ĐẠI HỌC QUỐC GIA TPHCM
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

CHƯƠNG II

TÌM KIẾM VÀ SẮP XẾP

Nguyễn Trọng Chỉnh
chinhnt@uit.edu.vn

GIẢI THUẬT SẮP XẾP

- KHÁI NIỆM
- PHƯƠNG PHÁP SELECTION SORT
- PHƯƠNG PHÁP INSERTION SORT

GIẢI THUẬT SẮP XẾP

❖KHÁI NIỆM

*** Sắp xếp** là quá trình xử lý một danh sách sao cho các phần tử trong danh sách thỏa một quan hệ thứ tự R nào đó.

*** Nghịch thế**: giả sử có danh sách A chứa các phần tử $a_0, a_1, .. a_{n-1}$ và một quan hệ thứ tự R cần thiết lập trên A, khi đó a_j R a_i nếu i < j là một nghịch thế.

Ví dụ: Cho A = {1, 2, 9, 4, 5, 6}, quan hệ thứ tự R là ≤, khi đó, 9 và 4 là một nghịch thế vì 4 ≤ 9 và vị trí của 4 là j = 3 và vị trí của 9 là i = 2, i < j.

GIẢI THUẬT SẮP XẾP

❖KHÁI NIỆM

*** Dãy chưa có thứ tự**: là dãy chứa nghịch thế.

*** Nguyên tắc sắp xếp**: hoán vị các phần tử của dãy sao cho dãy không còn chứa nghịch thế.

Ví dụ: Cho A = {1, 2, 9, 6, 5, 4}, quan hệ thứ tự R là $\leq$, số nghịch thế là 6. Quá trình sắp xếp như sau:

- Đổi chổ A[2] và A[5], A = {1, 2, 4, 6, 5, 9} số nghịch thế: 1.
- Đổi chổ A[3] và A[4], A = {1, 2, 4, 5, 6, 9} số nghịch thế: 0.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SELECTION SORT

Selection Sort, hay còn gọi là chọn trực tiếp

*** Ý tưởng**: Cho một dãy A={a_i}, i=0,..,n-1. A có thứ tự tăng dần nếu a_j là min(a_j, a_{j+1}, .., a_{n-1})

Ví dụ: Dãy A = {5, 3, 4, 1, 2} có:

A[0] = 1 = min(5, 3, 4, 1, 2). A = {1, 3, 4, 5, 2}

A[1] = 2 = min(3, 4, 5, 2). A = {1, 2, 4, 5, 3}

A[2] = 3 = min(4, 5, 3). A = {1, 2, 3, 4, 5}

A[3] = 4 = min(4, 5). A = {1, 2, 3, 4, 5}

A[4] = 5 = min(5). A = {1, 2, 3, 4, 5}

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SELECTION SORT

*** Giải thuật**: (theo ngôn ngữ tự nhiên)

Đầu vào: mảng A gồm n phần tử chưa có thứ tự.

Đầu ra: mảng A gồm n phần tử đã có thứ tự.

+ B1: i ← 0
+ B2: j ← i + 1, min ← i
+ B3: nếu A[min] > A[j] thì min ← j
+ B4: nếu j < n thì j ← j + 1 đến B3.
+ B5: hoán đổi A[i] với A[min]
+ B6: nếu i < n - 1 thì i ← i + 1, đến B2.
+ B7: kết thúc.

B3, B4, B5: tìm min của A[i], A[i+1], .., A[n – 1]

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SELECTION SORT

*** Giải thuật**: (theo mã giả)

Đầu vào: mảng A gồm n phần tử chưa có thứ tự.

Đầu ra: mảng A gồm n phần tử đã có thứ tự.

```
for i ← 0 to n – 2
    min ← i
    for j ← i + 1 to n – 1
        if A[min] > A[j] min ← j
    hoandoi(A[min], A[i])
```

GIẢI THUẬT SẮP XẾP

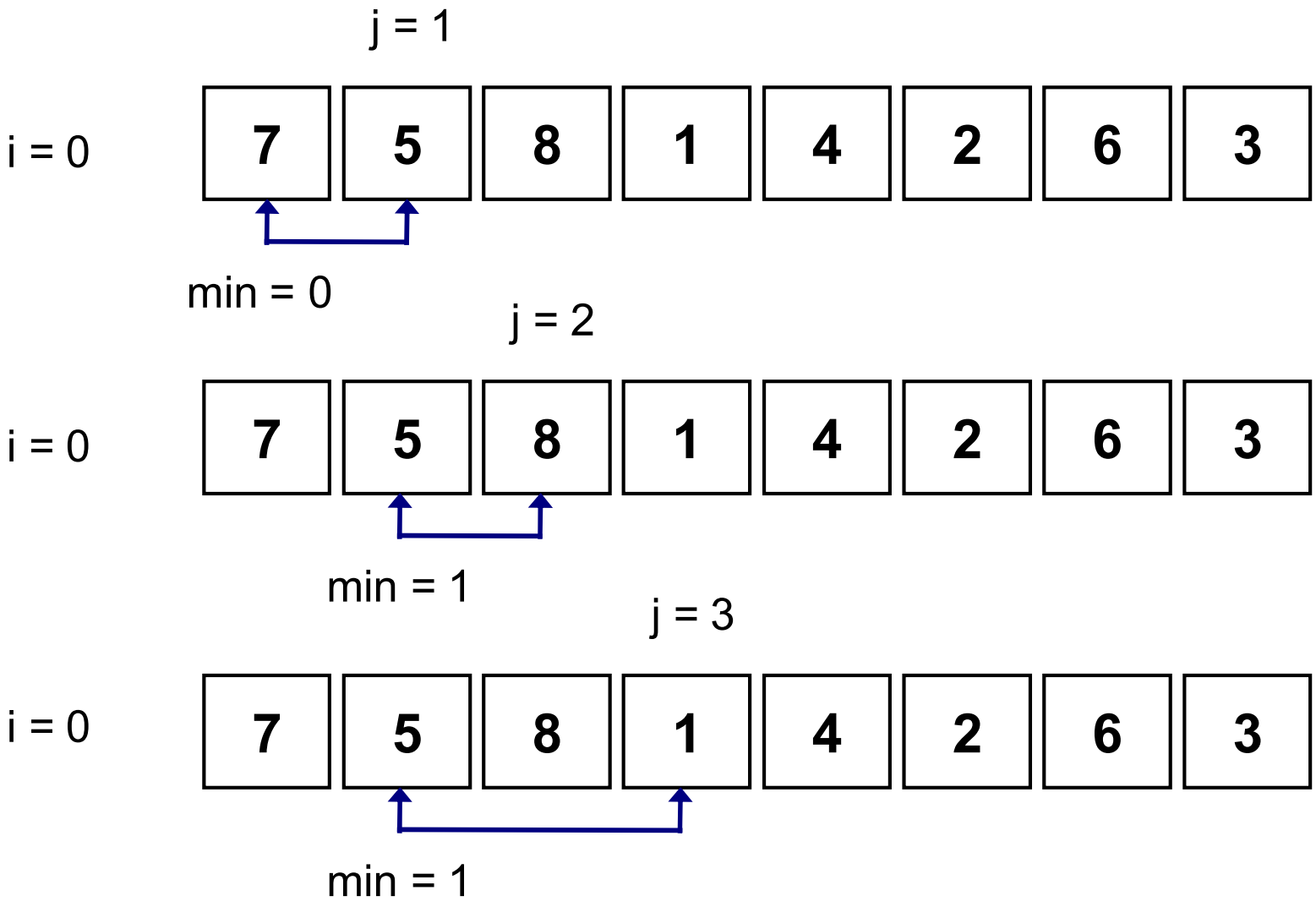
❖PHƯƠNG PHÁP SELECTION SORT

*** Cài đặt**:

```
void hoandoi(int &a, int &b) { int c = a; a = b; b = c;}
void SelectionSort(int a[], int n) {
int min, i;
for (i=0; i<n-1; i++) {
    min = i;
    for (int j=i+1; j<n; j++)
        if (a[min] > a[j]) min = j;
    hoandoi(a[i], a[min]);
}
```

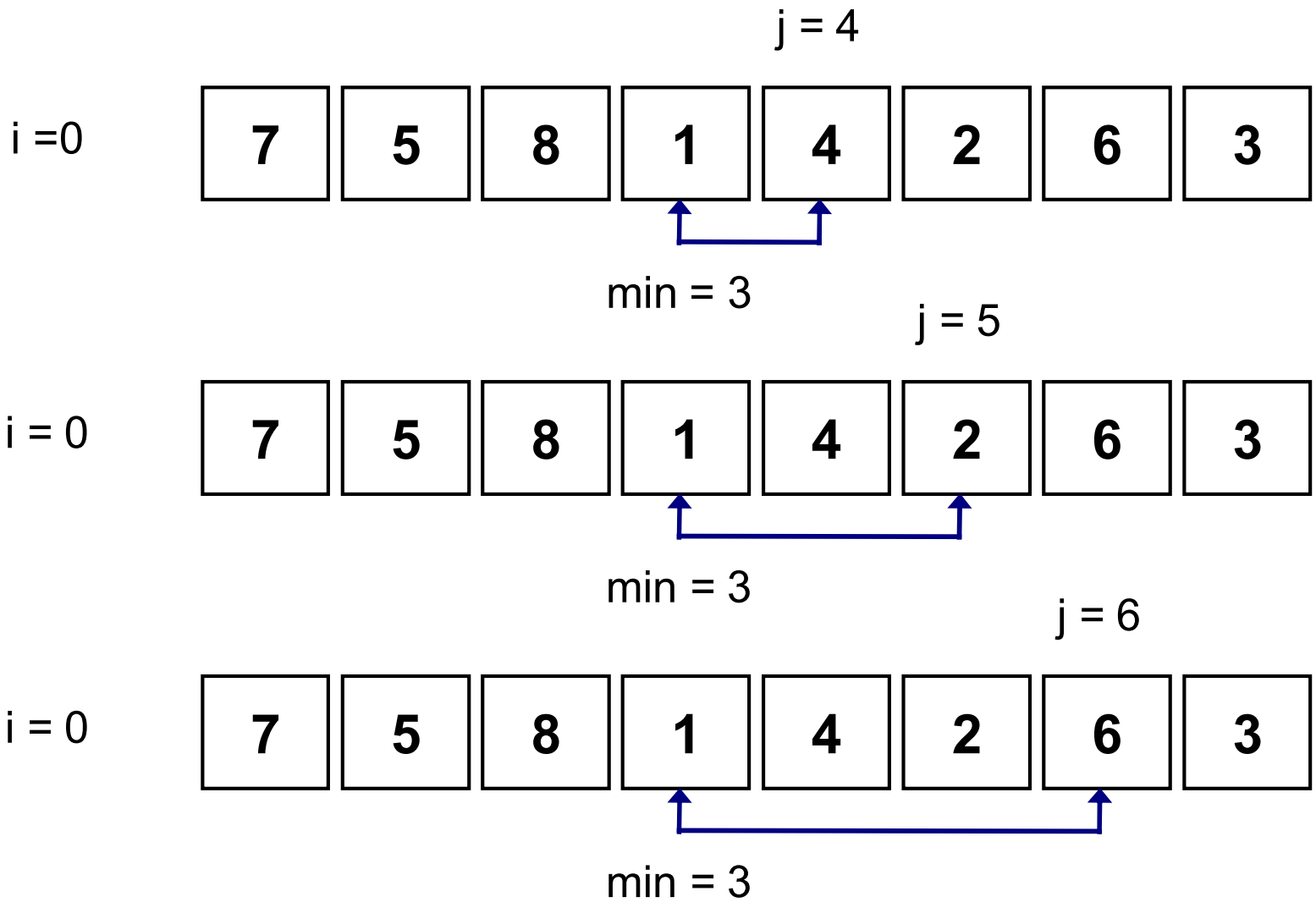
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SELECTION SORT



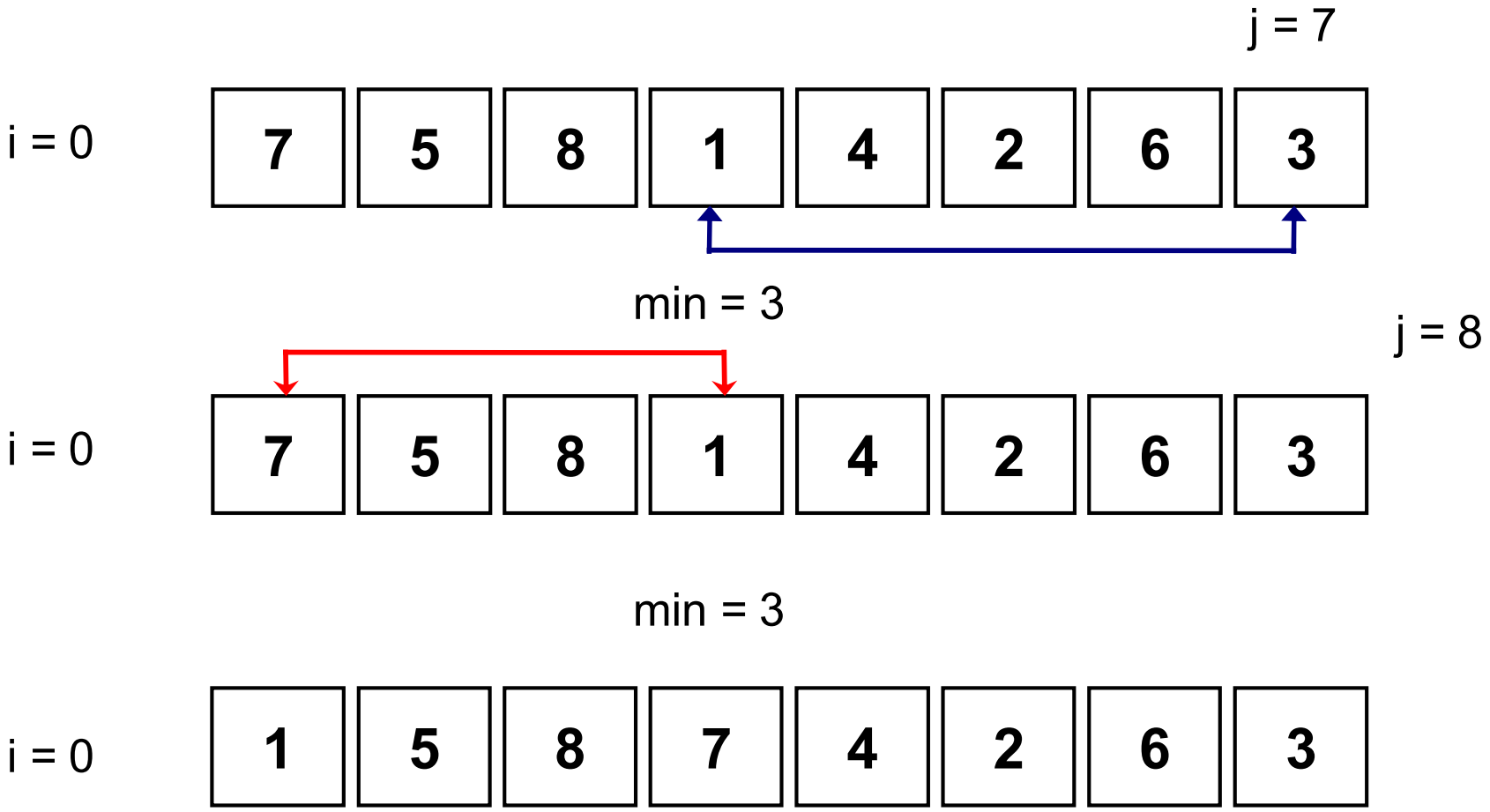
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SELECTION SORT



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SELECTION SORT



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SELECTION SORT

i = 1	**1**	5	8	7	4	2	6	3
i = 2	**1**	**2**	8	7	4	5	6	3
i = 3	**1**	**2**	**3**	7	4	5	6	8
i = 4	**1**	**2**	**3**	**4**	7	5	6	8
i = 5	**1**	**2**	**3**	**4**	**5**	7	6	8
i = 6	**1**	**2**	**3**	**4**	**5**	**6**	7	8
i = 7	**1**	**2**	**3**	**4**	**5**	**6**	**7**	8

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SELECTION SORT

*** Đánh giá theo trường hợp xấu nhất**:

Xét số phép so sánh giá trị khóa: Bỏ qua các phép tính để thực hiện vòng lặp, ta có:

- Số phép tính để tìm min trong đoạn [i, n - 1]: n - i - 1
- Số lần lặp tìm min: n - 1
- $T(n) = \Sigma(n - i - 1), i = 0,..,n-2$

  $= (n-1)(n-1) - (n-2)(n-1)/2$

  $= n(n-1)/2$

Độ phức tạp tính toán theo số phép so sánh là $O(n^2)$.

GIẢI THUẬT SẮP XẾP

❖ PHƯƠNG PHÁP SELECTION SORT

*** Đánh giá theo trường hợp xấu nhất**:

Xét số phép gán giá trị khóa: Bỏ qua các phép tính để thực hiện vòng lặp, ta có:

- Số phép tính để hoán vị 3
- Số lần lặp hoán vị: n - 1
- $T(n) = \Sigma 3, i = 0,..,n-2$

  $= 3(n-1)$

Độ phức tạp tính toán theo phép gán là O(n).

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SELECTION SORT

Câu hỏi:

Cho biết trường hợp xấu nhất và tốt nhất nếu dùng thuật toán Chọn trực tiếp (Selection sort) để sắp xếp tang dần là trường hợp dãy A có đặc điểm như thế nào? Cho biết ước lượng số phép gán và phép so sánh trong hai trường hợp này?

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SELECTION SORT

Bài tập:

Trình bày ý tưởng và giải thuật để sắp xếp một dãy số nguyên theo thứ tự giảm dần theo phương pháp Chọn trực tiếp

Áp dụng: trình bày từng bước quá trình sắp xếp theo giải thuật đã nêu cho dãy số sau:

A = {5, 9 ,7 ,2, 6, 3, 1, 8}

GIẢI THUẬT SẮP XẾP

Ý tưởng: chọn phần tử i (i = 0..n-2) là min của các số A[i], A[i+1], ..., A[n-2].

Thuật toán:

Đầu vào: Mảng A gồm n phần tử chưa có thứ tự giảm dần

Đầu vào: Mảng A gồm n phần tử có thứ tự giảm dần.

```
for i ← 0 to n – 2
    max ← i
    for j ← i + 1 to n – 1
        if A[max] < A[j] max ← j
    hoandoi(A[max], A[i])
```

GIẢI THUẬT SẮP XẾP

Quá trình sắp xếp dãy A = {5, 9 ,7 ,2, 6, 3, 1, 8} theo thứ tự giảm dần:

Lần 1: i = 0, max = 1, A = {9, 5, 7, 2, 6, 3, 1, 8}

Lần 2: i = 1, max = 7, A = {9, 8, 7, 2, 6, 3, 1, 5}

Lần 3: i = 2, max = 2, A = {9, 8, 7, 2, 6, 3, 1, 5}

Lần 4: i = 3, max = 4, A = {9, 8, 7, 6, 2, 3, 1, 5}

Lần 5: i = 4, max = 7, A = {9, 8, 7, 6, 5, 3, 1, 2}

Lần 6: i = 5, max = 5, A = {9, 8, 7, 6, 5, 3, 1, 2}

Lần 7: i = 6, max = 7, A = {9, 8, 7, 6, 5, 3, 2, 1}

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INSERTION SORT

Insertion Sort, hay còn gọi là chèn trực tiếp.

*** Ý tưởng**: Giả sử dãy A={a_i}, i=0,..,n-1 có i phần tử đầu tiên a_0, a_2, ..,a_{i-1} đã có thứ tự tăng dần. Để A có i+1 phần tử đầu tiên có thứ tự thì cần chèn phần tử a_i vào vị trí k trong đoạn [0,i] sao cho $a_{k-1} \leq a_i < a_k$. Trường hợp dãy có 1 phần tử thì được xem là đã có thứ tự.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INSERTION SORT

*** Giải Thuật**: (theo ngôn ngữ tự nhiên)

Đầu vào: mảng A có n phần tử chưa có thứ tự

Đầu ra: mảng A có n phần tử đã có thứ tự tăng dần.

GIẢI THUẬT SẮP XẾP

- PHƯƠNG PHÁP INSERTION SORT
  - B1: i ← 1
  - B2: x ← a[i],
  - B3: k ← i-1,
  - B4: nếu k >= 0 và A[k] > x thì k ← k-1, quay lại B4.
  - B5: k ← k+1, j ← i,
  - B6: nếu j > k thì A[j] = A[j-1], j ← j-1, quay lại B6.
  - B7: A[k] ← x, i ← i + 1
  - B8: Nếu i < n, qua bước 2.
  - B9: Kết thúc

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INSERTION SORT

*** Giải Thuật**: (theo mã giả)

Đầu vào: mảng A có n phần tử chưa có thứ tự

Đầu ra: mảng A có n phần tử đã có thứ tự tăng dần.

GIẢI THUẬT SẮP XẾP

- PHƯƠNG PHÁP INSERTION SORT

```
for i ← 1 to n – 1
    x ← a[i], k ← i-1
    while k >= 0 AND A[k] > x
        k ← k-1
    k ← k+1, j ← i
    while j > k
        A[j] ← A[j-1], j ← j-1
    A[k] ←  x
```

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INSERTION SORT

*** Cài đặt**:

```
void InsertionSort(int a*, int n) {
    int k, i, j, x;
    for (i = 1; i < n; i++) {
        x = a[i]; k = i - 1;
        while ( k >= 0 && a[k] > x) k--; // xác định vị trí k
        k=k+1;
        for (j = i; j > k; j--) a[j] = a[j - 1]; // dời chỗ.
            a[k] = x;
    }
}
```

GIẢI THUẬT SẮP XẾP

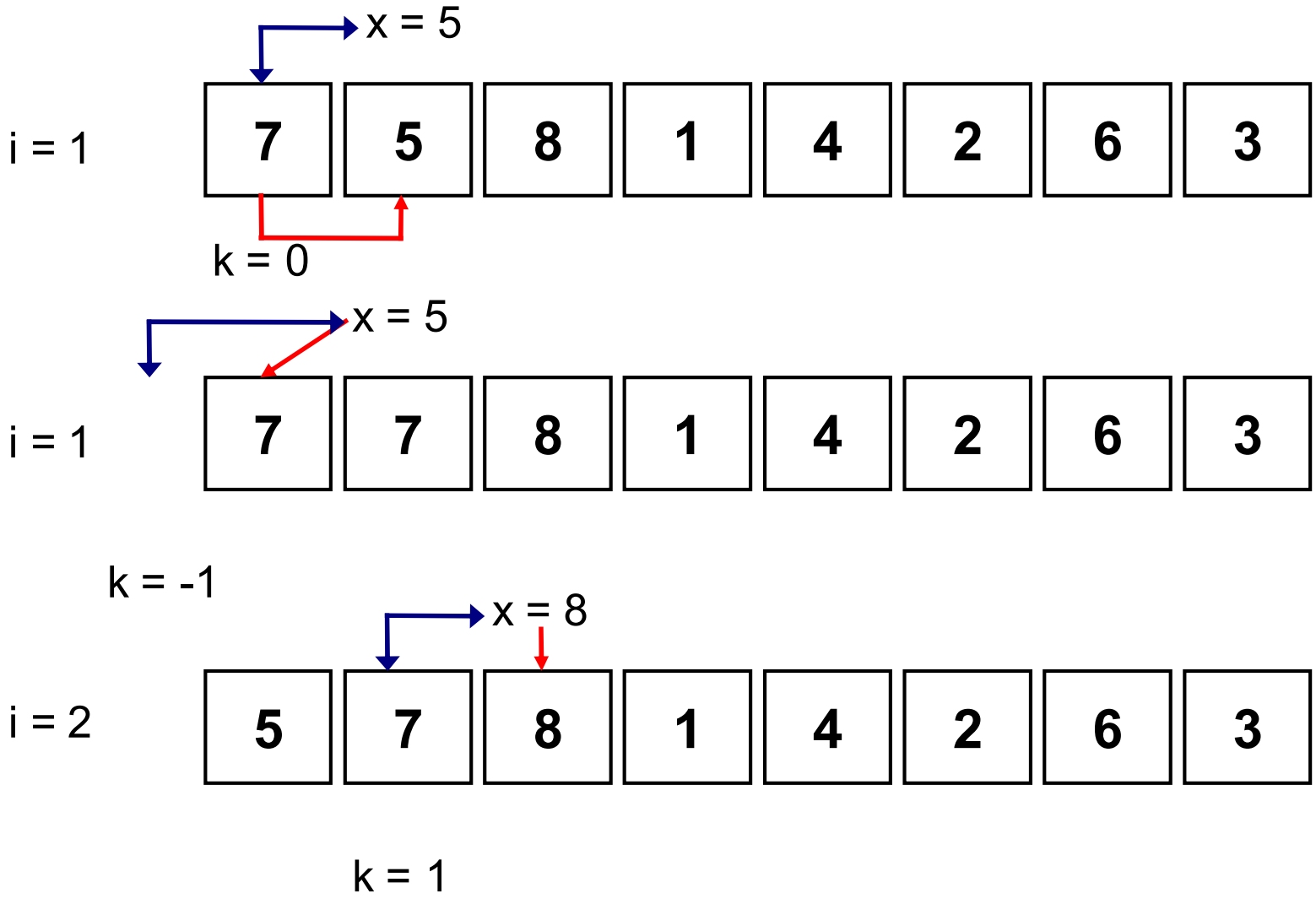
❖PHƯƠNG PHÁP INSERTION SORT

*** Cài đặt**: kết hợp tìm vị trí và dời chỗ

```
void InsertionSort(int a*, int n) {
    int k, i, x;
    for (i = 1; i < n; i++) {
        x = a[i]; k = i - 1;
        while ((k >= 0) && (a[k] > x))
         { a[k + 1] = a[k]; k--; } // tìm vị trí kết hợp với dời chỗ
        a[k+1] = x;
    }
}
```

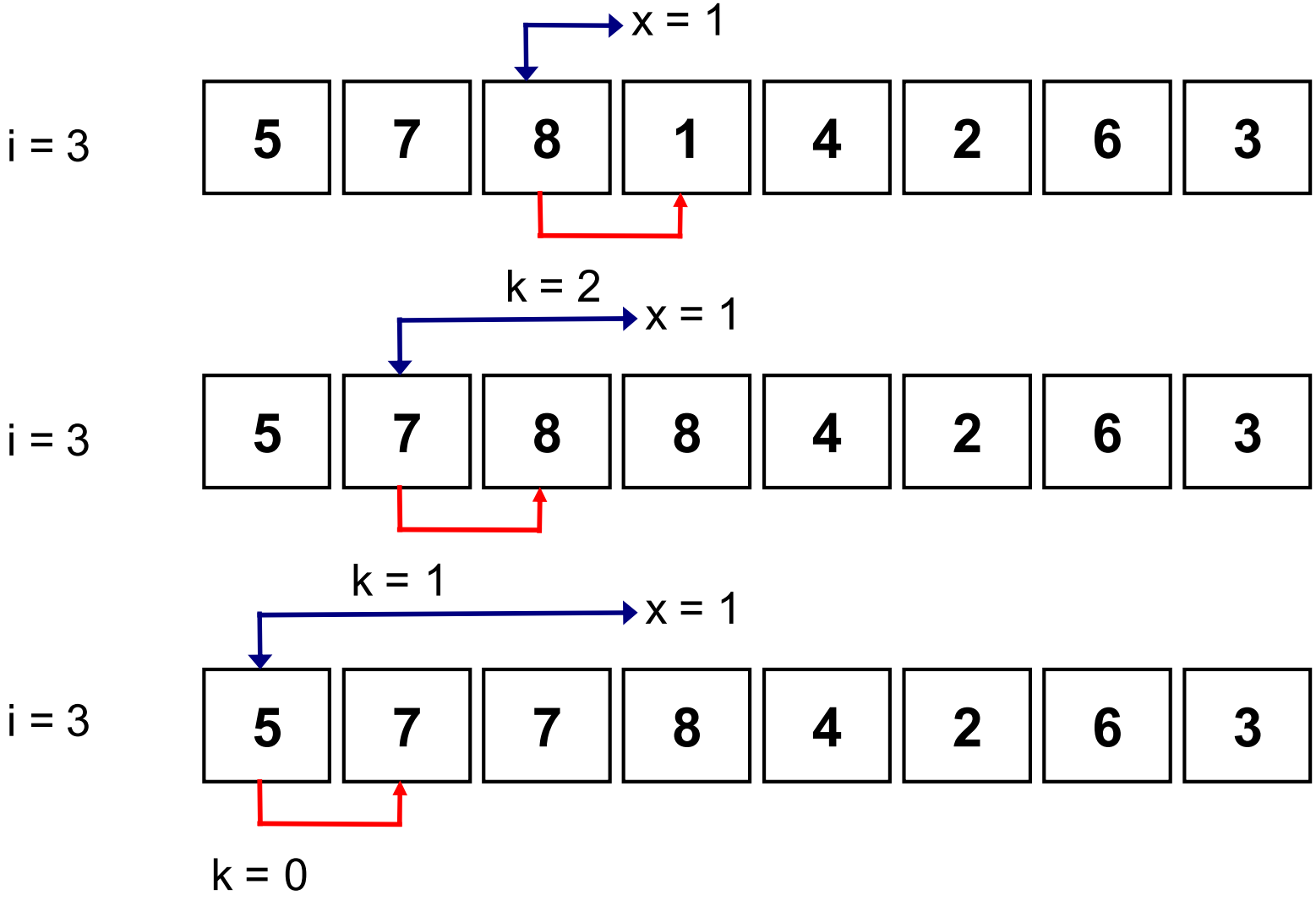
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INSERTION SORT



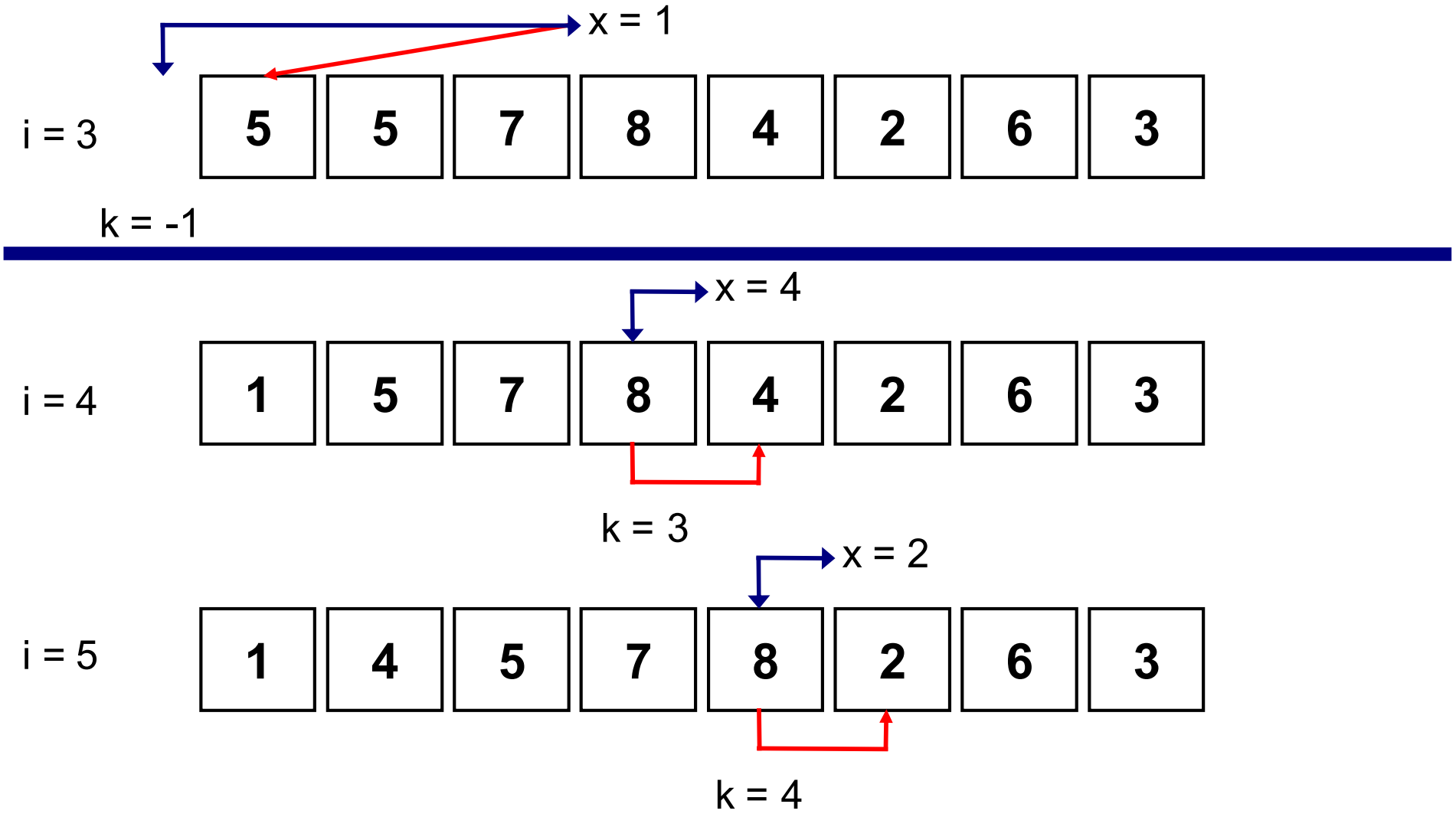
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INSERTION SORT



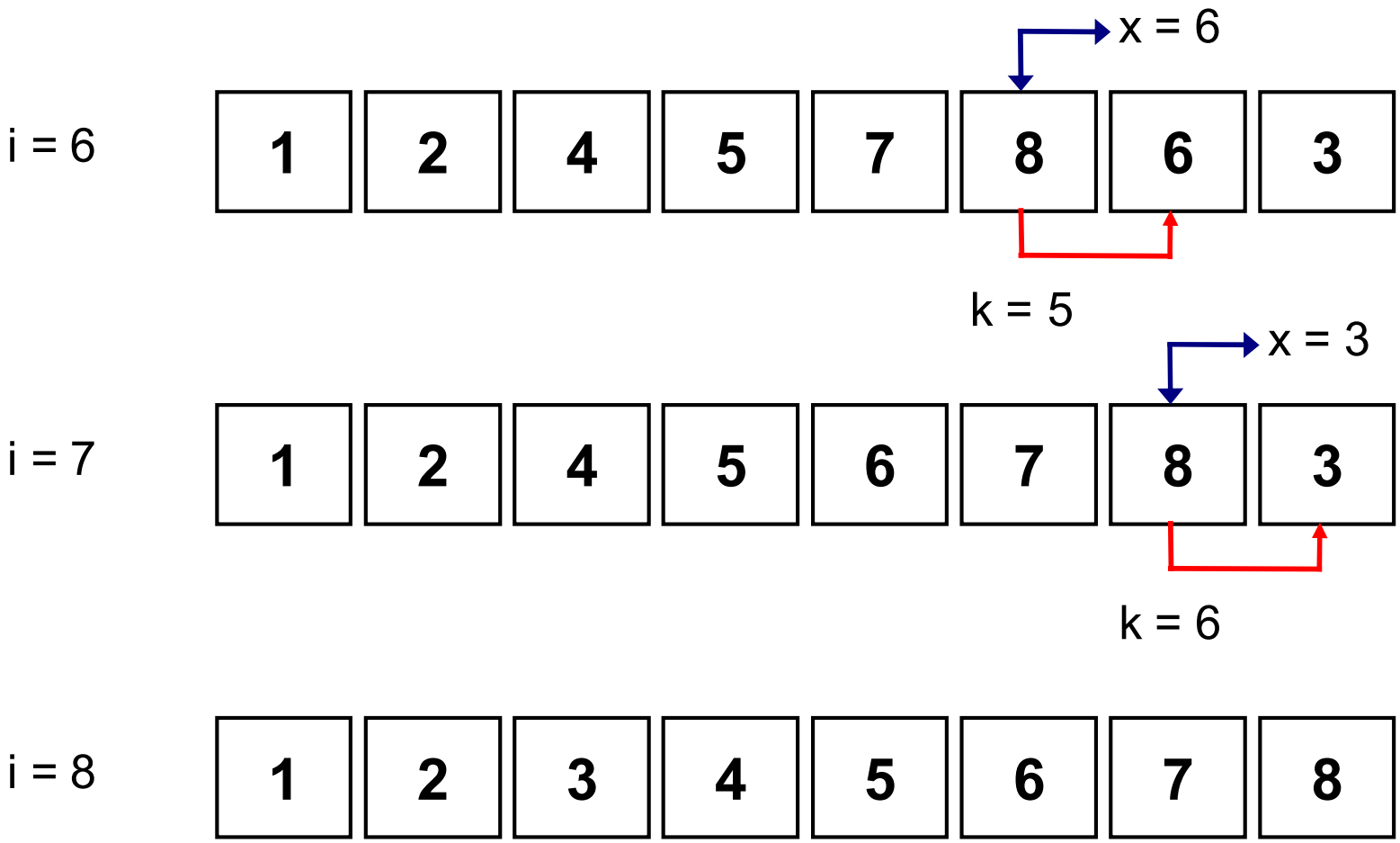
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INSERTION SORT



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INSERTION SORT



GIẢI THUẬT SẮP XẾP

❖ PHƯƠNG PHÁP INSERTION SORT

*** Đánh giá theo trường hợp xấu nhất**:

Xét số phép so sánh giá trị khóa: Bỏ qua các phép tính để thực hiện vòng lặp và các phép gán, ta có:

- Số phép tính để tìm vị trí k trong [0, i]: i
- Số lần lặp để chèn a_i: n - 1
- $T(n) = \Sigma(i), i = 1,..,n-1$

  $= n(n-1)/2$

Độ phức tạp tính toán theo số phép so sánh là $O(n^2)$.

GIẢI THUẬT SẮP XẾP

❖ PHƯƠNG PHÁP INSERTION SORT

*** Đánh giá theo trường hợp xấu nhất**:

Xét số phép gán giá trị khóa: Bỏ qua các phép tính để thực hiện vòng lặp, ta có:

- Số phép tính để dời chỗ trong [0, i-1]: i
- Số phép tính để đưa a_i vào vị trí: 2
- Số lần lặp để đưa ai vào vị trí : n - 1
- $T(n) = \Sigma(i + 2), i = 1,..,n-1$

  $= n(n-1)/2 + 2(n-1) = (n-1)(n+4)/2$

Độ phức tạp tính toán theo phép gán là $O(n^2)$.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INSERTION SORT

Câu hỏi:

Cho biết trường hợp xấu nhất và tốt nhất nếu dùng thuật toán Chèn trực tiếp (Insertion sort) để sắp xếp tang dần là trường hợp dãy A có đặc điểm như thế nào? Cho biết ước lượng số phép gán và phép so sánh trong hai trường hợp này?

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INSERTION SORT

Bài tập:

Trình bày ý tưởng và giải thuật để sắp xếp một dãy số nguyên theo thứ tự giảm dần theo phương pháp Chèn trực tiếp.

Áp dụng: trình bày từng bước quá trình sắp xếp theo giải thuật đã nêu cho dãy số sau:

A = {5, 9 ,7 ,2, 6, 3, 1, 8}

GIẢI THUẬT SẮP XẾP

Ý tưởng:

Ban đầu, phần có thứ tự của dãy A chỉ có 1 phần tử đầu tiên.

Sau đó, với mỗi phần tử i, (i = 1..n-1) trong dãy A:

- Tìm vị trí k $\in$ [0, i] sao cho A[k] >= A[i] > A[k+1]
- Lấy A[i] ra khỏi A và chèn vào vị trí k của A.

GIẢI THUẬT SẮP XẾP

Thuật toán:

Đầu vào: Mảng A gồm n phần tử chưa có thứ tự giảm dần

Đầu vào: Mảng A gồm n phần tử có thứ tự giảm dần.

```
for i ← 1 to n – 1
    x ← a[i], k ← i-1
    while k >= 0 AND A[k] < x
        k ← k-1
    k ← k+1, j ← i
    while j > k
        A[j] ← A[j-1], j ← j-1
     A[k] ← x
```

GIẢI THUẬT SẮP XẾP

Quá trình sắp xếp dãy A = {5, 9 ,7 ,2, 6, 3, 1, 8} theo thứ tự giảm dần:

Trước tiên: A = {5, 9 ,7 ,2, 6, 3, 1, 8}

Lần 1: i = 1, k = 0, A = {9, 5, 7, 2, 6, 3, 1, 8}

Lần 2: i = 2, k = 1, A = {9, 7, 5, 2, 6, 3, 1, 8}

Lần 3: i = 3, k = 3, A = {9, 8, 7, 2, 6, 3, 1, 5}

Lần 4: i = 4, k = 3, A = {9, 8, 7, 6, 2, 3, 1, 5}

Lần 5: i = 5, k = 4, A = {9, 8, 7, 6, 3, 2, 1, 5}

Lần 6: i = 6, k = 6, A = {9, 8, 7, 6, 3, 2, 1, 5}

Lần 7: i = 7, k = 4, A = {9, 8, 7, 6, 5, 3, 2, 1}

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INSERTION SORT

*** BInsertion Sort**:

Để cải tiến việc xác định vị trí chèn phần tử, việc tìm kiếm vị trí thích hợp trong dãy $a_0,..,a_{i-1}$ có thể thực hiện bằng giải thuật tìm kiếm nhị phân.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INSERTION SORT

*** Cài đặt**:

```
void BInsertionSort(int a*, int n) {
    int k, i, j, l, r, x;
    for (i = 1; i < n; i++) {
        x = a[i]; r = i - 1; l = 0;
        while (l <= r) { k = (l+r)/2; if (a[k] > x) r=k-1; else l=k+1; }
        if (a[k] <= x ) k=k+1;
        for (j = i; j > k; j--) a[j] = a[j - 1]; // dời chỗ.
        a[k] = x;
    }
}
```

ĐẠI HỌC QUỐC GIA TPHCM
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

CHƯƠNG II

TÌM KIẾM VÀ SẮP XẾP

Nguyễn Trọng Chỉnh
chinhnt@uit.edu.vn

GIẢI THUẬT SẮP XẾP

- PHƯƠNG PHÁP BUBBLE SORT
- PHƯƠNG PHÁP INTERCHANGE SORT (đọc thêm)
- PHƯƠNG PHÁP HEAP SORT
- PHƯƠNG PHÁP SHELL SORT (đọc thêm)

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP BUBBLE SORT

Bubble Sort, hay còn gọi là sắp xếp nổi bọt

*** Ý tưởng**: Giả sử dãy A={a_i} có n phần tử. Bắt đầu từ cuối dãy, các phần tử nhỏ sẽ được đẩy dần về phía trái dãy đến vị trí đúng của nó bằng cách thực hiện các phép hoán vị 2 phần tử liên tiếp có tạo nghịch thế. Khi một phần tử nhỏ nhất được đưa về đầu dãy thì chỉ xét việc hoán vị cho dãy mới gồm các phần tử còn lại.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP BUBBLE SORT

*** Giải Thuật**: (theo ngôn ngữ tự nhiên)

Đầu vào: mảng A gồm n phần tử chưa có thứ tự

Đầu ra: mảng A gồm n phần tử đã có thứ tự.

- Bước 1: i $\leftarrow$ 0
- Bước 2: j $\leftarrow$ n-1
- Bước 3: Nếu j $\leq$ i thực hiện bước 5. Ngược lại, nếu A[j]<A[j - 1] thì hoán đổi A[j] và A[j-1].
- Bước 4: j $\leftarrow$ j - 1, qua bước 3.
- Bước 5: i $\leftarrow$ i +1. Nếu i < n-1 qua bước 2.
- Bước 6: Kết thúc.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP BUBBLE SORT

*** Giải Thuật**: (theo mã giả)

Đầu vào: mảng A gồm n phần tử chưa có thứ tự

Đầu ra: mảng A gồm n phần tử đã có thứ tự.

```
for i ← 0 to n – 2
    for j ← n - 1 down to i + 1
        if A[j] < A[j - 1]
            hoandoi(A[j], A[j - 1])
```

GIẢI THUẬT SẮP XẾP

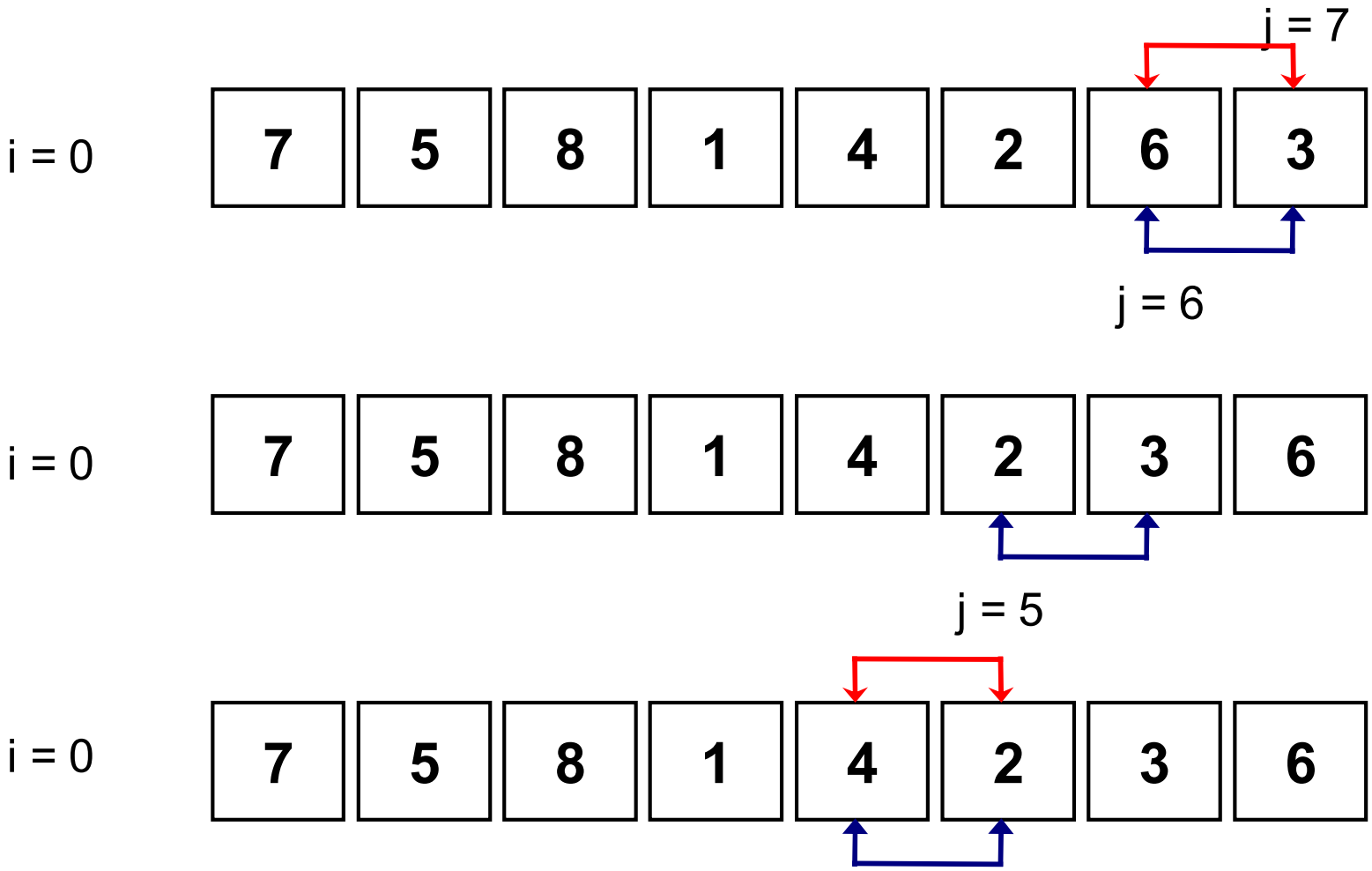
❖PHƯƠNG PHÁP BUBBLE SORT

*** 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])
                hoandoi(a[j], a[j-1]);
    }
}
```

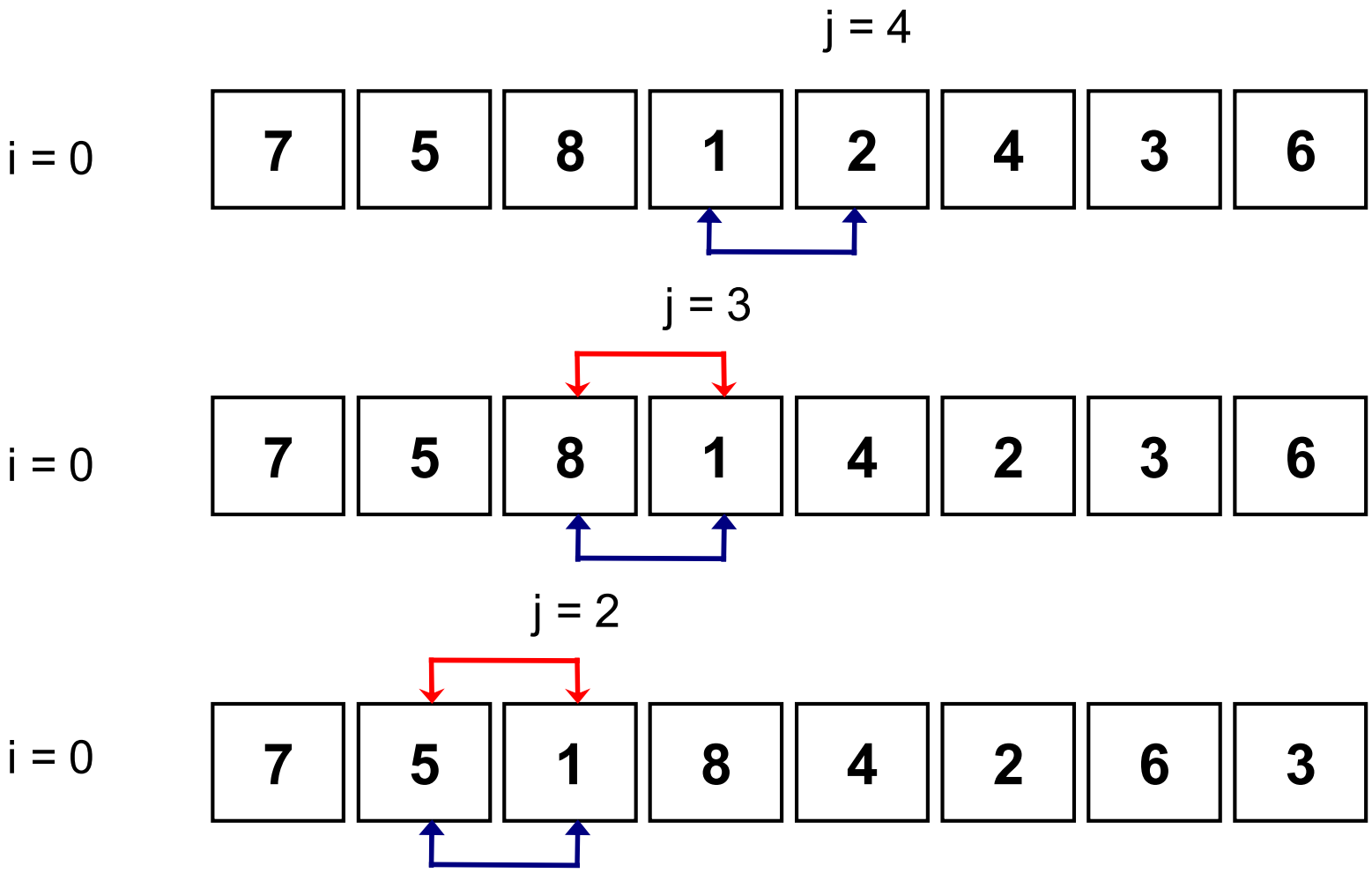
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP BUBBLE SORT



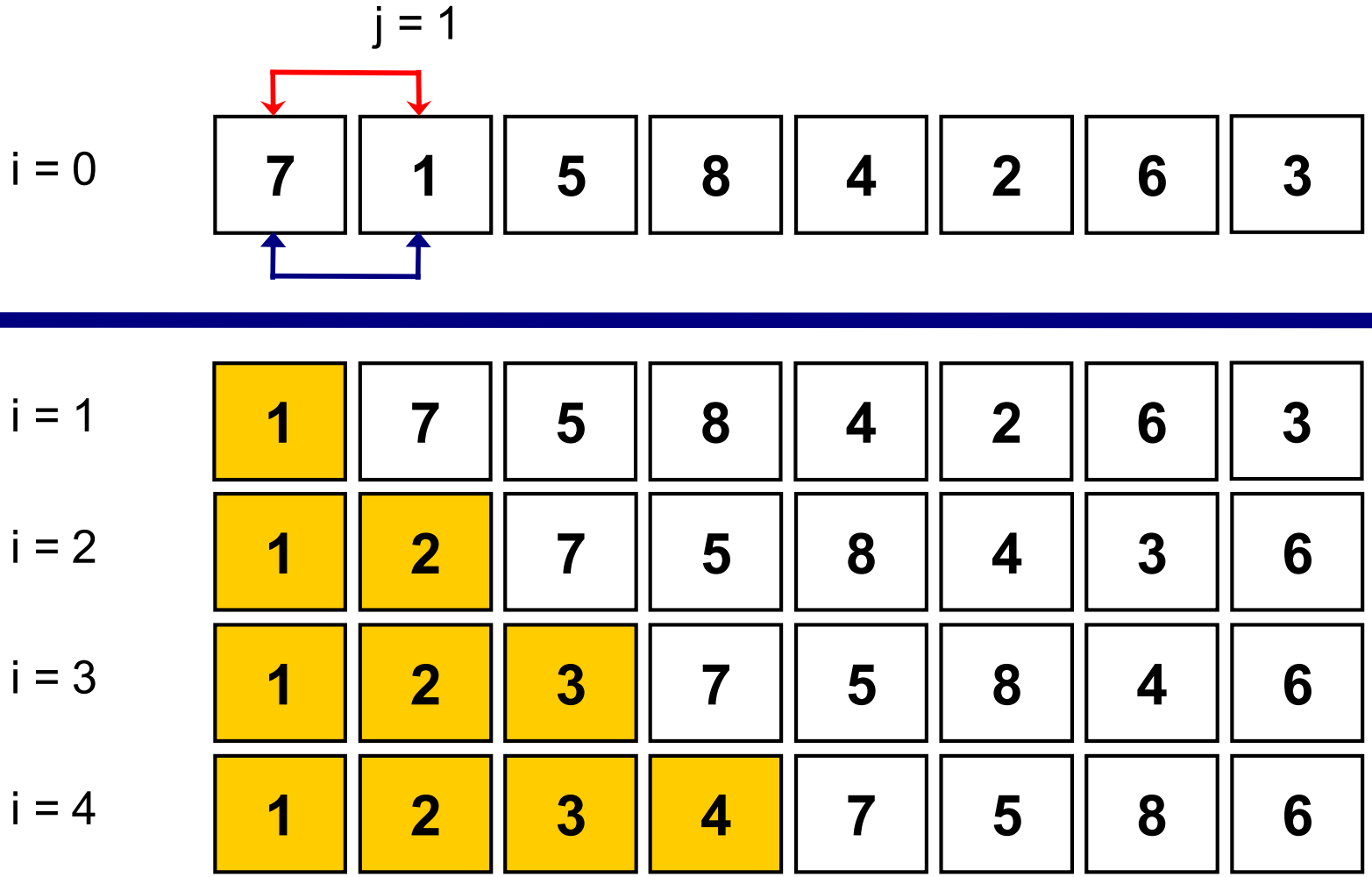
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP BUBBLE SORT



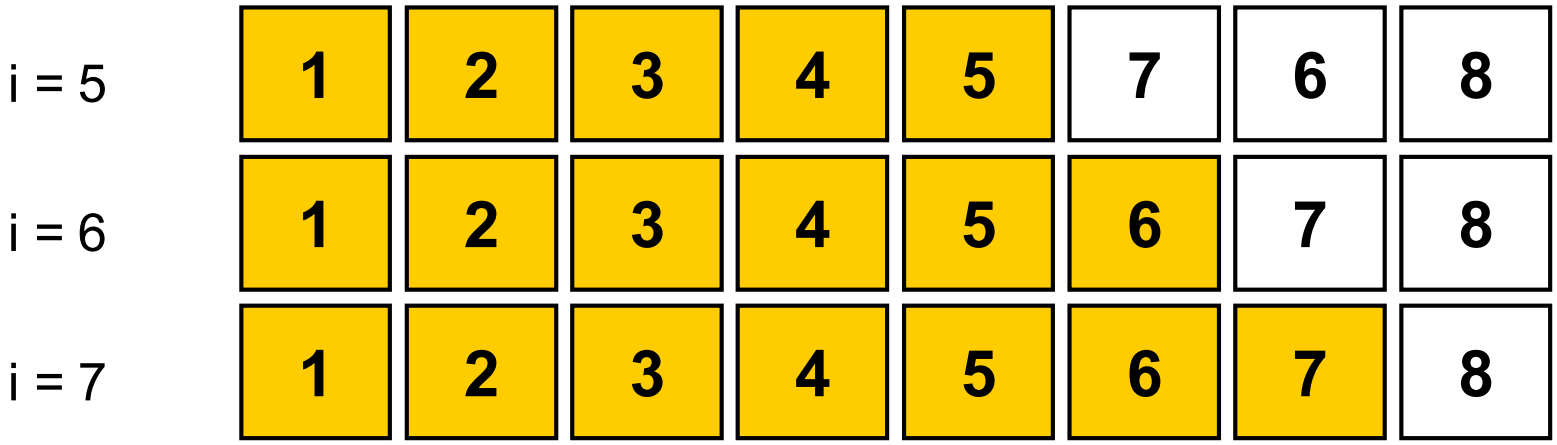
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP BUBBLE SORT



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP BUBBLE SORT



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP BUBBLE SORT

*** Đánh giá theo trường hợp xấu nhất**:

- Xét số phép so sánh giá trị khóa: Bỏ qua các phép tính để thực hiện vòng lặp, ta có:

$T(n) = n(n-1)/2$

Độ phức tạp tính toán theo số phép so sánh là $O(n^2)$.

- Xét số phép gán giá trị khóa: Bỏ qua các phép tính để thực hiện vòng lặp, ta có:

$T(n) = 3*n(n-1)/2$

Độ phức tạp tính toán theo số phép gán là $O(n^2)$

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP BUBBLE SORT

*** Giải thuật Shaker Sort**:

Cải tiến từ bubble sort với lượt đẩy phần tử lớn về cuối mảng trong mỗi lần lặp và ghi nhớ vị trí xuất hiện nghịch thế cuối cùng của mỗi lượt đẩy. Các vị trí này xác định phạm vi các phần tử cần tiến hành sắp xếp trong lần lặp tiếp theo.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP BUBBLE SORT

*** Giải thuật Shaker Sort**:

Thuật toán (theo mã giả):

Đầu vào: mảng A gồm n phần tử chưa có thứ tự

Đầu ra: mảng A gồm n phần tử đã có thứ tự.

GIẢI THUẬT SẮP XẾP

```
b ← 1, e ← n - 1
while (b < e)
    j ← e, tmp ← e
    while (j > b)
    {
        if A[j] < A[j - 1]
        {
            hoandoi(A[j], A[j - 1]), tmp ← j
        }
        j ← j – 1
    }
    b ← tmp
```

GIẢI THUẬT SẮP XẾP

```
    j ← b, tmp ← b
    while (j <= e)
    {
        if A[j] < A[j - 1]
        {
            hoandoi(A[j], A[j - 1]), tmp ← j
        }
        j ← j + 1
    }
     e ← tmp
}
```

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP BUBBLE SORT

BÀI TẬP: Cho dãy A = {3, 6, 1, 2, 4, 5}

1) Trình bày từng bước quá trình sắp xếp dãy A theo thứ tự tăng dần với phương pháp Bubble Sort

2) Trường hợp tốt nhất và xấu nhất của phương pháp Bubble Sort xảy ra khi dãy A có đặc điểm như thế nào? Số phép tính trong trường hợp tốt nhất?

GIẢI THUẬT SẮP XẾP

Sắp xếp dãy A = 3, 6, 1, 2, 4, 5

- i = 0: 1, 3, 6, 2, 4, 5
- i = 1: 1, 2, 3, 6, 4, 5
- i = 2: 1, 2, 3, 4, 6, 5
- i = 3: 1, 2, 3, 4, 5, 6
- i = 4: 1, 2, 3, 4, 5, 6

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INTERCHANGE SORT

Interchange Sort, hay còn gọi là đổi chỗ trực tiếp

*** Ý tưởng**: Giả sử dãy A={a_i} có n phần tử. Nếu A có thứ tự thì A không chứa nghịch thế. Vì vậy, với mỗi vị trí thứ i, phải triệt tiêu tất cả nghịch thế giữa phần tử tại vị trí đó với các vị trí còn lại. Bắt đầu từ đầu dãy, sau khi đã triệt tiêu nghịch thế tại vị trí thứ i thì sẽ không có nghịch thế giữa i và i+1 nên chỉ cần xét đãy từ i+1 đến n-1 cho lần triệt tiêu nghịch thế tại vị trí i+1.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INTERCHANGE SORT

*** Giải thuật**:

Đầu vào: mảng A gồm n phần tử chưa có thứ tự

Đầu ra: mảng A gồm n phần tử đã có thứ tự.

- Bước 1: i ← 0
- Bước 2: j ← i+1
- Bước 3: Nếu j > n-1 thực hiện bước 5. Ngược lại, nếu A[j]<A[i] thì hoán đổi A[j] và A[i].
- Bước 4: j ← j + 1, qua bước 3.
- Bước 5: i ← i +1. Nếu i < n-1 qua bước 2.
- Bước 6: Kết thúc.

GIẢI THUẬT SẮP XẾP

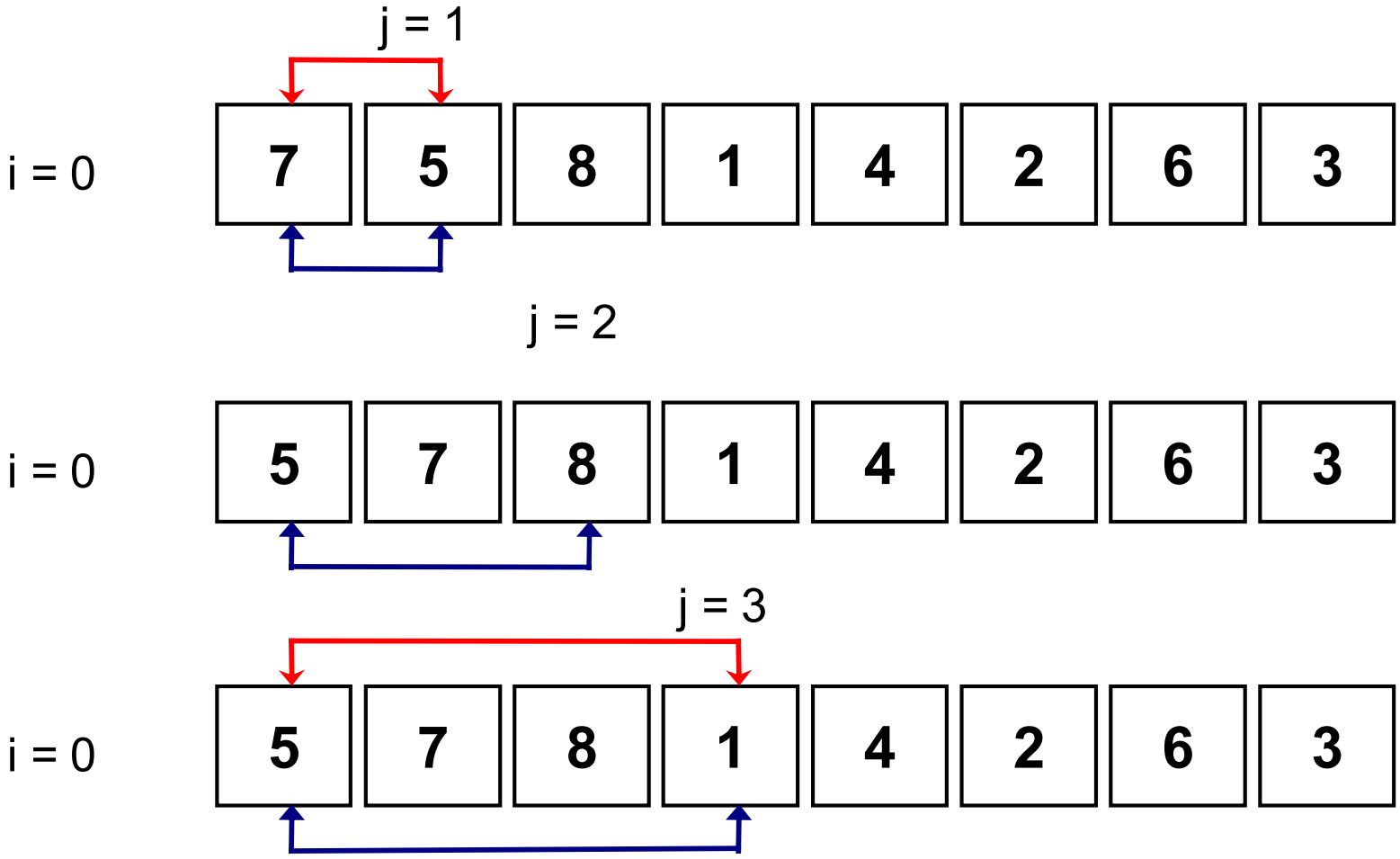
❖PHƯƠNG PHÁP INTERCHANGE SORT

*** 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])
                hoandoi(a[j], a[i]);
    }
}
```

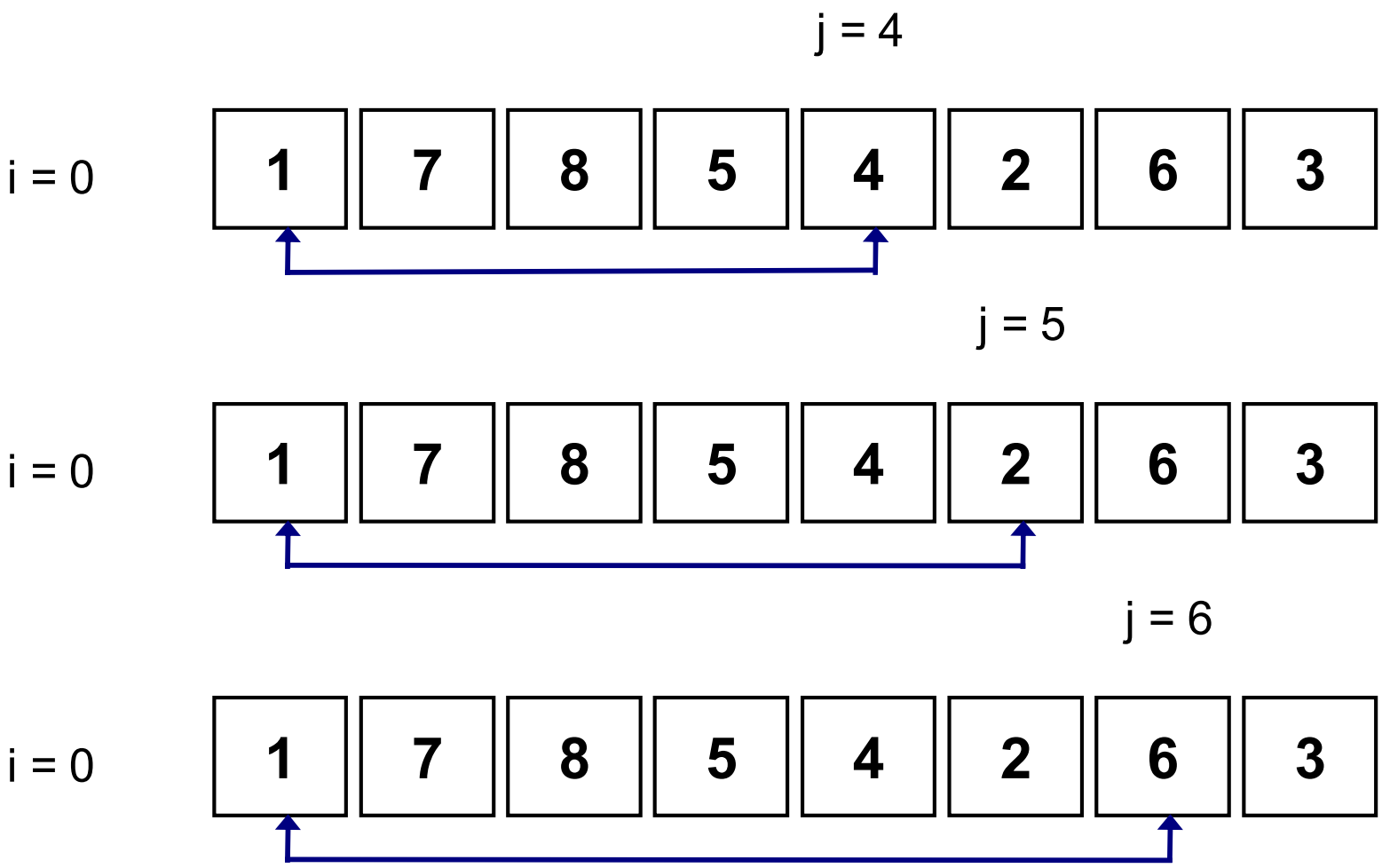
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INTERCHANGE SORT



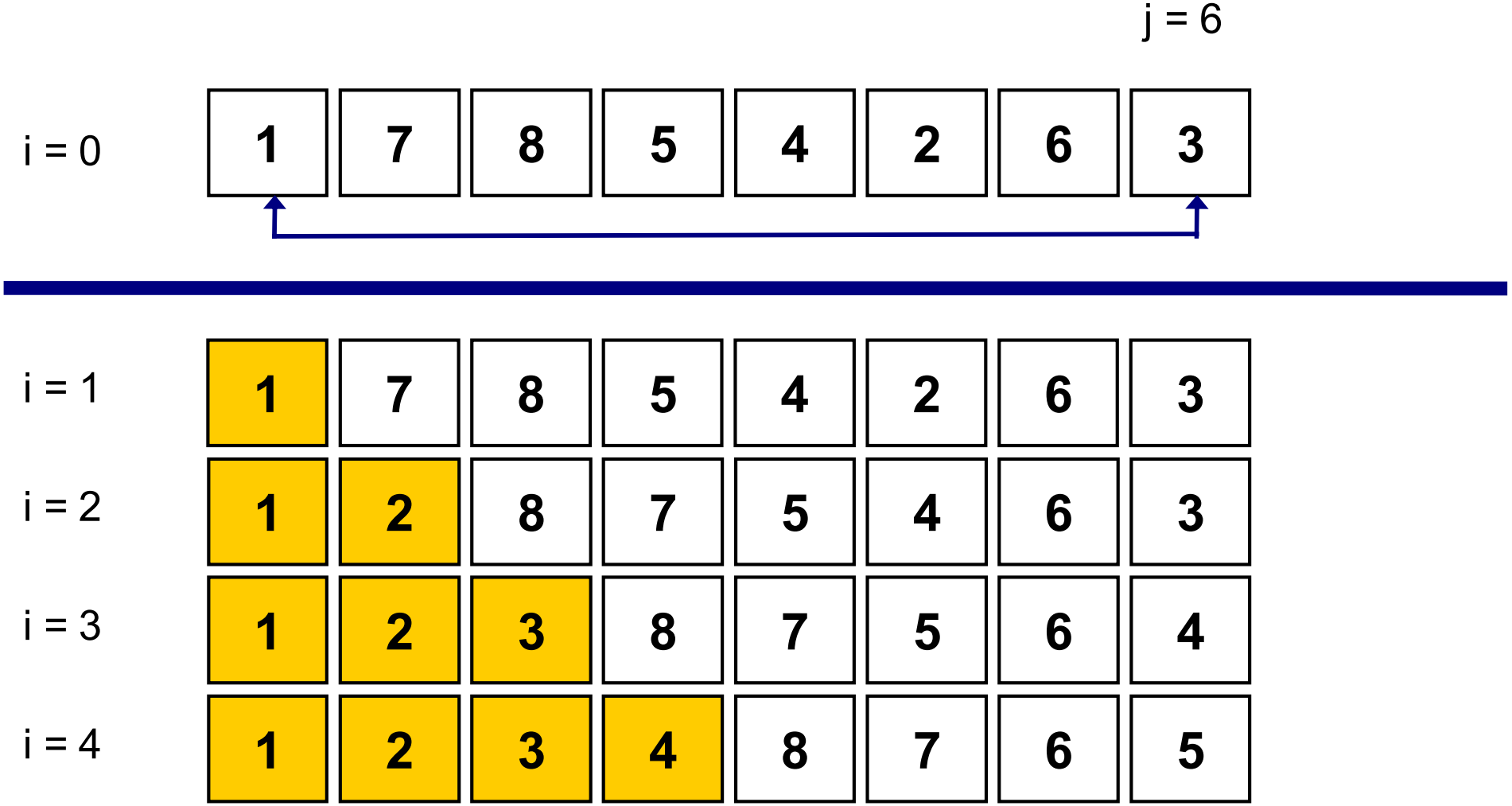
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INTERCHANGE SORT



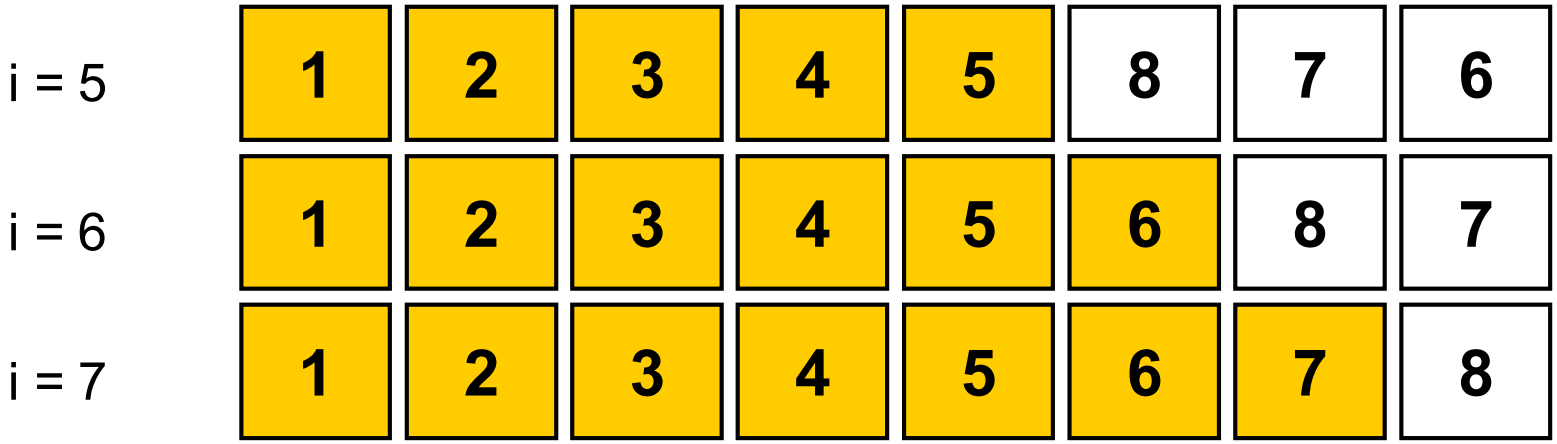
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INTERCHANGE SORT



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INTERCHANGE SORT



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP INTERCHANGE SORT

*** Đánh giá theo trường hợp xấu nhất**:

- Xét số phép so sánh giá trị khóa: Bỏ qua các phép tính để thực hiện vòng lặp, ta có:

$T(n) = n(n-1)/2$

Độ phức tạp tính toán theo số phép so sánh là $O(n^2)$.

- Xét số phép gán giá trị khóa: Bỏ qua các phép tính để thực hiện vòng lặp, ta có:

$T(n) = 3*n(n-1)/2$

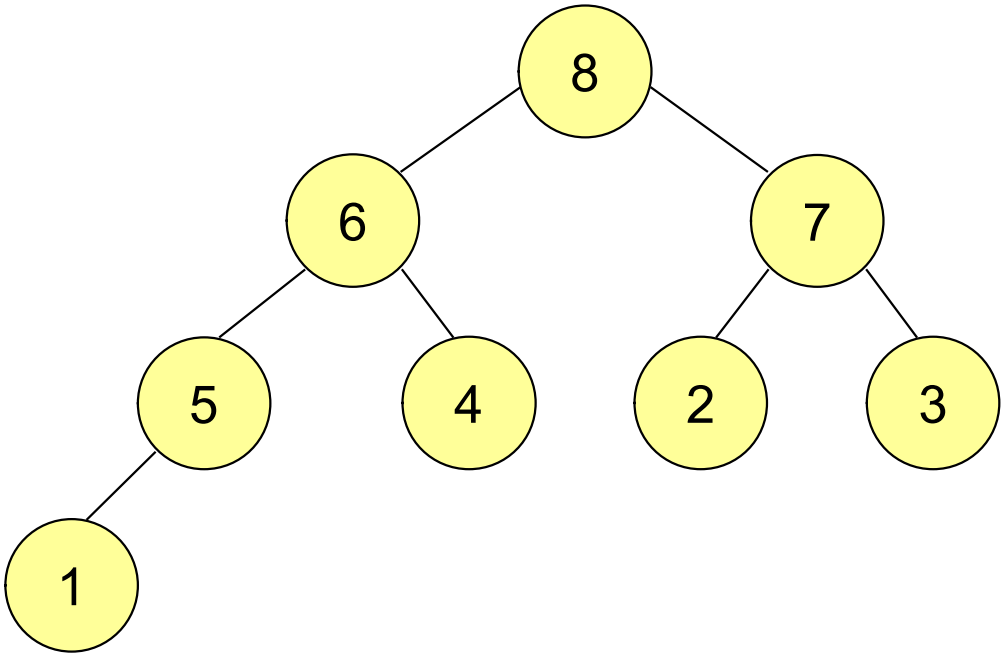
Độ phức tạp tính toán theo số phép gán là $O(n^2)$

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT

***Khái niệm Heap** là một cây nhị phân hoàn chỉnh (complete) trong đó nếu A là nút cha của B và quan hệ R được thỏa trên cây thì A R B.

Ví dụ: Cây nhị phân sau là heap nếu quan hệ R là ≥



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT

*** Tính chất của Heap**

- Phần tử gốc của heap mang giá trị cực trị (Nếu xét quan hệ $\geq$ thì phần tử gốc là cực đại, nếu xét quan hệ $\leq$ thì phần tử gốc là cực tiểu)
- Các nút lá của một heap là một heap.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT

*** Điều chỉnh một cây thành heap**

Giả sử điều chỉnh cây thành heap, quan hệ R là ≥

- B1: Đánh dấu n nút trên cây theo thứ tự từng mức, tại mỗi mức đánh dấu theo thứ tự từ trái sang.

- B2: Xét lần lượt các nút i, i = n-1,..,0. Điều chỉnh cây có gốc k = i thành heap.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT

*** Điều chỉnh một cây thành heap**

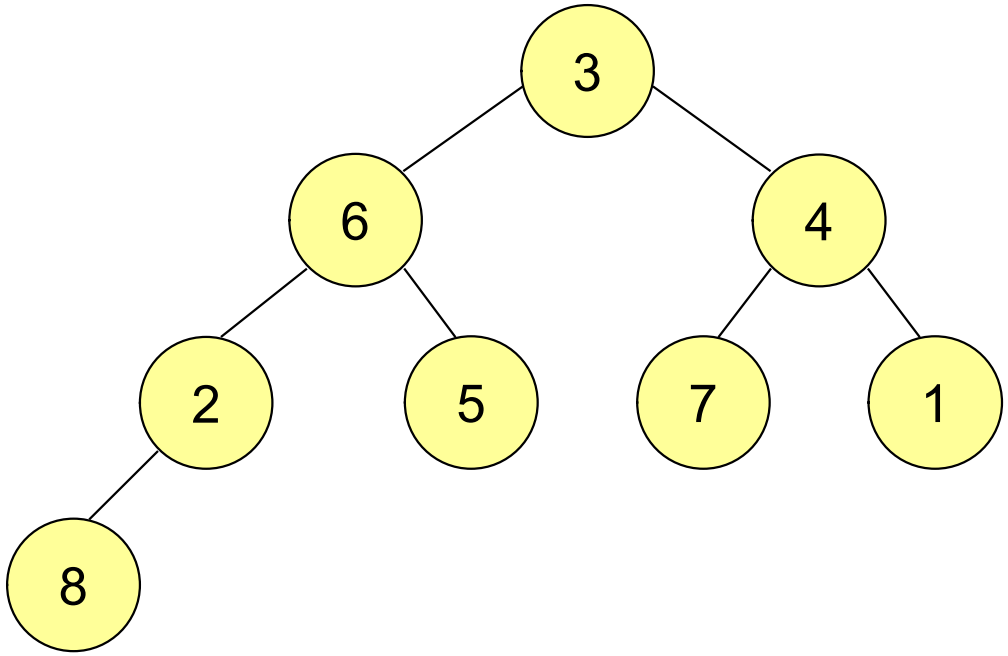
Việc điều chỉnh cây có gốc k thành heap được thực hiện như sau:

- B1: Nếu nút k là nút lá, qua B5.
- B2: Chọn nút con lớn nhất j của nút k.
- B3: Nếu khóa tại nút k lớn hơn khóa tại nút j, qua bước 4; ngược lại, hoán vị khóa tại nút k và nút j
- B4: k ← j. Qua bước B1.
- B5: Kết thúc.

GIẢI THUẬT SẮP XẾP

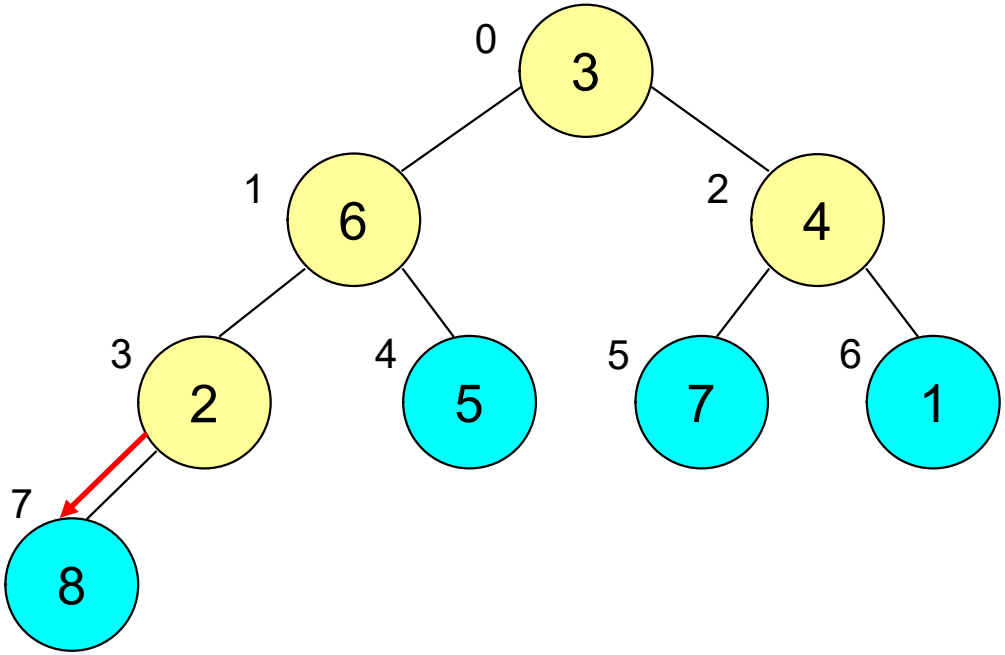
❖PHƯƠNG PHÁP HEAP SORT

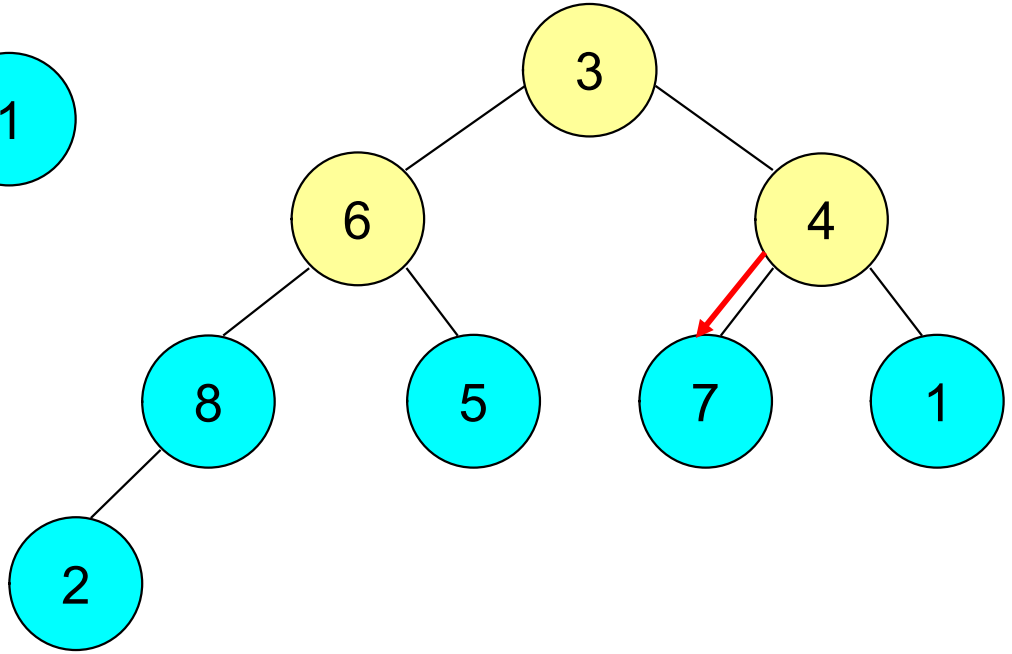
Ví dụ: Điều chỉnh cây nhị phân sau thành một heap với quan hệ R là ≥



GIẢI THUẬT SẮP XẾP

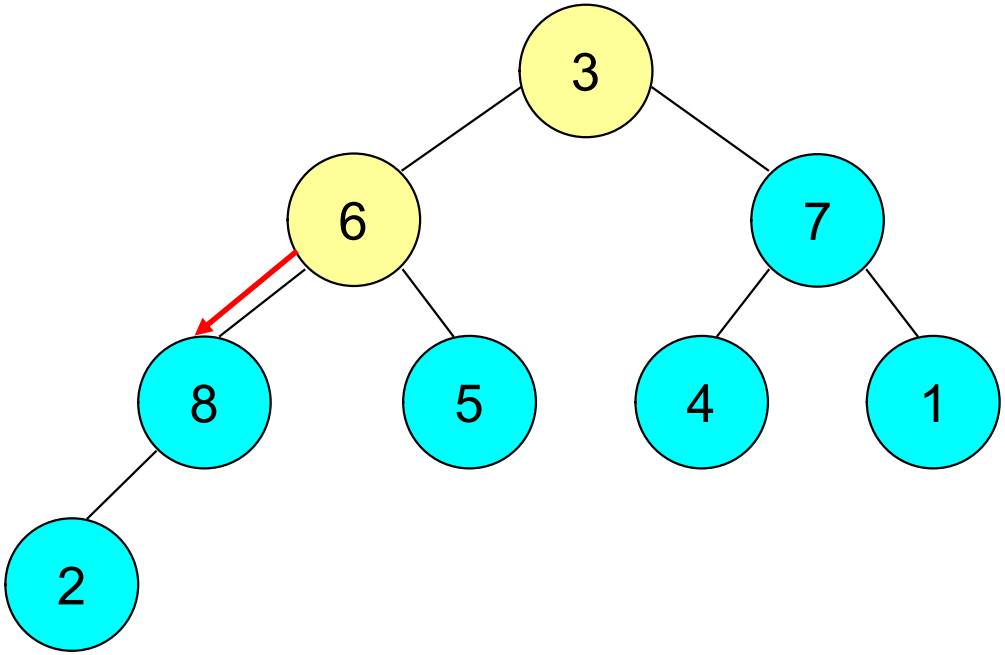
- PHƯƠNG PHÁP HEAP SORT

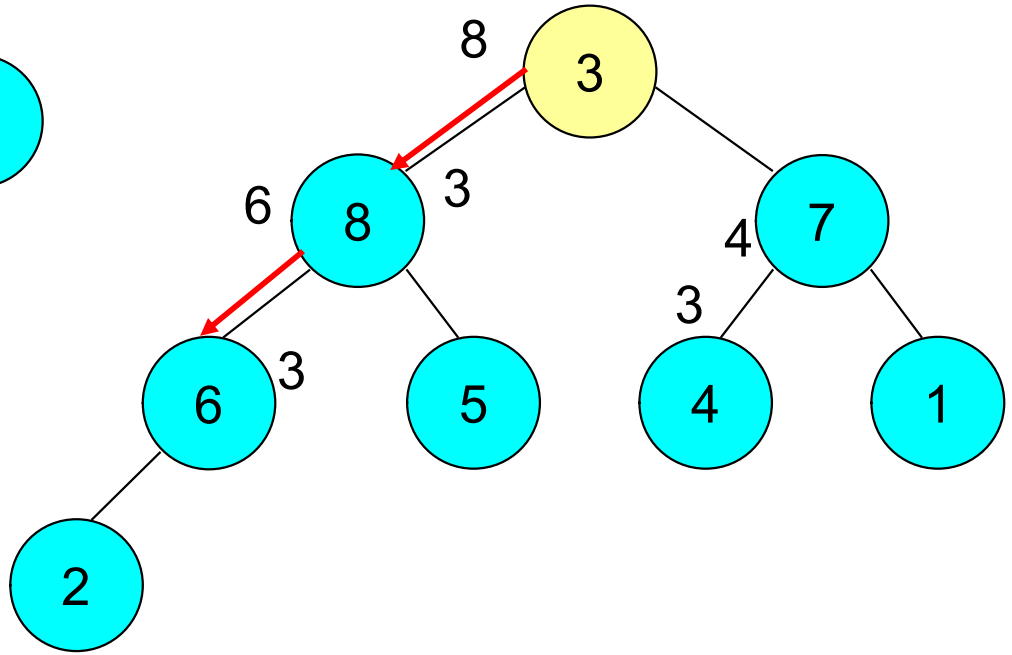




GIẢI THUẬT SẮP XẾP

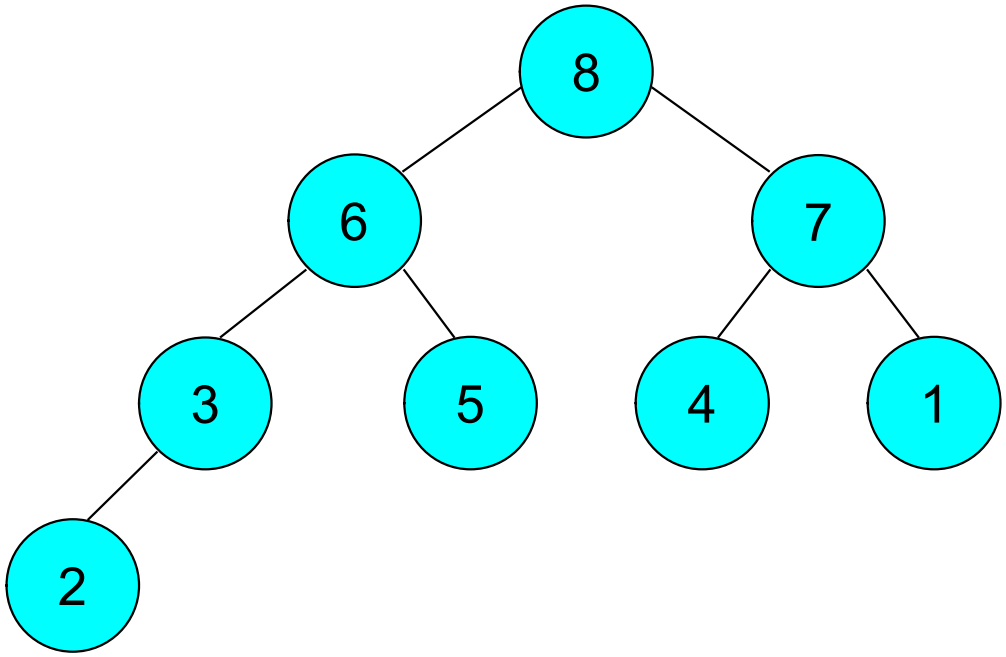
- PHƯƠNG PHÁP HEAP SORT





GIẢI THUẬT SẮP XẾP

- PHƯƠNG PHÁP HEAP SORT



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT

*** Xóa nút gốc của heap**

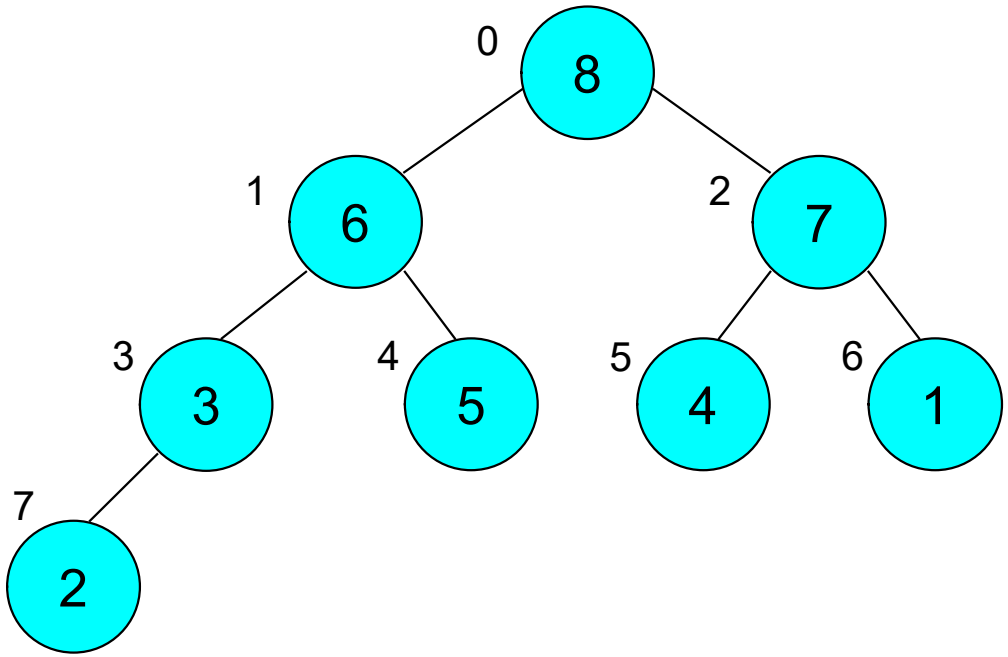
Để xóa nút gốc của heap mà vẫn đảm bảo cây còn lại là 1 heap, thực hiện như sau:

- B1: hoán vị nút tại vị trí 0 và vị trí n-1, xóa nút n-1.
- B2: Điều chỉnh cây có gốc k = 0 thành một heap.

GIẢI THUẬT SẮP XẾP

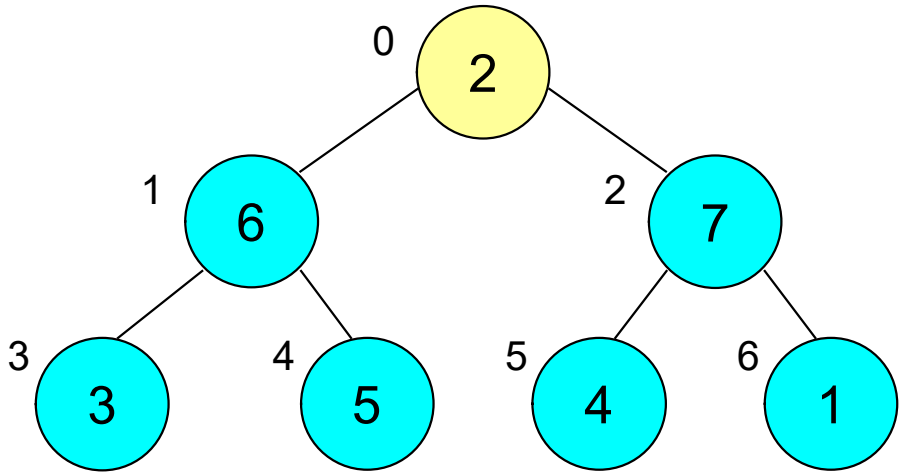
❖PHƯƠNG PHÁP HEAP SORT

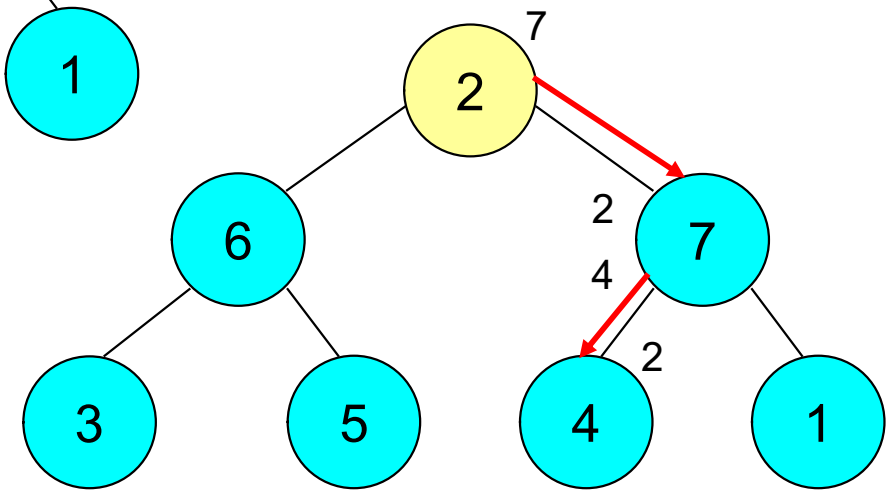
Ví dụ: xóa nút gốc của heap nhị phân sau:



GIẢI THUẬT SẮP XẾP

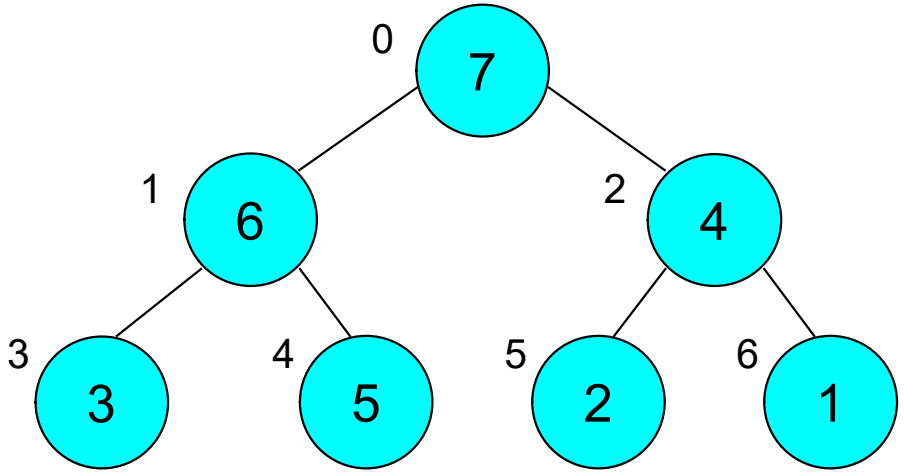
- PHƯƠNG PHÁP HEAP SORT





GIẢI THUẬT SẮP XẾP

- PHƯƠNG PHÁP HEAP SORT

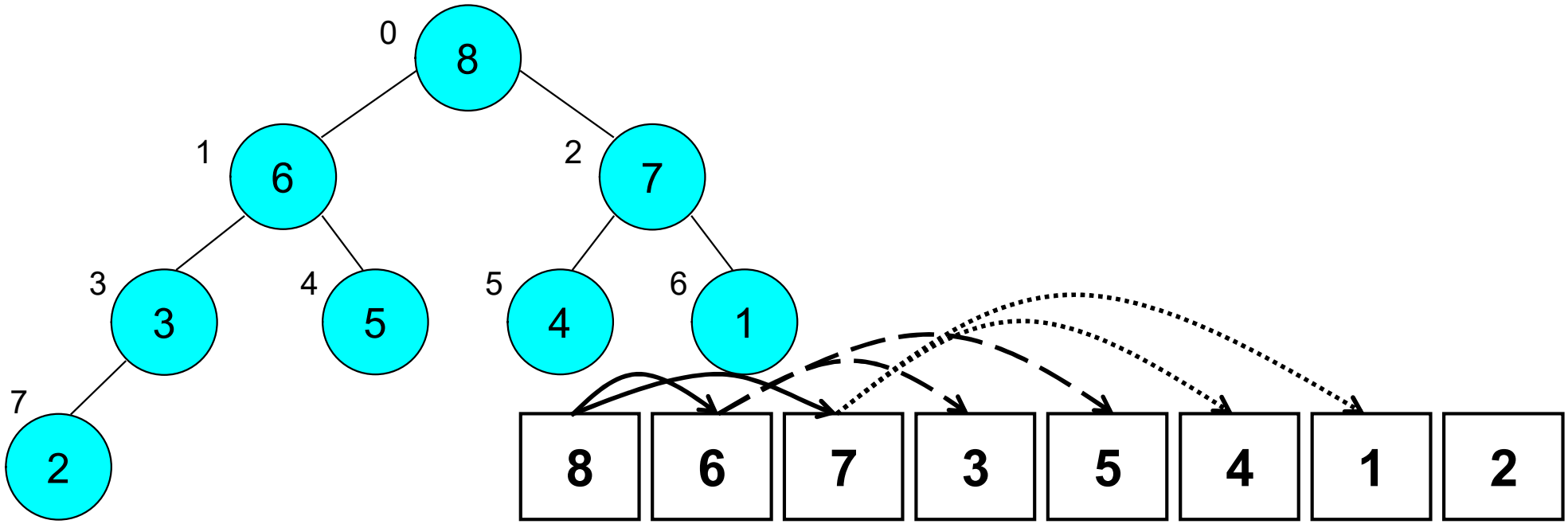


GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT

*** Lưu trữ cấu trúc heap nhị phân**

Heap nhị phân có thể được lưu trữ theo cấu trúc cây nhị phân. Tuy nhiên, để tăng hiệu quả, dùng mảng một chiều để lưu trữ heap như sau:



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT

*** Lưu trữ cấu trúc heap nhị phân**

Tính chất của mảng heap nhị phân:

- Phần tử thứ i là cha của hai phần tử i*2 + 1và (i+1)*2. Vì vậy, $a_i \geq a_{i*2+1}$ và $a_i \geq a_{(i+1)*2}$
- Các nút i ≥ n/2 là đỉnh của một heap một phần tử.

Nhận xét: Heap là một cấu trúc dữ liệu dạng cây. Tuy nhiên nó có thể được lưu trữ bằng một dãy tuần tự. Vì thế, cấu trúc dữ liệu có thể khác với cấu trúc lưu trữ.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT

*** Cài đặt giải thuật tạo heap nhị phân trên mảng.**

```
void Heapify(int *a, int k, int n) {
    int x, j = 2*k+1;
    while (j < n) {
        if (j + 1 < n)
            if (a[j] < a[j + 1]) j = j + 1;
        if (a[k] >= a[j]) return;
        x = a[k]; a[k] = a[j]; a[j] = x; k = j; j = 2*k + 1;
    }
}
```

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT

*** Cài đặt giải thuật tạo heap nhị phân trên mảng.**

```
void BuildHeap(int *a, int n) {
    int i;
    i = (n - 1) / 2;
    while (i >= 0) {
        Heapify(a, i, n);
        i--;
    }
}
```

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT

*** Ý tưởng:**

Heap sort cải tiến dựa trên Selection Sort trong đó việc chọn phần tử thích hợp (lớn nhất hoặc nhỏ nhất) được thực hiện trên heap để giảm chi phí so sánh. Để sắp xếp mảng theo thứ tự tăng dần, sử dụng heap max (đỉnh heap là max) để chọn phần tử lớn nhất trong i phần tử còn lại của mảng. Phần tử này sẽ có vị trí i-1 trong mảng đã sắp xếp.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT

*** Giải thuật:**

Đầu vào là mảng A có n phần tử chưa có thứ tự

Đầu ra là mảng A có n phần tử có thứ tự tăng dần

- B1: Tạo heap nhị phân trên mảng A có n phần tử
- B2: Hoán đổi A[0] và A[n-1] để đưa phần tử max về cuối dãy.
- B3: n ← n - 1. Nếu n > 0 thì điều chỉnh A với n phần tử thành heap, qua B2.
- B4: Kết thúc.

GIẢI THUẬT SẮP XẾP

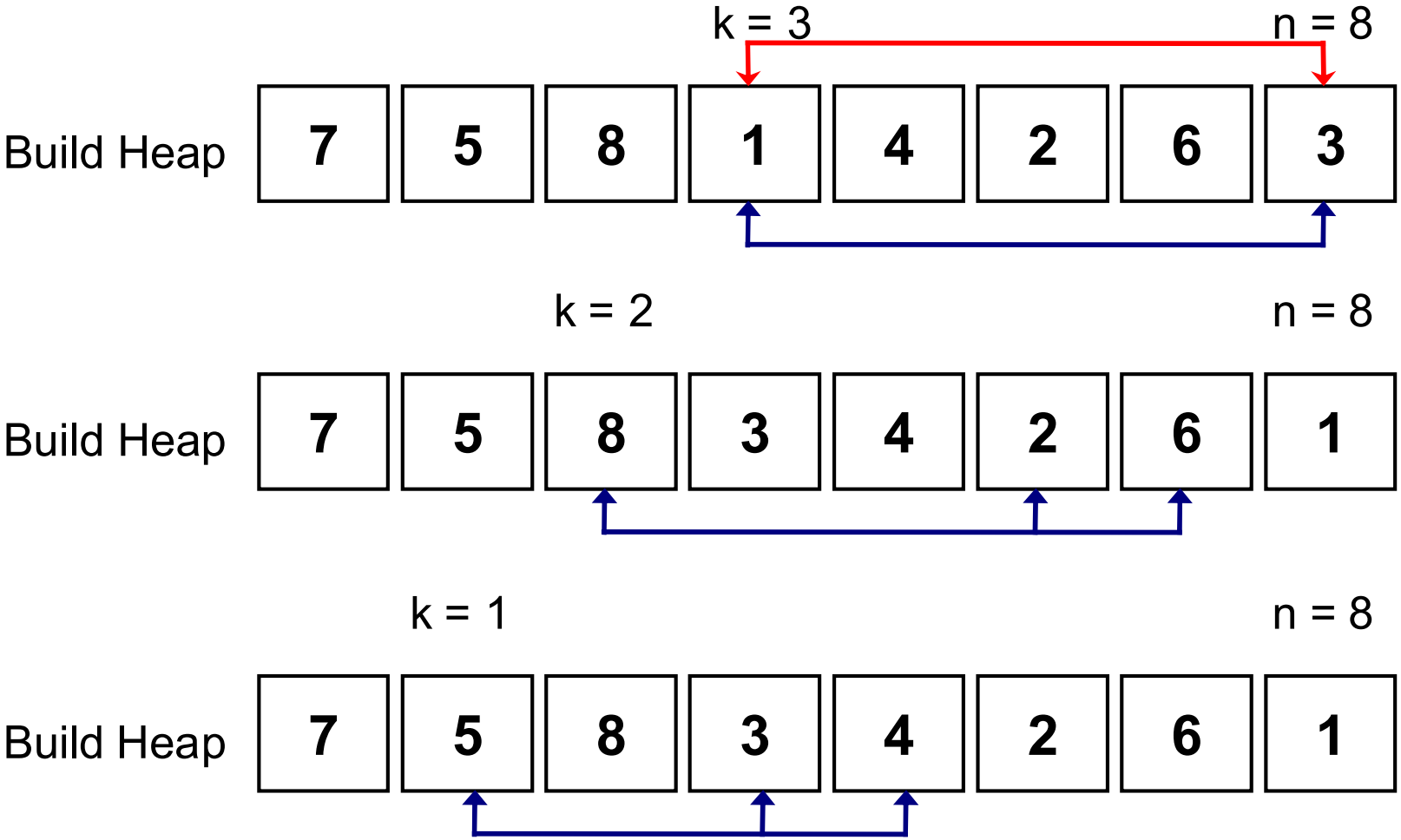
- PHƯƠNG PHÁP HEAP SORT

*** Cài đặt:**

```
void HeapSort(int *a, int n) {
    BuildHeap(a, n);
    while (n > 0) {
        n = n - 1;
        hoandoi(a[0], a[n]);
        Heapify(a, 0, n);
    }
}
```

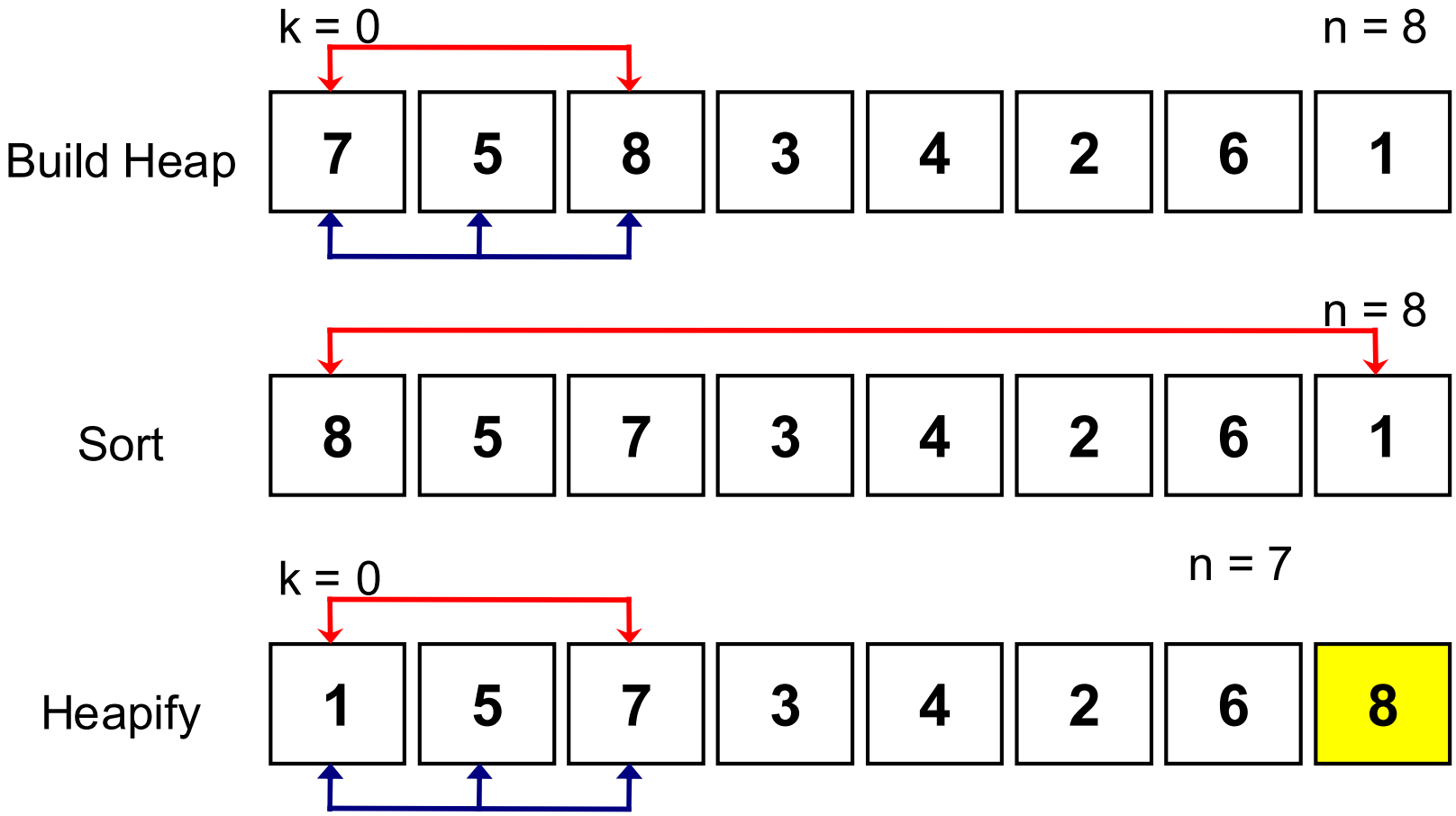
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT



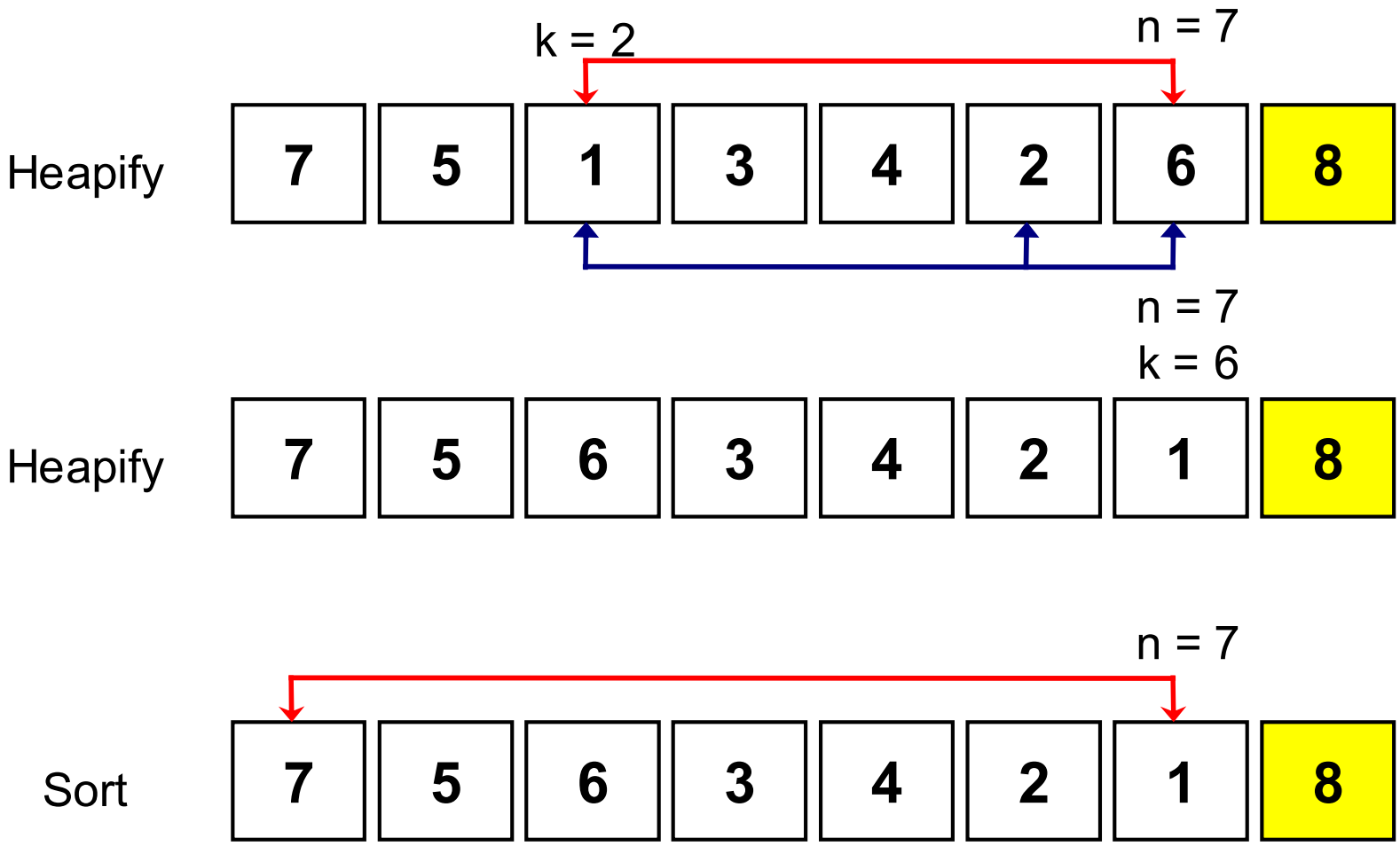
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT



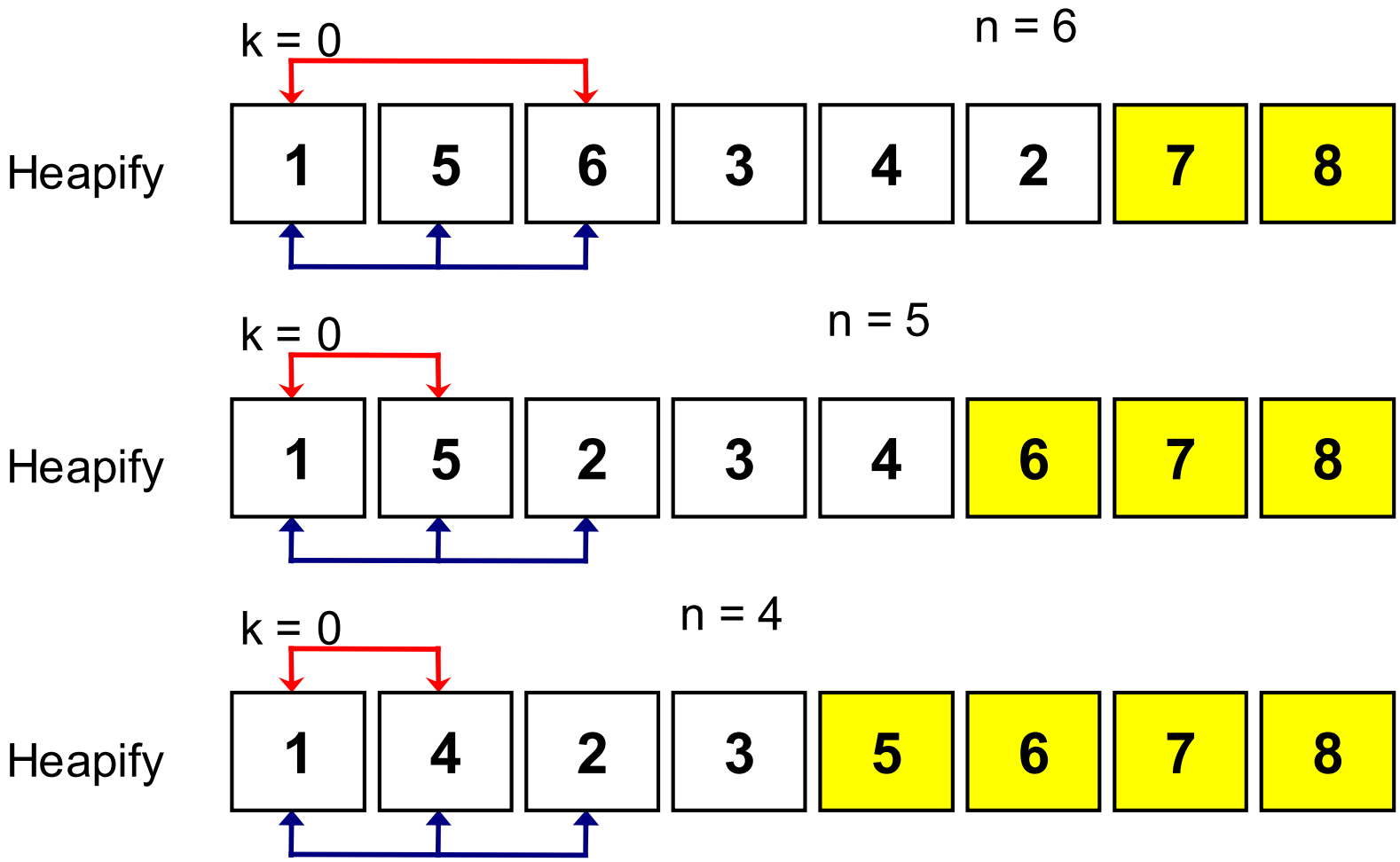
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT



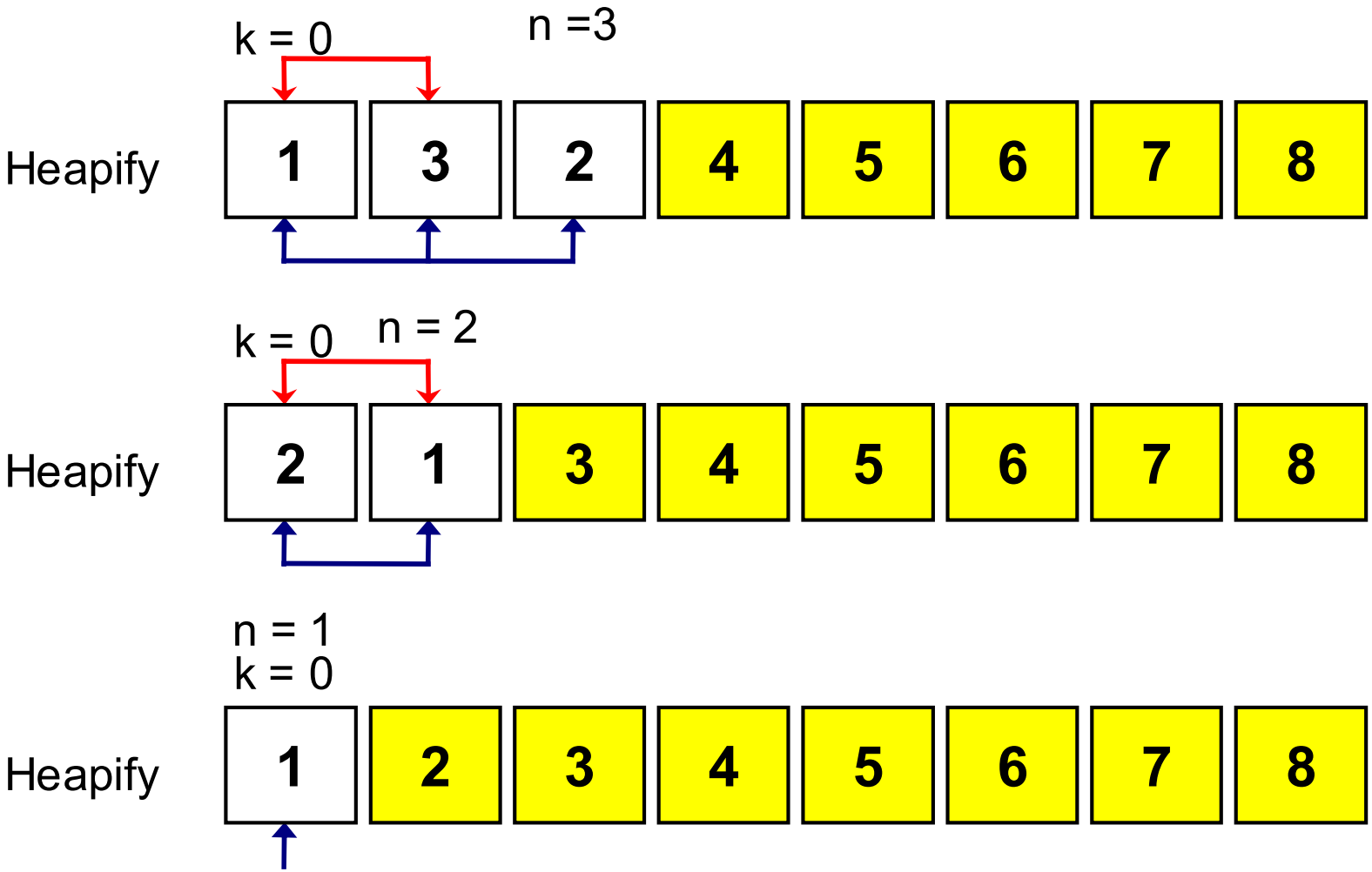
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT



GIẢI THUẬT SẮP XẾP

- PHƯƠNG PHÁP HEAP SORT

*** Đánh giá giải thuật:**

Trong mọi trường hợp giải thuật Heap Sort có độ phức tạp trung bình O(nlogn) cho cả phép so sánh và phép gán.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP HEAP SORT

BÀI TẬP: Cho dãy A = {3, 4, 1, 2, 6, 5}

1) Trình bày từng bước quá trình sắp xếp dãy A theo thứ tự tăng dần với phương pháp Heap Sort

2) Trường hợp tốt nhất và xấu nhất của phương pháp Heap Sort xảy ra khi dãy A có đặc điểm như thế nào? Số phép tính trong trường hợp tốt nhất?

GIẢI THUẬT SẮP XẾP

* Xây dựng Heap Max từ dãy A:

3, 4, 1, 2, 6, 5

- Heapify i = 2,

  con max j = 5, A[2] < A[5] → đổi chỗ

  3, 4, 5, 2, 6, 1

- Heapify i = 1,

  con max j = 4, A[1] < A[4] → đổi chỗ

  3, 6, 5, 2, 4, 1

- Heapify i = 0,

  con max j = 1, A[0] < A[1] → đổi chỗ

GIẢI THUẬT SẮP XẾP

6, 3, 5, 2, 4, 1

Heapify i = 1, con max j = 4, A[1]<A[4] → đổi chỗ

6, 4, 5, 2, 3, 1

* Sắp xếp n = 6: 1, 4, 5, 2, 3, 6
* Heapify i = 0, con max j = 2, A[0] < A[2] → đổi chỗ

5, 4, 1, 2, 3, 6

* Sắp xếp n = 5: 3, 4, 1, 2, 5, 6
* Heapify i = 0, con max j = 1, A[0] < A[1] → đổi chỗ

4, 3, 1, 2, 5, 6

* Sắp xếp n = 4: 2, 3, 1, 4, 5, 6

GIẢI THUẬT SẮP XẾP

* Sắp xếp n = 4: 2, 3, 1, 4, 5, 6
* Heapify i = 0, con max j = 1, A[0] < A[1] → đổi chỗ

  3, 2, 1, 4, 5, 6
* Sắp xếp n = 3: 1, 2, 3, 4, 5, 6
* Heapify i = 0, con max j = 1, A[0] < A[1] → đổi chỗ

  2, 1, 3, 4, 5, 6
* Sắp xếp n = 2: 1, 2, 3, 4, 5, 6
* Sắp xếp n = 1: 1, 2, 3, 4, 5, 6

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SHELL SORT

*** Ý tưởng**: Cải tiến từ giải thuật Insertion Sort. Trong đó, để sắp xếp dãy A = $\{a_i\}$, i = 0,.., n-1, sẽ chia dãy thành những dãy con đan xen nhau, những dãy này cách nhau h phần tử và sắp xếp những dãy này theo phương pháp Insertion Sort. Sau đó giảm độ dài h để tạo thành những dãy đan xen mới và sắp xếp. Cuối cùng, dãy A được sẽ được sắp xếp khi sắp xếp với dãy cuối cùng có h = 1. Mục tiêu của việc chia dãy là để giảm trường hợp dời chỗ trong Insertion Sort.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SHELL SORT

Ví dụ:

Sắp xếp dãy A = {3, 4, 1, 2, 6, 5} theo thứ tự tăng dần.

<u>Sắp xếp theo Selection Sort, số lần dời chỗ 5:</u>

Lần 1, i = 1, k = 1, A = {3, 4, 1, 2, 6, 5}, dời chỗ 0 lần

Lần 2, i = 2, k = 0, A = {1, 3, 4, 2, 6, 5}, dời chỗ 2 lần

Lần 3, i = 3, k = 1, A = {1, 2, 3, 4, 6, 5}, dời chỗ 2 lần

Lần 4, i = 4, k = 4, A = {1, 2, 3, 4, 6, 5}, dời chỗ 0 lần

Lần 5, i = 5, k = 4, A = {1, 2, 3, 4, 5, 6}, dời chỗ 1 lần

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SHELL SORT

Sắp xếp theo Shell Sort, h_0=2, h_1=1, số lần dời chỗ 3:

h_0=2 A = {**3**, 4, **1**, 2, **6**, 5} (2 dãy đỏ và xanh)

Lần 1, i = 2, k = 0, A = {**1**, 4, **3**, 2, **6**, 5}, dời chỗ 1 lần
Lần 2, i = 3, k = 1, A = {**1**, 2, **3**, 4, **6**, 5}, dời chỗ 1 lần
Lần 3, i = 4, k = 4, A = {**1**, 2, **3**, 4, **6**, 5}, dời chỗ 0 lần
Lần 4, i = 5, k = 4, A = {**1**, 2, **3**, 4, **6**, 5}, dời chỗ 0 lần

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SHELL SORT

h_0=1 A = {1, 2, 3, 4, 6, 5} (1 dãy xanh)

Lần 1, i = 1, k = 1, A = {1, 2, 3, 4, 6, 5}, dời chỗ 0 lần
Lần 2, i = 2, k = 2, A = {1, 2, 3, 4, 6, 5}, dời chỗ 0 lần
Lần 3, i = 3, k = 3, A = {1, 2, 3, 4, 6, 5}, dời chỗ 0 lần
Lần 4, i = 4, k = 4, A = {1, 2, 3, 4, 6, 5}, dời chỗ 0 lần
Lần 5, i = 5, k = 4, A = {1, 2, 3, 4, 6, 5}, dời chỗ 1 lần

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP SHELL SORT

***Thuật toán** (Xem Giáo trình)

***Cài đặt** (Xem Giáo trình)

***Đánh giá giải thuật** (Xem Giáo trình)

ĐẠI HỌC QUỐC GIA TPHCM
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

CHƯƠNG II

TÌM KIẾM VÀ SẮP XẾP

Nguyễn Trọng Chỉnh
chinhnt@uit.edu.vn

GIẢI THUẬT SẮP XẾP

- PHƯƠNG PHÁP QUICK SORT
- PHƯƠNG PHÁP MERGE SORT
- PHƯƠNG PHÁP RADIX SORT (Sinh viên tự đọc)

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT

*** Ý tưởng:** (Quick sort - sắp xếp dựa trên phân hoạch)

Giả sử dãy A có n phần tử có thứ tự tăng dần.

- Phần tử chốt (pivot) a_k, k = 0,..,n-1 có giá trị khóa không nhỏ hơn các phần tử của dãy đã có thứ tự $a_0,..,a_{k-1}$ và có giá trị khóa không lớn hơn các phần tử của dãy đã có thứ tự $a_{k+1},..a_{n-1}$.
- Để sắp xếp, chọn một phần tử a_k bất kỳ trong A, chia dãy A thành hai dãy $a_0,..,a_{k-1}$ có giá trị không lớn hơn a_k và dãy $a_{k+1},..,a_{n-1}$ có giá trị không nhỏ hơn, sắp xếp hai dãy này theo cách như trên.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT

*** Giải thuật:** Sử dụng giải thuật đệ quy như sau:

Đầu vào: dãy $a_l,a_{l+1},..,a_r$ chưa có thứ tự.

Đầu ra: dãy $a_l, a_{l+1},..,a_r$ có thứ tự tăng dần.

- B1: Nếu l < r thì k ← (l + r)/2, x ← a_k, i ← l, j ← r; ngược lại qua B6.
- B2: Nếu a_i > x, qua B3; ngược lại i ← i+1, qua B2.
- B3: Nếu a_j < x, qua B4; ngược lại j ← j - 1, qua B3.
- B4: Nếu i < j thì hoán đổi a_i và a_j.
- B5: Thực hiện Quick Sort cho dãy $a_l, a_{l+1},..,a_{r=j}$ và $a_{l=i}, a_{i+1},..,a_r$
- B6: Kết thúc.

GIẢI THUẬT SẮP XẾP

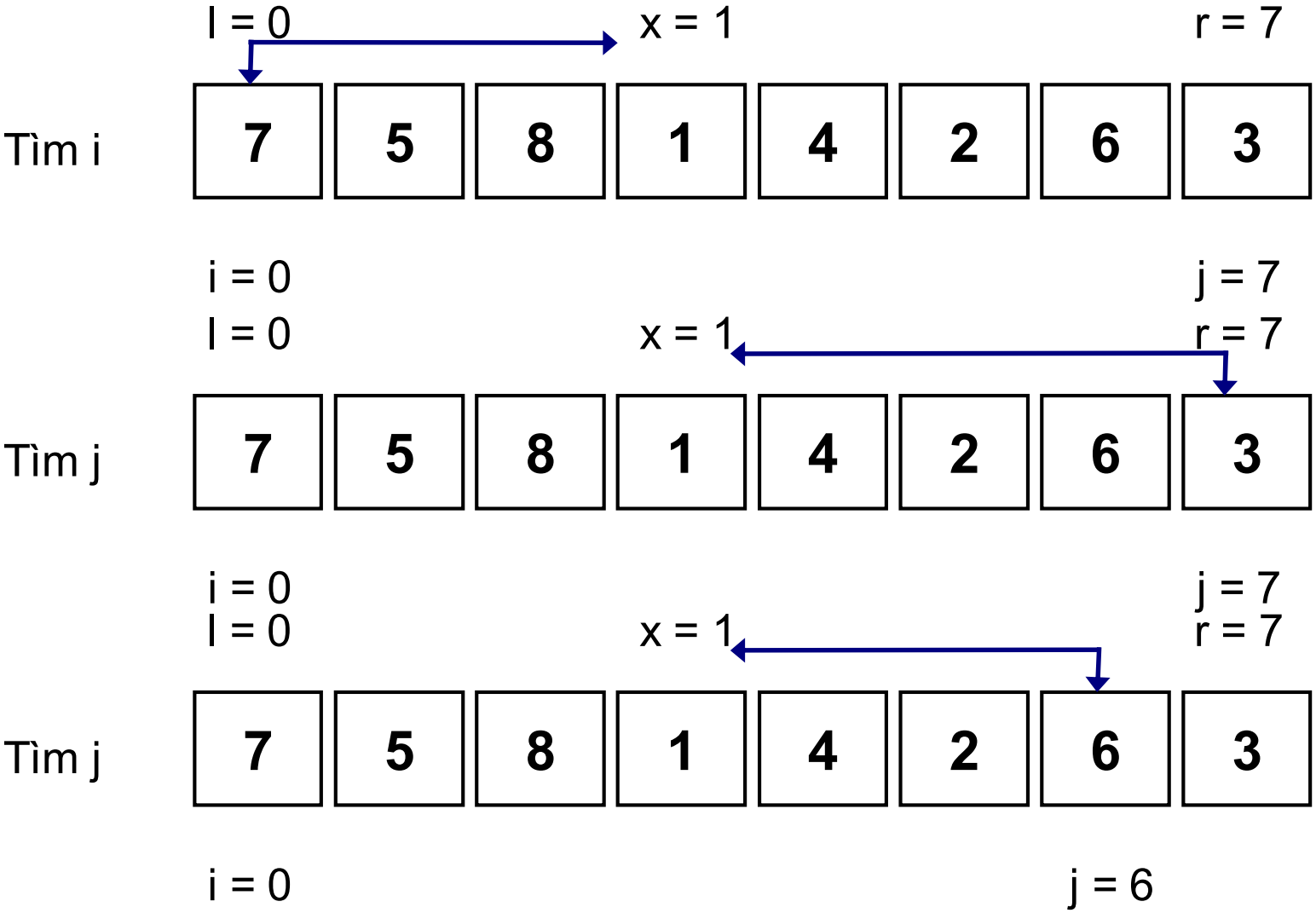
❖PHƯƠNG PHÁP QUICK SORT

*** Cài đặt:**

```
void QuickSort(int *a, int l, int r) {
    int i, j, x;
    if (l >= r) return;
    x = a[(l+r)/2]; i = l; j = r;
    while(i < j) {
        while (a[i] < x) i++; while (a[j] > x) j--;
        if (i <= j) {hoandoi(a[i], a[j]); i++; j--; }
    }
    QuickSort(a, l, j); QuickSort(a, i, r);
}
```

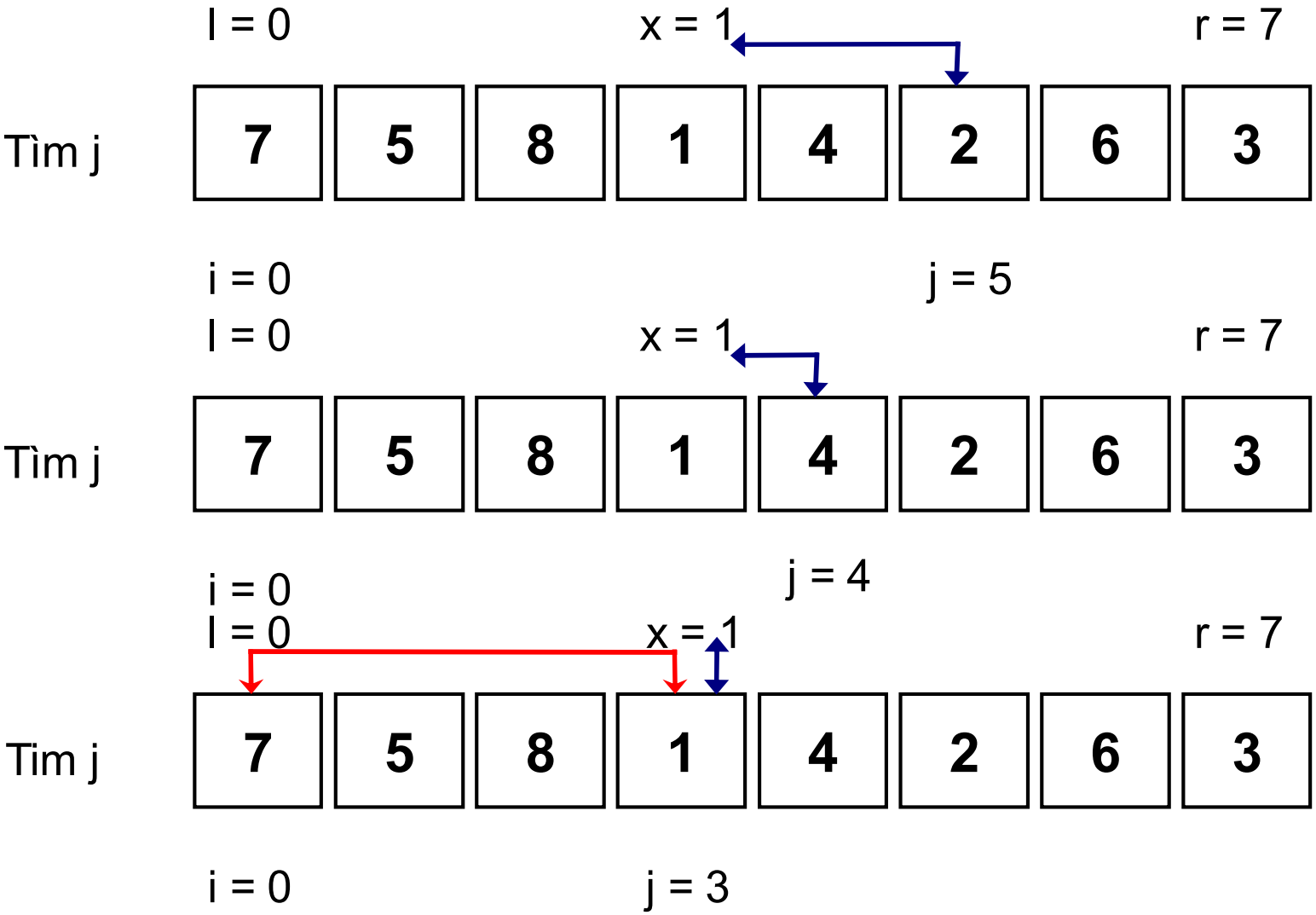
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT



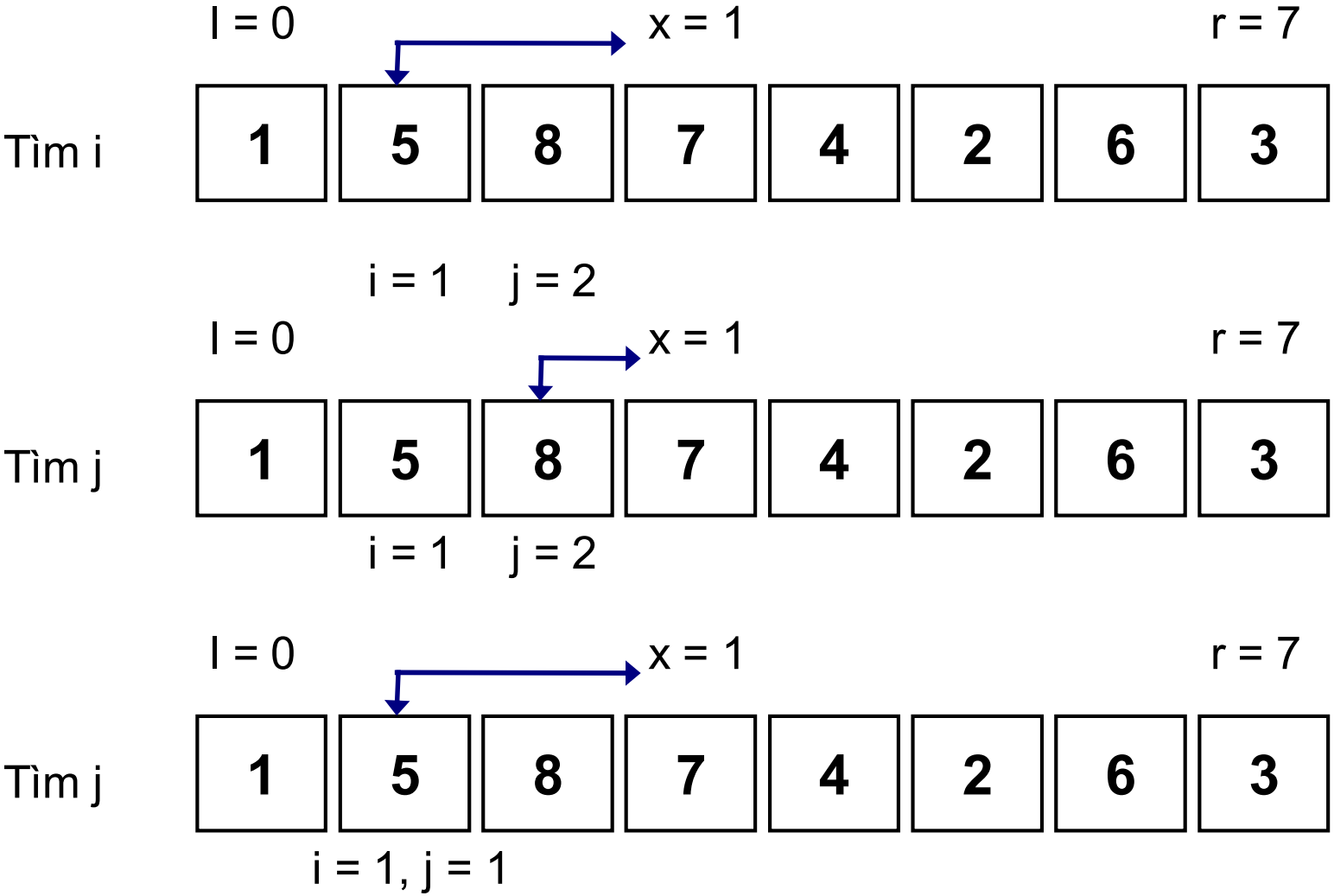
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT



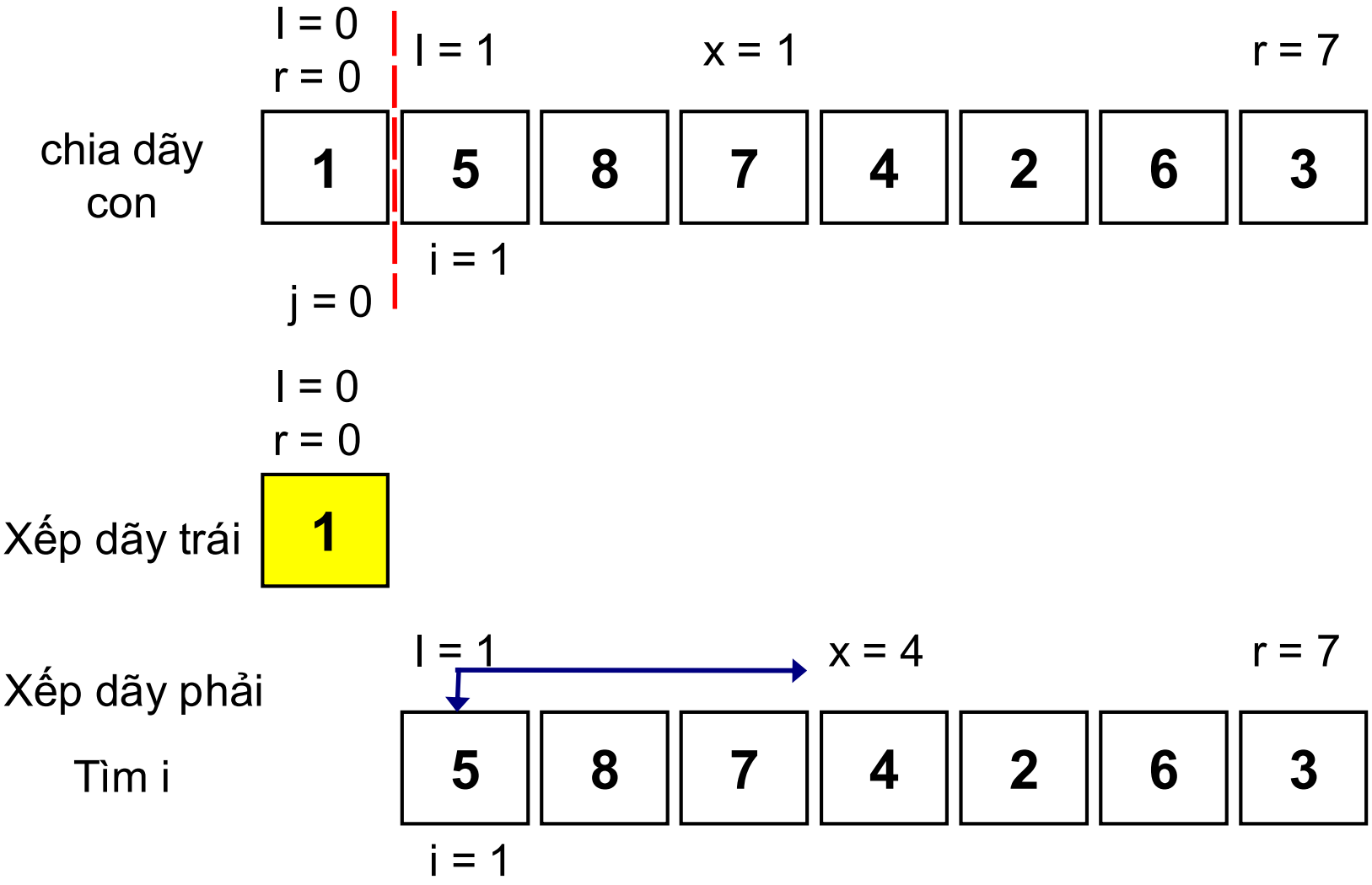
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT



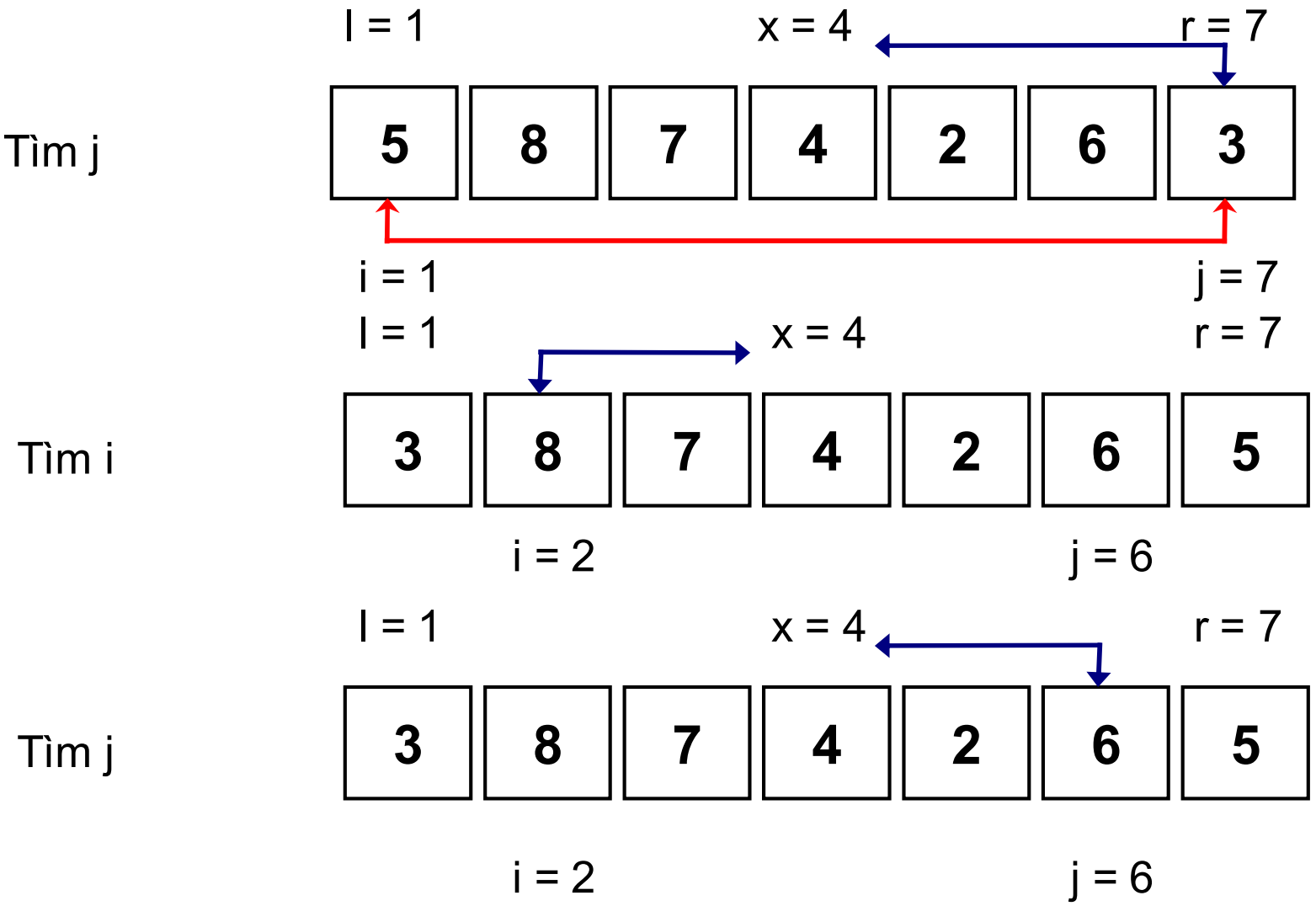
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT



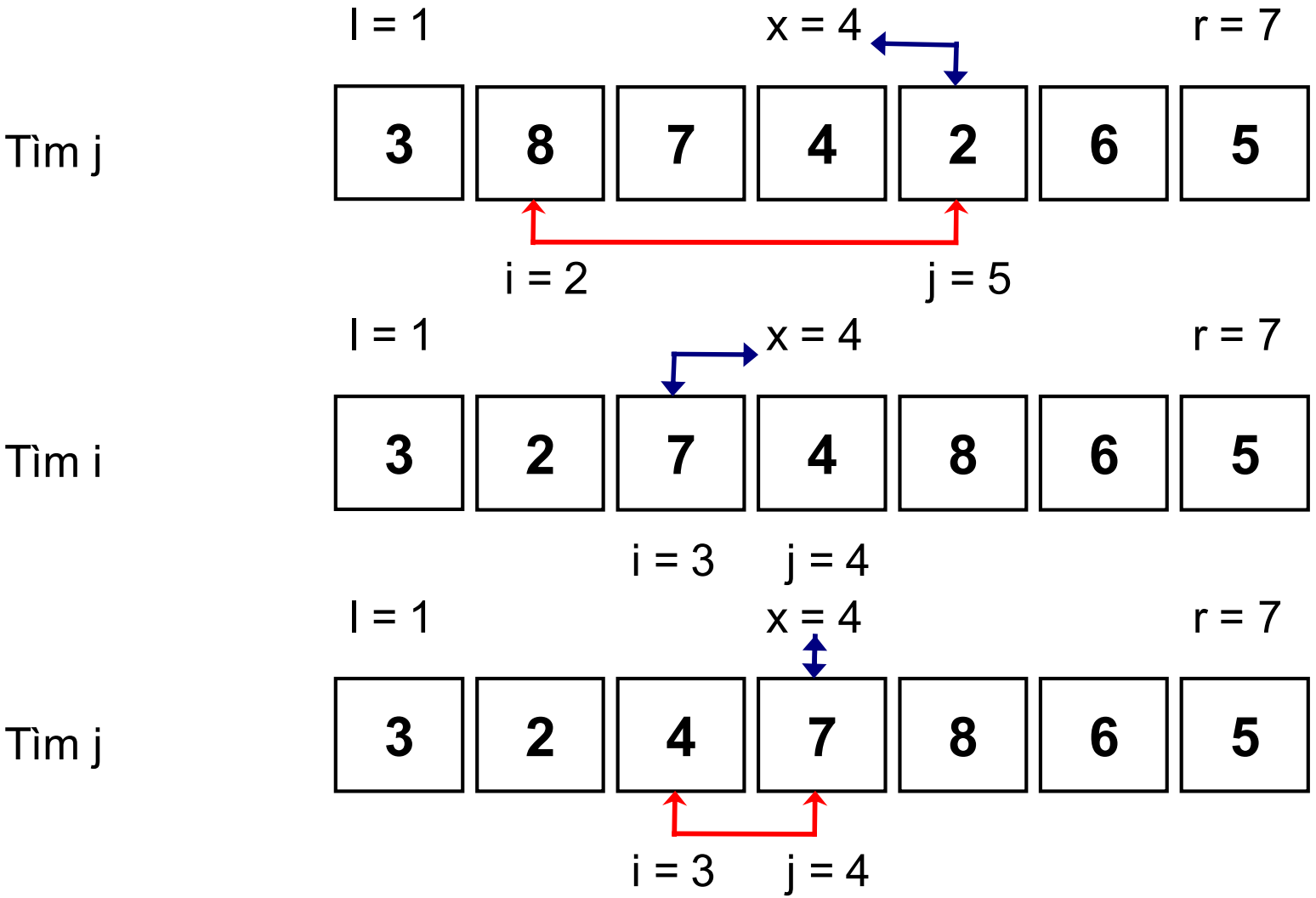
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT



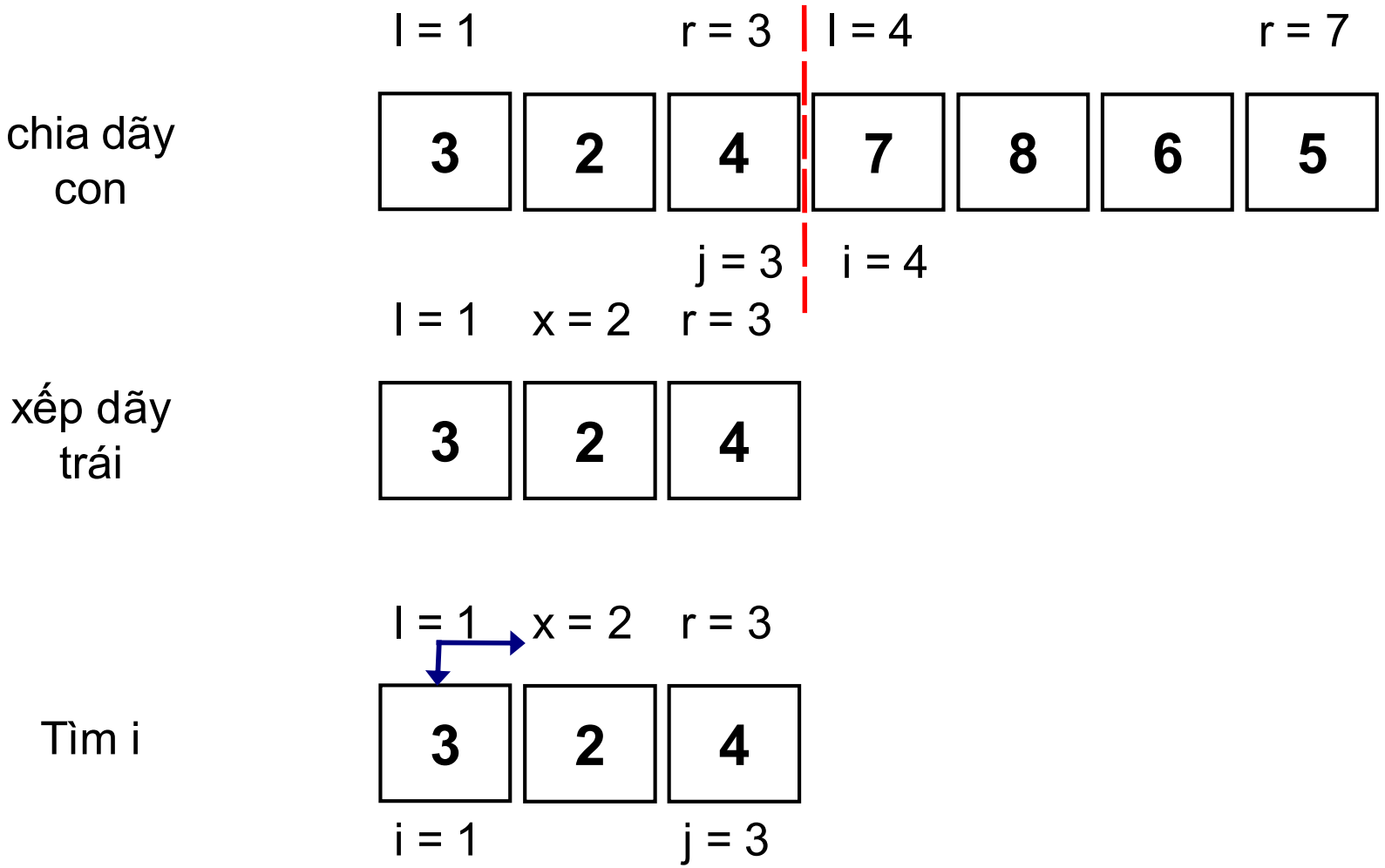
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT



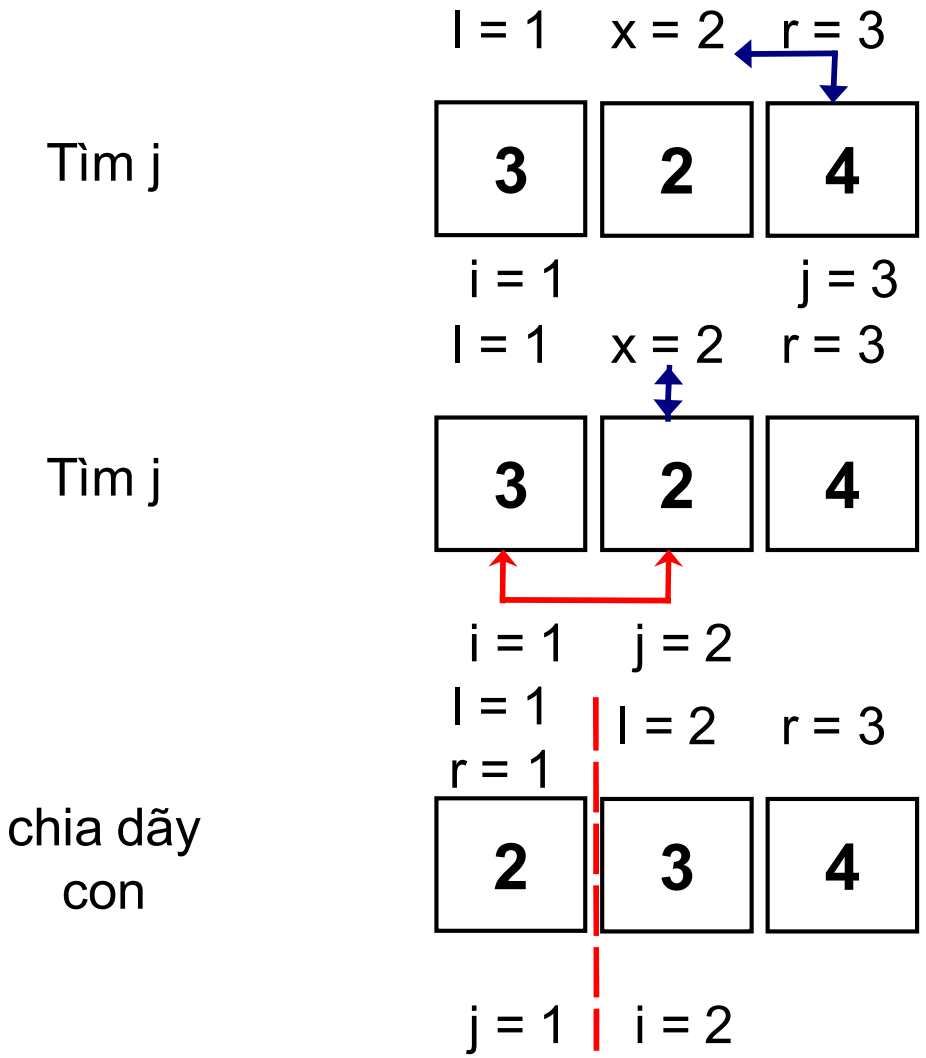
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT



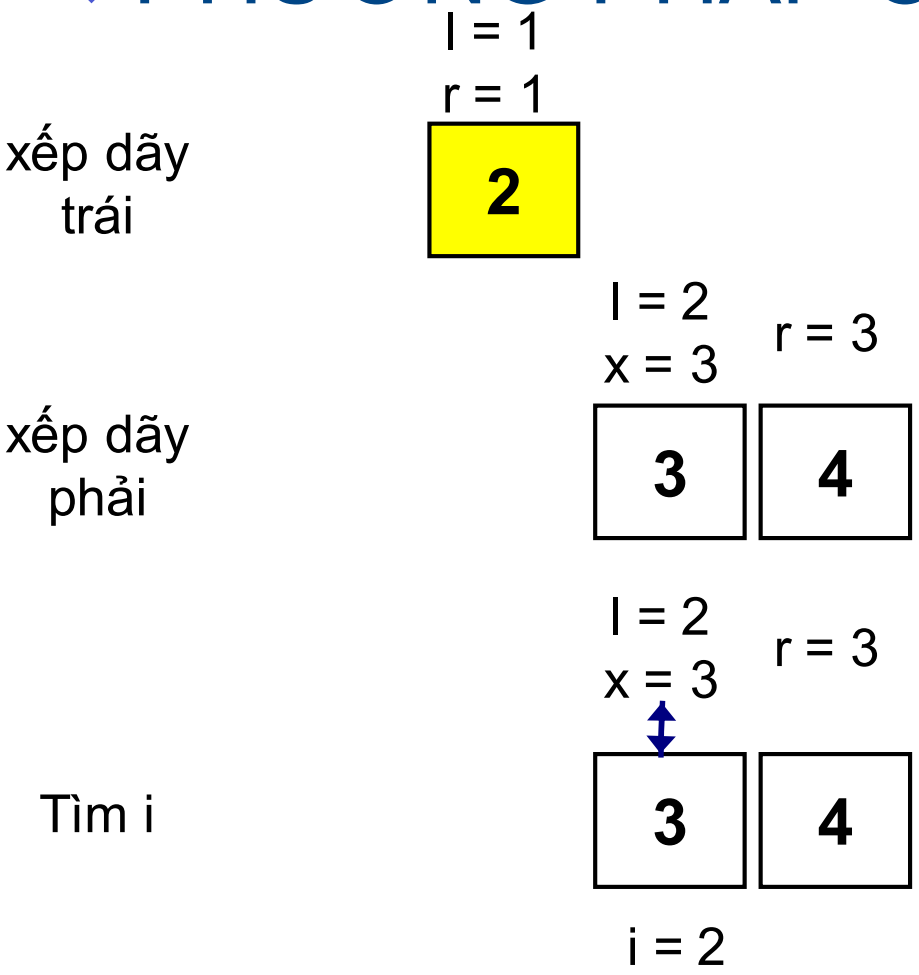
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT



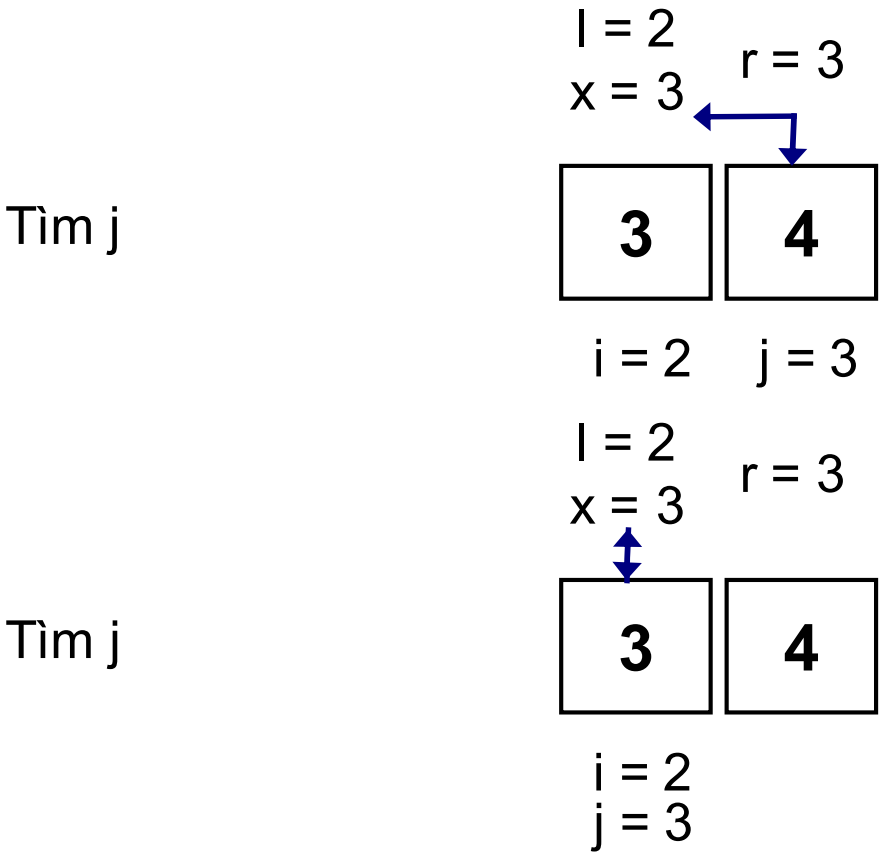
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT



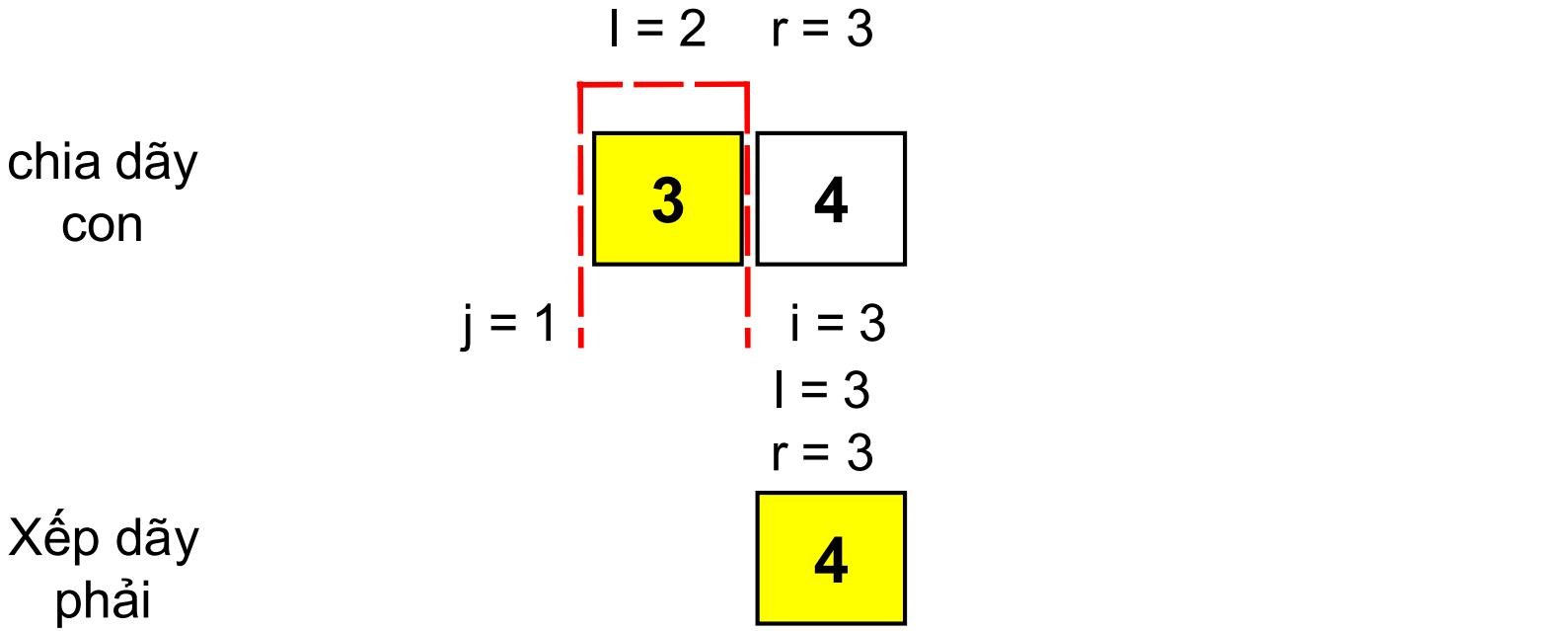
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT



GIẢI THUẬT SẮP XẾP

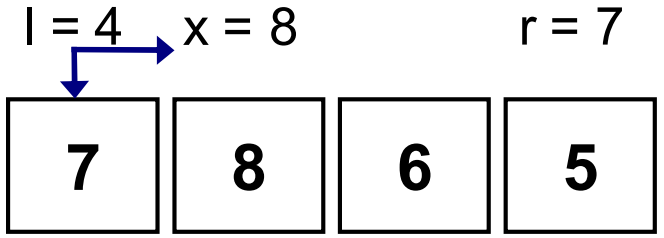
❖PHƯƠNG PHÁP QUICK SORT



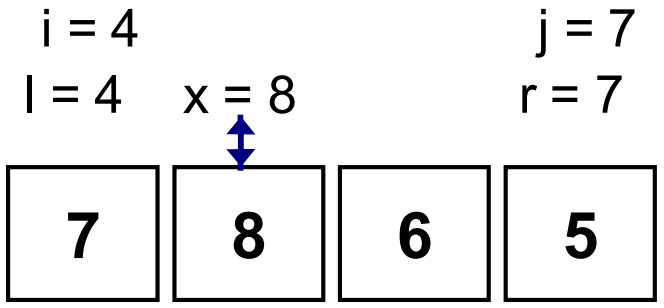
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT

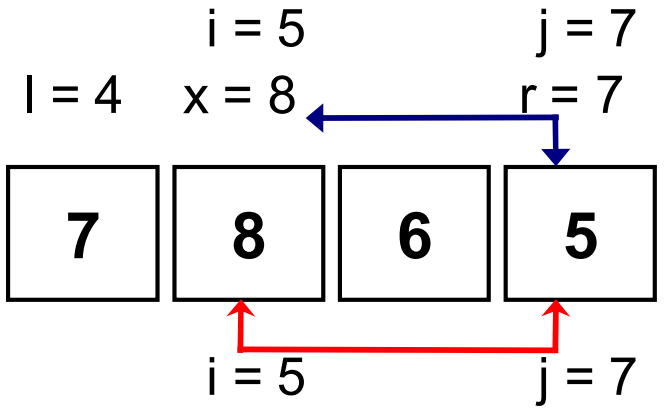
Tìm i



Tìm i



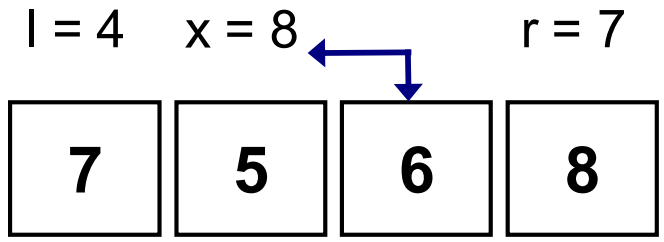
Tìm j

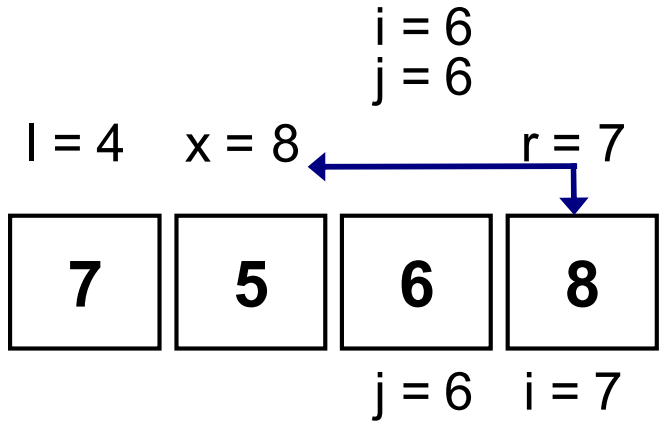


GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT

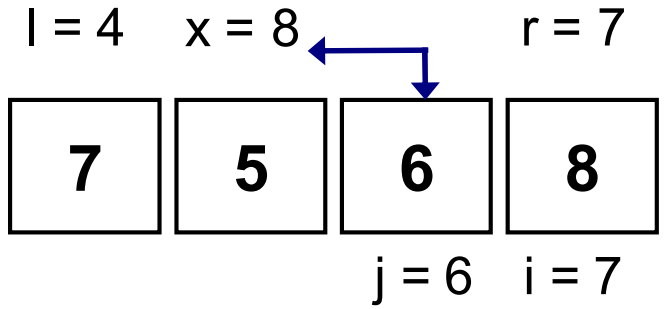
Tìm i





Tìm i

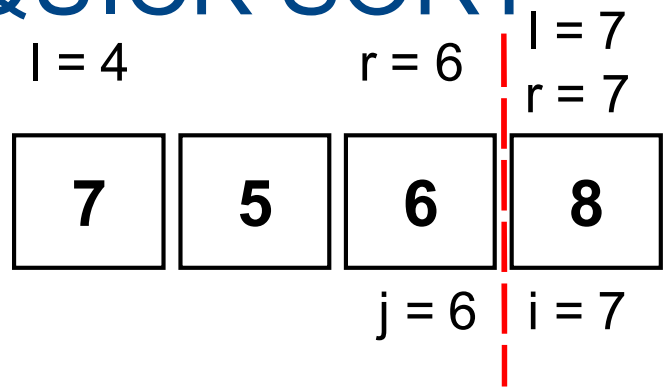
Tìm j



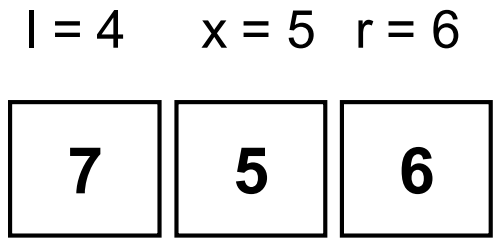
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT

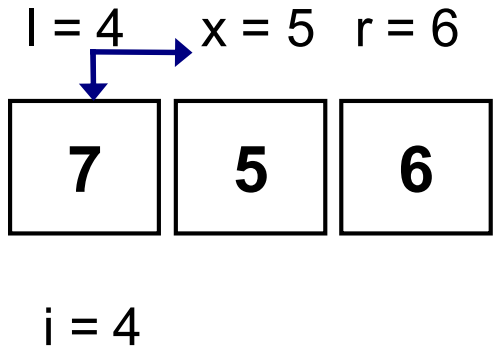
Chia dãy con



Xếp dãy trái



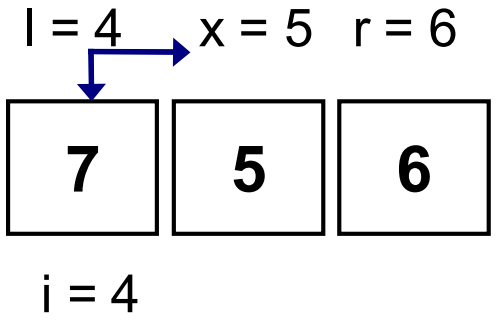
Tìm i



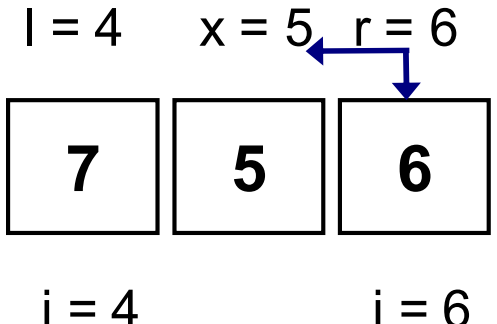
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT

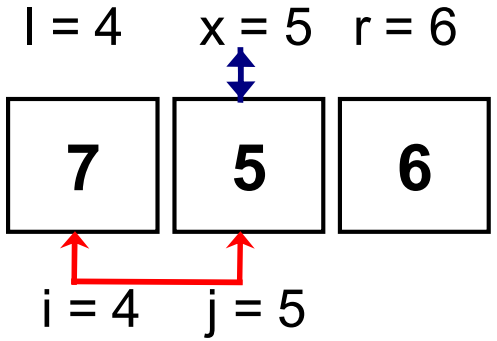
Tìm i



Tìm j



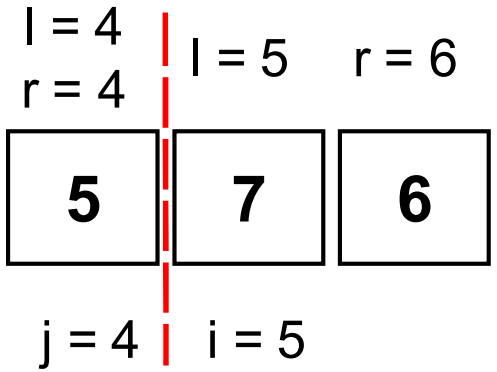
Tìm j



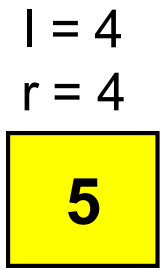
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT

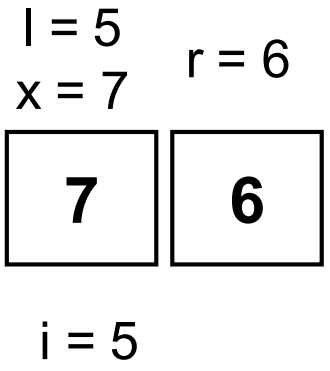
Chia dãy con



Xếp dãy trái



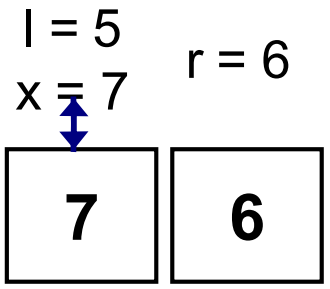
Xếp dãy phải



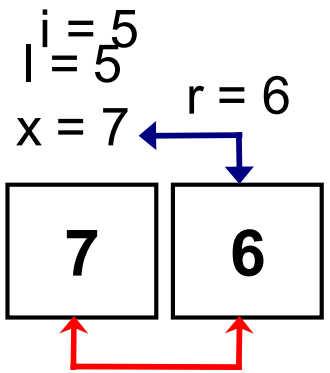
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT

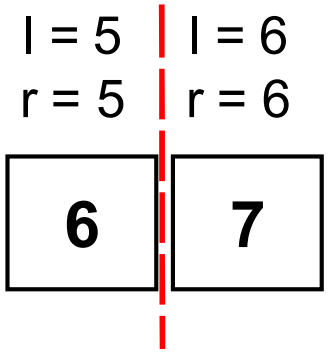
Tìm i



Tìm j

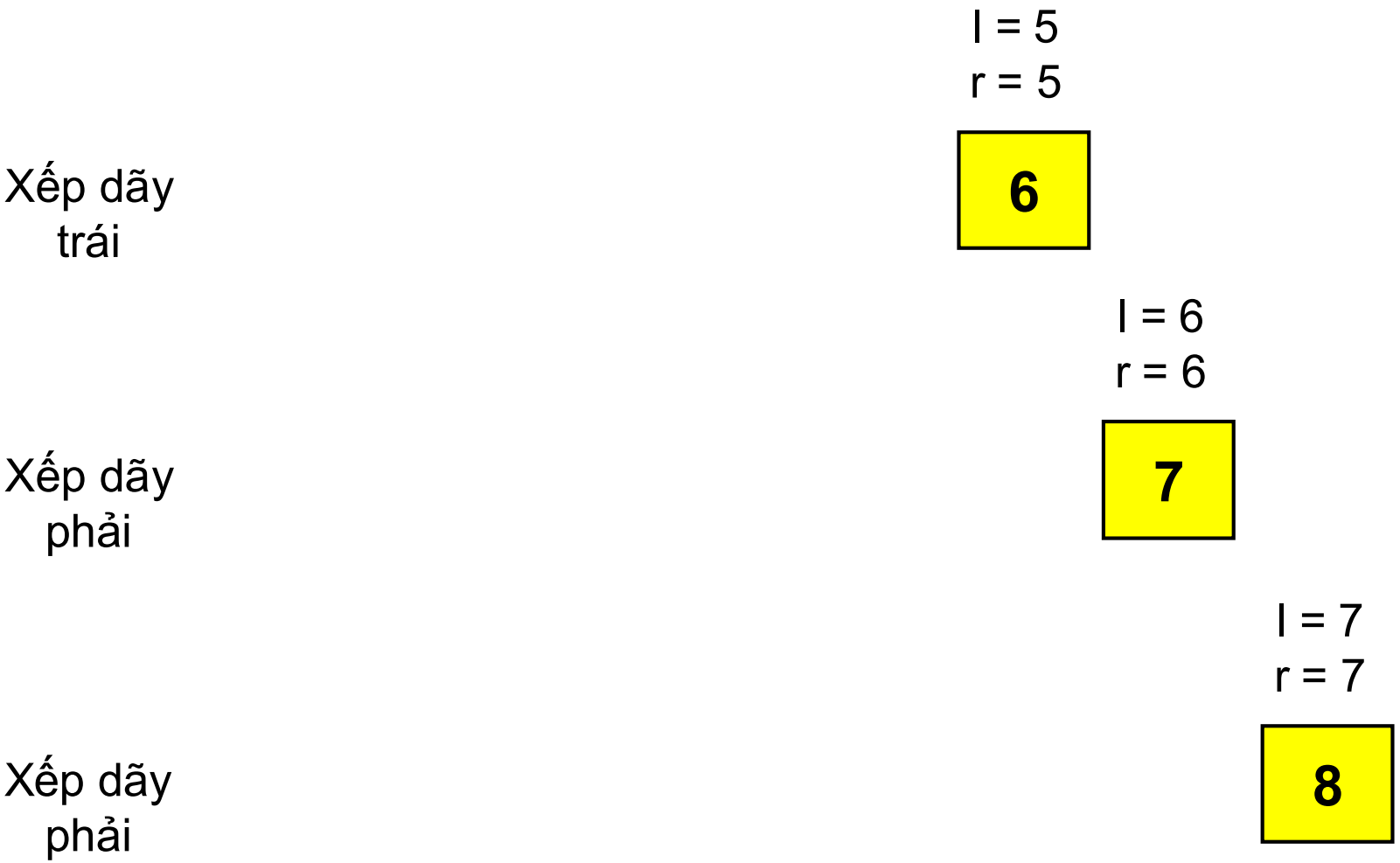


Chia dãy con



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT

*** <u>Đánh giá:</u>** Theo phép so sánh

- Trường hợp tốt nhất: O(nlogn)
- Trường hợp trung bình:O(nlogn)
- Trường hợp xấu nhất: $O(n^2)$

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT

*** Nhận xét:**

- Việc lựa chọn phần tử chốt ảnh hưởng đến chí phí thực hiện thuật toán. Phần tử chốt tốt nhất nếu nó là phần tử trung vị (median) của dãy và xấu nhất nếu nó là phần tử lớn nhất hoặc nhỏ nhất của dãy. Phần tử trung vị là phần tử có giá trị lớn một nửa số phần tử của dãy chứa nó.
- Để hạn chế khả năng xảy ra trường hợp xấu nhất, có thể chọn tùy ý 3 hoặc 5 giá trị, sau đó chọn median của các giá trị này.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP QUICK SORT

*** Nhận xét:**

- Quick sort có hiệu quả khi sắp xếp các dãy số lớn mà không có ràng buộc về thời gian.
- Heap sort thích hợp với yêu cầu sắp xếp trong thời gian giới hạn đối với mọi trường hợp.
- Đối với dữ liệu nhỏ, nên sử dụng các phương pháp sắp xếp cơ bản, ví dụ Selection Short (thậm chí là Bubble Sort), vì:
  - Cài đặt dễ dàng.
  - Chi phí tính toán nhiều hơn các phương pháp khác không đáng kể.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT

*** Ý tưởng:** (Merge sort – Sắp xếp trộn)

Dãy A chưa có thứ tự khi nó chứa nhiều dãy con có thứ tự nhưng thứ tự này không được duy trì từ dãy con này sang dãy con khác. Việc trộn các phần tử của các dãy này theo thứ tự cần xếp sẽ tạo nên dãy A có thứ tự.

Ví dụ: A = {1,5,6,2,3,4} với thứ tự cần xếp là tăng dần. A có A_1={1,5,6} và A_2={2,3,4}. Trộn hai dãy A1 và A2 theo thứ tự phần tử nhỏ được đưa vào A trước

GIẢI THUẬT SẮP XẾP

- PHƯƠNG PHÁP MERGE SORT

1) A_1={5,6}, A_2={2,3,4}, A = {1}
2) A_1={5,6}, A_2={3,4}, A = {1,2}
3) A_1={5,6}, A2={4}, A = {1,2,3}
4) A_1={5,6}, A_2={}, A = {1,2,3,4}
5) A_1={}, A2={}, A = {1,2,3,4,5,6}

GIẢI THUẬT SẮP XẾP

❖ PHƯƠNG PHÁP MERGE SORT

*** Ý tưởng:**

Với dãy A có n phần tử, ban đầu có thể xem A gồm n dãy con có thứ tự, mỗi dãy có k=1 phần tử. Chia dãy A thành hai dãy A_1 và A_2, sau đó trộn từng cặp gồm 1 dãy con của A_1 và 1 dãy con của A_2 thành dãy A. Kết quả là dãy A sẽ có $\lceil n/2 \rceil$ dãy cần trộn, mỗi dãy có k=2 phần tử. Thực hiện tương tự, mỗi lần số dãy sẽ được giảm một nữa và số phần tử k mỗi dãy tăng gấp đôi. Thực hiện đến khi số dãy còn 1, tức là dãy A có thứ tự. Phương pháp này được gọi là trộn trực tiếp.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT

*** Giải Thuật:**

Đầu vào: dãy $a_1,a_2,..,a_n$ chưa có thứ tự

Đầu ra: dãy $a_1,a_2,..,a_n$ đã có thứ tự tăng dần

- B1: k←1
- B2: Tách dãy thành hai dãy B và C bằng cách phân phối luân phiên k phần tử cho mỗi dãy

  B = $a_1,..,a_k,a_{2k+1},..,a_{3k},...$ C = $a_{k+1},..,a_{2k},a_{3k+1},..,a_{4k},..$
- B3: Trộn từng cặp dãy con k phần tử của B và C vào A
- B4: k←2*k, nếu k < n thì qua B2.
- B5: Kết thúc.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT

*** Cài đặt:**

```
#define min(x,y) (x > y) ? y : x
int b[MAX], c[MAX];
void Merge(int *a, int nb, int nc, int k) {
    int p=0,pb=0,pc=0,ib =0,ic = 0,kb,kc;
    kb = min(k, nb); kc = min(k, nc);
    while ((nb > 0) && (nc > 0)) {
        if (b[pb+ib] <= c[pc+ic]) {
            a[p++]=b[pb+ib]; ib++;
            if (ib == kb) {
                for (; ic<kc; ic++) a[p++] = c[pc+ic];
```

GIẢI THUẬT SẮP XẾP

- PHƯƠNG PHÁP MERGE SORT

```
                pb+=kb;pc+=kc; ib=0;ic=0;nb-=kb;nc-=kc;
                kb = min(k, nb); kc = min(k, nc);
            }
        }
        else {
            a[p++]=c[pc+ic]; ic++;
            if (ic == kc) {
                for (; ib<kb; ib++) a[p++] = b[pb+ib];
                pb+=kb;pc+=kc; ib=0;ic=0;nb-=kb;nc-=kc;
                kb = min(k, nb); kc = min(k, nc);
            }
        }
    }
}
```

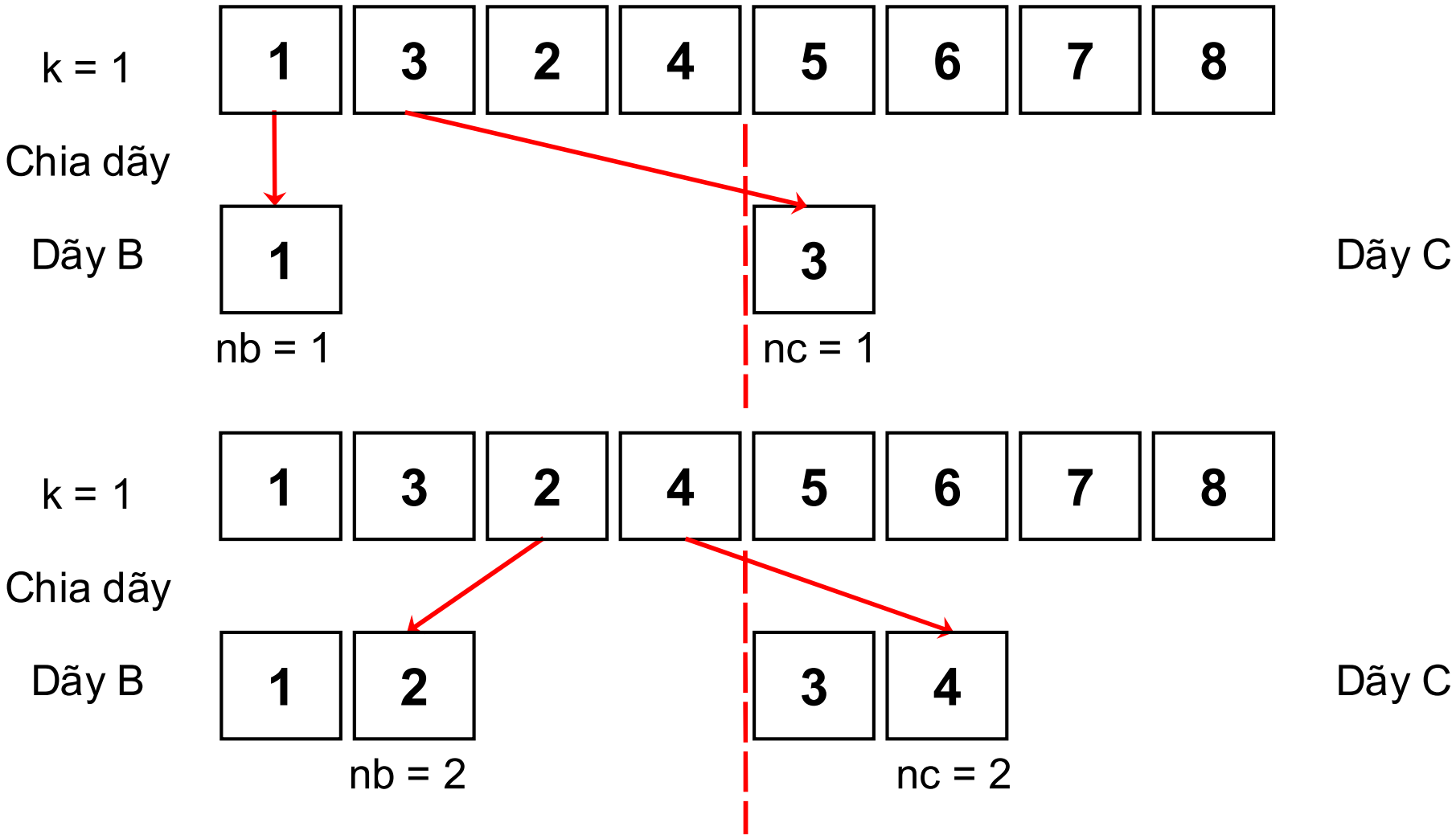
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT

```
void MergeSort(int *a, int n) {
    int p, pb, pc, i, k = 1;
    while (k < n) {
        p = 0; pb = 0; pc = 0;
        while (p < n) {
            for (i = 0; (p < n) && (i < k); i++) b[pb++]=a[p++];
            for (i = 0; (p < n) && (i < k); i++) c[pc++]=a[p++];
        }
        Merge(a, pb, pc, k);
        k *= 2;
    }
}
```

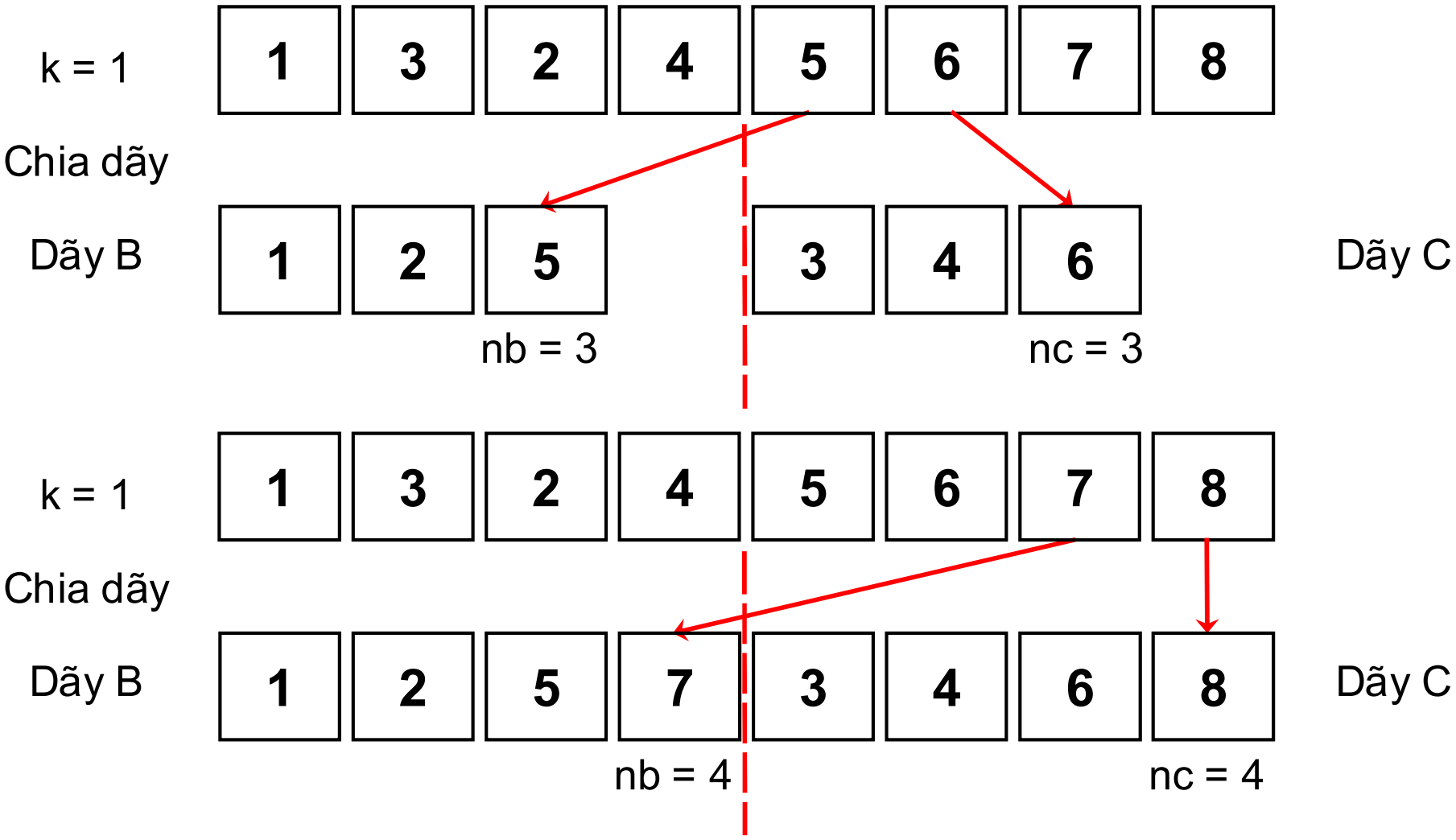
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT



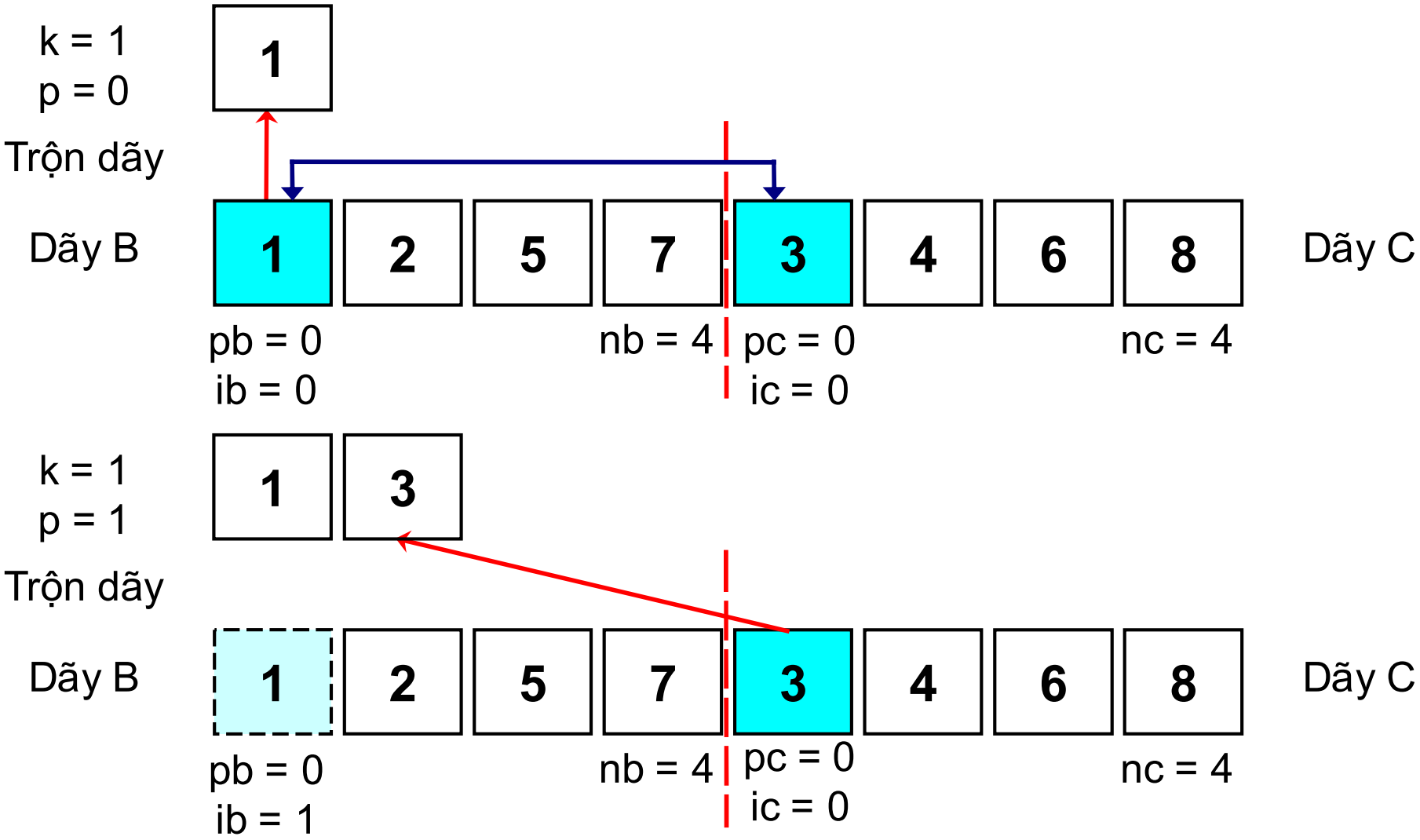
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT



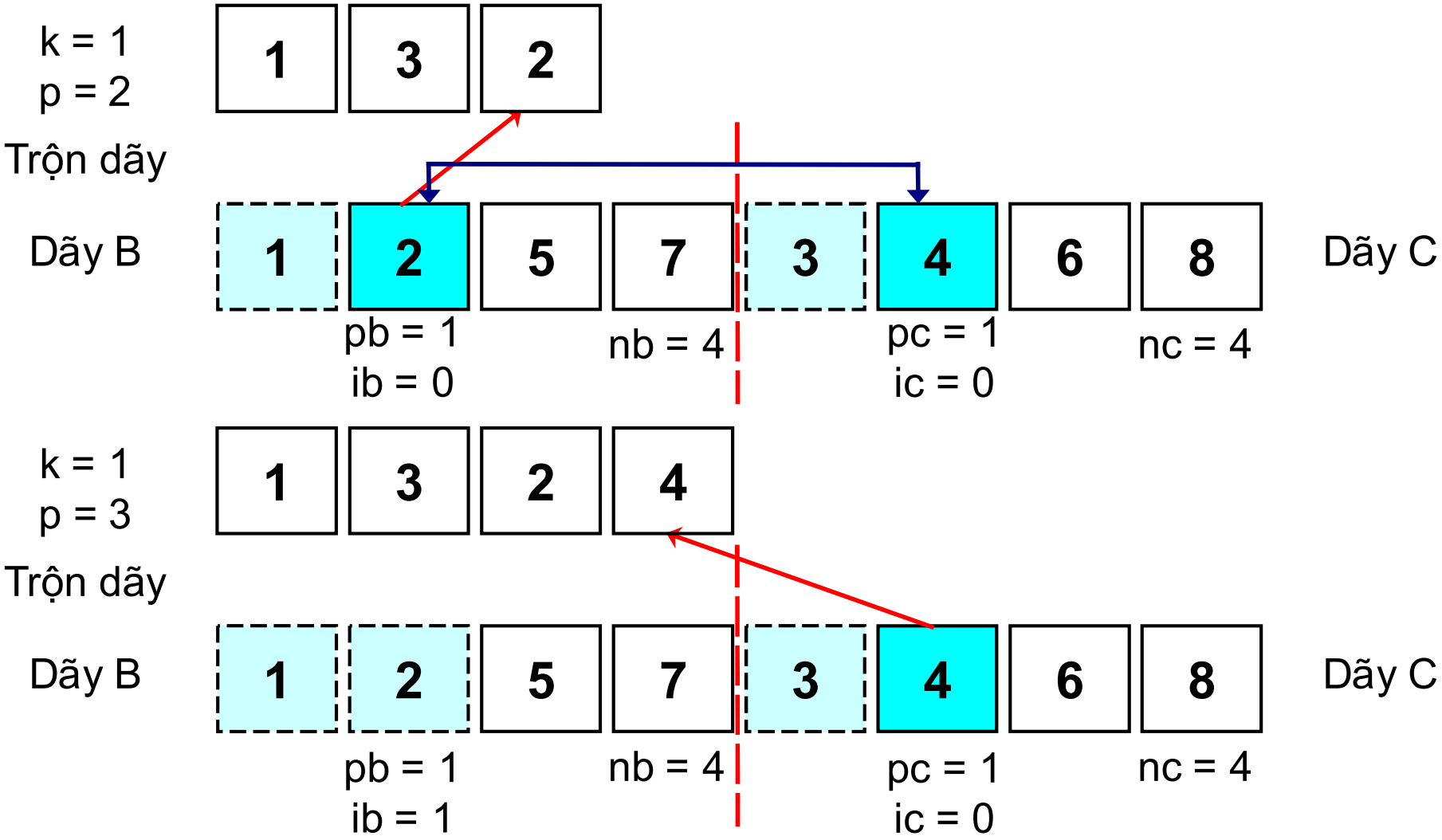
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT



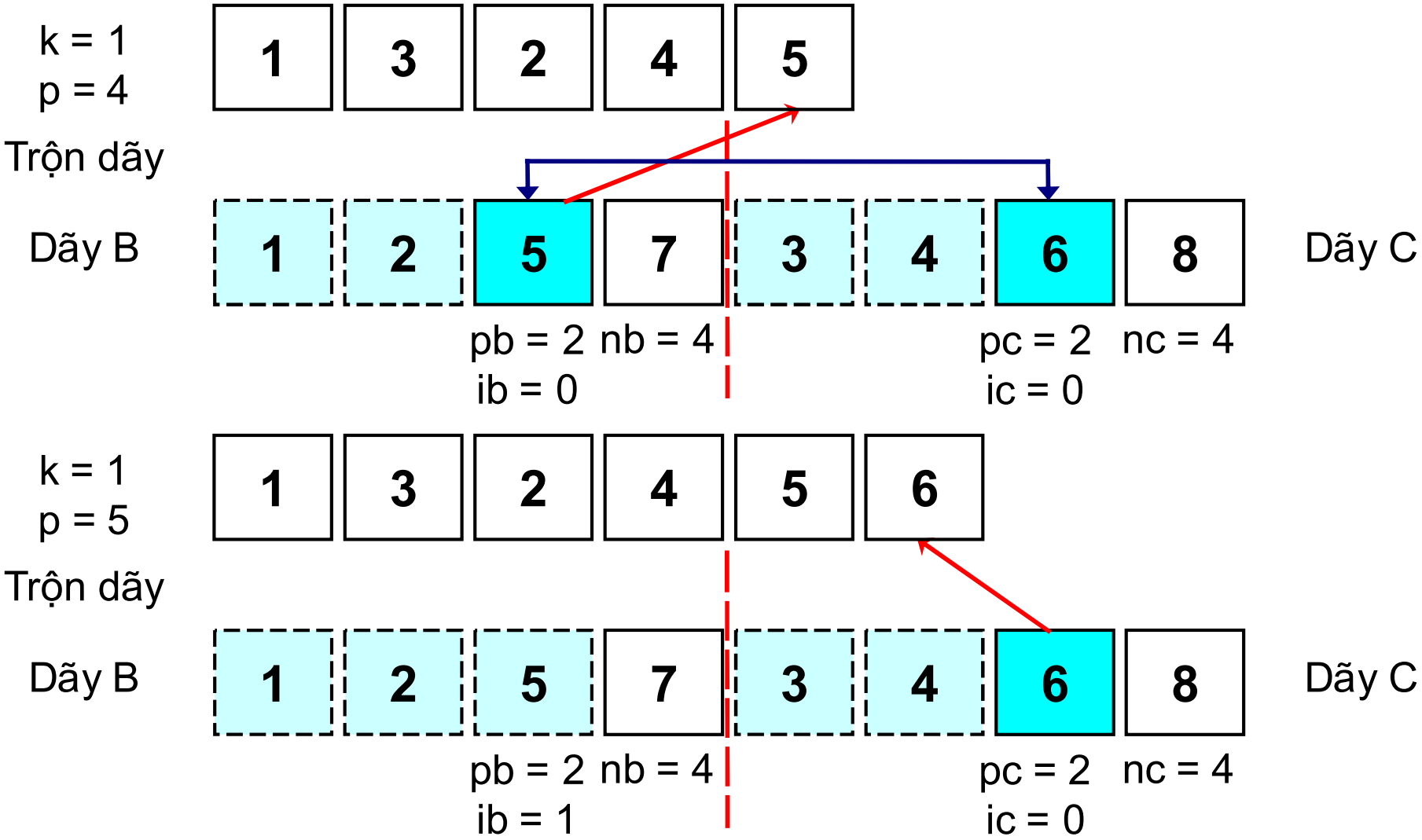
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT



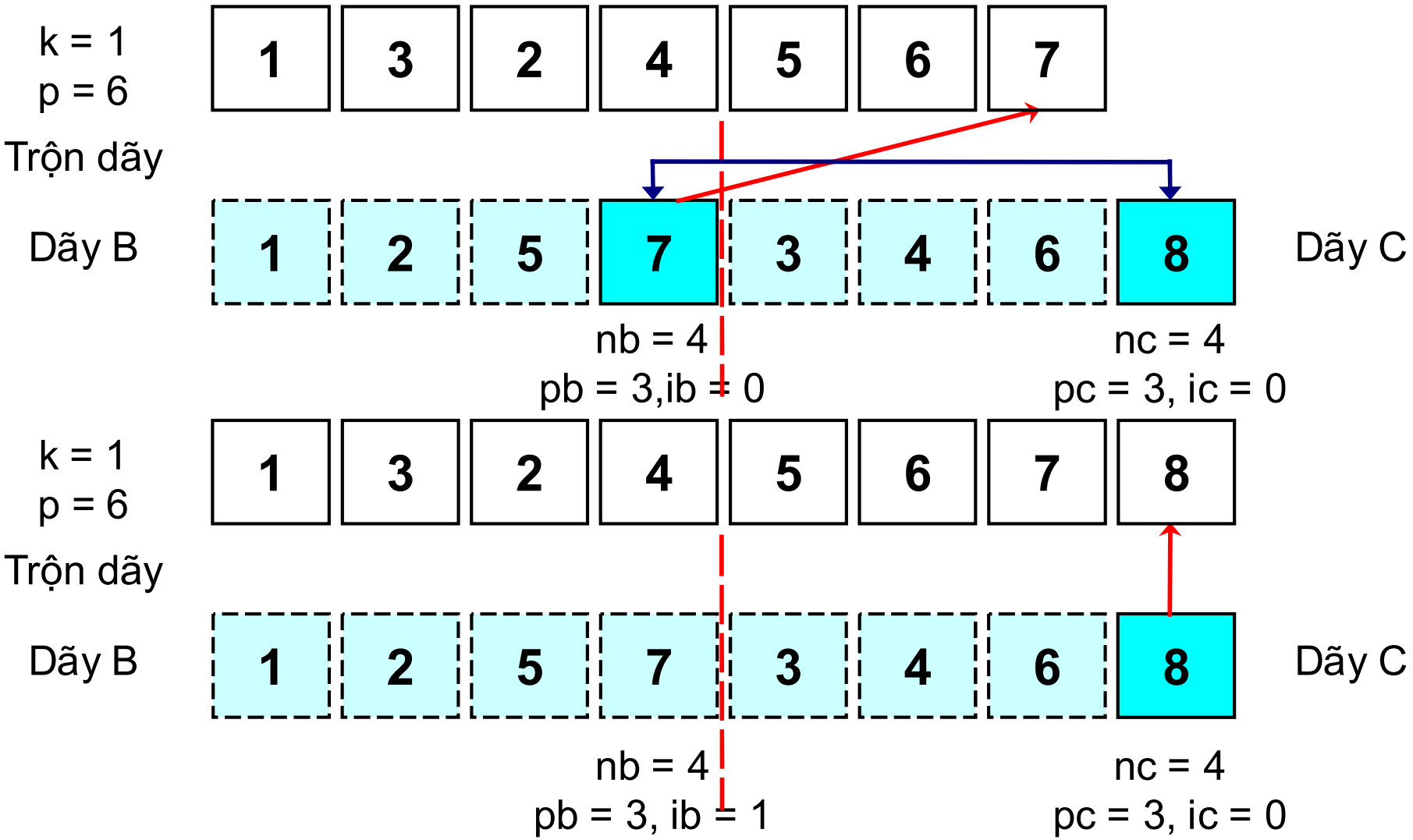
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT



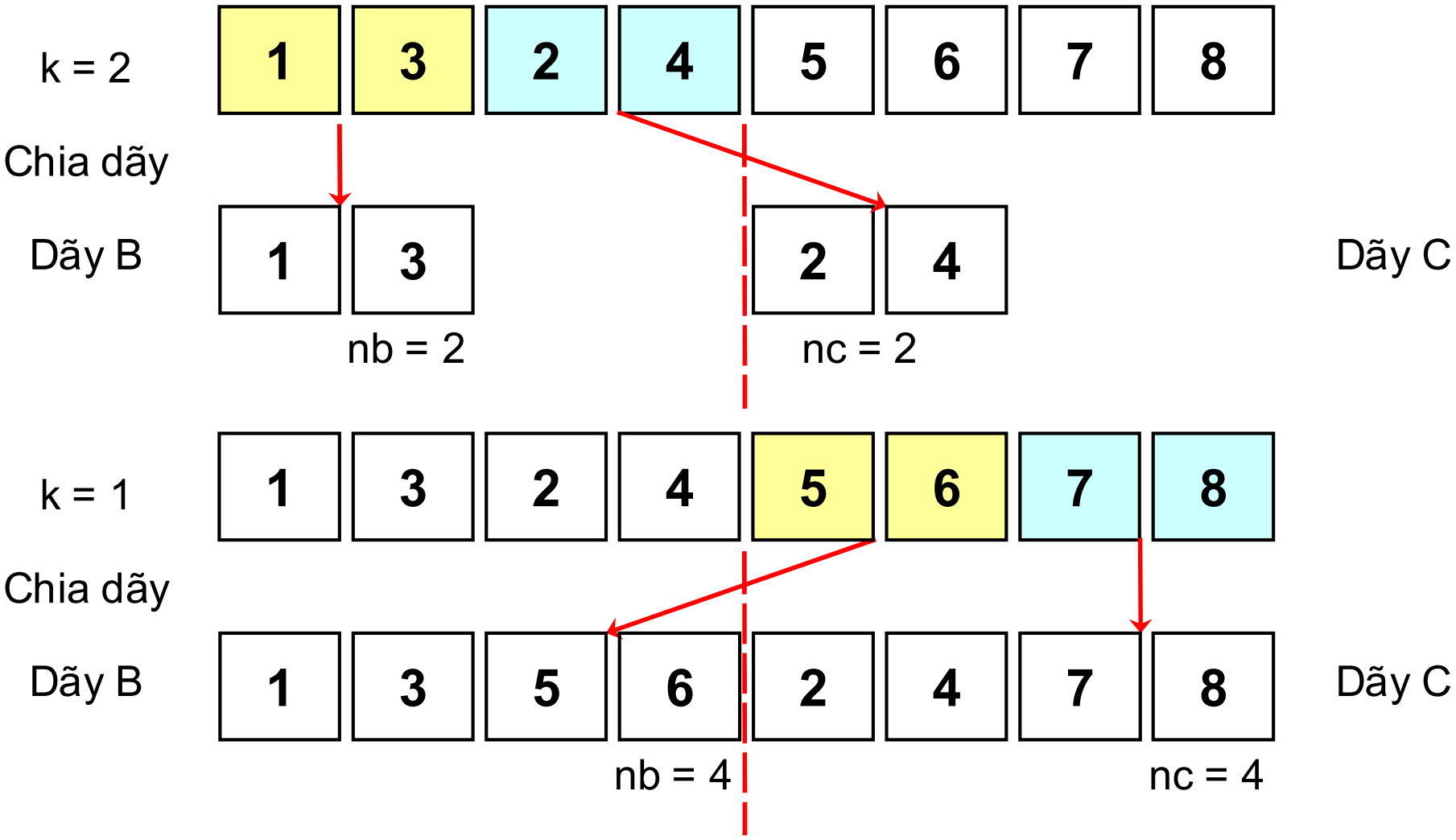
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT



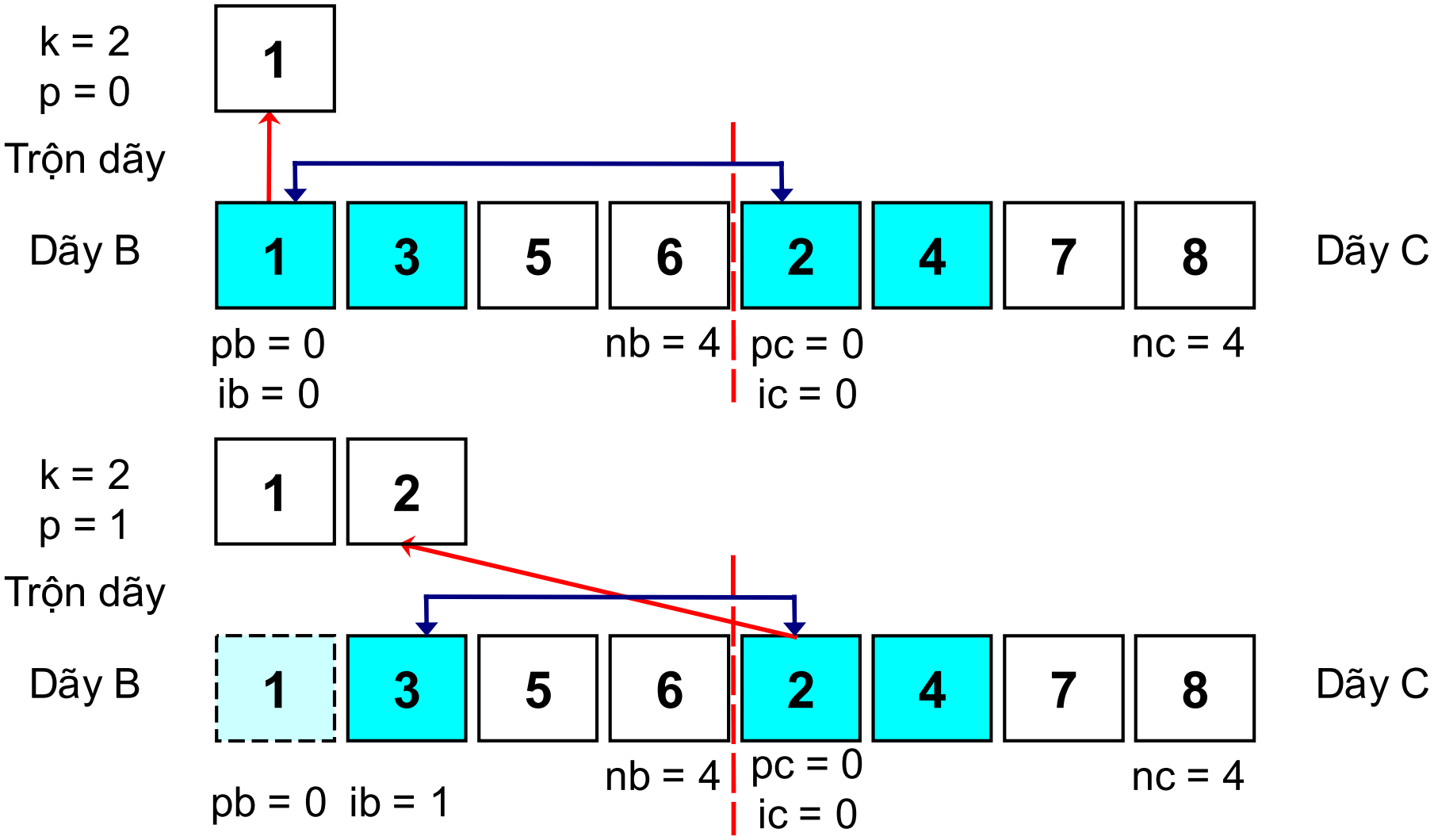
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT



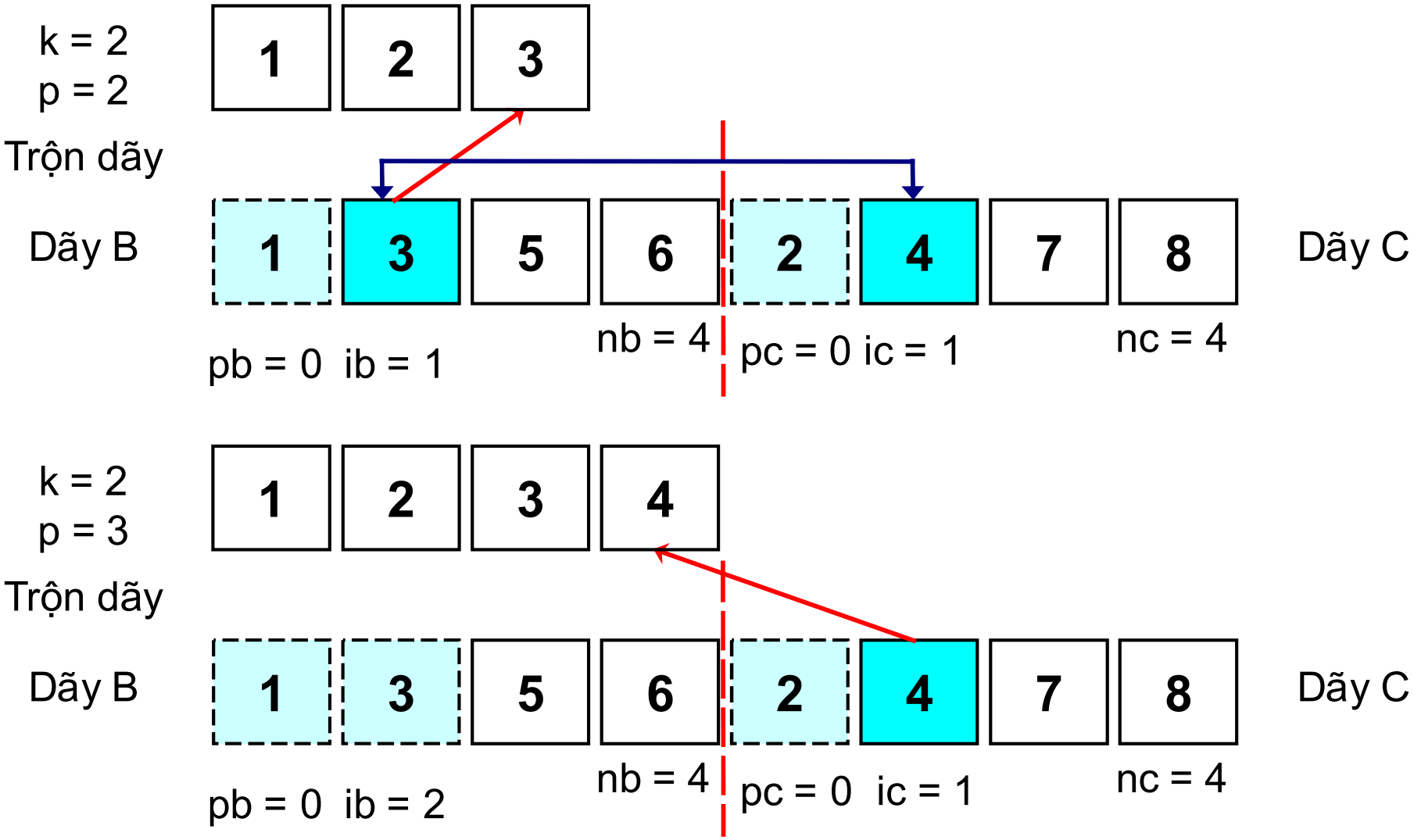
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT



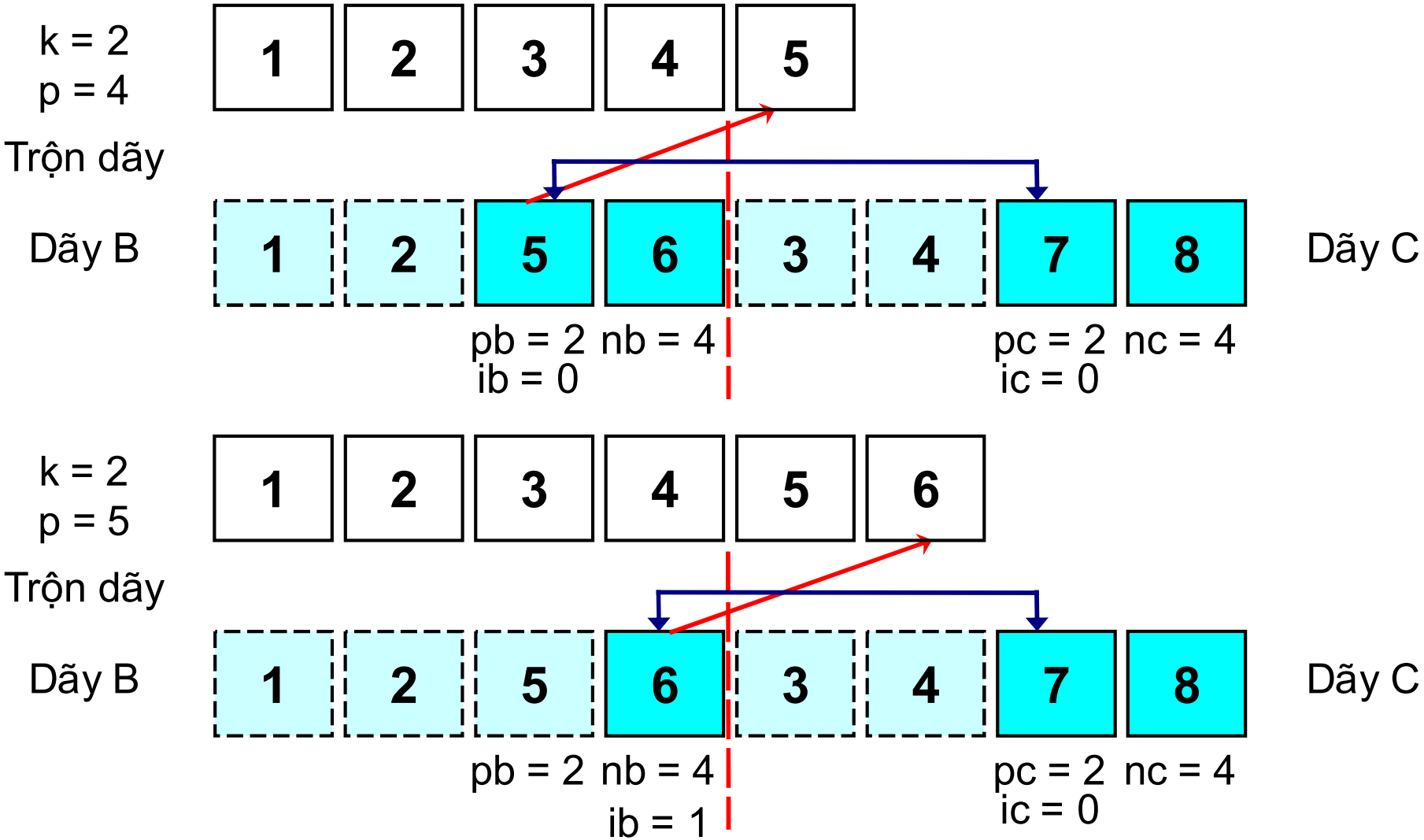
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT



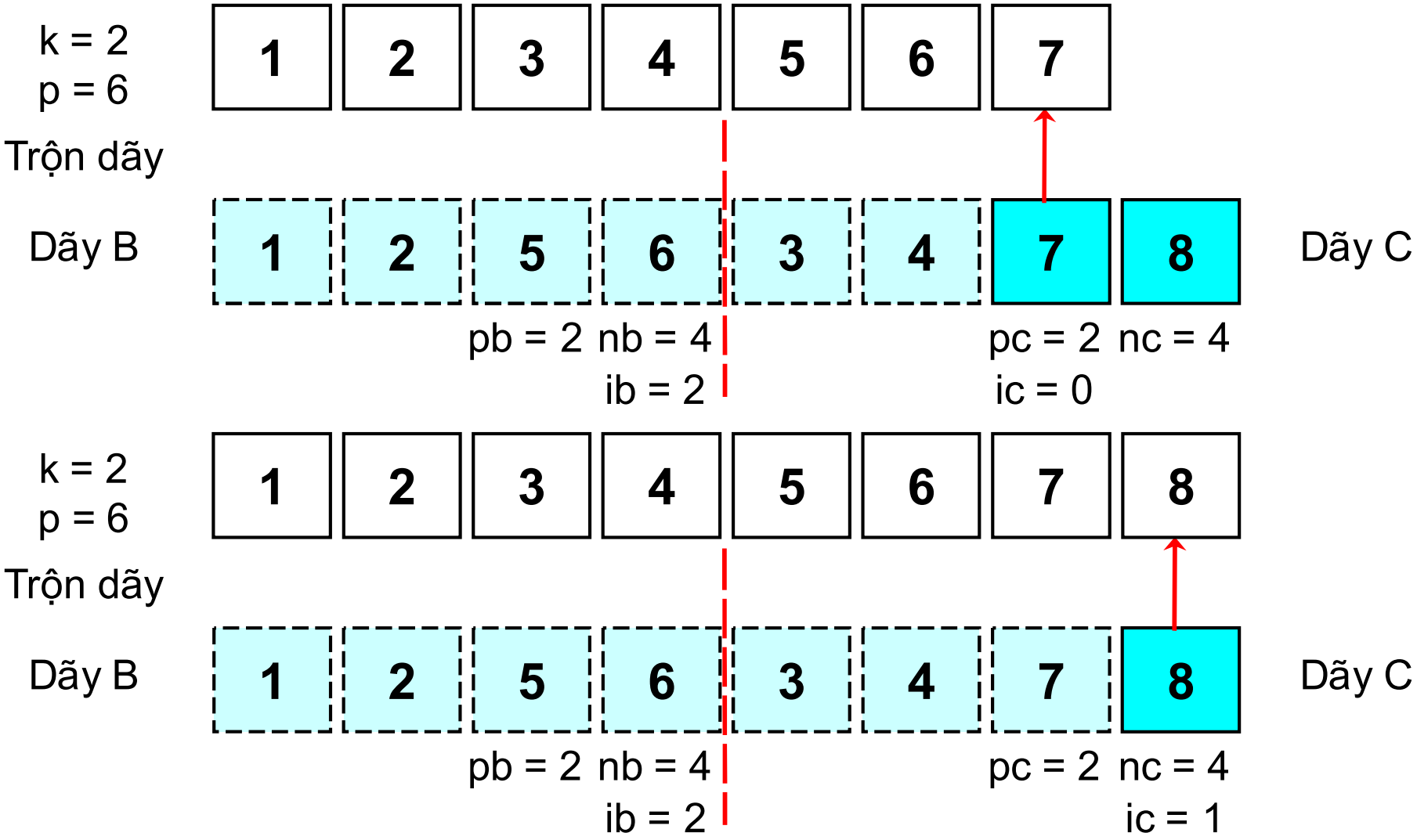
GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT



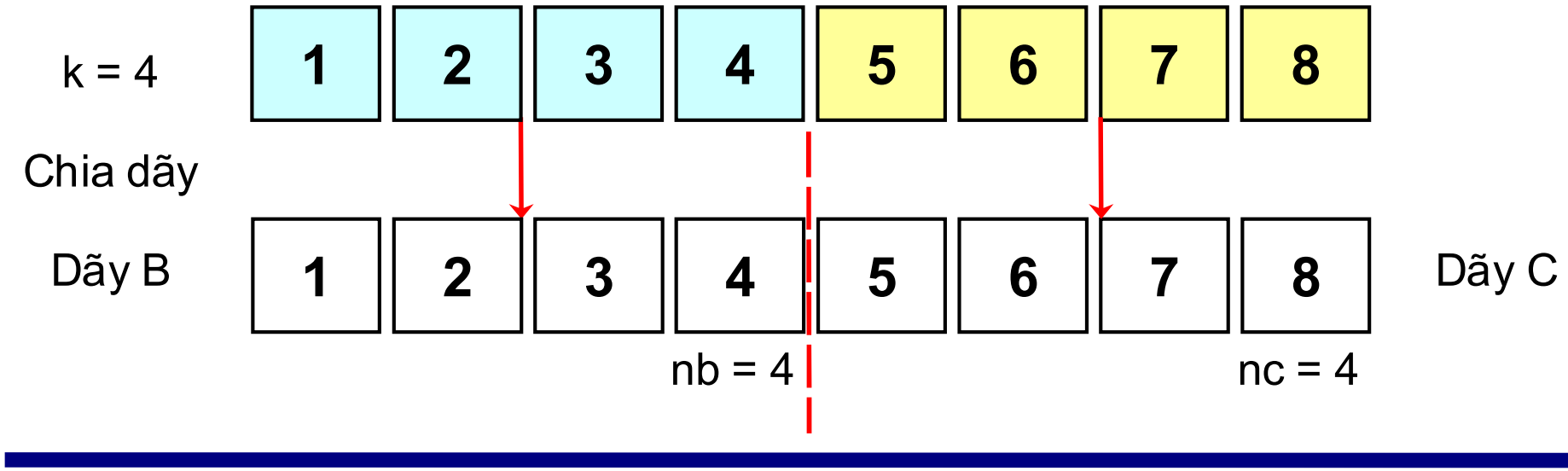
GIẢI THUẬT SẮP XẾP

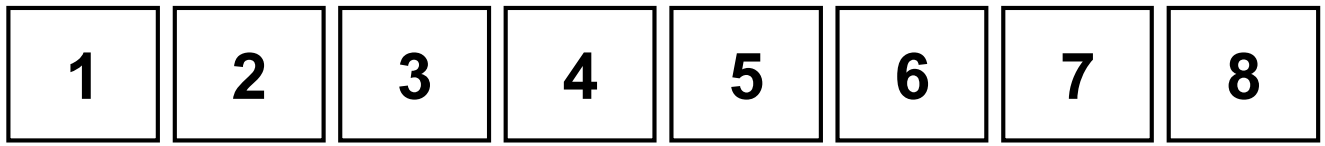
❖PHƯƠNG PHÁP MERGE SORT



GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT





GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT

*** Đánh giá độ phức tạp của giải thuật:**

Độ phức tạp của giải thuật Merge Sort như trên trong tất cả trường hợp cho cả phép so sánh và phép gán là O(nlogn).

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT

*** Nhận xét:**

- Giải thuật Merge Sort như trên chưa tận dụng được đặc điểm của dãy cần sắp xếp.

- Quá trình chia dãy và trộn hai dãy con cần thêm không gian bộ nhớ nên cần áp dụng cho cấu trúc dữ liệu thích hợp hơn như danh sách liên kết hoặc file.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT

*** Trộn tự nhiên (Natural Merge Sort):**

Là phương pháp cải tiến để việc phân chia dãy con phù hợp với đặc điểm của dãy hơn. Thay vì xuất phát từ n dãy con có thứ tự gồm 1 phần tử, trộn tự nhiên xuất phát từ r dãy con đã có thứ tự sẵn gọi là đường chạy (run). Quá trình chia dãy sẽ phân phối luân phiên các run thành 2 dãy con.

Ví dụ, đối với dãy A = {1,3,2,5,4,6,7,8} sẽ có các run là {1,3}, {2,5} và {4,6,7,8}. Hai dãy con sẽ là A_1={{1,3},{5,6,7,8}} và A_2={2,4}

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT

*** Giải thuật trộn tự nhiên:**

Đầu vào: dãy A = $\{a_1,a_2,..,a_n\}$ chưa có thứ tự

Đầu ra: dãy A = $\{a_1,a_2,..,a_n\}$ đã có thứ tự tăng dần

- B1: r $\leftarrow$ 0
- B2: Nếu chưa phân phối hết các phần tử của dãy thì phân phối cho B một đường chạy, r $\leftarrow$ r + 1; ngược lại qua B4
- B3: Nếu chưa phân phối hết các phần tử của dãy thì phân phối cho C một đường chạy, r $\leftarrow$ r + 1, qua B2.
- B4: Trộn từng cặp đường chạy của B và C vào A.
- B5: Nếu r > 1, qua B1.
- B6: Kết thúc.

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP MERGE SORT

*** Cài đặt:**

```
#define min(x,y) (x > y) ? y : x
int b[MAX], c[MAX];
void MergeSort(int *a, int n) {
    int p, pb, pc, r, lane, t, i, j, k;
    do{
        r = 0; p = 0; pb = 0; pc = 0; k = 0; lane = 1;
        t = a[p++]; b[pb++] = t;
        while (p < n) { // tách và trộn từng dãy
            if (lane == 1) {
                if (t <= a[p]) { t = a[p++]; b[pb++] = t; }
                else { lane = 2; t = a[p++]; c[pc++] = t; r++;}
            }
```

GIẢI THUẬT SẮP XẾP

- PHƯƠNG PHÁP MERGE SORT

```
else {
      if (t <= a[p]) { t = a[p++]; c[pc++] = t;}
      else { lane = 0; r++; }
}
if ((lane == 0)||(p == n)) {
      i = 0; j = 0;
      while ((pb > i) && (pc > j)) {
            if (b[i] < c[j]) {
                  a[k++] = b[i++];
                  if (i == pb) for ( ;j<pc;j++) a[k++]=c[j];
            }
```

GIẢI THUẬT SẮP XẾP

- PHƯƠNG PHÁP MERGE SORT

```
                    else {
                        a[k++] = c[j++];
                        if (j == pc) for ( ;i <pb;i++) a[k++]=b[i];
                    }
                }
                if (p < n) {
                    pb=0; pc=0; lane=1;t=a[p++]; b[pb++]=t;
                }
            }
        }
    } while (r > 1);
}
```

GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP RADIX SORT

* **Ý tưởng:** Tham khảo giáo trình

* **Thuật toán:** Tham khảo giáo trình