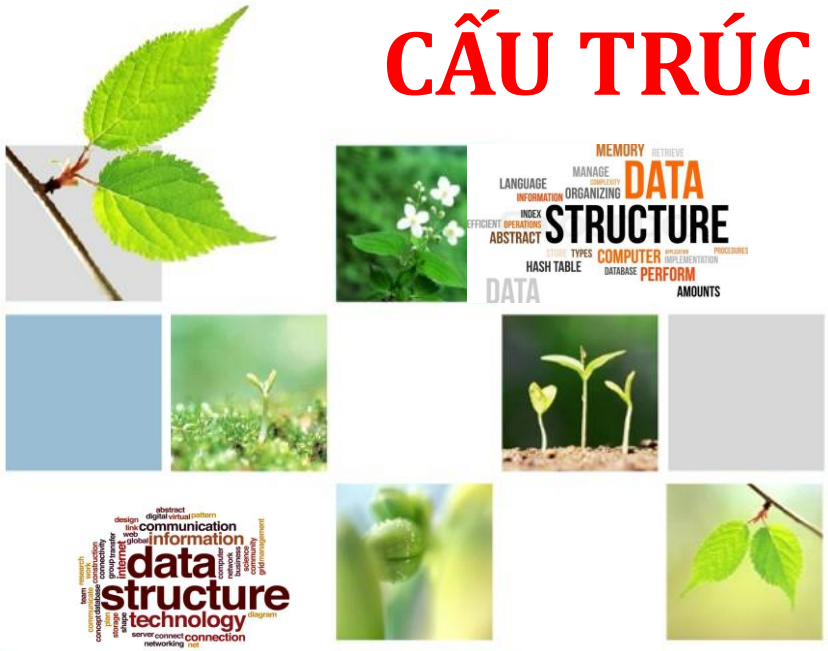
BUỔI 03: TÌM KIẾM & SẮP XẾP (tt)

LÝ THUYẾT

CẤU TRÚC DỮ LIỆU & GIẢI THUẬT

IT003.N210



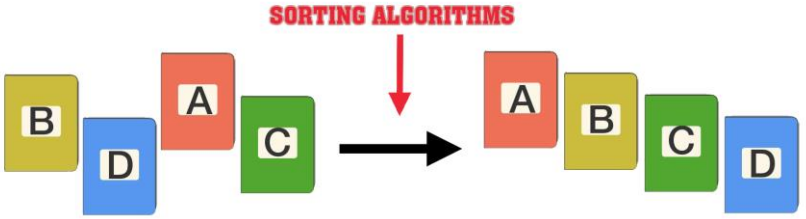
Giảng viên: Ths Đặng Văn Em

Email: vanem@uit.edu.vn

Điện thoại: 0966661006

CÁC GIẢI THUẬT SẮP XẾP:

1) Định nghĩa bài toán sắp xếp.
2) Phân loại sắp xếp
   a) Online vs Offline sorting
   b) Stable vs Unstable sorting
   c) Internal vs External sorting
3) Các thuật toán sắp xếp.
   a) Selection sort.
   b) Insert sort
   c) Counting sort
   d) Radix sort

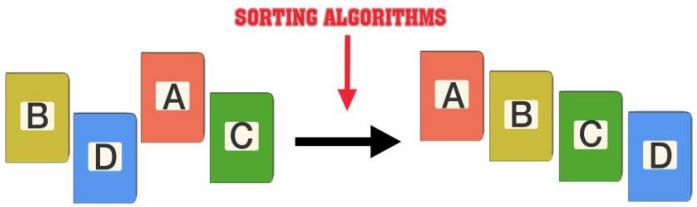


Định nghĩa bài toán sắp xếp:

Cho danh sách **A** gồm n phần tử $a_0, a_1, .., a_{n-1}$

Hoán đổi vị trí của các phần tử a_i và a_j sao cho đảm bảo thứ tự $\Re$ trong A, nghĩa là

$$a_i \Re a_j, \forall i < j$$

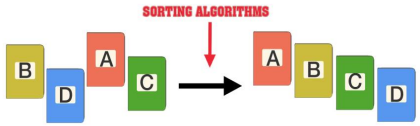


Phân loại sắp xếp:

Các giải thuật sắp xếp có thể phân loại theo nhiều tiêu chí:

a) Tính chất danh sách A

- Toàn bộ phần tử của A được xử lý đồng thời trong quá trình sắp xếp → **Offline Sorting**
  - ✓ Selection Sort
  - ✓ Bubble Sort
  - ✓ Quick Sort
  - ✓ ....



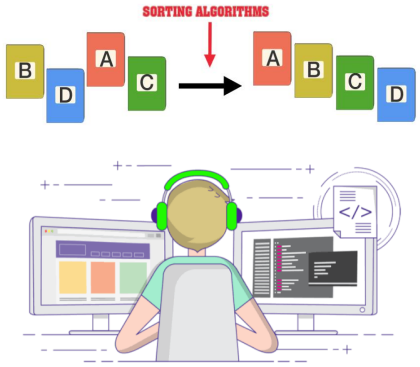
Phân loại sắp xếp:

Các giải thuật sắp xếp có thể phân loại theo nhiều tiêu chí:

a) Tính chất danh sách A

❖ Từng phần tử của A được sắp xếp tuần tự mà không cần biết trước toàn bộ A→ **Online Sorting**

- ✓ Insertion Sort
- ✓ Tree Sort (tạo Binary Search Tree)



Phân loại sắp xếp:

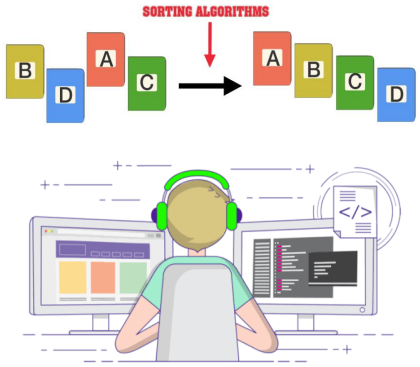
Các giải thuật sắp xếp có thể phân loại theo nhiều tiêu chí:

b) Trật tự của kết quả sắp xếp:

- Thứ tự trước/sau của các phần tử có cùng giá trị khóa không đổi so với ban đầu → **Stable Sorting**
- **Ví dụ:** Bubble Sort

  Trước khi sắp xếp: $A=\{1,5_1,2,5_2,3,5_3\}$

  Sau khi sắp xếp (tăng dần): $A=\{1,2,3,5_1,5_2,5_3\}$



Phân loại sắp xếp:

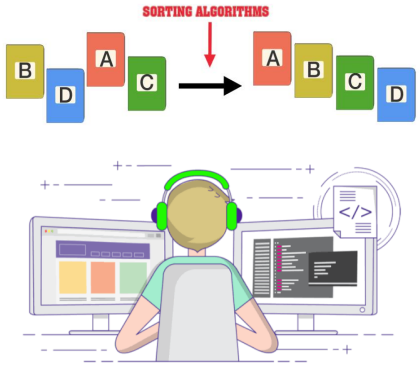
Các giải thuật sắp xếp có thể phân loại theo nhiều tiêu chí:

b) Trật tự của kết quả sắp xếp:

- Thứ tự trước/sau của các phần tử có cùng giá trị khóa thay đổi so với ban đầu → **Unstable Sorting.**
- Ví dụ: Interchange Sort:

  Trước khi sắp xếp: $A=\{1,5_1,2,5_2,3,5_3\}$

  Sau khi sắp xếp (tăng dần): $A=\{1,2,3,5_2,5_1,5_3\}$



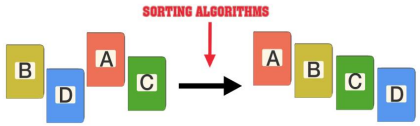
Phân loại sắp xếp:

Các giải thuật sắp xếp có thể phân loại theo nhiều tiêu chí:

c) Nơi lưu trữ chính của danh sách :

❖ Toàn bộ danh sách A được lưu trữ trên RAM trong quá trình sắp xếp → **Internal Sorting.**

- ✓ Interchange Sort
- ✓ Insertion Sort
- ✓ Quick Sort
- ✓ ...



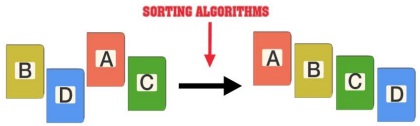
Phân loại sắp xếp:

Các giải thuật sắp xếp có thể phân loại theo nhiều tiêu chí:

c) Nơi lưu trữ chính của danh sách :

❖ Toàn bộ danh sách A được lưu trữ trên bộ nhớ ngoài (HDD) trong quá trình sắp xếp do kích thước danh sách quá lớn → **External Sorting**.

✓ Merge Sort



Phân loại sắp xếp:

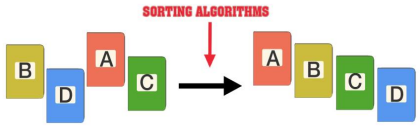
Các giải thuật sắp xếp có thể phân loại theo nhiều tiêu chí:

b) Tính chất danh sách A

Online vs Offline sorting

a) Stable vs Unstable sorting

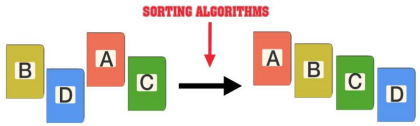
b) Internal vs External sorting



GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):

- **Từ khóa: Selection sort**
- **Phân tích:** Giả sử danh sách $A=\{a_0, a_1, .., a_{n-1}\}$ đã có thứ tự $\Re$.

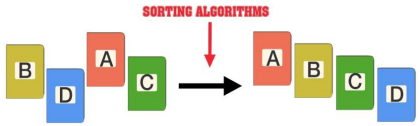
  Khi đó:
  - ✓ a_0 là phần tử nhỏ nhất theo $\Re$ trong A
  - ✓ a_1 là phần tử nhỏ nhất theo $\Re$ trong $A \setminus \{a_0\}$
  - ✓ a_2 là phần tử nhỏ nhất theo $\Re$ trong $A \setminus \{a_0, a_1\}$



GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):

❖ **Ý tưởng:**

- ✓ Chọn **phần tử nhỏ nhất** trong N phần tử trong dãy hiện hành ban đầu.
- ✓ Đưa phần tử này về **vị trí đầu dãy** hiện hành.
- ✓ Xem dãy hiện hành **chỉ còn N-1** phần tử của dãy hiện hành ban đầu
  - Bắt đầu từ **vị trí thứ 2**;
  - **Lặp lại quá trình trên** cho dãy hiện hành đến khi dãy hiện hành chỉ còn 1 phần tử



GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):

❖ **Thuật toán:**

Đầu vào: **A**={a_0, a_1, .., a_{n-1}} chưa có thứ tự $\Re$

Đầu ra: **A**={a_0, a_1, .., a_{n-1}} đã có thứ tự $\Re$

❖ **Các bước thực hiện:**

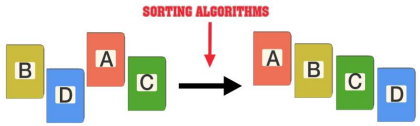
Bước 1: **i = 0**;

Bước 2: Tìm phần tử **a[min]** nhỏ nhất trong dãy hiện hành từ **a[i]** đến **a[N]** .

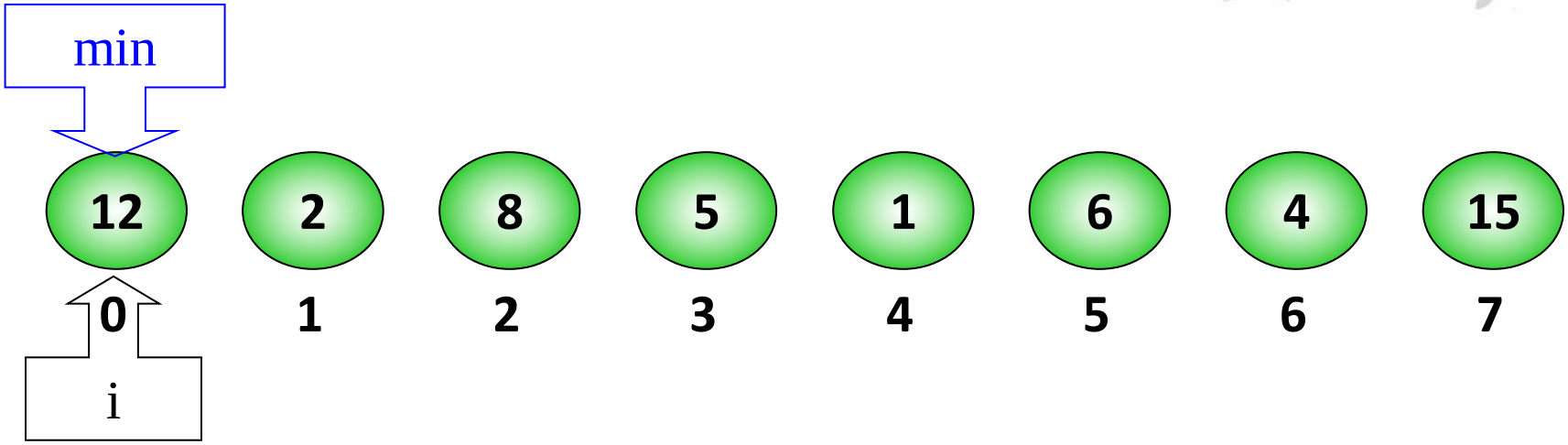
Bước 3: Đổi chỗ **a[min]** và **a[i]**

Bước 4: Nếu i < N-1 thì **i=i+1**; Lặp lại Bước 2;

Ngược lại: **Dừng**.



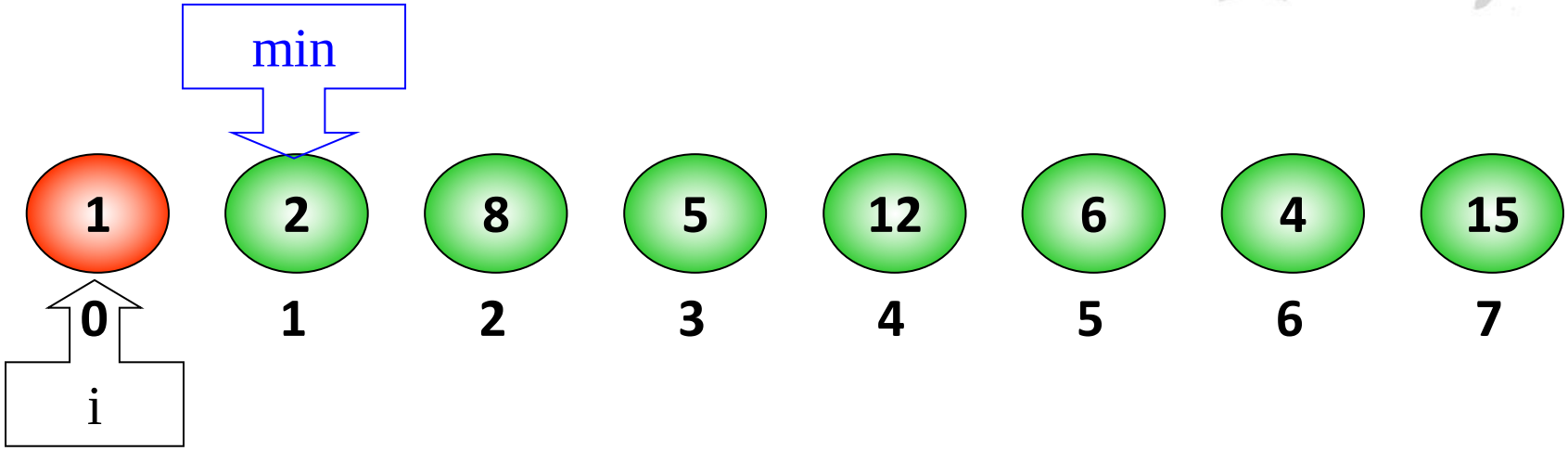
GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):



Vị trí nhỏ nhất(0,7)

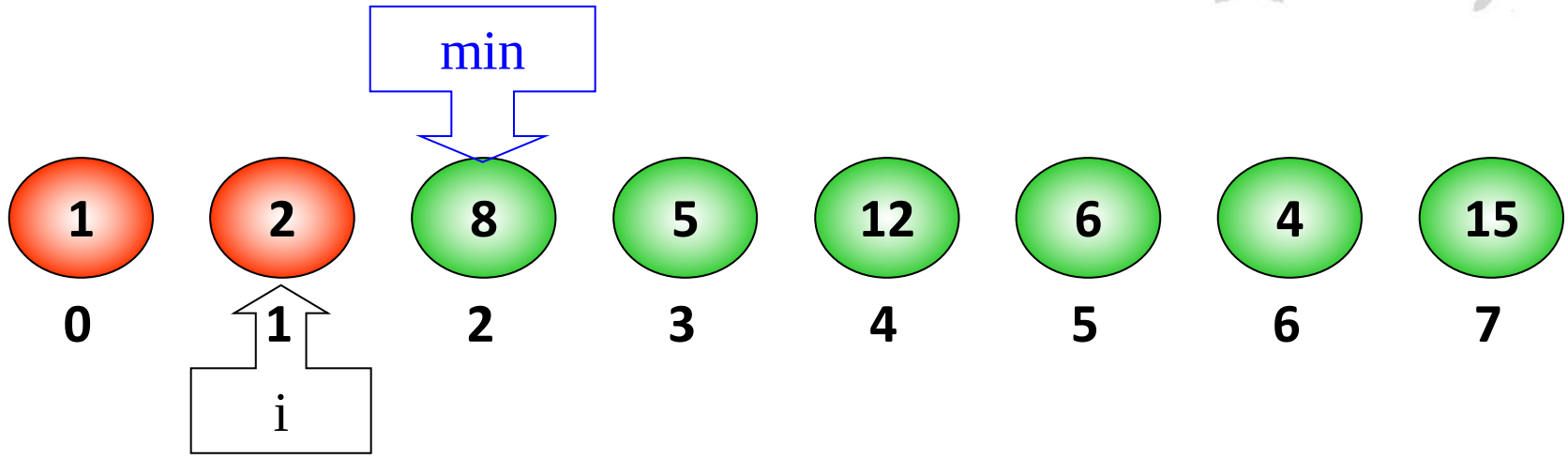
Swap(a[0], a[4])

GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):



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

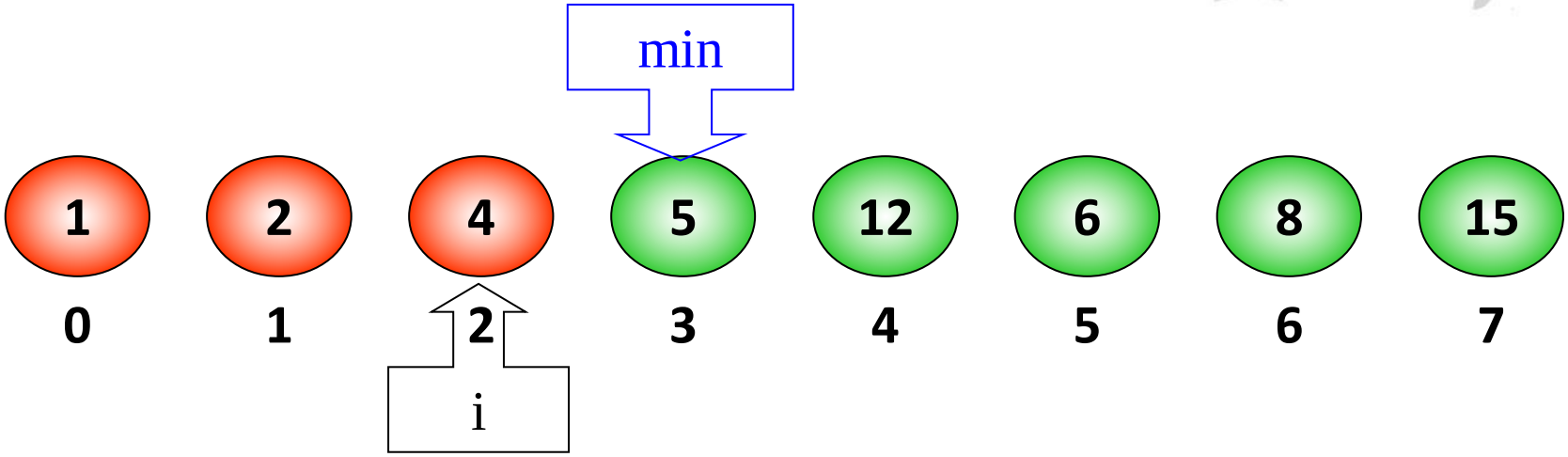
GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):



Vị trí nhỏ nhất(2,7)

Swap(a[2], a[6])

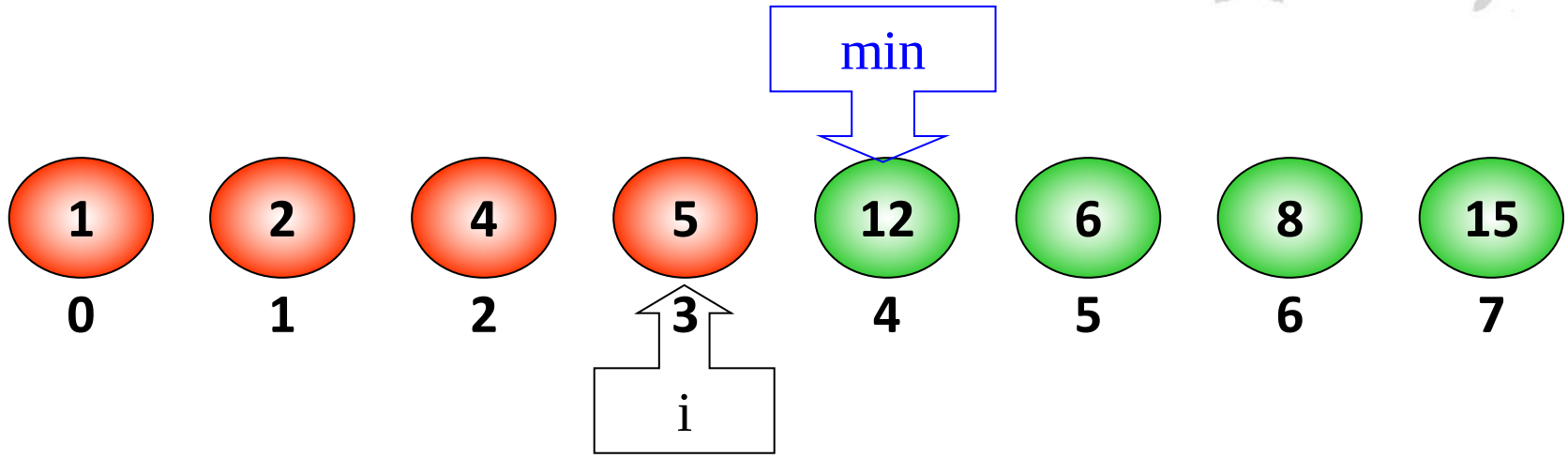
GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):



Vị trí nhỏ nhất(3, 7)

Swap(a[3], a[3])

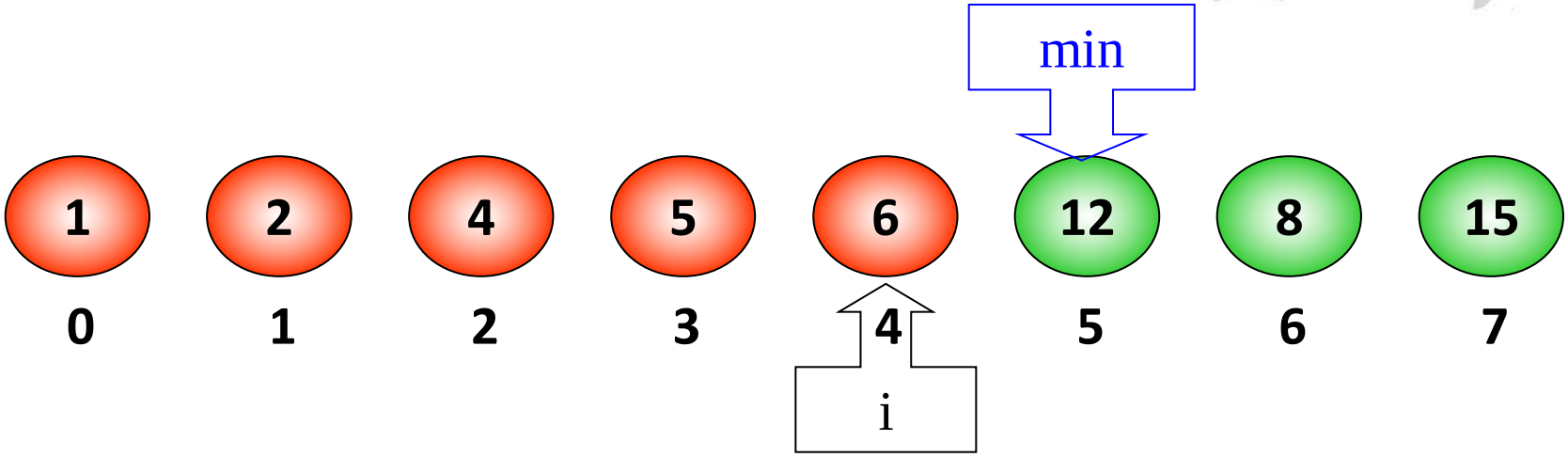
GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):



Vị trí nhỏ nhất(4, 7)

Swap(a[4], a[5])

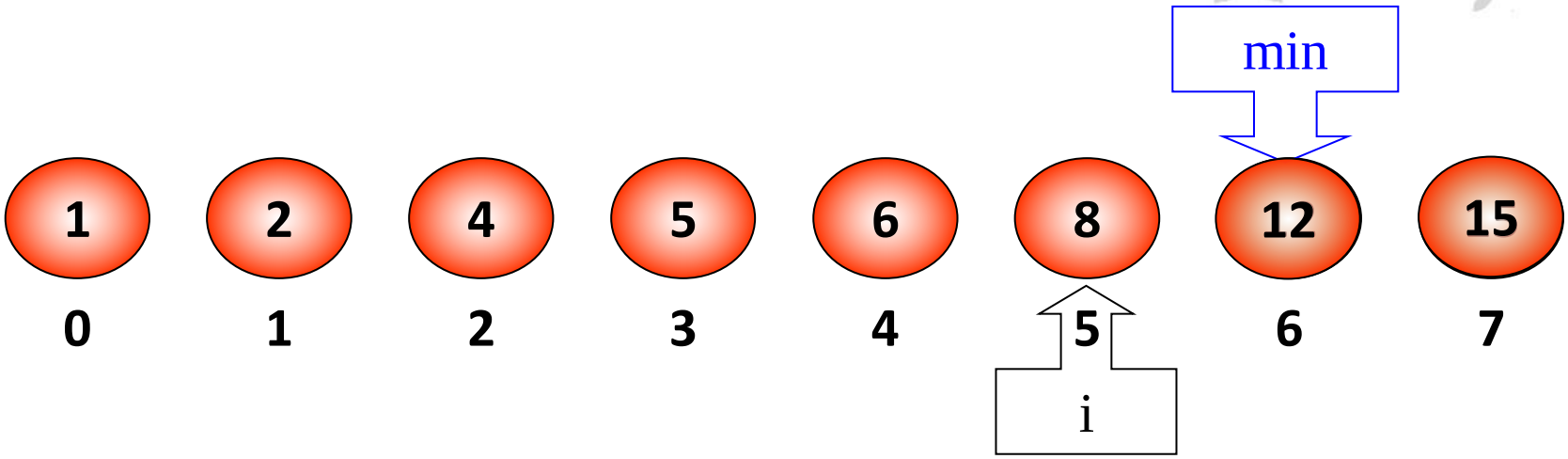
GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):



Vị trí nhỏ nhất(5,7)

Swap(a[5], a[6])

GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):



Vị trí nhỏ nhất(6, 7)

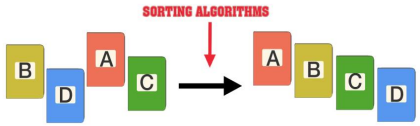
GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):

❖ Cài đặt minh họa:

```
void SelectionSort(int a[], int n)
{
    int min,i,j;
    for (i=0; i<n-1 ; i++)
    {
       min = i;
       for(j = i+1; j <N ; j++)
          if (a[j ] < a[min])
             min = j;
          Swap(a[min],a[i]);
    }
}
```

Ví dụ: Cho A[]

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



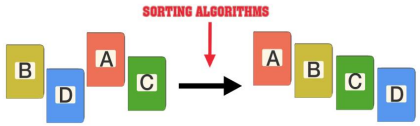
GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):

❖ Cài đặt minh họa:

```
void SelectionSort (List A)
{
   Node *min, *i, *j;
   i = A.pHead;
   while (i)
      {
         min = i; j = i->pNext;
         while (j)
         {
            if (j->info < min->info) min = j;
            j = j->pNext;
         }
         swap(i->info, min->info);
         i = i->pNext;
      }
}
```

Ví dụ: Cho danh sách đơn

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



GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):

❖ Thực hiện từng bước giải thuật sắp xếp chọn trực tiếp mảng

A[] = 3 5 7 11 13 17 19 23 29 31 1 41 43 47 53 59

A[] = 11 4 7 20 13 17 19 2 29 31 1 41 43 27 52 60

A[] = 10 6 7 20 13 17 19 21 29 31 1 41 42 27 5 60

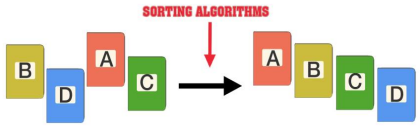
GIẢI THUẬT CHỌN TRỰC TIẾP (SELECTION SORT):

❖ **Đánh giá:**

	TỐT NHẤT (đúng thứ tự)	TRUNG BÌNH (chưa có thứ tự)	XẤU NHẤT (thứ tự ngược)
Theo phép so sánh	$O(n^2)$	$O(n^2)$	$O(n^2)$
Theo phép gán giá trị khóa	$O(n)$	$O(n)$	$O(n)$

GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

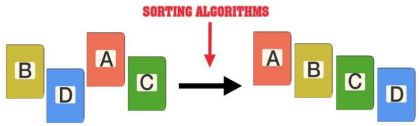
- **Từ khóa: Insert sort**
- **Phân tích:** Giả sử danh sách **A**={a_0, a_1, .., a_{n-1}} đã có thứ tự $\Re$. Khi đó, để tạo danh sách A có n+1 phần tử có thứ tự $\Re$, cần tìm vị trí **k** để chèn phần tử a_n sao cho đảm bảo $a_i \Re a_n$ $\forall i < k$



GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ Ý tưởng:

- ✓ Giả sử có một dãy **A**={a_0, a_1, .., a_{n-1}} trong đó **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 dãy mới a_0 , a_1,... ,a_i trở nên có thứ tự. Vị trí này chính là vị trí giữa hai phần tử a_{k-1} và a_k thỏa $a_{k-1} < a_i < a_k$ (1≤k≤i)

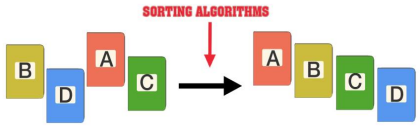


GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ **Giải thuật:**

Đầu vào: $A=\{a_0, a_1, .., a_{n-1}\}$ chưa có thứ tự $\Re$

Đầu ra: $A=\{a_0, a_1, .., a_{n-1}\}$ đã có thứ tự $\Re$



GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ **Các bước thực hiện:**

Bước 1: **i = 1**;

Bước 2: x = a[i]; *//Tìm vị trí pos thích hợp trong đoạn* ***a[1] đến a[i-1]*** *để chèn a[i] vào*

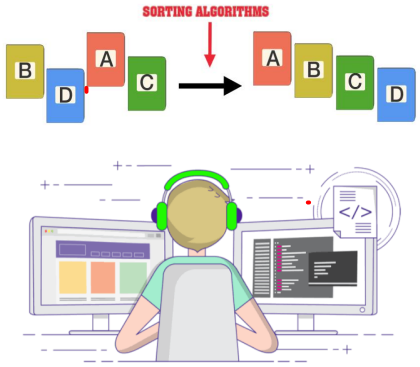
Bước 3: Dời chỗ các phần tử từ **a[pos] đến a[i-1]** sang phải 1 vị trí để dành chổ cho a[i]

Bước 4: a[pos] = x; *//có đoạn a[1]..a[i] đã được sắp*

Bước 5: i = i+1;

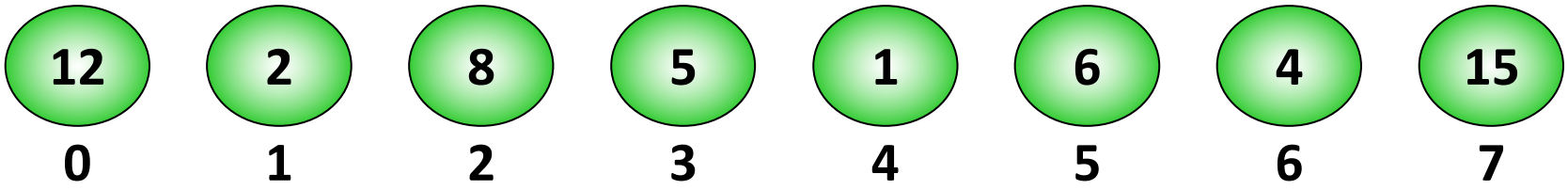
Nếu i < n : Lặp lại Bước 2

Ngược lại: **Dừng**



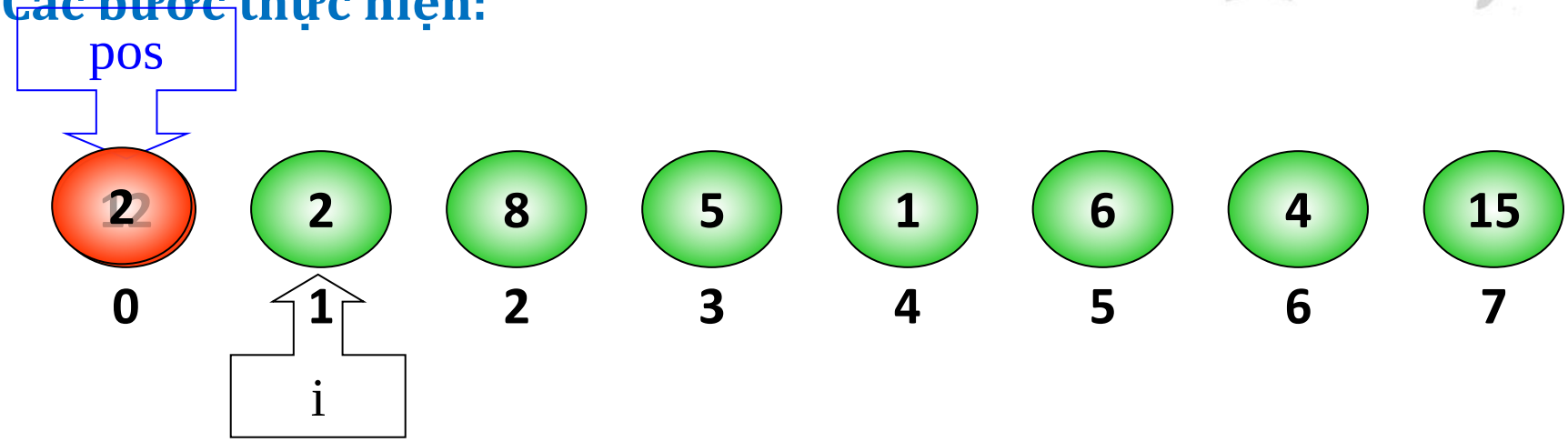
GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ **Các bước thực hiện:**



GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ **Các bước thực hiện:**

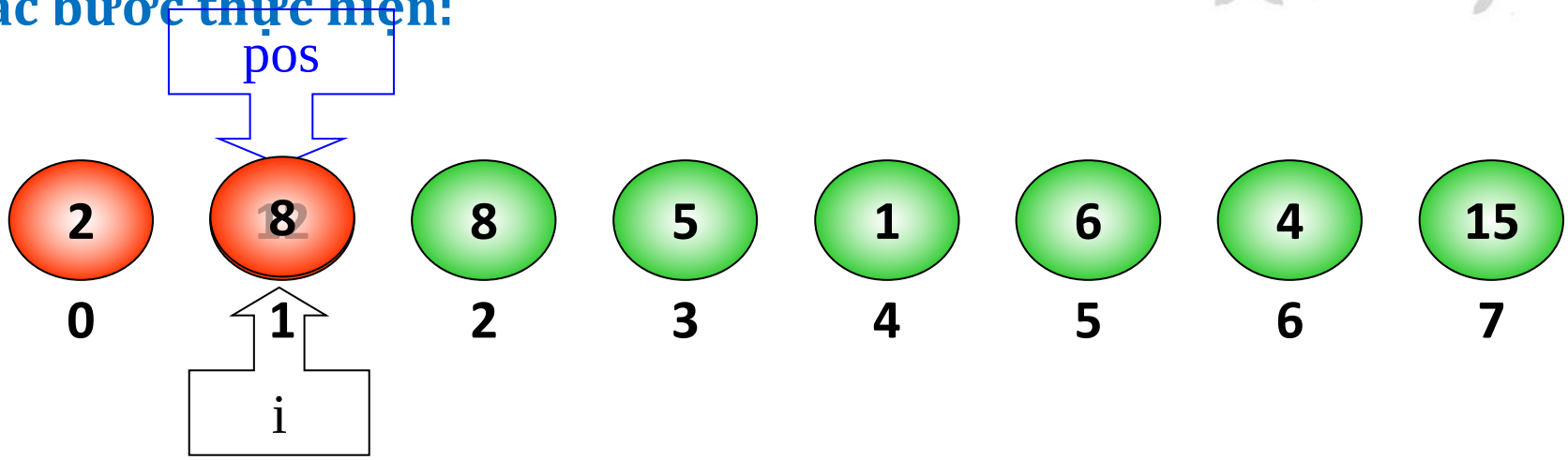


X

Insert a[1] into (0,0)

GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ **Các bước thực hiện:**

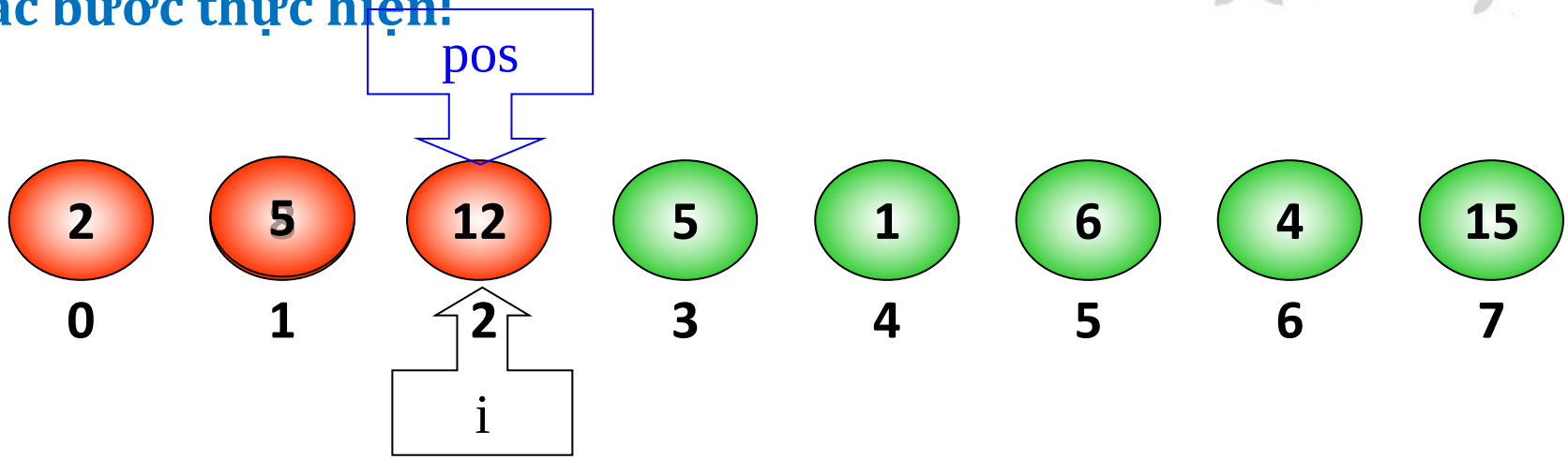


X

Insert a[2] into (0, 1)

GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ **Các bước thực hiện:**

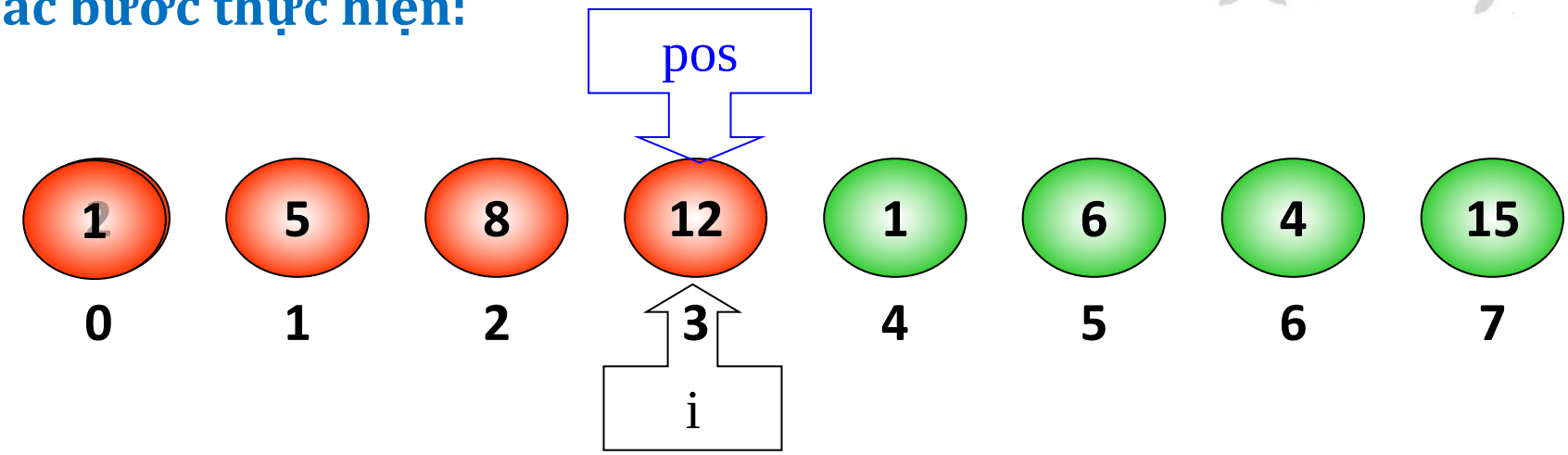


X

Insert a[3] into (0, 2)

GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ **Các bước thực hiện:**

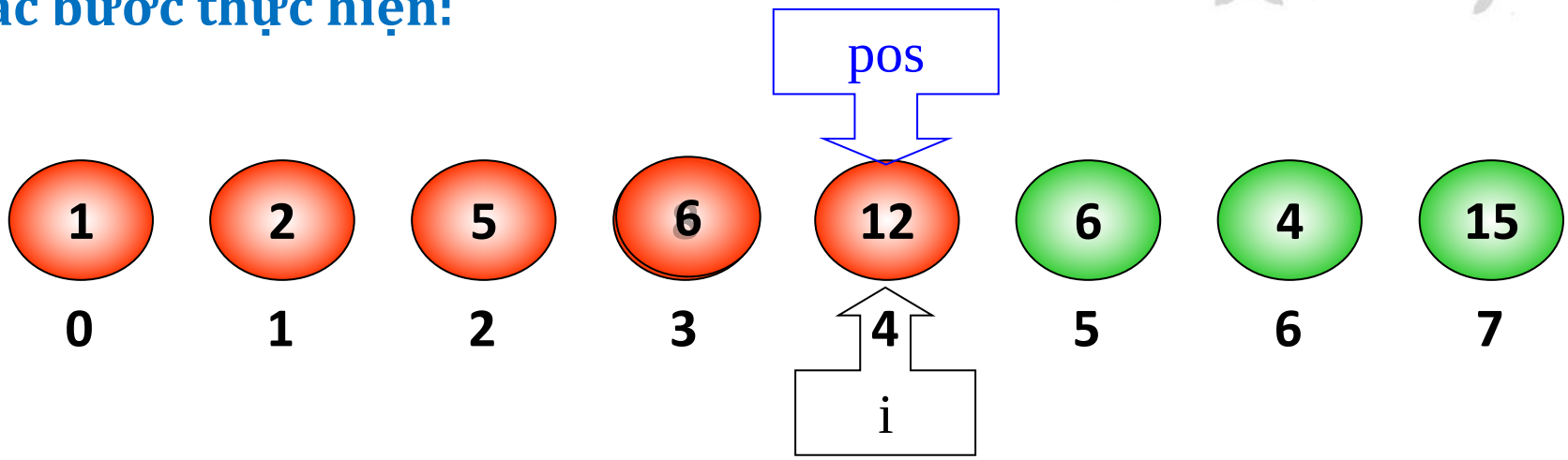


X

Insert a[4] into (0, 3)

GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ **Các bước thực hiện:**

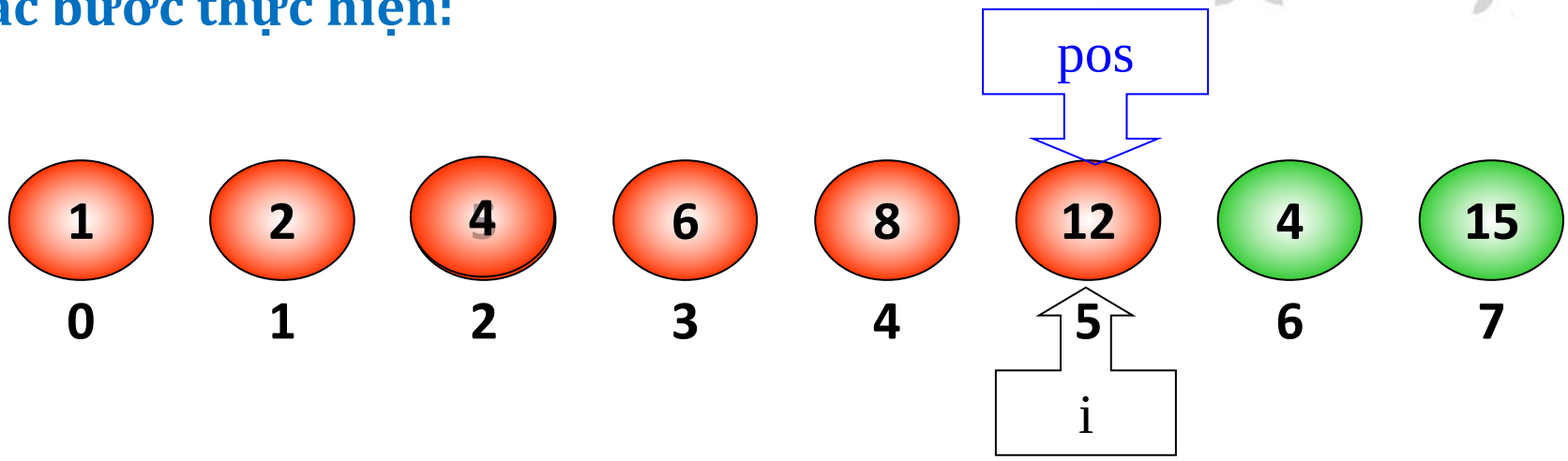


X

Insert a[5] into (0, 4)

GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ **Các bước thực hiện:**

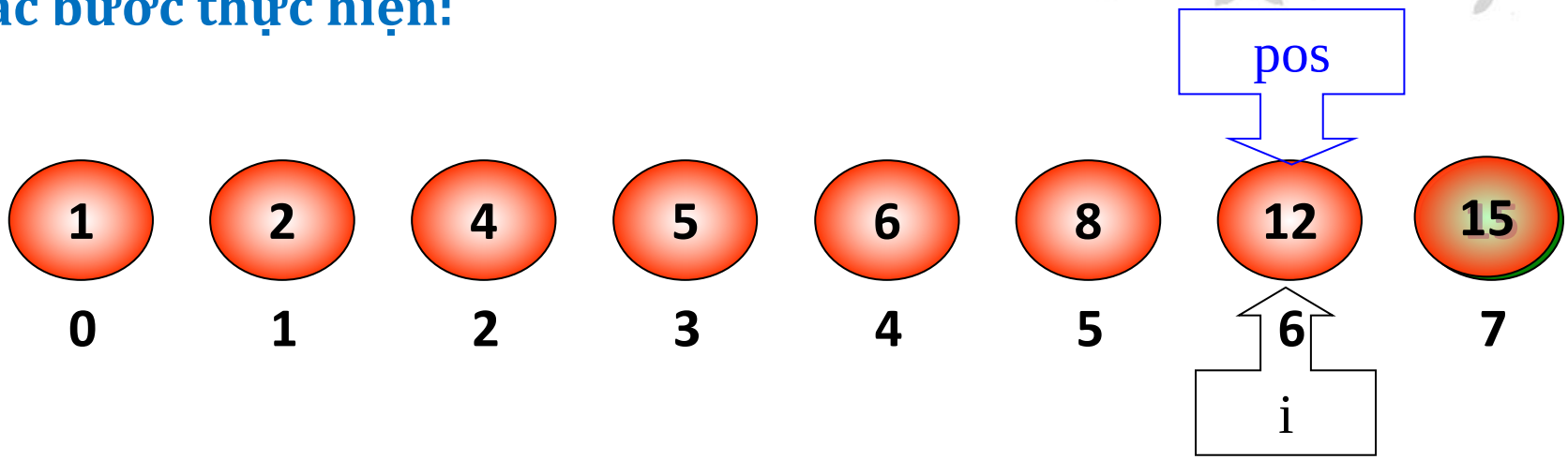


X

Insert a[6] into (0, 5)

GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ **Các bước thực hiện:**

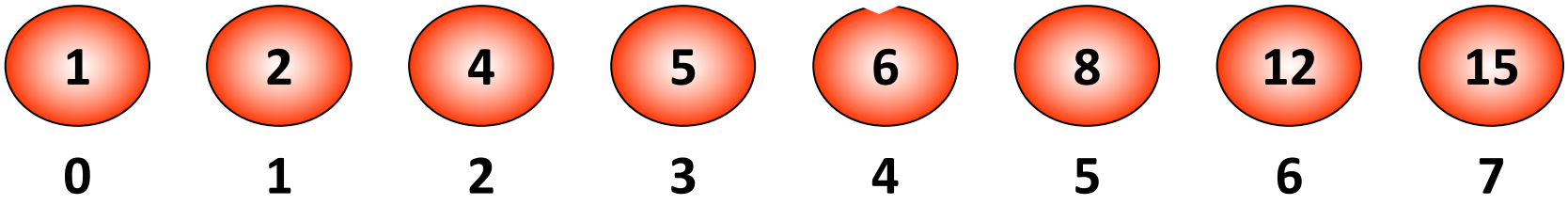


X

Insert a[8] into (0, 6)

GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ **Các bước thực hiện:**



GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ Cài đặt minh họa:

```
void InsertionSort(int a[], int n )
{
   for(int i=1 ; i<n ; i++)
   {
      int x = a[i]; int pos = i-1;
      while((pos >= 0)&&(a[pos] > x))
      {
         a[pos+1] = a[pos];
         pos--;
      }
      a[pos+1] = x];
   }
}
```

Ví dụ: Mảng A[]

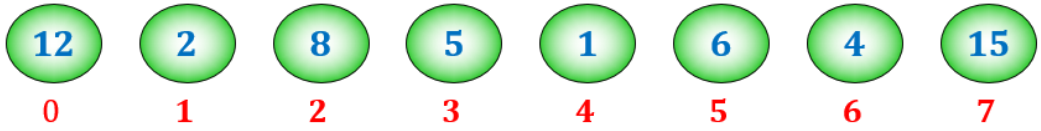
12	2	8	5	1	6	4	15
0	1	2	3	4	5	6	7

GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ Cài đặt minh họa:

```
void insertionSort(List &A)
{
   Node *i = A.pHead, *k, *e;
   while (i->pNext)
   {
        q = NULL; k = A.pHead;
        e=removeAfter(A,i);
        while (k != i->pNext)
        {
           if (!(k->info < e->info)) break;
           q = k; k = q->pNext;
        }
        addAfter(A, e, q);
        if (i->pNext == e)
        i = i->pNext;
   }
}
```

Ví dụ: Cho danh sách đơn



GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ Thực hiện từng bước giải thuật sắp xếp chọn trực tiếp mảng

A[] = 23 29 31 3 5 7 11 13 17 9 1 41 43 47 53 59

A[] = 10 6 7 20 13 17 9 21 29 11 1 41 42 27 5 60

GIẢI THUẬT SẮP XẾP CHÈN TRỰC TIẾP (INSERT SORT):

❖ **Đánh giá:**

	TỐT NHẤT (đúng thứ tự)	TRUNG BÌNH (chưa có thứ tự)	XẤU NHẤT (thứ tự ngược)
Theo phép so sánh	**O(n)**	**$O(n^2)$**	**$O(n^2)$**
Theo phép gán giá trị khóa	**O(1)**	**$O(n^2)$**	**$O(n^2)$**

GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

- **Từ khóa:** **Counting sort**
- **Điều kiện:** Giá trị khóa là số nguyên dương, không âm có giá trị lớn nhất không quá lớn.
- **Phân tích:** Nếu giá trị khóa là số nguyên và có thể cấp phát mảng với kích thước bằng giá trị lớn nhất → chỉ cần đếm số lượng giá trị khóa xuất hiện trong danh sách A.

GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Ý tưởng:

- Giá trị khóa của a_i là chỉ số của mảng **B** có **k** phần tử (**k** là giá trị khóa lớn nhất của **A**)
- Quá trình sắp xếp danh sách **A** là đếm số phần tử của mỗi chỉ số của **B** trong **A**. Từ đó tính ra thứ tự của các khóa a_i trong **A.**
- Kết quả sắp xếp có được bằng cách lấy vị trị của phần tử **A** được lưu trong **B**

GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Giải thuật:

- **Bước 1:** Đếm số lần xuất hiện của từng phần tử trong mảng cần sắp xếp A[]. Kết quả được lưu vào mảng C[]
- **Bước 2:** Xem xét sửa đổi giá trị của C[]. C[i] thể hiện giới hạn trên của chỉ số của phần tử i sau khi sắp xếp.
- **Bước 3:** Duyệt qua từng phần tử của A[] và đặt nó vào đúng chỉ số của mảng chứa các giá trị đã sắp xếp B[] dựa vào C[].

GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Minh họa giải thuật:

Giả sử A={3,2,5,1,4} và thứ tự cấp sắp xếp **<** (tăng dần)

A	3	2	5	1	4

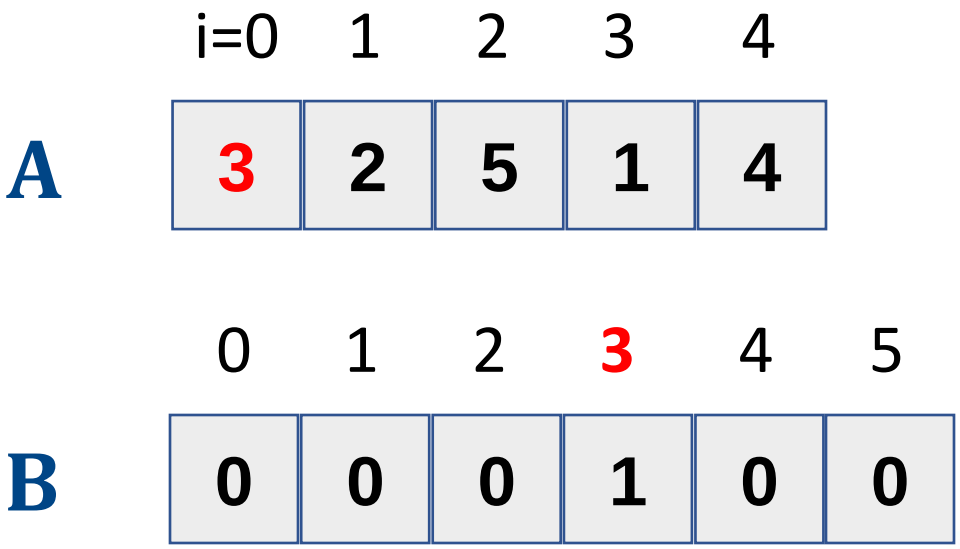
	0	1	2	3	4	5
B	0	0	0	0	0	0

Đếm các khóa

GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Minh họa giải thuật:

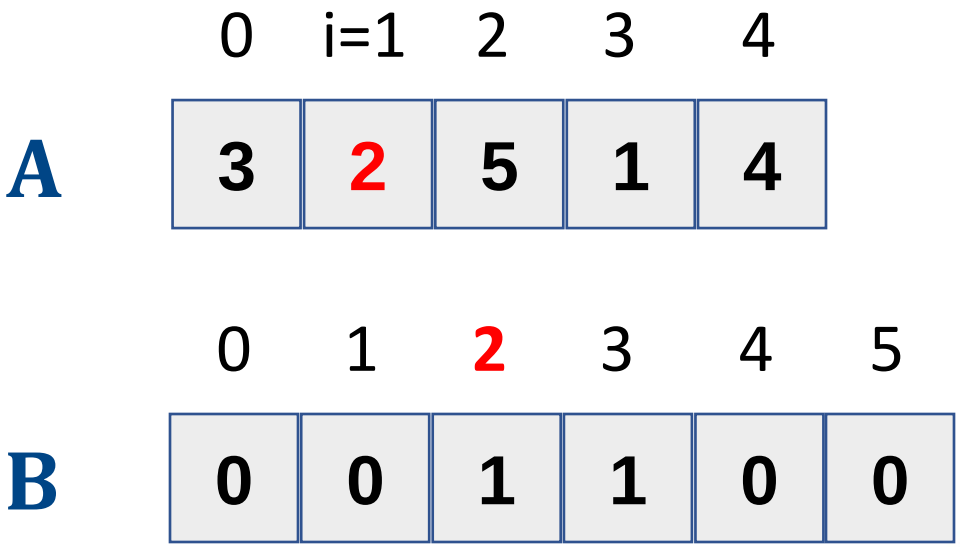
Giả sử A={3,2,5,1,4} và thứ tự cấp sắp xếp < (tăng dần)



GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Minh họa giải thuật:

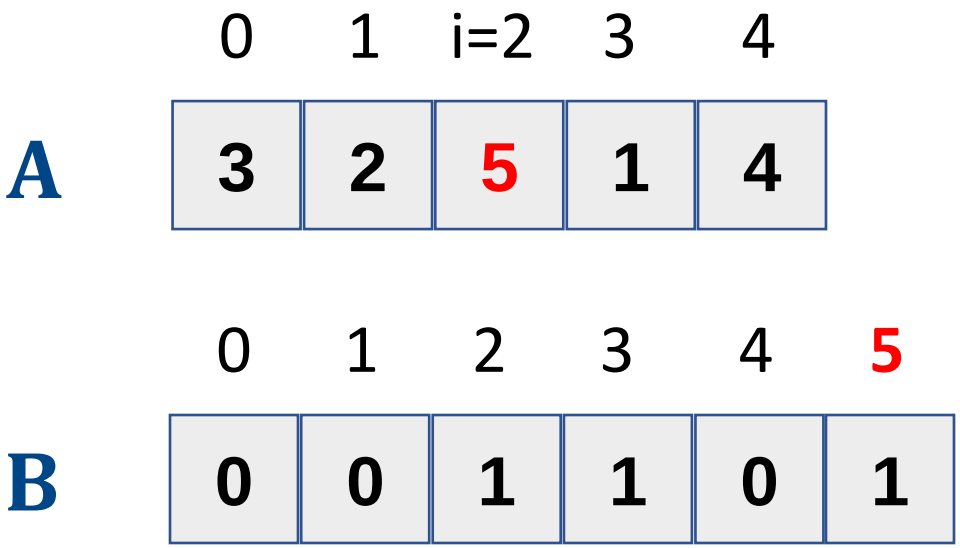
Giả sử A={3,2,5,1,4} và thứ tự cấp sắp xếp **<** (tăng dần)



GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Minh họa giải thuật:

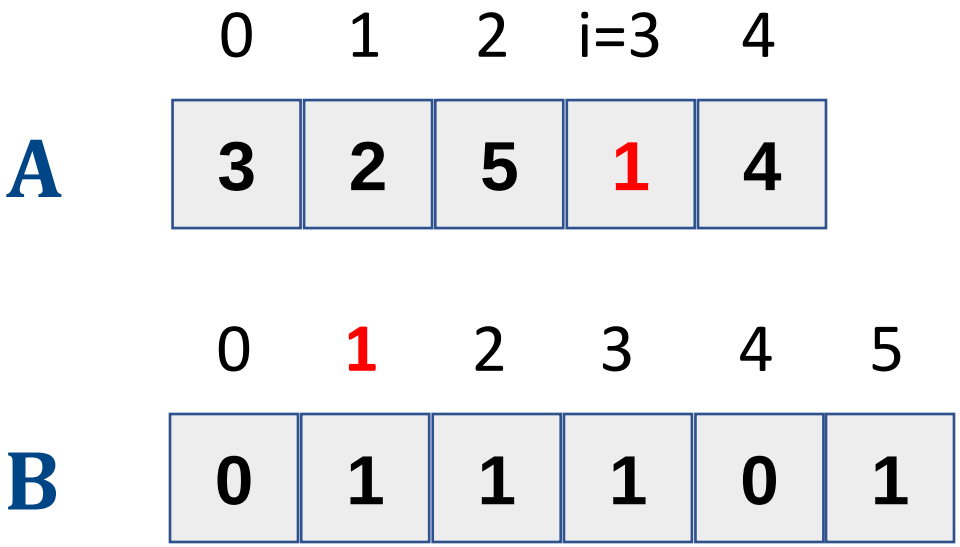
Giả sử A={3,2,5,1,4} và thứ tự cấp sắp xếp **<** (tăng dần)



GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Minh họa giải thuật:

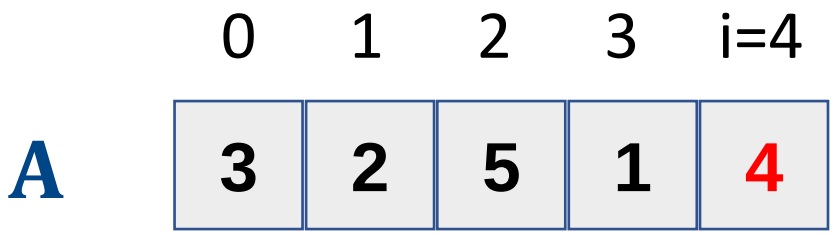
Giả sử A={3,2,5,1,4} và thứ tự cấp sắp xếp **<** (tăng dần)

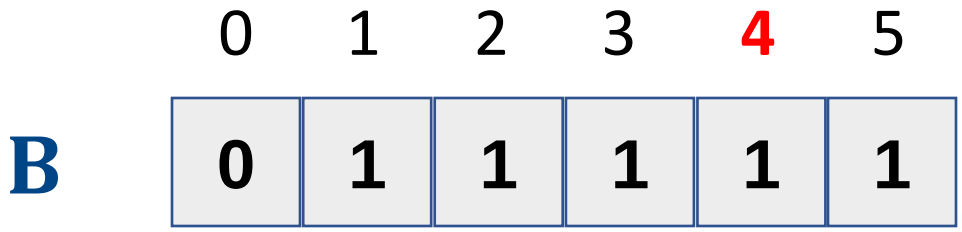


GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Minh họa giải thuật:

Giả sử A={3,2,5,1,4} và thứ tự cấp sắp xếp **<** (tăng dần)





Tính toán thứ tự

GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Minh họa giải thuật:

Giả sử A={3,2,5,1,4} và thứ tự cấp sắp xếp **<** (tăng dần)

	0	1	2	3	4
A	**3**	**2**	**5**	**1**	**4**

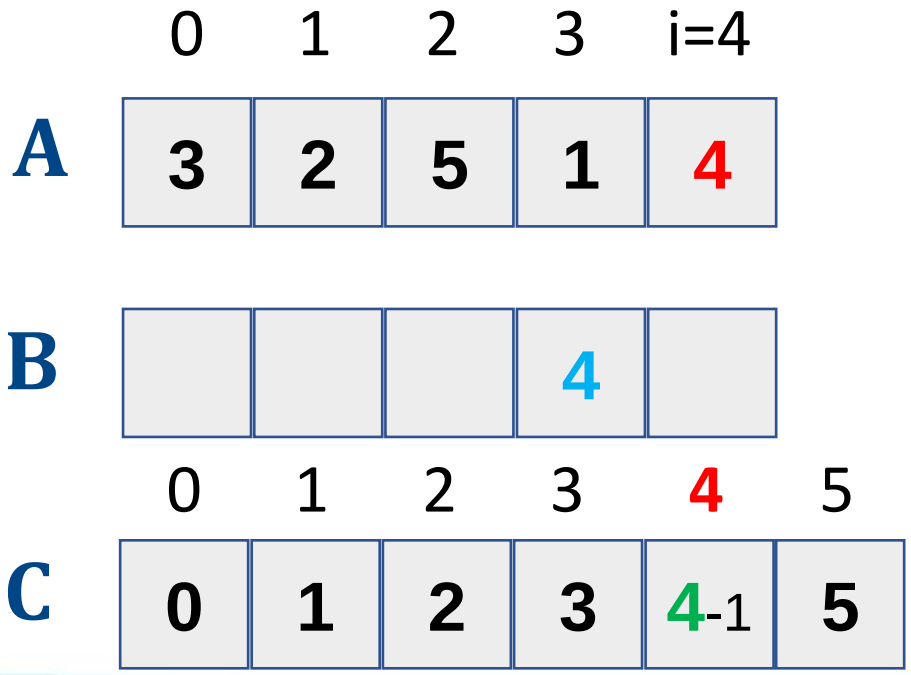
	0	1	2	3	4	5
B	**0**	**1**	**2**	**3**	**4**	**5**

Ghi kết quả

GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Minh họa giải thuật:

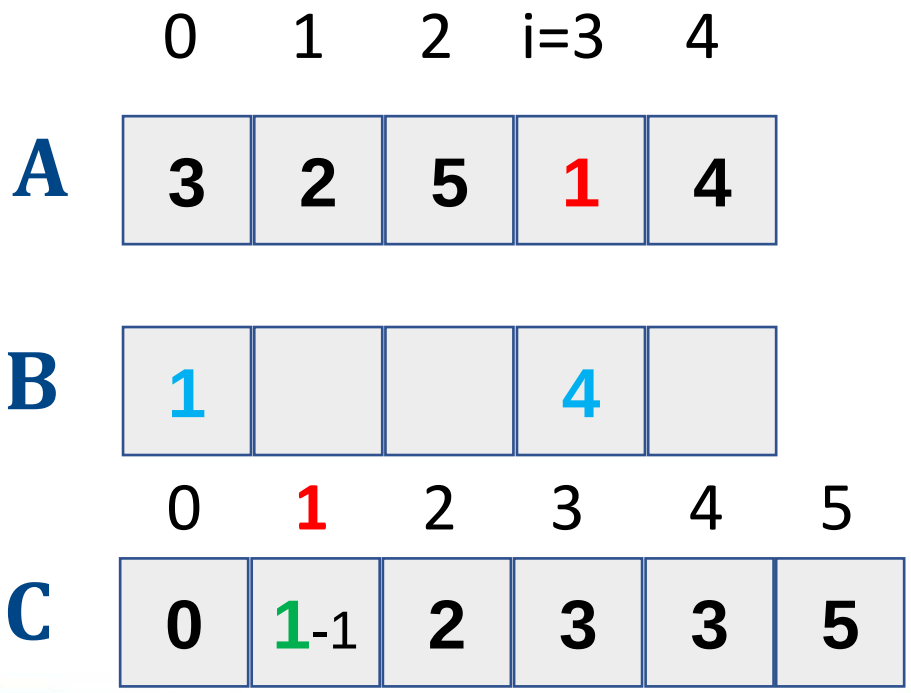
Giả sử A={3,2,5,1,4} và thứ tự cấp sắp xếp < (tăng dần)



GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Minh họa giải thuật:

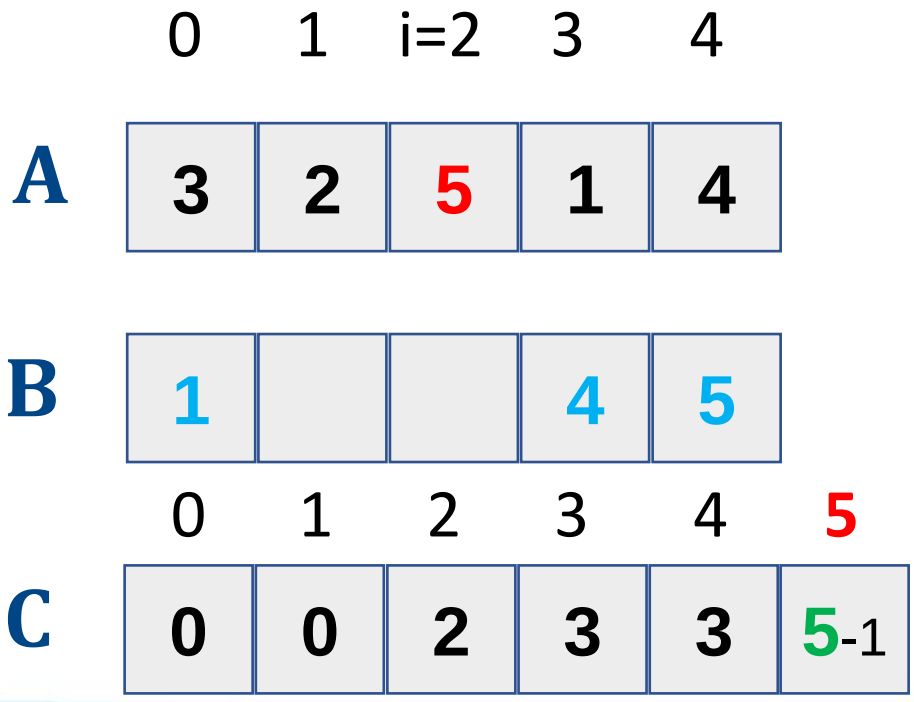
Giả sử A={3,2,5,1,4} và thứ tự cấp sắp xếp < (tăng dần)



GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Minh họa giải thuật:

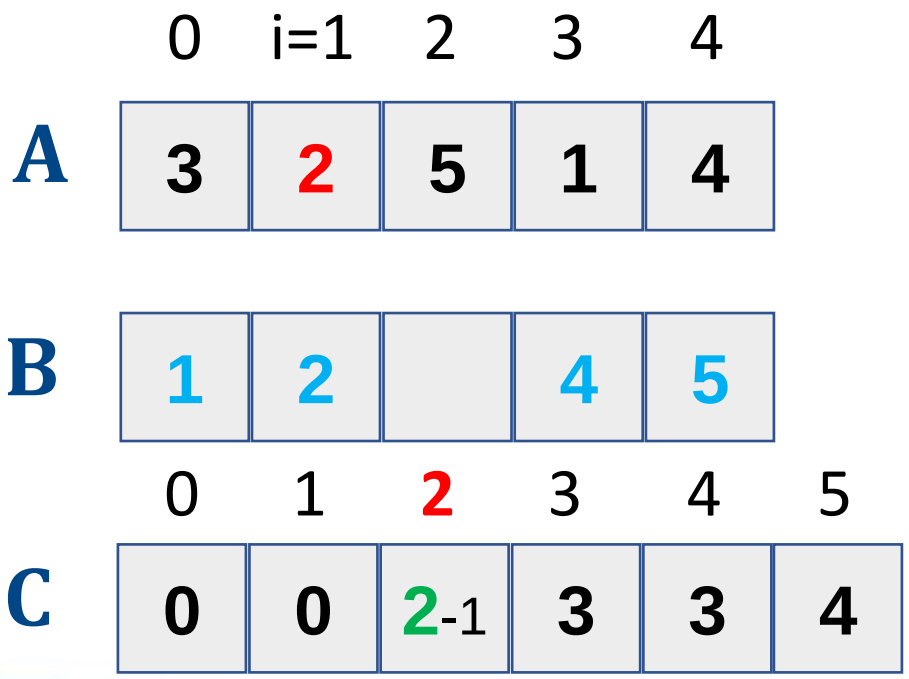
Giả sử A={3,2,5,1,4} và thứ tự cấp sắp xếp < (tăng dần)



GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Minh họa giải thuật:

Giả sử A={3,2,5,1,4} và thứ tự cấp sắp xếp < (tăng dần)



GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Minh họa giải thuật:

Giả sử A={3,2,5,1,4} và thứ tự cấp sắp xếp < (tăng dần)

	i=0	1	2	3	4
A	3	2	5	1	4

	1	2	3	4	5
B					

	0	1	2	3	4	5
C	0	0	1	3-1	3	4

GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Cài đạt giải thuật minh họa:

```
void counting_sort(int input[], int n)
{
   int output[n];
   int max = input[0];
   int min = input[0];

   for(int i = 1; i < n; i++)
   {
      if(input[i] > max)
         max = input[i];
      else if(input[i] < min)
         min = input[i];
```

GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Cài đạt giải thuật minh họa:

```
    }
    int k = max - min + 1;
    int count_array[k];
    for(i=0; i<k; i++)
       count_array[i]=0;
    for(i = 0; i < n; i++)
       count_array[input[i] - min]++;
    for(i = 1; i < k; i++)
       count_array[i] += count_array[i - 1];
    for(i = 0; i < n; i++)
    {
       output[count_array[input[i] - min] - 1] = input[i];
       count_array[input[i] - min]--;
    }
    for(i = 0; i < n; i++)
       input[i] = output[i];
}
```

GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

- Thực hiện từng bước giải thuật sắp xếp chọn trực tiếp mảng

A[] = 3 11 5 7 11 1 4 6 7 5 9

A[] = 11 4 7 2 3 7 9 2 8 1 1 14 6

GIẢI THUẬT SẮP XẾP ĐẾM (COUNTING SORT):

Đánh giá:

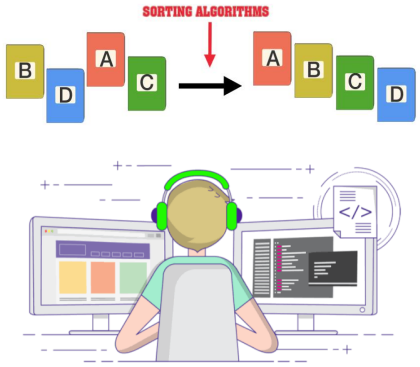
- Trong mọi trường hợp, độ phức tạp tính toán của Counting Sort là **O(n+k)**, trong đó **k** là kích thước của mảng **B**.

- Counting Sort là một trong những thuật toán sắp xếp không dựa vào kết quả so sánh giá trị khóa của các phần tử trong danh sách.

GIẢI THUẬT SẮP XẾP THEO CƠ SỐ (RADIX SORT):

Ý tưởng:

- Radix sort (Sắp xếp cơ số) là một thuật toán sắp xếp không so sánh . Nó tránh so sánh bằng cách tạo và phân phối các phần tử vào các nhóm theo cơ số của chúng .
- Sắp xếp cơ số còn được gọi là sắp xếp theo nhóm và sắp xếp kỹ thuật số



GIẢI THUẬT SẮP XẾP THEO CƠ SỐ (RADIX SORT):

Thuật toán Radix Sort:

- Sắp xếp các số theo chữ số đơn vị: sắp xếp các số theo giá trị chữ số đơn vị (từ phải sang trái) của chúng.
- Sắp xếp các số theo chữ số hàng chục: sắp xếp các số theo giá trị chữ số hàng chục của chúng.
- Lặp lại cho tới khi các số được sắp xếp hoàn toàn