Đại Học Bách Khoa Hà Nội
Viện Điện Tử - Viễn Thông

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Chương 11: Cấu trúc cây
Phần 1: Cơ bản về cây

Nội dung

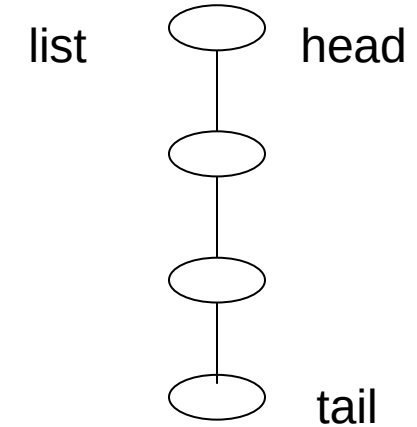
- Các khái niệm cơ bản
  - Mô tả cấu trúc cây (tree)
  - Các khái niệm cơ bản trong cây
  - Các tính chất cơ bản
  - Các thao tác cơ bản
- Cây nhị phân (binary tree)
  - Khái niệm
  - Phân loại
- Các trường hợp đặc biệt
  - Cây suy biến (degenerate binary tree)
  - Cây đầy đủ (full binary tree)
  - Cây hoàn chỉnh (complete binary tree)

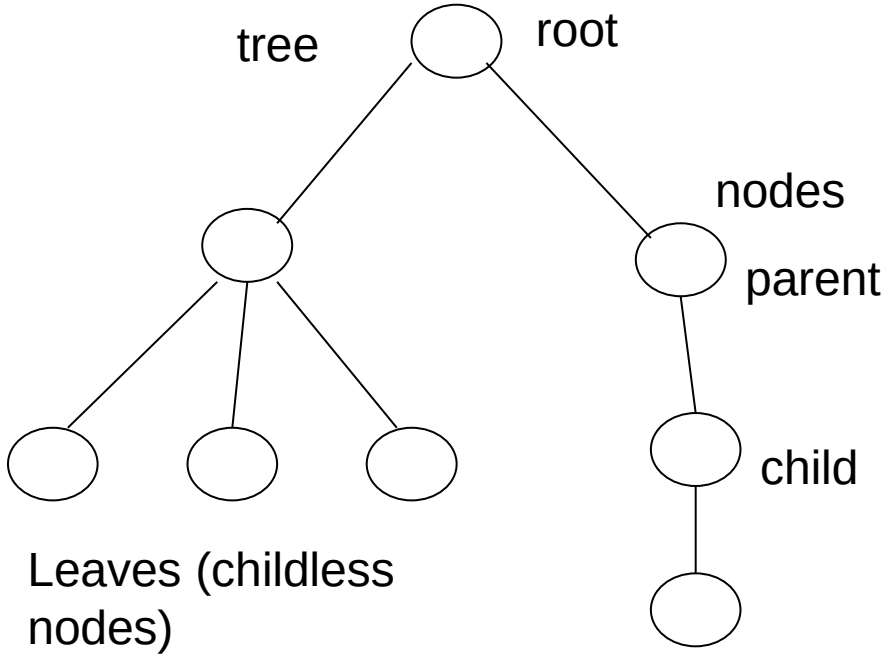
Nội dung

- Một số tính chất của cây nhị phân
- Cài đặt cây nhị phân
  - Dùng cấu trúc lưu trữ móc nối
  - Dùng cấu trúc lưu trữ tuần tự
- Duyệt cây nhị phân
  - Phép thăm và duyệt
  - Các phép duyệt cây
    - Duyệt theo thứ tự trước
    - Duyệt theo thứ tự giữa
    - Duyệt theo thứ tự sau
    - Duyệt theo từng mức (tầng)
  - Một số ứng dụng của cấu trúc cây nhị phân

Các khái niệm cơ bản

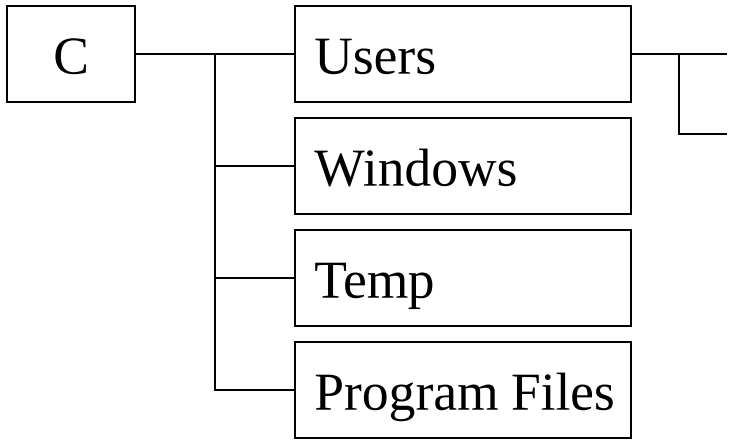
- Mô tả cấu trúc cây (tree)
  - Các phần tử hay các nút
    - nút (node)
    - gốc (root)
  - Các quan hệ phân cấp giữa các cặp nút.
    - nhánh
    - quan hệ cha-con
    - quan hệ cấp trên-cấp dưới
  - Cây rỗng

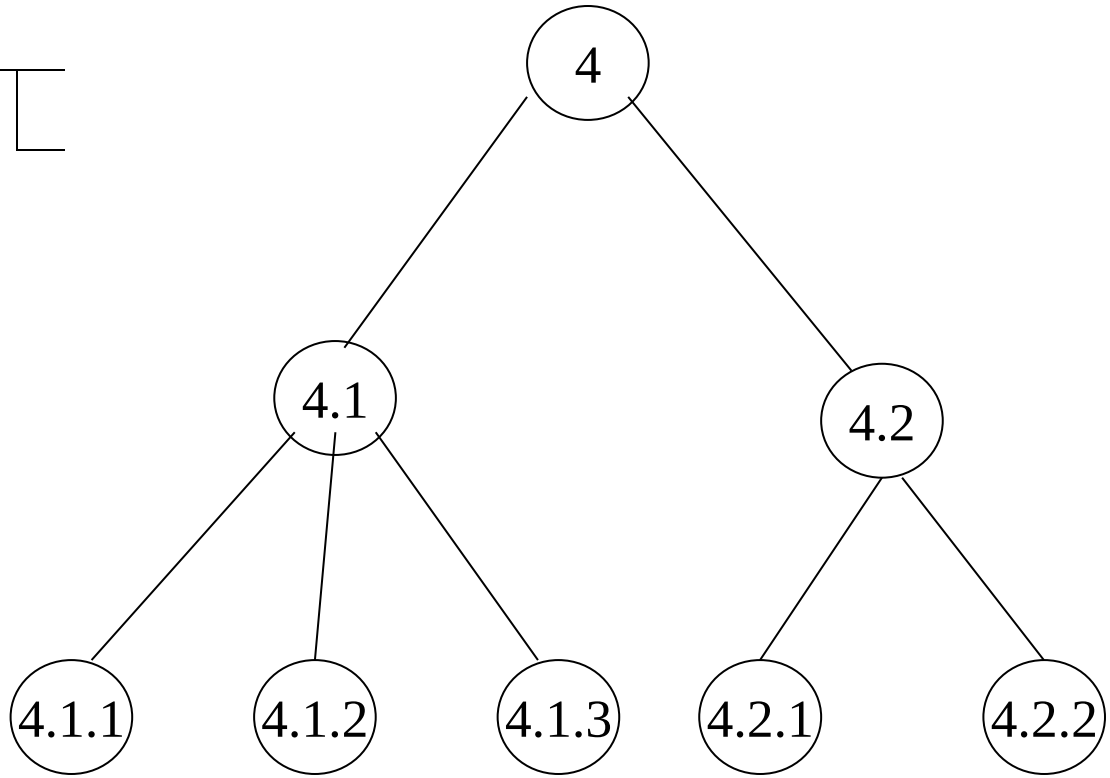




Các khái niệm cơ bản

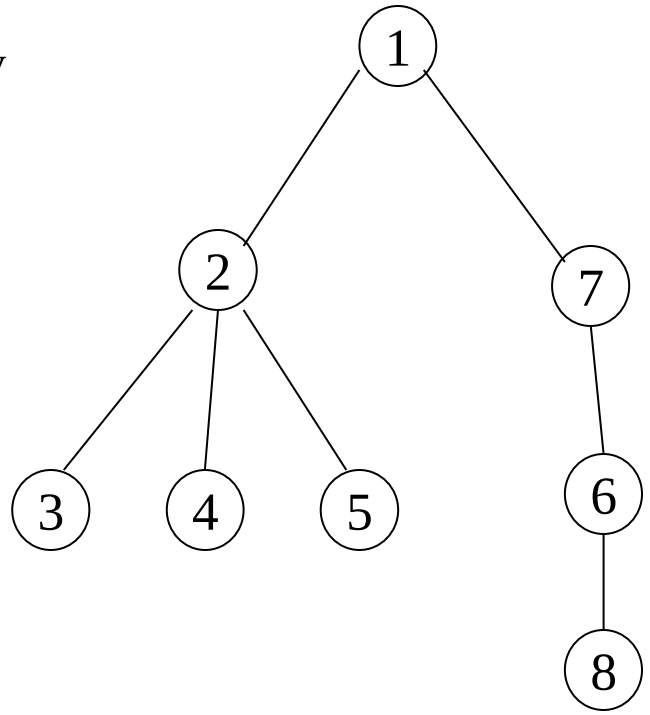
- Giới thiệu chung – Ví dụ
  - Ví dụ về các tiêu đề
  - Ví dụ thư mục trong windows





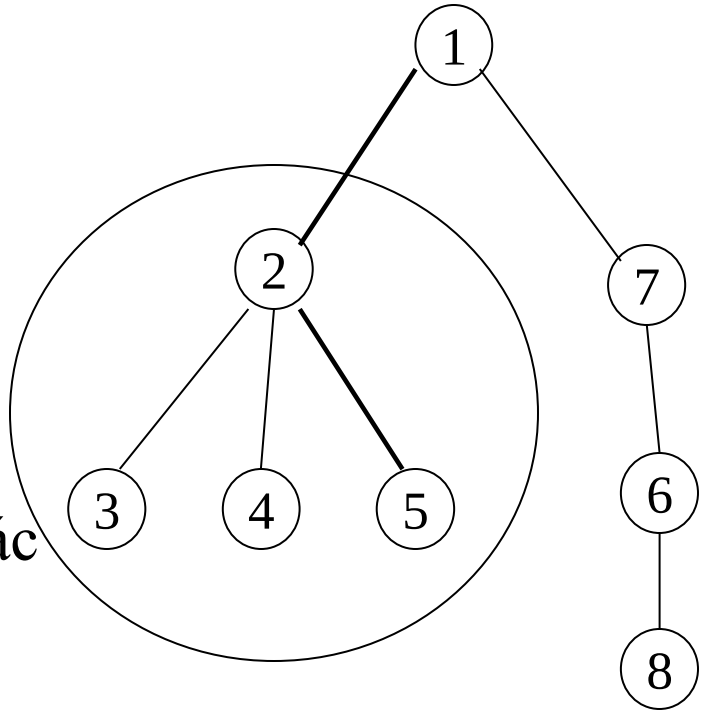
Các khái niệm cơ bản

- Giới thiệu chung – Khái niệm
  - Kích thước cây (size): là số nút có trên cây S(T)
  - Cấp của một nút (degree): là số con mà nút này có d(A)
  - Cấp của cây: d(T)
    - là cấp của nút có số con nhiều nhất trên cây
  - Nút nhánh (node-branch node)
    - là nút có ít nhất một con
    - tên gọi khác: nút trong, nút không tận cùng
  - Nút lá (leaf node or terminal node)
    - là nút không có con nào
    - tên gọi khác: nút ngoài, nút tận cùng
  - Mức của một nút (level): L(A)
    - Nếu A là nút gốc thì L(A) = 1
    - Nếu B là cha của A thì L(A) = L(B) +1



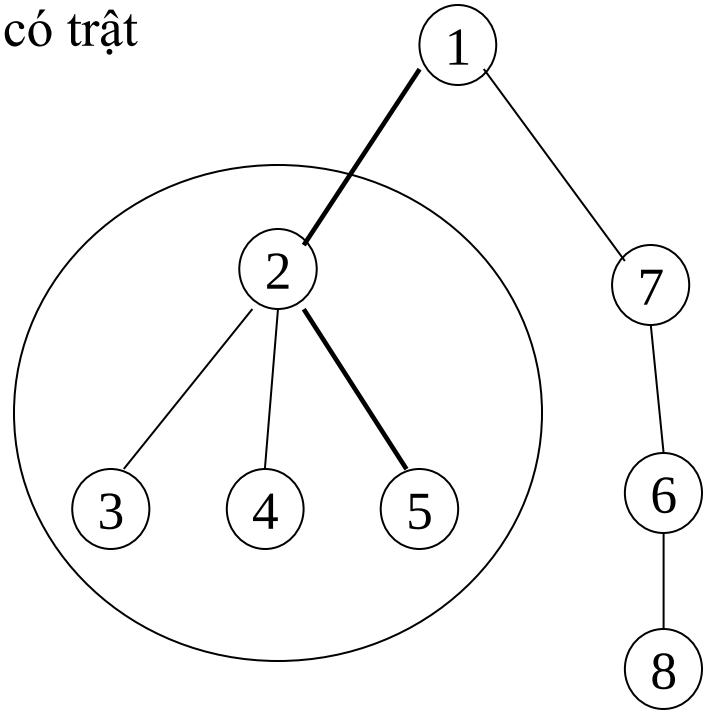
Các khái niệm cơ bản

- Giới thiệu chung – Khái niệm
  - Chiều cao của một nút (height) – Độ sâu (depth)
    - h(A) = L(A)-1
  - Chiều cao của cây: h(T)
    - là chiều cao của nút cao nhất trong cây
  - Đường đi giữa nút gốc và một nút khác (path): p(A)
    - là dãy các nhánh nối từ gốc đến nút đó
  - Độ dài đường đi (path length): PL(A)
    - là tổng độ dài các nhánh tạo thành đường đi
    - Thực tế: nhánh thường là khái niệm logic biểu diễn quan hệ phân cấp giữa một cặp nút, nên không có độ dài. Vì vậy, thường quy ước độ dài của tất cả các nhánh trong một cây đều bằng 1 (đơn vị). Khi đó, độ dài đường đi tương đương với chiều cao



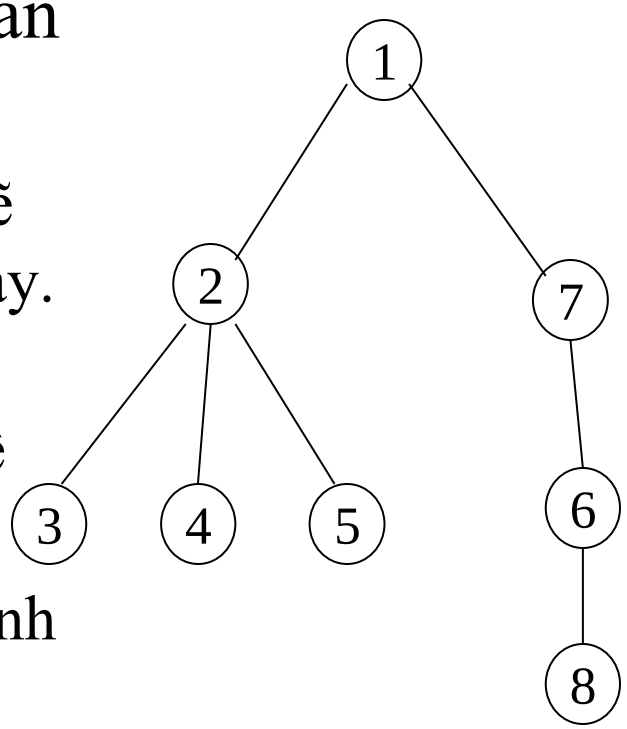
Các khái niệm cơ bản

- Giới thiệu chung – Khái niệm
  - Cây con
  - Cây được sắp (có thứ tự - ordered tree)
    - Có thứ tự (ordered tree): có trật tự tuyến tính giữa các nút con
    - Không có thứ tự (unordered tree): không có trật tự tuyến tính
  - Rừng (forest)
    - là cấu trúc bao gồm nhiều cây
    - có thể có rừng rỗng



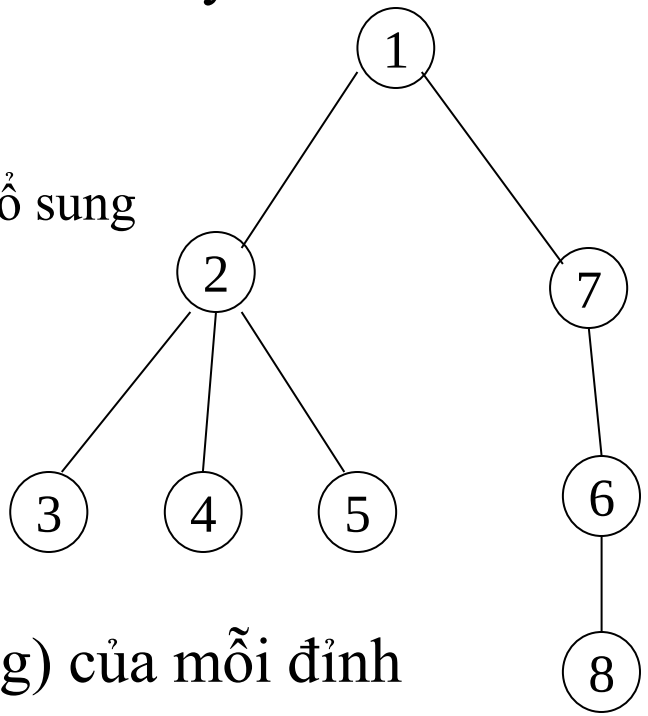
Các khái niệm cơ bản

- Giới thiệu chung – Các tính chất cơ bản
  - Số nút của cây bằng số nhánh cộng 1
  - Cây có tính chất đệ quy. Tính chất này sẽ được sử dụng trong nhiều thao tác sau này.
  - Cây là một cấu trúc dữ liệu động, tức là kích thước của nó (số nút của cây) có thể thay đổi.
  - Cấu trúc cây không còn cấu trúc tuyến tính nữa mà là cấu trúc phân cấp.
  - Chỉ tồn tại duy nhất một đường đi từ gốc đến một nút khác.



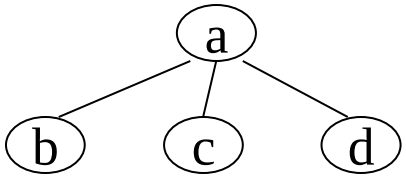
Các khái niệm cơ bản

- Giới thiệu chung – Các thao tác cơ bản trên cây
  - Khởi tạo: chuẩn bị CTLT để lưu trữ cấu trúc cây
  - Bổ sung một nút mới vào cây
    - Xác định vị trí
    - Xác định quan hệ nút mới và nút tại vị trí bổ sung
  - Lấy ra một nút
    - Xác định vị trí
    - Cấu trúc lại cây
  - Tìm cha mỗi đỉnh
    - Parent(x)
  - Tìm con bên trái ngoài cùng (con trưởng) của mỗi đỉnh
    - EldestChild(x)
  - Tìm em liền kề mỗi đỉnh
    - NextSibling(x)

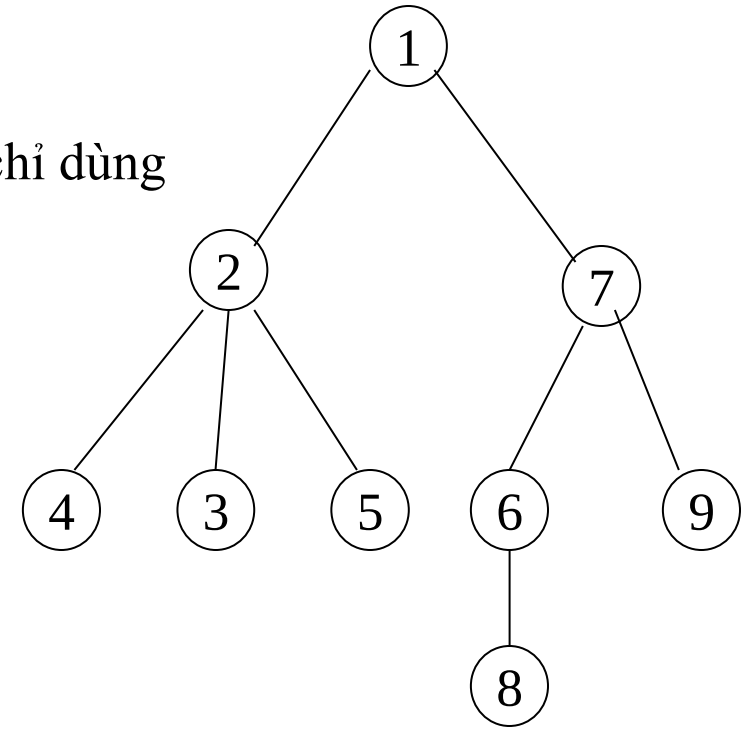


Các khái niệm cơ bản

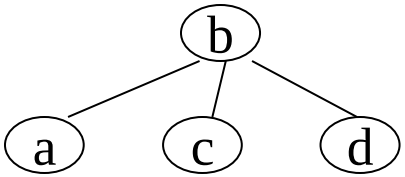
- Giới thiệu chung – Các thao tác cơ bản trên cây
  - Duyệt cây
    - Preorder: 1, 2, 4, 3, 5, 7, 6, 8, 9



=> a, b, c, d

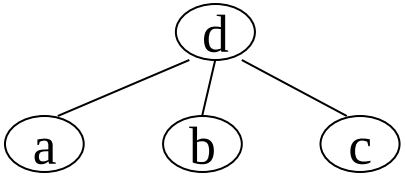


    - Inorder: 4, 2, 3, 5, 1, 8, 6, 7, 9 : Thường chỉ dùng với cây NP



=> a, b, c, d

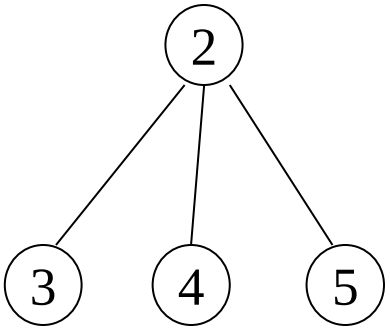
    - Postorder: 4, 3, 5, 2, 8, 6, 9, 7, 1



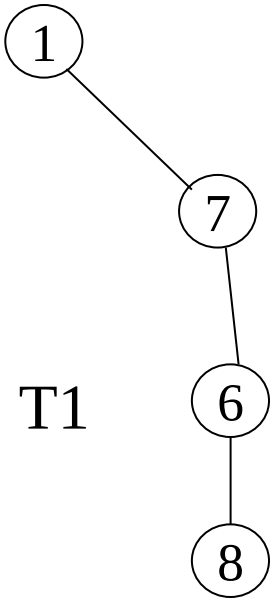
=> a, b, c, d

Các khái niệm cơ bản

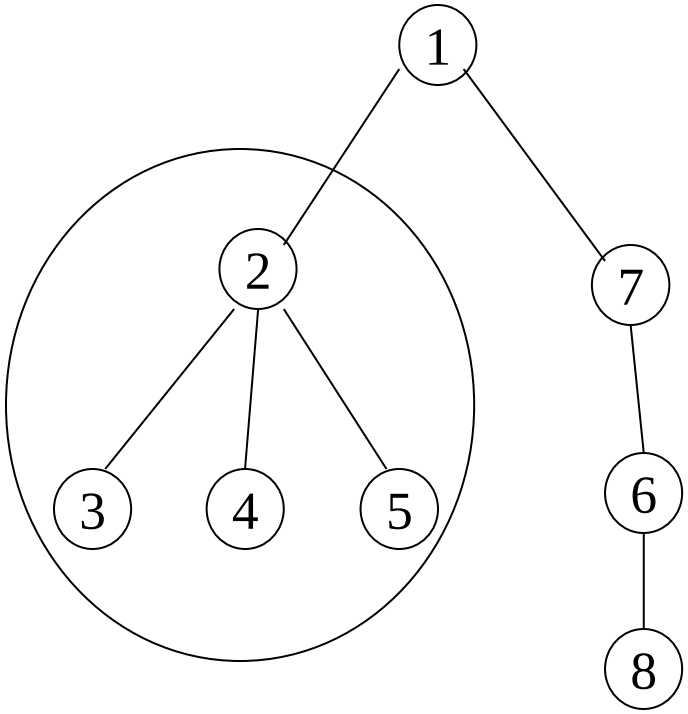
- Giới thiệu chung – Các thao tác cơ bản trên cây
  - Ghép cây
    - MergeTree(T1,T2,x): x=1
    - Vai trò của x: nút ghép, gốc của T2 sẽ là con của x, và ta có thể ghép trái, ghép phải
  - Cắt cây
    - CutTree(T1,x,T2): x=2



T2

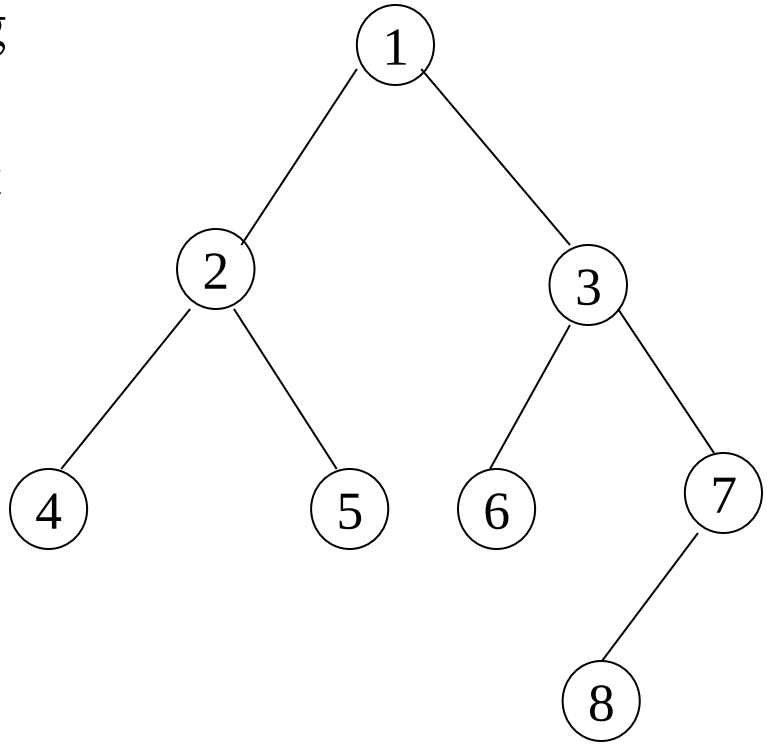


T1



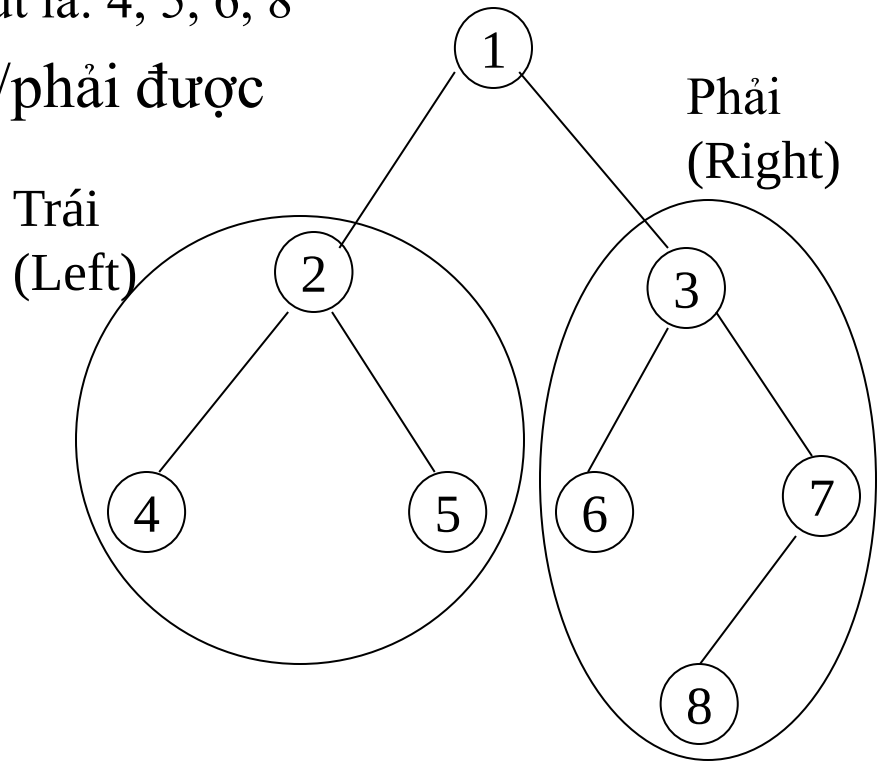
Cây nhị phân (binary tree)

- Mô tả cây nhị phân và các khái niệm cơ bản
  - Cây có thứ tự và là cây cấp hai, tức là mỗi nút có tối đa 2 con.
  - Hai con của một nút được phân biệt thứ tự và quy ước nút trước gọi là nút con trái và nút sau được gọi là nút con phải
  - Khi biểu diễn cây nhị phân, ta cũng cần có sự phân biệt rõ ràng giữa con trái và con phải, nhất là khi nút chỉ có một con.



Cây nhị phân (binary tree)

- Mô tả cây nhị phân và các khái niệm cơ bản
  - Loại nút:
    - Nút có đủ hai con được gọi là nút kép: 2, 3
    - Nút chỉ có một con gọi là nút đơn: 7
    - Nút không có con được gọi là nút lá: 4, 5, 6, 8
  - Cây con có gốc là nút con trái/phải được gọi là cây con trái/phải.

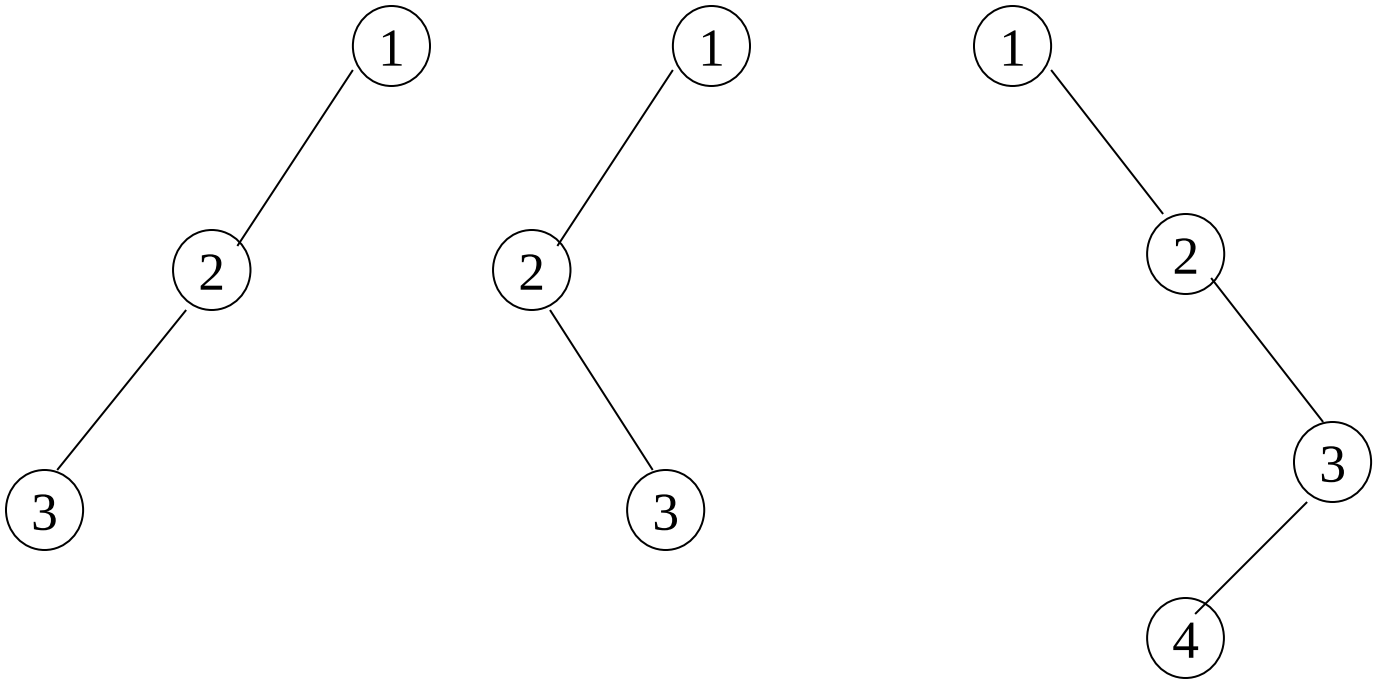


Cây nhị phân (binary tree)

- Các thao tác cơ bản
  - Cây nhị phân cũng có các thao tác cơ bản tương tự như của cây tổng quát.:
    - Khởi tạo: chuẩn bị CTLT để lưu trữ cấu trúc cây
    - Bổ sung một nút mới vào cây
    - Lấy ra một nút
    - Tìm cha mỗi đỉnh: Parent(x)
    - Tìm con bên trái của mỗi đỉnh: LeftChild(x)
    - Tìm con bên phải của mỗi đỉnh: RightChild(x)
    - Ghép cây
    - Cắt cây
    - Duyệt cây

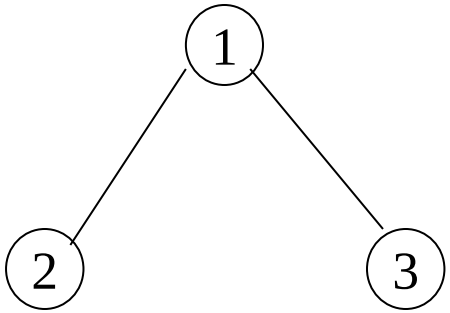
Cây NP - Các trường hợp đặc biệt

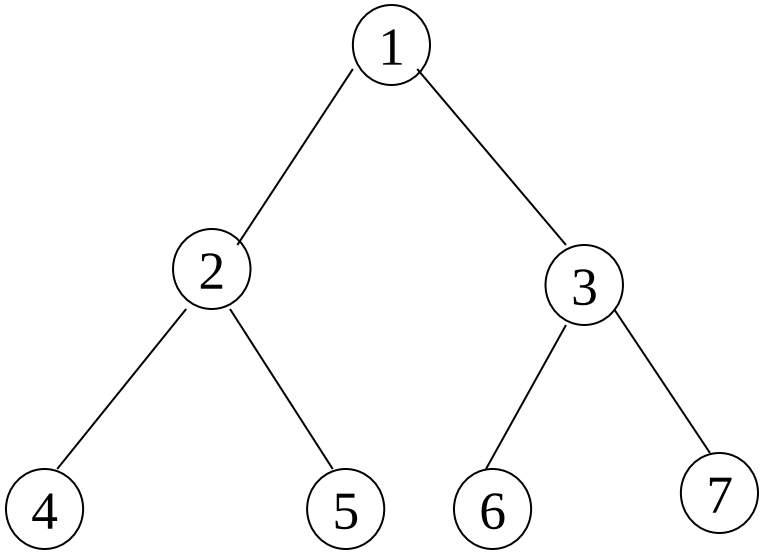
- Cây suy biến (degenerate binary tree)
  - Là cây nhị phân mà mỗi nút chỉ có tối đa một con. Nó đã bị suy biến về cấu trúc danh sách.
  - Như vậy, danh sách chỉ là dạng suy biến, trường hợp đặc biệt của cấu trúc cây, và nó vẫn giữa được tính chất đệ quy của cây



Cây NP - Các trường hợp đặc biệt

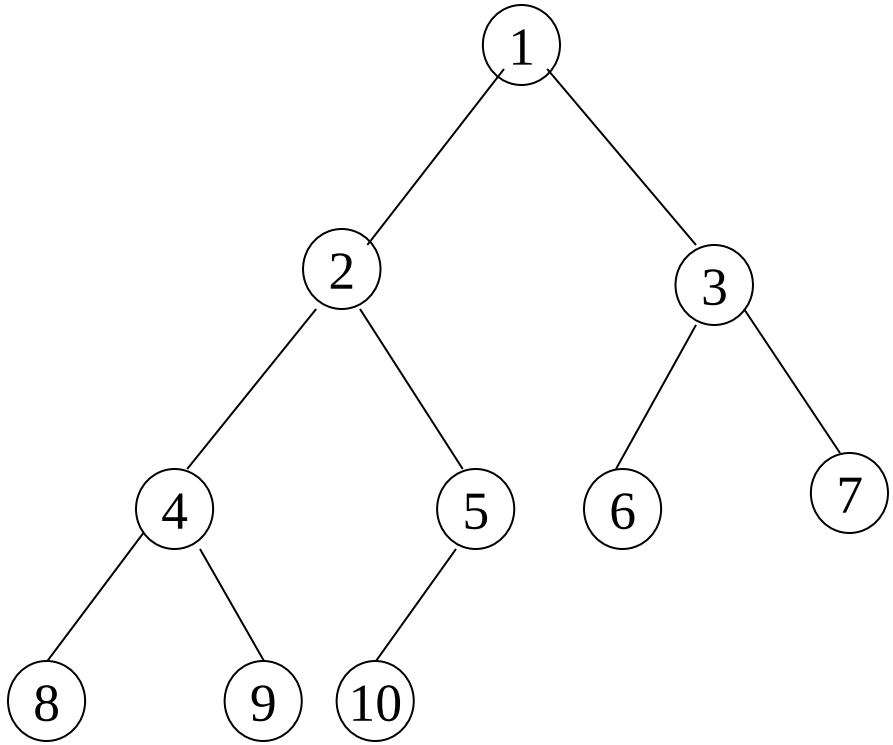
- Cây đầy đủ (full binary tree)
  - Là cây nhị phân có số nút tối đa ở tất cả các mức. Tức là ở mức i sẽ có đúng 2^{i-1} nút. Do đó, nếu cây nhị phân có chiều cao h thì kích thước của nó là: $S(T) = 2^0+2^1+\ldots+2^h = 2^{h+1}-1$;





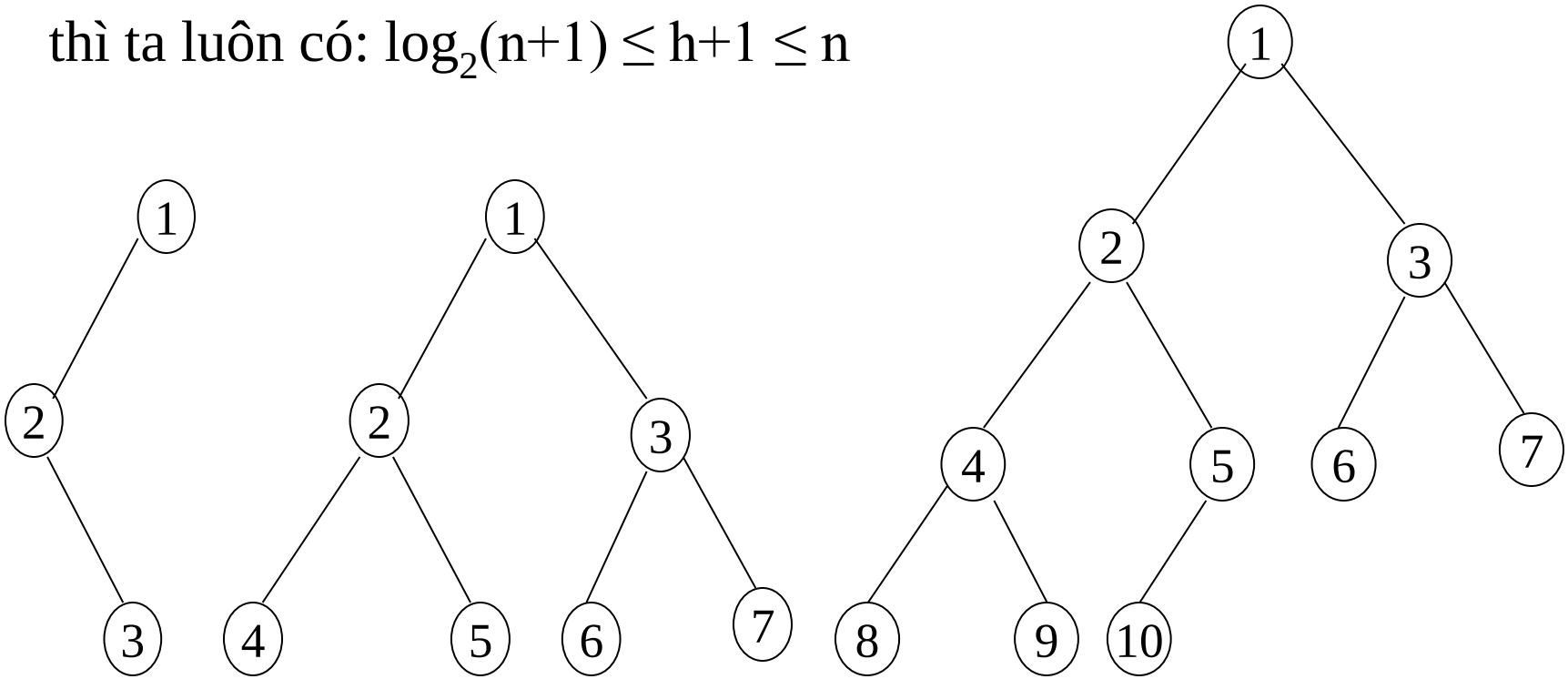
Cây NP - Các trường hợp đặc biệt

- Cây hoàn chỉnh (complete binary tree)
  - Gọi h là chiều cao của cây T. Ta gọi nó là cây hoàn chỉnh nếu:
    - T là cây đầy đủ đến chiều cao h-1
    - Các nút có chiều cao h thì dồn hết về bên trái.



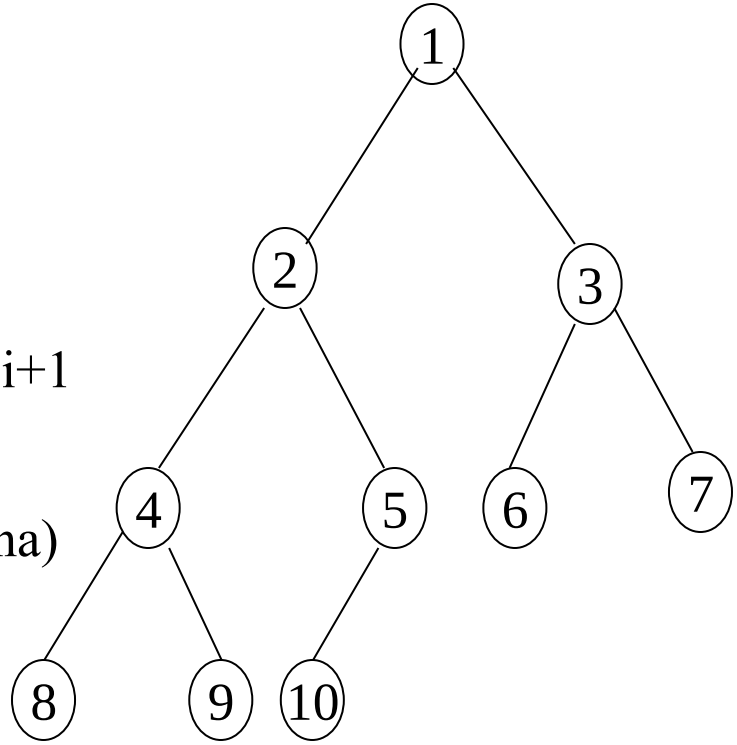
Một số tính chất của cây nhị phân

- Một số tính chất của cây nhị phân
  - Tính đệ quy: nếu ta chặt một nhánh bất kì của cây nhị phân thì ta sẽ thu được hai cây con cũng đều là cây nhị phân.
  - Gọi h và n lần lượt là chiều cao và kích thước của cây nhị phân thì ta luôn có: $\log_2(n+1) \leq h+1 \leq n$



Một số tính chất của cây nhị phân

- Một số tính chất của cây nhị phân
  - Đối với cây hoàn chỉnh và cây đầy đủ: nếu ta đánh số các nút lần lượt từ 1..N, với N là kích thước của cây theo thứ tự như hình vẽ thì ta có nhận xét sau:
  - Tại nút có số thứ tự i:
    - nếu 2i>N: nút i là nút lá
    - nếu 2i=N: nút i là nút đơn
    - nếu 2i<N: nút i là nút kép và có hai con trái, phải tương ứng là 2i và 2i+1
  - Tại nút có số thứ tự j:
    - nếu j=1: nút j là nút gốc (không có nút cha)
    - nếu j>1: nút j/2 (nếu j chẵn) hoặc (j-1)/2 (nếu j lẻ) là nút cha của nút j.



Cây nhị phân (binary tree)

- Các thao tác cơ bản - Giải thuật duyệt cây
  - Phép duyệt cây
    - Phép thăm một nút (visit): là thao tác truy nhập vào một nút của cây để xử lý nút đó.
    - Phép duyệt (traversal): là phép thăm một cách hệ thống tất cả các nút của cây, mỗi nút đúng một lần.

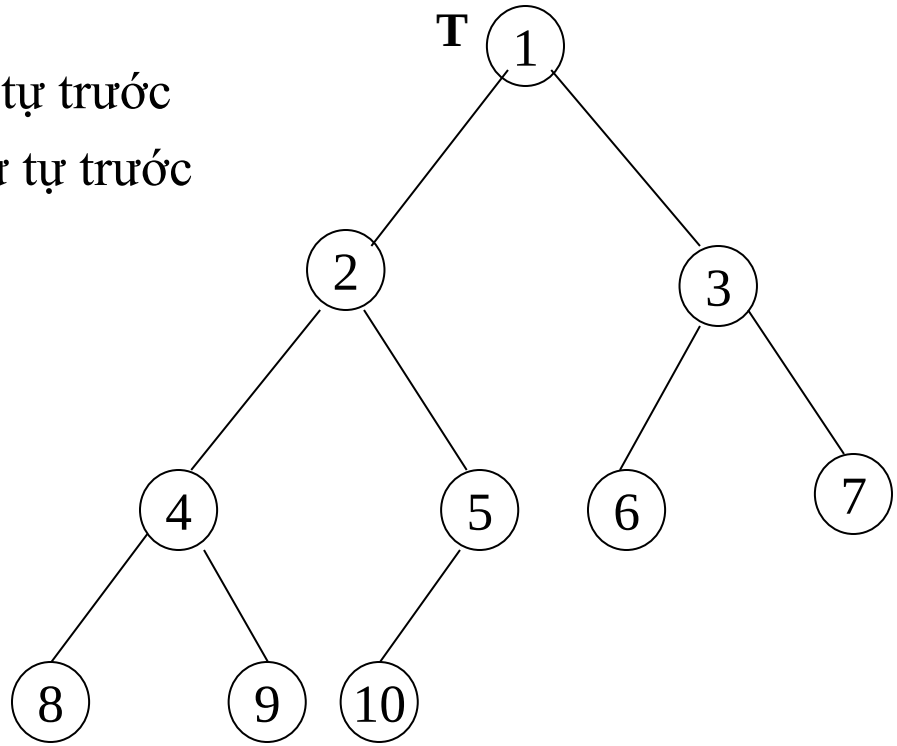
Cây nhị phân (binary tree)

- Các thao tác cơ bản - Giải thuật duyệt cây
  - Hướng thứ nhất: dựa vào tính chất đệ quy của cây để phát triển các giải thuật đệ quy để duyệt cây.
    - Cây nhị phân T bất kỳ được hình thành từ ba thành phần: gốc, cây con trái của T và cây con phải của T. Cả ba thành phần này đều là cây nhị phân. Theo hướng này, ta có 3 giải thuật duyệt cây như sau:
      - Duyệt theo thứ tự trước (preorder traversal)
      - Duyệt theo thứ tự giữa (inorder traversal)
      - Duyệt theo thứ tự sau (postorder traversal)
  - Hướng thứ hai:
    - Chuyển cấu trúc cây về cấu trúc danh sách. Sau đó đưa bài toán duyệt cây về bài toán duyệt danh sách đơn giản hơn nhiều.
    - Theo hướng này, chúng ta sẽ sử dụng một danh sách để lưu các nút của cây, sau đó sẽ duyệt danh sách.
    - Tuỳ theo loại danh sách (hàng đợi, ngăn xếp,…) và thứ tự duyệt trong danh sách mà chúng ta có những cách duyệt cây khác nhau.

Cây nhị phân (binary tree)

- Các thao tác cơ bản - Giải thuật duyệt cây
  - Giải thuật duyệt theo thứ tự trước (preorder traversal, còn gọi là duyệt cây theo chiều sâu): đây là giải thuật đệ quy
    - Trường hợp đệ quy: để duyệt cây nhị phân T, ta theo các bước sau:
      - Thăm gốc
      - Duyệt cây con trái theo thứ tự trước
      - Duyệt cây con phải theo thứ tự trước
    - Trường hợp điểm dừng:
      - Khi cây T rỗng.

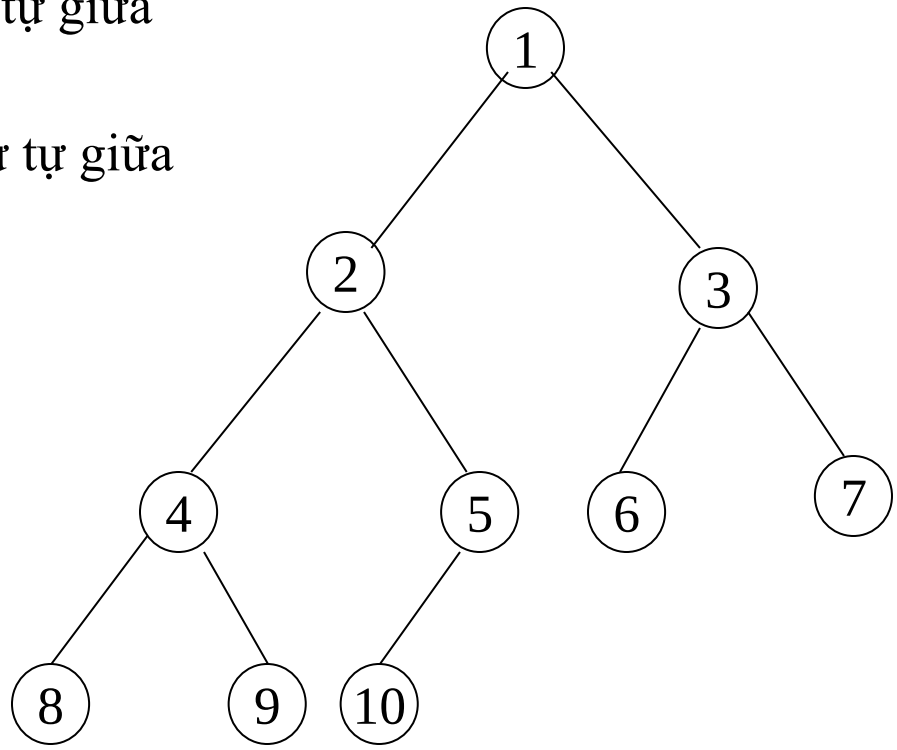
VD: 1, 2, 4, 8, 9, 5, 10, 3, 6, 7



Cây nhị phân (binary tree)

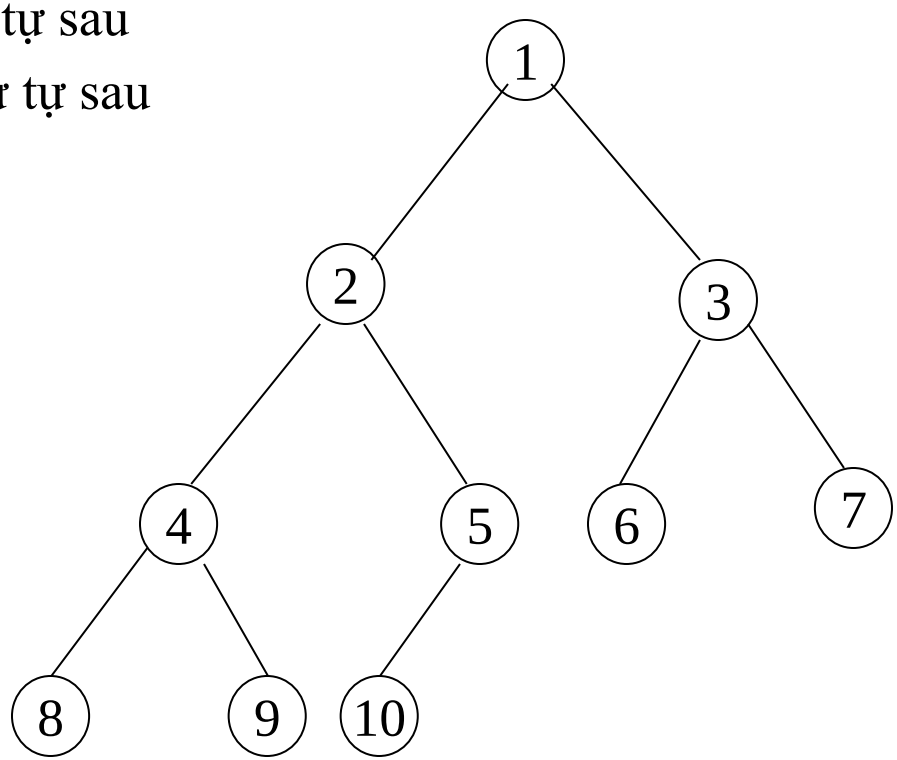
- Các thao tác cơ bản - Giải thuật duyệt cây
  - Giải thuật duyệt theo thứ tự giữa (inorder travelsal): đây là giải thuật đệ quy.
    - Trường hợp đệ quy: để duyệt cây nhị phân T, ta theo các bước sau:
      - Duyệt cây con trái theo thứ tự giữa
      - Thăm gốc
      - Duyệt cây con phải theo thứ tự giữa
    - Trường hợp điểm dừng:
      - Khi cây T rỗng.

VD: 8, 4, 9, 2, 10, 5, 1, 6, 3, 7



Cây nhị phân (binary tree)

- Các thao tác cơ bản - Giải thuật duyệt cây
  - Giải thuật duyệt theo thứ tự sau (postorder travelsal): đây là giải thuật đệ quy.
    - Trường hợp đệ quy: để duyệt cây nhị phân T, ta theo các bước sau:
      - Duyệt cây con trái theo thứ tự sau
      - Duyệt cây con phải theo thứ tự sau
      - Thăm gốc
    - Trường hợp điểm dừng:
      - Khi cây T rỗng.
    - VD: 8, 9, 4, 10, 5, 2, 6, 7, 3, 1



Cây nhị phân (binary tree)

- Các thao tác cơ bản - Giải thuật duyệt cây
  - Giải thuật duyệt cây theo từng mức (theo chiều rộng): giải thuật ***không*** đệ quy:
    - Đưa bài toán duyệt cây về duyệt danh sách.
    - Để thứ tự duyệt là theo từng mức (từng tầng) và sử dụng một cấu trúc danh sách kiểu hàng đợi Q.

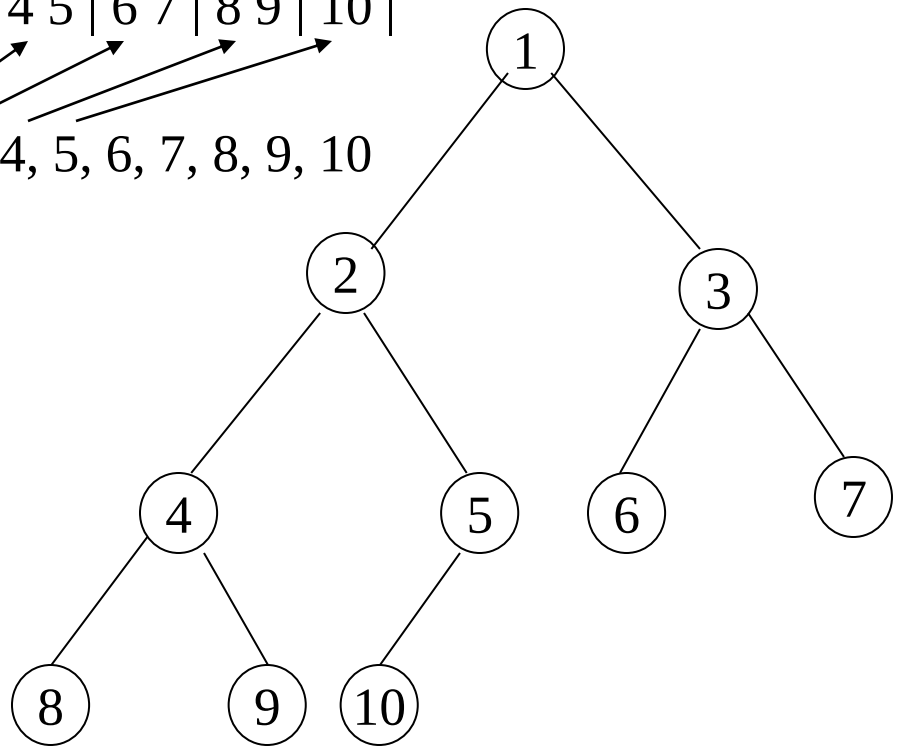
```
Initialize (Q)                          // Khởi tạo: Q=∅.
InsertQ (T, Q);                         //Bổ sung nút gốc vào Q
While ( Q <> ∅ ) do
        P = DeleteQ (Q);                //Lấy một nút ra khỏi Q để chuẩn bị thăm
        Visit (P);                      //Thăm P
        if <P có con trái PL> then InsertQ (PL, Q);
        if <P có con phải PR> then InsertQ (PR, Q);
End While ;
```

Cây nhị phân (binary tree)

- Các thao tác cơ bản - Giải thuật duyệt cây
  - Giải thuật duyệt cây theo từng mức (theo chiều rộng): giải thuật ***không*** đệ quy

    VD:

    - Thứ tự đưa vào hàng: 1 | 2 3 | 4 5 | 6 7 | 8 9 | 10 |
    - Thứ tự thăm: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10



Cài đặt cây nhị phân

- Có thể sử dụng hai loại CTLT để cài đặt cây
  - Sử dụng CTLT tuần tự:
    - Thường chỉ thích hợp khi cài đặt các cây NP đặc biệt.
    - Đối với cây thông thường, ta phải bổ sung thêm các nút để nó trở thành đặc biệt (thành cây đầy đủ hoặc hoàn chỉnh). Do đó, làm cho việc cài đặt phức tạp và tốn bộ nhớ, nên ít khi được sử dụng.
    - Thường chỉ áp dụng khi hệ thống không cung cấp CTLT móc nối.
  - Sử dụng CTLT móc nối:
    - Đây là cách thường được dùng do tính linh hoạt của CTLT móc nối, rất thích hợp để cài đặt các CTDL phi tuyến.
    - Do đó, trong phần tiếp theo ta chỉ xét phương pháp cài đặt này.

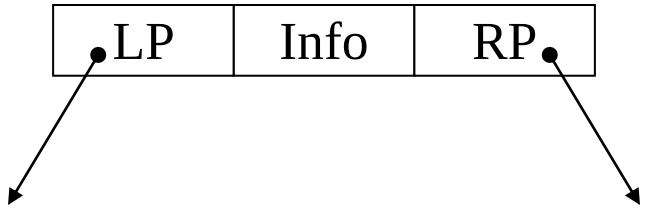
Cài đặt cây nhị phân

- Nguyên tắc cài đặt: Sử dụng cấu trúc lưu trữ móc nối
  - Đối với cây nhị phân, ta cần lưu trữ hai thành phần:
  - Các phần tử (nút) của cấu trúc cây:
    - Dùng các nút của CTLT để lưu trữ các phần tử.
  - Các nhánh (quan hệ cha-con) giữa các cặp nút:
    - Sử dụng con trỏ để biểu diễn các nhánh.
    - Do mỗi nút có nhiều nhất hai con nên trong mỗi nút của CTLT ta sẽ có hai con trỏ để trỏ đến tối đa hai con này.

Cài đặt cây nhị phân

- Dùng cấu trúc móc nối
  - Cấu tạo mỗi nút (xem hình):
    - Trường Info: chứa thông tin về một phần tử.
    - Trường LP và RP tương ứng là hai con trỏ sẽ trỏ vào hai nút con trái và con phải.
    - Nếu không có con thì giá trị con trỏ tương ứng sẽ rỗng (NIL/NULL)
    - Khai báo cấu trúc một nút:

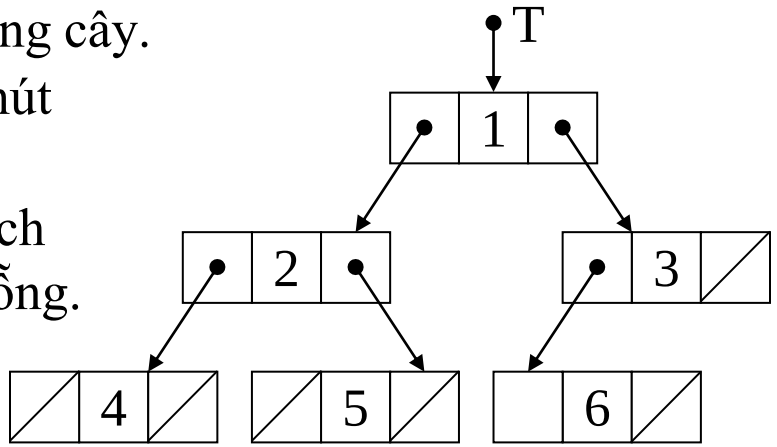
```
struct Node {
    type Info;
    Node * LP, *RP;
};

typedef Node* PNode;
```



Cài đặt cây nhị phân

- Dùng cấu trúc móc nối
  - Tổ chức cấu trúc cây như sau (xem hình):
    - Ít nhất một con trỏ T trỏ vào nút gốc, vừa đại diện cho cây, vừa là điểm truy nhập vào các nút khác trong cây.
    - Các thao tác muốn truy nhập vào các nút trong cây phải thông qua con trỏ này.
    - Nhận xét: với cách tổ chức này, khi kích thước cây là N thì sẽ có N+1 con trỏ rỗng.
    - Khai báo cấu trúc cây: có 2 cách,

sau này để đơn giản ta sẽ dùng cách 1

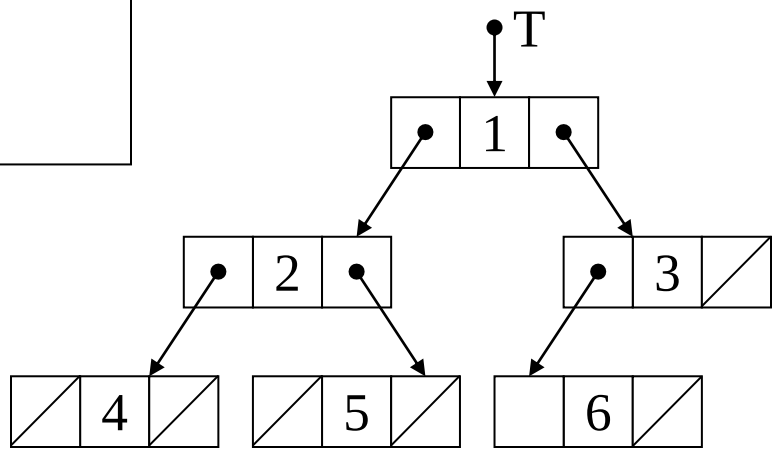



```
Cách 1:     typedef  PNode  BinaryTree;
Cách 2:     struct  BinaryTree {
                        PNode  T;  //con trỏ trỏ vào nút gốc
                        int  n;        // kích thước cây
                    }
```

Cài đặt cây nhị phân

- Dùng cấu trúc móc nối - Cài đặt một số thao tác cơ bản
  - Thao tác khởi tạo: tạo ra một cây rỗng.

```
void InitBT (BinaryTree & T) {
    T = NULL;
}
```

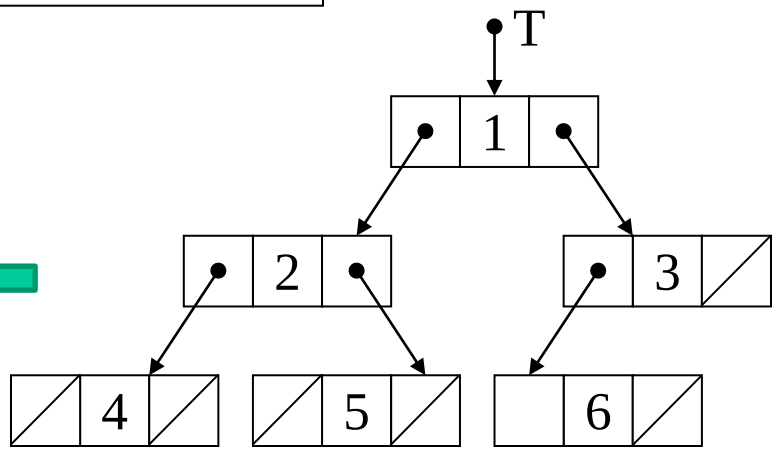


Duyệt cây nhị phân

- Dùng cấu trúc móc nối - Cài đặt một số thao tác cơ bản
  - Duyệt cây theo thứ tự trước (preorder traversal)

```
{Duyệt cây theo thứ tự trước}
void  PreOrderTraversal (BinaryTree  T) {
      if  (T==NULL) return;
      Visit(T);
      PreOrderTraversal(T->LP);
      PreOrderTraversal(T->RP);
}
```

1, 2, 4, 5, 3, 6

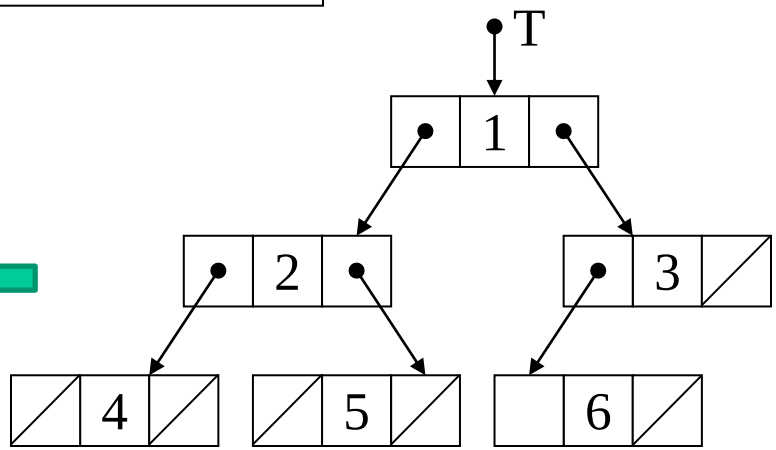


Duyệt cây nhị phân

- Dùng cấu trúc móc nối - Cài đặt một số thao tác cơ bản
  - Duyệt cây theo thứ tự giữa (inorder traversal)

```
{Duyệt cây theo thứ tự giữa}
void  InOrderTraversal (BinaryTree  T) {
      if  (T==NULL) return;
      InOrderTraversal(T->LP);
      Visit(T);
      InOrderTraversal(T->RP);
}
```

4, 2, 5, 1, 6, 3

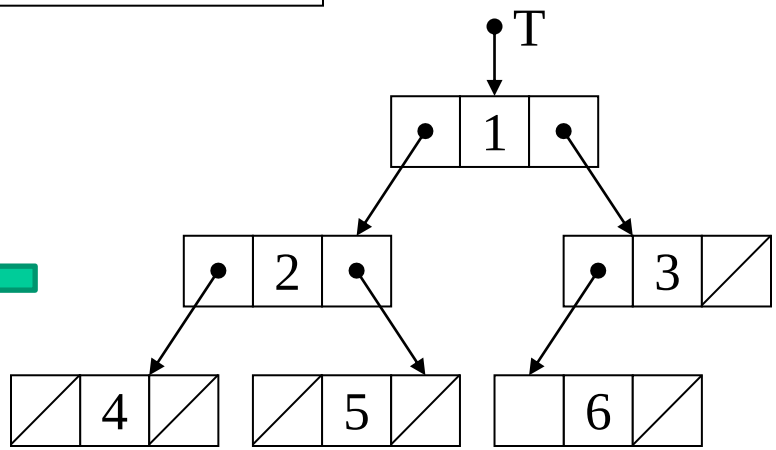


Duyệt cây nhị phân

- Dùng cấu trúc móc nối - Cài đặt một số thao tác cơ bản
  - Duyệt cây theo thứ tự sau (postorder travelsal)

```
{Duyệt cây theo thứ tự sau}
void   PostOrderTraversal (BinaryTree  T) {
        if  (T==NULL) return;
        PostOrderTraversal(T->LP);
        PostOrderTraversal(T->RP);
        Visit(T);
}
```

4, 5, 2, 6, 3, 1



Cây nhị phân tìm kiếm

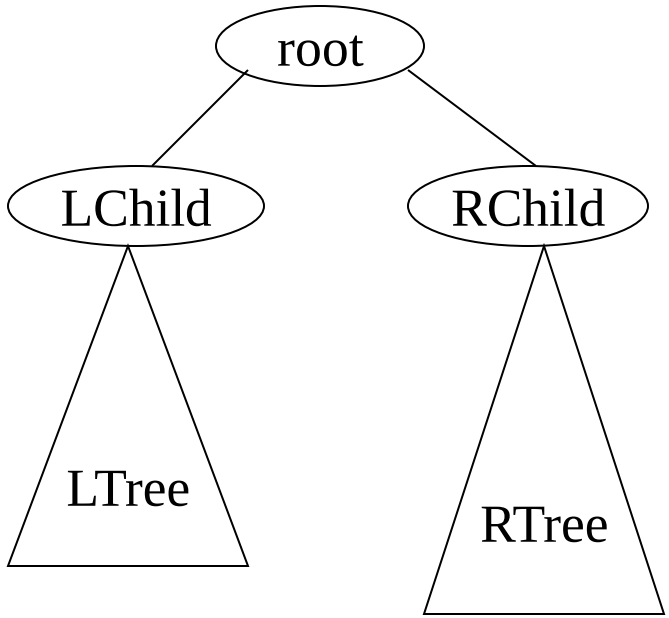
- Giới thiệu
  - Mục đích: ứng dụng vào lưu trữ và tìm kiếm thông tin một các hiệu quả nhất
  - Nguyên tắc: lưu trữ và tìm kiếm thông qua **"khoá"**
    - Nhóm thuộc tính khoá: bao gồm các thuộc tính, tính chất giúp chúng ta hoàn toàn xác định được đối tượng

Cây nhị phân tìm kiếm

- Là cây nhị phân thỏa mãn các điều kiện sau
  - Khoá của các đỉnh thuộc cây con trái nhỏ hơn khoá của gốc
  - Khóa của gốc nhỏ hơn khoá của các đỉnh thuộc cây con phải
  - Cây con trái và cây con phải cũng là cây nhị phân tìm kiếm

BSearchTree = BinaryTree; and
Key(LChild) < **Key**(root)**;** and
Key(root) < **Key**(RChild)**;**

AND
LTree = **BSearchTree;** and
RTree = **BSearchTree;**



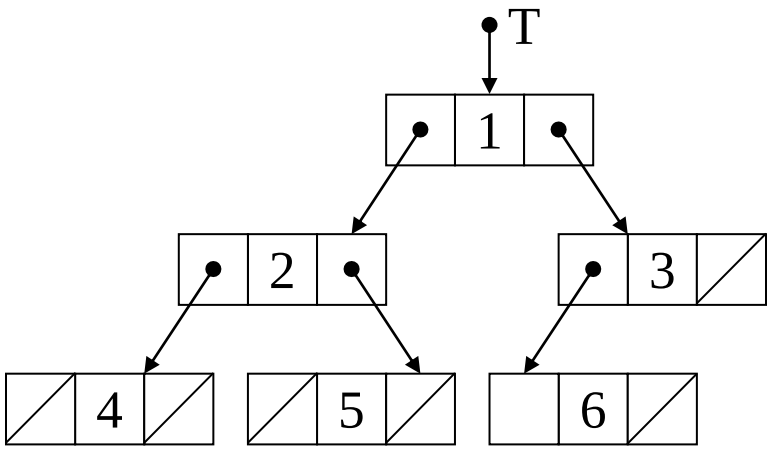
Cây nhị phân tìm kiếm

- Khai báo cấu trúc nút

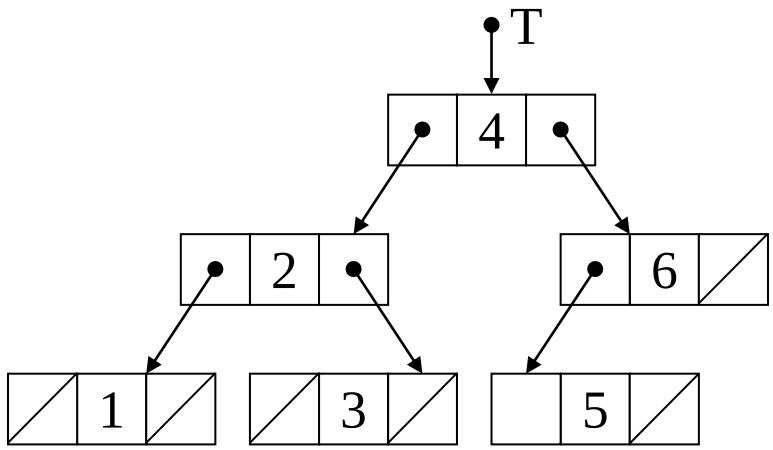
```
struct Node {
    keytype Key;      //Thường là kiểu có thứ tự, có thể so sánh được
    Node * LP, *RP;
};

typedef Node* PNode;
typedef PNode BinaryTree;
typedef BinaryTree BSearchTree;
```

Cây nhị phân tìm kiếm

- Khai báo cấu trúc cây
  - Ví dụ:



Cây nhị phân



Cây nhị phân tìm kiếm

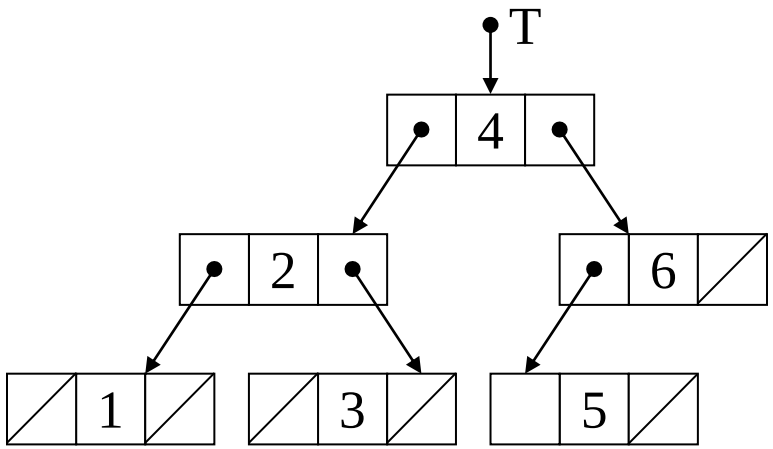
Cây nhị phân tìm kiếm

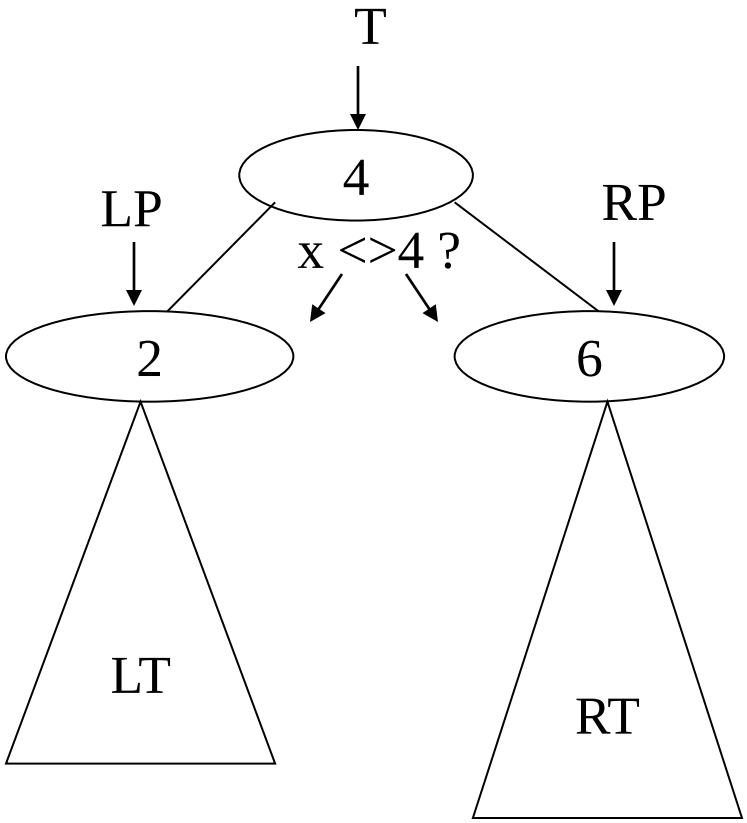
- Cài đặt các thao tác cơ bản: Tìm một nút có giá trị x cho trước. Hàm sẽ trả về con trỏ trỏ vào nút tìm được nếu có, trái lại trả về con trỏ NULL

```
PNode SearchT (BSearchTree Root, keytype  x){
      if  (Root==NULL)  return  NULL;
      if  (x == Root->Key) return  Root;
       else   if  (x < Root->Key)  return  SearchT (Root->LP, x);
      else  return  SearchT (Root->RP, x);
}
```

Cây nhị phân tìm kiếm

- Cài đặt các thao tác cơ bản: Bổ sung một nút
  - Minh họa:





Cây nhị phân tìm kiếm

- Cài đặt các thao tác cơ bản: Bổ sung một nút

```
void InsertT (BSearchTree & Root, keytype x){
    PNode Q;
    if (Root==NULL) {                    //khi cây rỗng
            Q = new Node;                //tạo ra đỉnh mới
            Q->Key = x;
            Q->LP =Q->RP = NULL;
            Root = Q;
    }
    else {
            if (x < Root->Key) InsertT (Root->LP, x);
            else if (x > Root->Key) InsertT (Root->RP, x);
    }
}
```

Cây nhị phân tìm kiếm

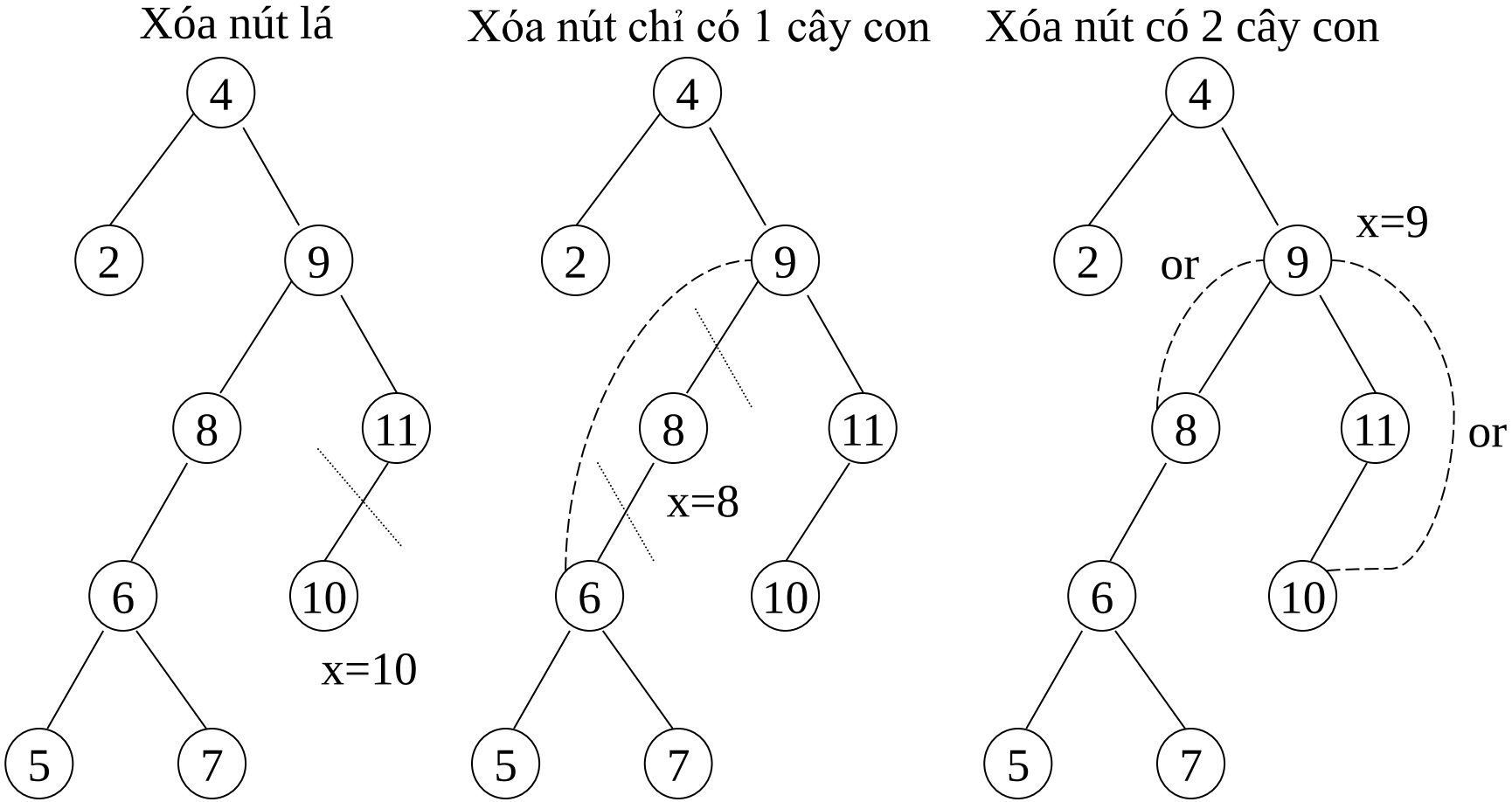
- Cài đặt các thao tác cơ bản: Lấy ra một nút

  Ý tưởng

  – Tìm đến nút có giá trị khóa cần loại bỏ
  – Thực hiện phép loại bỏ nút đó
  – Thực hiện xắp xếp lại để đảm bảo tính chất cây tìm kiếm NP
    - Nếu đỉnh cần loại bỏ là lá => không cần làm gì
    - Nếu đỉnh cần loại bỏ chỉ có 1 cây con => thực hiện phép nối
    - Nếu đỉnh cần loại bỏ có hai cây con LTree, RTree => cần thay khoá của đỉnh hai cây con này bằng giá trị Max(LTree) hoặc Min(RTree)
  – Lưu ý:
    - Giá trị Max(LTree) cần tìm ở phần tử bên phải ngoài cùng của cây con Ltree: *luôn đi theo bên phải của cây đến khi không được nữa Q->RP=NULL thì đó là nút ngoài cùng bên phải*
    - Giá trị Min(RTree) cần tìm ở phần tử bên trái ngoài cùng của cây con Rtree: *luôn đi theo bên trái của cây đến khi không được nữa Q->LP=NULL thì đó là nút ngoài cùng bên trái*

Cây nhị phân tìm kiếm

- Cài đặt các thao tác cơ bản: Lấy ra một nút
  - Minh họa:



Cây nhị phân tìm kiếm

- Cài đặt các thao tác cơ bản: Lấy ra một nút

```
void  DeleteT (BSearchTree & Root, keytype  x){
      if  (Root != NULL) {
                  if  (x < Root->Key)  DeleteT (Root->LP, x);
                  else  if  (x > Root->key)  DeleteT (Root->RP, x);
                  else  DelNode (Root);                  //Xoá gốc của cây
      }
}
```

```
void  DelNode (PNode & P) {  //Xóa giá trị ở nút P & sắp lại cây
      PNode Q, R;
      if (P->LP == NULL) {       //Xóa nút chỉ có cây con phải
                  Q = P;
                  P = P->RP;
      } else      if (P->RP = NULL)          //Xóa nút chỉ có cây con trái
      {
```

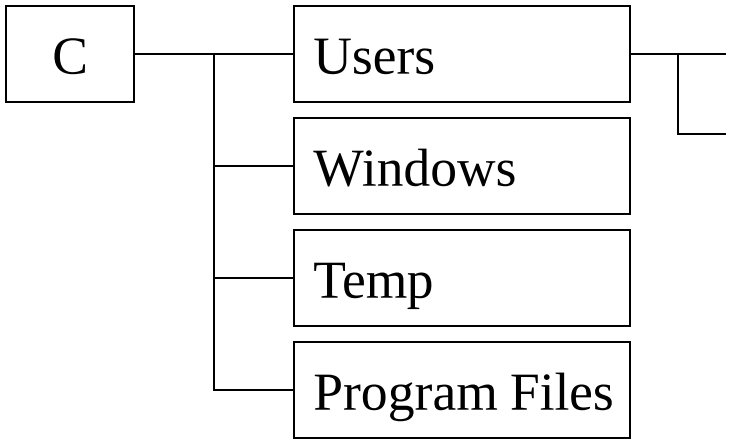
Cây nhị phân tìm kiếm

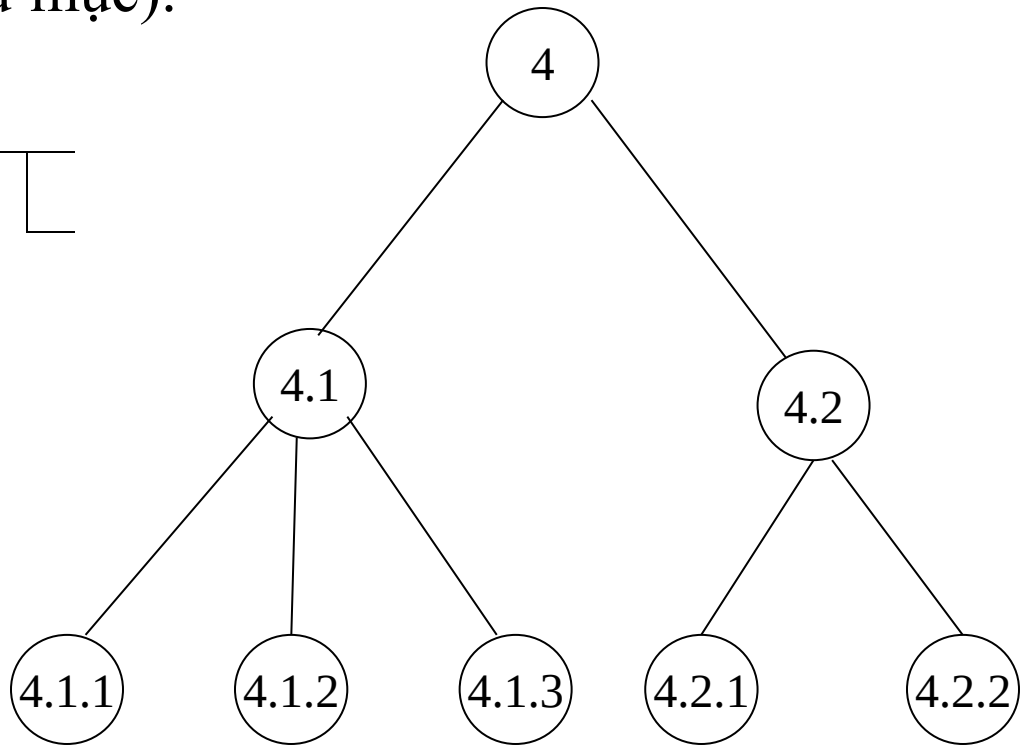
- Cài đặt các thao tác cơ bản: xoá một nút (tiếp...)

```
                                Q = P;
                                P = P->LP;
                    } else      {                          //Xóa nút có 2 cây con
                                Q = P->LP;
                                if  (Q->RP == NULL) {
                                    P->Key = Q->Key;
                                    P->LP = Q->LP;
                                } else {
                                    do {            //Dùng R để lưu parent của Q
                                            R = Q;
                                            Q = Q->RP;
                                    } while  (Q->RP != NULL);
                                    P->Key = Q->Key;       //Lấy giá trị ở Q đưa lên
                                    R->RP = Q->LP;         //Chuyển con của Q lên vị trí Q
                                }
[Tìm giá trị Max ở
cây con trái. Nó luôn
là giá trị ở nút ngoài
cùng bên phải cây
con => giải thuật:
Luôn đi theo bên
phải của Q, khi nào
không đi đươc nữa
Q->RP = NULL thì
đó là nút ngoài cùng
bên phải]
                    }
            delete  Q;                          //Xoá Q
        }
```

Cây tổng quát

- Giới thiệu
  - Cây tổng quát là cây mà mỗi nút có thể có nhiều hơn hai con. Nói chung, cây càng có nhiều con thì việc cài đặt càng phức tạp, nhất là những cây mà số con tối đa thường không cố định (VD: cây gia phả, cây thư mục).





Cây tổng quát

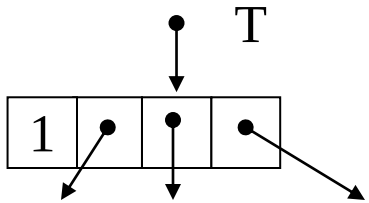
- Các phương pháp cài đặt
  - Cài đặt trực tiếp sử dụng CTLT móc nối
  - Cài đặt gián tiếp qua cây nhị phân

Cây tổng quát

- Cài đặt trực tiếp sử dụng CTLT móc nối

  Ý tưởng:

  - Tương tự khi cài đặt cây NP
  - Chỉ thích hợp khi ta biết trước cấp của cây (số con tối đa của một nút) và thường là số cấp không lớn (4-5).
  - Khi số cấp của cây lớn thì cách cài đặt này sẽ không hiệu quả do tốn khá nhiều bộ nhớ để lưu các con trỏ rỗng.
  - Nguyên tắc cài đặt cụ thể: nếu gọi cấp của cây là d, thì cấu tạo một nút gồm 2 phần:
    - Phần Info chứa nội dung thông tin của nút
    - Phần móc nối sẽ có d con trỏ p_1, p_2,…p_d để trỏ đến tối đa d con của nút.



Cây tổng quát

- Cài đặt trực tiếp sử dụng CTLT móc nối
  - Tổ chức cấu trúc cây
    - Vẫn có một con trỏ T trỏ vào nút gốc của cây.
    - N là kích thước của cây => số con trỏ rỗng trong cây T là: N(d-1) + 1.
    - Khi d càng lớn thì số con trỏ rỗng trong cây sẽ càng nhiều, lớn hơn số nút của cây nhiều lần => lãng phí bộ nhớ rất lớn.
    - Hơn nữa, cách cài đặt này đòi hỏi chúng ta phải biết trước số con lớn nhất có trong một nút của cây. Điều này không phải lúc nào cũng có được trong các cây trong thực tế.

Cây tổng quát

- Cài đặt gián tiếp qua cây nhị phân

  Ý tưởng:

  – Tận dụng các kết quả cài đặt từ cây NP, đồng thời khắc phục các hạn chế của cách cài đặt trực tiếp.

  – Chuyển cây tổng quát về cây NP.

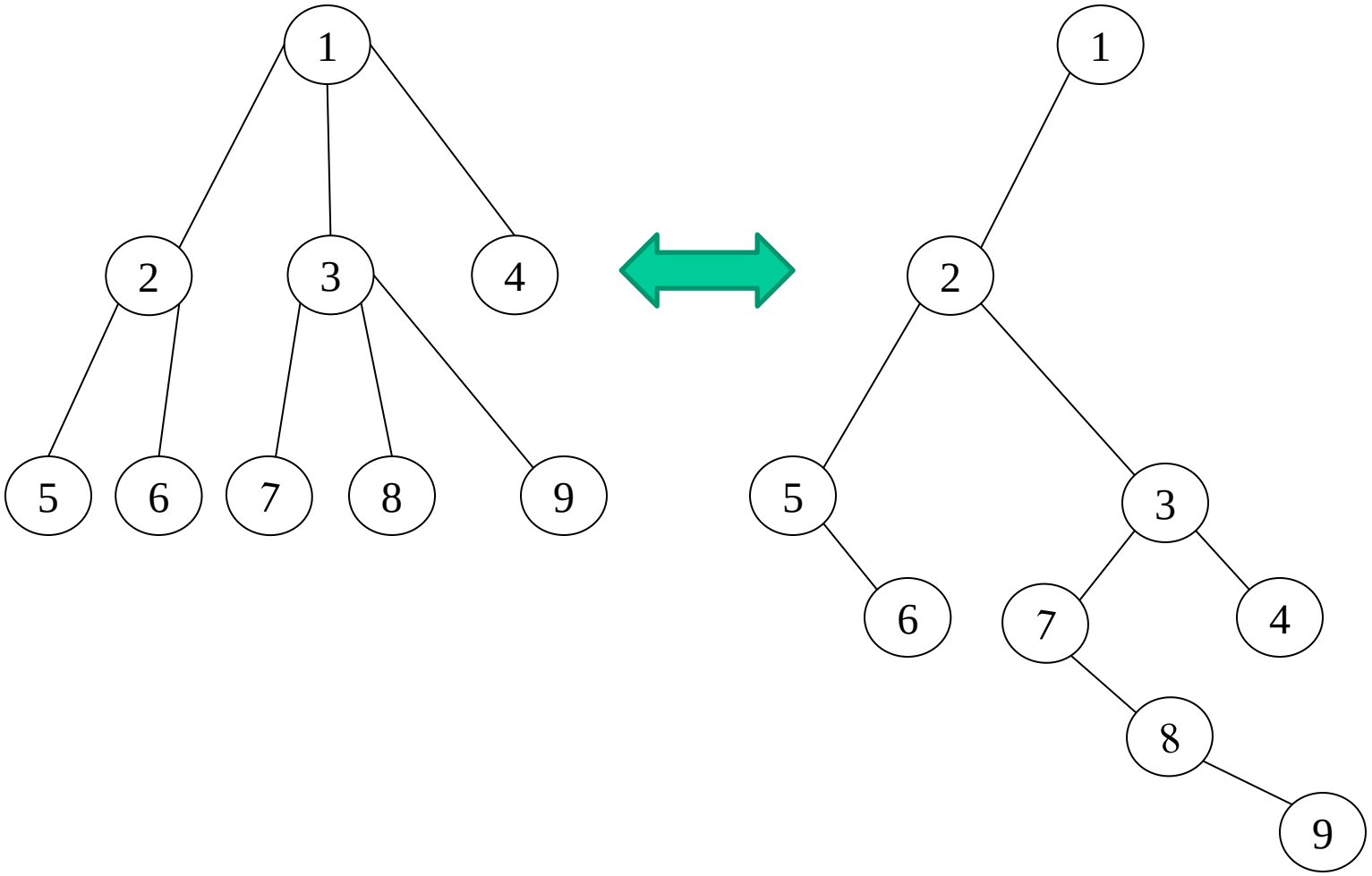
  – Cài đặt gián tiếp thông qua cài đặt cây NP. Muốn như vậy, ta phải xác định một quy tắc chuyển đổi hai chiều kiểu song ánh từ cây tổng quát sang cây NP và ngược lại.

  – Quy tắc:

    - Coi cây tổng quát là cây có thứ tự. Nếu cây tổng quát không có thứ tự thì ta đưa thêm vào một thứ tự trong cây đó.
    - Chuyển nút con trái nhất (con cả) của một nút thành nút con trái của nó
    - Chuyển nút em kế cận của một nút thành nút con phải của nút đó.

Cây tổng quát

– Ví dụ



Cây tổng quát

- Cài đặt gián tiếp qua cây nhị phân

  Nhận xét:

  – Cách cài đặt này cho phép ta cài đặt một cây bất kì, có thể có cấp rất lớn và rất hiệu quả về bộ nhớ.

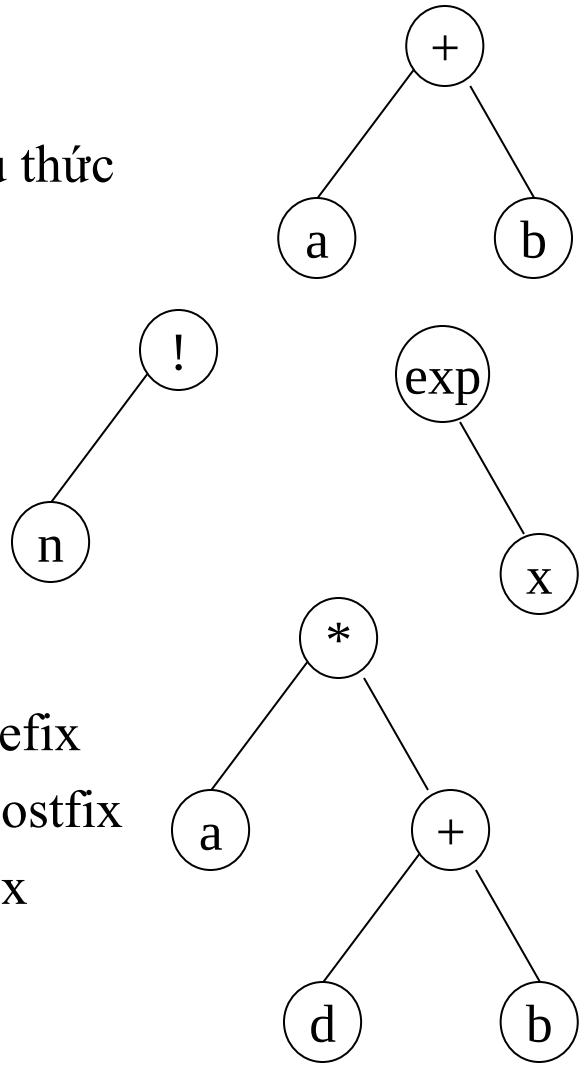
  – Cần thêm quá trình chuyển đổi ngữ nghĩa thuận và nghịch giữa cây tổng quát và cây nhị phân. Ví như bổ sung một nút là con cả của một nút trong cây tổng quát sẽ thành bổ sung một nút thành nút con trái trong cây NP => giảm tốc độ thực hiện các thao tác cơ bản như bổ sung, tìm kiếm trên cây.

Ứng dụng của cấu trúc cây

- Một số ứng dụng cây NP
  - Cây nhị phân tìm kiếm: ứng dụng vào lưu trữ và tìm kiếm thông tin một các hiệu quả nhất (tốn ít bộ nhớ và thời gian chạy nhanh nhất)
  - Cây biểu thức: giúp ta thực hiện hàng loạt các thao tác tính toán cơ bản cũng như phức tạp (do người dùng định nghĩa)
  - Cây cân bằng
- Một số ứng dụng khác
  - Cây biểu diễn các tập hợp
  - Cây hỗ trợ ra quyết định: decision tree

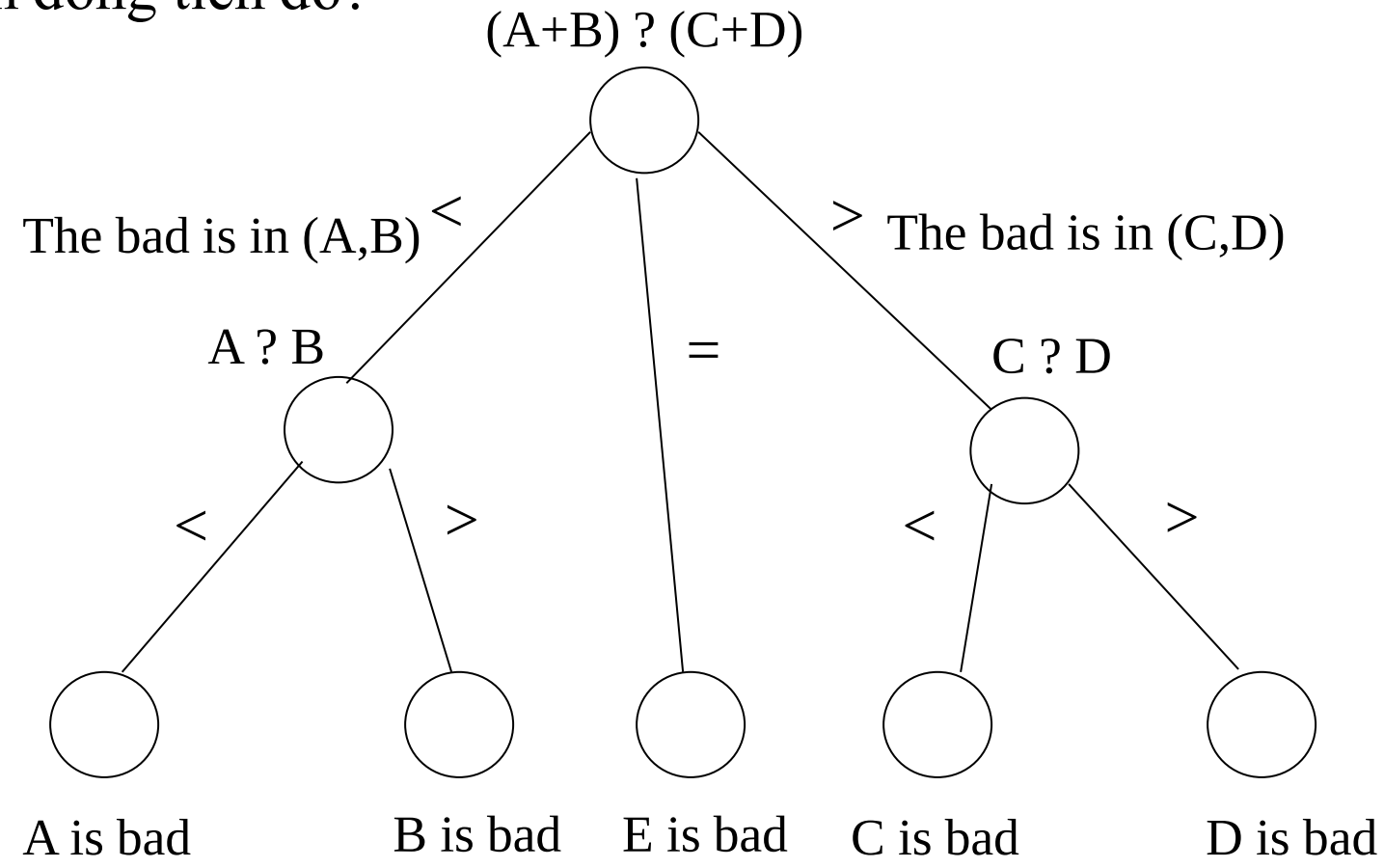
Ứng dụng cấu trúc cây NP

- Cây biểu thức (Ví dụ minh họa)
  - Biểu diễn biểu thức
    - Biểu thức = Biểu thức <Phép toán> Biểu thức
    - Đặc biệt: Biểu thức = const
    - Phép toán: +, / , *, -, exp, !, ...
  - Dùng cây nhị phân
    - Gốc: phép toán
    - Lá: toán hạng
  - Nhận xét:
    - Duyệt cây theo Preorder => biểu thức prefix
    - Duyệt cây theo Postorder => biểu thức postfix
    - Duyệt cây theo Inorder => biểu thức infix
  - Ví dụ: a+b, n!, exp(x), a*(d+b)



Ứng dụng cấu trúc cây NP

- Cây hỗ trợ ra quyết định (Ví dụ minh họa)
  - VD: Có 5 đồng tiền vàng. Trong đó có duy nhất 1 đồng bị nhẹ hơn. Tìm đồng tiền đó?



Bài tập

- **Bài 1**: Xây dựng một template class BinaryTree cho phép chứa các phần tử có kiểu khác nhau của cây nhị phân. Đồng thời nó có cài đặt các thao tác sau:
  - ***InitBTree():*** Khởi tạo một cây rỗng
  - ***InsertParent (type x, PNode P):*** bổ sung phần tử x trở thành nút cha của nút P trong cây
  - ***InsertLeftChild(type x, PNode P):*** bổ sung phần tử x trở thành nút con trái của nút P trong cây
  - ***InsertRightChild(type x, PNode P):*** bổ sung phần tử x trở thành nút con phải của nút P trong cây
  - ***PNode FindNode(type x):*** tìm một nút có giá trị bằng x trong cây, hàm trả về con trỏ trỏ vào nút tìm thấy, trái lại trả về NULL
  - ***DeleteCurrentNode(PNode P):*** xóa nút hiện trỏ bởi P trong cây
  - ***DeleteLeftChild(PNode P):*** xóa nút con trái của P
  - ***DeleteRightChild(PNode P):*** xóa nút con phải của P
  - ***GetSize():*** trả về kích thước của cây

Bài tập

- Bài 2: xây dựng một class Folder biểu diễn cây thư mục, trong đó một Folder có thể chứa nhiều File và SubFolder. Yêu cầu định nghĩa class Folder và cài đặt các thao tác cơ bản:
  - ***Init():*** tạo ra một thư mục rỗng
  - ***PNode GetCurrentFolder():*** trả về con trỏ trỏ vào thư mục hiện tại
  - ***Insert(Folder sub):*** bổ sung Folder sub thành con của thư mục hiện tại
  - ***Insert(File f):*** bổ sung File f thành con trong thư mục hiện tại
  - ***PNode Find(string fname) :*** tìm tên file hoặc thư mục có tên fname cho trước. Hàm trả về con trỏ trỏ vào nút tìm được

Bài tập

- Bài 3: xây dựng một cây biểu thức, trong đó các nút trong chứa các phép toán +,-,*,/; còn các nút lá chứa các toán hạng là các hằng số hoặc biến số. Yêu cầu định nghĩa class Expression để biểu diễn cho cây trên, đồng thời cài đặt các hàm sau:
  - Init(): khởi tạo một biểu thức rỗng
  - Init(string exp): khởi tạo một cây biểu thức từ chuỗi biểu thức exp
  - Value(): tính giá trị cây biểu thức

Xin cảm ơn!