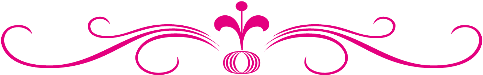


**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM**

**TP.HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

Đồ án một chỉ

***ĐỀ TÀI***: ***Xây dựng chương trình hỗ trợ học môn cấu trúc dữ liệu và giải thuật.***

**Giảng viên hướng dẫn : Trần Văn Thọ**

# Sinh viên thực hiện : Nguyễn Đắc Huề

Lê Phước Nhất

**Lớp :**  ....................................

## MSSV :....................................

**Biểu đồ hình trụ tăng giảm, khoảng cách giữa các land, xây dựng thời gian chạy, sắp xếp theo thứ tự tăng giảm dần, tạo mới nút or load dữ liệu các nut, giới hạn số node 100 nút, đổi màu khi được sắp xếp,lý thuyết với demo đi kèm, trong khoảng thời gian chạy xong lấy kết quả thời gian so sánh tối ưu**

Chương 2: TÌM KIẾM VÀ SẮP XẾP

**2.1 Giới thiệu:**

Trong chương này sẽ xem xét các giải thuật tìm kiếm và sắp xếp thông dụng. Cấu

trúc dữ liệu chính để minh họa các thao tác này chủ yếu là mảng một chiều. Đây cũng là một trong những cấu trúc dữ liệu thông dụng nhất.

Khi khảo sát giải thuật tìm kiếm, chúng ta sẽ làm quen với hai giải thuật. Giải thuật thứ nhất là giải thuật tìm kiếm tuần tự, giải thuật này có độ phức tạp tuyến tính O(n).

Giải thuật thứ hai là giải thuật tìm kiếm nhị phân, giải thuật này có độ phức tạp

O(log2(n)). Tuy giải thuật tìm kiếm nhị phân có ưu điểm tìm kiếm nhanh hơn giải thuật tìm kiếm tuyến tính nhưng giải thuật tìm kiếm nhị phân phải sắp xếp trước khi tìm kiếm.

Vì thế cần phải dựa vào điều kiện thực tế để quyết định chọn giải thuật nào.

**2.2. Nhu cầu tìm kiếm và sắp xếp dữ liệu trong một hệ thống thông tin**

Trong hầu hết các hệ lưu trữ, quản lý dữ liệu, thao tác tìm kiếm thường được thực hiện nhiều để khai thác thông tin.

*Ví dụ:* Tra cứu từ điển, tìm sách trong thư viện, …

Do các hệ thống thông tin thường phải lưu trữ một khối lượng dữ liệu đáng kể, nên việc xây dựng các giải thuật cho phép tìm kiếm nhanh sẽ có ý nghĩa rất lớn. Cho nên nếu dữ liệu trong hệ thống được tổ chức theo một trật tự nào đó, thì việc tìm kiếm sẽ tiến hành nhanh chóng và hiệu quả hơn.

*Ví dụ:* Sách trong thư viện được sắp xếp theo từng chủ đề.

Hiện nay có nhiều giải thuật tìm kiếm và sắp xếp được xây dựng, mức độ hiệu quả

của từng giải thuật còn phụ thuộc vào tính chất của cấu trúc dữ liệu cụ thể mà nó tác

động đến.

**2.3. Tìm kiếm:**

2.3.1. Khái quát về tìm kiếm

Trong thực tế, khi thao tác, khai thác dữ liệu chúng ta hầu như lúc nào cũng phải

thực hiện thao tác tìm kiếm. Việc tìm kiếm nhanh hay chậm tùy thuộc vào trạng thái và trật tự của dữ liệu trên đó. Kết quả của việc tìm kiếm có thể là không có (không tìm thấy) hoặc có (tìm thấy). Nếu kết quả tìm kiếm là có tìm thấy thì nhiều khi chúng ta còn phải xác định xem vị trí của phần tử dữ liệu tìm thấy là ở đâu?

Công việc tìm kiếm sẽ hoàn

Tìm được phần tử có khóa Key bằng với giá trị cần tìm x, lúc đó phép tìm kiếm thành công Cho biết vị thành công trí tìm thấy.

Cho biết vị trí tìm thấy.

 Không tìm được phần tử có khóa Key nào bằng với giá trị có khóa Key nào bằng với giá trị tìm kiếm thất bại.

2.3.2. Giải thuật tìm kiếm tuyến tính (Linear Search)

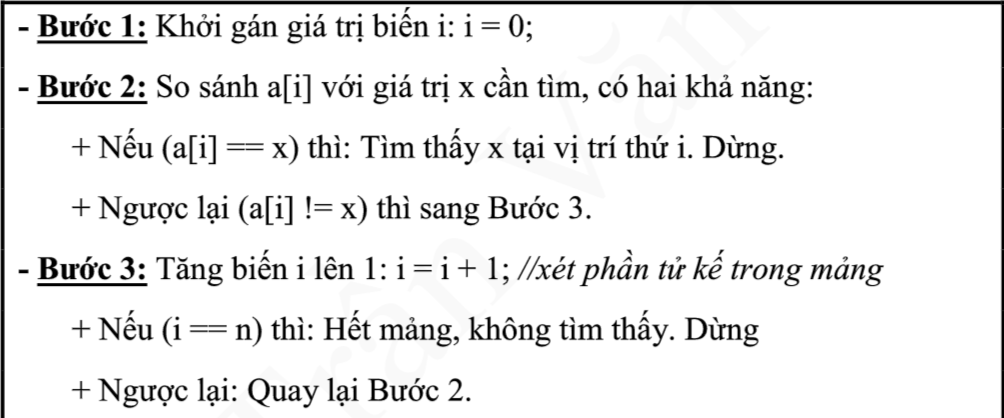
2.3.2.1. Giải thuật

Gọi x là giá trị cần tìm và a là mảng chứa dữ liệu có n phần tử.

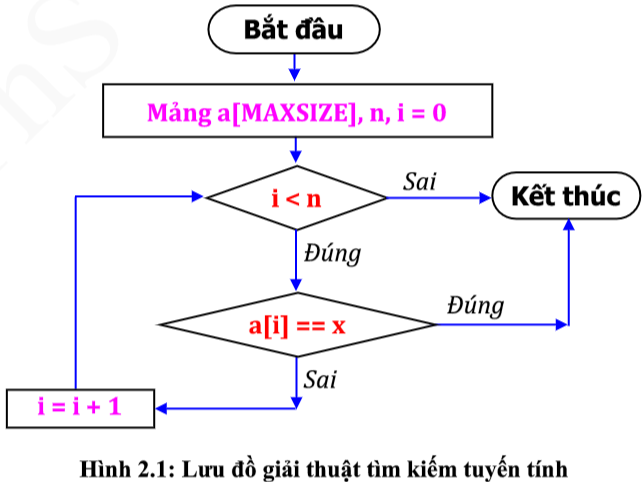
 Ý tưởng giải thuật:Tiến hành so sánh x với từng phần tử của mảng a, lần lượt so

Tiến hành so sánh x với từng phần tử của mảng a, lần lượt so sánh với phần tử thứ 1, phần tử thứ 2, …. Quá trình so sánh được thực hiện tuần tự cho đến khi gặp phần tử có khóa cần tìm x, hoặc đã tìm hết mảng mà không thấy x.

 Các bước tiến hành giải thuật:

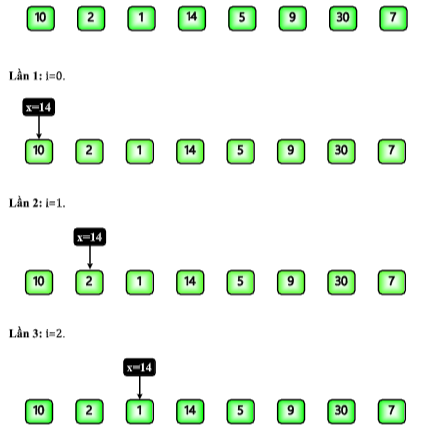


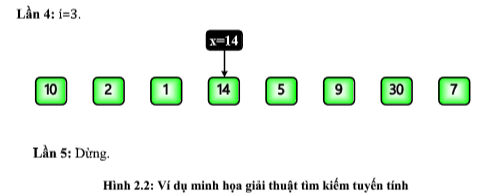
2.3.2.2. Lưu đồ



2.3.2.3. Ví dụ minh họa giải thuật

Cho dãy số a gồm 8 phần tử như sau:



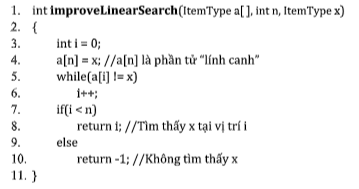


2.3.2.4. Nhận xét.

- Giải thuật tìm kiếm tuyến tính không phụ dần của các phần tử trong mảng trong mảng.

- Số phép so sánh của giải thuật trong trường hợp xấu nhất là 2n + 1.

Để giảm thiểu số phép so sánh trong vòng lặp cho giải thuật chỉ còn n + 1, có thể thêm phần tử “lính canh” vào cuối dãy như sau:



2.3.3. Giải thuật tìm kiếm nhị phân (Binary Search)

2.3.3.1. Giải thuật

Giả sử ta có dãy chứa các giá trị đã được sắp xếp tăng (có thứ tự tăng theo một

tiêu chuẩn nào đó), các phần tử trong dãy có quan hệ ai - 1 ≤ ai ≤ ai + 1, từ đó kết luận được nếu x > ai thì x chỉ có thể xuất hiện trong đoạn [ai + 1, an] của dãy, ngược lại nếu x < ai thì x chỉ có thể xuất hiện trong đoạn [a0, ai - 1] của dãy. Giải thuật tìm nhị phân áp dụng nhận xét trên đây để tìm các giới hạn phạm vi tìm kiếm sau mỗi lần

so sánh x với một phần tử trong dãy.

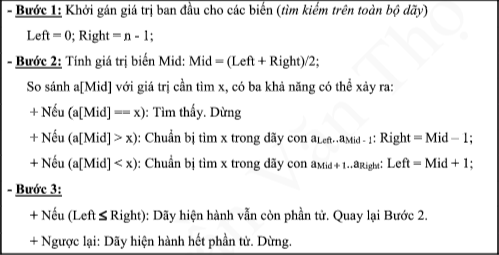
**Ý tưởng giải thuật:**

xét một dãy khóa đã sắp xếp tăng dần từ a[left] <= ... <= a[mid] <= …<= a[right].

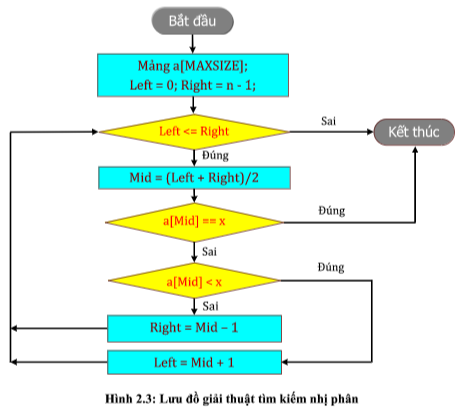
Tại mỗi bước tiến hành so sánh khóa cần tìm x với phần tử nằm ở giữa a[mid] của dãy hiện hành, dựa vào kết quả so sánh này để quyết định giới hạn dãy tìm kiếm ở bước kế tiếp là nửa bên trái từ a[left] <=…<= a[mid - 1] hay nửa bên phải từ a[mid +1]<=…<=a[right]. Quá trình tìm kiếm sẽ kết thúc khi left >right: không tìm thấy phần tử nào có khóa x, hoặc tìm được một phần tử a[mid] = x.

**Các bước tiến hành giải thuật:**

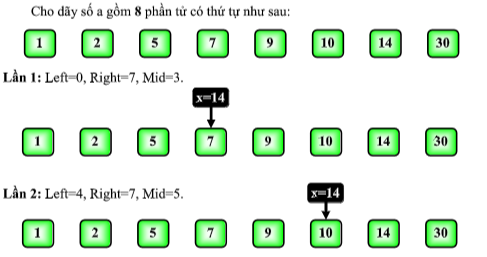
Gọi x là khóa cần tìm, a là mảng chứa các giá trị dữ liệu gồm n phần tử đã sắp xếp, Left và Right là chỉ số đầu và cuối của đoạn cần tìm, Mid là chỉ số của phần tử nằm giữa của đoạn cần tìm. Công việc tìm kiếm được tiến hành như sau:

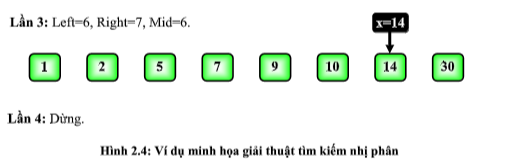


2.3.3.2. Lưu đồ:



2.3.4.3.Ví dụ minh họa giải thuật





2.3.4.3. Nhận xét

Giải thuật tìm nhị phân phụ thuộc vào thứ tự của các phần tử trong mảng để định hướng trong quá trình tìm kiếm, do vậy chỉ áp dụng được cho những dãy đã có thứ tự.

Giải thuật tìm kiếm nhị phân tiết kiệm thời gian hơn rất nhiều so với giải thuật tìm

tiết kiệm thời gian hơn rất nhiều so với giải thuật tìm kiếm tuyến tính do ONhị phân (log2n) < OTuyến tính(n). Tuy nhiên khi muốn áp dụng giải thuật tìm kiếm nhị phân cần phải xét đến thời gian sắp xếp dãy số để thỏa điều kiện dãy số có thứ tự, thời gian này là không nhỏ, và khi dãy số biến động cần phải tiến hành sắp xếp lại, … tất cả các nhu cầu đó tạo ra khuyết điểm chính cho giải thuật tìm kiếm nhị phân.

2.4. Các giải thuật sắp xếp

2.4.1. Định nghĩa sắp xếp

Sắp xếp danh sách (dãy) có n phần tử a0, a1, ..., an-1 là quá trình xử lý các phần tử trong danh sách để đặt chúng theo một thứ tự thỏa mãn một số tiêu chuẩn nào đó dựa trên thông tin lưu tại mỗi phần tử. Để quyết định những tình huống cần thay đổi vị trí các phần tử trong dãy, cần dựa vào kết quả của một loạt phép so sánh. Như vậy, hai thao tác cơ bản của giải thuật sắp xếp là so sánh và đổi chỗ, khi xây dựng một giải thuật sắp xếp cần chú ý tìm cách giảm thiểu những phép so sánh và đổi chỗ không cần thiết để tăng hiệu quả của giải thuật.

Ðối với các danh sách được lưu trữ trong bộ nhớ chính, nhu cầu tiết kiệm bộ nhớ

được xem xét ưu tiên hàng đầu, do vậy những giải thuật sắp xếp đòi hỏi cấp phát thêm

vùng nhớ để lưu trữ dãy kết quả, ngoài vùng nhớ lưu trữ danh sách ban đầu, thường ít được quan tâm. Thay vào đó, các giải thuật sắp xếp trực tiếp trên danh sách ban đầu - gọi là các giải thuật sắp xếp tại chỗ - lại được đầu tư phát triển. Phần này giới thiệu một số giải thuật sắp xếp từ đơn giản đến phức tạp có thể áp dụng thích hợp cho việc sắp xếp nội.

Để sắp xếp một danh sách chưa có thứ tự thành danh sách có thứ tự (giả sử có thứ

tự tăng), ta phải tiến hành triệt tiêu tất cả các cặp nghịch thế trong danh sách, hay nói

cách khác là tìm cách sắp xếp lại danh sách để sao cho tất cả các cặp đều là thuận thế.

Khái niệm nghịch thế:

Xét một danh sách (mảng một chiều) có n phần tử a0, a1…, an-1.

Nếu có i < j và ai > aj thì ta gọi đó là 1 cặp nghịch thế.

Ví dụ: Cho mảng a như sau: 14 5 7 8 3

* Vậy dãy trên ta có các cặp nghịch thế sau: (14, 5); (7, 3); (8, 3); ….

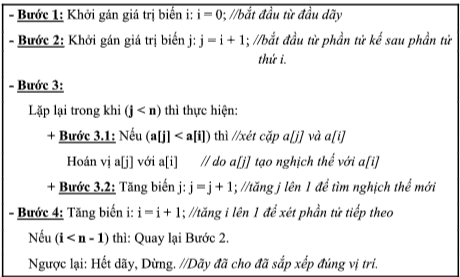
2.4.2. Giải thuật sắp xếp đổi chỗ trực tiếp (Interchange Sort)

2.4.2.1. Giải thuật

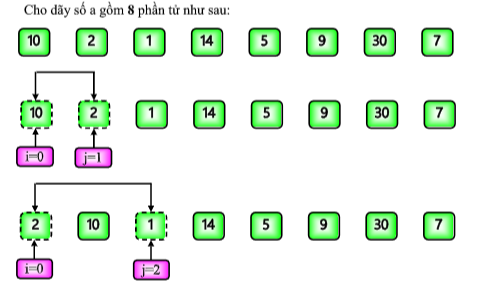
Ý tưởng giải thuật: Xuất phát từ đầu dãy, tìm tất cả nghịch thế chứa phần tử này, lần lượt làm triệt tiêu chúng bằng cách đổi chỗ phần tử này với phần tử tương ứng trong cặp nghịch thế, cuối cùng đưa được phần tử nhỏ nhất (lớn nhất) về vị trí đúng là đầu dãy hiện hành, sau đó không quan tâm đến nó nữa, xem dãy hiện xem dãy hiện hành chỉ còn (n - 1) phần tử của dãy ban đầu, và bắt đầu từ phần tử ở vị trí thứ 2 hành chỉ còn (n - 1) phần tử của dãy ban đầu, trong lần sắp xếp tiếp theo. Lặp lại quá trình xử lý như trên với các phần tử còn lại trong dãy, cho đến khi dãy hiện hành không còn phần tử nào thì dừng. cho đến khi dãy hiện hành không còn phần tử nào thì dừng.

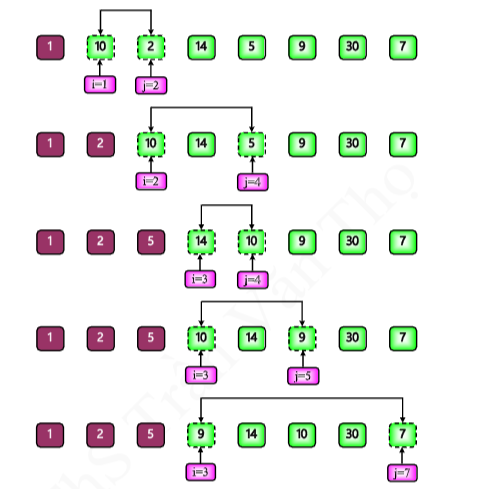
Các bước tiến hành giải thuật:

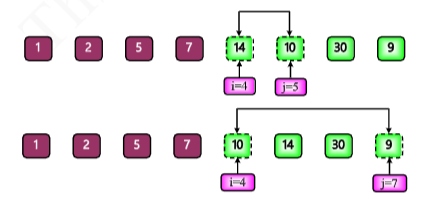
Giả sử a là mảng một chiều có n phần tử chưa được sắp xếp thứ tự. Ta áp dụng ý tưởng giải thuật để tiến hành sắp xếp mảng a tăng dần như sau:

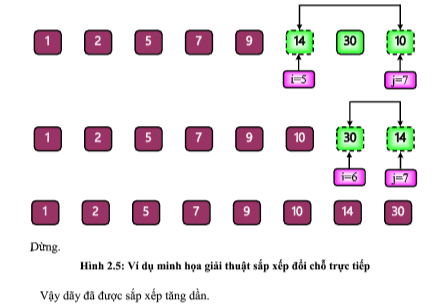


2.4.2.2. Ví dụ minh họa







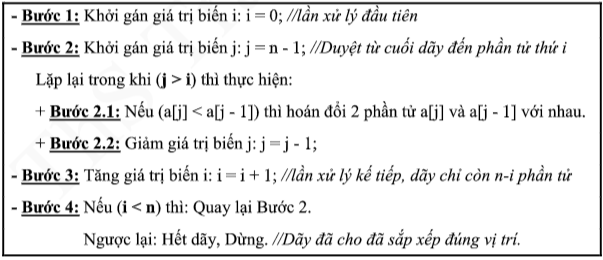


2.4.3. Giải thuật sắp xếp nổi bọt (Bubble Sort)

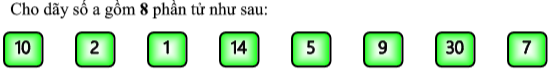
2.4.3.1. Giải thuật

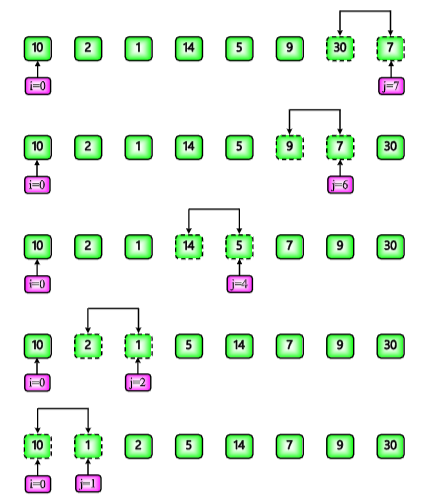
Ý tưởng giải thuật: Xuất phát từ cuối dãy (đầu dãy) và tiến hành đổi chỗ các cặp phần tử kế cận nhau để đưa phần tử nhỏ hơn (lớn hơn) về vị trí đầu (cuối) của dãy hiện hành. Sau khi đã được chuyển về đúng vị trí thì phần tử này sẽ không cần xét đến ở bước tiếp theo, do vậy ở lần xử lý thứ i sẽ có phần tử đầu dãy ở vị trí là i. Lặp lại quá trình xử lý như trên với các phần tử còn lại trong dãy, cho đến trí là i. Lặp lại quá trình xử lý như trên với các phần tử còn lại trong dãy, khi dãy hiện hành không còn phần tử nào thì dừng.

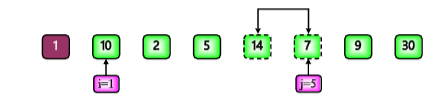
Các bước tiến hành giải thuật:

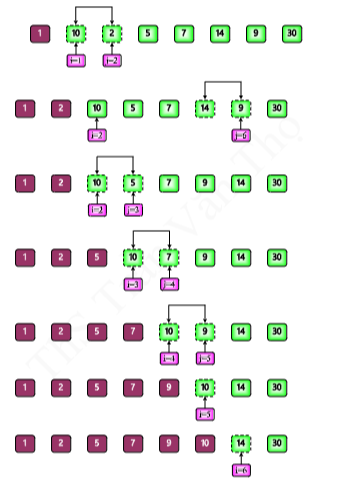


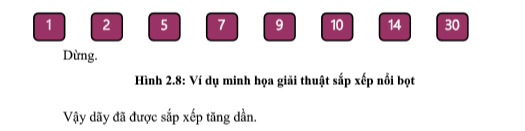
2.4.3.2. Ví dụ minh họa giải thuật











2.4.4. Giải thuật sắp xếp chọn trực tiếp (Selection Sort)

2.4.4.1. Giải thuật

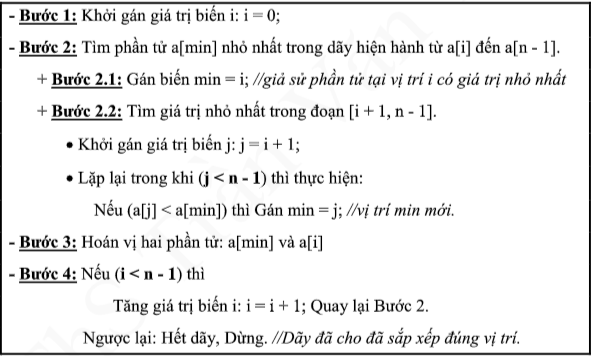
Ý tưởng giải thuật: Chọn phần tử nhỏ nhất (lớn nhất) trong n phần tử ban đầu, đưa phần tử này về vị trí đúng là đầu dãy hiện hành, sau đó không quan tâm đến nó nữa, xem dãy hiện hành chỉ còn (n - 1) phần tử của dãy ban đầu, và bắt đầu từ phần tử ở vị trí thứ 2 trong lần sắp xếp tiếp theo. Lặp lại quá trình xử lý như trên các phần tử còn lại trong dãy, cho đến khi dãy hiện hành chỉ còn đúng một cho đến khi dãy hiện hành chỉ còn đúng một phần tử thì dừng.

Các bước tiến hành giải thuật:

Dãy ban đầu a có n phần tử chưa được sắp xếp, sẽ thực hiện (n - 1) lượt việc

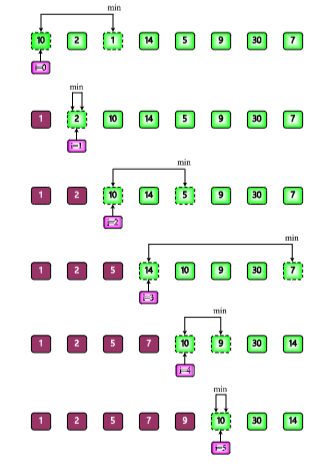
đưa phần tử nhỏ nhất trong dãy hiện hành về vị trí đúng ở đầu dãy. Các bước tiến

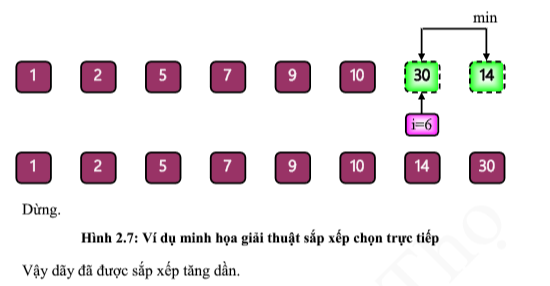
hành như sau:



2.4.4.2. Ví dụ minh họa giải thuật







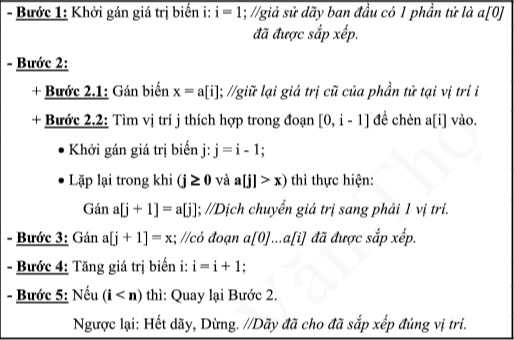
2.4.5. Giải thuật sắp xếp chèn trực tiếp (InsertionSort)

2.4.5.1. Giải thuật

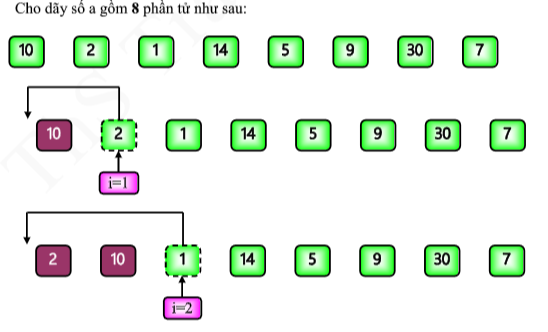
Ý tưởng giải thuật: là một giải thuật sắp xếp bắt chước cách sắp xếp quân bài của những người chơi bài. Muốn sắp một bộ bài theo trật tự người chơi bài rút lần lượt từ quân thứ 2, so với các quân đứng trước nó để chèn vào vị trí thích hợp.

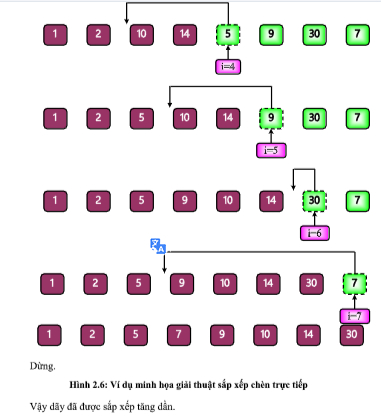
Cơ sở lập luận của giải thuật sắp xếp chèn có thể mô tả như sau: Giả sử có một dãy ban đầu có n phần tử là a0, a1, ..., an–1 chưa được sắp xếp. Xét dãy con gồm I phần tử đầu a0, a1, ..., ai-1. Với i=1, dãy gồm một phần tử đã được sắp xếp. Giả sử trong dãy có i-1 phần tử đầu a0, a1, ..., ai-2 đã được sắp xếp,trong dãy có a0, a1, ..., ai-2 đã được sắp xếp, tìm cách chèn phần tử có vào dãy có giá trị bằng x vào dãy có i-1 phần tử đầu thì phải tìm vị trí thích hợp của nó trong dãy a0, a1, ..., ai-2, vị trí thích hợp đó là vị trí đứng trước phần tử lớn hơn nó và sau phần tử nhỏ hơn hoặc bằng nó, sau khi chèn phần tử có giá trị x vào dãy có i-1 phần tử đầu a0, a1, ..., ai-2 sẽ thu được dãy có i phần tử đầu a0, a1, ... , ai-1 được sắp xếp.

Các bước tiến hành giải thuật:



2.4.5.2.Ví dụ minh họa giải thuật





2.4.6. Giải thuật sắp xếp Shaker Sort

2.4.6.1. Giải thuật

Ý tưởng giải thuật: Đây là giải thuật cải tiến của giải thuật Bubble Sort nhưng

được cải tiến từ sắp xếp 1 chiều thành 2 chiều. Trong mỗi lần sắp xếp, duyệt mảng

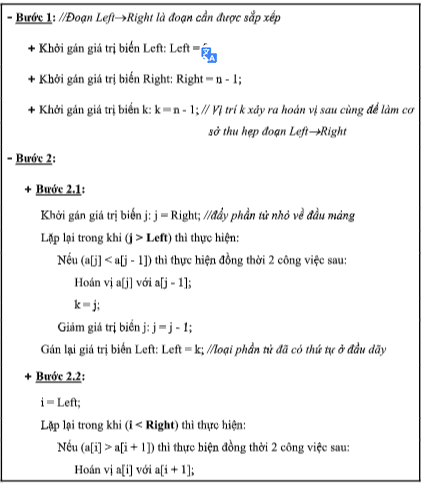
theo hai lượt từ hai phía khác nhau:

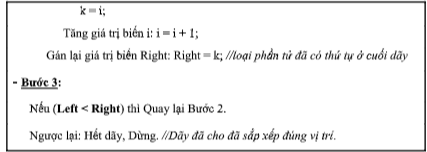
 Lượt đi: Đẩy phần tử nhỏ về đầu mảng.

 Lượt về: Đẩy phần tử lớn về cuối mảng.

Ghi nhận lại những đoạn đã sắp xếp nhằm tiết kiệm các phép so sánh thừa.

Các bước tiến hành giải thuật:





2.4.7. Giải thuật sắp xếp vun đống (Heap Sort)

2.4.7.1. Ðịnh nghĩa cấu trúc dữ liệu Heap

Giả sử xét trường hợp sắp xếp tăng dần, khi đó Heap được định nghĩa là một dãy

các phần tử aLeft, aLeft + 1, ..., aRight thỏa các quan hệ với mọi i[Left, Right/2]:

 ai >= a2i

ai >= a2i + 1, {trong đó (ai, a2i), (ai, a2i + 1) là các cặp phần tử liên đới}

Và có hai tính chất sau:

Tính chất 1:Phần tử a0 (đầu Heap) luôn là phần tử lớn nhất trong Heap.

Tính chất 2: Cắt bỏ một số phần tử về phía phải của Heap (cuối Heap) thì dãy

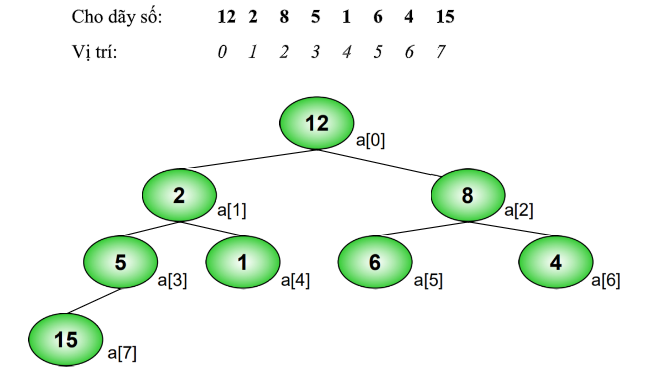
con còn lại vẫn là một Heap.

2.4.7.2. Giải thuật

Ý tưởng giải thuật:

Heap Sort tận dụng được các phép so sánh ở bước i-1 mà giải thuật sắp xếp chọn trực tiếp không tận dụng được.

Để làm được điều này Heap Sort thao tác dựa trên cây.



* Ở cây trên, phần tử ở mức i chính là phần tử lớn trong cặp phần tử ở mức i +

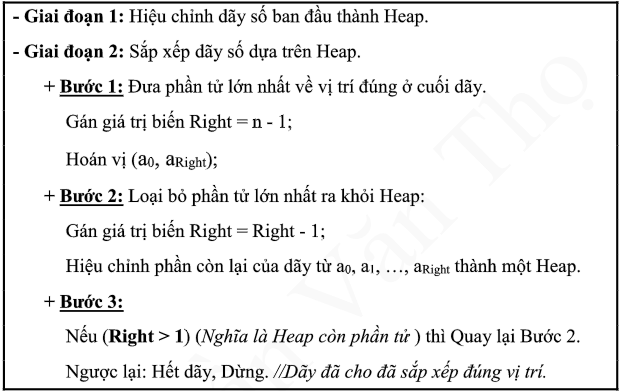
1, do đó phần tử ở nút gốc là phần tử lớn nhất.

* Nếu loại bỏ gốc ra khỏi cây, thì việc cập nhật cây chỉ xảy ra trên những

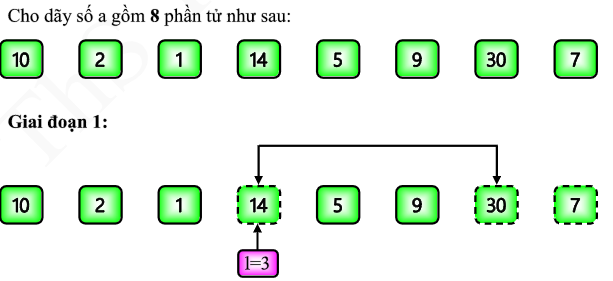
nhánh liên quan đến phần tử mới loại bỏ, còn các nhánh khác thì bảo toàn.

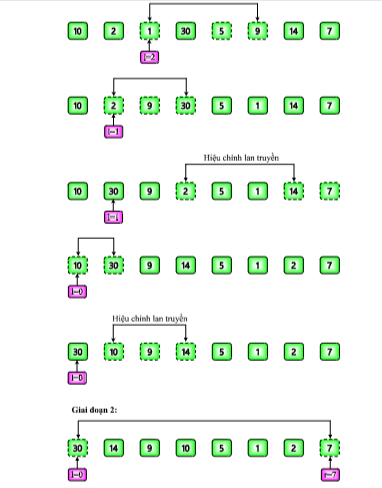
* Bước kế tiếp có thể sử dụng lại kết quả so sánh của bước hiện tại.

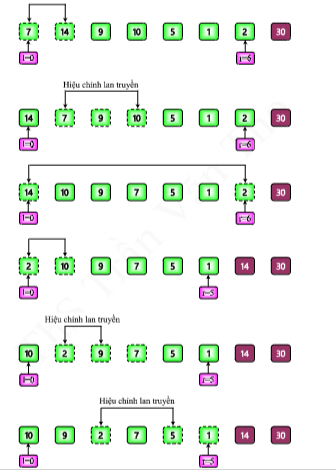
**Các bước tiến hành giải thuật:**

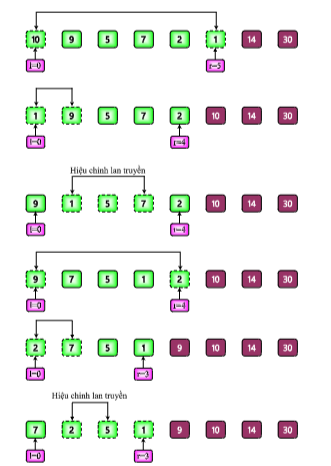


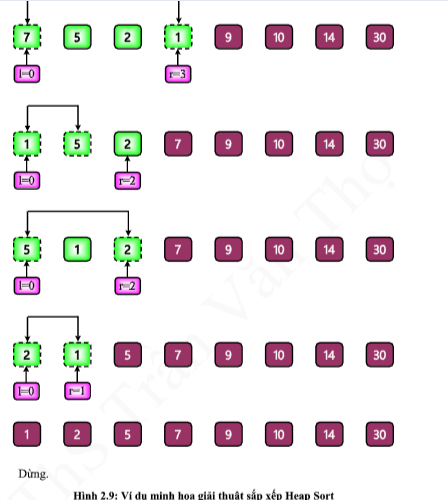
**2.4.7.3.Ví dụ minh họa giải thuật**











**2.4.8.Sort Giải thuật sắp xếp Shell**

**2.4.8.1. Giải thuật**

Ý tưởng giải thuật: Đây là giải thuật cải tiến của giải thuật Insertion Sort.

Phân hoạch dãy ban đầu thành các dãy con. Tiến hành sắp xếp các dãy con theo phương pháp chèn trực tiếp. Dùng phương pháp chèn trực tiếp sắp xếp lại cả dãy.

Giải thuật Shell Sort dựa trên ý tưởng sắp xếp các phần tử theo giải thuật chèn

trực tiếp nhưng với độ dài bước giảm dần. Ý tưởng của giải thuật sắp xếp này là

phân hoạch (phân chia) dãy ban đầu thành những dãy con gồm các phần tử ở cách

nhau h vị trí:

Dãy ban đầu: a0 a2 … an-1 được xem như sự xen kẽ các dãy con sau:

Dãy con thứ nhất: a0 ah+0 a2h+0

Dãy con thứ hai: a1 ah+1 a2h+1

Dãy con thứ h: ah-1 a2h-1 a3h-1 …

a3h-1 …

Ví dụ: 15, 7, 3, 1

h có dạng 3i + 1: 364, 121, 40, 13, 4, 1 h có dạng 3i

Dãy fibonaci: 34, 21, 13, 8, 5, 3, 2, 1

h là dãy các số nguyên tố giảm dần đến 1: 13, 11, 7, 5, 3, 1.

Các bước tiến hành giải thuật:

