Chương 3: Đệ quy và cây

Trịnh Anh Phúc [1]

[1] Bộ môn Khoa Học Máy Tính, Viện CNTT & TT,
Trường Đại Học Bách Khoa Hà Nội.

Ngày 21 tháng 9 năm 2020

Giới thiệu

1. Đệ quy
   - Thuật toán đệ qui
   - Một số ví dụ minh họa
2. Cây
   - Định nghĩa và thuật ngữ liên quan
     - Thứ tự và phép duyệt cây
   - Cây nhị phân và tính chất
   - Cài đặt
   - Ứng dụng
3. Tổng kết

Thuật toán đệ qui

Khái niệm đệ qui

Trong thực tế chúng ta thường gặp những đối tượng đệ quy bao gồm chính nó hoặc được định nghĩa bởi chính nó. Ta nói các đối tượng đó được xác định một cách đệ qui

- Điểm quân số
- Các hàm được định nghĩa đệ qui
- Định nghĩa đệ qui về cây ....

Hàm đệ qui (recursive functions)

Định nghĩa

Các hàm đệ qui được xác định bởi số nguyên không âm n theo sơ đồ

- **Bước cơ sở** (Basic step) : Xác định giá trị hàm tại thời điểm $n = 0$ hay $f(0)$
- **Bước đệ qui** (Recursive step) : Cho giá trị của hàm $f(k)$ tại $k \leq n$ đưa ra qui tắc tính giá trị của $f(n+1)$.

Hàm đệ qui (tiếp)

- VD1 :

$$\text{Bước cơ sở} : f(0) = 3 \quad n = 0$$
$$\text{Bước đệ quy} : f(n+1) = 2f(n) + 3 \;\; n > 0$$

- VD2 :

$$\text{Bước cơ sở} : f(0) = 1$$
$$\text{Bước đệ quy} : f(n+1) = f(n) \times (n+1)$$

1. Đệ quy
   - Thuật toán đệ qui
   - Một số ví dụ minh họa
2. Cây
   - Định nghĩa và thuật ngữ liên quan
     - Thứ tự và phép duyệt cây
   - Cây nhị phân và tính chất
   - Cài đặt
   - Ứng dụng
3. Tổng kết

Thuật toán đệ qui

Thuật toán đệ qui

Định nghĩa : *Thuật toán đệ qui là thuật toán tự gọi đến chính mình với đầu vào kích thước nhỏ hơn.*

- Tất nhiên việc sử dụng thủ tục đệ qui thích hợp với xử lý dữ liệu, hàm, cây được định nghĩa cũng một cách đệ qui như các ví dụ vừa nêu.
- Hầu hết các ngôn ngữ lập trình đều cho phép gọi đệ qui của hàm - lệnh gọi đến chính nó trong thân chương trình.

Thuật toán đệ qui

Cấu trúc của thuật toán đệ qui

Function RecAlg(input)
begin
    if (kích thước đầu vào là nhỏ nhất) **then**
        thực hiện bước cơ sở /* giải bài toán với kích thước cơ sở*/
    else
        RecAlg(input với đầu vào nhỏ hơn);/* các bước đệ qui*/
        Tổ hợp lời giải của các bài toán con để thu được **lời-giải**;
        return(lời-giải)
    endif
end;

1. Đệ quy
   - Thuật toán đệ qui
   - Một số ví dụ minh họa

2. Cây
   - Định nghĩa và thuật ngữ liên quan
     - Thứ tự và phép duyệt cây
   - Cây nhị phân và tính chất
   - Cài đặt
   - Ứng dụng

3. Tổng kết

Thuật toán đệ qui

Ví dụ 1 : Tính n!

Hàm f(n) = n! được định nghĩa đẹ qui như sau

- Bước cơ sở : f(0) = 0! = 1
- Bước đệ qui : f(n) = n f(n-1), với n>0

Hàm đệ qui viết bằng ngôn ngữ C

```
int Fact(int n){
    if(n==0) return 1;
    else return n*Fact(n-1);
}
```

Thuật toán đệ qui

Ví dụ 2 : Tính số Fibonacci

Dãy số Fibonacci đc định nghĩa như sau :

- Bước cơ sở : F(0) = 1, F(1) = 1;
- Bước đệ qui : F(n) = F(n-1) + F(n-2) với n $\geq$ 2

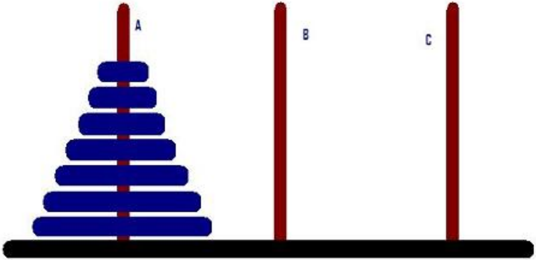
Hàm đệ qui viết bằng ngôn ngữ C

```
int FibRec(int n){
    if(n<=1) return 1;
    else return FibRec(n-1) + FibRec(n-2);
}
```

Thuật toán đệ qui

Ví dụ 3 : Bài toán tháp Hà Nội

Trò chơi tháp Hà Nội được trình bày như sau : Có 3 cọc A, B, C. Trên cọc A có một chồng gồm *n* cái đĩa đường kính giảm dần (xem hình vẽ). Cần phải chuyển chồng đĩa từ cọc A sang cọc C tuân theo qui tắc, mỗi lần chuyển một đĩa và chỉ được xếp đĩa có đường kính nhỏ lên trên đĩa có đường kính lớn hơn đồng thời được dùng cọc B làm cọc trung gian.



Thuật toán đệ qui

Ví dụ 3 : Bài toán tháp Hà Nội (tiếp)

Bài toán đặt ra là tìm cách chơi đòi hỏi số lần di chuyển đĩa ít nhất. Các lập luận sau đây được sử dụng để xây dựng thuật toán giải quyết bài toán đặt ra

- Nếu $n = 1$ thì ta chỉ việc chuyển đĩa cọc A sang cọc C
- Trong trường hợp $n \geq 2$ việc di chuyển đĩa gồm các bước đệ qui như sau
  1. chuyển $n - 1$ đĩa từ cọc A đến cọc B sử dụng cọc C làm trung gian. Bước này cũng phải thực hiện với số lần di chuyển nhỏ nhất, nghĩa là ta phải giải bài toán tháp Hà Nội với $n - 1$ đĩa.
  2. chuyển 1 đĩa đường kính lớn nhất từ cọc A đến cọc C.
  3. chuyển $n - 1$ đĩa từ cọc B đến cọc C - sử dụng cọc A làm trung gian. Bước này cũng phải thực hiện với số lần di chuyển nhỏ nhất, nghĩa là ta lại phải giải bài toán tháp Hà Nội với $n - 1$ đĩa.

Thuật toán đệ qui

Ví dụ 3 : Bài toán tháp Hà Nội (tiếp)

Trong trường hợp $n \geq 2$, hai bước nhỏ 1 và 3 cần số lần di chuyển ít nhất khi thực hiện hai bước này là $2 \times h_{n-1}$, do đó nếu gọi số lần di chuyển đĩa ít nhất là h_n, ta có công thức đệ qui sau

$$\begin{aligned} h_1 &= 1, \\ h_2 &= 2h_1 + 1, \ldots \\ h_n &= 2h_{n-1} + 1, n \geq 2 \end{aligned}$$

sử dụng phương pháp thế từng bước, ta có

$$h_n = 2^n - 1$$

như vậy độ phức tạp của thuật toán là hàm số mũ

Thuật toán đệ qui

Ví dụ 3 : Bài toán tháp Hà Nội (tiếp)

Mã giả của thuật toán đệ qui giải bài toán tháp Hà Nội như sau

```
Procedure HanoiTower(n,a,b,c)
    if (n=1) then
        <chuyển đĩa từ cọc A sang cọc C>
    else
        HanoiTower(n-1,A,C,B)
        HanoiTower(1,A,B,C)
        HanoiTower(n-1,B,A,C)
    endif
End
```

Định nghĩa đệ quy

Định nghĩa cây

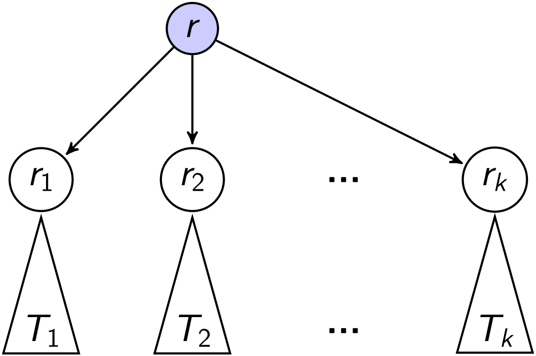
Cây bao gồm các nút, có một nút đặt biệt được gọi là nút gốc (root) và các cạnh nối các nút. Cây được định nghĩa đệ qui như sau

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

Định nghĩa đệ quy

Định nghĩa cây (tiếp)

Hình minh họa định nghĩa đệ qui của cây



Định nghĩa và các khái niệm

Các ứng dụng của kiểu dữ liệu trừu tượng cây

Cây trong ứng dụng thực tế

- Biểu đồ lịch thi đấu
- Cây gia phả
- Biều đồ phân cấp quản lý
- Cây thư mục quản lý file
- Cây biểu thức
- ....

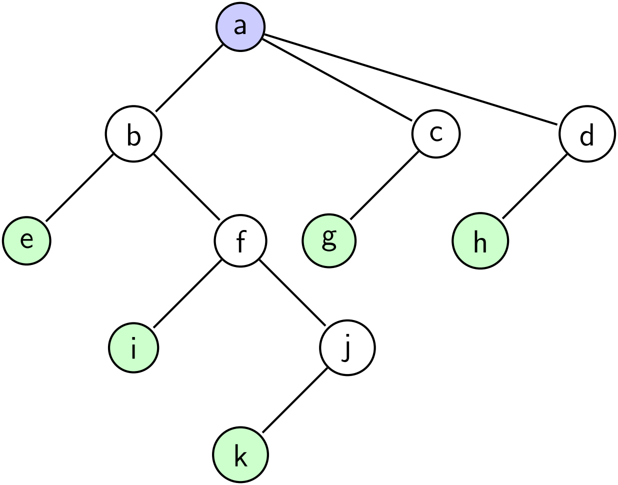
Cây

Các thuật ngữ chính

- Nút - node
- Gốc - root
- Lá - leaf
- Con - child
- Cha - parent
- Tổ tiên - ascentors
- Hậu duệ - descendants
- Anh em - sibling
- Chiều cao - hight
- Nút trong - internal node
- Đường đi - path

Cây

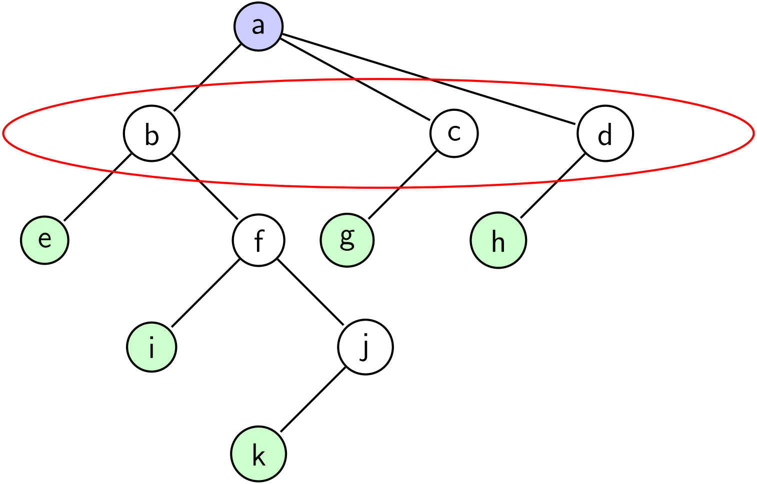
Phân loại các nút trong cây



Chú thích : Nút gốc mầu xanh thẫm, nút lá mầu xanh lá cây còn nút

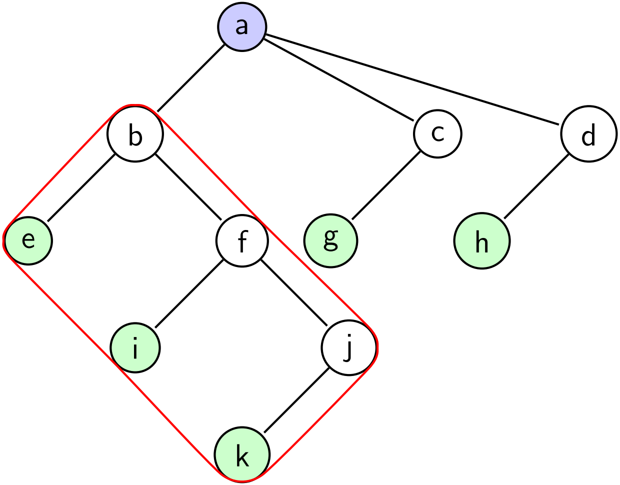
Cây

Các nút cùng cha gọi là các nút anh&em. Trong hình là ba nút b,c,d có cùng nút cha là a, được đánh dấu bởi hình elíp đỏ.



Cây

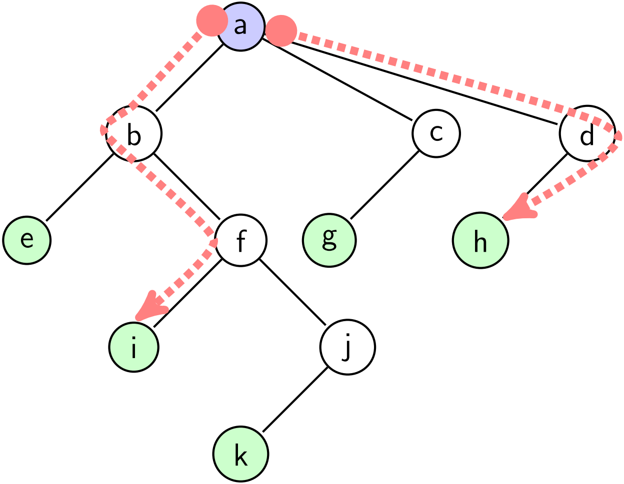
Cây con của nút gốc a,



Chú thích : Vòng tròn bao mầu đỏ chỉ ra một cây con của nút gốc a.

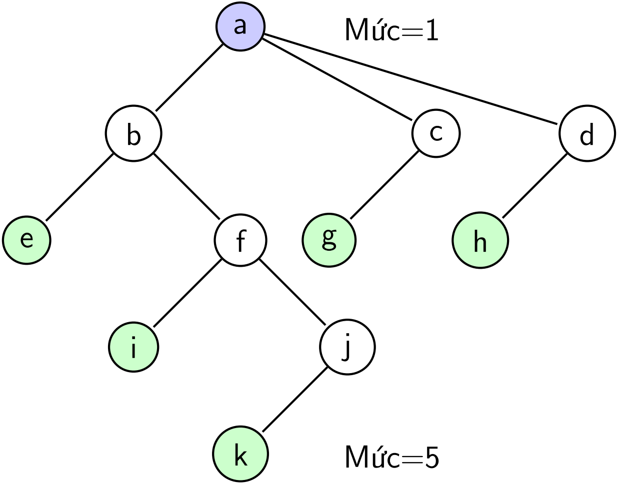
Cây

Đường đi trên cây từ nút gốc a đến các nút lá i và h (gạch nét dứt mầu đỏ). Đường thứ nhất {a,b,f,i} và đường thứ hai là {a,d,h}.



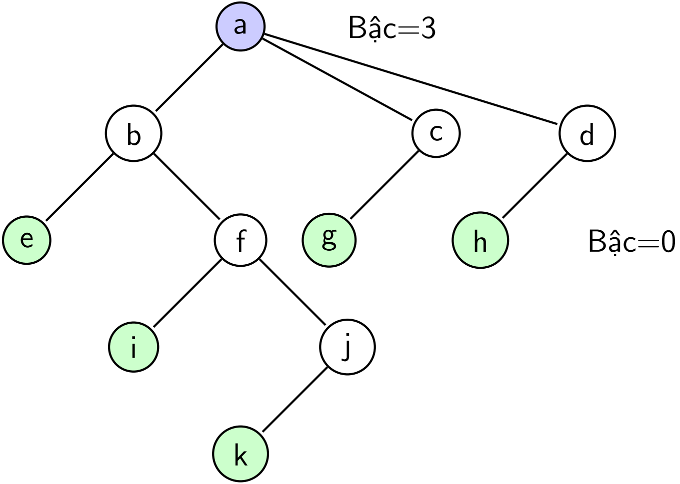
Cây

Độ cao của cây và độ sâu của cây. Do nút gốc có mức 1 nên nút lá xa nhất k có mức chính là độ cao và độ sâu của cây, vậy cây trên có độ cao là 5.



Cây

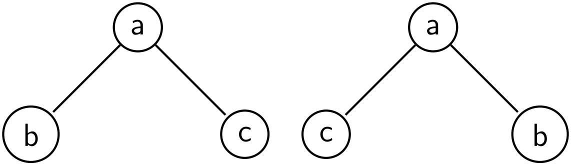
Bậc (degree) của một nút là số nút con của nó. Vậy nút gốc a có bậc 3 trong khi nút lá như h luôn có bậc 0.



Cây

Cây có thứ tự

Thứ tự của các nút trên cây : Các nút con của một nút thường được sắp xếp theo thứ tự từ **trái sang phải**



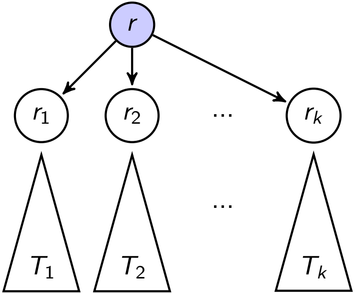
Như vậy rõ ràng hai cây trên **khác nhau** do thứ tự nút con của nút a là khác nhau. Hay nút b được xếp trước nút c trong cây bên trái, trong khi nó được xếp sau nút c trong cây bên phải.

Cây

Xếp thứ tự các nút

Có ba cách thông thường để xác định thứ tự các nút

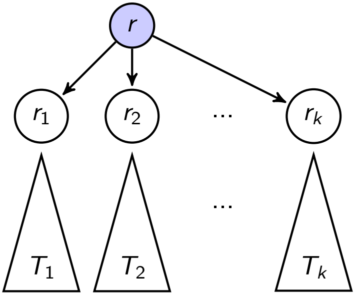
1. Thứ tự trước (Preorder)
2. Thứ tự giữa (Inorder)
3. Thứ tự sau (Postorder)



Cây

Thứ tự trước - Preorder traversal

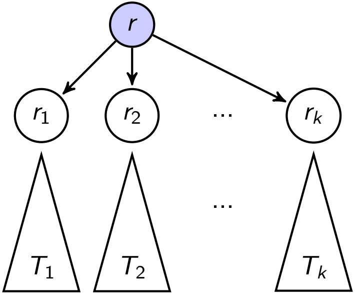
- Gốc r của cây
- tiếp đến là các nút của T_1 được duyệt theo thứ tự trước
- tiếp đến là các nút của T_2 được duyệt theo thứ tự trước...
- và cuối cùng là các nút của T_k được duyệt theo thứ tự trước



Cây

Thứ tự sau - Postorder traversal

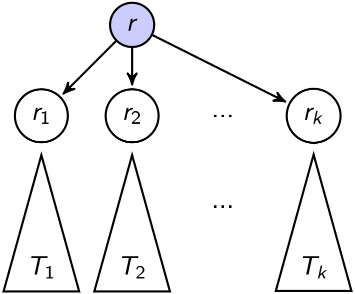
- các nút của cây T_1 theo thứ tự sau
- tiếp đến là các nút của T_2 được duyệt theo thứ tự sau...
- tiếp đến là các nút của T_k được duyệt theo thứ tự sau
- sau cùng là nút gốc r



Cây

Thứ tự giữa - Inorder traversal

- các nút của cây T_1 theo thứ tự giữa
- tiếp đến nút gốc r
- tiếp đến các nút của cây $T_2, \cdots, T_k$ mỗi nhóm nút được xét theo thứ tự giữa.



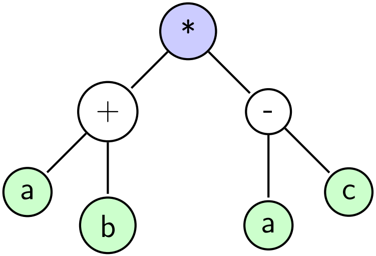
Cây

Cây có nhãn - labaled tree

Thông thường người ta gán cho mỗi nút nột nhãn hoặc một giá trị, cũng giống như ta 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 giá trị của nút đó. Ví dụ rõ nhất là cây biểu thức ...

Cây

Cây biểu thức - expression tree



Biểu thức : (a+b)*(a-c)

Qui tắc biểu diễn cây biểu thức là :

- Mỗi nút lá là một số hạng và chỉ gồm số hạng đó
- Mỗi nút trong được gán một phép toán. Với phép toán hai ngôi E_1 q E_2, ví dụ của q $= \{+,-,*,/\}$, thì cây con trái biểu diễn biểu thức E_1 còn cây con phải biểu diễn E_2.

1. Đệ quy
   - Thuật toán đệ qui
   - Một số ví dụ minh họa

2. Cây
   - Định nghĩa và thuật ngữ liên quan
     - Thứ tự và phép duyệt cây
   - **Cây nhị phân và tính chất**
   - Cài đặt
   - Ứng dụng

3. Tổng kết

Cây nhị phân

Cây nhị phân - binary tree

Định nghĩa : Cây nhị phân là cây mà mỗi nút có nhiều nhất là hai nút con. Vì mỗi nút chỉ có hai con nên ta sẽ gọi chúng là con trái và con phải. Chú ý là cây nhị phân giản lược so với cây tổng quát nên ta không cần xác định thứ tự các nút con.

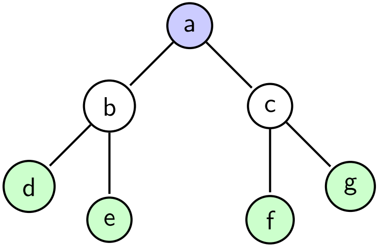
Tính chất của cây nhị phân

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

Cây nhị phân

Cây nhị phân đầy đủ - full binary tree

Định nghĩa : Cây nhị phân đầy đủ là cây nhị phân mà mỗi nút có đúng hai nút con đồng thời các nút lá cùng độ sâu.



Tính chất của cây nhị phân đầy đủ

- Cây nhị phân đầy đủ với độ sâu d có $2^d - 1$ nút.

Cây nhị phân

Cây nhị phân hoàn chỉnh - complete binary tree

Định nghĩa Cây nhị phân độ sâu d thỏa mãn :

- là cây nhị phân đầy đủ nếu không tính đến các nút ở độ sâu d, hay mức cao nhất.
- tất cả các nút tại độ sâu d lệch sang trái nhất có thể.

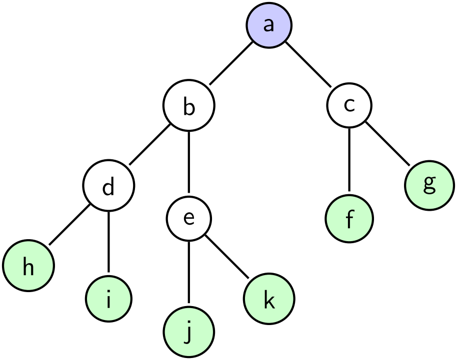
Tính chất

- Cây nhị phân hoàn chỉnh có độ sâu d thì số nút của nó nằm trong khoảng từ 2^{d-1} đến $2^d - 1$

Cây nhị phân

Cây nhị phân hoàn chỉnh (tiếp)

Ví dụ về cây nhị phân hoàn chỉnh



1. Đệ quy
   - Thuật toán đệ qui
   - Một số ví dụ minh họa
2. Cây
   - Định nghĩa và thuật ngữ liên quan
     - Thứ tự và phép duyệt cây
   - Cây nhị phân và tính chất
   - Cài đặt
   - Ứng dụng
3. Tổng kết

Cây nhị phân

Biểu diễn cây nhị phân

Để biểu diễn cây nhị phân trong máy tính, ta cũng có hai cách

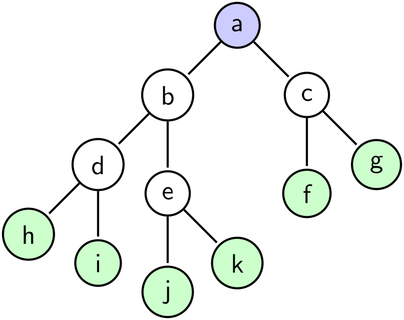
- sử dụng mảng
- sử dụng con trỏ

Khi biểu diễn cây nhị phân sử dụng mảng, ta làm tương tự như khi biểu diễn cây tổng quát. Tuy nhiên, trong trường hợp cây nhị phân hoàn chỉnh, sử dụng cách biểu diễn này ta có thể cài đặt hiệu quả nhiều phép toán trên cây, cả phép toán đối với nút con.

Cây nhị phân

Biểu diễn cây nhị phân hoàn chỉnh với mảng

Chú ý, các nút được đánh chỉ số trong mảng từ trên xuống dưới và từ trái qua phải. Với chỉ sô $i = 1, 2, \cdots, n$ với n là số nút trên cây



a	b	c	d	e	f	g	h	i	j	k

Cây nhị phân

Biểu diễn cây nhị phân hoàn chỉnh với mảng (tiếp)

Các phép toán trên cây nhị phân hoàn chỉnh biểu diễn bằng mảng $A[i]$ với chỉ số $i = 1, 2, \cdots, n$ với n là số nút trên cây.

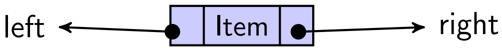
Để tìm	**Sử dụng**	**Hạn chế**
Con trái của $A[i]$	$A[2*i]$	$2*i \leq n$
Con phải của $A[i]$	$A[2*i+1]$	$2*i+1 \leq n$
Cha của $A[i]$	$A[i/2]$	$i > 1$
Gốc	$A[1]$	A khác rỗng
Nút $A[i]$ là lá	Đúng	$2*i > n$

Cây nhị phân

Biểu diễn cây nhị phân bằng con trỏ

Mỗi nút trong cây nhị phân sẽ gồm ba thành phần

- con trỏ đến con trái (left)
- phần tử chứa kiểu dữ liệu (item)
- con trỏ đến con phải (right)



Cài đặt trong C

```
typedef struct Tnode{
    DataType Item;
    struct Tnode *left;
    struct Tnode *right;
} treeNode;
```

Cây nhị phân

Biểu diễn cây nhị phân bằng con trỏ

Các phép toán cơ bản cây nhị phân có kiểu dữ liệu số nguyên

```
typedef struct Tnode{
    int Item;
    struct Tnode *left;
    struct Tnode *right;
}treeNode;
treeNode *makeTreeNode(int x);
int depth(treeNode *tree);
int countNodes(treeNode *tree);
void printPreorder(treeNode *tree);
void printPostorder(treeNode *tree);
void printInorder(treeNode *tree);
```

Cây nhị phân

Biểu diễn cây nhị phân bằng con trỏ với phép toán tạo nút mới

```
treeNode *makeTreeNode(int x){
    treeNode *newNode = NULL;
    newNode = (treeNode*)malloc(sizeof(treeNode));
    if(newNode==NULL){
        printf("Het bo nho \n");
        exit(1);
    }else{
        newNode->left = NULL;
        newNode->right = NULL;
        newNode->Item = x;
    }
    return newNode;
}
```

Cây nhị phân

Biểu diễn cây nhị phân bằng con trỏ với phép toán đếm số nút

```
int countNodes(treeNode *tree){
    if(tree==NULL) return 0;
    else{
        int ld = countNodes(tree->left);
        int rd = countNodes(tree->right);
        return 1+ld+rd;
    }
}
```

Cây nhị phân

Biểu diễn cây nhị phân bằng con trỏ với phép toán tính độ sâu

```
int depth(treeNode *tree){
    if(tree==NULL) return 0;
    int ld = depth(tree->left);
    int rd = depth(tree->right);
    return 1+(ld>rd ? ld : rd);
}
```

Cây nhị phân

Biểu diễn cây nhị phân bằng con trỏ với phép toán duyệt cây theo thứ tự trước

Duyệt đệ qui theo thứ tự trước

- Thăm nút hiện tại
- Thăm cây con trái theo thứ tự trước
- Thăm cây con phải theo thứ tự trước

Mã nguồn ngôn ngữ C

```
void printPreorder(treeNode *tree){
    if(tree!=NULL)
    {
        printf("%5d",tree->Item);
        printPreorder(tree->left);
        printPreorder(tree->right);
    }
}
```

Cây nhị phân

Biểu diễn cây nhị phân bằng con trỏ với phép toán duyệt cây theo thứ tự giữa

Duyệt đệ qui theo thứ tự giữa

- Thăm cây con trái theo thứ tự giữa
- Thăm nút hiện tại
- Thăm cây con phải theo thứ tự giữa

Mã nguồn ngôn ngữ C

```
void printInorder(treeNode *tree){
    if(tree!=NULL)
    {
        printInorder(tree->left);
        printf("%5d",tree->Item);
        printInorder(tree->right);
    }
}
```

Cây nhị phân

Biểu diễn cây nhị phân bằng con trỏ với phép toán duyệt cây theo thứ tự sau

Duyệt đệ qui theo thứ tự sau

- Thăm cây con trái theo thứ tự sau
- Thăm cây con phải theo thứ tự sau
- Thăm nút hiện tại

Mã nguồn ngôn ngữ C

```
void printPostrder(treeNode *tree){
    if(tree!=NULL)
    {
        printPostorder(tree->left);
        printPostorder(tree->right);
        printf("%5d",tree->Item);
    }
}
```

Cây nhị phân

Ứng dụng 1 : cây nhị phân biểu thức - expression binary tree

Cây biểu thức nhị phân trong đó :

- mỗi nút lá chứa một toán hạng
- mỗi nút trong chứa phép toán một ngôi
- Các cây con trái và các cây con phải chứa hai vế của biểu thức phép toán hai ngôi.

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

Cây nhị phân

Ứng dụng 1 : cây nhị phân biểu thức - expression binary tree (tiếp)

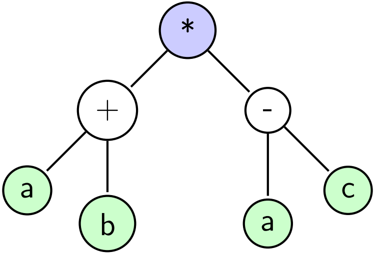
Duyệt cây nhị phân biểu thức có thể cho ta các biểu thức dưới dạng trung tố, tiền tố và hậu tố

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

Cây nhị phân

Ứng dụng 1 : cây nhị phân biểu thức (tiếp)

Ví dụ minh họa



- Biểu thức tiền tố : * + a b - a c
- Biểu thức trung tố : (a+b)*(a-c)
- Biểu thức hậu tố : a b + a c - *

Cây gọi đệ qui

Ứng dụng 2 : Cây gọi đệ qui

Đây là một công cụ *quan trọng* khi phân tích giải thuật đệ qui, cây gọi đệ qui được định nghĩa như sau

- Nút gốc r của cây là lần gọi đầu của giải thuật
- Nút lá của cây tương ứng bước cơ sở
- Nhánh cây từ nút cha $f(n+1)$ đến các nút con $f(k)$ với $k \leq n$ tương ứng bước gọi đệ qui của hàm $f(n+1)$

Cây gọi đệ qui (tiếp)

Ví dụ tính dãy số Fibonnacci

Dãy số Fibonacci đc định nghĩa đệ qui như sau :

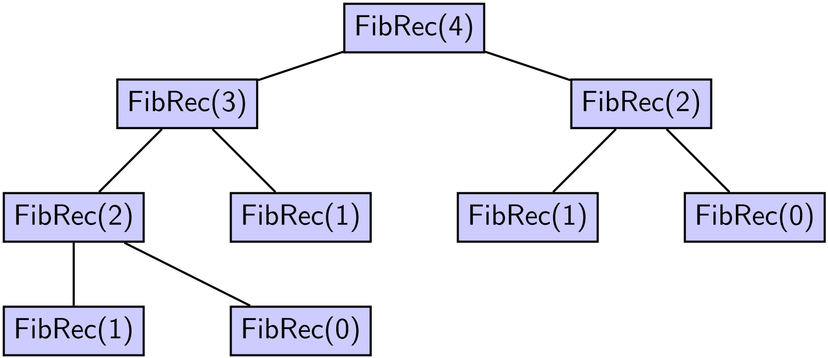
- Bước cơ sở : F(0) = 1, F(1) = 1;
- Bước đệ qui : F(n) = F(n-1) + F(n-2) với n $\geq$ 2

Hàm đệ qui viết bằng ngôn ngữ C

```
int FibRec(int n){
    if(n<=1) return 1;
    else return FibRec(n-1) + FibRec(n-2);
}
```

Cây gọi đệ qui (tiếp)

Ví dụ tính dãy số Fibonnacci (tiếp)



Cây gọi đệ qui tính số Fibonnacci với lần gọi đầu FibRec(4)

Tổng kết

- Khái niệm về đệ quy và thuật toán đệ quy
- Cấu trúc dữ liệu đệ quy là cây
- Cài đặt cây
- Ứng dụng