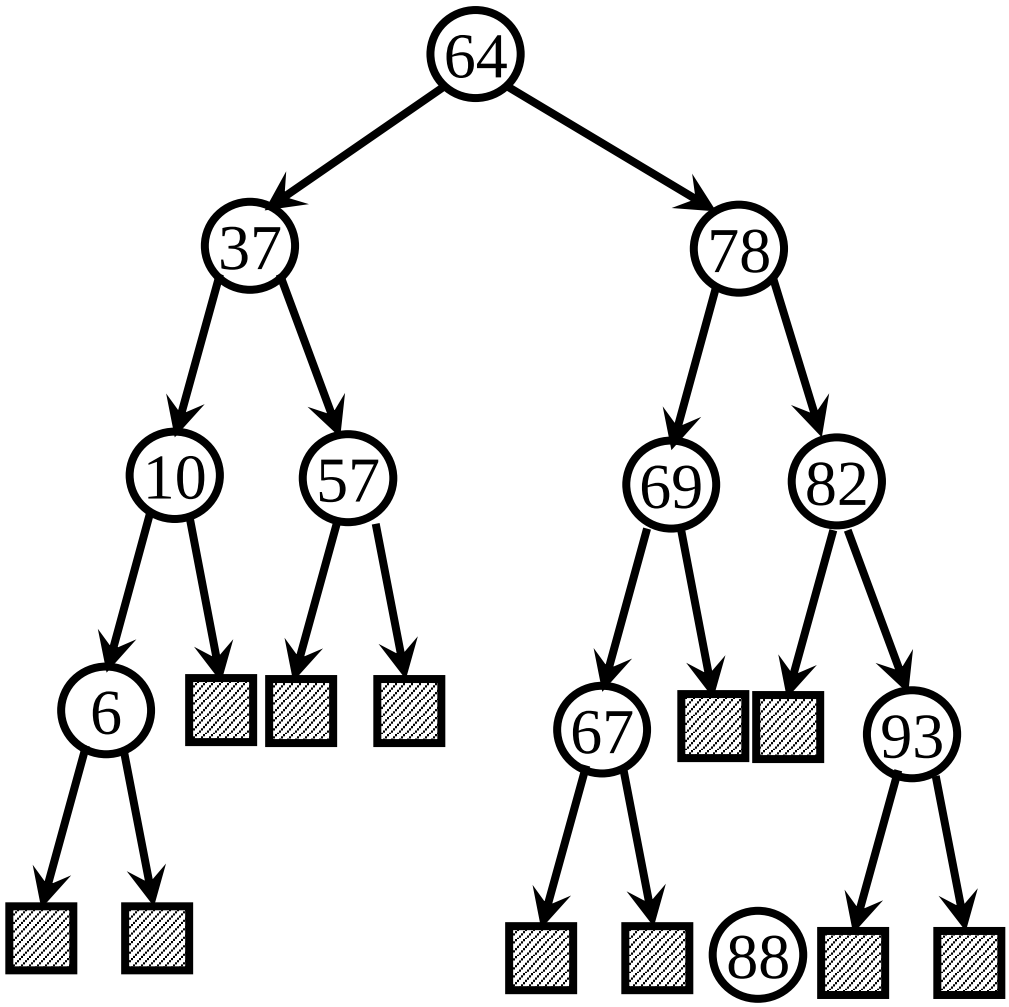
6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM



TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

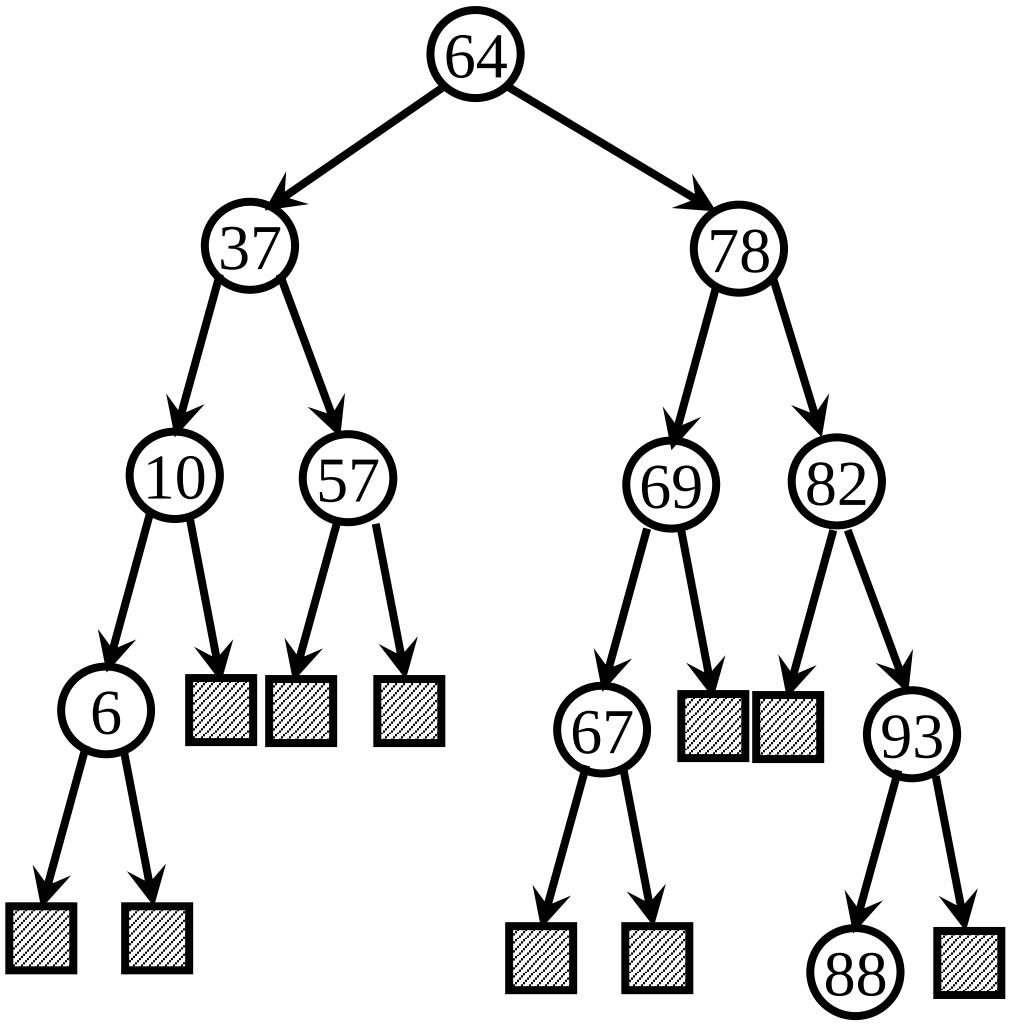
6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM



TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

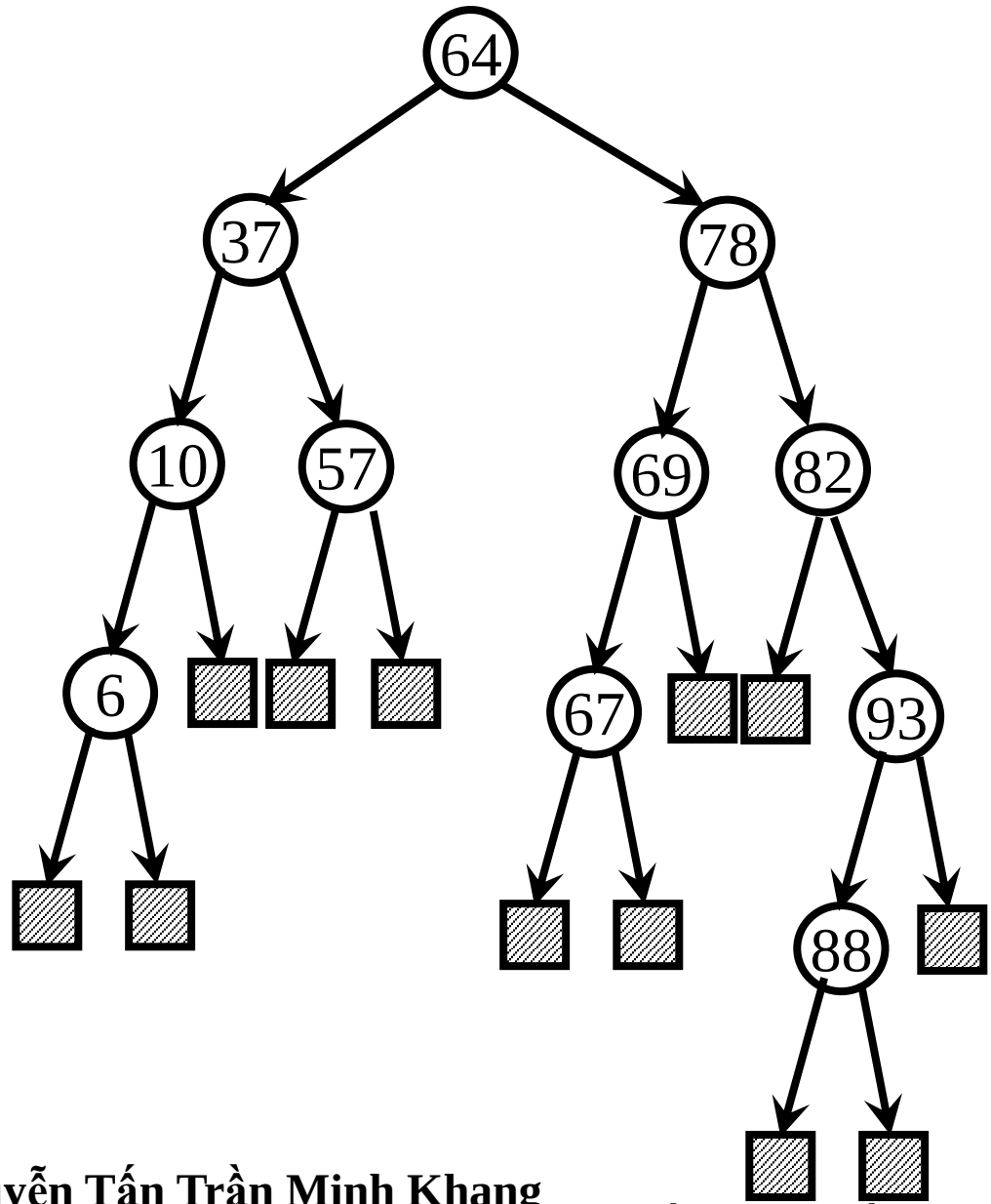
6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM



TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM



TS. Nguyễn Tấn Trần Minh Khang

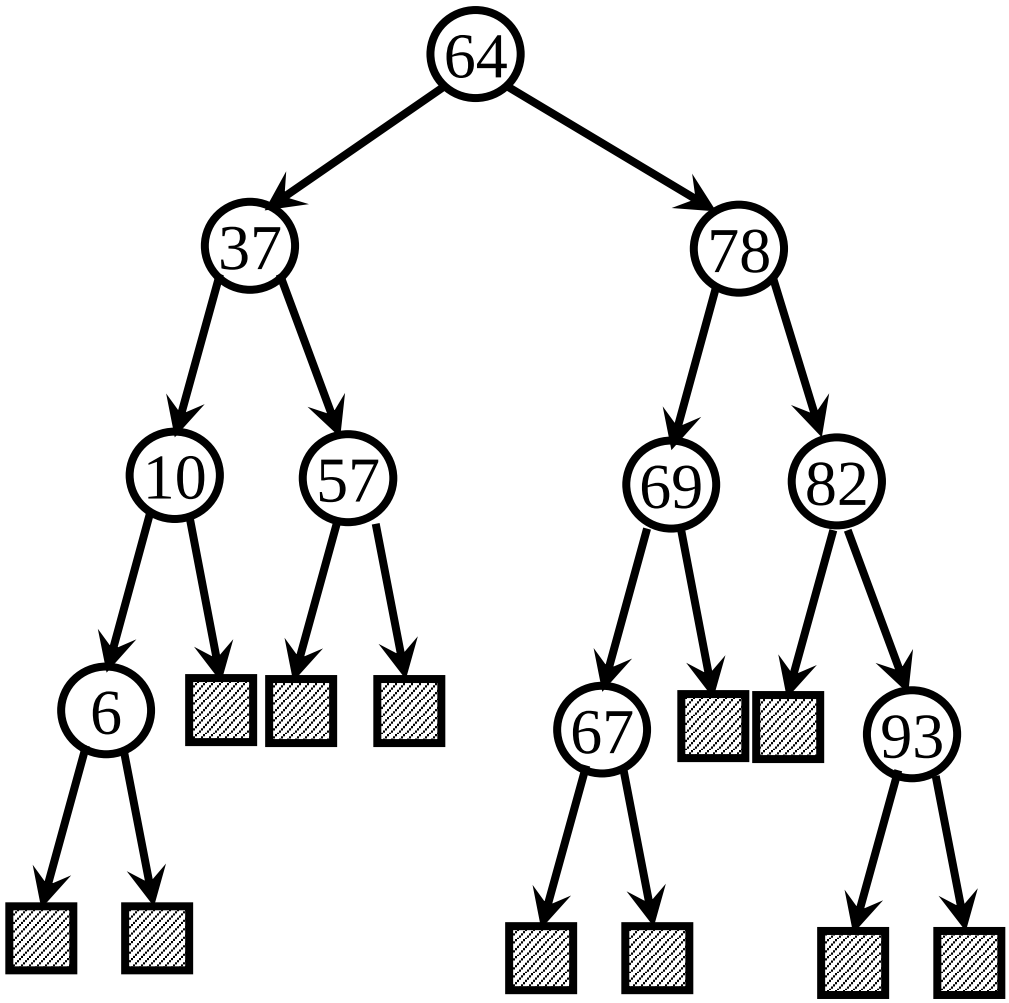
ThS. Cáp Phạm đình Thăng

6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM

- Vấn đề 1: Thêm một node vào cây nhị phân tìm kiếm các số nguyên.
- Định nghĩa hàm

```
11. int InsertNode(TREE &t,int x)
12. {
13.   if(t!=NULL)
14.   {
15.     if(t->info<x)
16.       return InsertNode(t->pRight,x);
17.     if(x<t->info)
18.       return InsertNode(t->pLeft,x);
19.     return 0;
20.   }
21.   …
22. }
```

6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM



TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM

- Vấn đề 1: Thêm một node vào cây nhị phân tìm kiếm các số nguyên.
- Định nghĩa hàm

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

6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM

- Vấn đề 2: Thêm một node vào cây nhị phân tìm kiếm các số thực.
- Định nghĩa hàm

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

6. THÊM GIÁ TRỊ VÀO CÂY NHỊ PHÂN TÌM KIẾM

- Vấn đề 3: Thêm một node vào cây nhị phân tìm kiếm trừu tượng.
- Định nghĩa hàm

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

7. NHẬP CÂY NHỊ PHÂN TÌM KIẾM

- Vấn đề 1: Định nghĩa hàm nhập cây nhị phân tìm kiếm ở mức trừu tượng.
- Định nghĩa hàm trừu tượng

```
11. void Input(TREE &t)
12. {
13.     int n;
14.     printf("Nhap n:");
15.     scanf("%d",&n);
16.     Init(t);
17.     for(int i=1;i<=n;i++)
18.     {
19.         KDL x;
20.         Nhap(x);
21.         InsertNode(t,x);
22.     }
23. }
```

7. NHẬP CÂY NHỊ PHÂN TÌM KIẾM

- Vấn đề 2: Định nghĩa hàm nhập cây nhị phân tìm kiếm các số nguyên.
- Định nghĩa hàm

```
11. void Input(TREE &t)
12. {
13.     int n;
14.     printf("Nhap n:");
15.     scanf("%d",&n);
16.     Init(t);
17.     for(int i=1;i<=n;i++)
18.     {
19.         int x;
20.         printf("Nhap so…:");
21.         scanf("%d",&x);
22.         InsertNode(t,x);
23.     }
24. }
```

7. NHẬP CÂY NHỊ PHÂN TÌM KIẾM

- Vấn đề 3: Định nghĩa hàm nhập cây nhị phân tìm kiếm các số thực.
- Định nghĩa hàm

```
11. void Input(TREE &t)
12. {
13.     int n;
14.     printf("Nhap n:");
15.     scanf("%d",&n);
16.     Init(t);
17.     for(int i=1;i<=n;i++)
18.     {
19.         float x;
20.         printf("Nhap so…:");
21.         scanf("%f",&x);
22.         InsertNode(t,x);
23.     }
24. }
```

8. DUYỆT CÂY NHỊ PHÂN TÌM KIẾM

- Khái niệm: Duyệt cây nhị phân tìm kiếm là thăm qua tất cả các node trong cây mỗi node một lần
- Định nghĩa hàm trừu tượng

```
1. KDL Process(TREE t)
2. {
3.     if(t==NULL)
4.         return ...
5.     ...Process(t->pLeft);
6.     ...<Duyet nút gốc>...
7.     ...Process(t->pRight);
8.     return ...
9. }
```

8. DUYỆT CÂY NHỊ PHÂN TÌM KIẾM

- Ví dụ 1: Định nghĩa hàm xuất tất cả các node trong cây nhị phân tìm kiếm các số nguyên.
- Định nghĩa hàm

```
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.     return;
9. }
```

8. DUYỆT CÂY NHỊ PHÂN TÌM KIẾM

- Ví dụ 2: Định nghĩa hàm đếm số lượng số chính phương trong cây nhị phân tìm kiếm các số nguyên.
- Định nghĩa hàm

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

9. MỘT CHƯƠNG TRÌNH ĐƠN GIẢN VỀ CÂY BST

- Bài toán: Viết chương trình thực hiện các yêu cầu sau
  - Nhập cây nhị phân tìm kiếm các số thực.
  - Xuất các giá trị trong cây ra màn hình.
  - Tính tổng các giá trị dương có trong cây.
- Chương trình

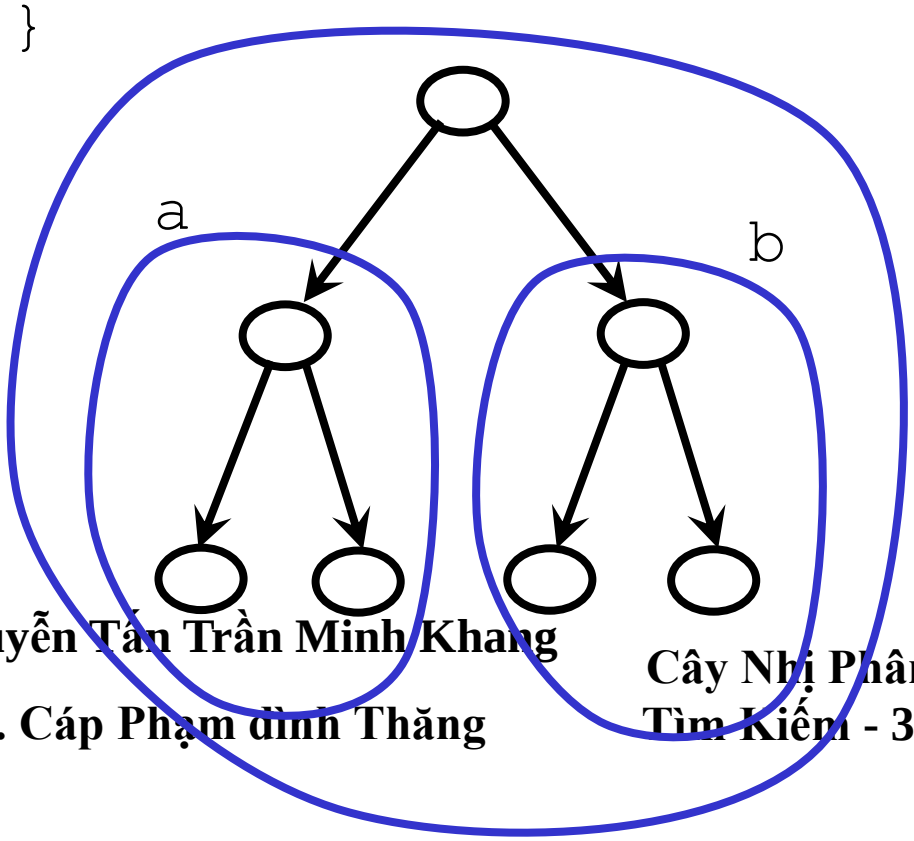
```
11.#include "stdafx.h"
12.#include "stdio.h"
13.#include "conio.h"
14.#include "math.h"
15.#include "string.h"
16.struct node
17.{
18.    float info;
19.    struct node *pLeft;
20.    struct node *pRight;
21.};
22.typedef struct node NODE;
23.typedef NODE*TREE;
24.void Init(TREE&);
25.NODE*GetNode(float);
26.int InsertNode(TREE&,float);
27.void Input(TREE&);
28.void Output(TREE);
29.float TongDuong(TREE);
```

```
23.void main()
24.{
25.    TREE tree;
26.    Input(tree);
27.    Output(tree);
28.    float s=TongDuong(tree);
29.    printf("\nTong la: %8.3f",s);
30.    return;
31.}
32.NODE* GetNode(float x)
33.{
34.    NODE *p = new NODE;
35.    if(!p)
36.        return NULL;
37.    p->info = x;
38.    p->pLeft  = NULL;
39.    p->pRight = NULL;
40.    return p;
41.}
```

```
42. int InsertNode(TREE&t,float x)
43. {
44.   if (t!=NULL)
45.   {
46.     if (x<t->info)
47.       return InsertNode(t->pLeft,x);
48.     if (x>t->info)
49.       return InsertNode(t->pRight,x);
50.     return 0;
51.   }
52.   t = GetNode(x);
53.   if(t==NULL)
        return 0;
42.   return 1;
43. }
44. void Init(TREE &t)
45. {
46.    t=NULL;
47. }
```

```
61. void Input(TREE&t)
62. {
63.     int n;
64.     printf("Nhap n:");
65.     scanf("%d",&n);
66.     Init(t);
67.     for(int i=1;i<=n;i++)
68.     {
69.         float x;
70.         printf("Nhap so thuc:");
71.         scanf("%f",&x);
72.         InsertNode(t,x);
73.     }
74. }
75. void Output(TREE t)
76. {
77.     if(t==NULL)
78.         return;
79.     Output(t->pLeft);
80.     printf("%8.3f",t->info);
81.     Output(t->pRight);
82. }
```

```
83. float TongDuong(TREE t)
84. {
85.     if(t==NULL)
86.         return 0;
87.     float a=TongDuong(t->pLeft);
88.     float b=TongDuong(t->pRight);
89.     if (t->info>0)
90.         return (a+b+t->info);
91.     return a+b;
92. }
```



10. BÀI TẬP

- Bài toán: Viết chương trình thực hiện các yêu cầu sau
  - Nhập cây nhị phân tìm kiếm các số nguyên.
  - Xuất các giá trị trong cây ra màn hình.
  - Đếm số lượng các giá trị chẵn có trong cây.
- Chương trình

THU HỒI BỘ NHỚ

- Vấn đề: Định nghĩa hàm thu hồi tất cả các bộ nhớ đã cấp phát cho cây nhị phân tìm kiếm các số nguyên.

```
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;
```

THU HỒI BỘ NHỚ

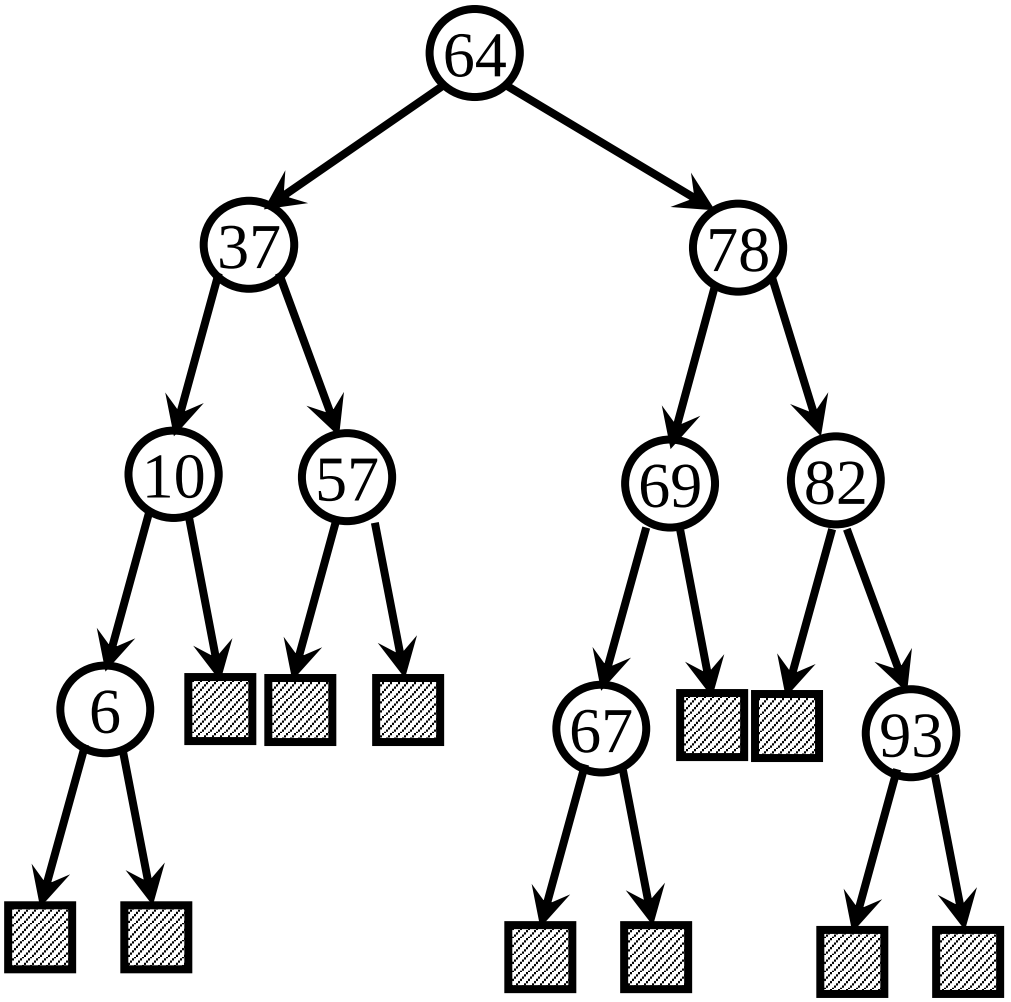
- Vấn đề: Định nghĩa hàm thu hồi tất cả các bộ nhớ đã cấp phát cho cây nhị phân tìm kiếm các số nguyên.
- Định nghĩa hàm trừu tượng

```
1. void RemoveAll(TREE &t)
2. {
3.     if(t==NULL)
4.         return;
5.     RemoveAll(t->pLeft);
6.     RemoveAll(t->pRight);
7.     delete t;
8. }
```

XÓA PHẦN TỬ TRÊN CÂY NHỊ PHÂN TÌM KIẾM

- Thao tác xóa một phần tử trên cây nhị phân tìm kiếm.
  - Áp dụng giải thuật tìm kiếm để xác định địa chỉ của NODE cần xóa.
  - Nếu tìm thấy xóa NODE (phần tử) đó khỏi cây.
- Các trường hợp xẩy ra khi xóa NODE.
  - NODE cần xóa là NODE lá.
  - NODE cần xóa có duy nhất một con.
  - NODE cần xóa có đủ hai con.

XÓA PHẦN TỬ TRÊN CÂY NHỊ PHÂN TÌM KiẾM



TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

XÓA PHẦN TỬ TRÊN CÂY NHỊ PHÂN TÌM KiẾM

- Vấn đề: Định nghĩa hàm xóa một NODE có giá trị x trên cây nhị phân tìm kiếm 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;
```

XÓA PHẦN TỬ TRÊN CÂY NHỊ PHÂN TÌM KiẾM

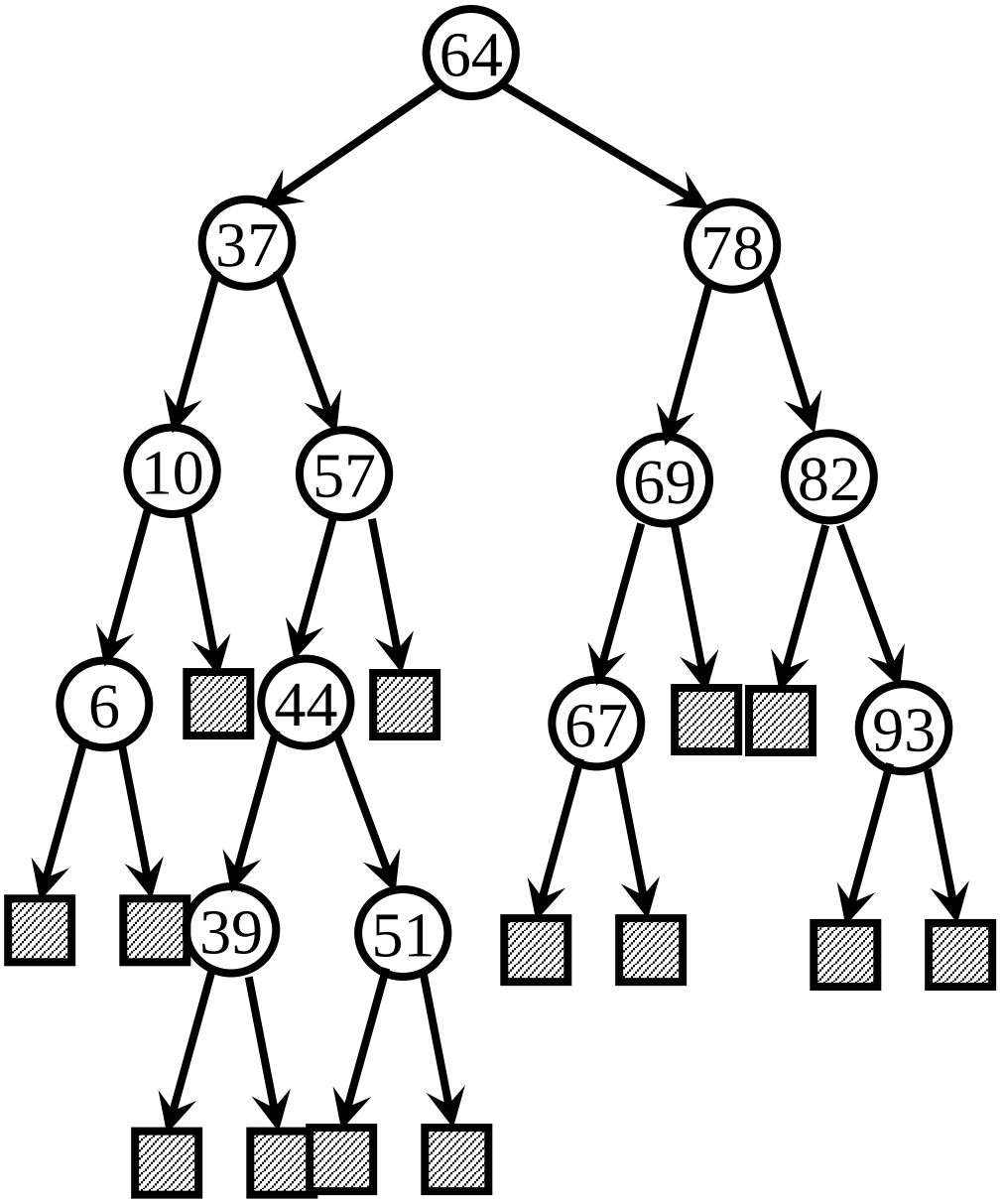
- Định nghĩa hàm

```
11. int BSTDelete(TREE &t,int x)
12. {
13.   if(t==NULL)
14.     return 0;
15.   if(t->info>x)
16.     return BSTDelete(t->pLeft,x);
17.   if(t->info<x)
18.     return BSTDelete(t->pRight,x);
19.   //Tìm thấy phần tử cần xóa
20.   _Delete(t);
21.   return 1;
22. }
```

- Định nghĩa hàm

```
11. void _Delete(TREE &t)
12. {
13.     NODE*temp = t;
14.     if(!t->pLeft&&!t->pRight)
15.     {
16.         t = NULL;
17.         delete temp;
18.         return;
19.     }
20.     if(t->pLeft&&!t->pRight)
21.     {
22.         t = t->pLeft;
23.         delete temp;
24.         return;
25.     }
26.     if(!t->pLeft&&t->pRight)
27.     {
28.         t = t->pRight;
29.         delete temp;
30.         return;
31. }
```

XÓA PHẦN TỬ TRÊN CÂY NHỊ PHÂN TÌM KiẾM



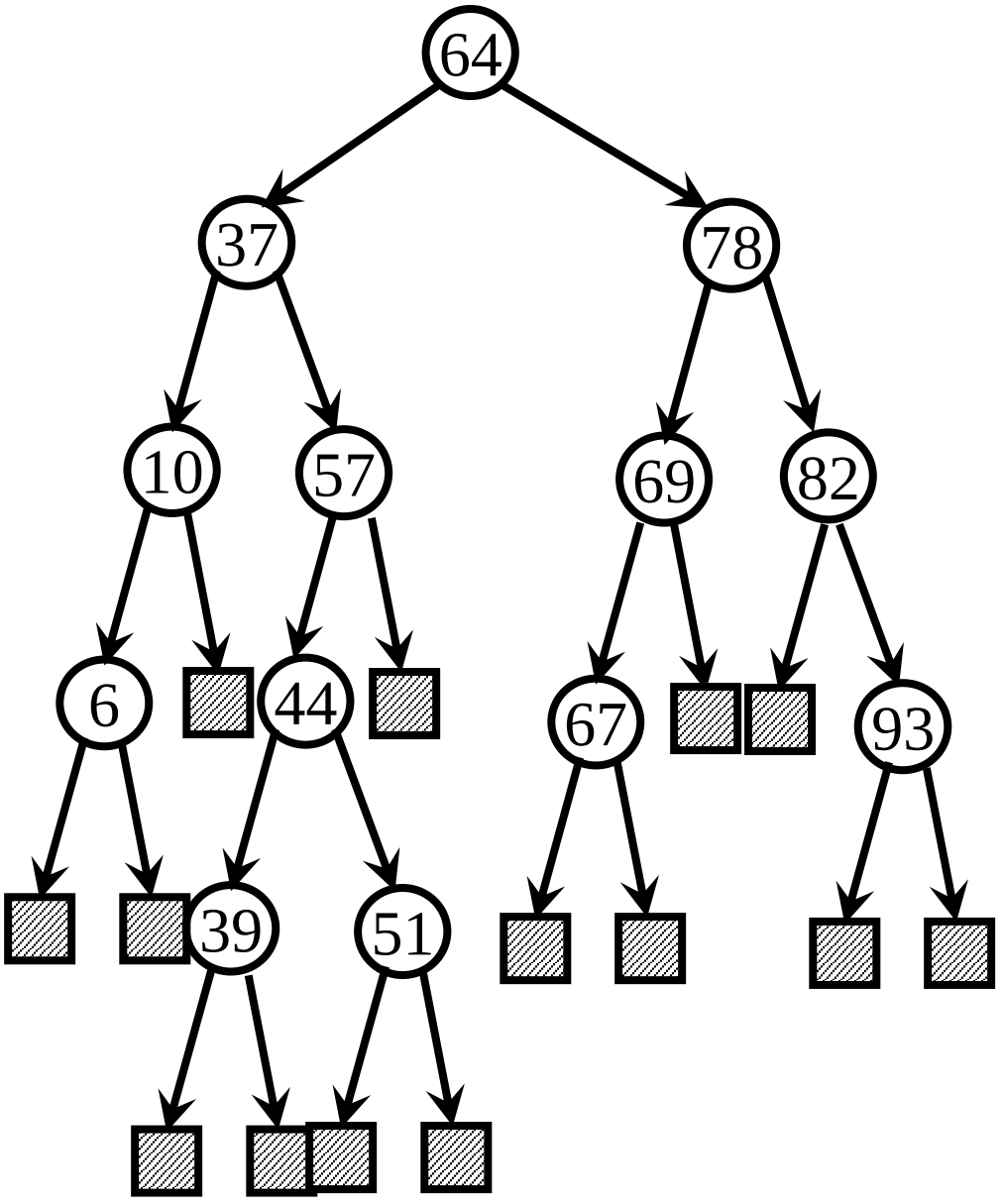
TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

- Định nghĩa hàm

```
11. void _Delete(TREE &t)
12. {
13.   …
14.   temp=SearchStandFor(t->pLeft,t);
15.   delete temp;
16. }
17. NODE*SearchStandFor(TREE&p,
18.                     TREE&q)
19. {
20.   if(p->pRight)
21.     return SearchStandFor(
22.                 p->pRight,q);
23.   //Đã tới nơi.
24.   NODE *temp = p;
25.   q->info = p->info;
26.   p = p->pLeft;
27.   return temp;
28. }
```

XÓA PHẦN TỬ TRÊN CÂY NHỊ PHÂN TÌM KiẾM



TS. Nguyễn Tấn Trần Minh Khang

ThS. Cáp Phạm đình Thăng

- Định nghĩa hàm

```
11. void _Delete(TREE &t)
12. {
13.   …
14.   temp=SearchStandFor(t->pRight,t);
15.   delete temp;
16. }
17. NODE*SearchStandFor(TREE&p,
18.                     TREE&q)
19. {
20.   if(p->pLeft)
21.     return SearchStandFor(
22.                   p->pLeft,q);
23.   //Đã tới nơi.
24.   NODE *temp = p;
25.   q->info = p->info;
26.   p = p->pRight;
27.   return temp;
28. }
```

NHẬP CÂY TỪ TẬP TIN

- Bài toán: Định nghĩa hàm nhập cây nhị phân tìm kiếm các số nguyên từ tập tin nhị phân. Biết rằng tập tin nhị phân lần lượt lưu các giá trị trong cây.

NHẬP CÂY TỪ TẬP TIN

```
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;
```

NHẬP CÂY TỪ TẬP TIN

- 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.
- Định nghĩa hàm

```
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. }
```

NHẬP CÂY TỪ TẬP TIN

- Định nghĩa hàm

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

NHẬP CÂY TỪ TẬP TIN

```
10. int Input(char*fn,TREE&t)
11. {
12.     FILE*fp=fopen(fn,"rb");
13.     if(fp==NULL)
14.         return 0;
15.     int temp;
16.     Init(t);
17.     while(fread(&temp,
18.         sizeof(int),1,fp)==1)
19.     {
20.         InsertNode(t,temp);
21.     }
22.     fclose(fp);
23.     return 1;
24. }
```