UIT
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Data Structures & Algorithms

SẮP XẾP

Nội dung

1. Bài toán sắp xếp.
2. Tổng quan các phương pháp sắp xếp nội.
3. Interchange Sort – Đổi chỗ trực tiếp.
4. Selection Sort – Chọn trực tiếp
5. Bobble Sort – Nổi bọt
6. Shaker Sort - Nổi bọt **cải tiến**
7. Insertion Sort – Chèn trực tiếp
8. Binary Insertion Sort – Chèn nhị phân

Bài Toán Sắp Xếp (Sorting)

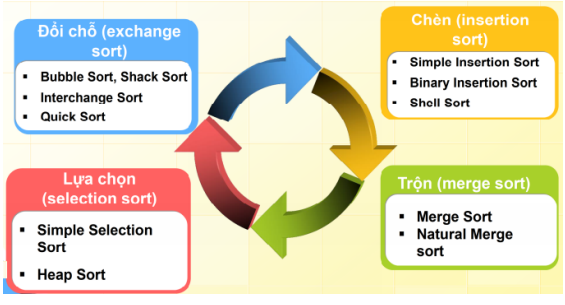
- ✓ Sắp xếp là quá trình bố trí lại các phần tử của một tập hợp theo thứ tự nào đó →tìm kiếm dễ dàng, nhanh.
- ✓ Việc sắp xếp được tiến hành dựa trên khóa của phần tử. Vídụ: danh mục điện thoại gồm: Tên cơ quan, địa chỉ, số điện thoại.
- ✓ Đơn giản bài toán:
  - Khóa là các giá trị số
  - Phần tử chỉ có trường khóa, không có các thành phần khác.
  - Sắp xếp mảng a: a[0], a[1],…,a[n-1] theo một thứ tự nhất định
  - Sắp xếp theo thứ tự tăng dần

Các phương pháp sắp xếp

Có 2 loại giải thuật sắp xếp:

- **Sắp xếp nội (internal sorting):**
  - Toàn bộ dữ liệu được đưa vào bộ nhớ trong.
  - Kích thước dữ liệu không lớn.
  - Thời gian thực hiện rất nhanh.
- **Sắp xếp ngoại (external sorting):**
  - Chỉ 1 phần nhỏ dữ liệu được đưa vào bộ nhớ trong, phần còn lại lưu ở bộ nhớ ngoài.
  - Kích thước dữ liệu rất lớn.
  - Thời gian thực hiện chậm.

Các phương pháp sắp xếp nội



Các phương pháp sắp xếp nội

- Selection Sort
- Insertion Sort
- Interchange Sort
- Bubble Sort
- Shaker Sort
- Binary Insertion Sort
- Shell Sort
- Heap Sort
- Quick Sort
- Merge Sort
- Radix Sort

Phức tạp hơn
Chi phí thấp

Đơn giản
Chi phí cao

Lớp bài toán khác

Đổi chỗ trực tiếp (Interchange Sort)

Bài toán: Viết hàm liệt kê tất cả các cặp giá trị trong mảng một chiều các số nguyên. Lưu ý: cặp (1,2) và cặp (2,1) là giống nhau

Ví dụ:

12	43	1	34	22

Các cặp giá trị trong mảng là:

- (12,43),(12,01),(12,34),(12,22)
- (43,01), (43,34), (43,22)
- (01,34), (01,22)
- (34,22)

Đổi chỗ trực tiếp (Interchange Sort)

```
11  void lietke(int a[], int n)
12  {
13      for(int i=0;i<n-1;i++)
14          for(int j=i+1;j<n;j++)
15          {
16              cout<<a[i]<<","<<a[j]<<endl;
17          }
18  }
19
```

Đổi chỗ trực tiếp (Interchange Sort)

Nghịch thế

Khái niệm: Một cặp giá trị (a_i,a_j) được gọi là nghịch thế khi a_i và a_j không thỏa điều kiện sắp thứ tự.

Ví dụ 1: Cho mảng một chiều các số thực *a* có *n* phần tử: a_0, a_1, a_2,…, a_{n-2}, a_{n-1}. Hãy sắp mảng theo thứ tự tăng dần. Khi đó cặp giá trị (a_i,a_j) $(i<j)$ được gọi là nghịch thế khi $a_i \geq a_j$

Ví dụ 2: Cho mảng một chiều các số thực *a* có *n* phần tử: a_0, a_1, a_2,…, a_{n-2}, a_{n-1}. Hãy sắp mảng theo thứ tự giảm dần. Khi đó cặp giá trị (a_i,a_j) $(i<j)$ được gọi là nghịch thế khi $a_i \leq a_j$

Đổi chỗ trực tiếp (Interchange Sort)

Ví dụ 3: Hãy liệt kê các cặp giá trị nghịch thế trong mảng sau, biết rằng yêu cầu là sắp xếp mảng tăng dần

14	29	-1	10	5	23

Kết quả:

- (14, -1), (14,10), (14,5)
- (29,-1), (29,10), (29,5), (29,23)
- (10,5)

Interchange Sort – Ý Tưởng

Thuật toán interchange sort sẽ duyệt qua tất cả các cặp giá trị trong mảng và hoán vị hai giá trị trong một cặp nếu cặp giá trị đó là nghịch thế.

Interchange Sort – Giải Thuật

Input: mảng a: a[0], a[1],…,a[n-1] chưa sắp xếp

Output: mảng a đã sắp xếp

Các bước thực hiện:

```
for (i = 0; i<n-1; i++)
    for (j = i+1; j < n; j++)
        if a[j]<a[i]                    //xét cặp a[i], a[j]
            Đổi chỗ a[i] và a[j]
```

✓ **Cài đặt:**

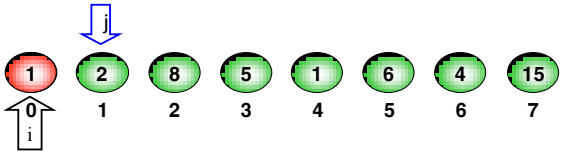
```
void InterchangeSort(int a[], int n )
{       int     i, j;
        for (i = 0 ; i < n-1 ; i++)
                for (j = i+1; j < n ; j++)
                        if(a[j]< a[i])      // nếu có sự sai vị trí thì đổi chỗ
                        {temp = a[i]; a[i] = a[j]; a[j] = temp;}
}
```

Interchange Sort – Độ Phức Tạp

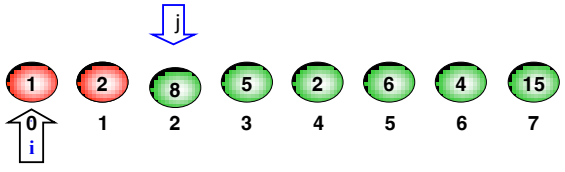
Trường hợp	Số lần so sánh	Số lần hoán vị
Tốt nhất	$\sum_{i=0}^{n-1}(n-i+1) = n(n-1)/2$	0
Xấu nhất	n(n-1)/2	n(n-1)/2

Độ phức tạp tính toán : $T(n) = O(n^2)$

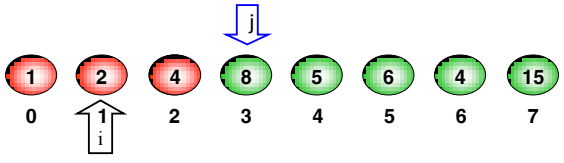
Interchange Sort – Minh Họa



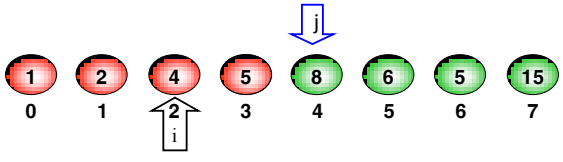
Interchange Sort – Minh Họa



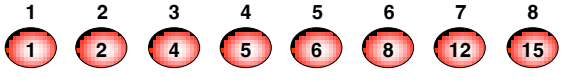
Interchange Sort – Minh Họa



Interchange Sort – Minh Họa

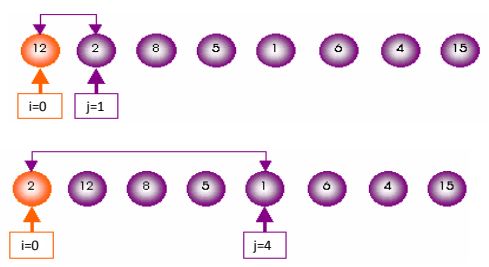


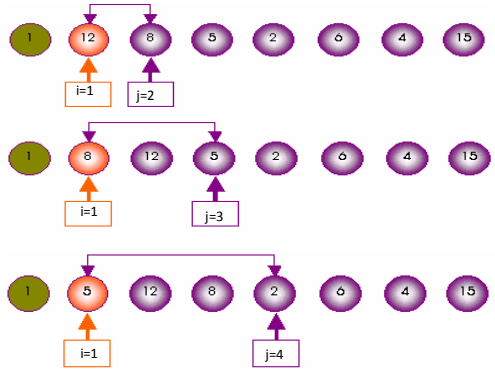
Interchange Sort – Minh Họa

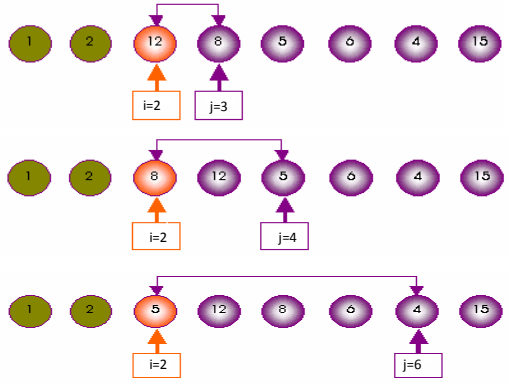


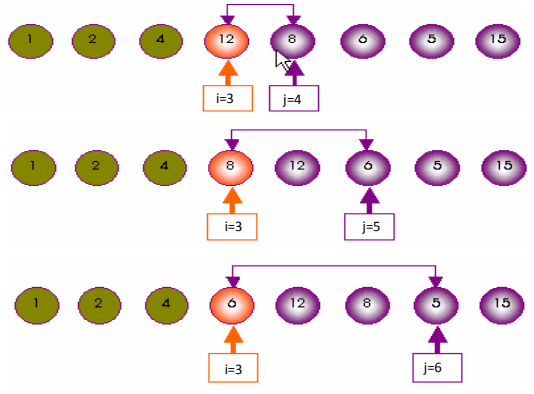
- Cho dãy số a:

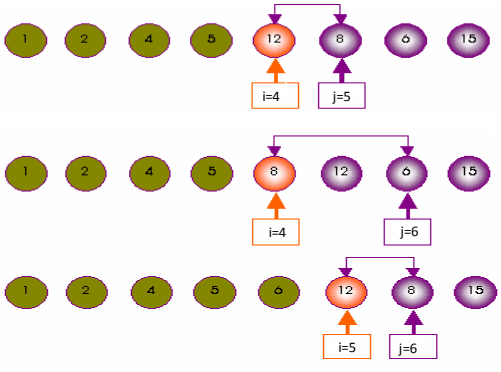
12 2 8 5 1 6 4 15

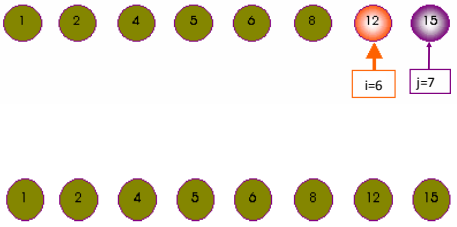












Interchange Sort – Bài Tập

Cho biết kết quả theo từng bước khi áp dụng thuật toán Interchange Sort sắp xếp dãy sau theo chiều tăng dần

45, 7, 12, 33, 21, 5, 2, 57, 15,

Selection Sort – Chọn trực tiếp

Bài toán: Viết hàm tìm vị trí giá trị lớn nhất trong mảng một chiều các số thực

Ví dụ:

0	1	2	3	4
12	43	1	34	22

Kết quả: 1

Selection Sort – Chọn trực tiếp

```
int vitrilonnhat (float a[], int n)
{
    int vtln =0;
    for(int i=0;i<n;i++)
        if(a[i]>a[vtln])
            vtln=i;
    return vtln;

}
```

Selection Sort – Chọn trực tiếp

Bài toán: Cho mảng một chiều a có n phần tử: a_0, a_1, a_2,…, a_{n-2}, a_{n-1}. Hãy sắp xếp các phần tử trong mảng tăng dần.

Selection Sort – Ý Tưởng

Chọn phần tử nhỏ nhất cho các vị trí 0, 1, …, n-1, cụ thể:

- **Lần chọn 0:**
  - Chọn phần tử nhỏ nhất (a[min]) trong khoảng vị trí từ **0** đến **n-1**
  - Đổi chỗ hai nút tại vị trí min và 0
- **Lần chọn 1:**
  - Chọn phần tử nhỏ nhất (a[min]) trong khoảng vị trí từ **1** đến **n-1**
  - Đổi chỗ hai nút tại vị trí min và **1**
- **Lần chọn i:**
  - Chọn phần tử nhỏ nhất (a[min]) trong khoảng vị trí từ **i** đến **n-1**
  - Đổi chỗ hai nút tại vị trí min và **i**
  - → lần chọn cuối n-2

Selection Sort

Input: mảng a: a[0], a[1],…,a[n-1] chưa sắp xếp

Output: mảng a đã sắp xếp

Các bước thực hiện:

for (i=0; i < n -1; i++)

{

b1: Tìm phần tử a[min] nhỏ nhất trong dãy hiện hành từ a[i] đến a[n-1]

b2: Hoán vị a[min] và a[i]

}

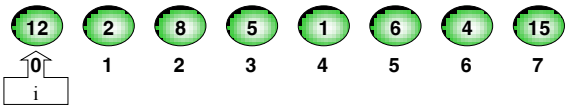
Selection Sort

```
void SelectionSort (int a[],int N )
{
    int    min, temp;                    // chỉ số phần tử nhỏ nhất trong dãy hiện hành
    for (int  i=0; i<n-1 ; i++)
    {
          // tìm phần tử nhỏ nhất
          min = i;
          for (int j = i + 1; j < n ; j++)
                if (a[j ] < a[min])
                      min = j;

          // đổi chỗ a[i] và a[min]
          temp = a[i];        a[min] = temp;     a[i] = a[min];
    }
}
```

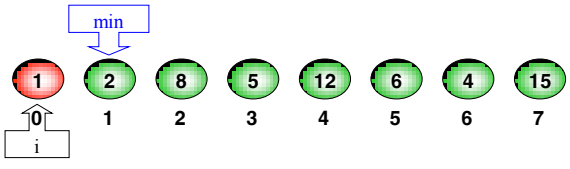
Selection Sort – Minh Họa

Vị trí nhỏ nhất(0,7) **Swap(a[0], a[4])**



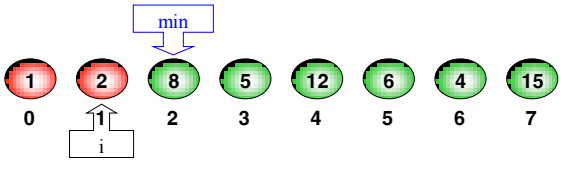
Selection Sort – Minh Họa

Vị trí nhỏ nhất(1,7) **Swap(a[1], a[1])**



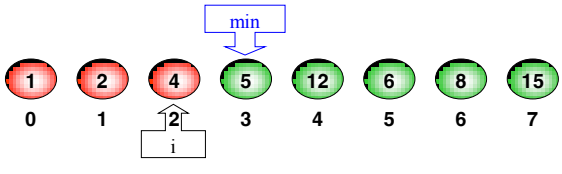
Selection Sort – Minh Họa

Vị trí nhỏ nhất(2,7) **Swap(a[2], a[6])**



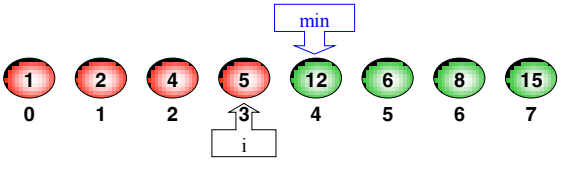
Selection Sort – Minh Họa

Vị trí nhỏ nhất(3, 7) **Swap(a[3], a[3])**

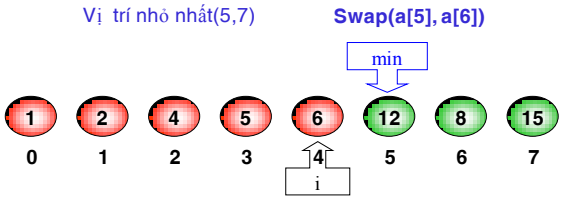


Selection Sort – Minh Họa

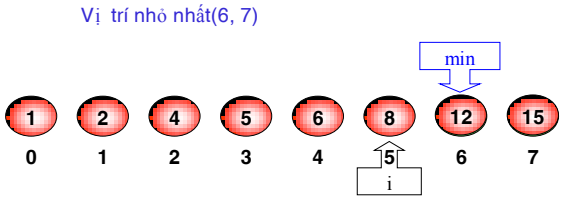
Vị trí nhỏ nhất(4, 7) **Swap(a[4], a[5])**



Selection Sort – Minh Họa



Selection Sort – Minh Họa



Selection Sort – Độ Phức Tạp

- Có n-1 lần chọn, từ lần chọn 0 đến n-2
- Ở mỗi lần chọn có 1 lần đổi chỗ.
- Ở lần chọn thứ i có n – i – 1 lần so sánh

Selection Sort – Độ Phức Tạp

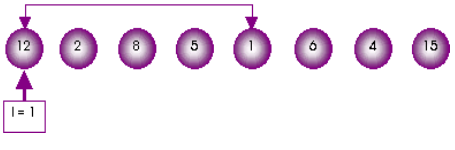
Trường hợp	Số lần so sánh C(n)	Số lần hoán vị
Tốt nhất	*- Ở lượt thứ i, bao giờ cũng cần (n-i-1) lần so sánh để xác định phần tử nhỏ nhất hiện hành* $C(n) = \sum_{i=0}^{n-2} n - i - 1$ $= (n-1) + (n-2) +$ $\ldots + (n-i-1) + \ldots + 1$ $= n(n-1)/2$	0
Xấu nhất	$n(n-1)/2$	$M(n) = n-1$

Độ phức tạp tính toán : $T(n) = O(n^2)$

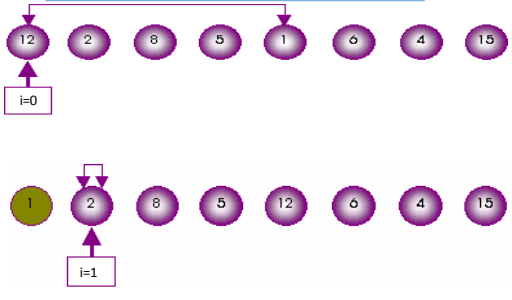
Selection Sort – Minh Họa

• Cho dãy số a:

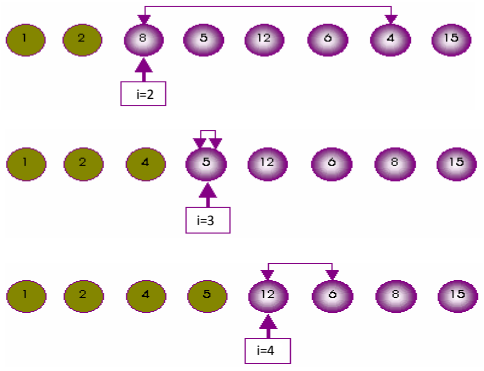
12 2 8 5 1 6 4 15



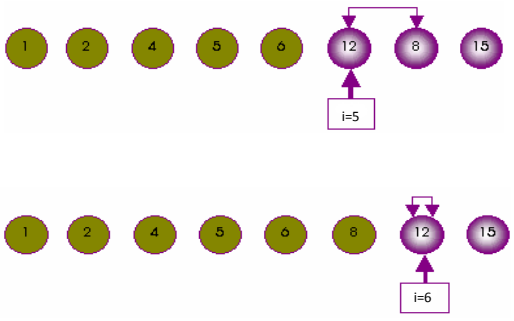
Selection Sort – Minh Họa



Selection Sort – Minh Họa



Selection Sort – Minh Họa



Selection Sort– Bài Tập

Cho biết kết quả theo từng bước khi áp dụng thuật toán Selection Sort sắp xếp dãy sau theo chiều tăng dần

45, 7, 12, 33, 21, 5, 2, 57, 15,

Bubble Sort – Nổi bọt

- Bài toán: Hãy liệt kê các cặp giá trị nằm kế tiếp nhau trong mảng một chiều các số nguyên.
- Ví dụ:

0	1	2	3	4
12	43	1	34	22

- Kết quả: (12,43), (43,1), (1,34), (34,22)

Bubble Sort – Nổi bọt

```
for(int i=0;i<n-1;i++)
    cout<<"("<<a[i]<<","<<a[i+1]<<")";
```

```
for(int i=n-1;i>0;i--)
    cout<<"("<<a[i]<<","<<a[i-1]<<")";
```

Bubble Sort – Nổi bọt

Bài toán: Hãy **đẩy** phần tử lớn **từ cuối dãy** lên đầu dãy dựa vào việc so sánh các cặp kề nhau

```
7
8
9
```

```
for(int i=n-1;i>0;i--)
    if(a[i] >a[i-1])
        swap(a[i],a[i-1]);
```

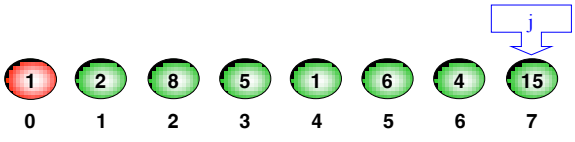
Bubble Sort – Ý tưởng

- Xuất phát từ cuối dãy (hoặc đầu dãy), đổi chỗ các cặp phần tử kế cận để đưa phần tử nhỏ hơn trong cặp phần tử đó về vị trí đúng đầu dãy hiện hành, sau đó sẽ không xét đến nó ở bước tiếp theo, do vậy ở lần xử lý thứ i sẽ có vị trí đầu dãy là i.
- Lặp lại xử lý trên cho đến khi không còn cặp phần tử nào để xét.

Bubble Sort – Ý tưởng

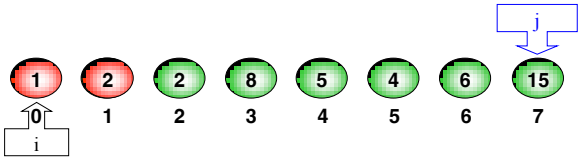
- Bước 1 : i = 0; // lần xử lý đầu tiên
- Bước 2 : j = N-1;//Duyệt từ cuối dãy ngược về vị trí i
  Trong khi (j > i) thực hiện:
  Nếu a[j]<a[j-1]
  Doicho(a[j],a[j-1]);
  j = j-1;
- Bước 3 : i = i+1; // lần xử lý kế tiếp
  Nếu i =N: Hết dãy. Dừng
  Ngược lại : Lặp lại Bước 2.

Bubble Sort – Minh Họa



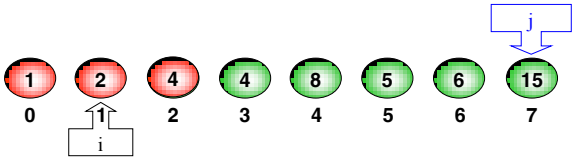
51

Bubble Sort – Minh họa



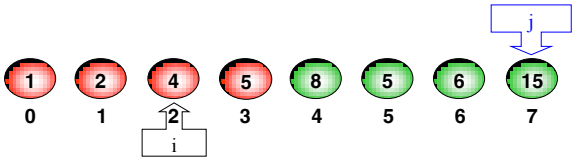
52

Bubble Sort – Minh họa



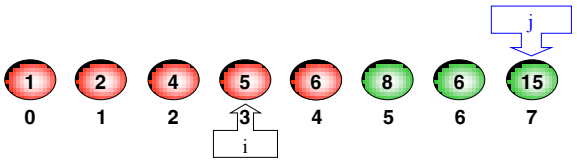
53

Bubble Sort – Minh họa



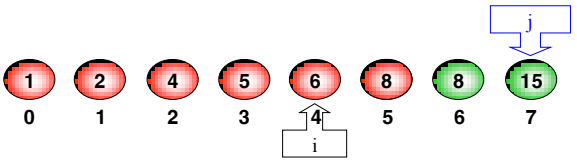
54

Bubble Sort – Minh họa



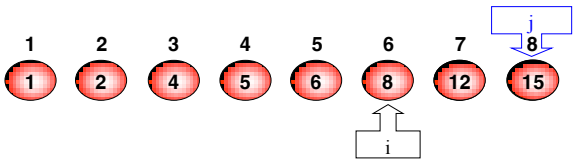
55

Bubble Sort – Minh họa



56

Bubble Sort – Minh họa



57

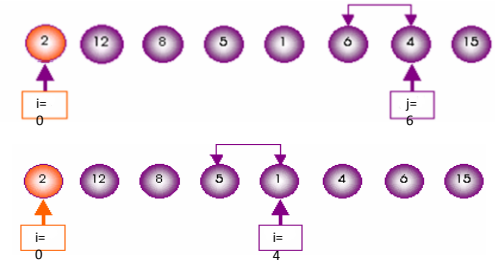
Bubble Sort – Cài đặt

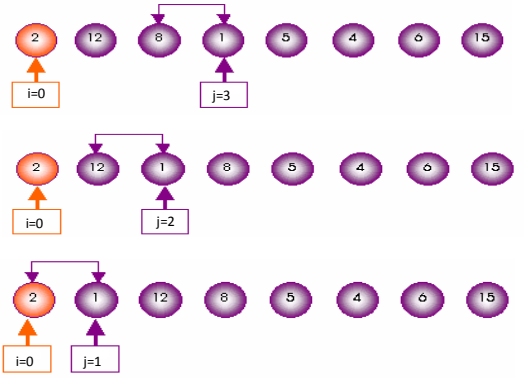
```
void BubbleSort(int a[],int n)
{
    int   i, j;
    for (i = 0 ; i<n-1 ; i++)
        for (j =n-1; j >i ; j --)
            if(a[j]< a[j-1])// nếu sai vị trí thì đổi chỗ
                Swap(a[j], a[j-1]);
}
```

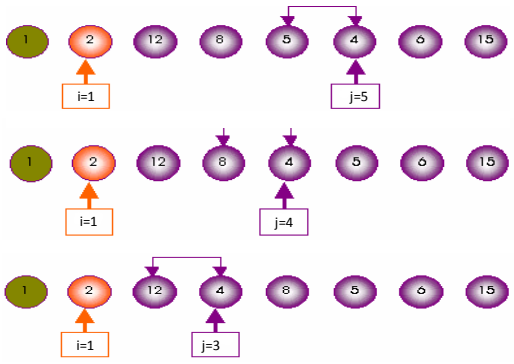
Bubble Sort – Ví dụ

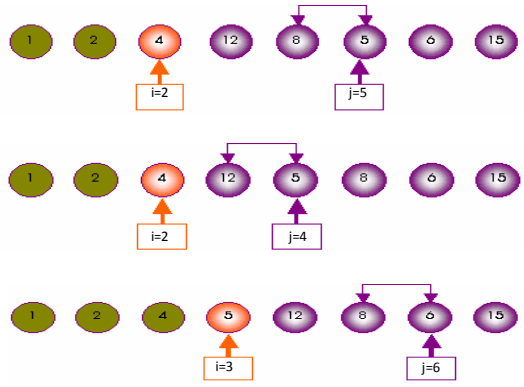
- Cho dãy số a:

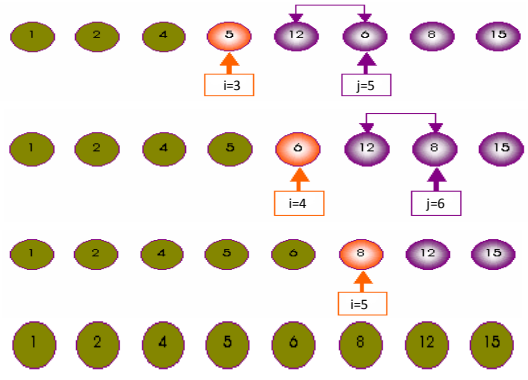
2 12 8 5 1 6 4 15











Bubble Sort– Độ Phức Tạp

Trường hợp	Số lần so sánh	Số lần hoán vị
Tốt nhất	$\sum_{i=1}^{n-1}(n-i+1)=\frac{n(n-1)}{2}$	0
Xấu nhất	$\frac{n(n-1)}{2}$	$\sum_{i=1}^{n-1}(n-i+1)=\frac{n(n-1)}{2}$

Độ phức tạp tính toán : $T(n) = O(n^2)$

Bubble Sort– Bài Tập

Cho biết kết quả theo từng bước khi áp dụng thuật toán Bubble Sort sắp xếp dãy sau theo chiều tăng dần

45, 7, 12, 33, 21, 5, 2, 57, 15,

Shaker Sort– Đặt vấn đề

- Trong mỗi lần sắp xếp, duyệt mảng **theo 2 lượt từ 2 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.

Shaker Sort– Thuật toán

Input: mảng a: a[0], a[1],…,a[n-1] chưa sắp xếp

Output: mảng a đã sắp xếp

Các bước thực hiện:

1: left = 0; right = n -1; //từ l đến r là đoạn cần được sắp xếp

k = n-1; // ghi nhận vị trí k xảy ra hoán vị sau cùng, để làm cơ sở thu hẹp đoạn left đến right

2: while (left<right)

{ 2.1: for (j = right; j > left; j --) // đẩy phần tử nhỏ về đầu mảng

if (a[j] < a[j-1]))

{ Đổi chỗ 2 nút tại vị trí j và j - 1

k = j; //lưu lại nơi xảy ra hoán vị }

2.2: left = k; //loại các phần tử đã có thứ tự ở đầu dãy

Shaker Sort– Thuật toán

Input: mảng a: a[0], a[1],…,a[n-1] chưa sắp xếp

Output: mảng a đã sắp xếp

Các bước thực hiện:

2.3: for (j = left; j < right; j ++) // đẩy phần tử lớn về cuối mảng

if (a[j] > a[j+1]))

{ Đổi chỗ 2 nút tại vị trí j và j +1

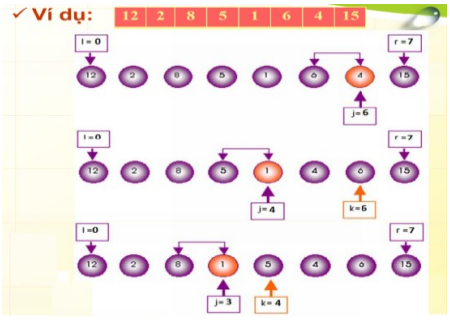
k = j; //lưu lại nơi xảy ra hoán vị }

2.4: right = k; //loại các phần tử đã có thứ tự ở cuối dãy

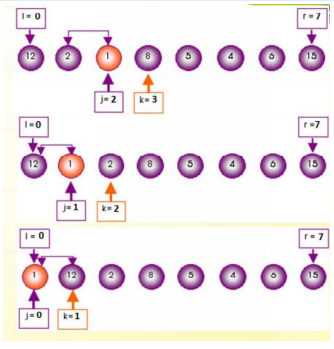
Shaker Sort – Cài đặt

```
void ShakeSort(int a[], int n )
{   int     j, left, right, k;
    left = 0;       right = n-1; k = n-1;
    while (left < right)
    {       for (j = right; j > left; j --)
                    if ( a[j]< a[j-1])
                    {       Hoanvi(a[j],a[j-1]);
                            k = j;                  }
            left = k;
            for (j = left; j < right; j ++)
                    if (a[j]> a[j+1])
                    {       Hoanvi(a[j],a[j-1]);
                            k = j;                  }
            right = k;
    }
}
```

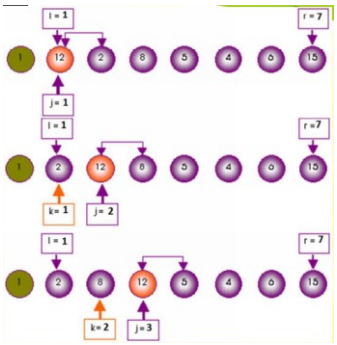
Shaker Sort – Ví dụ



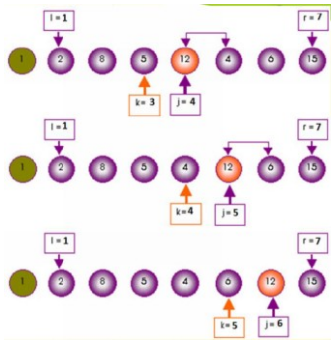
Shaker Sort – Ví dụ



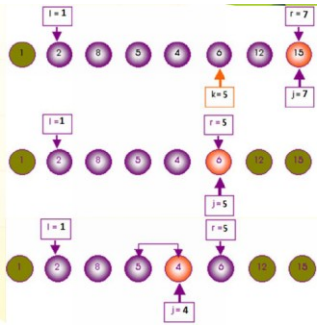
Shaker Sort – Ví dụ



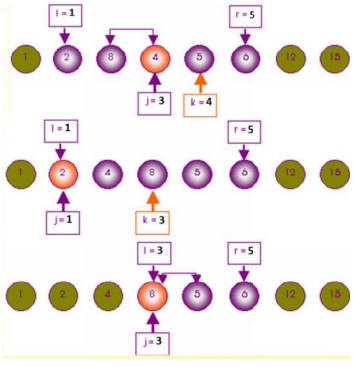
Shaker Sort – Ví dụ



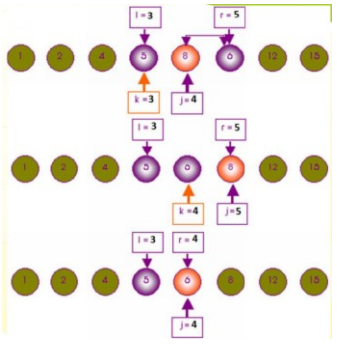
Shaker Sort – Ví dụ



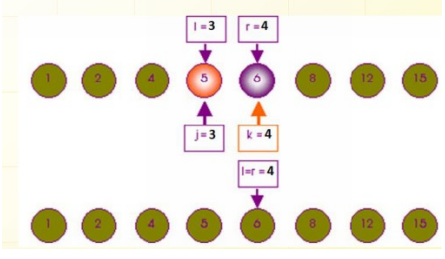
Shaker Sort – Ví dụ



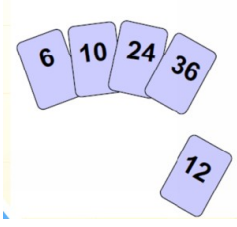
Shaker Sort – Ví dụ



Shaker Sort – Ví dụ



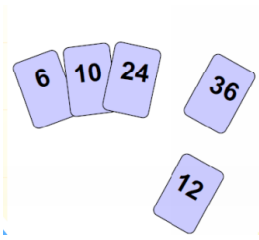
Insertion Sort– Chèn trực tiếp



Mỗi lần "chèn" thêm một quân bài vào tay cầm bài, các quân bài trên tay đã được sắp xếp.

Để chèn 12, cần phải tạo khoảng trống cho nó bằng cách dịch chuyển 36 trước và sau đó dịch chuyển 24.

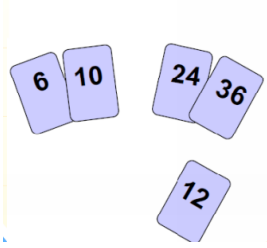
Insertion Sort– Chèn trực tiếp



Mỗi lần "chèn" thêm một quân bài vào tay cầm bài, các quân bài trên tay đã được sắp xếp.

Để chèn 12, cần phải tạo khoảng trống cho nó bằng cách dịch chuyển 36 trước và sau đó dịch chuyển 24.

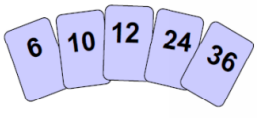
Insertion Sort– Chèn trực tiếp



Mỗi lần "chèn" thêm một quân bài vào tay cầm bài, các quân bài trên tay đã được sắp xếp.

Để chèn 12, cần phải tạo khoảng trống cho nó bằng cách dịch chuyển 36 trước và sau đó dịch chuyển 24.

Insertion Sort– Chèn trực tiếp



Mỗi lần "chèn" thêm một quân bài vào tay cầm bài, các quân bài trên tay đã được sắp xếp.

Để chèn 12, cần phải tạo khoảng trống cho nó bằng cách dịch chuyển 36 trước và sau đó dịch chuyển 24.

Insertion Sort– Chèn trực tiếp

Bài toán 1: Định nghĩa hàm thêm một giá trị x vào mảng một chiều các số thực có n phần tử đã được sắp tăng sao cho mảng vẫn được sắp tăng.

Insertion Sort– Chèn trực tiếp

```
11. void ThemBaoToan(float a[],
                     int &n,float x)
12. {
13.     int i = n-1;
14.     while(i>=0 && a[i]>x)
15.     {
16.         a[i+1] = a[i];
17.         i--;
18.     }
19.     a[i+1] = x;
20.     n++;
21. }
```

Insertion Sort– Chèn trực tiếp

```
11. void ThemBaoToan(float a[],
                     int &n,float x)
12. {
13.     int i = n-1;
14.     for(;(i>=0 && a[i]>x);)
15.     {
16.         a[i+1] = a[i];
17.         i--;
18.     }
19.     a[i+1] = x;
20.     n++;
21. }
```

Insertion Sort– Chèn trực tiếp

```
11. void ThemBaoToan(float a[],
                     int &n,float x)
12. {
13.     for(int i=n-1;
           (i>=0 && a[i]>x);)
14.     {
15.         a[i+1] = a[i];
16.         i--;
17.     }
18.     a[i+1] = x;
19.     n++;
20. }
```

Insertion Sort– Chèn trực tiếp

```
11. void ThemBaoToan(float a[],
                     int &n,float x)
12. {
13.     for(int i=n-1;
           (i>=0 && a[i]>x);i--)
14.     {
15.         a[i+1] = a[i];
16.     }
17.     a[i+1] = x;
18.     n++;
19. }
```

Insertion Sort– Ý Tưởng

Thuật toán Insertion sort sắp xếp dựa trên tư tưởng là không gian cần sắp xếp đã được sắp xếp một phần và ta chỉ cần thêm một giá trị mới vào không gian này sao cho không gian mới được sắp xếp

Insertion Sort– Ý Tưởng

- Giả sử i phần tử đầu tiên a_0 , a_1 ,... ,a_{i-1} đã có thứ tự.
- Tìm cách chèn phần tử a_i vào vị trí thích hợp của đoạn đã được sắp để có đoạn mới a_0 , a_1 ,... ,a_i trở nên có thứ tự.

→ *Ban đầu, xem đoạn gồm 1 phần tử a_0 đã được sắp*

- *Lần chèn 1: chèn a_1 vào đoạn a_0 để có đoạn a_0 a_1 được sắp*
- *Lần chèn 2: chèn a_2 vào đoạn a_0 a_1 để có đoạn a_0 a_1 a_2 được sắp*
- *Tiếp tục cho đến khi thêm xong a_{n-1} vào đoạn a_0 a_1 ...a_{n-2} sẽ có a_0 a_1 ...a_{n-1} được sắp.*

Insertion Sort– Ý Tưởng

Input: mảng a: a[0], a[1],...,a[n-1] chưa sắp xếp

Output: mảng a đã sắp xếp

Các bước thực hiện:

for (i=1; i< n; i++) // giả sử có đoạn a_0 đã được sắp

{

b1: x = a[i] // x là phần tử cần chèn

b2: Tìm vị trí j thích hợp trong đoạn a[0] đến a[i-1] để chèn a[i] vào

b3: Dời chỗ các phần tử từ a[j] đến a[i-1] sang phải 1 vị trí để dành chỗ cho a[i]

b4: a[j] = x; // có đoạn a[0]..a[i] đã được sắp

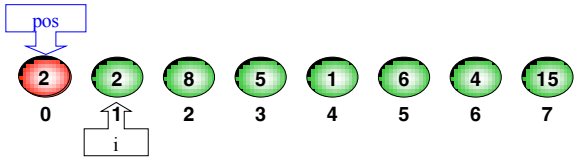
}

Insertion Sort – Minh Họa

12	2	8	5	1	6	4	15
0	1	2	3	4	5	6	7

Insertion Sort – Minh Họa

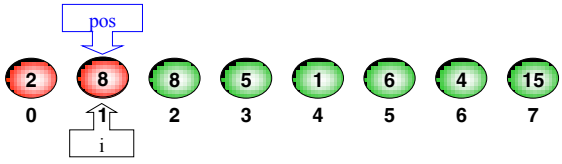
Insert a[1] into (0,0)



x

Insertion Sort – Minh Họa

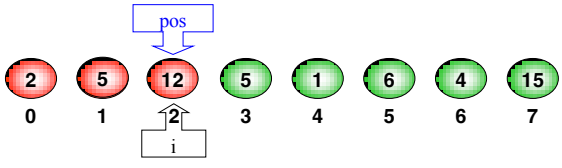
Insert a[2] into (0, 1)



x

Insertion Sort – Minh Họa

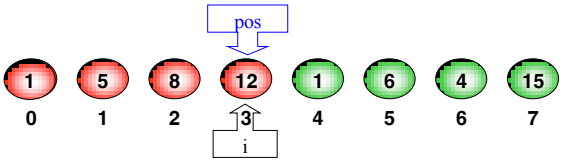
Insert a[3] into (0, 2)



x

Insertion Sort – Minh Họa

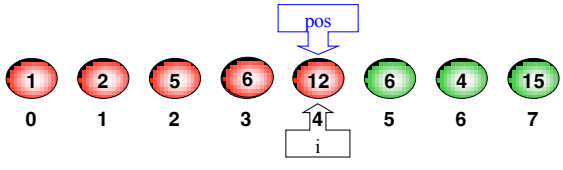
Insert a[4] into (0, 3)



x

Insertion Sort – Minh Họa

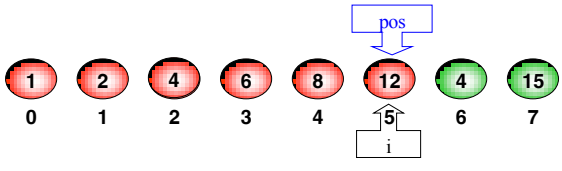
Insert a[5] into (0, 4)



x

Insertion Sort – Minh Họa

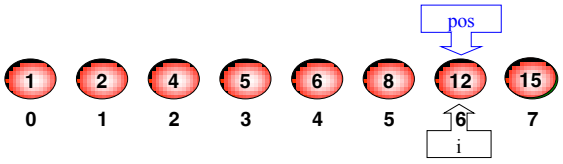
Insert a[6] into (0, 5)



x

Insertion Sort – Minh Họa

Insert a[8] into (0, 6)



x

Insertion Sort – Minh Họa

Insertion Sort – Cài đặt

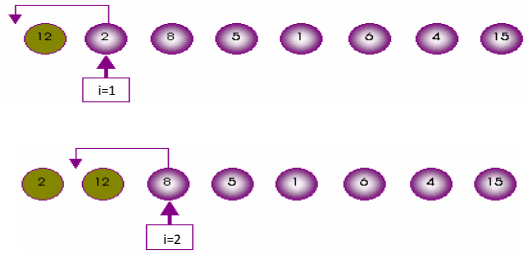
```
void InsertionSort (int a[], int n )
{  int  i, j, x;
   for (i=1 ; i<n ; i++)                    //đoạn a[0] đã sắp
   {
         x = a[i];                          //lưu giá trị a[i] tránh bị ghi đè khi dời chỗ
         // tìm vị trí chèn x kết hợp dời chỗ các phần tử sẽ đứng sau x trong dãy mới
         for (j = i-1, j >= 0 && x < a[j]; j -- )//chèn được
               a[j+1] = a[j];               //dời chỗ

         a[j+1] = x;                        // chèn x vào vị trí hợp lệ
   }
}
```

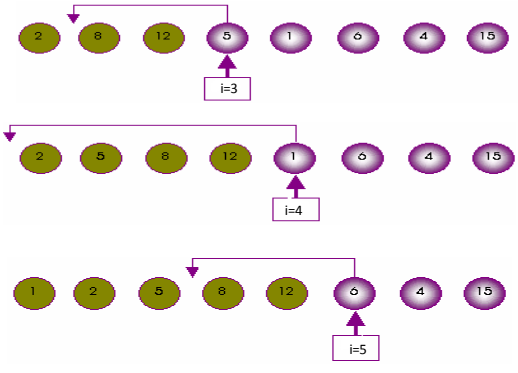
Insertion Sort – Minh Họa

➢Cho dãy số :

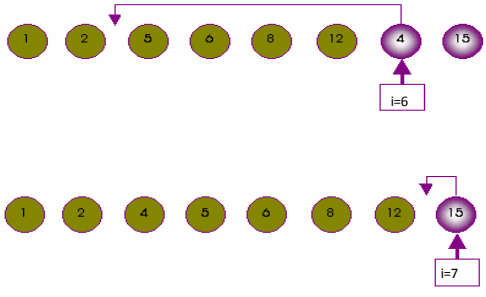
12 2 8 5 1 6 4 15



Insertion Sort – Minh Họa



Insertion Sort – Minh Họa



Insertion Sort– Độ Phức Tạp

- Có n-1 lần chèn
- Ở mỗi lần chèn, ta phải:
  1) Tìm kiếm vị trí chèn hợp lệ (vị trí j)
  2) Tuần tự dời các phần tử từ vị trí j trở đi xuống 1 vị trí
  3) Đưa phần tử cần chèn vào vị trí j

Insertion Sort– Độ Phức Tạp

Trường hợp	Số lần so sánh	Số lần đổi chỗ
Tốt nhất	*- Mỗi bước lặp chỉ cần 1 phép so sánh* $C(n) = n-1$	0
Xấu nhất	*- Có n-1 lần chèn* *- Số lần so sánh tối đa trong lần chèn thứ i là i* $C(n) <= 1+2+\ldots+n-1 = (n^2-n)/2$	*- Số lần đổi chỗ tối đa trong lần chèn thứ i là i* $M(n) <= 1+2+\ldots+n-1 = (n^2-n)/2$

Độ phức tạp tính toán : $T(n) = O(n^2)$

Insertion Sort– Bài Tập

Cho biết kết quả theo từng bước khi áp dụng thuật toán Insertion Sort sắp xếp dãy sau theo chiều tăng dần

45, 7, 12, 33, 21, 5, 2, 57, 15,

Binary Insertion Sort– Chèn nhị phân

Insertion Sort chèn vào một dãy đã sắp xếp !

Tìm kiếm chỗ để chèn?

Cải tiến Insertion Sort bằng cách tìm vị trí cần chèn bằng thuật tìm kiếm nhị phân, sau đó thực hiện việc chèn !

Binary Insertion Sort– Chèn nhị phân

```
void BInsertionSort(int a[],int n )
{
    int l,r,m,i;
    int x;//lưu giá trị a[i] tránh bị ghi đè khi dời chỗ các phần tử.
    for(int i=1 ; i<n ; i++)
    {       x = a[i]; l = 0; r = i-1;
            while(l<=r)  // tìm vị trí chèn x
            {       m = (l+r)/2;        // tìm vị trí thích hợp m
                    if(x < a[m]) r = m-1;
                    else    l = m+1;
            }
            for(int j = i-1 ; j >=l ; j--)
                    a[j+1] = a[j];// dời các phần tử sẽ đứng sau x
            a[l] = x;               // chèn x vào dãy
    }
}
```