Mục lục

[Chương 1: Tổng quan về cấu trúc dữ liệu và giải thuật 3](#_Toc30432838)

[1.1. Tầm quan trọng của cấu trúc dữ liệu và giải thuật. 3](#_Toc30432839)

[1.1.1. Xây dựng cấu trúc dữ liệu 3](#_Toc30432840)

[1.1.2. Xây dựng cấu trúc giải thuật 3](#_Toc30432841)

[1.1.3. Mối quan hệ giữa cấu trúc dữ liệu và giải thuật 3](#_Toc30432842)

[1.2. Đánh giá cấu trúc dữ liệu và giải thuật 3](#_Toc30432843)

[1.2.1. Các tiêu chuẩn đánh giá cấu trúc dữ liệu và giải thuật 3](#_Toc30432844)

[1.2.2. Đánh giá độ phức tạp của thuật toán 3](#_Toc30432845)

[1.3. Kiểu dữ liệu 3](#_Toc30432846)

[Chương 2: kỹ thuật tìm kiếm 3](#_Toc30432847)

[2.1. Khái quát về tìm kiếm. 3](#_Toc30432848)

[2.2. Các giải thuật tìm kiếm nội. 3](#_Toc30432849)

[2.2.3. Tìm kiếm thuyến tính ( Linear Search ) 3](#_Toc30432850)

[2.2.4. Tìm nhị phân ( Binary Search ) 4](#_Toc30432851)

[2.3. Các giải thuật tìm kiếm ngoại ( Tìm kiếm trên tập tin ) 7](#_Toc30432852)

[Chương 3: kỹ thuật sắp xếp ( Sorting) 7](#_Toc30432853)

[3.1. Tổng quát về sắp xếp 7](#_Toc30432854)

[3.2. Các thuật toán sắp xếp 7](#_Toc30432855)

[3.2.1. Sắp xếp bằng phương pháp đổi chỗ (exchange sort ) 7](#_Toc30432856)

[3.2.2. Thuật toán sắp xếp chọn ( selection sort ) 10](#_Toc30432857)

[3.3. Các thuật toán sắp xếp ngoại ( Sắp xếp tập tin ) 12](#_Toc30432858)

[Chương 4: Danh sách ( List ) 12](#_Toc30432859)

[4.1. Khái niệm về danh sách 12](#_Toc30432860)

[4.2. Các phép toán trên danh sách. 12](#_Toc30432861)

[4.3. Danh sách đặc ( Condensed List ) 12](#_Toc30432862)

[4.4. Danh sách liên kết (Linked List ) 13](#_Toc30432863)

[4.5. Danh sách hạn chế 13](#_Toc30432864)

[4.5.1. Hàng đợi ( Queue ) 13](#_Toc30432865)

[4.5.2. Ngăn xếp ( Stack ) 13](#_Toc30432866)

[Chương 5: Cây (Tree) 13](#_Toc30432867)

[5.1. Khái niệm – biểu diễn cây 13](#_Toc30432868)

[5.1.1. Định nghĩa cây 13](#_Toc30432869)

[5.1.2. Một số khái niệm liên quan. 14](#_Toc30432870)

[5.1.3. Biểu diễn cây. 15](#_Toc30432871)

[5.2. Cây nhị phân. 15](#_Toc30432872)

[5.2.1. Định nghĩa. 15](#_Toc30432873)

[5.2.2. Cây nhị phân tìm kiếm. 16](#_Toc30432874)

[5.3. Cây cân bằng ( Balanced Tree ) 17](#_Toc30432875)

[Link chương trình sắp xếp và tìm kiếm. 18](#_Toc30432876)

[Tài liệu tham khảo 18](#_Toc30432877)

# Chương 1: Tổng quan về cấu trúc dữ liệu và giải thuật

## Tầm quan trọng của cấu trúc dữ liệu và giải thuật.

### Xây dựng cấu trúc dữ liệu

### Xây dựng cấu trúc giải thuật

### Mối quan hệ giữa cấu trúc dữ liệu và giải thuật

Cấu trúc dữ liệu + giải thuật = chương trình

## Đánh giá cấu trúc dữ liệu và giải thuật

### Các tiêu chuẩn đánh giá cấu trúc dữ liệu và giải thuật

* Cấu trúc dữ liệu và giải thuật phải tiết kiệm tài nguyên và bộ nhớ.
* Phản ánh đúng thực tế của bài toán.
* Phải dễ dàng trong việc thao tác dữ liệu.

### Đánh giá độ phức tạp của thuật toán

## Kiểu dữ liệu

# Chương 2: kỹ thuật tìm kiếm

## 2.1. Khái quát về tìm kiếm.

## 2.2. Các giải thuật tìm kiếm nội.

### 2.2.3. Tìm kiếm thuyến tính ( Linear Search )

Tìm kiếm tuyến tính còn được gọi là tìm kiếm tuần tự ( Sequential Search )

1. **Tư tưởng**

Lần lượt so sánh phần tử X của mảng M bắt đầu từ phần tử đầu tiên cho đến khi tìm được phần tử có giá trị X hoặc đã duyệt qua hết các phần tử của mảng M thì kết thúc.

1. **Thuật toán**

B1: k=1 //đặt index của mảng

B2: IF M[k] != X and k =< N

B2.1: k++

B2.2: lặp lại B2.

B3: IF K =< N

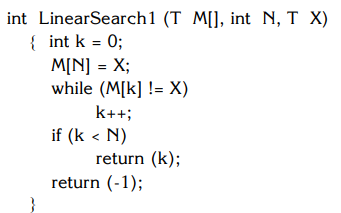
Tìm thấy tại vị trí k

B4: ELSE

Không tìm thấy phần tử có giá trị X

B5: End

1. **Cài đặt hàm thuật toán.**



### 2.2.4. Tìm nhị phân ( Binary Search )

1. **Tư tưởng**

Phạm vi tìm kiếm từ phần tử đầu tiên ( first = 1 ) đến phần tử cuối cùng ( last= N )

So sánh giá trị X với phần tử đúng ở giữa của dãy M là M[mid]

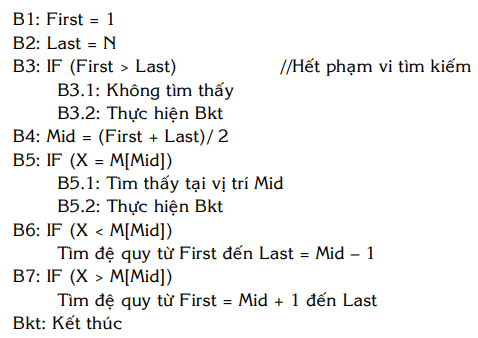
Nếu X = M[mid] => tìm thấy

Nếu X < M[mid] => Rút ngắn phạm vi tìm kiếm về nửa đầu của dãy M ( last = mid – 1 )

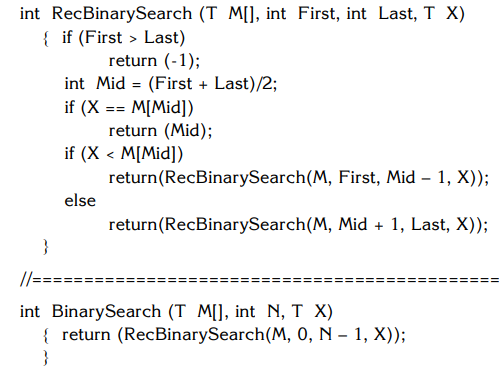
Nếu X > M[mid] => Rút ngắn phạm vi tìm kiếm về nửa sau của dãy M ( first = mid + 1 )

Lặp lại quá trình này đến khi chúng ta tìm thấy phần tử X hoặc phạm vi tìm kiếm của chúng ta không còn nữa ( last < first )

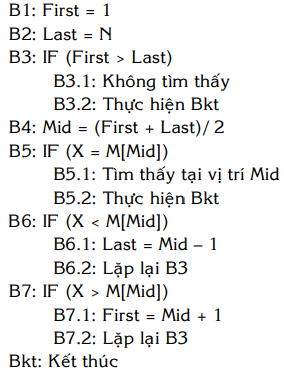
1. **Thuật toán đệ quy ( Recursion Algorithm )**



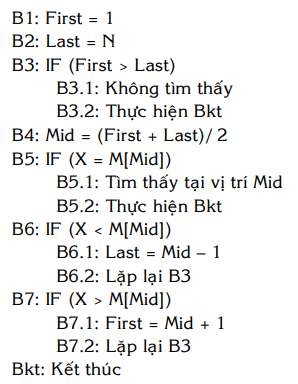
1. **Cài đặt thuật toán đệ quy**



1. **Thuật toán không đệ quy ( Non - Recursion Algorithm )**



1. **Cài đặt thuật toán không đệ quy**



Note:

* Thuật toán tìm kiếm nhị phân chỉ có thể vận dụng trong trường hợp dãy/mảng đã có thứ tự. Trong trường hợp tổng quát chúng ta chỉ có thể áp dụng thuật toán tìm kiếm tuần tự.
* Các thuật toán đệ quy có thể ngắn gọn song tốn kém bộ nhớ để ghi nhận mã lệnh chương trình ( mỗi lần gọi là đệ quy ) khi chạy chương trình, do vậy có thể làm cho chương trình chạy chậm lại. Do vậy, trong thực tế, khi viết chương trình, chúng ta có thể sử dụng thuật toán không đệ quy.

## 2.3. Các giải thuật tìm kiếm ngoại ( Tìm kiếm trên tập tin )

# Chương 3: kỹ thuật sắp xếp ( Sorting)

## Tổng quát về sắp xếp

## Các thuật toán sắp xếp

* Sắp xếp bằng phương pháp đếm ( counting sort )
* Sắp xếp bằng phương pháp đổi chỗ ( exchange sort )
* Sắp xếp bằng phương pháp lựa chọn ( selection sort )
* Sắp xếp bằng phương pháp chèn (insertion sort )
* Sắp xếp bằng phương pháp trộn ( merge sort )

Giả sử các phương pháp sắp xếp ở đây đều sắp xếp mảng theo thứ tự tăng dần

### Sắp xếp bằng phương pháp đổi chỗ (exchange sort )

Các thuật toán trong phần này sẽ tìm cách đổi chỗ các phần tử đứng sai vị trí ( so với mảng đã sắp xếp ) trong mảng M cho nhau để cuối cùng tất cả các phần tử trong mảng M đều về đúng vị trí mảng đã sắp xếp.

Các thuật toán sắp xếp bằng phương pháp đổi chỗ bao gồm :

* Thuật toán sắp xếp nổi bọt ( bubble sort )
* Thuật toán sắp xếp lắc (shaker sort )
* Thuật toán sắp xếp giảm độ tăng hay độ dài bước giảm dần ( shell sort )
* Thuật toán sắp xếp dựa trên sự phân hoạch ( quick sort )

Ở đây, chúng ta chủ yếu tập trung vào thuật toán sắp xếp nổi bọt và sắp xếp dựa trên sự phân hoạch

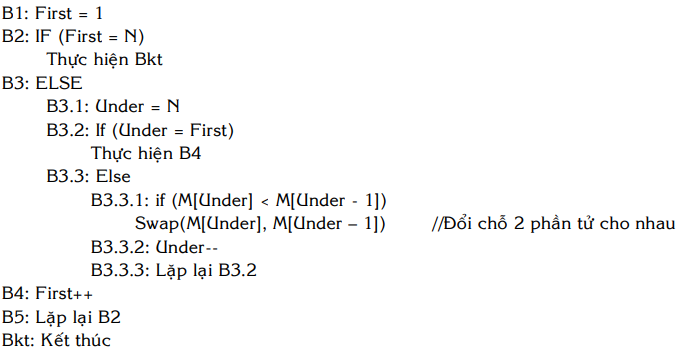
1. **Thuật toán sắp xếp nổi bọt.**

* **Tư tưởng:**

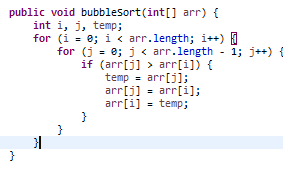
+ Đi từ đầu đến cuối mảng, trong quá trình đi, nếu phần tử nào ở dưới ( đứng phía sau ) nhỏ hơn phần tử đứng ngay trên ( trước ) thì nó theo nguyên tắc của bọt khí phần tử nhẹ sẽ bị “trồi” lên phía trên phần tử nặng ( 2 phần tử này sẽ đổi chỗ cho nhau ). Kết quả là phần tử nhỏ nhất ( nhẹ nhất ) sẽ được đưa lên ( trồi lên ) trên bề mặt ( đầu mảng ) rất nhanh.

+ sau mỗi lần đi chúng ta sẽ đưa được 1 phần tử trồi lên đúng chỗ. Do vậy sau N-1 lần đi, thì các phần tử trong mảng M có thứ tự tăng.

* **Thuật toán:**



* Cài đặt thuật toán:



* Thuật toán sắp xếp nổi bọt khá đơn giản và dễ hiểu, dễ cài đặt.
* Trong thuật toán sắp xếp nổi bọt, mỗi lần đi từ đầu đến cuối mảng và đầu mảng thì phần tử nhẹ được trồi lên trên rất nhanh trong khi phần tử nặng lại “chìm” xuống khá chậm chạp do không tận dụng được chiều đi xuống ( chiều từ đầu mảng đến cuối mảng )
* Thuật toán nổi bọt không phát hiện ra được các đoạn phần tử nằm hai đầu của mảng đã nằm đúng vị trí để có thể giảm bớt quãng đường đi trong mỗi lần đi.

1. **Thuật toán sắp xếp dựa trên sự phân hoạch**.

Thuật toán sắp xếp dựa trên sự phân hoạch ( hay còn gọi là thuật toán sắp xếp nhanh )

* **Tư tưởng:**

+ Phân hoạch dãy M thành 4 dãy con có thứ tự tương đối thỏa mãn điều kiện:

Dãy con thứ nhất ( đầu dãy M ) gồm các phần tử có giá trị nhỏ hơn giá trị trung bình của dãy M

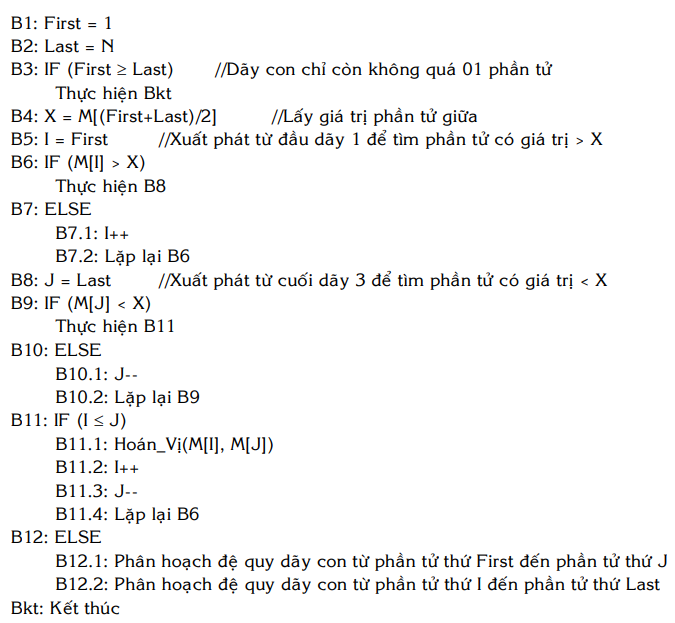
Dãy con thứ 2 (giữa dãy M) gồm các phần tử có giá trị trung bình bằng giá trị trung bình của dãy M

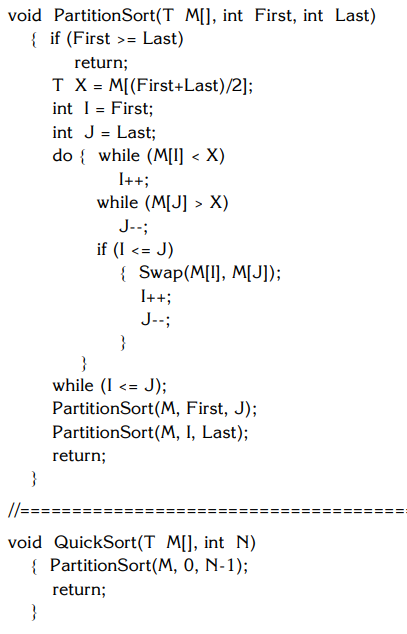
Dãy con thứ 3 ( cuối dãy M ) gồm các phần tử có giá trị trung bình lớn hơn giá trị trung bình của dãy M.

+ Nếu dãy con thứ nhất và dãy con thứ 3 có nhiều hơn 01 phần tử thì chúng ta tiếp tục phân hoạch đệ quy các dãy con này.

+ Việc tìm giá trị trung bình của dãy M hoặc tìm kiếm phần tử M có giá trị bằng giá trị trung bình của dãy M rất khó khăn và mất thời gian. Trong thực tế, chúng ta chọn một phần tử bất kì ( thường là phần tử đứng ở vị trí giữa ) trong các dãy phần tử cần phân hoạch. Phần tử con này được gọi là phần tử biên ( boundary element ). Các phần tử trong dãy con thứ nhất sẽ có giá trị nhỏ hơn giá trị phần tử biên và các giá trị của dãy con thứ 3 có giá trị lớn hơn giá trị của phần tử biên.

+ Việc phân hoạch một dãy được thực hiện bằng cách tìm các cặp phần tử đứng ở dãy 1 có giá trị lớn hơn giá trị phần tử giữa và phần tử đứng ở dãy 3 có giá trị nhỏ hơn giá trị phần tử ở giữa ) để đổi chỗ ( hoán vị ) cho nhau.

* **Thuật toán**
* C**ài đặt thuật toán.**



### Thuật toán sắp xếp chọn ( selection sort )

Thuật toán này sẽ tìm cách chọn các phần tử thỏa mãn điều kiện lựa chọn để đưa về đúng vị trí của phần tử đó, cuối cùng tất cả các phần tử trong M đề về đúng vị trí.

Các thuật toán sắp xếp chọn bao gồm:

* Thuật toán sắp xếp chọn trực tiếp (straight selection sort )
* Thuật toán dựa trên khối/heap hay sắp xếp trên cây ( heap sort )

ở đây chúng ta sử dụng thuật toán chọn trực tiếp.

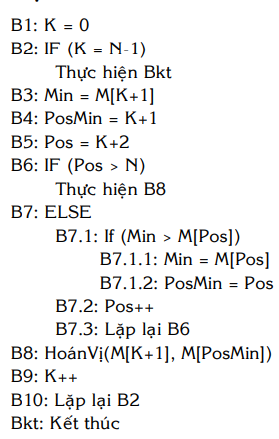
* **Tư tưởng:**

+ Ban đầu dãy có N phần tử chưa có thứ tự. Ta chọn phần tử có giá trị nhỏ nhất trong N phần tử chưa có thứ tự này để đưa lên đầu nhóm N phần tử.

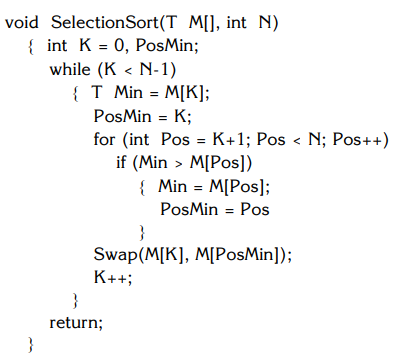
+ Sau lần thứ nhất lựa chọn phần tử nhỏ nhất và đưa lên đầu nhóm chúng ta còn lại N-1 phần tử đứng ở phía sau dãy M chưa có thứ tự. Chúng ta tiếp tục chọn phần tử nhỏ nhất của dãy N-1 phần tử chưa có thứ tự này để đưa lên đầu dãy N-1 phần tử,…Do vậy, sau N-1 lần lựa chọn phần tử nhỏ nhất, thì các phần tử của dãy M có thứ tự tăng.

+ Như vậy, thuật toán này chủ yếu chúng ta đi tìm kiếm giá trị nhỏ nhất trong nhóm N-K phần tử chưa có thứ tự đứng ở phía sau dãy M. Việc này đơn giản cho chúng ta vận dụng thuật toán tìm kiếm tuần tự.

* **Thuật toán:**



* **Cài đặt thuật toán:**



## Các thuật toán sắp xếp ngoại ( Sắp xếp tập tin )

* Sắp xếp bằng phương pháp trộn ( merge sort )
* Sắp xếp theo chỉ mục (index sort )

# Chương 4: Danh sách ( List )

## 4.1. Khái niệm về danh sách

* Danh sách là tập hợp các phần tử có kiểu dữ liệu xác định và giữa chúng có 1 mối liên hệ nào đó.
* Số phần tử của danh sách là chiều dài của danh sách. Một danh sách có chiều dài bằng 0 là một danh sách rỗng.

## 4.2. Các phép toán trên danh sách.

* Tạo mới danh sách
* Thêm một phần tử vào danh sách
* Tìm kiếm một phần tử trong danh sách
* Sửa danh sách
* Sắp xếp danh sách
* Tách một danh sách thành nhiều danh sách
* Nhập nhiều danh sách thành một danh sách
* Sao chép danh sách
* Hủy danh sách

## 4.3. Danh sách đặc ( Condensed List )

Là danh sách mà không gian bộ nhớ lưu trữ các phần tử được đặt liên tiếp nhau trong bộ nhớ.

## 4.4. Danh sách liên kết (Linked List )

Là tập hợp các phần tử mà giữa chúng có một sự kết nối với nhau thông qua vùng liên kết của chúng.

## 4.5. Danh sách hạn chế

Trong các thao tác trên sách sách không phải lúc nào cũng có thể thực hiện được tất cả mà nhiều khi các thao tác này bị hạn chế trong một số loại danh sách, đó là danh sách hạn chế.

Như vậy, danh sách hạn chế là danh sách mà các thao tác trên đó bị hạn chế trong một chừng mực nào đó tùy thuộc vào danh sách.

Có 2 loại danh sách hạn chế chủ yếu là:

* Hàng đợi ( Queue)
* Ngăn xếp ( Stack )

### 4.5.1. Hàng đợi ( Queue )

Hàng đợi là một danh sách mà trong đó thao tác thêm một phần tử vào danh sách sẽ được thực hiện ở đầu này và thao tác lấy ra một phần tử của danh sách sẽ được thực hiện ở đầu kia.

### 4.5.2. Ngăn xếp ( Stack )

Ngăn xếp là một danh sách mà thao tác thêm hoặc lấy một phần tử từ trong sách sách sẽ được thực hiện ở cùng 1 đầu.

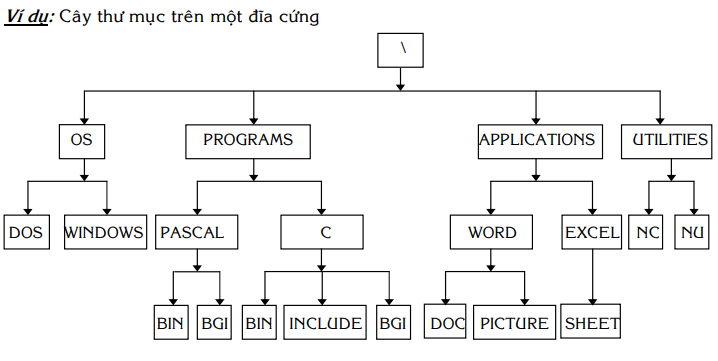
# Chương 5: Cây (Tree)

## Khái niệm – biểu diễn cây

### Định nghĩa cây

Cây là một tập hợp các phần tử ( các nút ) được tổ chức và có đặc điểm như sau :

* Hoặc là một tập hợp rỗng ( Cây rỗng )
* Hoặc là một tập hợp khác rỗng trong đó có một nút duy nhất được làm gốc, các nút còn lại được phân thành nhóm trong đó mỗi nhóm lại là một cây được gọi là cây con.
* Như vậy, một cây con có thể là một tập rỗng các nút hoặc cũng có thể là một tập hợp khác rỗng trong đó có một nút làm nút gốc cây con.



## Một số khái niệm liên quan.

1. **Bậc của nút.**

Là số cây con của nút đó

1. **Bậc của một cây.**

Là bậc lớn nhất của cây đó.

Cây có bậc N gọi là N-phân ( N-tree )

1. **Nút gốc.**

Là nút không phải là nút gốc của cây con nào trong cây.

1. **Nút kết thúc.**

Hay là nút lá, là nút có bậc bằng 0 ( nút không có cây con )

1. **Nút trung gian.**

Hay là nút giữa, là nút không phải là nút gốc, cũng không phải là nút kết thúc

1. **Mức của một nút.**

Bằng mức của nút gốc cây con chứa nó + 1, trong đó mức của nút gốc bằng 1.

1. **Chiều cao hay chiều sâu của một cây.**

Chiều cao của một cây hay chiều sâu của một cây bằng mức cao nhất của các nút trong cây.

1. **Nút trước và nút sau của một nút.**

Nút T được gọi là nút trước của nút S nếu cây con có gốc T chứa cây con có gốc S. Khi đó, nút S được gọi là nút sau của nút T.

1. **Nút cha và nút con của một nút.**

Nút B được gọi là nút cha của nút C nếu nút B là nút trước của nút C và mức của nút C lớn hơn mức của nút B 1 mức. Khi đó nút C được gọi là nút con của nút B.

1. **Chiều dài đường đi của một nút.**

Chiều dài đường đi của một nút là số đỉnh ( số nút ) tính từ nút gốc đi đến nút đó.

Như vậy, chiều dài đường đi của nút gốc luôn bằng 1, chiều dài đường đi tới một nút bằng chiều dài đường đi của nút cha nó cộng thêm 1.

1. **Chiều dài đường đi của một cây.**

Là tổng tất cả các chiều dài đường đi của một cây đó.

1. **Rừng.**

Là tập hợp các cây.

Như vậy, một cây khi mất nút gốc sẽ trở thành rừng.

### Biểu diễn cây.

**Có nhiều cách biểu diễn cây :**

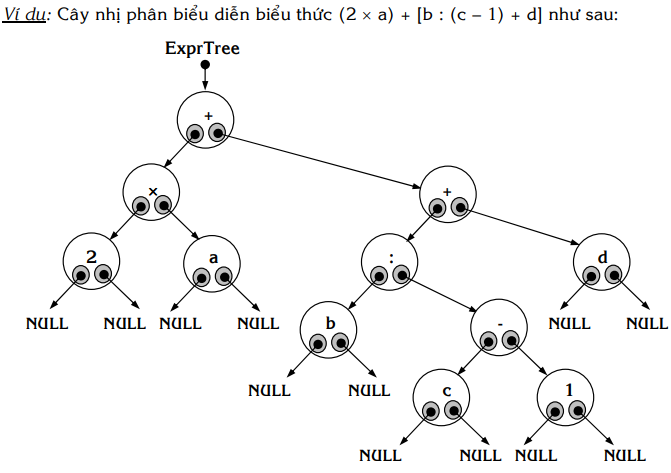
* Sử dụng đồ thị: như ví dụ
* Sử dụng giản đồ tập hợp.
* Sử dụng dạng phân cấp chỉ số : như menu bài viết..
* ….

**Biểu diễn cây trong bộ nhớ máy tính:**

## Cây nhị phân.

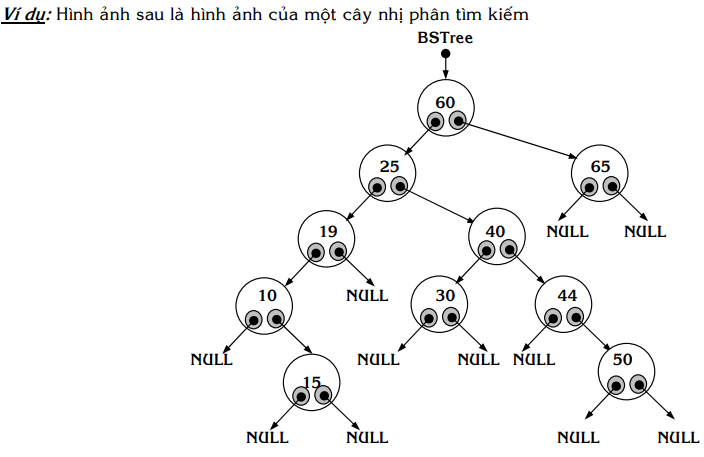
### Định nghĩa.

Là cây có bậc bằng 2.



### Cây nhị phân tìm kiếm.

Là cây nhị phân có thành phần khóa của mọi nút lớn hơn thành phần khóa của tất cả các nút trong cây con trái của nó và nhỏ hơn thành phần khóa của tất cả các nút trong cây con phải của nó.



## Cây cân bằng ( Balanced Tree )

1. **Định nghĩa**

* **Cây cân bằng tương đối:** Cây cân bằng tương đối là một cây nhị phân thỏa mãn điều kiện là đối với mọi nút của cây thì chiều cao của cây con trái và chiều cao của cây con phải của nút đó kém hơn nhau không quá 1. Cây cân bằng tương đối còn được gọi là cây AVL ( **AVL-tree** )
* **Cây cân bằng hoàn toàn:** Là một cây nhị phân thỏa mãn điều kiện là đối với mọi nút của cây thì số nút ở cây con trái và số nút ở cây con phải của nút đó hơn kém nhau không quá 1.
* Như vậy, cây cân bằng hoàn toàn chắc chắn là một cây cân bằng tương đối.

1. **Cấu trúc dữ liệu của cây cân bằng.**

# Link chương trình sắp xếp và tìm kiếm.

# Tài liệu tham khảo

<http://www.e-ptit.edu.vn/wp-content/uploads/2019/08/Gi%C3%A1o-tr%C3%ACnh-c%E1%BA%A5u-tr%C3%BAc-d%E1%BB%AF-li%E1%BB%87u-v%C3%A0-gi%E1%BA%A3i-thu%E1%BA%ADt.pdf>