CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Bài 04: Cây

Nguyễn Thị Tâm
nguyenthitam.hus@gmail.com

Ngày 15 tháng 10 năm 2025

Nội dung

Cây

Định nghĩa cây

Cây bao gồm các nút, có một nút đặc biệt gọi là nút gốc (root) và các cạnh nối các nút. Cây được định nghĩa đệ quy như sau:

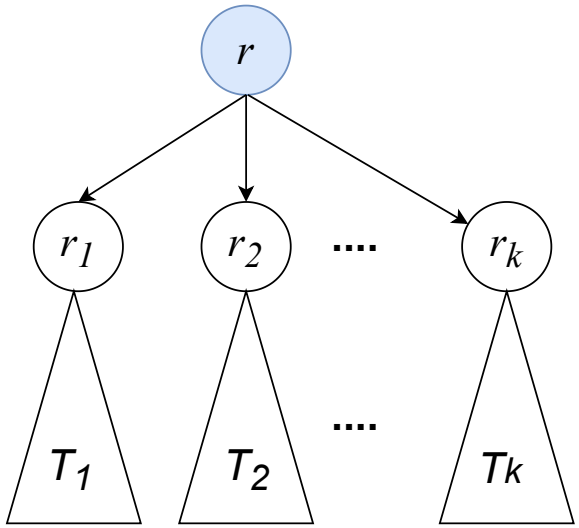
- Bước cơ sở: một nút r được gọi là cây và r gọi là gốc cây
- Bước đệ quy: Giả sử $T_1, T_2, ..., T_k$ là các cây với gốc là $r_1, r_2, ..., r_k$ ta có thể xây dựng cây mới bằng cách đặt r làm nút cha (parent) của các nút $r_1, r_2, ..., r_k$. Trong cây mới, tạo ra r là gốc và $T_1, ..., T_k$ là các cây con có gốc r. Các nút $r_1, r_2, .., r_k$ gọi là các con của nút r

Cây

Định nghĩa cây

Cây bao gồm các nút, có một nút đặc biệt gọi là nút gốc (root) và các cạnh nối các nút. Cây được định nghĩa đệ quy như sau:

- Bước cơ sở: một nút r được gọi là cây và r gọi là gốc cây
- Bước đệ quy: Giả sử $T_1, T_2, ..., T_k$ là các cây với gốc là $r_1, r_2, ..., r_k$ ta có thể xây dựng cây mới bằng cách đặt r làm nút cha (parent) của các nút $r_1, r_2, ..., r_k$. Trong cây mới, tạo ra r là gốc và $T_1, ..., T_k$ là các cây con có gốc r. Các nút $r_1, r_2, .., r_k$ gọi là các con của nút r



Cây

Các ứng dụng

Các ứng dụng của dữ liệu trừu tượng cây

- Cây gia phả
- Biểu đồ phân cấp quản lý
- Cây thư mục quản lý tệp
- Cây biểu thức
- ....

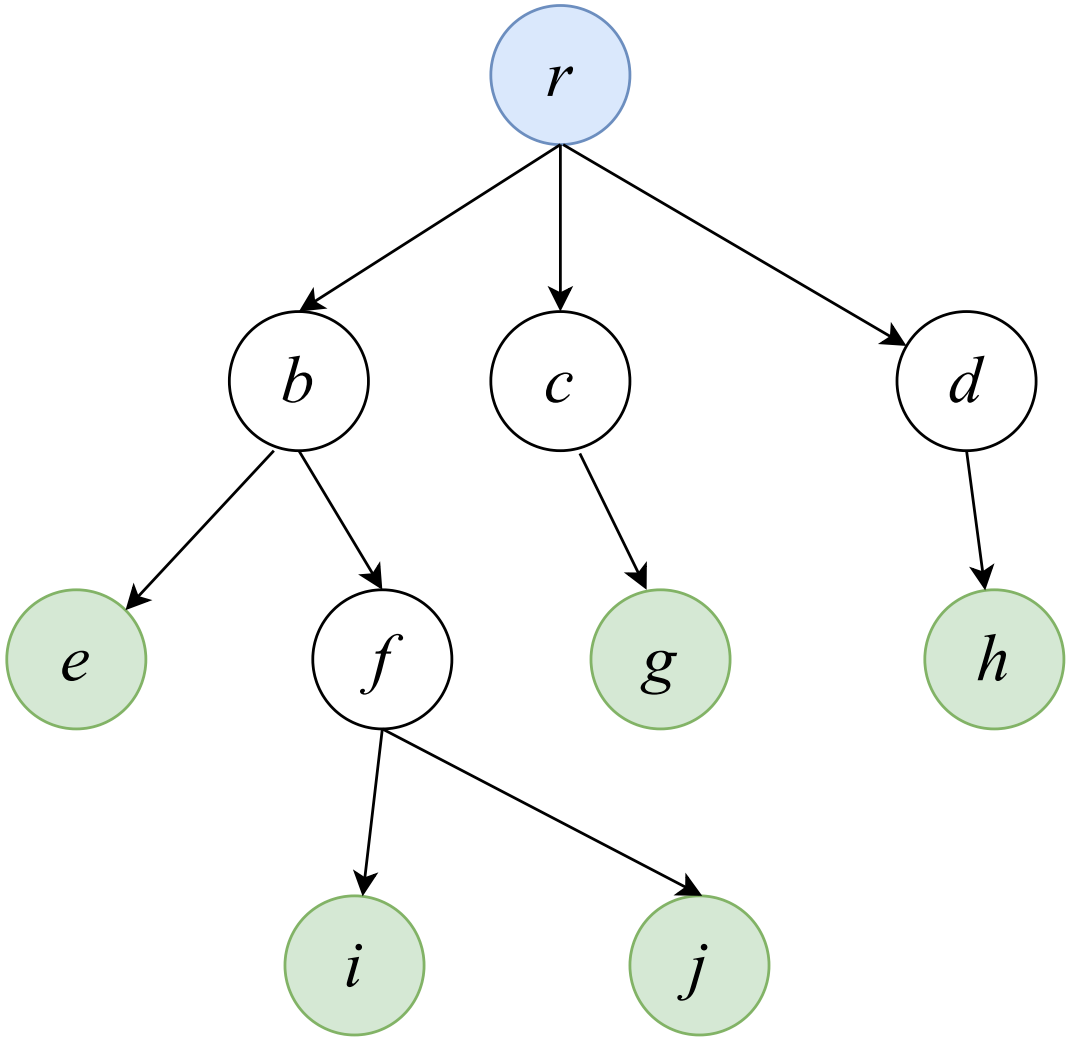
Cây

Các thuật ngữ chính

- Nút lá (*leaf*) là nút không có bất cứ con nào
- Nút p được gọi là tổ tiên (*ascentors*) của q nếu p nằm trên đường đi từ gốc đến q. q được gọi là hậu duệ (*descendants*) của p
- Anh em (*sibling*): các nút có cùng một cha
- Độ sâu (*depth*) của một nút là độ dài của đường đi từ gốc đến nút đó
- Chiều cao (*height*) của một nút là độ dài đường đi từ nút đó đến nút sâu nhất.
- Chiều cao của một cây là chiều cao tối đa trong tất cả các nút trong cây, trong khi đó độ sâu của một cây là độ sâu tối đa trong tất cả các nút trong cây. Đối với một cây nhất định, độ sâu và chiều cao trả về cùng một giá trị. Tuy nhiên, đối với từng nút cụ thể, độ sâu và chiều cao có thể khác nhau
- Kích thước của một nút (*size*) là số lượng các nút hậu duệ của nút đó (bao gồm cả chính nó).

Cây

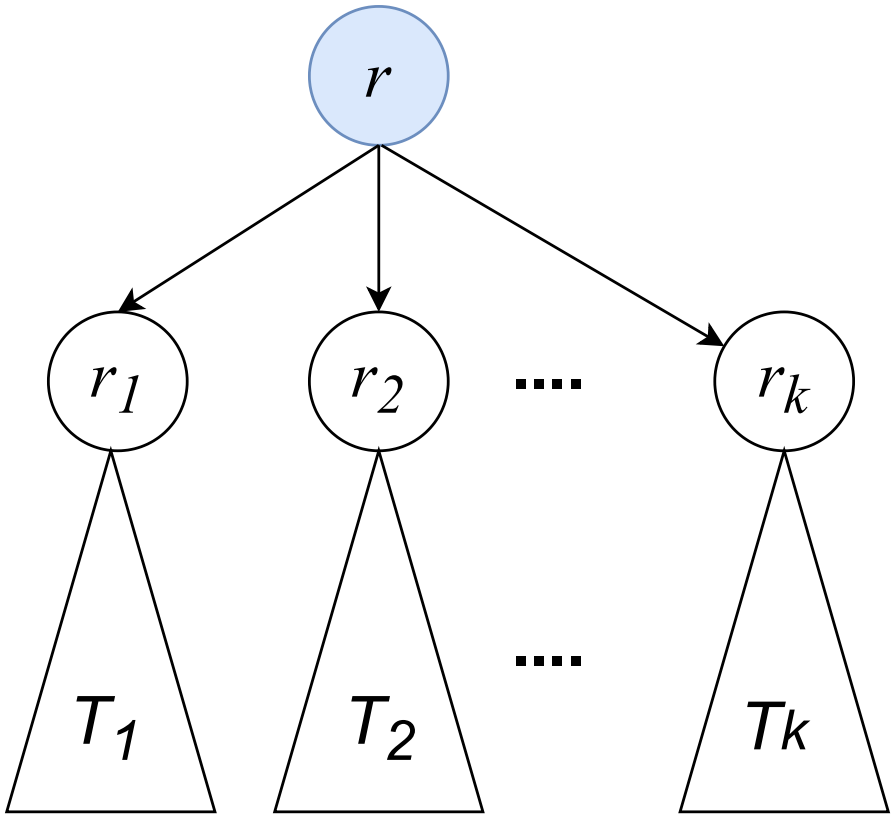
Phân loại các nút trong cây



Cây

Duyệt cây

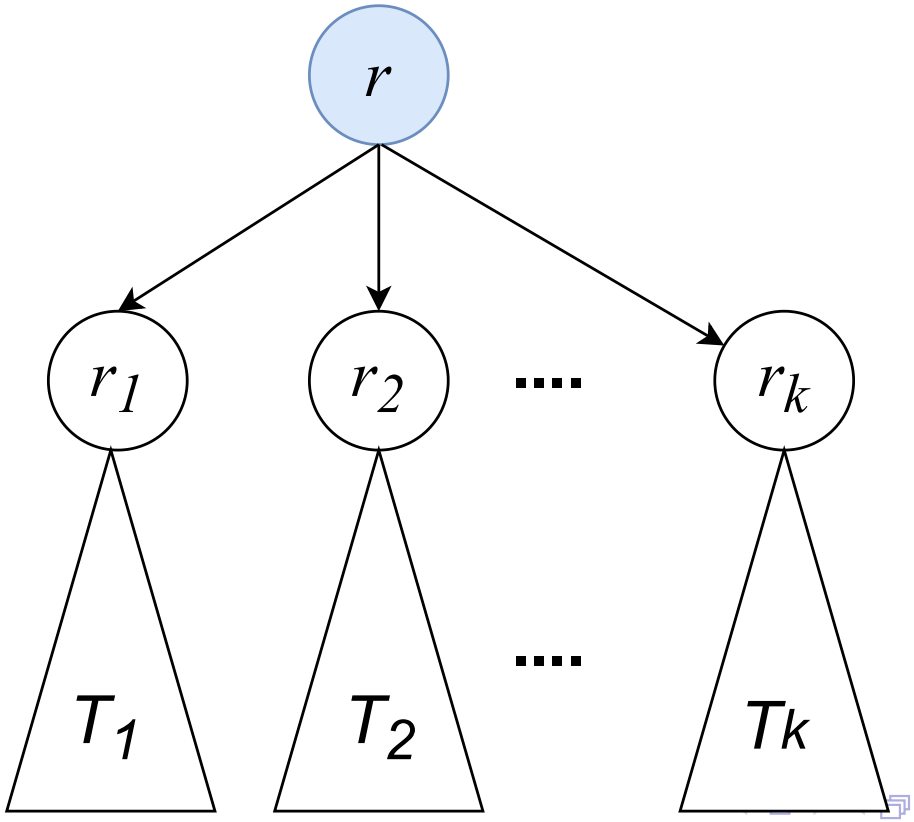
- Thứ tự trước (preorder)
- Thứ tự giữa (inorder)
- Thứ tự sau (postorder)



Duyệt cây

Duyệt thứ tự trước - Preorder traversal

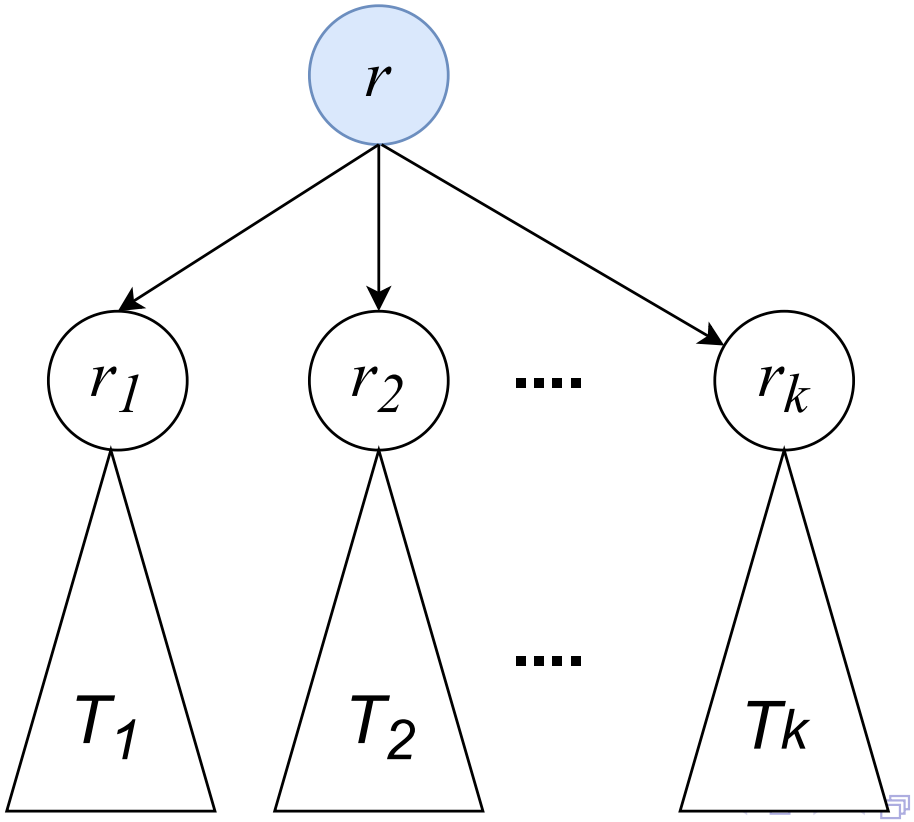
- Gốc r của cây
- Các nút của T_1 được duyệt theo thứ tự trước
- Các nút của T_2 được duyệt theo thứ tự trước
- ....
- Các nút của T_k được duyệt theo thứ tự trước



Duyệt cây

Duyệt thứ tự sau - Postorder traversal

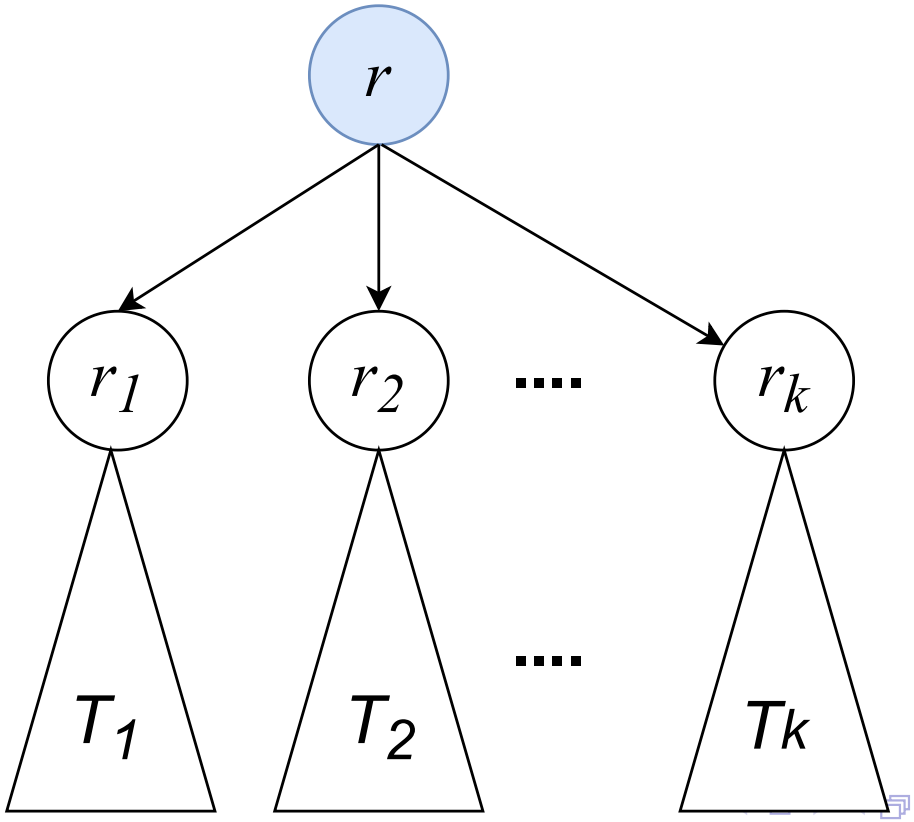
- Các nút của T_1 được duyệt theo thứ tự sau
- Các nút của T_2 được duyệt theo thứ tự sau
- ....
- Các nút của T_k được duyệt theo thứ tự sau
- Gốc r của cây



Duyệt cây

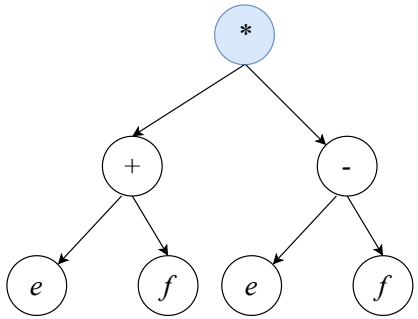
Duyệt thứ tự giữa - Inorder traversal

- Các nút của T_1 được duyệt theo thứ tự giữa
- Gốc r của cây
- Các nút của T_2 được duyệt theo thứ tự giữa
- ....
- Các nút của T_k được duyệt theo thứ tự giữa



Cây có nhãn - labeled tree

- Mỗi nút được gán một nhãn hoặc một giá trị, cũng giống như gán mỗi nút trong danh sách tuyến tính một phần tử (element). Nghĩa là nhãn của nút không chỉ là tên gọi mà mang ý nghĩa là giá trị của nút đó. Ví dụ như cây biểu thức
- Cây biểu thức



Hình 1: Biểu thức: (e+f)*(e-f)

- Quy tắc biểu diễn cây biểu thức
  - Mỗi nút lá là toán hạng
  - Mỗi nút trong là toán tử. Với phép toán hai ngôi $E_1 q E_2$, với $q = \{+, -, *, /\}$, thì cây con trái biểu diễn biểu thức E_1, cây con phải biểu diễn biểu thức E_2

Cây nhị phân (1)

Định nghĩa cây nhị phân - binary tree

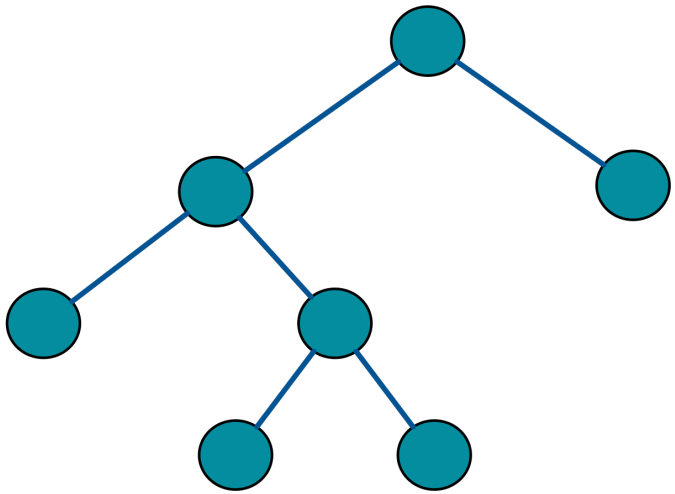
Cây nhị phân là cây mà mỗi nút có nhiều nhất hai nút con gọi là con trái và con phải. Một số tính chất của cây nhị phân

- Số đỉnh lớn nhất ở trên mức i của cây nhị phân là $2^i, i \geq 0$
- Mỗi cây nhị phân với chiều cao h có không quá $2^{h+1} - 1$ nút với $h \geq 0$
- Một cây nhị phân có n nút có chiều cao tối thiểu là $\lceil log_2(n+1) \rceil$

Cây nhị phân (2)

Định nghĩa cây nhị phân đầy đủ - full binary tree

Cây nhị phân đầy đủ là cây nhị phân mà mỗi nút của nó hoặc có 0 hoặc có hai con



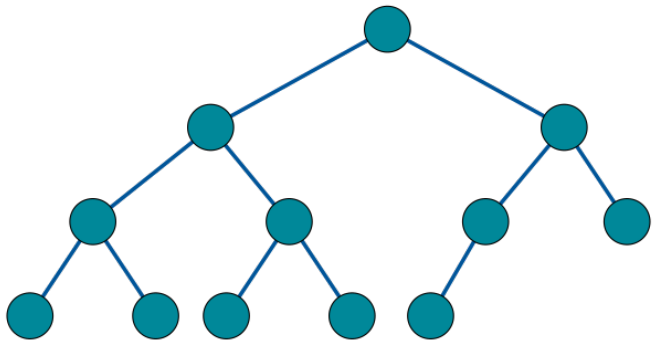
Hình 2: Cây nhị phân đầy đủ

Cây nhị phân (3)

Định nghĩa cây nhị phân hoàn chỉnh - complete binary tree

Cây nhị phân có độ sâu d thoả mãn:

- Tất cả các mức từ 0 đến $d-1$ đều được lấp đầy hoàn toàn
- Tất cả các nút tại độ sâu d lệch sang trái nhất có thể
- Cây nhị phân đầy đủ với độ sâu d thì số nút của nó nằm trong khoảng từ 2^d đến $2^{d+1}-1$.

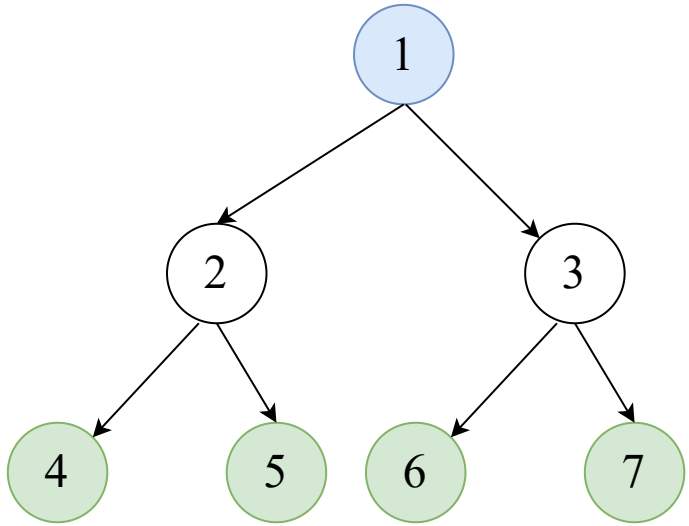


Hình 3: Cây nhị phân hoàn chỉnh

Cây nhị phân (4)

Định nghĩa cây nhị phân hoàn hảo - perfect binary tree

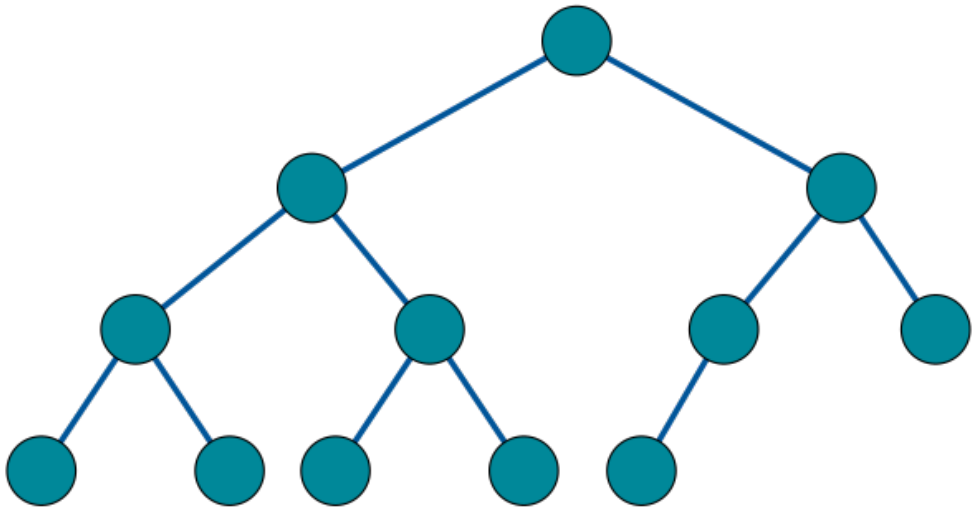
Cây nhị phân hoàn hảo là cây nhị phân mà tất cả các nút trong có đủ hai con và tất cả các nút lá ở cùng một mức



Cây nhị phân (5)

Định nghĩa cây nhị phân cân bằng - balanced binary tree

Cây nhị phân được gọi là cân bằng nếu chiều cao của cây con trái và cây con phải của mọi nút không lệch nhau quá 1 đơn vị

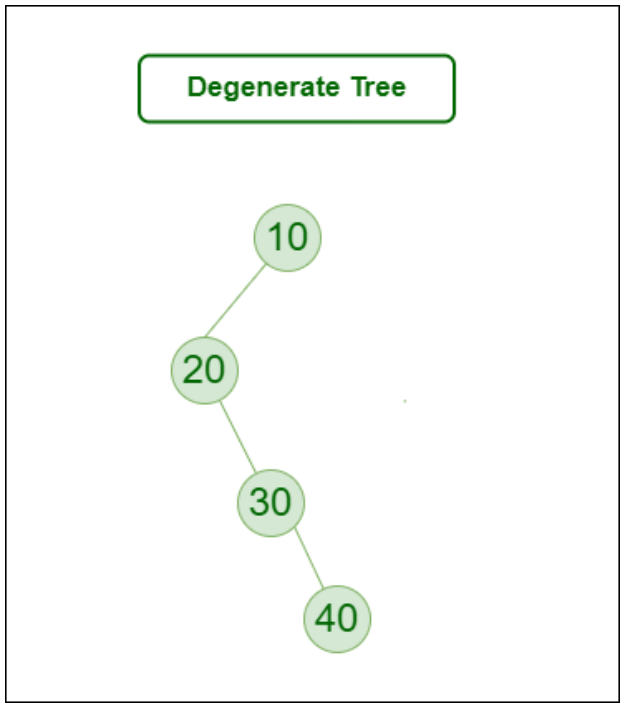


Hình 4: Cây nhị phân Cân bằng

Cây nhị phân (6)

Định nghĩa cây nhị phân thoái hoá - degenerate Tree

Cây nhị phân thoái hoá là một loại cây nhị phân mà mỗi nút trong (internal node) của nó đều chỉ có một con. Những cây nhị phân kiểu như vậy có hiệu suất tương tự với Linked List – Danh sách liên kết.

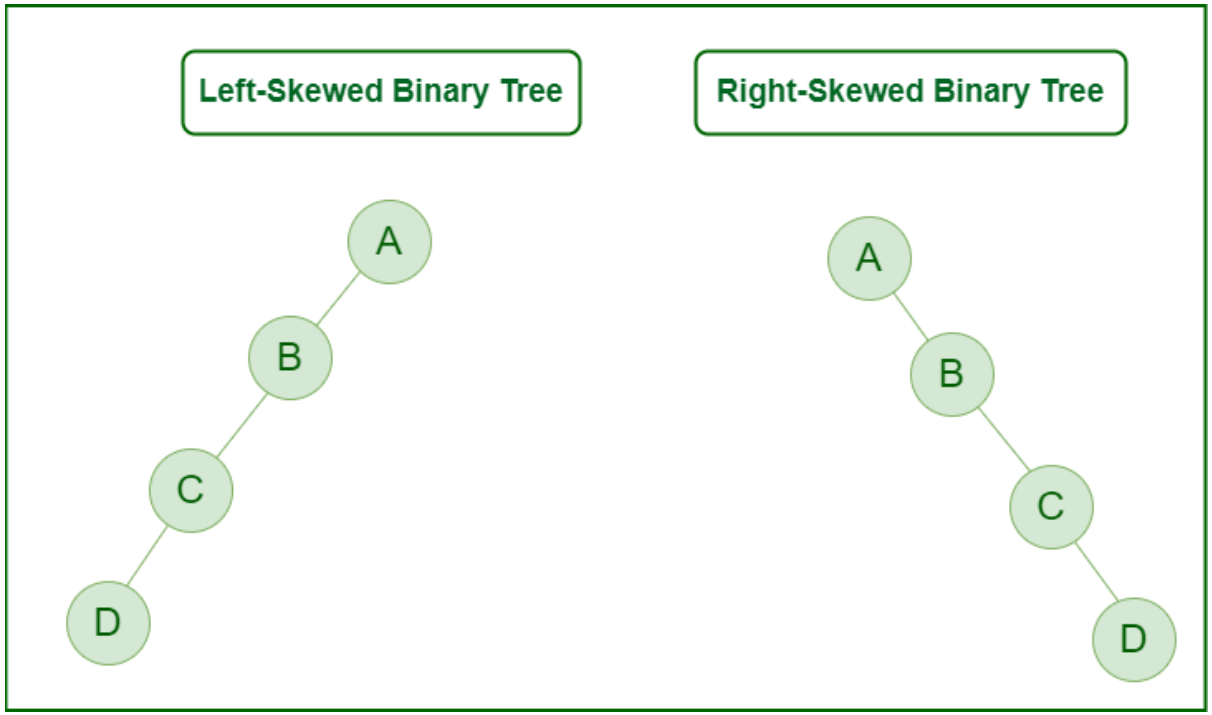


Hình 5: Cây nhị phân thoái hóa

Cây nhị phân (7)

Cây nhị phân lệch trái, lệch phải

Là cây nhị phân thoái hóa, trong đó các nút chỉ có con trái (nếu lệch trái), và chỉ có con phải (nếu lệch phải).



Hình 6: Cây nhị phân thoái hóa

Biểu diễn cây nhị phân

```
class Node {
    int data;
    Node left, right;

    public Node(int data){
        this.data = data;
    }
}
```

Một số bài toán trên cây nhị phân

1. Tìm chiều cao của cây
2. Tìm kích thước của cây
3. Đếm số nút lá trên cây
4. Duyệt cây
5. Kiểm tra cây đầy đủ
6. Kiểm tra các nút lá có cùng mức
7. Kiểm tra cây hoàn hảo

Một số bài toán trên cây nhị phân

Tìm chiều cao của cây

```
public int height(Node node) {
    if (node == null) {
        return 0;
    }
    int hleft = height(node.left);
    int hright = height(node.right);
    return Math.max(hleft, hright) + 1;
}
```

Một số bài toán trên cây nhị phân

Đếm số nút lá trên cây

```
int countLeaves(Node root) {
    if (root == null) {
        return 0;
    }
    if (root.left == null && root.right == null) {
        return 1;
    }
    return countLeaves(root.left) + countLeaves(root.right);
}
```

Một số bài toán trên cây nhị phân

Kiểm tra cây nhị phân đầy đủ

```
boolean isFullTree(Node root)
{
    if (root == null)
        return true;
    if (root.left == null && root.right == null)
        return true;
    if (root.left == null || root.right == null)
        return false;
    return isFullTree(root.left) && isFullTree(root.right);
}
```

Một số bài toán trên cây nhị phân

Kiểm tra các nút lá có cùng mức

```
int level = 0;
boolean check(Node root)
{
  return checkLevel(root,0);
}
boolean checkLevel(Node root, int cLevel){
    if(root == null){
        return true;
    }
    if(root.left == null && root.right == null){
        if(level == 0){
            level = cLevel;
            return true;
        }
        return level == cLevel;
    }
    return checkLevel(root.left,cLevel+1) &&
         checkLevel(root.right,cLevel+1);
}
```

Một số bài toán trên cây nhị phân

Kiểm tra cây nhị phân hoàn hảo

```
boolean isPerfectTree(Node root){
    return isFullTree(root) && check(root);
}
```

Cây nhị phân biểu thức - expression binary tree

Cây nhị phân biểu thức trong đó

- Mỗi nút lá chứa một toán hạng
- Mỗi nút trong chứa một phép toán
- Nếu là phép toán hai ngôi, thì cây con bên trái và cây con bên phải tương ứng với hai vế của phép toán đó

Các mức thể hiện mức độ ưu tiên của phép toán

- Mức của các nút trên cây cho biết trình tự thực hiện các phép toán (lưu ý không sử dụng dấu ngoặc trong cây nhị phân biểu thức)
- Các phép toán mức cao được thực hiện trước
- Phép toán ở gốc được thực hiện sau cùng

Duyệt cây nhị phân biểu thức

Duyệt cây nhị phân biểu thức cho ta các biểu diễn biểu thức dưới dạng trung tố, tiền tố và hậu tố

- Duyệt cây theo thứ tự trước (preorder) cho ta biểu thức dưới dạng tiền tố
- Duyệt cây theo thứ tự giữa (inorder) cho ta biểu thức dưới dạng trung tố
- Duyệt cây theo thứ tự sau (postorder) cho ta biểu thức dưới dạng hậu tố

Duyệt cây nhị phân biểu thức

Duyệt cây theo thứ tự trước

```
void preorder(Node node) {
   if (node == null)
      return;
   System.out.print(node.data + " ");
   preorder(node.left);
   preorder(node.right);
}
```

Duyệt cây nhị phân biểu thức

Duyệt cây theo thứ tự trước

```
void preorder(Node node) {
   if (node == null)
      return;
   System.out.print(node.data + " ");
   preorder(node.left);
   preorder(node.right);
}
```

Duyệt cây theo thứ tự giữa

```
void inorder(Node node) {
   if (node == null)
   return;
   inorder(node.left);
   System.out.print(node.data + " ");
   inorder(node.right);
}
```

Cây quyết định - decision tree

- Cây quyết định là cây mà mỗi nút là truy vấn đối với dữ liệu. Các cạnh của cây tương ứng với khả năng trả lời câu hỏi. Mỗi lá tương ứng với một đầu ra
- Để xác định quyết định, dữ liệu được đi vào từ gốc cây - truy vấn ở gốc. Sau đó đi xuống đến lá thì biết được đầu ra - thực chất là quyết định cuối cùng
- Các ứng dụng
  - Quyết định phân loại
  - Quyết định hỏi/đáp
  - Quyết định mờ (fuzzy decision tree) là mô hình hiệu quả trong học máy

Cây quyết định

Bài toán tra từ điển

Bài toán tra từ điển

Cho một mảng gồm n số được sắp xếp theo thứ tự tăng dần và một số x. Cần xác định xem x có mặt trong mảng đã cho hay không? Nếu câu trả lời khẳng định thì cần chỉ ra vị trí của x trong mảng.

Mã Huffman

Mã Huffman

Giả sử trong văn bản có bảng chữ cái C, với mỗi chữ cái $c \in C$ ta biết tần suất xuất hiện của nó trong văn bản là $f(C)$. Cần tìm cách mã hoá văn bản sử dụng ít bộ nhớ nhất.

Hai cách mã hoá phổ biến

- Mã hoá với độ dài cố định: Bảng mã ASCII mở rộng dùng 8 bit để biểu diễn 256 ký tự khác nhau
- Mã phi tiền tố (prefix free code): mỗi kí tự c được mã hoá bởi một xâu nhị phân $code(c)$ sao cho mã của kí tự bất kì không là đoạn đầu của bất cứ mã của kí tự nào khác trong các kí tự còn lại. Tuy nhiên, cách mã hoá này đòi hỏi việc mã hoá và giải mã phức tạp

Mã Huffman (1)

Mã Huffman

Mỗi mã phi tiền tố có thể biểu diễn bởi một cây nhị phân T mà mỗi lá của nó tương ứng với một chữ cái và cành của nó được gán cho một trong hai số 1 và 0. Mã của một chữ cái c là một dãy nhị phân gồm các số gán cho các cạnh trên đường đi từ gốc đến c

Bài toán

Tìm cách mã hoá tối ưu, tức là cây nhị phân T làm tối thiểu hoá tổng độ dài có trọng số

$$B(T) = \sum_{c \in C} f(c) \times depth(c)$$

trong đó, $depth(c)$ là độ dài đường đi từ gốc đến lá tương ứng với kí tự c, $f(c)$ là tần số xuất hiện của kí tự c

Mã Huffman (2)

Thuật toán

Ý tưởng

Chữ cái có tần suất xuất hiện ít hơn cần được gán cho lá có khoảng cách đến gốc là lớn hơn. Ngược lại, chữ cái có tần suất xuất hiện nhiều cần được gán cho nút gần gốc

Mã Huffman (3)

Thuật toán

Với ý tưởng trên, thuật toán Huffman coding gồm 3 bước:

1. Đếm tần suất xuất hiện của các phần tử trong chuỗi đầu vào.
2. Xây dựng cây Huffman (cây nhị phân mã hóa).
3. Từ cây Huffman, ta có được các giá trị mã hóa. Lúc này, ta có thể xây dựng chuỗi mã hóa từ các giá trị này.

Quá trình xây dựng cây Huffman gồm các bước sau:

1. Tạo danh sách chứa các nút lá bao gồm phần tử đầu vào và trọng số nút là tần suất xuất hiện của nó.
2. Từ danh sách này, lấy ra 2 phần tử có tần suất xuất hiện ít nhất. Sau đó gắn 2 nút vừa lấy ra vào một nút gốc mới với trọng số là tổng của 2 trọng số ở nút vừa lấy ra để tạo thành một cây.
3. Đẩy cây mới vào lại danh sách.
4. Lặp lại bước 2 và 3 cho đến khi danh sách chỉ còn 1 nút gốc duy nhất của cây.
5. Nút còn lại chính là nút gốc của cây Huffman.

Mã Huffman (4)

Ví dụ

Cho xâu "DHKHTN" với số lần xuất hiện của các kí tự giảm dần như sau:

Ký tự	'H'	'D'	'K'	'T'	'N'
Tần suất	2	1	1	1	1

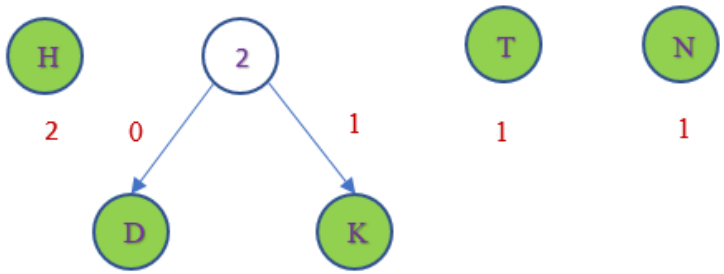
- Mỗi ký tự sẽ là một nút lá, kèm theo trọng số là tần suất xuất hiện của nó
- 2 nút có trọng số thấp nhất sẽ được chọn để dựng cây tại mỗi bước

Mã Huffman (5)

Ví dụ

Cho xâu "DHKHTN" với số lần xuất hiện của các kí tự giảm dần như sau:

Ký tự	'H'	'DK'	'T'	'N'
Tần suất	2	2	1	1

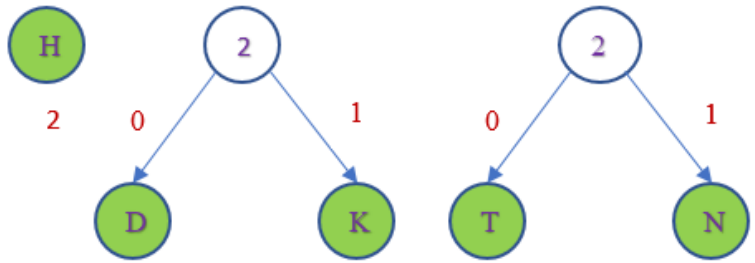


Mã Huffman (6)

Ví dụ

Cho xâu "DHKHTN" với số lần xuất hiện của các kí tự giảm dần như sau:

Ký tự	'H'	'DK'	'TN'
Tần suất	2	2	2

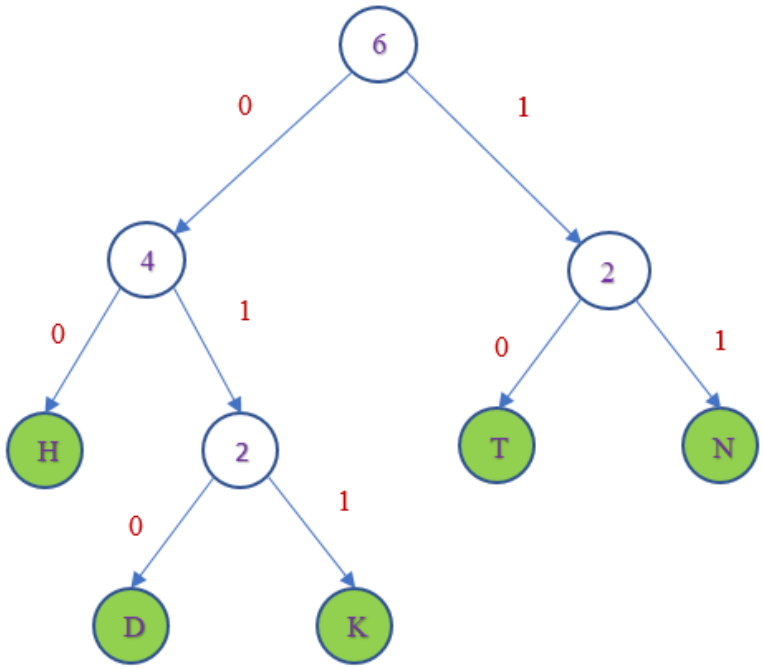


Mã Huffman (7)

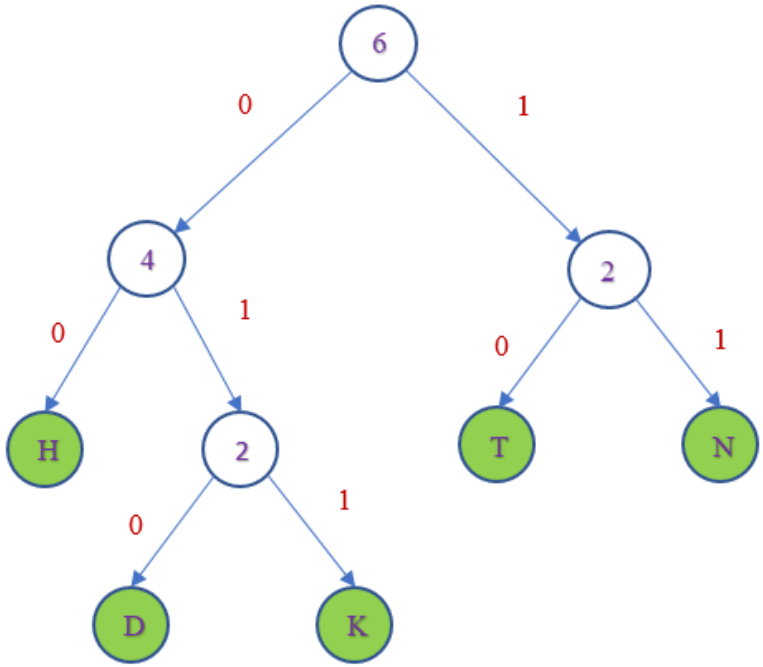
Ví dụ

Kết quả mã Huffman cho các ký tự trong văn bản "DHKHTN"

Ký tự	'H'	'D'	'K'	'T'	'N'
Mã Huffman	00	010	011	10	11



Mã Huffman (8)



Trong mỗi bước của thuật toán xây dựng cây Huffman, ta luôn phải chọn ra hai gốc có trọng số nhỏ nhất. Để làm việc này ta sắp xếp các gốc vào một hàng đợi ưu tiên theo tiêu chuẩn trọng số nhỏ nhất. Một trong các cấu trúc dữ liệu thuận lợi cho tiêu chuẩn này là cấu trúc đống (với phần tử có trọng số nhỏ nhất nằm trên đỉnh của đống).