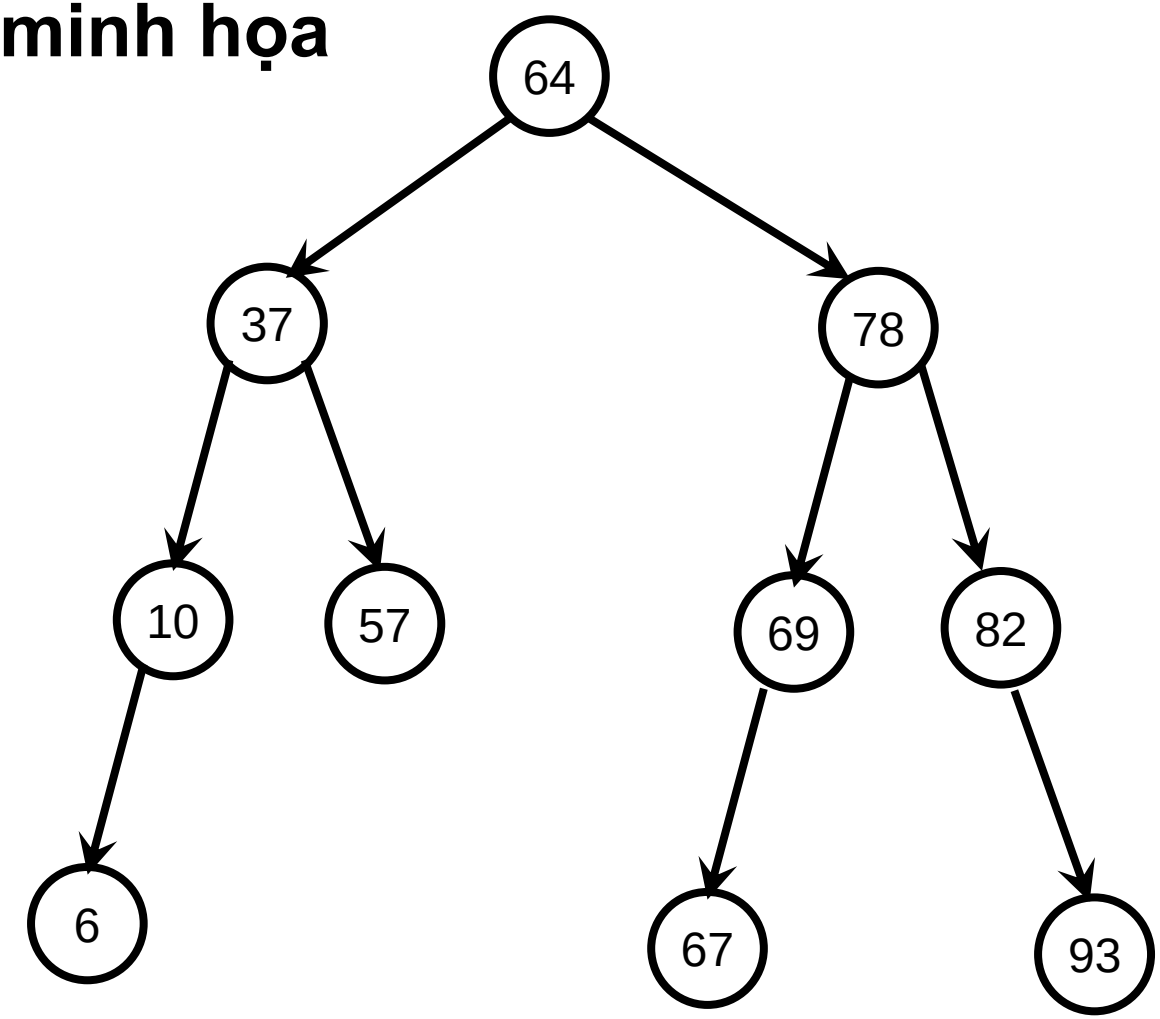
CÂY NHỊ PHÂN TÌM KIẾM
(BINARY SEARCH TREE)

ThS.Nguyễn Thúy Loan

CÂY NHỊ PHÂN TÌM KIẾM

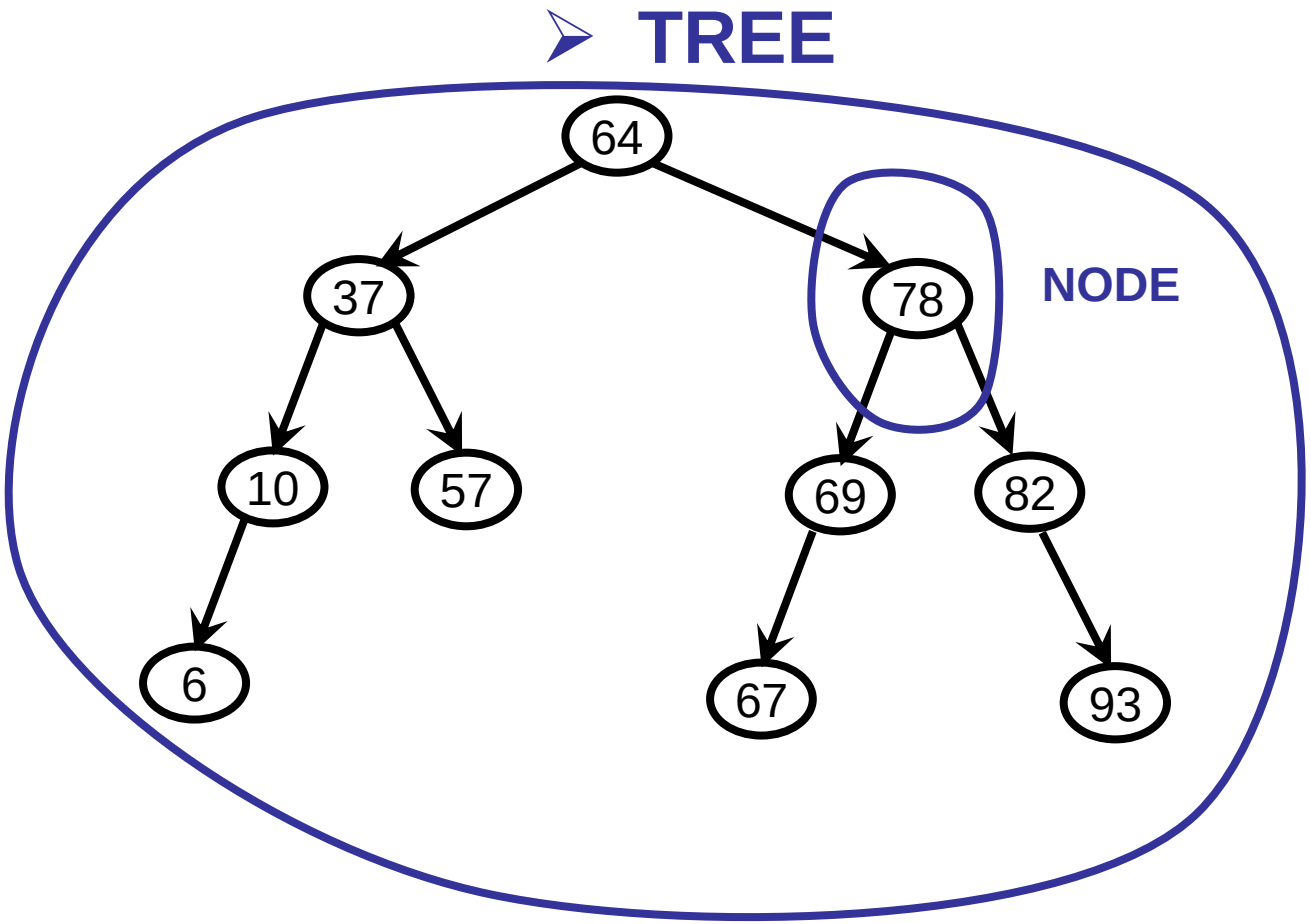
Hình ảnh minh họa



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.

Hình ảnh minh họa



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.

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 RỖNG

- Khái niệm: Cây nhị phân rỗng là cây không chứa một node nào. Hàm sẽ trả về giá trị 1 nếu cây nhị phân rỗng. Ngược lại hàm trả về giá trị 0.
- Định nghĩa hàm

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

5. TẠO NODE CHO CÂY NHỊ PHÂN TÌM KIẾM

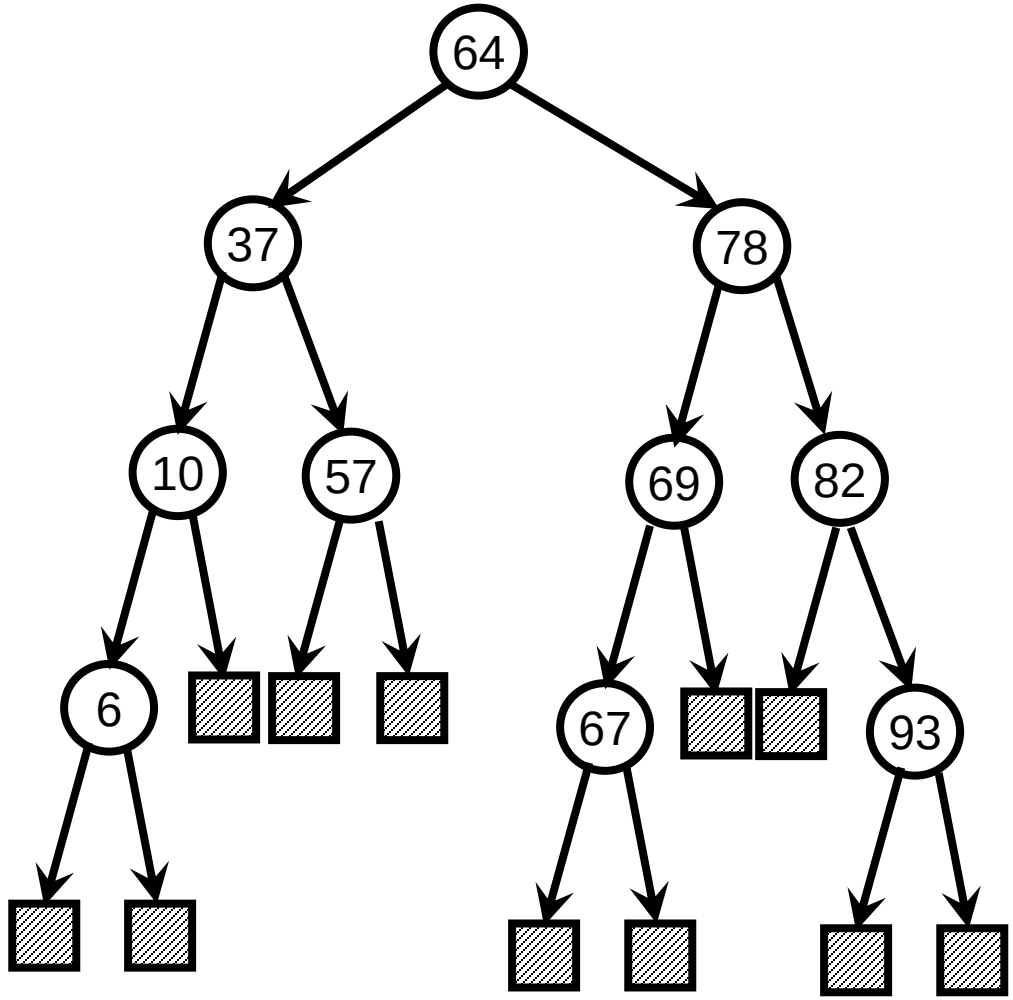
- Khái niệm: Tao node cho cây nhị phân tìm kiếm là xin cấp phát bộ nhớ có kích thước bằng kích thước của KDL NODE để chứa thông tin biết trước.
- Định nghĩa hàm

```
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.}
```

6. THÊM NODE CHO CÂY NHỊ PHÂN TÌM KIẾM

- Khái niệm: Thêm một node 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 node vào trong cây nhị phân tìm kiếm trả về một trong 3 giá trị -1, 0, 1 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 NODE CHO CÂY NHỊ PHÂN TÌM KIẾM



6. THÊM NODE CHO CÂY NHỊ PHÂN TÌM KIẾM

- Vấn đề 1: Hãy định nghĩa hàm thêm một node (thông tin) vào 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;
```

6. THÊM NODE CHO 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 NODE CHO 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 NODE CHO CÂY NHỊ PHÂN TÌM KIẾM

```
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(t->info > x)
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

- 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

Nhập cây nhị phân tìm kiếm các số nguyên.

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

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.{    if(t==NULL)
3.         return ...
4.     ...Process(t->pLeft);
5.     ...
6.     ...Process(t->pRight);
7.     return ...
8.}
```

8. DUYỆT CÂY NHỊ PHÂN TÌM KIẾM

VD : Định nghĩa hàm xuất tất cả các node trong cây nhị phân tìm kiếm các số nguyên.

```
1.void Xuat(TREE t)// LNR
2.{
3.     if(t==NULL)
4.          return;
5.     Xuat(t->pLeft);
6.     printf("%4d",t->info);
7.     Xuat(t->pRight);
8.     return;
9.}
```

9. MỘT CHƯƠNG TRÌNH ĐƠN GIẢN VỀ CÂY BST

- 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

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;
```

Chương trình

```
11.void Init(TREE&);
12.NODE*GetNode(float);
13.int InsertNode(TREE&,float);
14.void Input(TREE&);
15.void Output(TREE);
16.float TongDuong(TREE);
```

Chương trình

```
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. }
```

Chương trình

```
23.NODE* GetNode(float x)
24.{
25.      NODE *p = new NODE;
26.      if(!p)
27.                return NULL;
28.      p->info = x;
29.      p->pLeft  = NULL;
30.      p->pRight = NULL;
31.      return p;
32.}
```

Chương trình

```
42.int InsertNode(TREE&t,float x)
43.{ if (t!=NULL)
44.   {
45.     if (t->info > x)
46.       return InsertNode(t->pLeft,x);
47.     if (t->info < x)
48.       return InsertNode(t->pRight,x);
49.     return 0;
50.   }
51.     t = GetNode(x);
52.     if(t==NULL) return 0;
53.     return 1;
54.}
```

Chương trình

```
42.void Init(TREE &t)
43.{
44.    t=NULL;
45.}
```

Chương trình

```
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.}
```

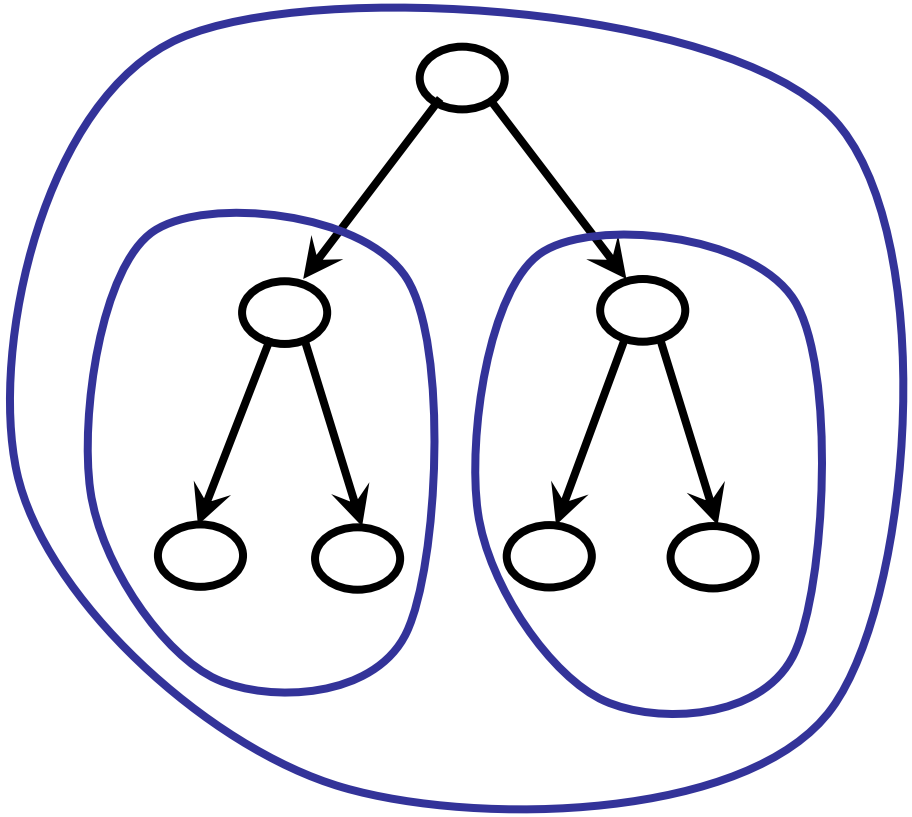
Chương trình

```
61.void Output(TREE t)
62.{
63.      if(t==NULL)
64.                  return;
65.    Output(t->pLeft);

66.    printf("%8.3f",t->info);

67.    Output(t->pRight);
68.}
```

TÍNH TỔNG



Chương trình

```
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.}
```

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)  return;
4.     RemoveAll(t->pLeft);
5.     RemoveAll(t->pRight);
6.     delete t;
7.}
```