

Cấu trúc dữ liệu và giải thuật

TS. Phạm Tuấn Minh

Khoa Công nghệ Thông tin, Đại học Phenikaa

minh.phamtuan@phenikaa-uni.edu.vn

<https://sites.google.com/site/phamtuanminh/>

Chương 3: Cây và bảng băm

- ❑ Các khái niệm cây
- ❑ Cây nhị phân tìm kiếm
- ❑ Cây AVL
- ❑ Bảng băm

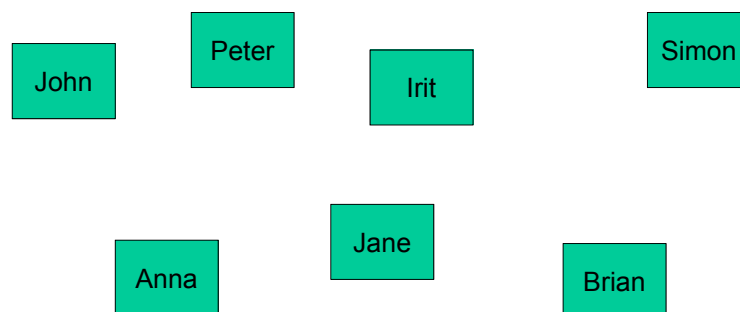
Cây

- ❑ Các cấu trúc dữ liệu không tuyến tính
- ❑ Cấu trúc dữ liệu cây
 - Cây nhị phân
- ❑ Cài đặt cây nhị phân
- ❑ Duyệt cây nhị phân
- ❑ Ứng dụng ví dụ

1-3

Ví dụ về sử dụng cấu trúc dữ liệu

- ❑ Giả sử có một tập các tên



- ❑ Quản lý tập tên như thế nào?

1-4

Cấu trúc dữ liệu tuyến tính

- Nếu các tên xếp danh sách xếp hàng



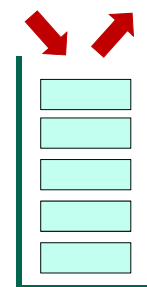
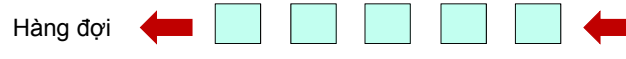
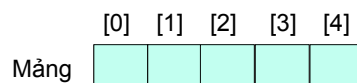
Danh sách

- Quản lý danh sách dữ liệu trên như thế nào ?

1-5

Cấu trúc dữ liệu tuyến tính

- Mảng, danh sách liên kết, hàng đợi, ngăn xếp

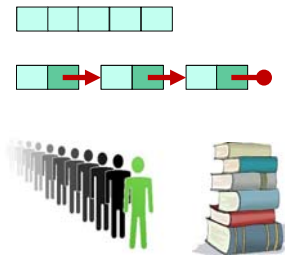


Ngăn xếp 1-6

Cấu trúc dữ liệu đã học

□ Tuyến tính:

- Tất cả các phần tử được sắp xếp có thứ tự
- Truy cập ngẫu nhiên
 - Mảng
- Truy cập tuần tự
 - Danh sách liên kết
- Truy cập tuần tự có hạn chế
 - Hàng đợi
 - Ngăn xếp

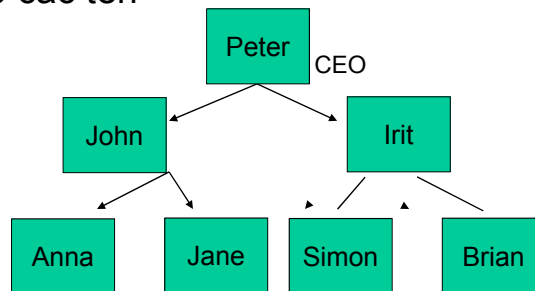


- Sử dụng để chứa danh sách các số, danh sách tên người
 - Cấu trúc tuyến tính

1-7

Cấu trúc dữ liệu không tuyến tính

□ Tập các tên



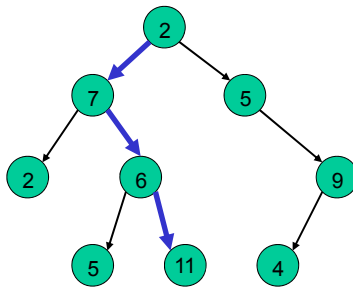
Cây

- Tổ chức công ty: Không dùng cấu trúc tuyến tính để lưu trữ do quan hệ phân cấp

1-8

Cấu trúc dữ liệu cây

- Vẫn sử dụng các biểu diễn nút và liên kết
- Đặc điểm mới:
 - Mỗi nút có liên kết tới nhiều hơn một nút khác
 - Không có lặp



Nếu đi theo một đường trên cây, sẽ nhận được một danh sách liên kết

1-9

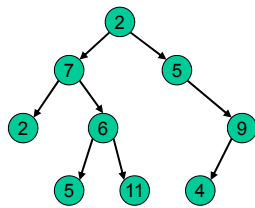
Cây

- Các cấu trúc dữ liệu không tuyến tính
- **Cấu trúc dữ liệu cây**
 - Cây nhị phân
- Cài đặt cây nhị phân
- Duyệt cây nhị phân
- Ứng dụng ví dụ

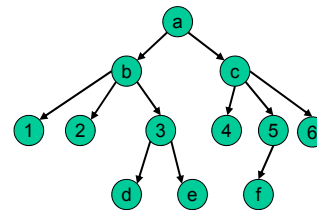
1-10

Cấu trúc dữ liệu cây

- Cấu trúc dữ liệu cây dạng như cây: gốc, nhánh, lá
 - Chỉ có một nút gốc
 - Mỗi nút nhánh trở tới một số nút khác
 - Mỗi nút chỉ có một nút cha (nút trở tới nó), trừ nút gốc



Cây nhị phân



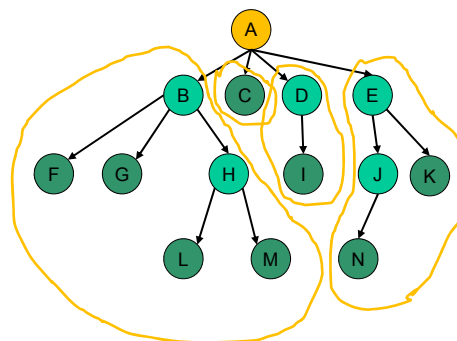
Cây tổng quát

- Cây tổng quát: Mỗi nút có liên kết tới không giới hạn các nút khác
- Cây nhị phân: Mỗi nút có liên kết tới tối đa hai nút

1-11

Cấu trúc dữ liệu cây

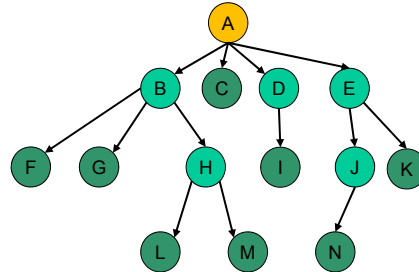
- Tương tự khái niệm cây gia đình
 - Một nút đặc biệt: nút gốc (root)
 - Mỗi nút có thể có nhiều con (children node)
 - Mỗi nút (trừ nút gốc) có một nút cha (parent node)
 - Các con của cùng cha là anh chị em (sibling node)
 - Cây con (subtree): Nút con và các nút cháu (descendant node) tạo thành cây con



1-12

Cấu trúc dữ liệu cây

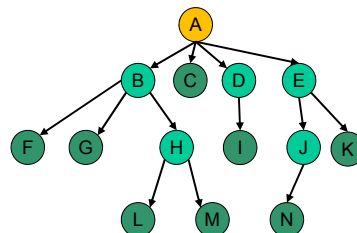
- ❑ Một cây bao gồm các nút
- ❑ Mỗi nút chứa một giá trị
- ❑ Các kiểu nút
 - Nút gốc (root): cây chỉ có một nút gốc, nút gốc không có nút cha
 - Nút trong (internal node): Là nút có nút con
 - Nút lá (leaf node): Là nút không có nút con



1-13

Sử dụng cây

- ❑ Các mô hình có quan hệ phân cấp giữa các phần tử
 - Chuỗi mệnh lệnh trong quân đội
 - Cấu trúc nhân sự của một công ty
- ❑ Cấu trúc cây cho phép
 - Mô tả một số bài toán tối ưu, mô tả trò chơi,...
 - Cho phép thực hiện nhanh
 - Tìm kiếm một nút với giá trị của nó
 - Thêm một giá trị vào danh sách
 - Xóa một giá trị từ danh sách



1-14

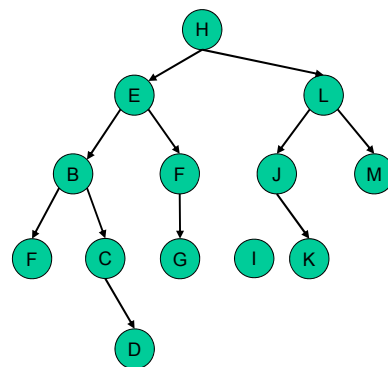
Ví dụ ứng dụng của cây

- ❑ Có thông tin sau
 - F có con là G
 - J có các con là I và K
 - B có các con là A và C
 - L có các con J và M
 - H có các con là E và L
 - C có con là D
 - E có các con là B và F
- ❑ Trả lời các câu hỏi
 - Ai không có con?
 - Ai là con cháu của L?
 - Ai là tổ tiên của J?
 - Ai có chính các 3 con cháu?

1-15

Ví dụ ứng dụng của cây

- ❑ Biểu diễn cây
 - F có con là G
 - J có các con là I và K
 - B có các con là A và C
 - L có các con J và M
 - H có các con là E và L
 - C có con là D
 - E có các con là B và F
- ❑ Trả lời các câu hỏi
 - Ai không có con?
 - Ai là con cháu của L?
 - Ai là tổ tiên của J?
 - Ai có chính xác 3 con cháu?



1-16

Cây

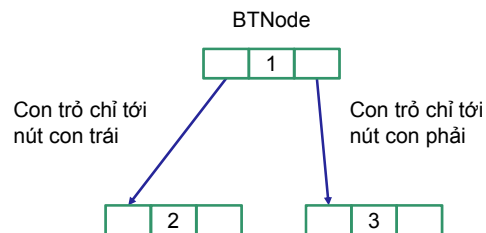
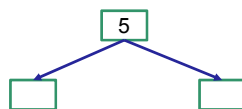
- ❑ Các cấu trúc dữ liệu không tuyến tính
- ❑ Cấu trúc dữ liệu cây
 - Cây nhị phân
- ❑ **Cài đặt cây nhị phân**
- ❑ Duyệt cây nhị phân
- ❑ Ứng dụng ví dụ

1-17

Cài đặt

- ❑ Nhắc lại cài đặt LinkedList
 - Nút có liên kết tới nhiều nhất một nút
 - Định nghĩa một ListNode với một con trỏ next và dữ liệu item
- ❑ BinaryTree
 - Một nút có liên kết tới nhiều nhất hai nút khác
 - Định nghĩa BTreeNode với
 - Hai con trỏ
 - Một đơn vị dữ liệu

```
typedef struct _listnode{  
    int item;  
    struct _listnode *next;  
} ListNode;
```



1-18

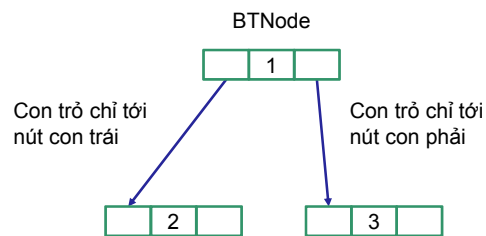
BTNode

```
typedef struct _listnode {  
    int item;  
    struct _listnode *next;  
} ListNode;
```

- Ví dụ một BTNode đơn giản chứa một số nguyên

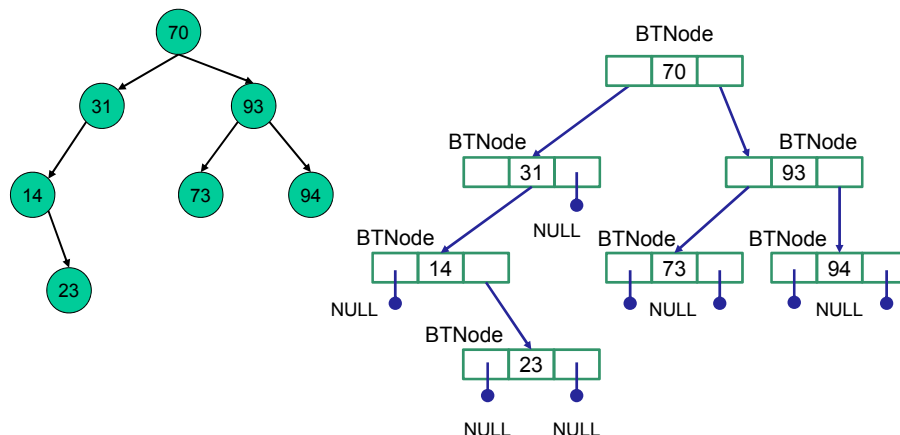
- Kiểu của item có thể là kí tự, chuỗi, cấu trúc,...

```
typedef struct _bnode {  
    int item;  
    struct _bnode *left;  
    struct _bnode *right;  
} BTreeNode;
```



1-19

Ví dụ cây nhị phân



1-20

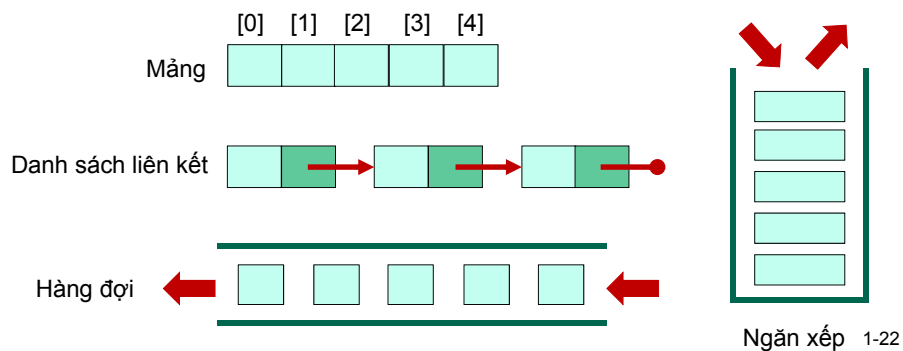
Cây

- ❑ Các cấu trúc dữ liệu không tuyến tính
- ❑ Cấu trúc dữ liệu cây
 - Cây nhị phân
- ❑ Cài đặt cây nhị phân
- ❑ **Duyệt cây nhị phân**
- ❑ Ứng dụng ví dụ

1-21

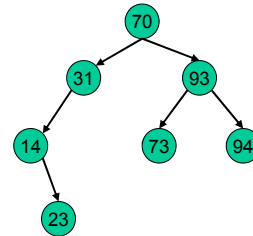
Duyệt cây

- ❑ Cho một cấu trúc dữ liệu tuyến tính và một phần tử cụ thể:
 - Rõ ràng xác định mỗi nút có một nút trước và một nút sau



Duyệt cây

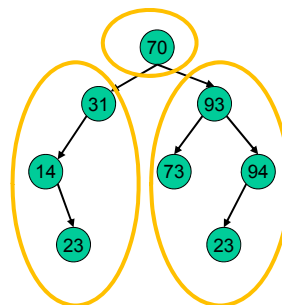
- ❑ Cho một cấu trúc dữ liệu tuyến tính và một phần tử cụ thể:
 - Rõ ràng xác định mỗi nút có một nút trước và một nút sau
- ❑ Cây là cấu trúc dữ liệu không tuyến tính
 - Lấy dữ liệu từ cây nhị phân như thế nào
 - Thứ tự duyệt qua các phần tử như thế nào? trái/trái/trái, rồi trái/trái/phải, rồi?
- ❑ Cần có cách để thăm mọi nút trên cây, không đi lặp lại



1-23

Duyệt cây

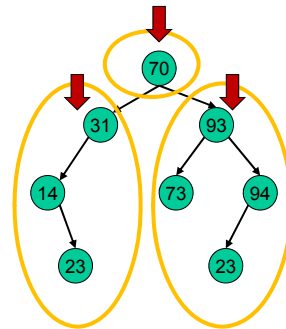
- ❑ Tại sao cần thực hiện duyệt cây
 - Duyệt cây là cơ sở cho các thao tác khác
- ❑ Thao tác rất phổ biến: Duyệt cây, tại mỗi nút, thực hiện một số công việc
- ❑ Ví dụ: Đếm số nút của cây
 - Tại mỗi nút N, kích thước của cây con đó = kích thước của cây con trái của N + kích thước của cây con phải của N + 1



1-24

Duyệt cây

- Duyệt cây là quá trình đệ quy
 - Đệ quy: Quá trình lặp tại các phần tử các thao tác giống nhau, chia bài toán thành các bài toán con
 - Tại mỗi nút: Thăm nút đó và hai nút con
- Đảm bảo thăm mọi nút và mỗi nút chỉ thăm một lần



1-25

Trong main(), gọi TreeTraversal(root)

TreeTraversal(Node N):

Visit N;
if (N has left child)

TreeTraversal(Node N):

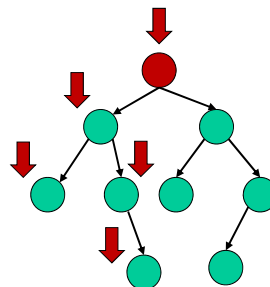
Visit N;
if (N has left child)

TreeTraversal(Node N):

Visit N;
if (N has left child)
TreeTraversal(LeftChild);
if (N has right child)

TreeTraversal(Node N):

Visit N;
if (N has left child)
TreeTraversal(LeftChild);
if (N has right child)
TreeTraversal(RightChild);
Return; // return to parent



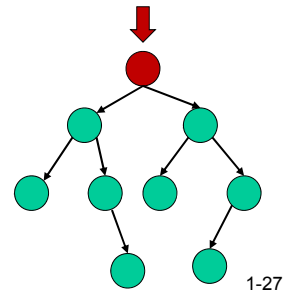
1-26

Duyệt cây

```
TreeTraversal(Node N):  
  Visit N;  
  if (N has left child)  
    TreeTraversal(LeftChild);  
  if (N has right child)  
    TreeTraversal(RightChild);  
  Return; // return to parent
```

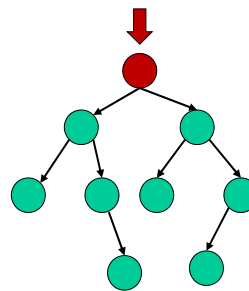
```
TreeTraversal2(Node N):  
  if N==NULL return;  
  Visit N;  
  TreeTraversal2(LeftChild);  
  TreeTraversal2(RightChild);  
  Return; // return to parent
```

- ❑ Mẫu 1: Cần kiểm tra sự tồn tại của nút con trái và phải trước khi duyệt
- ❑ Mẫu 2: Luôn thăm nút con, sau đó kiểm tra liên kết có NULL không
- ❑ Độ quy: Gọi lại trong chính hàm của nó
- ❑ Duyệt theo chiều sâu: Thăm các nút đi theo chiều sâu nhất có thể, trước khi quay lại và thăm nút bên



Cài đặt TreeTraversal2()

```
void TreeTraversal2(BTNode *cur) {  
  If (cur == NULL) return;  
  PrintNode(cur); //visit cur;  
  TreeTraversal2(cur->left);  
  TreeTraversal2(cur->right);  
}
```



Cây

- ❑ Các cấu trúc dữ liệu không tuyến tính
- ❑ Cấu trúc dữ liệu cây
 - Cây nhị phân
- ❑ Cài đặt cây nhị phân
- ❑ Duyệt cây nhị phân
- ❑ **Ứng dụng ví dụ**

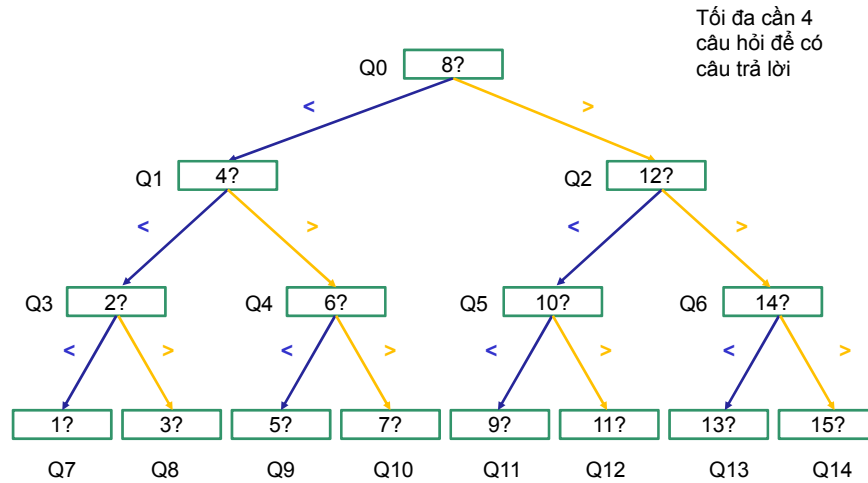
1-29

Trò chơi đoán số

- ❑ Người chơi nghĩ một số giữa 1 và 63
- ❑ Máy tính đưa ra câu hỏi, người chơi sẽ trả lời Yes/Too big/Too small
- ❑ Trò chơi kết thúc khi câu trả lời là Yes
- ❑ Ví dụ: khi người chơi nghĩ là 23, 6 câu hỏi để tìm được câu trả lời Yes là
 - 32? => too big
 - 6? => too small
 - 24? => too big
 - 20? => too small
 - 22? => too small
 - 23? => Yes!

1-30

Trò chơi đoán số [1, 15]



1-31

Trò chơi đoán số [1, 15]

```

void main() {
    int i; // 0-Yes; -1 -too big; 1-too small
    BTNode root, *cur;

    buildTree(&root, 8, 4);
    cur=&root;
    do {
        printf("is it %d?\n", cur->item);
        scanf("%d", &i);
        if (i== -1)
            cur=cur->left;
        else if (i==1)
            cur=cur->right;
    } while (i!=0 && cur!=NULL);
    if (i==0) printf("I guess it! %d\n", cur->item);
}
  
```

```

void buildTree(BTNode *r, int v, int h) {
    r->item=v;
    if (h>0) {
        r->left=malloc(sizeof(BTNode));
        r->right=malloc(sizeof(BTNode));
        buildTree(r->left, v-h, h/2);
        buildTree(r->right, v+h, h/2);
    }
    else {
        r->left=NULL;
        r->right=NULL;
    }
    return;
}
  
```

1-32

Cấu trúc dữ liệu và giải thuật

- Nội dung bài giảng được biên soạn bởi TS. Phạm Tuấn Minh.

1-33