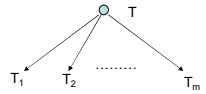
Cây (tree)

1

1

Cây

- Định nghĩa:
  - Rỗng là 1 cây
  - T_1 , T_2 , ……T_m là cây thì



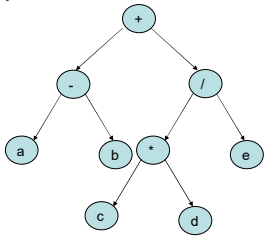
là cây (m-phân)

- T_1 , T_2 , ……T_m gọi là cây con của cây T

2

2

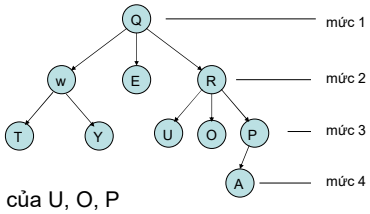
- Ví dụ
  - Cây thư mục trong hệ điều hành
  - Cây gia phả của 1 dòng họ
  - Cây biểu thức: a-b+c*d/e



3

3

Một số khái niệm



- R là cha của U, O, P
- U, O, P là con của R
- Q, R là đỉnh trước của A.
- A là đỉnh sau của Q, R
- T, Y, U, O, A là các nút lá (không có con)
- Q là nút gốc

4

4

Phép duyệt

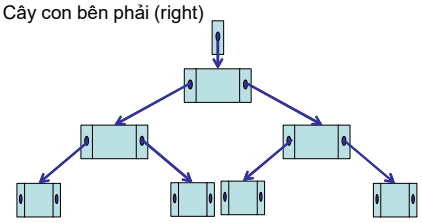
- Phép duyệt: Đưa ra tất cả các nút theo một thứ tự nào đó, mỗi nút 1 lần
  - Ví dụ: Tìm kiếm một file hay một folder trong cây thư mục
- Phổ biến:
  - Duyệt theo bề sâu (Depth First Search - DFS)
  - Duyệt theo bề rộng (Breadth First Search - BFS)

5

5

Cây nhị phân (Binary tree)

- Là cây bao gồm các nút, mỗi nút có tối đa 2 cây con:
  - Cây con bên trái (left)
  - Cây con bên phải (right)



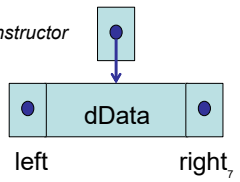
6

6

Cây nhị phân (Binary tree)

- Mô tả cấu trúc:

```
class Node
{
    public int dData;
    public Node left, right;
    public Node(int dd)      //constructor
    {
        dData = dd;
        left = right = null;
    }
} // end class Node
```



7

Cây nhị phân (Binary tree)

```
class Tree
{
    private Node root;
    public Tree() {
        root = null;
    }
    public boolean isEmpty() {
        return root==null;
    }
    // ……  Các method khác……
}
```

8

8

Các method

- Lấy dữ liệu tại gốc

```
public int getData()
{
    return root.dData;
}
```

Gán dữ liệu tại gốc

```
public void setData(int dd)
{
    root.dData = dd;
}
```

9

9

Các method

- Lấy cây con bên trái

```
public Tree getLeft()
{
    Tree t = new Tree();
    t.root = root.left;
    return t;
}
```

10

10

Các method

- Lấy cây con bên phải

```
public Tree getRight()
{
    Tree t = new Tree();
    t.root = root.right;
    return t;
}
```

11

11

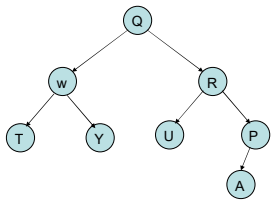
Duyệt theo bề sâu (DFS)

- Duyệt tiền tự (Preorder Search – ***N***LR)
  - Thăm nút gốc (*Node*)
  - Duyệt cây con bên trái (*Left*)
  - Duyệt cây con bên phải (*Right*)
- Duyệt trung tự (Inorder Search – L***N***R)
  (tương tự)
- Duyệt hậu tự (Postorder Search – LR***N***)
  (tương tự)

12

12

Duyệt theo bề sâu (DFS)



- NLR: QWTYRUPA
- LNR: TWYQURAP
- LRN: TYWUAPRQ

13

13

Ví dụ: Đếm số nút trên cây

```
public static int Sonut(Tree t)
{
      if  (t.isEmpty())
            return 0;
      else
            return 1 + Sonut(t.getLeft()) +
                              Sonut(t.getRight());
}
```

14

14

Ví dụ: Tổng giá trị các nút trên cây

```
public static int tong(Tree t)
{
      if  (t.isEmpty())
            return 0;
      else
            return t.getData() + tong(t.getLeft()) +
                                 tong(t.getRight());
}
```

15

15

Ví dụ: Đếm số nút lá trên cây

```
public static int Sonutla(Tree t)
{
   if(t.isEmpty())
      return 0;
   if (t.getLeft().isEmpty() &&
                        t.getRight().isEmpty())
      return 1;
  return Sonutla(t.getLeft())+Sonutla(t.getRight());
}
```

16

16

Ví dụ: Độ cao cây

```
private static int max(int a, int b )
{
      return (a>b? a: b);
}
public static int docao(Tree t)
{
   if(t.isEmpty())
      return 0;
   return 1 + max(docao(t.getLeft()),
                                 docao(t.getRight()));
}
```

17

17

Ví dụ: In cây (số nguyên)

```
public static void displayTree(Tree t, int k)
{
   if(!t.isEmpty())
   {
      displayTree(t.getLeft(), k+1);
      System.out.print("\n"+blank(4*k)+ t.getData());
      displayTree(t.getRight(), k+1);
   }
}
```

18

18

Ví dụ: In cây (số nguyên)

```
private static String blank(int n)
{
        String s = "";
        for (int i=0; i<n; i++)
              s = s + " ";
        return s;
}
public static void displayTree(Tree t)
{
        displayTree(t, 1);
}
```

19

19

Ví dụ: Duyệt tiền tự (NLR)

```
public static void preOrder(Tree t)
{
    if  (!t.isEmpty())
    {
        System.out.print(t.getData() + " ");
        preOrder(t.getLeft());
        preOrder(t.getRight());
    }
}
```

20

20

Ví dụ: Duyệt trung tự (LNR)

```
public static void inOrder(Tree t)
{
    if  (!t.isEmpty())
    {
        inOrder(t.getLeft());
        System.out.print(t.getData() + " ");
        inOrder(t.getRight());
     }
}
```

21

21

Ví dụ: Duyệt hậu tự (LRN)

```
public static void postOrder(Tree t)
{
    if(!t.isEmpty())
    {
        postOrder(t.getLeft());
        postOrder(t.getRight());
        System.out.print(t.getData() + " ");
     }
}
```

22

22

Đếm số nút trên mức chẵn/lẻ của cây

```
private static int Snmcl(Tree t, int m)
{
   if  (t.isEmpty())
       return 0;
   else
       return m%2+Snmcl(t.getLeft(), m+1) +
                          Snmcl(t.getRight(), m+1);
}
```

23

23

Đếm số nút trên mức chẵn/lẻ của cây

- Số nút mức lẻ

```
public static int Snml(Tree t)
{
   return Snmcl(t, 1);
}
```

- Số nút mức chẵn

```
   public static int Snmc(Tree t)
   {
       return Snmcl(t, 0);
   }
```

24

24

Ví dụ: Đếm số nút trên mức thứ k

```
public static int Snmk(Tree t, int k)
{
    if  (t.isEmpty())
        return 0;
    if (k==1)
        return 1;
   return Snmk(t.getLeft(), k-1) + Snmk(t.getRight(), k-1);
}
```

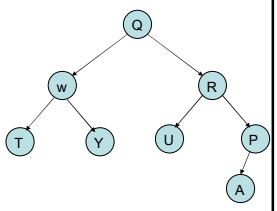
25

25

Duyệt theo bề rộng (BFS)

- Hay còn gọi là duyệt theo mức
- Sử dụng hàng đợi: Tuần tự từ mức thấp đến cao, từ trái qua phải
- Với cây trên ta có thứ tự các nút được duyệt lần lượt:

Q W R T Y U P A



SV cài đặt như bài tập

26

26

1. Khởi tạo hàng đợi q rỗng
2. M=0; p=NULL;
3. Nếu r!=NULL
   - Đưa p vào q
   - Đưa r vào q
4. Khi q!=rỗng
   - Lấy p ra khỏi q
   - Nếu p==NULL
     - Nếu q!=rỗng
       - //Xử lý hết mức
       - M = M+1;
       - Đưa p vào q
   - Ngược lại
     - Xử lý p.data
     - Nếu p.left != NULL thì Đưa p.left vào q
     - Nếu p.right != NULL thì Đưa p.right vào q

27

27

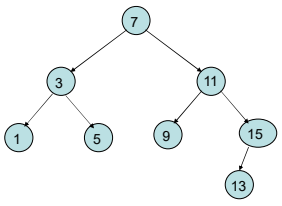
Cây tìm kiếm nhị phân (**B**inary **S**earch **T**ree – BST)

- Ý nghĩa: Phục vụ tìm kiếm nhị phân trên cấu trúc động
- Định nghĩa: Là cây nhị phân thỏa điều kiện mọi nút đều có khóa
  - Lớn hơn khóa tất cả các nút trên cây con bên trái
  - Nhỏ hơn khóa tất cả các nút trên cây con bên phải

28

28

BST

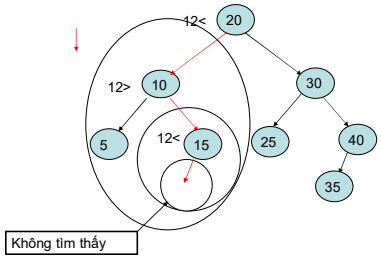


29

29

Tìm phần tử trên BST

- Tìm 12

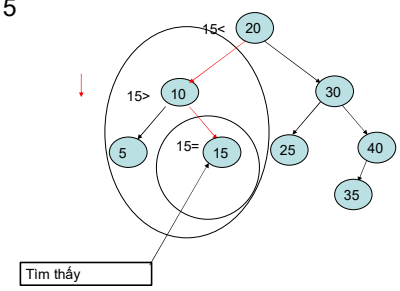


30

30

Tìm phần tử trên BST

- Tìm 15



31

31

Tìm kiếm phần tử x

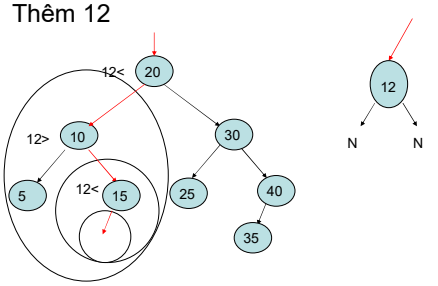
```
public Tree search(int key)
{   Tree t = new Tree();
    Node c = root;
    while (c.dData  != key)   {
        if (key < c.dData)
            c = c.left;
        else
            c = c.right;
        if(c == null) {
            t.root = null;
            return t;
        }
    }
    t.root = c;
     return t;
  }
```

32

32

Chèn phần tử vào BST

- Thêm 12

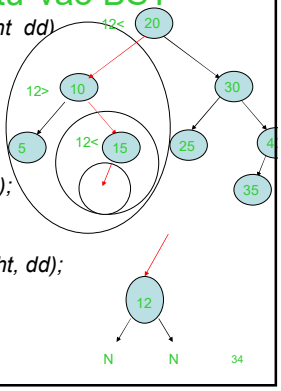


33

33

Chèn phần tử vào BST

```
private Node insert(Node r, int  dd)
{
   if (r==null)
      r = new Node(dd);
   else
      if (dd < r.dData)
         r.left = insert(r.left, dd);
      else
         if(dd > r.dData)
            r.right = insert(r.right, dd);
      return r;
}
```



34

34

Chèn phần tử vào BST

```
public void insertBST(int dd)
{
      root = insert(root, dd);
}
```

35

35

Ví dụ: Nhập cây (số nguyên >0)

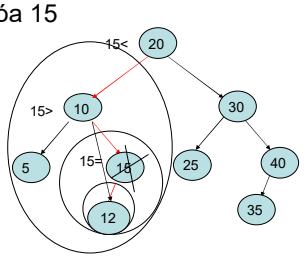
```
public static void inputTree(Tree t)
{
    int xa;
    Scanner x;
    x = new Scanner(System.in);
    System.out.print("Nhap day (0 de ket thuc nhap):");
    do {
        xa= x.nextInt();
        if (xa>0)
            t.insertBST(xa);
    } while (xa>0);
}
```

36

36

Xóa phần tử trên BST

- Xóa 15

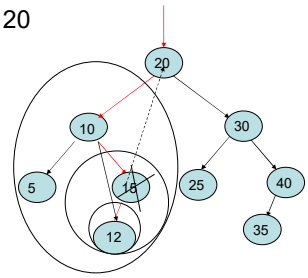


37

37

Xóa phần tử trên BST

- Xóa 20

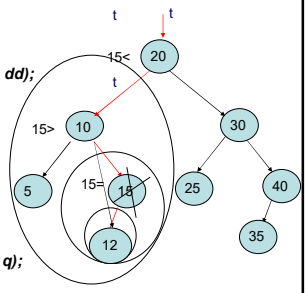


38

38

Xóa phần tử trên BST

```
private Node delete(Node t, int dd)
{
    if  (t!=null)
        if  (dd < t.dData)
            t.left = delete(t.left, dd);
        else
            if (dd > t.dData)
                t.right = delete(t.right, dd);
            else
            {
                Node q = t;
                if (t.left == null)
                    t = t.right;
                else
                    if (t.right == null)
                        t = t.left;
                    else
                        t.left = del(t.left, q);
            }
    return t;
}
```

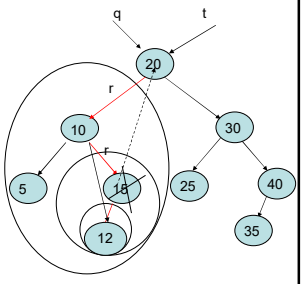


39

39

Tìm phần tử thay thế (phần tử lớn nhất)

```
private Node del(Node r, Node q)
{
    if (r.right!=null)
        r.right = del(r.right, q);
    else
    {
        q.dData = r.dData;
        r = r.left;
    }
    return r;
}
public void deleteBST(int dd)
{
    root = delete(root, dd);
}
```



40

40