**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN THỰC TẬP CƠ SỞ**

# CÀI ĐẶT CÁC THUẬT TOÁN TÌM KIẾM VÀ SẮP XẾP TRÊN MẢNG MỘT CHIỀU THEO HƯỚNG ĐỐI TƯỢNG

Giảng viên hướng dẫn **: TS. NGUYỄN THỦY THU TRANG**

Sinh viên thực hiện **: TRẦN QUỐC TRÍ**

Lớp **: 60.CNTT-2**

MSSV **: 60137903**

**NIÊN KHÓA: 2018 – 2022**

Mục Lục

**1. GIỚI THIỆU3**

**1.1**. Ý tưởng ban đầu3

**1.2.** Mục đích của bài nghiên cứu này3

**1.3.** Yêu cầu của đề tài4

**2. NỘI DUNG CHÍNH4**

2.1. Các thuật toán tìm kiếm cơ bản4

**2.1.1**. Thuật toán Bubble - Sort5

**2.1.2.** Thuật toán Selection - Sort5

**2.1.3.** Thuật toán Insertion - Sort6

**2.1.4**. Thuật toán Merge - Sort7

**2.1.5.** Thuật toán Quick - Sort9

**2.2.** Các thuật toán sắp xếp cơ bản10

**2.1.4.** Thuật toán Binary Search11

**2.1.4.** Thuật toán Linear Search12

**3. TỔNG KẾT13**

**3.1.** Hạn chế13

**3.2.** Tài liệu tham khảo13

1. **GIỚI THIỆU:**
   1. **Về ý tưởng ban đầu:**

Ngày nay, với sự phát triển nhanh đến chóng mặt của công nghệ máy tính, các công cụ tìm kiếm hiện đại như Google, Yahoo, Bing, mặc nhiên trở thành một phần không thể thiếu trong việc sử dụng máy tính nói chung và mạng internet nói chung. Với các thuật toán có tính tốt ưu và hiệu quả làm việc cực lớn, những công cụ này nhanh chóng mang lại hàng triệu kết quả tìm kiếm dựa trên từ khóa và sắp xếp các kết quả một cách khoa học dựa trên mức độ liên quan. Tuy nhiên các thuật toán này thường là độc quyền và được giữ bí mật cũng như sữ dụng làm công cụ kinh doanh cho các tập đoàn lớn. Do đó việc tìm hiệu chúng cho mục đích học tập là điều không thể, ngoài ra các thuật toán này là kết quả nghiên cứu của các chuyên gia hàng đầu cho nên dĩ nhiên không phù hợp với học sinh/ sinh viên để nghiên cứu. Do đó, cho mục đích học tập, ta nên bắt đầu tìm hiểu từ những thuật toán đơn giản nhất, đưa ra cái nhìn khái quát về các thuật toán tìm kiếm /sắp xếp, từ đó làm nền tẳng cho nhưng nghiên cứu chuyên sâu sau này.

* 1. **Mục đích của bài nghiên cứu này:**

Bài nghiên cứu này đưa ra lý thuyết và mẫu chương trình thực hiện các thuật toán tìm kiếm/ sắp xếp đơn giản nhất nhằm để người đọc có cái nhìn cụ thể về giải thuật cũng như tính ứng dụng của các thuật toán để làm nền tảng cho những nghiên cứu cao hơn sau này.

* 1. **Yêu cầu cụ thể của đề tài:**

**Mức 1:**

* + - 1. Trình bày lại lý thuyết đã được học.
      2. Sử dụng ngôn ngữ C++ với phương pháp lập trình hướng đối tượng để cài đặt các thuật toán tìm kiếm và sắp xếp trên mảng một chiều.
      3. Dữ liệu nhập có các lựa chọn sau:

- Nhập từ bàn phím

- Lấy giá trị ngẫu nhiên

- Nhập từ file

Dữ liệu xuất:

- Xuất ra màn hình

- Lưu vào file.

**Mức 2:** Mô phỏng bằng đồ họa.

Kết thúc thực tập: Viết thu hoạch ít nhất 30 trang theo mẫu đồ án tốt nghiệp, chương trình thực hiện.

1. **NỘI DUNG CHÍNH:**

**2.1. Các thật toán tìm kiếm cơ bản:**

**2.1.1. Thuật toán Buble- Sort:**

1. **Ý tưởng thuật toán:**

- Thuật toán buble – sort thực hiện sắp xếp một dãy số theo bằng cách lần lượt chọn một cặp số từ một đầu của dãy số đến đầu kia, nếu hai số này không theo thứ tự đã đề ra ta đổi vị trí của chúng cho nhau.

- Thực hiện lần lượt các bước trên từ đầu đến kết thúc của dãy số cho đến khi không còn số nào sai vị trí ta được 1 dãy số hoàn chỉnh.

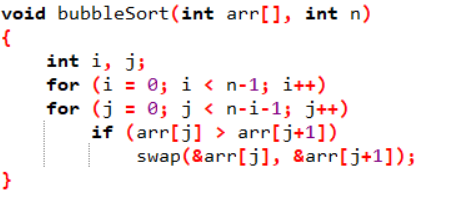
**B. Ví dụ thuật toán:**

- Giả sử chúng ta cần sắp xếp dãy số [5 1 4 2 8] này tăng dần.  
**Lần lặp đầu tiên:**  
( **5** **1** 4 2 8 ) –> ( **1** **5** 4 2 8 ), Ở đây, thuật toán sẽ so sánh hai phần tử đầu tiên, và đổi chỗ cho nhau do 5 > 1.  
( 1 **5** **4** 2 8 ) –>  ( 1 **4** **5** 2 8 ), Đổi chỗ do 5 > 4  
( 1 4 **5** **2** 8 ) –>  ( 1 4 **2** **5** 8 ), Đổi chỗ do 5 > 2  
( 1 4 2 **5** **8** ) –> ( 1 4 2 **5** **8** ), Ở đây, hai phần tử đang xét đã đúng thứ tự (8>5), vậy ta không cần đổi chỗ.

**Lần lặp thứ 2:**  
( **1** **4** 2 5 8 ) –> ( **1** **4** 2 5 8 )  
( 1 **4** **2** 5 8 ) –> ( 1 **2** **4** 5 8 ), Đổi chỗ do 4 > 2  
( 1 2 **4** **5** 8 ) –> ( 1 2 **4** **5** 8 )  
( 1 2 4 **5** **8** ) –>  ( 1 2 4 **5** **8** )  
Bây giờ, dãy số đã được sắp xếp, Nhưng thuật toán của chúng ta không nhận ra điều đó ngay được. Thuật toán sẽ cần thêm một lần lặp nữa để kết luận dãy đã sắp xếp khi và khi khi nó đi từ đầu tới cuối mà không có bất kỳ lần đổi chỗ nào được thực hiện.

**Lần lặp thứ 3:**  
( **1** **2** 4 5 8 ) –> ( **1** **2** 4 5 8 )  
( 1 **2** **4** 5 8 ) –> ( 1 **2** **4** 5 8 )  
( 1 2 **4** **5** 8 ) –> ( 1 2 **4** **5** 8 )  
( 1 2 4 **5** **8** ) –> ( 1 2 4 **5** **8** )

**C. Hàm thực hiện thuật toán trên ngôn ngữ C++**



**2.1.2. Thuật toán Selection – Sort:**

**A. Ý tưởng thuật toán**

**-** Thuật toán sắp xếp chọn sẽ sắp xếp một mảng bằng cách đi tìm phần tử có giá trị nhỏ nhất(giả sử với sắp xếp mảng tăng dần) trong đoạn đoạn chưa được sắp xếp và đổi cho phần tử nhỏ nhất đó với phần tử ở đầu đoạn chưa được sắp xếp(không phải đầu mảng). Thuật toán sẽ chia mảng làm 2 mảng con.

1. Một mảng con đã được sắp xếp
2. Một mảng con chưa được sắp xếp

- Tại mỗi bước lặp của thuật toán, phần tử nhỏ nhất ở mảng con chưa được sắp xếp sẽ được di chuyển về đoạn đã sắp xếp.

**B. Ví dụ thuật toán**

- Tìm phần tử nhỏ nhất trong trong arr[0...4] và đổi chỗ nó với phần tử đầu tiên

**[1] 24 15 22 62**

- Tìm phần tử nhỏ nhất trong trong arr[1...4] và đổi chỗ nó với phần tử đầu tiên của arr[1...4]

**1 [15] 24 22 62**

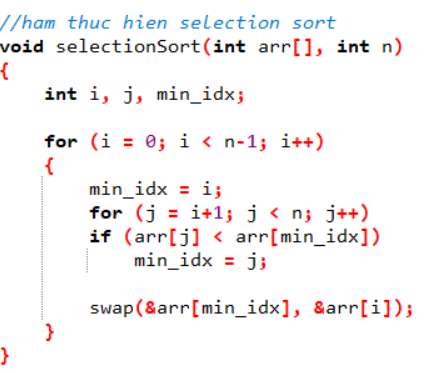
- Tìm phần tử nhỏ nhất trong trong arr[2...4] và đổi chỗ nó với phần tử đầu tiên của arr[2...4]

**1 15 [22] 24 62**

 - Tìm phần tử nhỏ nhất trong trong arr[3...4] và đổi chỗ nó với phần tử đầu tiên của arr[3...4]

**11 12 22 [24] 62**

**C. Chương trình thực hiện thuật toán trên ngôn ngữ C++**



**-Chú ý:** Ở đây, hàm swap được định nghĩa là hàm dùng để đổi vị trí của biến min\_idx với phần tử arr[j] trong trường hợp phần tử arr[j] của mảng có giá trị bé hơn giá trị của min\_idx hiện tại.

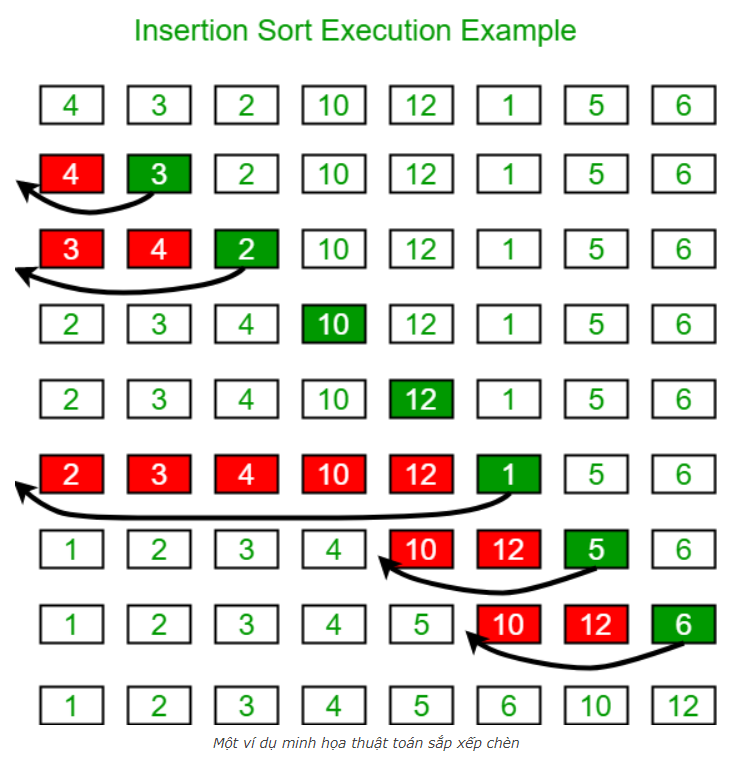
**2.1.3. Thuật toán Insertion-Sort:**

**A. Ý tưởng thuật toán:**

**-** Thuật toán sắp xếp chèn thực hiện sắp xếp dãy số theo cách duyệt từng phần tử và chèn từng phần tử đó vào đúng vị trí trong mảng con(dãy số từ đầu đến phần tử phía trước nó) đã sắp xếp sao cho dãy số trong mảng sắp đã xếp đó vẫn đảm bảo tính chất của một dãy số tăng dần.

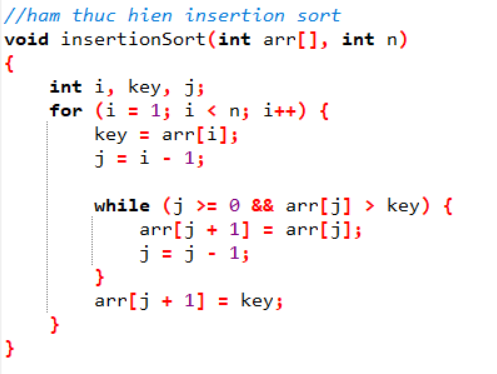
1. Khởi tạo mảng với dãy con đã sắp xếp có k = 1 phần tử(phần tử đầu tiên, phần tử có chỉ số 0)
2. Duyệt từng phần tử từ phần tử thứ 2, tại mỗi lần duyệt phần tử ở chỉ số i thì đặt phần tử đó vào một vị trí nào đó trong đoạn từ [0…i] sao cho dãy số từ [0…i] vẫn đảm bảo tính chất dãy số tăng dần. Sau mỗi lần duyệt, số phần tử đã được sắp xếp k trong mảng tăng thêm 1 phần tử.
3. Lặp cho tới khi duyệt hết tất cả các phần tử của mảng.

**B. Ví dụ thuật toán:**



**-** Hàng đầu tiên mô phỏng trạng thái ban đầu của mảng(dãy số chưa sắp xếp). Từ hàng thứ 2 trở đi, ta tìm chèn số đang xét vào vị trí thích hợp để đảm bảo dãy số vẫn tăng dần. Và khi lặp hết tất cả các số trong mảng, ta có trạng thái đã sắp xếp ở hàng cuối cùng.

**C. Hàm thực hiện thuật toán trên ngôn ngữ C++:**



**2.1.4. Thuật toán Merge – Sort:**

**A. Ý tưởng thuật toán:**

- Merge sort là một thuật toán chia để trị. Thuật toán này chia mảng cần sắp xếp thành 2 nửa. Tiếp tục lặp lại việc này ở các nửa mảng đã chia. Sau cùng gộp các nửa đó thành mảng đã sắp xếp. Hàm merge() được sử dụng để gộp hai nửa mảng. Hàm merge(arr, l, m, r) là tiến trình quan trọng nhất sẽ gộp hai nửa mảng thành 1 mảng sắp xếp, các nửa mảng là arr[l…m] và arr[m+1…r] sau khi gộp sẽ thành một mảng duy nhất đã sắp xếp.

**B. Ví dụ thuật toán:**

**-** Giả sử ta có 2 mảng con lần lượt là:

arr1 = [1 9 10 10] , n1 = 4 (nl: chiều dài của mảng con)

arr2 = [3 5 7 9], n2 = 4

sort\_arr = [] // Mảng lưu lại tiến trình gộp

- Khởi tạo i = 0, j = 0 tương ứng là chỉ số bắt đầu của arr1 và arr2

- Xét thấy arr1[i] < arr2[j] => chèn arr1[i] vào cuối mảng sort\_arr, tăng i lên 1 đơn vị

=> sort\_arr = [1], i = 1

- Xét thấy arr1[i] > arr2[j] => chèn arr2[j] vào cuối mảng sort\_arr, tăng j lên 1 đơn vị

=> sort\_arr = [1, 3], i = 1, j = 1

- Xét thấy arr1[i] > arr2[j] => chèn arr2[j] vào cuối mảng sort\_arr, tăng j lên 1 đơn vị

=> sort\_arr = [1, 3, 5], i = 1, j = 2

- Xét thấy arr1[i] > arr2[j] => chèn arr2[j] vào cuối mảng sort\_arr, tăng j lên 1 đơn vị

=> sort\_arr = [1, 3, 5, 7], i = 1, j = 3

- Xét thấy arr1[i] = arr2[j] => chèn arr1[i] hoặc arr2[j] vào cuối mảng sort\_arr

Giả sử tôi chọn arr1, tăng i lên 1 đơn vị

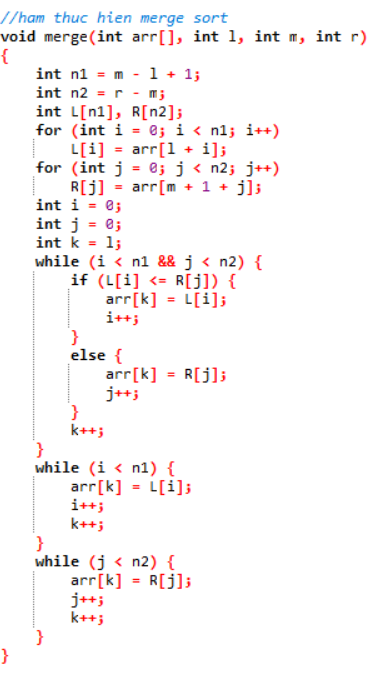
=> sort\_arr = [1, 3, 5, 7, 9], i = 2, j = 3

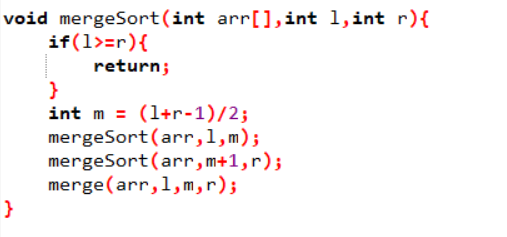
- Xét thấy arr1[i] > arr2[j] => chèn arr2[j] vào cuối mảng sort\_arr, tăng j lên 1 đơn vị

=> sort\_arr = [1, 3, 5, 7, 9, 9], i = 1, j = 4

Do j >= n2, tiếp tục tăng i chừng nào i < n1 thi thêm phần tử ở arr1[i]ư vào sort\_arr.

Sau cùng ta được mảng đã sắp xếp là sort\_arr = [1, 3, 5, 7, 9, 9, 10, 10]

**C. Hàm thực hiện thuật toán trên ngôn ngữ C++:** 

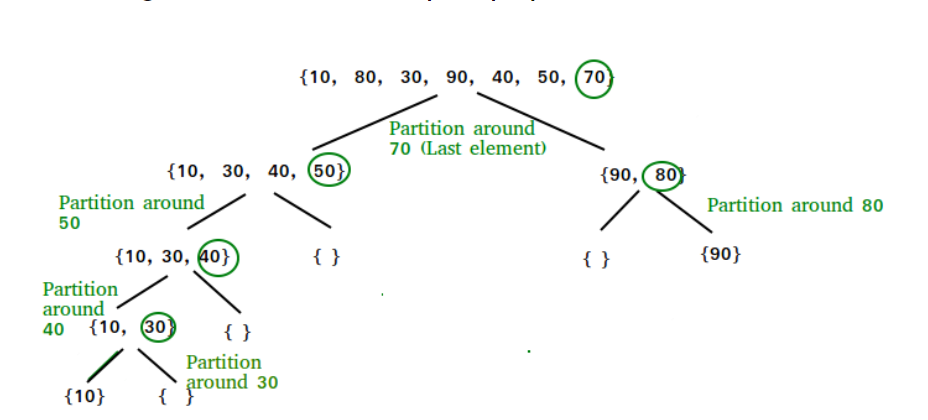


**2.1.5. Thuật toán Quick – Sort:**

**A. Ý tưởng thuật toán:**

**-** Giống như Merge sort, thuật toán sắp xếp quick sort là một thuật toán chia để trị( Divide and Conquer algorithm). Nó chọn một phần tử trong mảng làm điểm đánh dấu(pivot). Thuật toán sẽ thực hiện chia mảng thành các mảng con dựa vào pivot đã chọn. Việc lựa chọn pivot ảnh hưởng rất nhiều tới tốc độ sắp xếp. Nhưng máy tính lại không thể biết khi nào thì nên chọn theo cách nào. Dưới đây là một số cách để chọn pivot thường được sử dụng:

1. Luôn chọn phần tử đầu tiên của mảng.
2. Luôn chọn phần tử cuối cùng của mảng. (Được sử dụng trong bài viết này)
3. Chọn một phần tử random.
4. Chọn một phần tử có giá trị nằm giữa mảng(median element).

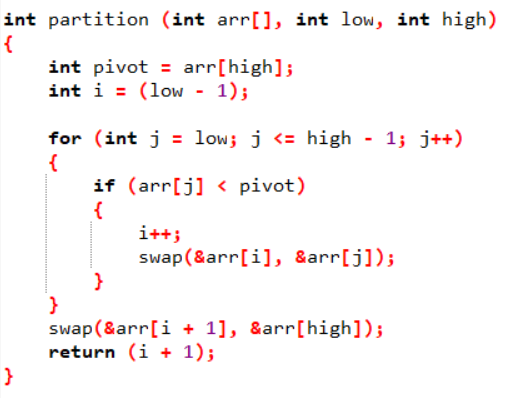


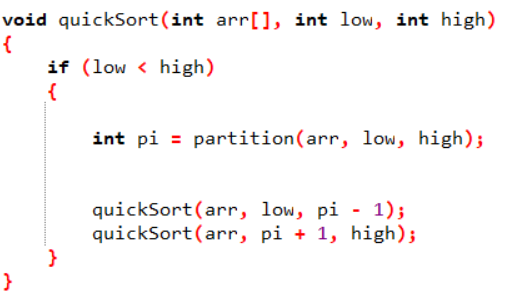
**-** Mấu chốt chính của thuật toán quick sort là việc phân đoạn dãy số (Xem hàm partition()). Mục tiêu của công việc này là: Cho một mảng và một phần tử x là pivot. Đặt x vào đúng vị trí của mảng đã sắp xếp. Di chuyển tất cả các phần tử của mảng mà nhỏ hơn x sang bên trái vị trí của x, và di chuyển tất cả các phần tử của mảng mà lớn hơn x sang bên phải vị trí của x.

Khi đó ta sẽ có 2 mảng con: mảng bên trai của x và mảng bên phải của x. Tiếp tục công việc với mỗi mảng con(chọn pivot, phân đoạn) cho tới khi mảng được sắp xếp.

- Đặt pivot là phần tử cuối cùng của dãy số arr. Chúng ta bắt đầu từ phần tử trái nhất của dãy số có chỉ số là left, và phần tử phải nhất của dãy số có chỉ số là right -1(bỏ qua phần tử pivot). Chừng nào left < right mà arr[left] > pivot và arr[right] < pivot thì đổi chỗ hai phần tử left và right. Sau cùng, ta đổi chỗ hai phần tử left và pivot cho nhau. Xem hình minh họa phía dưới. Khi đó, phần tử left đã đứng đúng vị trí và chia dãy số làm đôi(bên trái và bên phải)

**B. Hàm thực hiện thuật toán:**



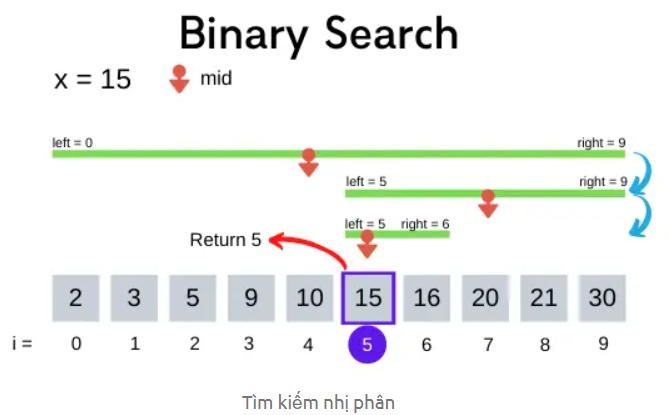


**2.2. Các thuật toán tìm kiếm cơ bản:**

**2.2.1. Tìm kiếm nhị phân ( Binary Search):**

**A. Ý tưởng thuật toán:**

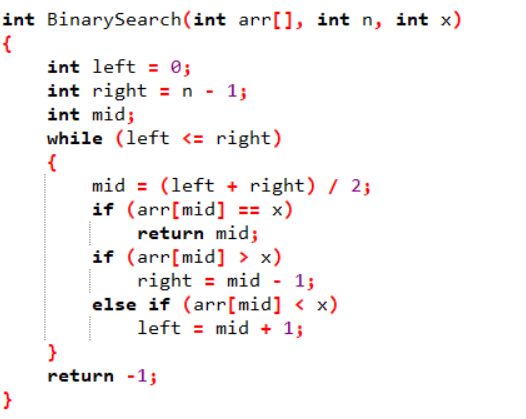
**-** Tìm kiếm nhị phân (binary search) hay còn một số tên gọi khác nữa như tìm kiếm nửa khoảng (half-interval search), tìm kiếm logarit (logarithmic search), chặt nhị phân (binary chop) là thuật toán tìm kiếm dựa trên việc chia đôi khoảng đang xét sau mỗi lần lặp, sau đó xét tiếp trong nửa khoảng có khả năng chứa giá trị cần tìm, cứ như vậy cho đến khi không chia đôi khoảng được nữa. Thuật toán tìm kiếm nhị phân **chỉ áp dụng được cho danh sách đã có thứ tự hay đã được sắp xếp.**



**-** Ví dụ như bạn có một dãy số tăng từ 1 đến 100, yêu cầu bạn tìm số 30. Bạn xem phần tử chính giữa của dãy số thì thấy là số 50, vậy thì bạn biết chắc là 30 chỉ có thể nằm trong khoảng dưới 50 thôi, vậy thì giới hạn tìm kiếm được thu hẹp lại một nửa. Ví dụ như tìm số 70 chẳng hạn thì 50 lại nhỏ hơn 70, do đó ta biết chắc 70 chỉ có thể nằm trong khoảng từ 51 đến 100 thôi. Cứ tiếp tục như thế cho đến khi tìm gặp hoặc không thể chia đôi khoảng nữa.

- Do cách tìm kiếm chia đôi khoảng này, sau mỗi lần lặp, khoảng đang xét lại được chia đôi, và tiếp tục khoảng tiếp lại chia đôi khoảng đã được chia trước đó. Do đó, độ phức tạp thời gian của thuật toán này sẽ là O(log(n)), tốt hơn rất rất nhiều so với tìm kiếm tuyến tính.

**B. Hàm thực hiện thuật toán:**



**2.2.2. Thuật toán tìm kiếm tuần tự ( Linear Search ) :**

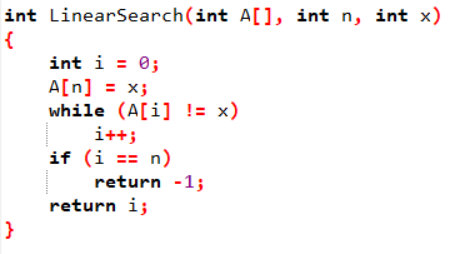
**A. Ý tưởng thuật toán:**

- Là thuật toán tìm kiếm cơ bản nhất và có độ phức tạp lớn nhất trong 3 thuật toán.

- Tìm kiếm tuyến tính (linear search) hay tìm kiếm tuần tự (sequential search) là thuật toán tìm kiếm bằng cách duyệt qua tất cả các phần tử của danh sách cho đến khi gặp phần tử cần tìm hoặc là đã hết danh sách. Do cách tìm kiếm duyệt từ đầu đến cuối này, độ phức tạp thời gian của thuật toán này sẽ là O(n).

- Về cách hoạt động, thuật toán này sữ quét lần lượt từng phần tử từ đầu đến cuối dãy, nếu gặp phần tử cần tìm thì in ra vị trí của nó và kết thúc.

**B. Hàm thực hiện thuật toán:**



**3. TỔNG KẾT:**

**-** Như vậy, qua những nội dung trên ta đã phần nào nắm được nội dung của các thuật toán tìm kiếm và sắp xếp đơn giản nhất. Từ đó viết được một chương trình hoàn chỉnh có đầy đủ các tính năng sắp xếp/ tìm kiếm như trên.

**3.1. Hạn chế:**

-Nội dung lý thuyết ở mức cơ bản, thiếu tính chi tiết

- Chưa có mô phỏng thuật toán bằng đồ họa ( mức 2)

-...

**3.2. Tài liệu tham khảo:**

**+** Tài liệu kỹ thuật lập trình, lập trình hương đối tượng trường Đại học Nha Trang

+ Trang web chia sẻ lập trình Geeksforgeeks ( https://www.geeksforgeeks.org/ )

+ Stackoverflow( https://stackoverflow.com/ )

+ Blog lập trình Topdev ( https://topdev.vn/blog/ )

**--------------------Hết-------------------**