Cây (Tree)

Cây

- Lịch sử ra đời
  - Được đề xuất bởi nhà toán học người Anh Arthur Cayley vào năm 1857
  - Để liệt kê các đồng phân của các hợp chất có dạng CnH2n + 2

Cây

- Cây là đồ thị vô hướng liên thông không có chu trình đơn
- Rừng (forest)
  - Đồ thị không chứa bất kỳ chu trình đơn nào liên kết với nhau

- **Định lý**: Một đồ thị vô hướng là cây nếu và chỉ nếu tồn tại một đường đi duy nhất giữa hai đỉnh bất kỳ nào của nó.

Cây có gốc (Rooted tree)

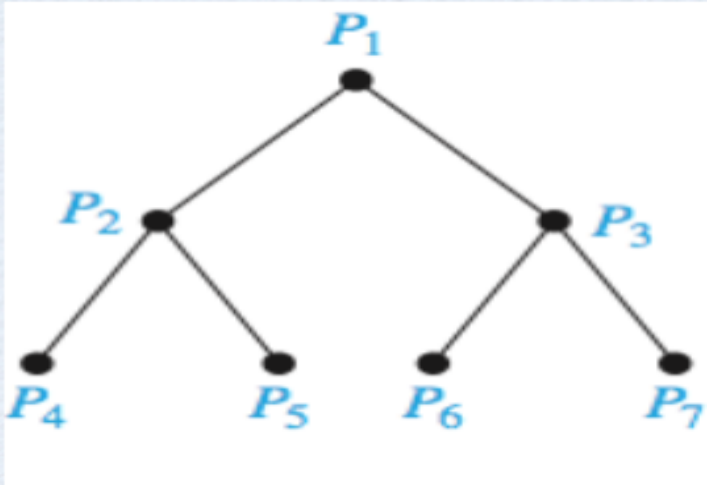
- Một đỉnh được chỉ định làm gốc và mọi cạnh được hướng ra từ gốc.
  - Cha mẹ, con, các đỉnh trong (internal nodes), lá
- Cây (có gốc) được gọi là cây m-ary nếu mỗi đỉnh trong của nó không có nhiều hơn m con.
  - Cây này được gọi là cây m-ary đầy đủ nếu các đỉnh trong của nó có chính xác m con.
  - Với m = 2, ta có cây nhị phân.

Cây có gốc sắp xếp

- Cây (có gốc) trong đó mỗi đỉnh trong được sắp xếp
- Cây nhị phân sắp xếp (thường được gọi là cây nhị phân)
  - Cây con bên trái và cây con bên phải

Cây

- Cây như là một mô hình tính toán
  - Các bộ vi xử lý (CPU) được tổ chức như trong hình 12 được sử dụng nhằm tìm tổng của 8 số.
  - Quy trình này gồm 3 bước.
    - Cộng x1 và x2 sử dụng P4
    - Cộng x3 và x4 sử dụng P5
    - Cộng x5 và x6 sử dụng P6
    - Cộng x7 và x8 sử dụng P7



Mạng kết nối dưới dạng cây của 07 bộ vi xử lý.

Các tính chất của cây

- Một cây có n đỉnh có n-1 cạnh
  - G liên thông
  - G không có chu trình
  - G có n - 1 cạnh

- Khi một trong ba điều kiện trên thỏa mãn, điều kiện còn lại cũng chắc chắn cũng thỏa mãn và G là một cây.

Các tính chất của cây

- Một cây đầy đủ m-ary với i đỉnh trong thì chứa n=m*i+1 đỉnh.

- Một cây đầy đủ m-ary với:
  - n đỉnh thì có i=(n-1)/m đỉnh trong và l = [(m − 1)n + 1]/m lá.
  - i đỉnh trong thì có n=mi+1 đỉnh và l=(m-1)i + 1 lá.
  - l lá thì có n=(ml-1)/(m-1) đỉnh và i=(l-1)/(m-1) đỉnh trong.

Cây m-ARY đối xứng

- Bậc của một đỉnh v trong cây có gốc là độ dài của đường đi duy nhất từ gốc tới đỉnh đó.

- Độ cao của cây có gốc là giá trị lớn nhất của bậc của các đỉnh trong cây đó.

Cây m-ARY

- **Định lý**: Tồn tại nhiều nhất mh lá trong một cây m-ary với chiều cao là h.

- Nếu một cây m-ary chiều cao h có l lá thì h>=[logm(l)]. Nếu cây đó đầy đủ và đối xứng thì h=[logm(l)].

Ứng dụng của cây

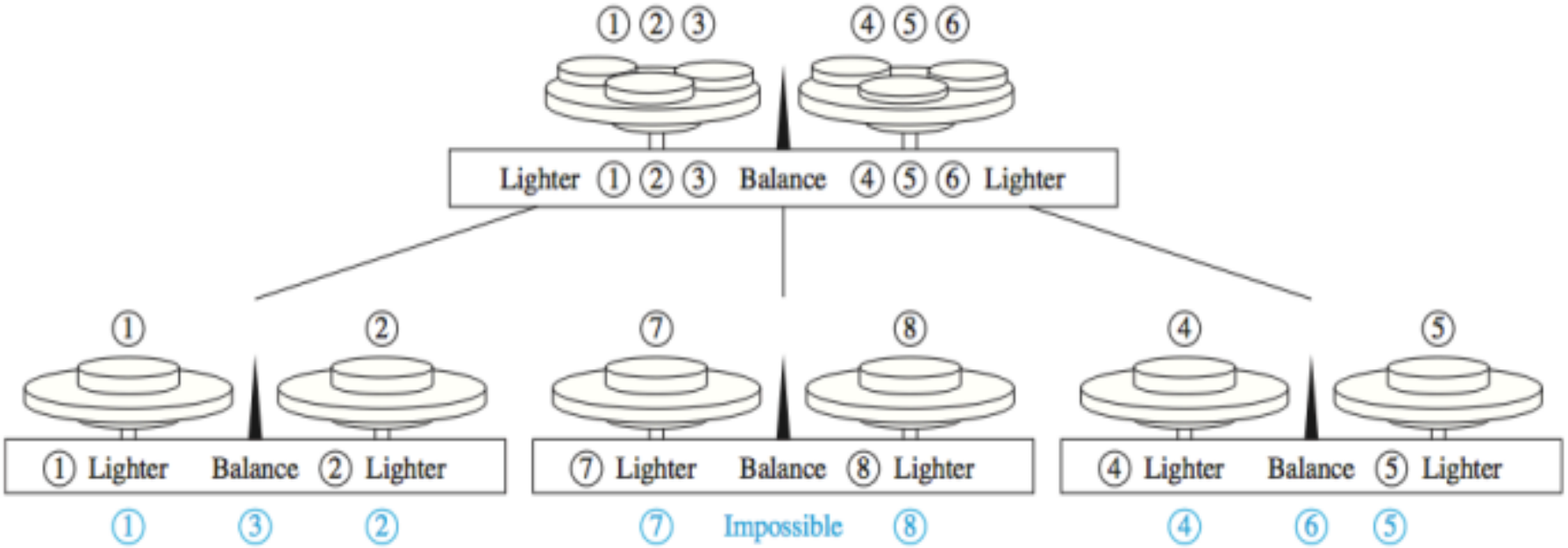
- Lưu trữ các đối tượng của một danh sách
  - Mỗi đối tượng được tìm thấy một cách dễ dàng?
- Hỗ trợ ra quyết định
  - Tìm ra một đối tượng với một tiêu chí cụ thể?
- Mã hóa một tập các ký tự thành chuỗi bit

Tìm kiếm cây nhị phân

- Mỗi đỉnh có duy nhất hai con: con trái và con phải.
  - Mỗi đỉnh chứa một giá trị trong danh sách (mảng) các giá trị.

- Giá trị của một đỉnh lớn hơn tất cả các giá trị của các đỉnh nằm trong cây con trái và nhỏ hơn tất cả các giá trị của các đỉnh nằm trong cây con phải.

Cây quyết định (decision tree)



Một cây quyết định cho việc phát hiện một đồng xu giả trong 6 đồng xu. (*Lighter: Nhẹ hơn; Balance: Cân bằng; Impossible: Không thể xác định*). Đồng xu giả được in màu (xanh) ở dưới mỗi lần cân cuối cùng.

Thuật toán sắp xếp

- **Định lý**: Một thuật toán sắp xếp dựa vào so sánh nhị phân cần ít nhất [logn!] so sánh.

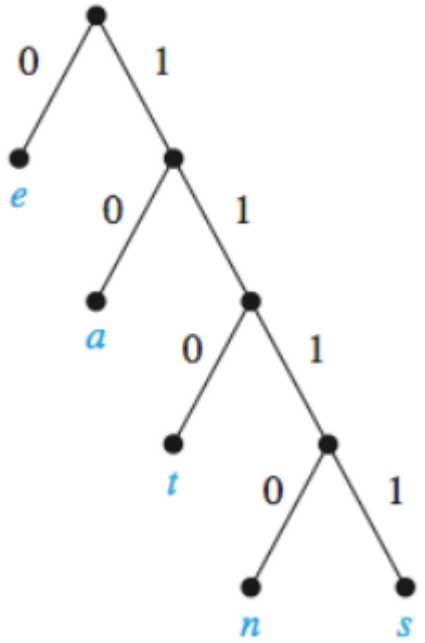
- **Kết luận**: Số so sánh cần sử dụng trong một thuật toán để sắp xếp n phần tử dựa vào so sánh nhị phân là **Ω**(nlogn)

Mã tiền tố (prefix code)

- Các dãy bit có độ dài khác nhau được sử dụng để mã hóa các ký tự.
  - Các ký tự có tần suất xuất hiện lớn nên được mã hóa bởi dãy bit ngắn
  - Các dãy bit dài được sử dụng để mã hóa các ký tự ít xuất hiện.
- Một vài phương pháp được sử dụng nhằm xác định dãy bit cho từng ký tự.
- Mã tiền tố
  - Dãy bit mã hóa một ký tự không bao giờ là tiền tố (phần đầu tiên) của dãy bit của các ký tự nào khác.

Mã tiền tố

- Một mã tiền tố có thể được biểu diễn bằng một cây nhị phân
  - Các ký tự là nhãn của các lá trên cây
  - Cạnh nối với con trái được gán là 0 và cạnh được nối với con phải được gán là 1.



Mã hóa Huffman

- Được đề xuất bởi David Huffman trong một công trình ông viết vào năm 1951 khi là một nghiên cứu sinh tại MIT.

- **Thuật toán**
  - Có đầu vào là tần suất xuất hiện của các ký tự trong một chuỗi
  - Đầu ra là một mã tiền tố nhằm mã hóa chuỗi đó sử dụng ít nhất có thể số lượng bit.

Mã hóa Huffman

Procedure Huffman(C: các ký tự a_i với tần số $w_i, i = 1, \ldots, n$)

F := rừng gồm n cây có gốc, mỗi cây bao gồm chỉ một đỉnh a_i và trọng số w_i

while F không phải là một cây

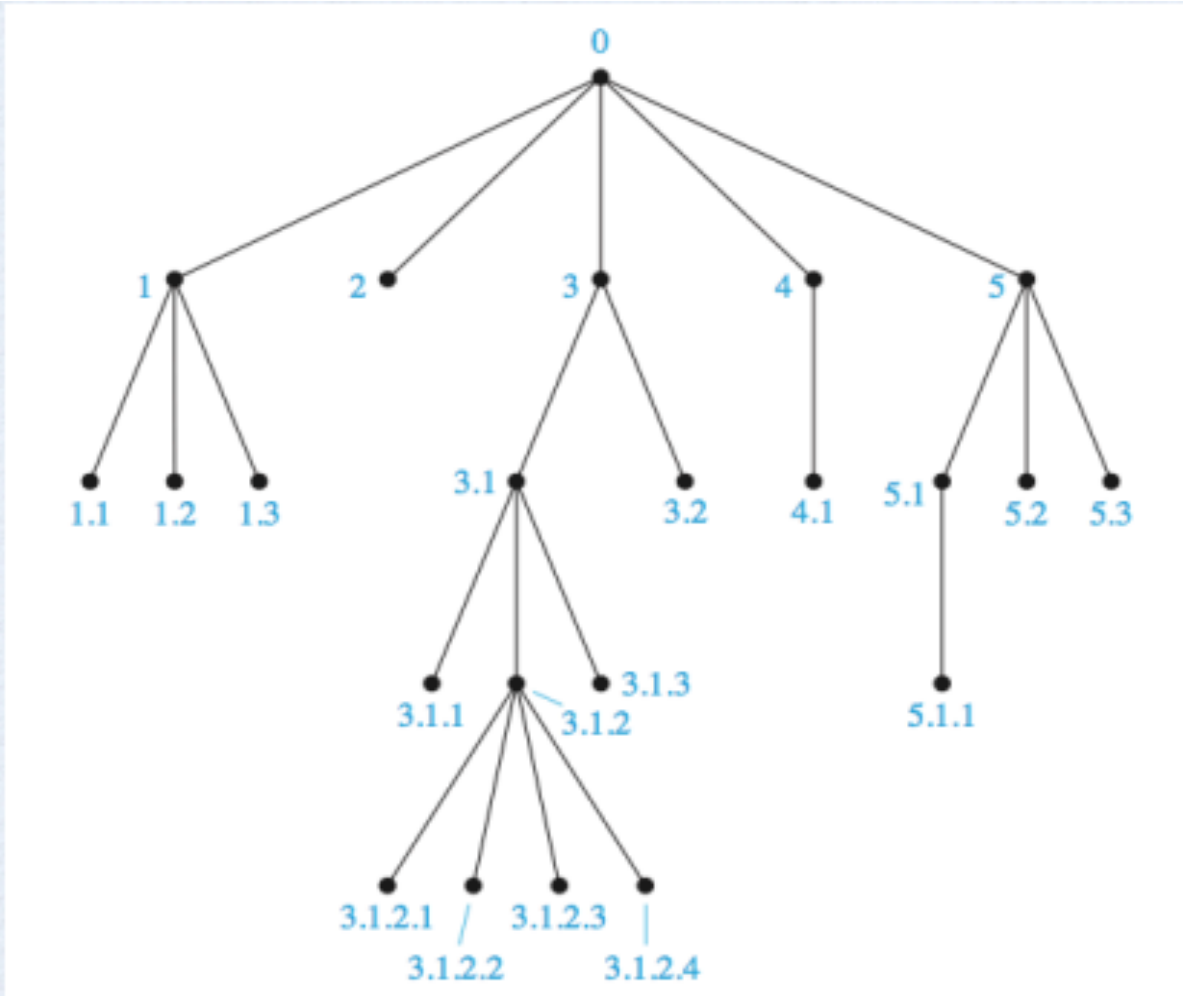
Thay 2 cây *T* và *T'* có trọng số nhỏ nhất trong *F* với *w(T)* ≥ *w(T')* bằng một cây có gốc mới trong đó *T* là cây con bên trái còn *T'* là cây con bên phải. Gán nhãn cạnh mới tới *T* là 0 còn cạnh mới tới *T'* là 1.

Gán *w(T) + w(T')* là trọng số cho cây mới tạo.

{Mã Huffman cho ký tự a_i là chuỗi các nhãn của các cạnh trên đường đi duy nhất từ gốc tới đỉnh lá a_i}

Duyệt cây (tree traversal)

- Hệ thống đánh địa chỉ



Thuật toán duyệt cây

- Duyệt tiền tự (pre-order)
- Duyệt trung tự (in-order)
- Duyệt hậu tự (post-order)

Duyệt tiền tự

Giả sử T là cây có gốc r đã sắp xếp. Nếu T chứa duy nhất gốc r thì r được duyệt tiền thứ tự. Ngược lại, giả sử T_1, T_2, ..., T_n là các cây con tại r từ trái sang phải của T. Duyệt tiền thứ tự bắt đầu tại vị trí của r. Nó tiếp tục duyệt tiền thứ tự qua T_1, rồi đến T_2, cứ tiếp tục như vậy cho đến khi T_n được duyệt tiền thứ tự

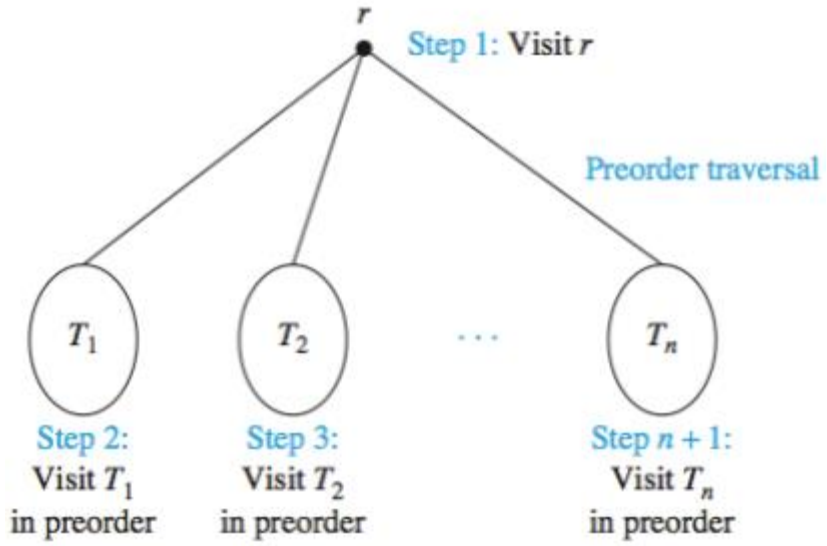


FIGURE 2 **Preorder Traversal.**

ALGORITHM 1 Preorder Traversal.

procedure *preorder*(*T*: ordered rooted tree)
r := root of *T*
list *r*
for each child *c* of *r* from left to right
  T(*c*) := subtree with *c* as its root
  preorder(*T*(*c*))

Duyệt trung tự

Giả sử T là cây có gốc r đã sắp xếp. Nếu T chứa duy nhất gốc r thì r được duyệt trung thứ tự. Ngược lại, giả sử T_1, T_2, ..., T_n là các cây con tại r từ trái sang phải của T. Duyệt trung thứ tự bắt đầu tại vị trí của T_1. Tiếp theo, nó duyệt trung thứ tự qua r. Nó tiếp tục duyệt trung thứ tự qua T_2, rồi đến T_3, cứ tiếp tục như vậy cho đến khi T_n được duyệt trung thứ tự

ALGORITHM 2 Inorder Traversal.

procedure *inorder*(T: ordered rooted tree)
r := root of T
if r is a leaf **then** list r
else
  l := first child of r from left to right
  $T(l)$:= subtree with l as its root
  inorder($T(l)$)
  list r
  for each child c of r except for l from left to right
    $T(c)$:= subtree with c as its root
    inorder($T(c)$)

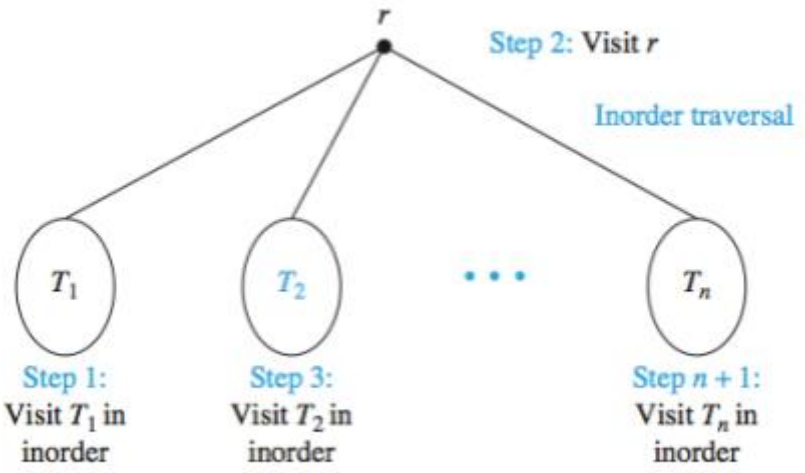


FIGURE 5 Inorder Traversal.

Duyệt hậu tự

Giả sử T là cây có gốc r đã sắp xếp. Nếu T chứa duy nhất gốc r thì r được duyệt hậu thứ tự. Ngược lại, giả sử T_1, T_2, ..., T_n là các cây con tại r từ trái sang phải của T. Duyệt hậu thứ tự bắt đầu tại vị trí của T_1. Nó tiếp tục duyệt hậu thứ tự qua T_2, rồi đến T_3, cứ tiếp tục như vậy cho đến khi T_n và kết thúc bằng việc duyệt r.

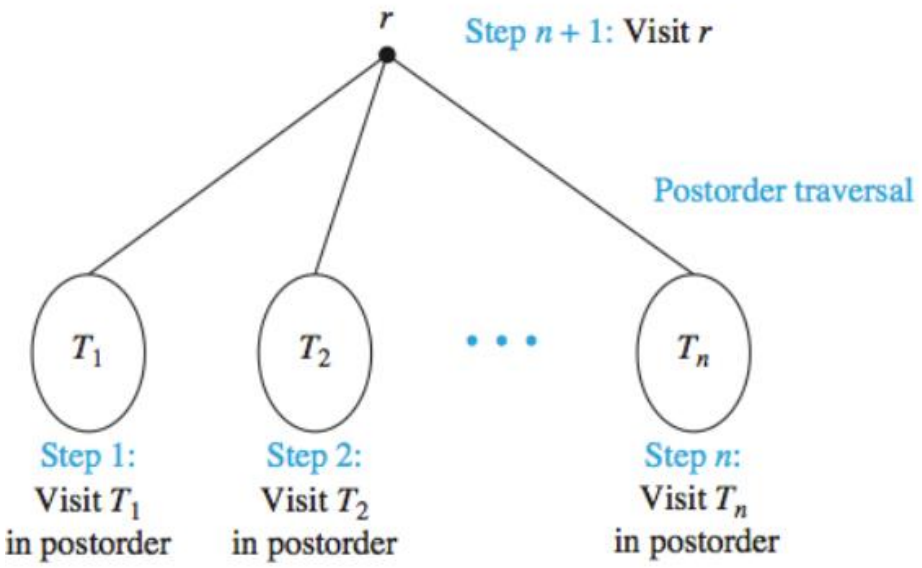


FIGURE 7 **Postorder Traversal.**

ALGORITHM 3 Postorder Traversal.

procedure *postorder*(*T*: ordered rooted tree)
r := root of *T*
for each child *c* of *r* from left to right
  T(*c*) := subtree with *c* as its root
  postorder(*T*(*c*))
list *r*