CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

CHƯƠNG II

TÌM KIẾM VÀ SẮP XẾP

NỘI DUNG CHƯƠNG II

I. NHU CẦU TÌM KIẾM, SẮP XẾP
II. CÁC GIẢI THUẬT TÌM KIẾM
III. CÁC GIẢI THUẬT SẮP XẾP
IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

Từ khóa: Heap Sort

Phân tích: Trong phương pháp chọn trực tiếp (Selection Sort), mỗi lần chọn phần tử cực tiểu theo quan hệ $\mathfrak{R}$ đều không tận dụng được các kết quả so sánh trước đó → độ phức tạp theo phép so sánh là $O(n^2)$.

→ Tận dụng kết quả so sánh bằng cấu trúc **Heap**

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

Ý tưởng:

- Xây dựng cấu trúc Heap:
  - Là một cây nhị phân hoàn chỉnh
  - Nếu giá trị khóa của nút cha và hai nút con lần lượt là K, K_1, K_2, thì:

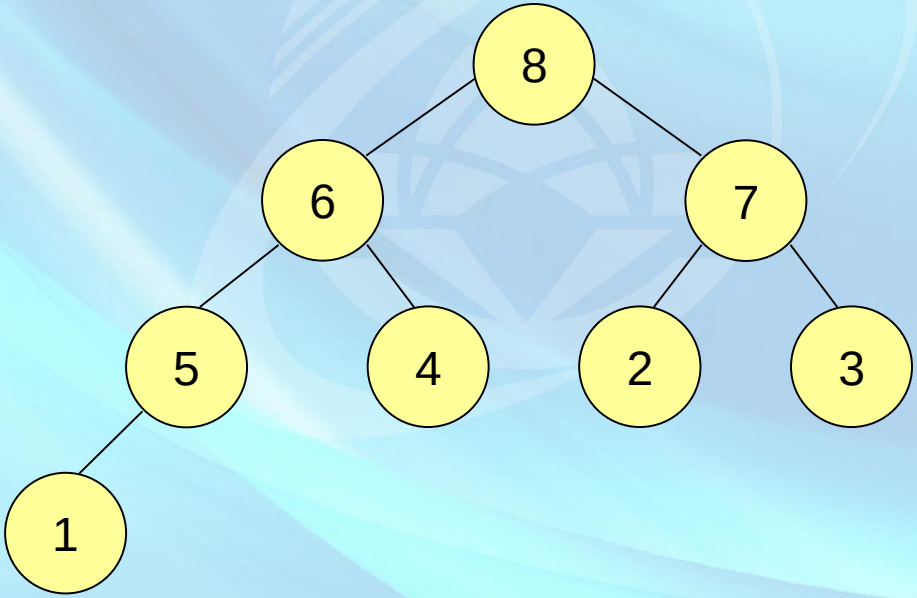
$$\begin{cases} K_1 \ \Re \ K \\ K_2 \ \Re \ K \end{cases}$$

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

Ý tưởng:

Ví dụ quan hệ $\mathfrak{R}$ là **<**



III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

Ý tưởng:

- Xây dựng cấu trúc Heap:
  - Nếu dùng mảng A để biểu diễn cấu trúc Heap, A có đặc điểm:
    - ✓ A_0 là cực đại theo $\mathfrak{R}$
    - ✓ A_{i*2+1} $\mathfrak{R}$ A_i và $A_{(i+1)*2}$ $\mathfrak{R}$ A_i
    - ✓ A_{i*2+1} và $A_{(i+1)*2}$ là phần tử liên đới với A_i

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

Ý tưởng:

- Sắp xếp dựa trên cấu trúc Heap:
  - Hoán đổi vị trí của A_0 và A_{n-1}, đưa giá trị cực đại về cuối dãy → dãy **A** trong bước tiếp theo đã giảm được một phần tử, còn lại là $A_0..A_{n-2}$
  - Bắt đầu từ A_0, điều chỉnh các phần tử trong $A_0..A_{n-2}$ để đảm bảo tính chất của Heap.
  - Thực hiện hoán đổi A_0 và điều chỉnh dãy mới đến khi dãy A chỉ còn lại một phần tử

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

Thuật toán:

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

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

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

Thuật toán: (Sắp xếp)

```
buildHeap(A, n)
while n > 1
  n ← n-1
  swap(A[0], A[n])
  heapify(A, 0, n)
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

Thuật toán: (tạo Heap)

```
//chương trình con buildHeap(A, n)
i ← (n-1)/2
while i ≥ 0
  heapify(A, i, n)
  i ← i-1
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

Thuật toán: (điều chỉnh Heap)

```
// chương trình con heapify(A, k, n)
j ← 2*k+1
while j < n
  if j+1 < n then
     if a[j] ℜ a[j+1] then j ← j+1 end if
  if (a[j] ℜ a[k]) then return end if
  swap(a[k], a[j])
  k = j
  j = 2*k+1
end while
```

III. CÁC GIẢI THUẬT SẮP XẾP

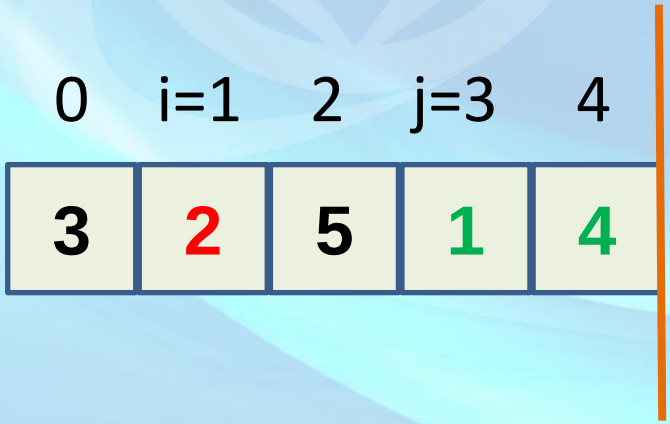
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Tạo Heap: i=1

heapify: j = 3, j+1 = 4



III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Tạo Heap: i=1

heapify: j = 3, j+1 = 4, hoán đổi A[1] và A[4]

III. CÁC GIẢI THUẬT SẮP XẾP

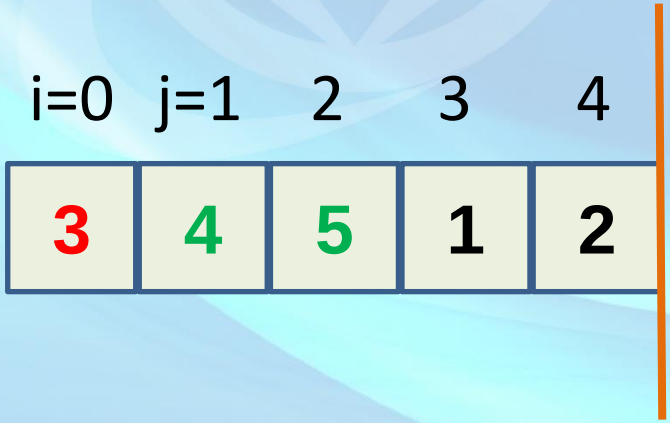
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Tạo Heap: i=0

heapify: j = 1, j+1 = 2



III. CÁC GIẢI THUẬT SẮP XẾP

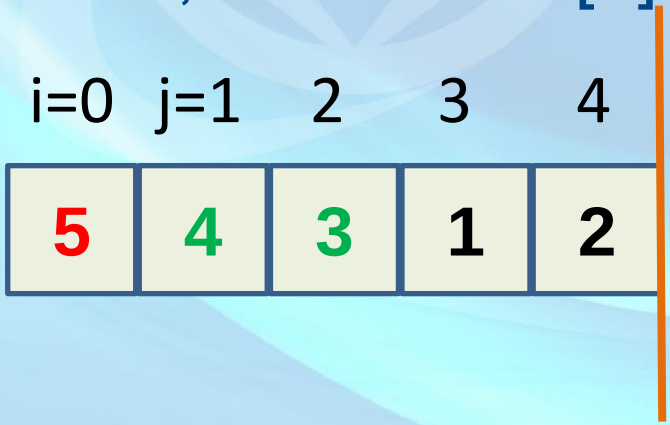
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Tạo Heap: i=0

heapify: j = 1, j+1 = 2, hoán đổi A[0] và A[2]



III. CÁC GIẢI THUẬT SẮP XẾP

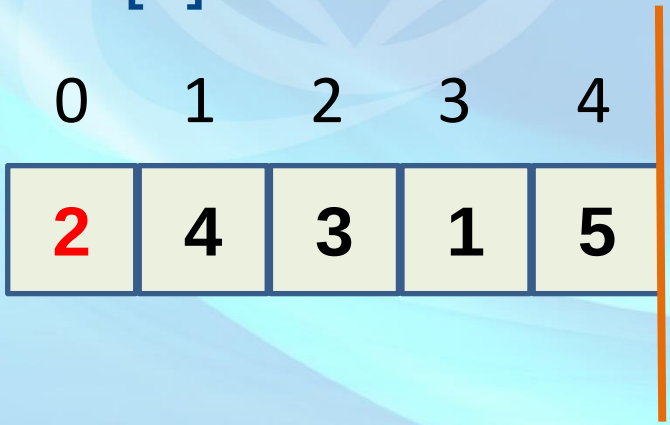
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=5

Hoán đổi: A[0] và A[4]



III. CÁC GIẢI THUẬT SẮP XẾP

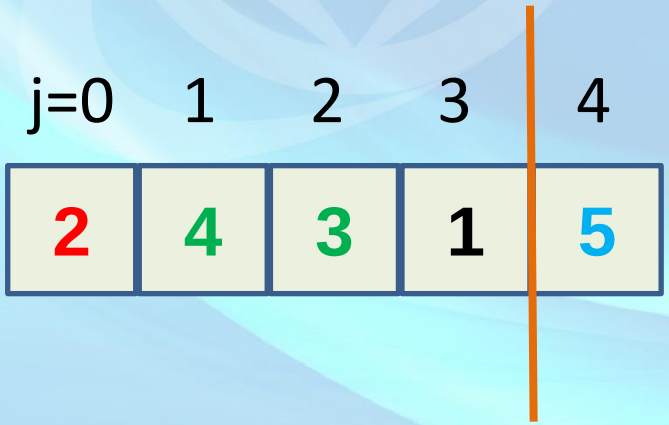
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=4

heapify: j=0



III. CÁC GIẢI THUẬT SẮP XẾP

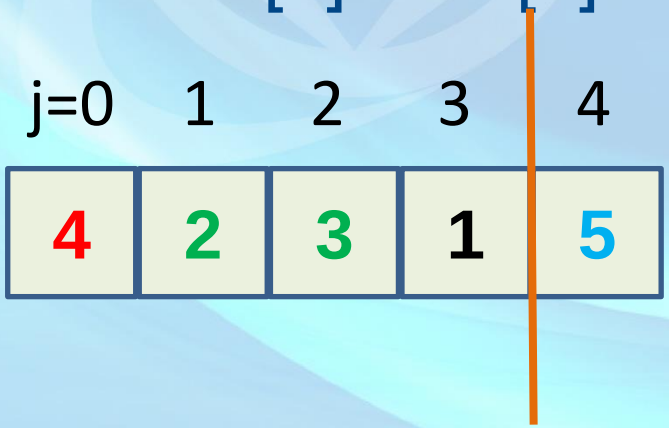
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=4

heapify: j=0, hoán đổi A[0] và A[1]



III. CÁC GIẢI THUẬT SẮP XẾP

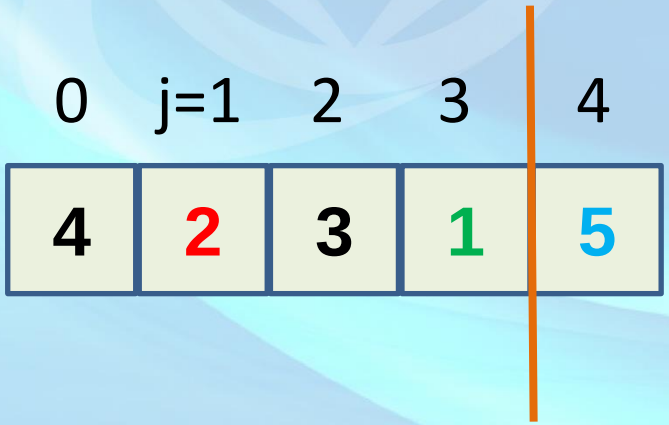
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=4

heapify: j=1,



III. CÁC GIẢI THUẬT SẮP XẾP

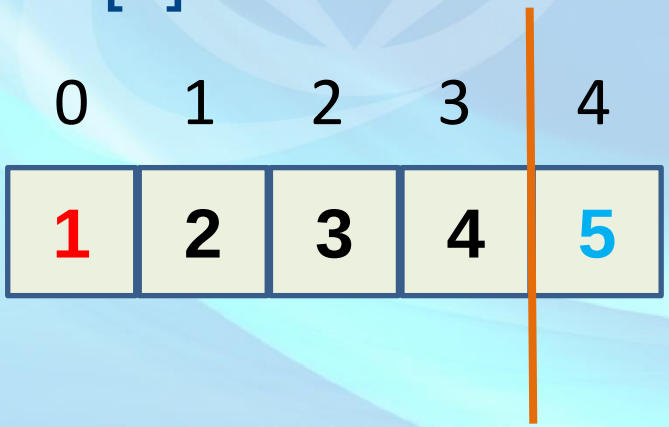
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=4

Hoán đổi A[0] và A[3]



III. CÁC GIẢI THUẬT SẮP XẾP

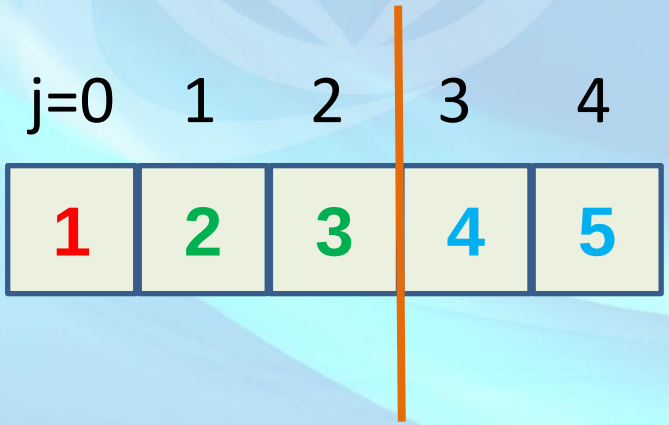
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=3

heapify j=0



III. CÁC GIẢI THUẬT SẮP XẾP

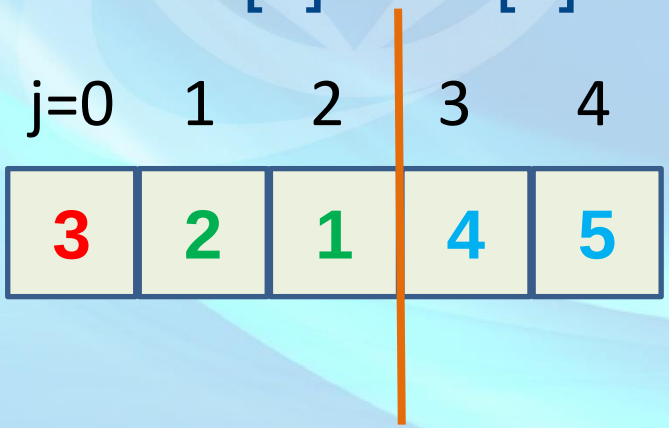
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=3

heapify j=0, hoán đổi A[0] và A[2]



III. CÁC GIẢI THUẬT SẮP XẾP

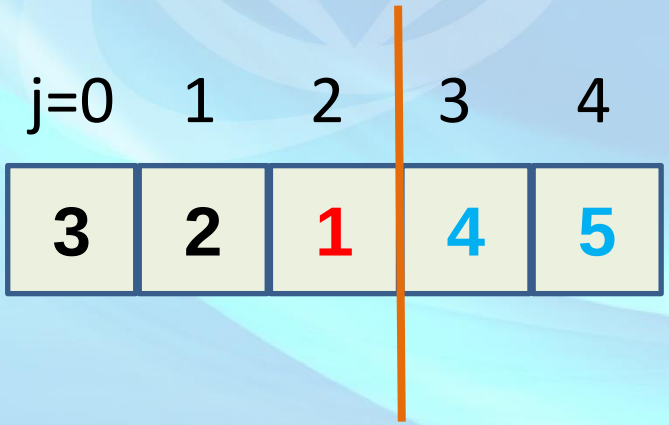
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=3

heapify j=2,



III. CÁC GIẢI THUẬT SẮP XẾP

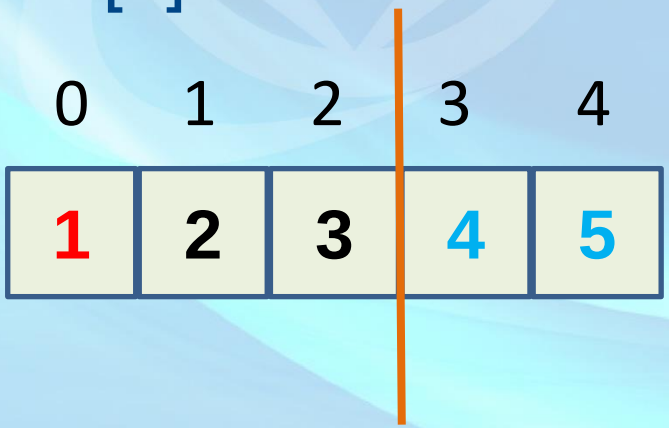
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=3

Hoán đổi A[0] và A[2]



III. CÁC GIẢI THUẬT SẮP XẾP

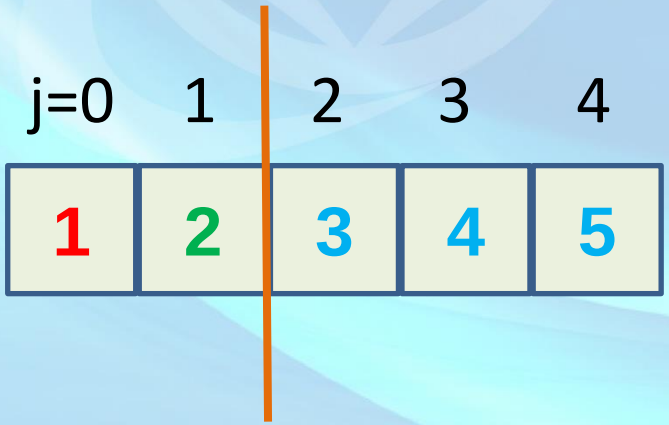
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=2

heapify: j=0



III. CÁC GIẢI THUẬT SẮP XẾP

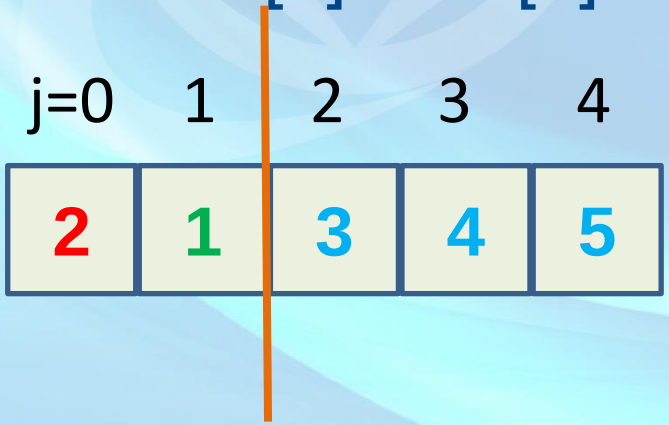
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=2

heapify: j=0, hoán đổi A[0] và A[1]



III. CÁC GIẢI THUẬT SẮP XẾP

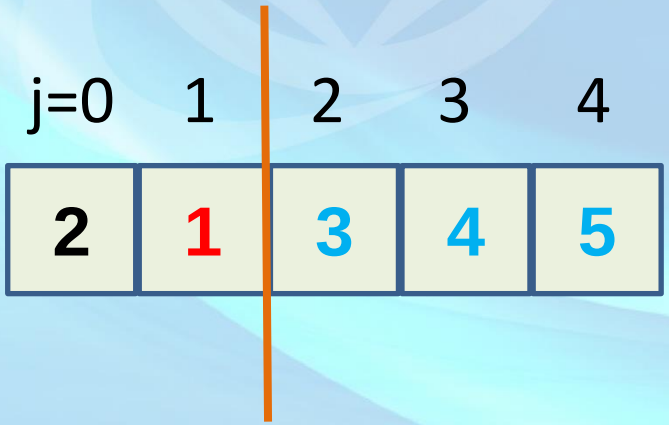
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=2

heapify: j=1,



III. CÁC GIẢI THUẬT SẮP XẾP

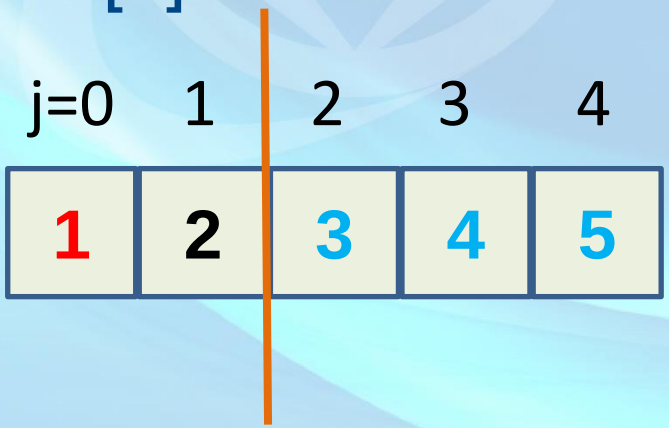
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=2

Hoán đổi A[0] và A[1]



III. CÁC GIẢI THUẬT SẮP XẾP

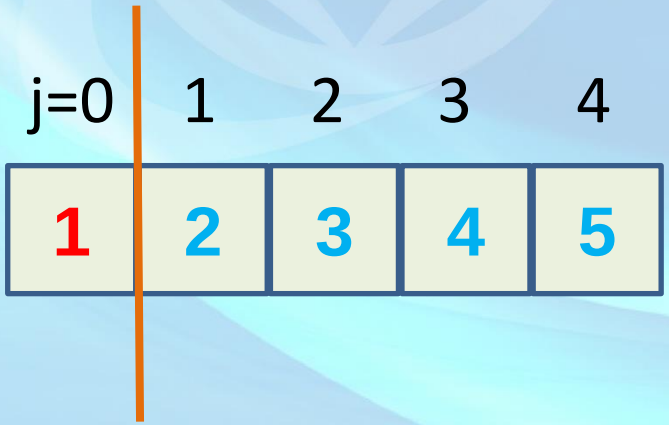
❖PHƯƠNG PHÁP VUN ĐỐNG

Quá trình tính toán:

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

Sắp xếp: n=1

Dừng



III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

Cài đặt: Trên mảng, Giả sử thứ thự là **<** (tăng dần)

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

```
void heapify(int *a, int k, int n) {
  int 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;
        swap(a[k], a[j]);
        k = j; j = 2*k + 1;
  }
}
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

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

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

```
void heapSort(int *a, int n) {
  buildHeap(a, n);
  while (n > 0) {
        n = n - 1;
        swap(a[0], a[n]);
        heapify(a, 0, n);
  }
}
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP VUN ĐỐNG

Đánh giá: phương pháp vun đống có thời gian sắp xếp ổn định.

	TỐT NHẤT (thứ tự ngược)	**TRUNG BÌNH** (chưa có thứ tự)	**XẤU NHẤT** (đúng thứ tự)
Theo phép so sánh	O(nlogn)	O(nlogn)	O(nlogn)
Theo phép gán giá trị khóa	O(nlogn)	O(nlogn)	O(nlogn)

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP PHÂN HOẠCH

Từ khóa: Quick Sort

Phân tích: Giả sửa dãy **A** đã có thứ tự $\Re$, với A_i bất kỳ:

- $A_j \Re A_i \ \forall\ j < i$
- $A_i \Re A_j \ \forall\ j > I$
- $A_0, .., A_{i-1}$ và $A_{i+1}, .., A_{n-1}$ đều có thứ tự $\Re$.

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP PHÂN HOẠCH

Ý tưởng: Áp dụng chiến lược chia để trị:

- Nếu A có không quá 1 phần tử → đã có thứ tự.
- Chọn phần tử chốt (pivot) **x**
- Chia dãy **A** thành hai phần:
  - Phần trước chứa A_i sao cho $A_i \Re x$
  - Phần sau chứa A_j sao cho $x \Re A_j$
- Sắp xếp hai dãy $A_0,..,A_{k-1}$ và $A_{k+1},..,A_{n-1}$ tương tự

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP PHÂN HOẠCH

Thuật toán:

quickSort(A)

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

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

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP PHÂN HOẠCH

Thuật toán:

```
if |A| < 2 then return end if
x ← A[0], A1 ← {}, A2 ← {}
for i ← 0 to n-1 do
  if A[i] ℜ x then A1←A1 ∪ {A[i]}
  else A2←A2 ∪ {A[i]} end if
end for
quickSort(A1), quickSort(A2)
A←A1 ∪ {x} ∪ A2
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP PHÂN HOẠCH

Cài đặt: Trên mảng, Giả sử thứ thự là **<** (tăng dần).

Khi cài đặt trên mảng theo đúng thuật toán:

- Phải sử dụng thêm hai mảng phụ A_1 và A_2

→ Lãng phí bộ nhớ

Cách giải quyết:

→ Xác định đoạn cần sắp xếp bằng chỉ số b, e: $A_b,..,A_e$

→ Giảm thao tác nối lại các đoạn đã sắp xếp

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP PHÂN HOẠCH

```
void quickSort(int *a, int b, int e) {
  if (b >= e) return;
  int x = a[0], i = b, j = e;
  while(i < j) {
     while (a[i] < x) i++;
     while (a[j] > x) j--;
     if (i < j) {
        swap(a[i], a[j]); i++; j--; }
  }
  quickSort(a, b, j); quickSort(a, i, e);
}
```

III. CÁC GIẢI THUẬT SẮP XẾP

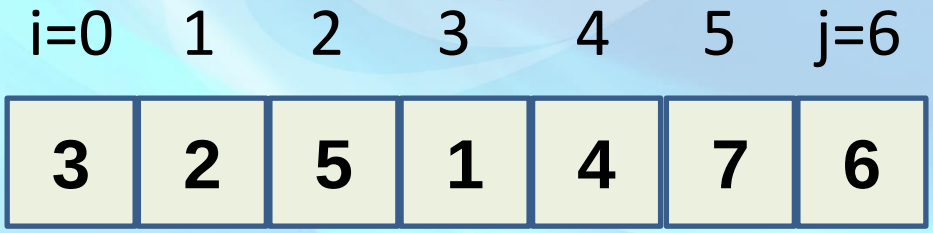
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 0, e = 6, x = 3

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

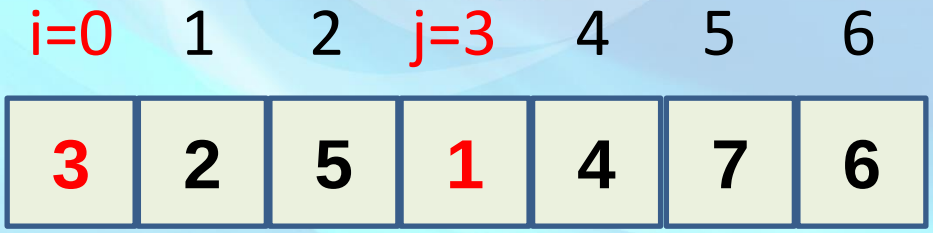
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 0, e = 6, x = 3

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

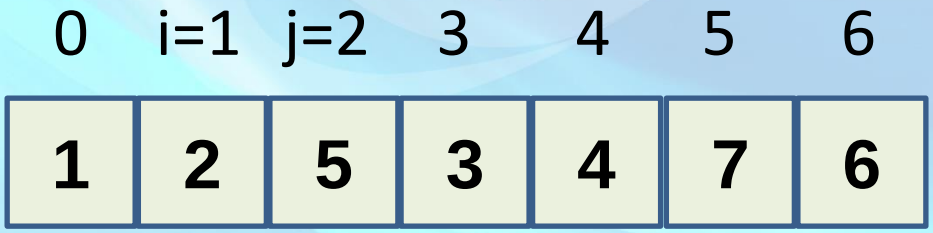
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 0, e = 6, x = 3

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

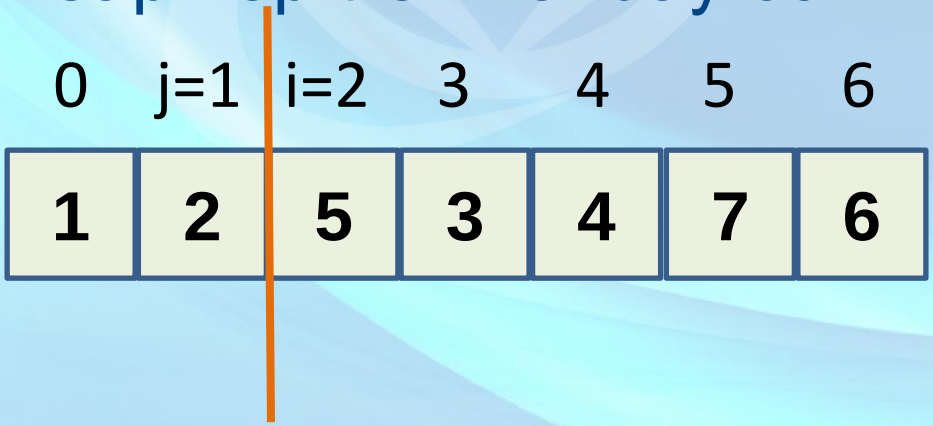
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 0, e = 6, x = 3

→ thực hiện sắp xếp trên hai dãy con



III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 0, e = 1, x = 1

→ phân chia dãy

III. CÁC GIẢI THUẬT SẮP XẾP

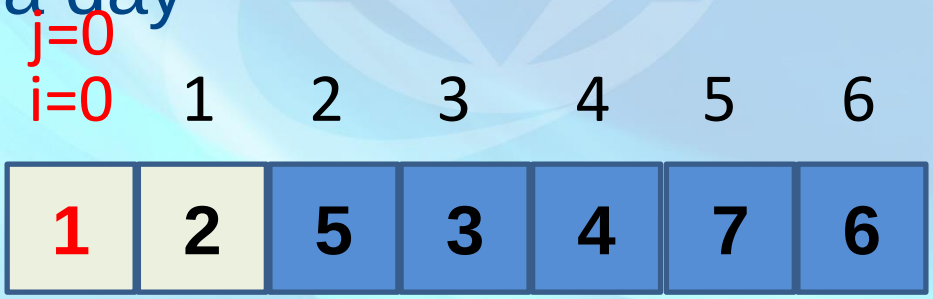
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 0, e = 1, x = 1

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

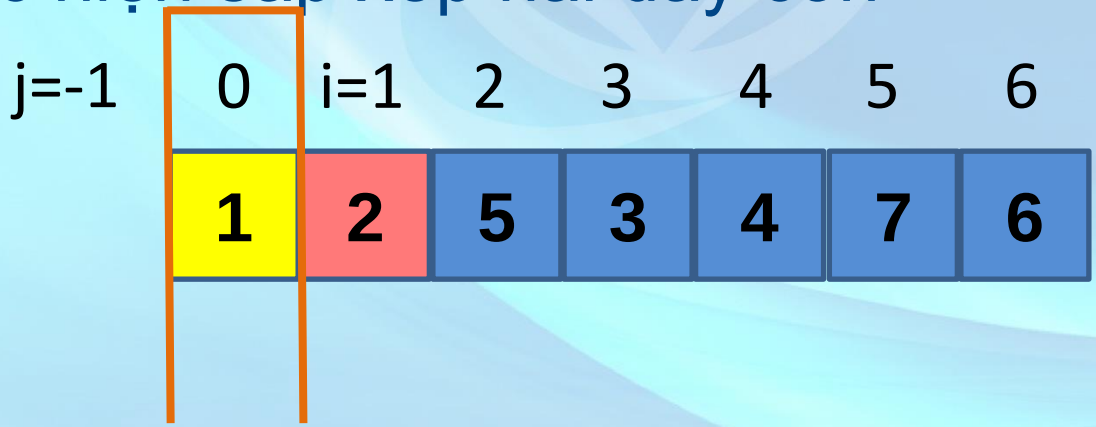
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 0, e = 1, x = 1

→ thực hiện sắp xếp hai dãy con



III. CÁC GIẢI THUẬT SẮP XẾP

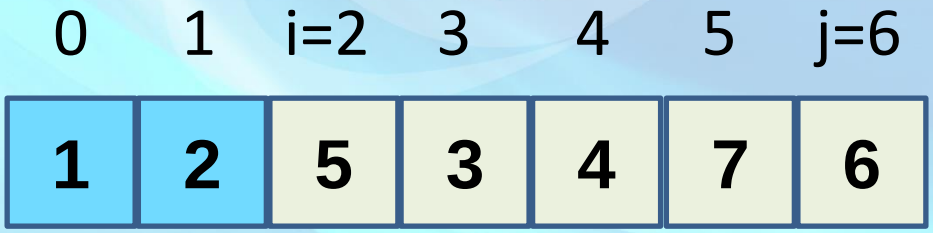
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 2, e = 6, x = 5

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 2, e = 6, x = 5

→ phân chia dãy

III. CÁC GIẢI THUẬT SẮP XẾP

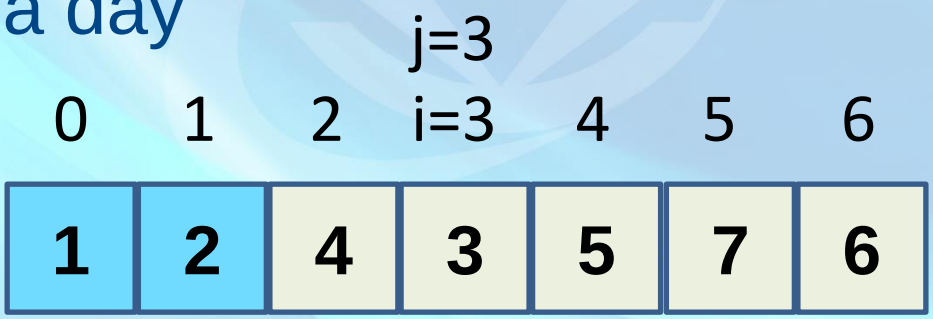
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 2, e = 6, x = 5

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

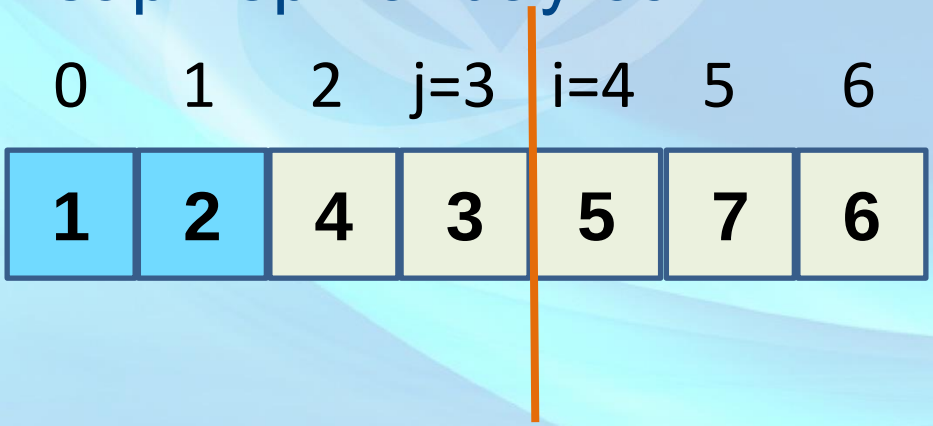
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 2, e = 6, x = 5

→ thực hiện sắp xếp hai dãy con



III. CÁC GIẢI THUẬT SẮP XẾP

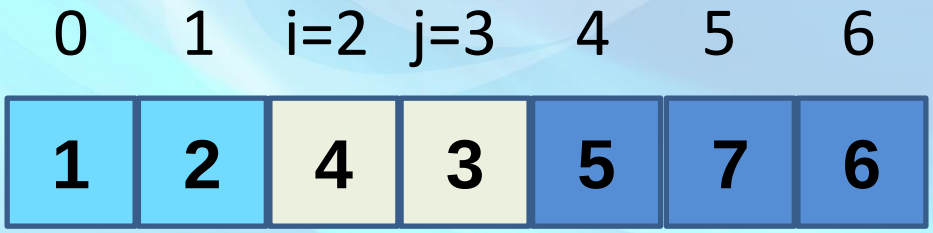
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 2, e = 3, x = 4

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

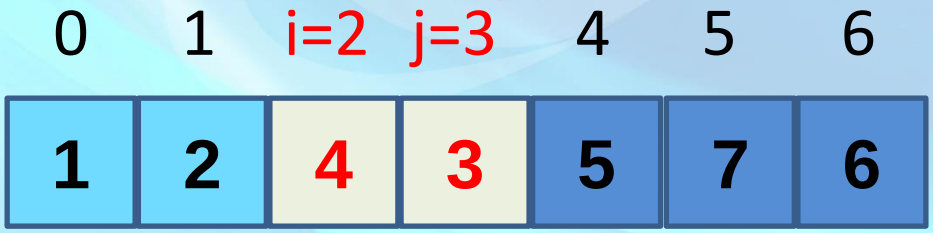
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 2, e = 3, x = 4

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

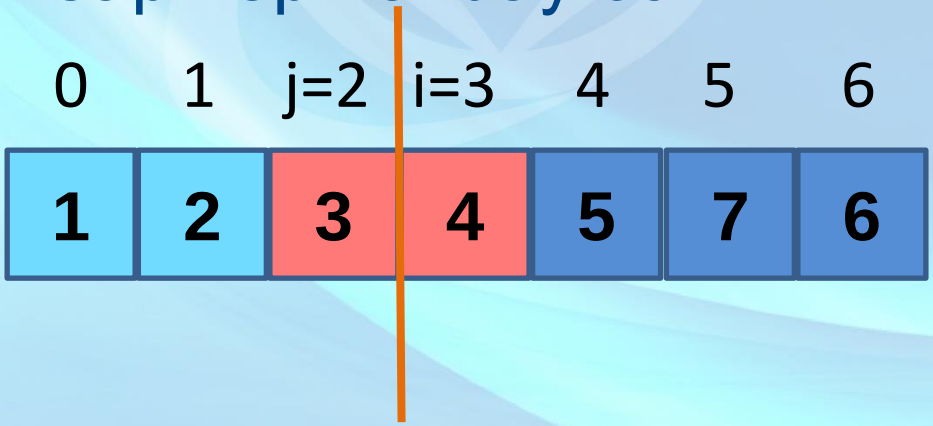
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 2, e = 3, x = 4

→ thực hiện sắp xếp hai dãy con



III. CÁC GIẢI THUẬT SẮP XẾP

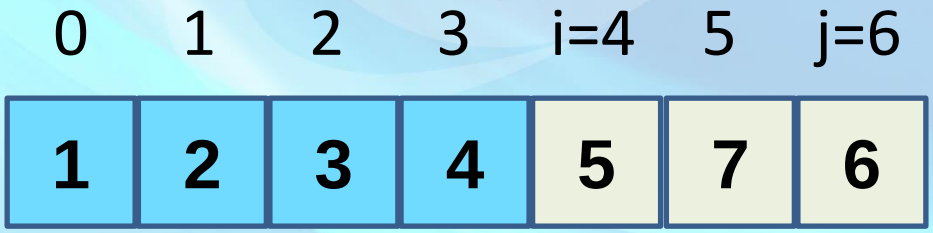
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 4, e = 6, x = 5

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

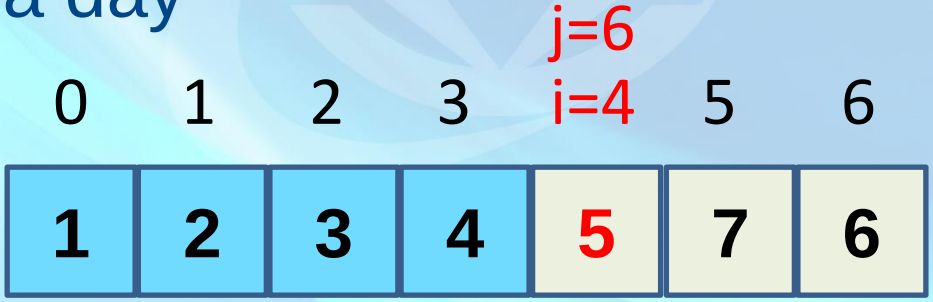
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 4, e = 6, x = 5

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 4, e = 6, x = 5

→ thực hiện sắp xếp hai dãy con

III. CÁC GIẢI THUẬT SẮP XẾP

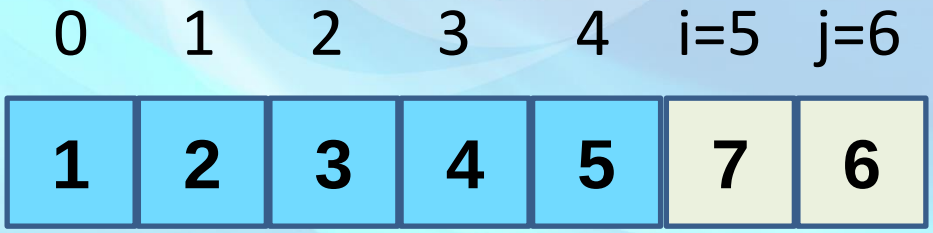
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 5, e = 6, x = 7

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

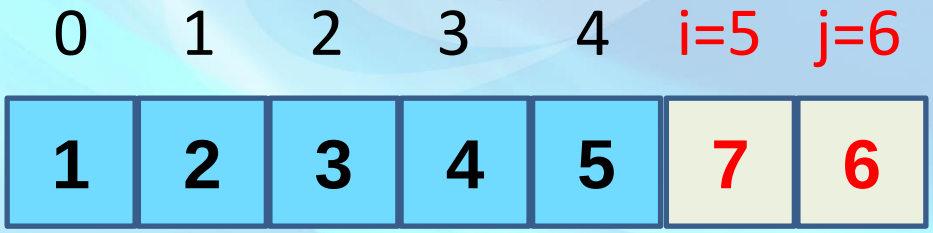
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 5, e = 6, x = 7

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

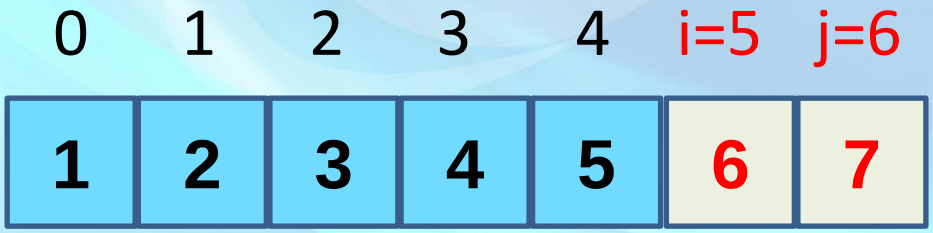
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 5, e = 6, x = 7

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

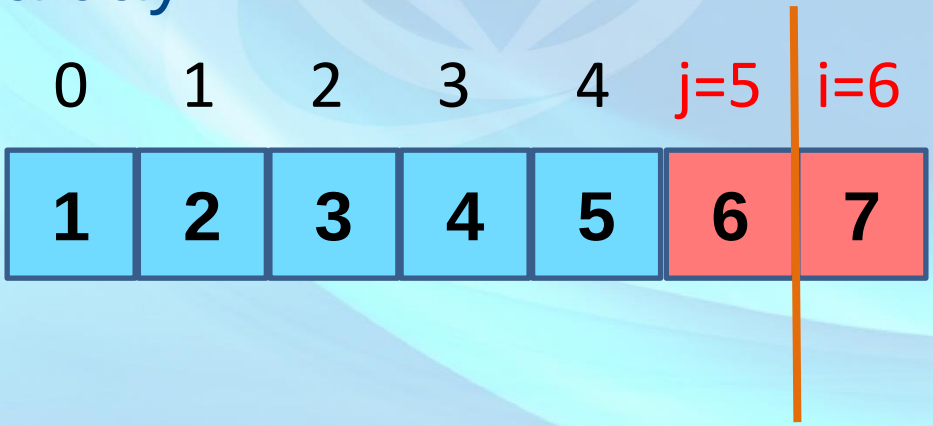
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 5, e = 6, x = 7

→ phân chia dãy



III. CÁC GIẢI THUẬT SẮP XẾP

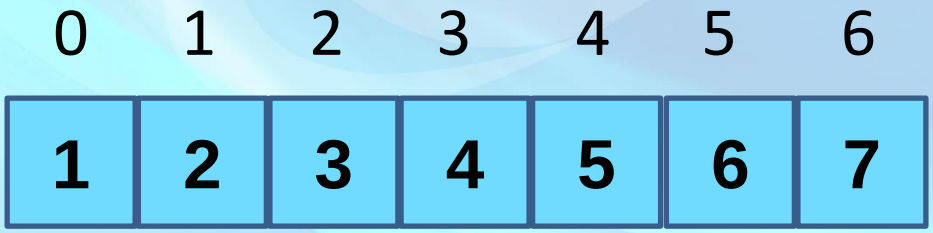
❖PHƯƠNG PHÁP PHÂN HOẠCH

Quá trình tính toán:

Giả sử A={3,2,5,1,4,7,6} và thứ tự cấp sắp xếp < (tăng dần)

b = 0, e = 6,

→ kết quả



III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP PHÂN HOẠCH

Đánh giá: phương pháp phân hoạch có thể bị suy biến với chi phí thời gian là $O(n^2)$.

	TỐT NHẤT (x là phần tử trung vị)	**TRUNG BÌNH** (x không phải cực trị)	**XẤU NHẤT** (x là cực trị)
Theo phép so sánh	O(nlogn)	O(nlogn)	$O(n^2)$
Theo phép gán giá trị khóa	O(nlogn)	O(nlogn)	$O(n^2)$

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP PHÂN HOẠCH

Đánh giá:

- Việc chọn giá trị chốt x rất quan trọng
- Rất khó để chọn x là điểm trung vị → chọn x trong 3 hoặc 5 giá trị được chọn tùy ý trong danh sách A.

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN

Từ khóa: Merge Sort

Phân tích: Giả sử dãy A_1 và A_2 có **k** phần tử đã có thứ tự $\mathfrak{R}$, khi đó, có thể tạo dãy **A** có thứ tự $\mathfrak{R}$ gồm các phần tử của A_1 và A_2 với:

- Chi phí thời gian là **O(k)**
- Trộn A_1 và A_2 theo thứ tự từ đầu danh sách.
- Phần tử trên hai danh sách được trộn theo thứ tự $\mathfrak{R}$

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN

Ý tưởng: Áp dụng chiến lược chia để trị:

- Danh sách có 1 phần tử luôn có thứ thự.
- Để sắp xếp danh sách **A**:
  - Chia **A** thành hai danh sách A_1 và A_2
  - Sắp xếp A_1 và A_2 theo thứ tự $\mathfrak{R}$
  - Trộn A_1 và A_2 theo thứ tự $\mathfrak{R}$

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN

Thuật toán: Trộn trực tiếp mergeSort(A)

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

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

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN

Thuật toán: Trộn trực tiếp (mergeSort)

- B1: k←1
- B2: Tách A 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 theo thứ tự ℜ
- B4: k←2*k,.
- B5: Nếu k < n thì qua B2; ngược lại thì kết thúc.

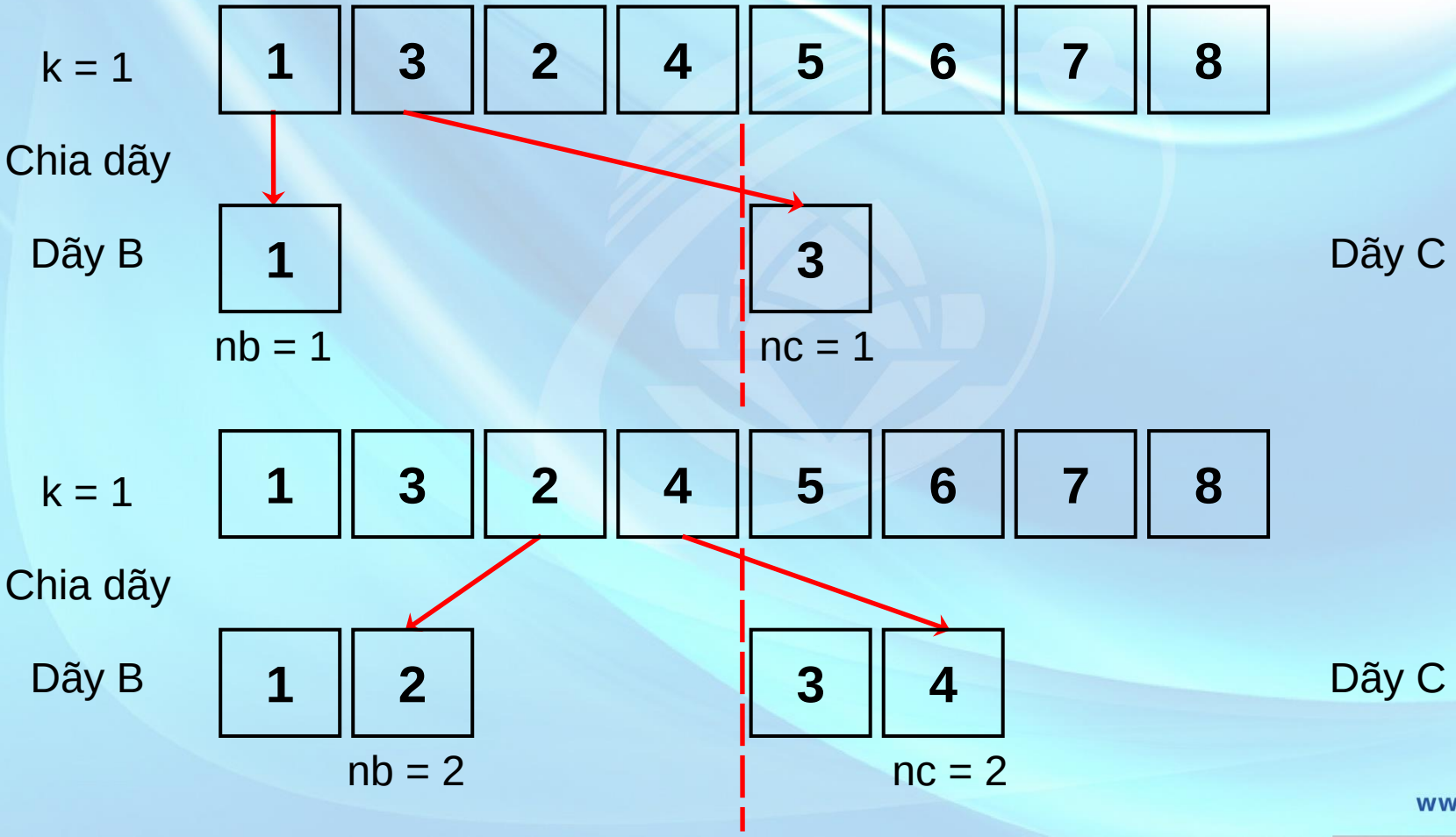
III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN

Ví dụ minh họa: Sắp xếp dãy số A = {1, 3, 2, 4, 5, 6, 7, 8}

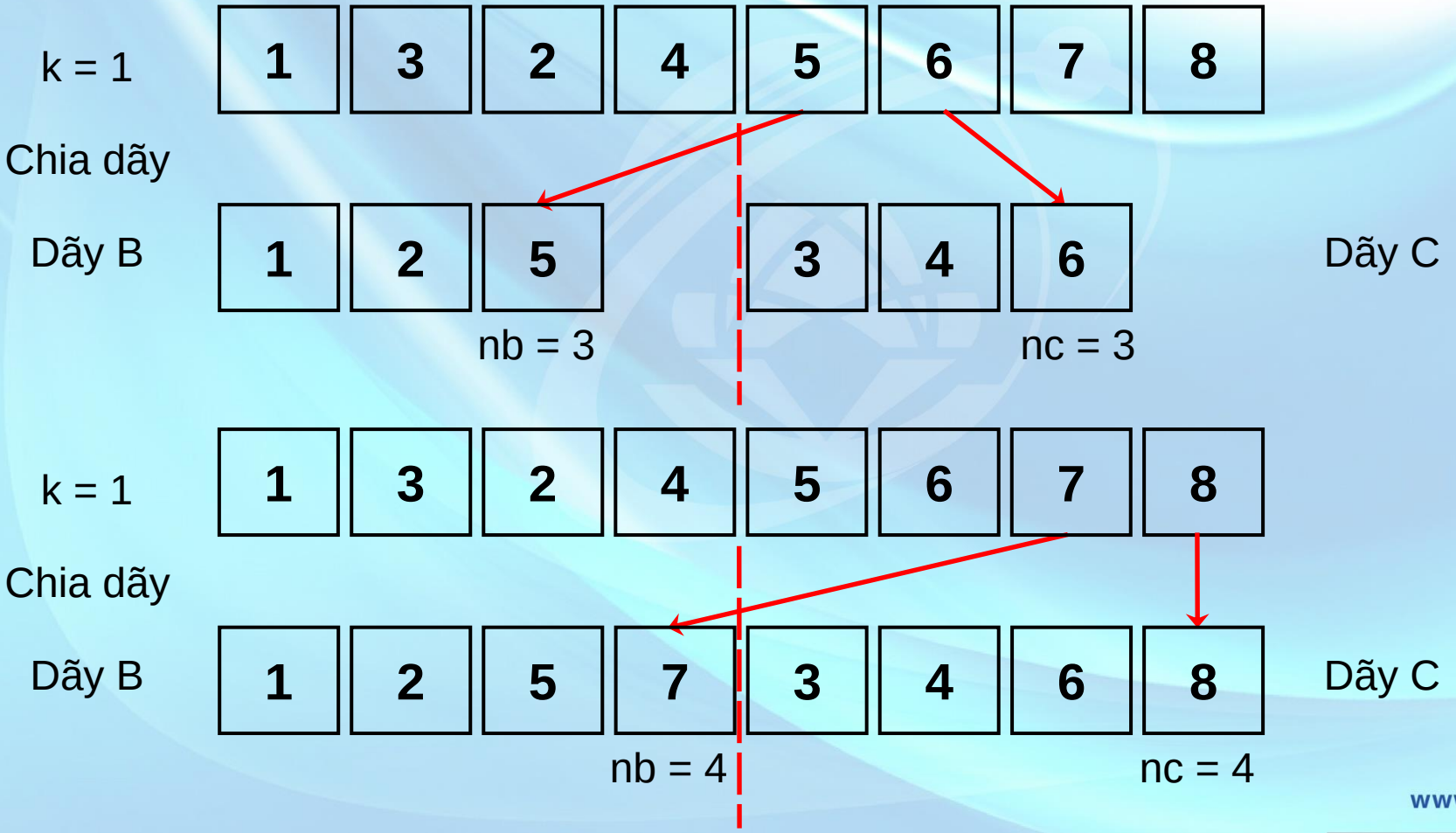
III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN



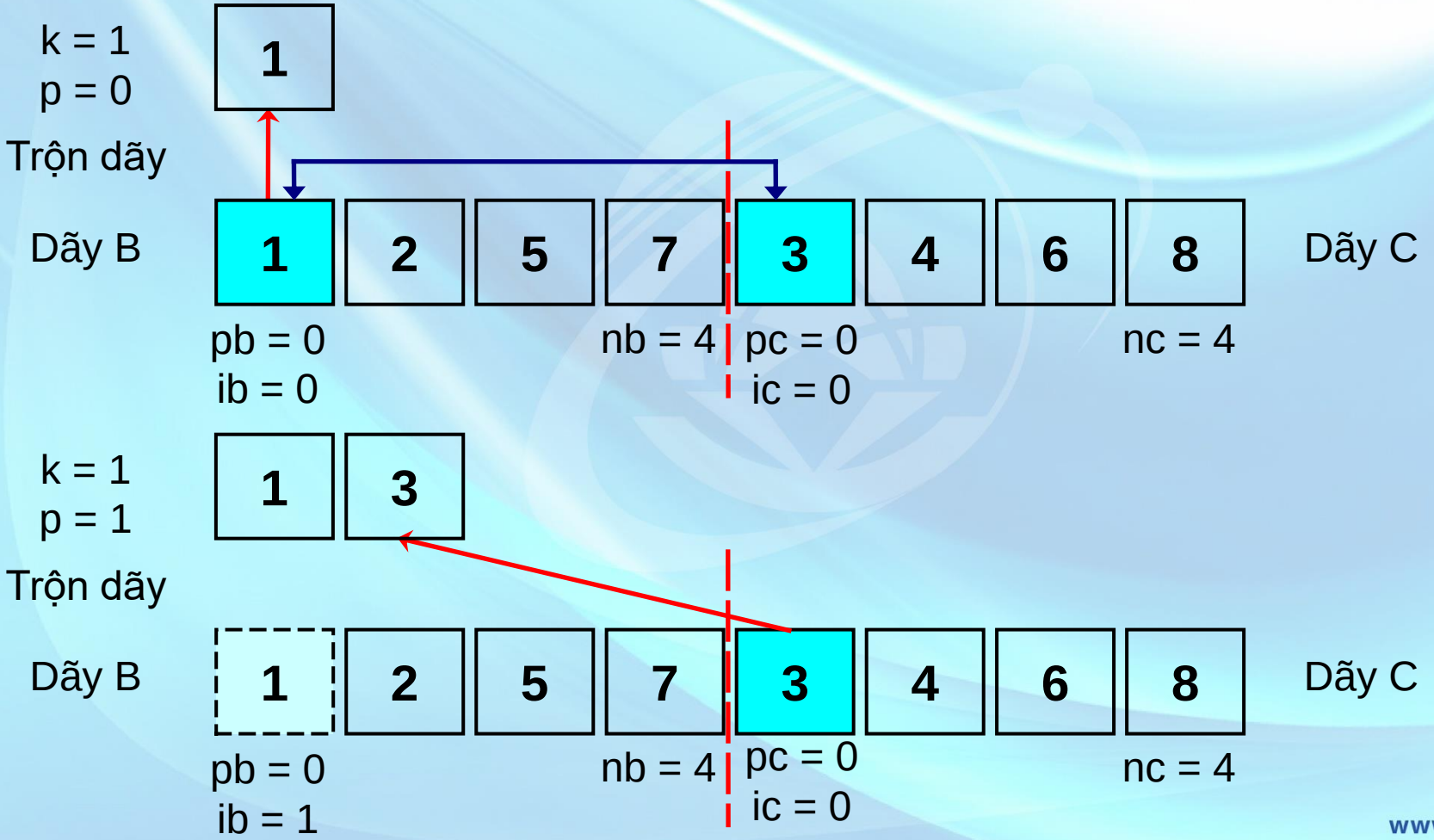
III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN



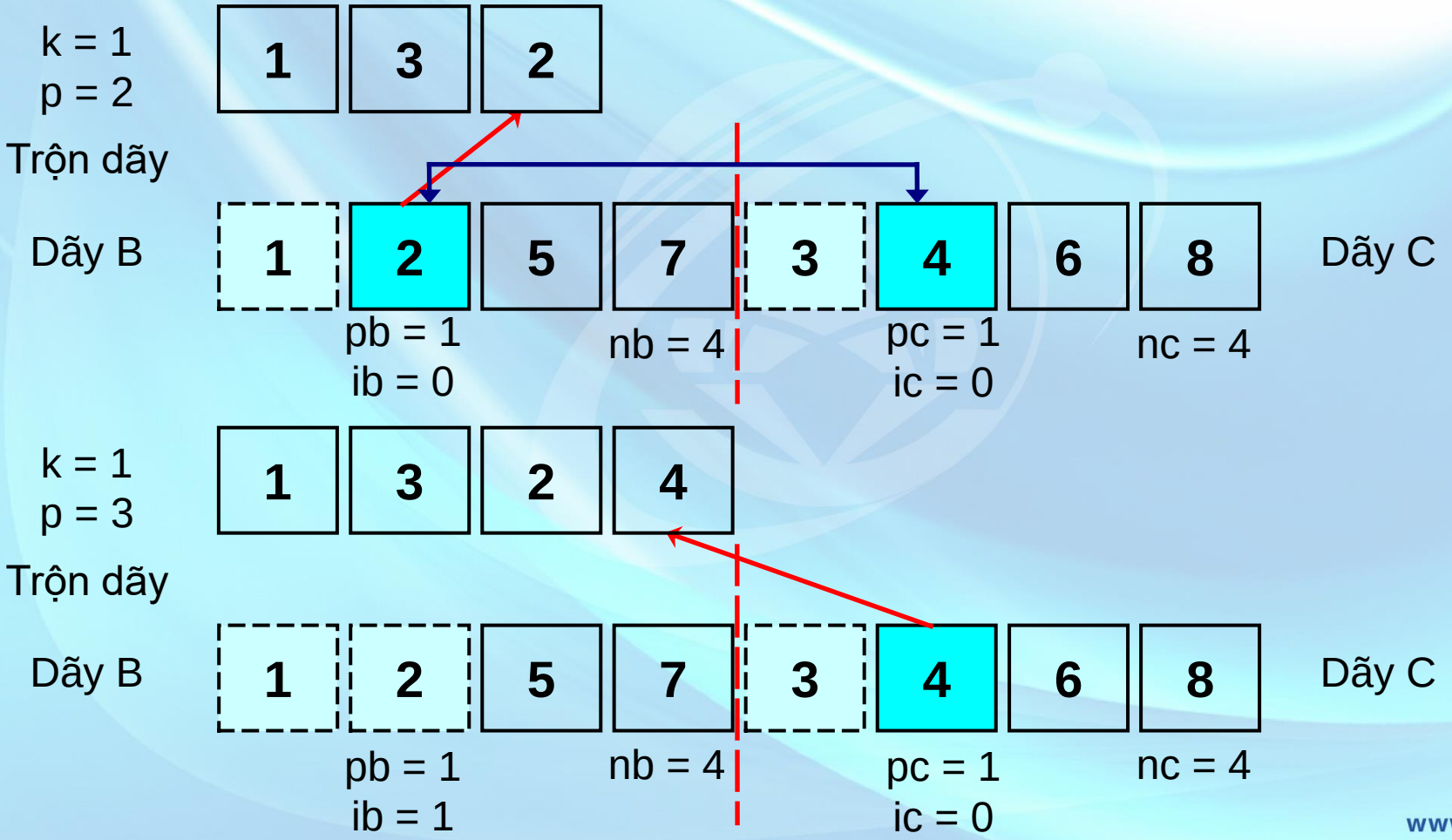
III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN



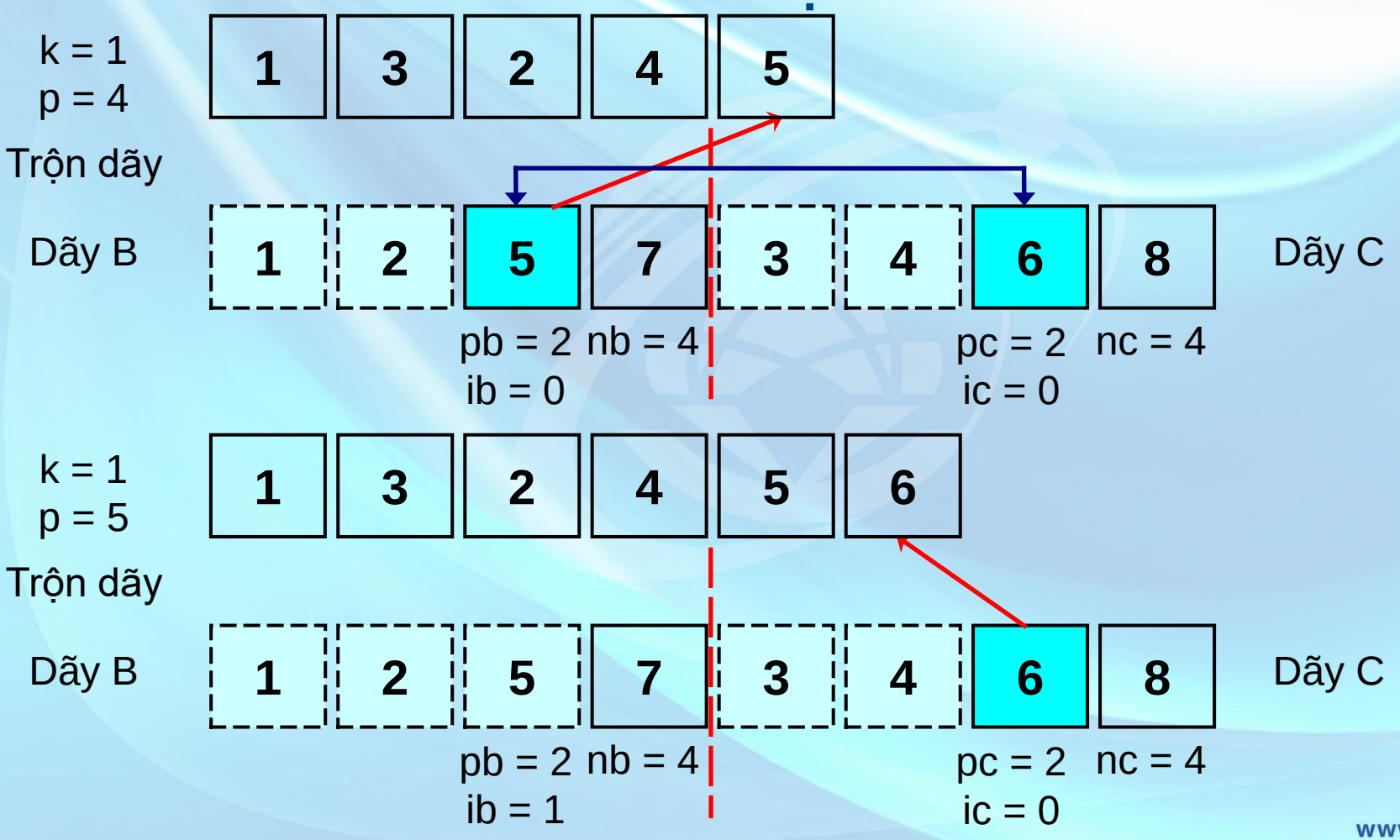
III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN



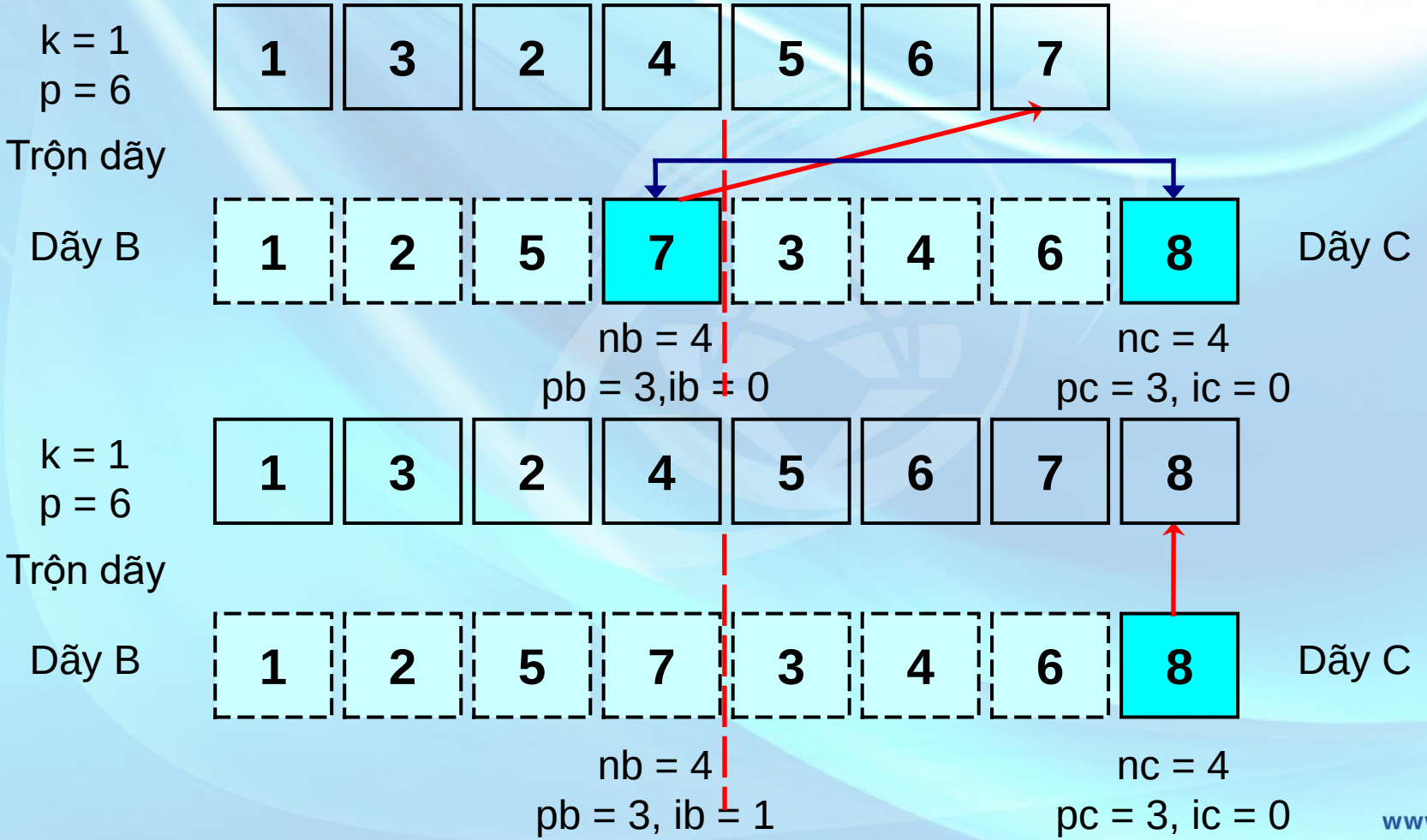
III. CÁC GIẢI THUẬT SẮP XẾP

PHƯƠNG PHÁP TRỘN



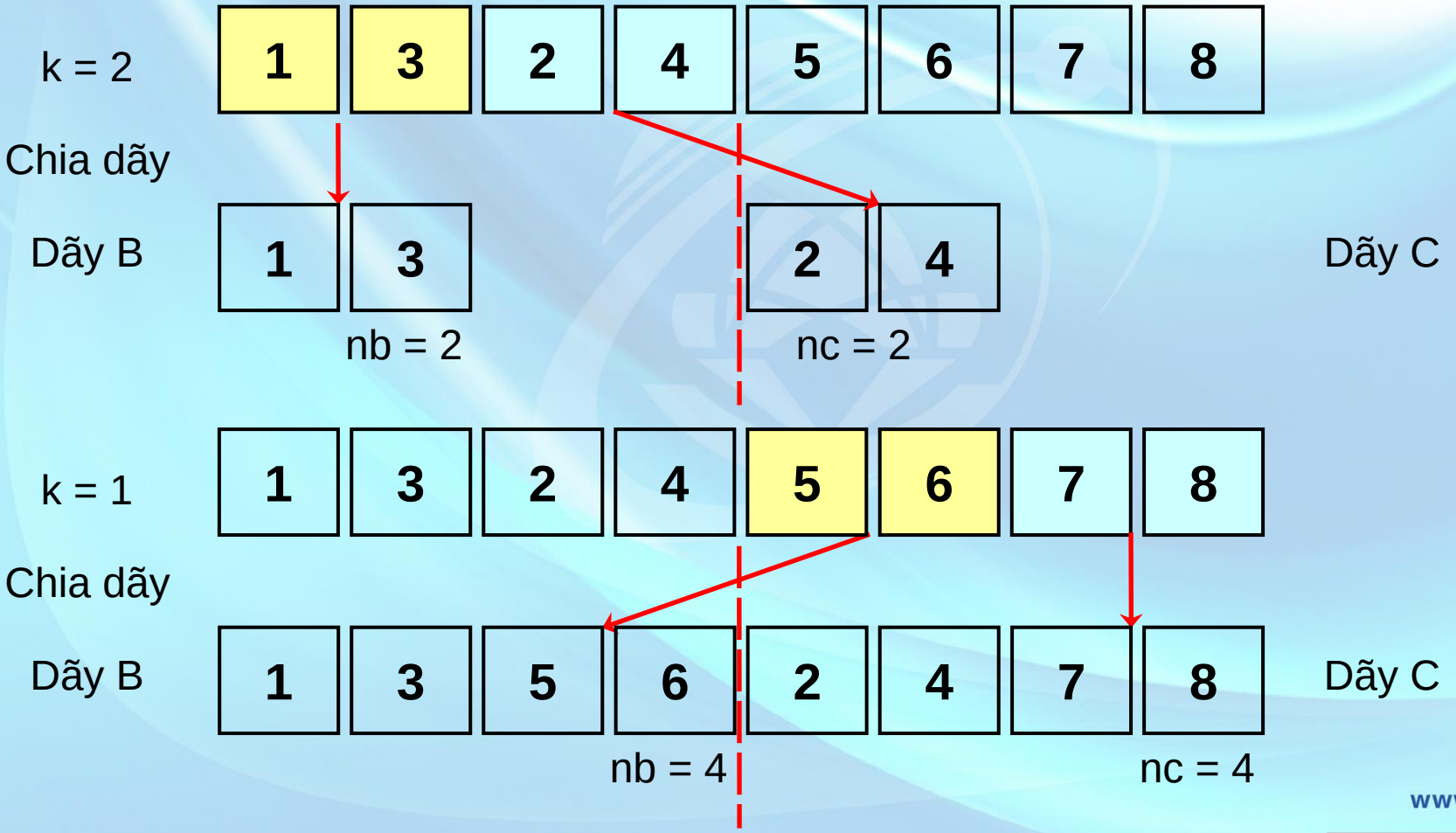
III. CÁC GIẢI THUẬT SẮP XẾP

PHƯƠNG PHÁP TRỘN



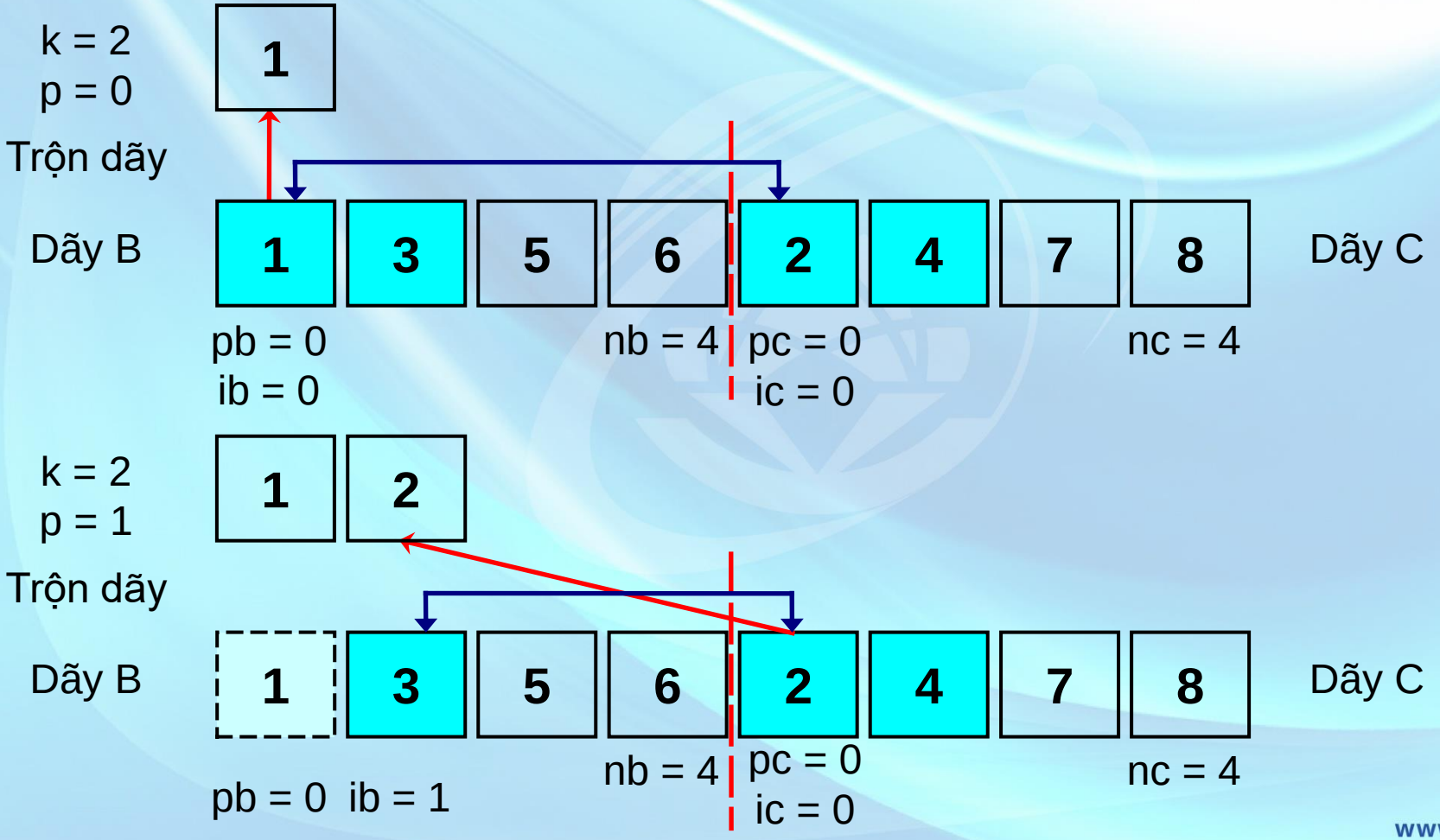
III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN



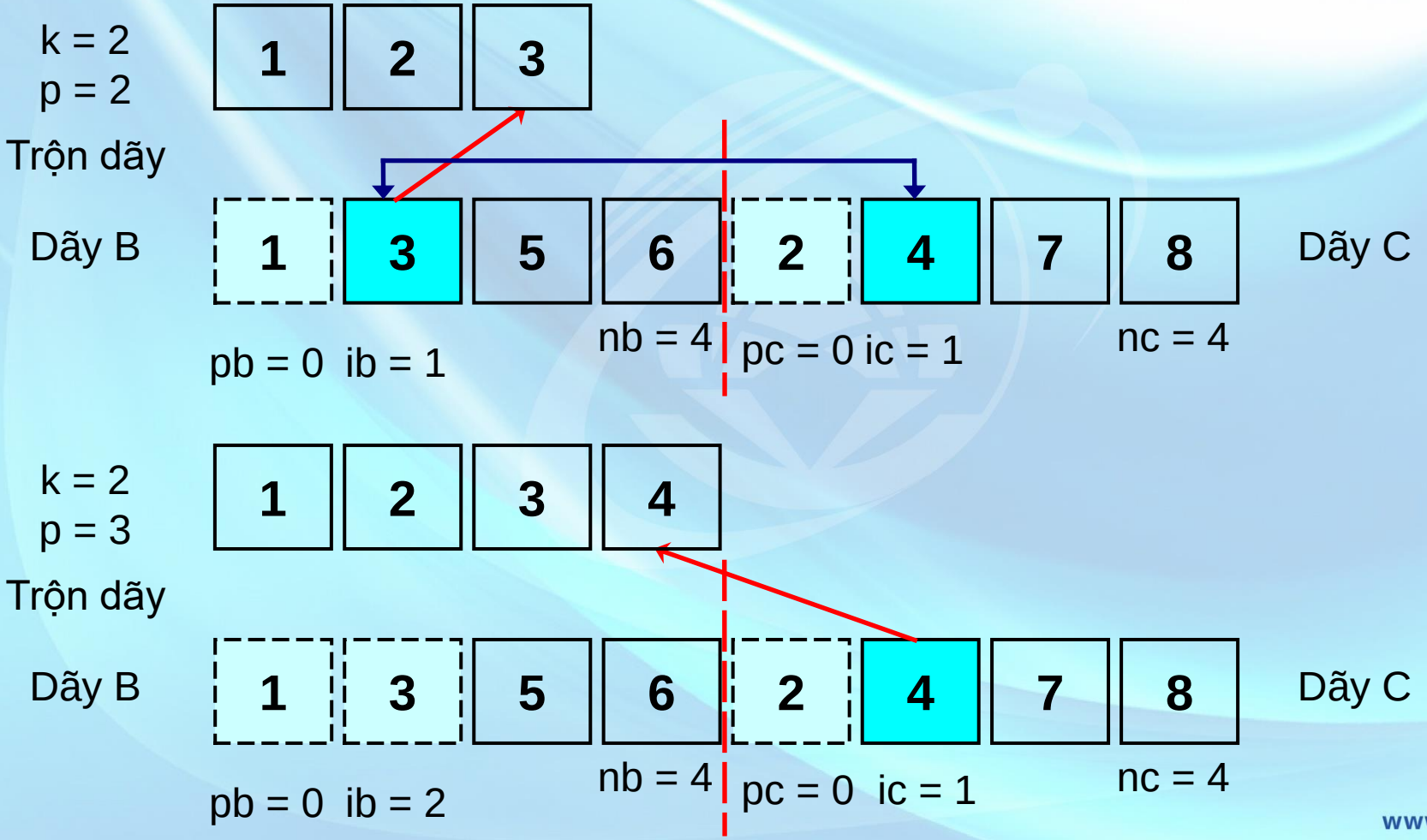
III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN



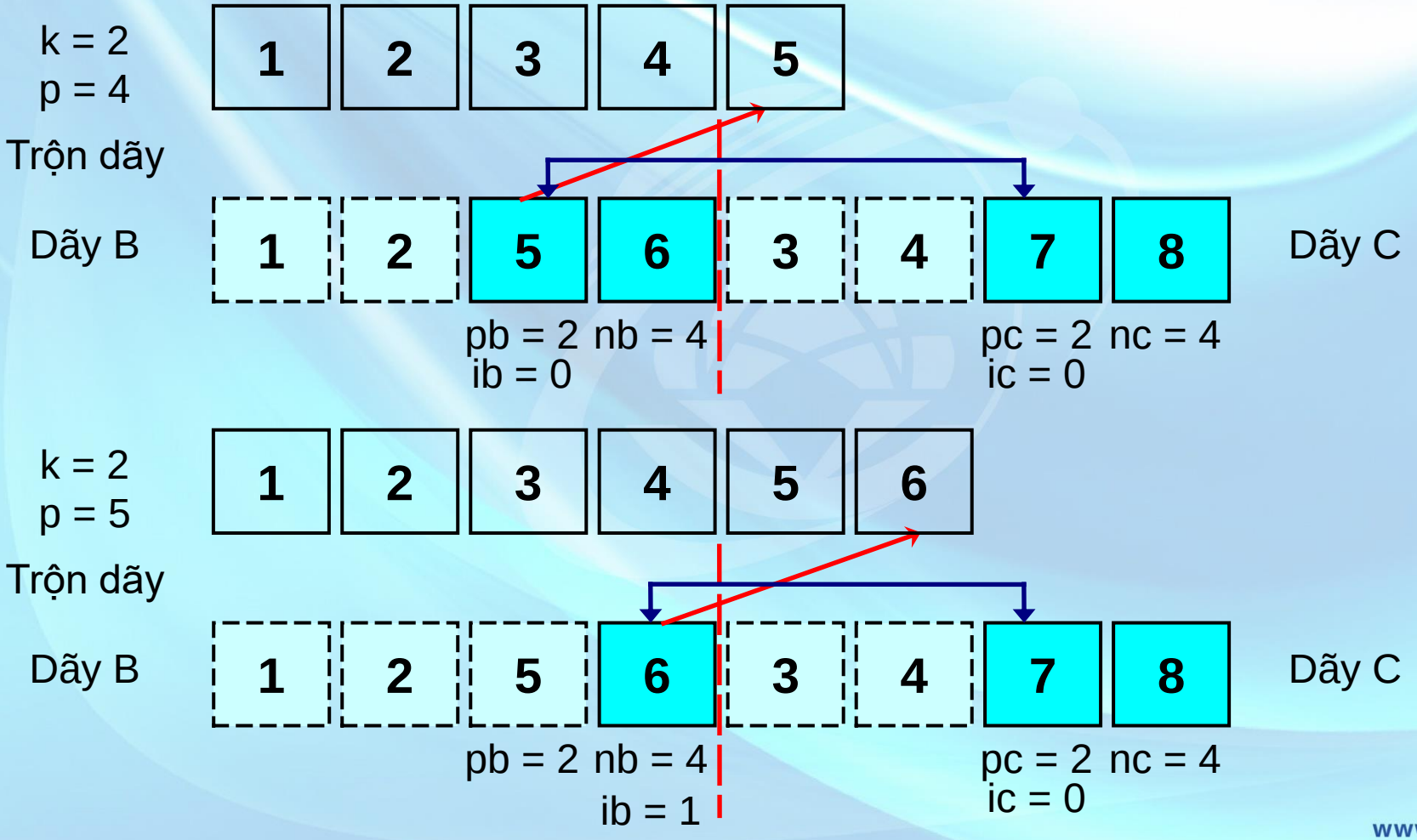
III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN



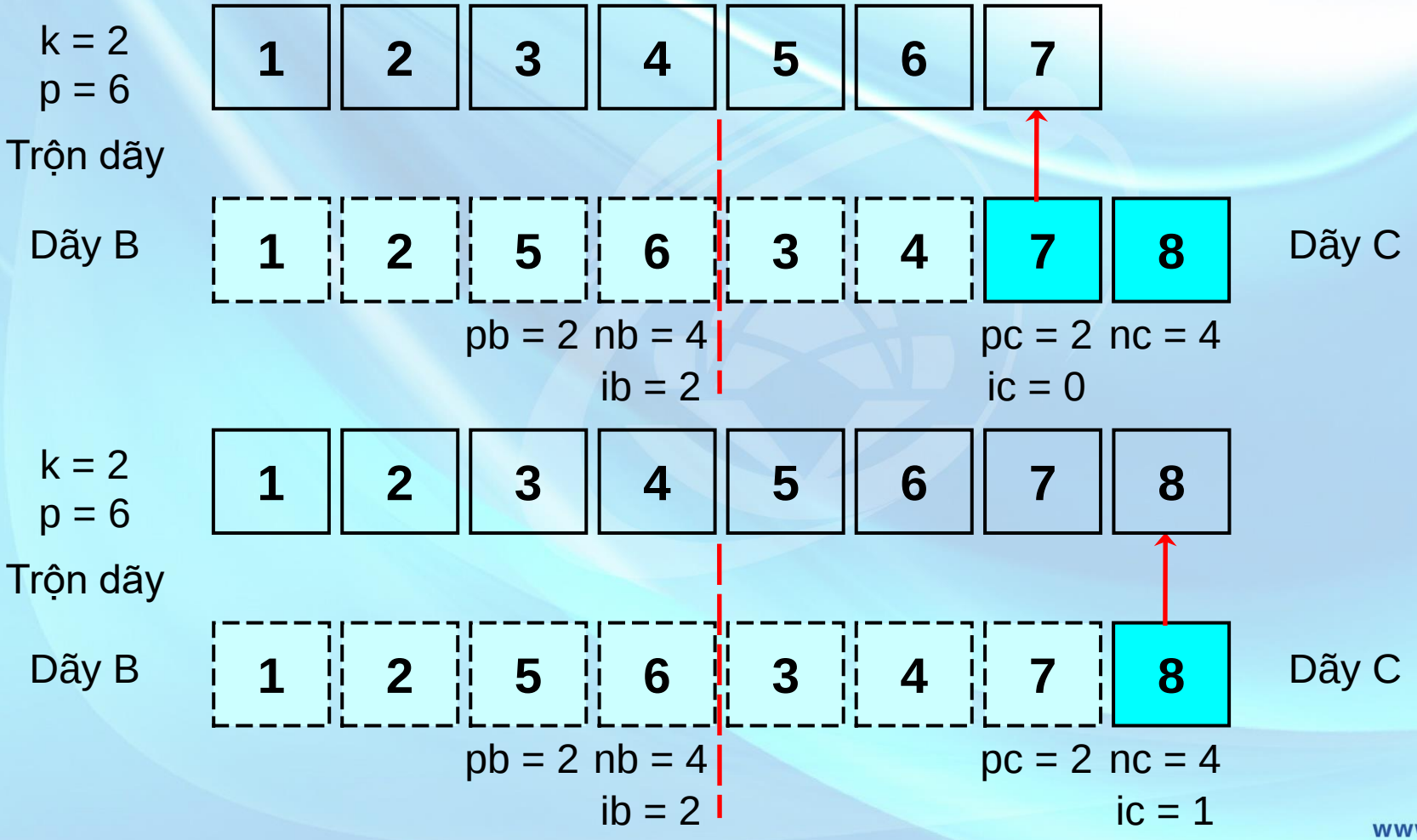
III. CÁC GIẢI THUẬT SẮP XẾP

PHƯƠNG PHÁP TRỘN



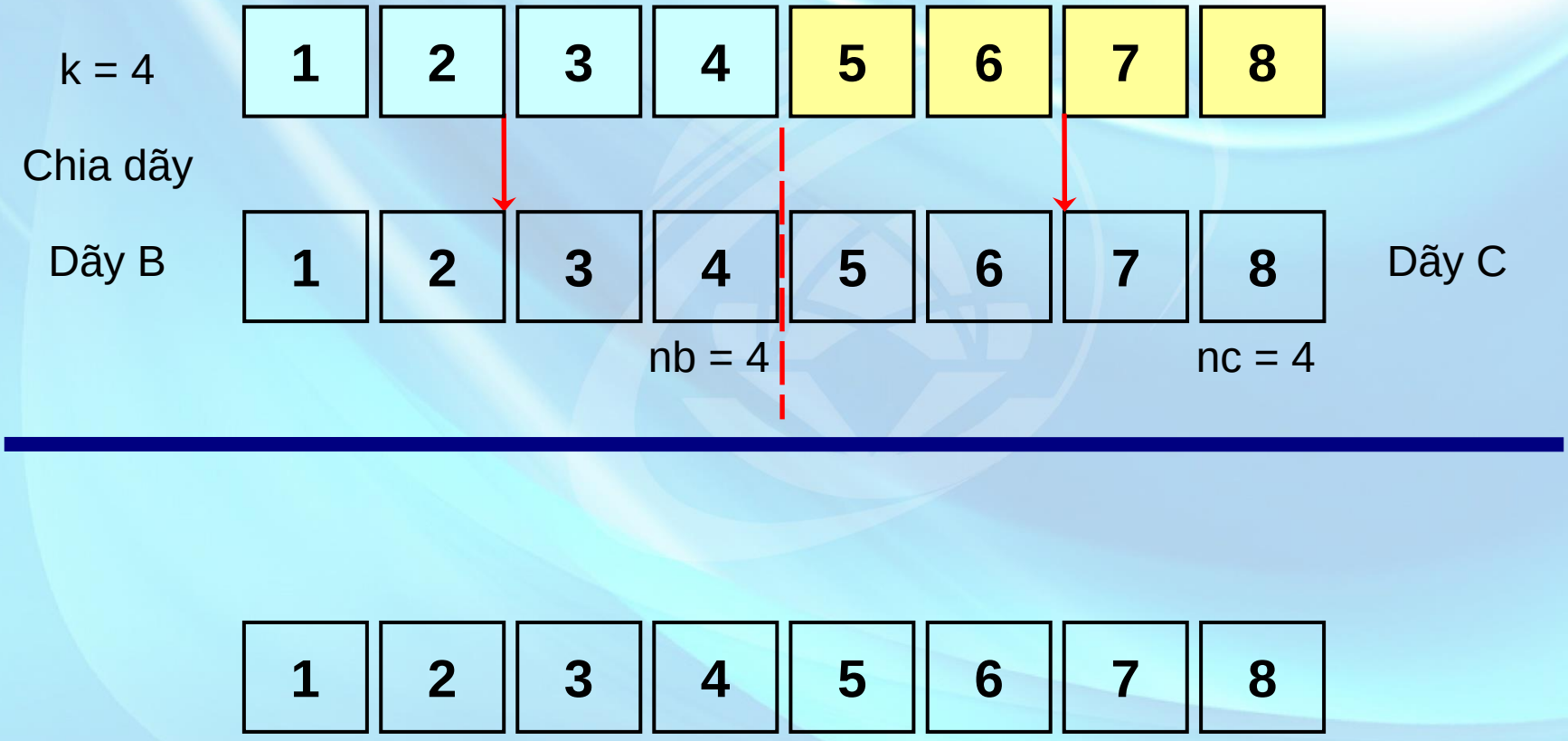
III. CÁC GIẢI THUẬT SẮP XẾP

PHƯƠNG PHÁP TRỘN



III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN



III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN

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=pb=pc=ib=ic=0, 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];
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖ PHƯƠNG PHÁP TRỘN (tt)

```
                pb+=kb;pc+=kc; ib=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=ic=0;nB-=kb;nC-=kc;
                kb = min(k, nB); kc = min(k, nC);
        }   }
} }
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN (tt)

```
void MergeSort(int A[], int n) {
    int p, pb, pc, i, k = 1;
    while (k < n) {
        p = pb = 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;
} }
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN

Đánh giá: Thuật toán trộn trực tiếp

	TỐT NHẤT	TRUNG BÌNH	XẤU NHẤT
Theo phép so sánh	O(nlogn)	O(nlogn)	O(nlogn)
Theo phép gán giá trị khóa	O(nlogn)	O(nlogn)	O(nlogn)

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN

Đánh giá: Thuật toán trộn trực tiếp

- Thích hợp cho các danh sách truy xuất tuần tự (file, danh sách đơn).
- Có thể thực hiện sắp xếp mà không cần nạp toàn bộ danh sách lên RAM (**External Sorting**)
- Trường hợp danh sách đã có những đoạn con có thứ tự → Trộn tự nhiên (**Natural Merge Sort**)

Sinh viên tự tìm hiểu và cài đặt thuật toán Trộn tự nhiên

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP TRỘN

Đánh giá: Độ phức tạp của thuật toán trộn tự nhiên

	TỐT NHẤT (có thứ tự)	**TRUNG BÌNH** (có thứ tự cục bộ)	**XẤU NHẤT** (không có thứ tự)
Theo phép so sánh	O(1)	O(nlogn)	O(nlogn)
Theo phép gán giá trị khóa	O(1)	O(nlogn)	O(nlogn)

IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖KHÁI NIỆM

Hàng đợi ưu tiên (Priority Queue):

- Là hàng đợi với các tao thác enqueue, dequeue
- Thao tác dequeue cho phép lấy phần tử nhỏ nhất ra khỏi hàng đợi.

IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖KHÁI NIỆM

Ứng dụng của hàng đợi ưu tiên:

- **External Sorting** với Merge Sort: mỗi danh sách con là một hàng đợi ưu tiên.
- Quản lý các ngắt (**interrupt handler**), thời gian chờ (**wakeup time**), … trong hệ điều hành.
- Dùng trong các chiến lược tham lam (**Greedy**)
  - Thuật toán **Prim** (Cây khung tối tiểu – Minimum Spanning Tree)
  - Thuật toán **A***

IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Hàng đợi ưu tiên là mảng:

- Các phần tử trong mảng đảm bảo tính chất của một heap nhị phân.
- Thao tác enqueue:
  - Thêm phần tử p vào cuối mảng
  - Thực hiện thao tác heapify phần tử p theo hướng về đầu mảng (**percolateUp**)

IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Hàng đợi ưu tiên là mảng:

- Thao tác dequeue:
  - Thay phần tử đầu mảng bằng phần tử cuối mảng p
  - Thực hiện heapify phần tử đầu mảng p theo hướng về cuối mảng (**perlocateDown**)

IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Hàng đợi ưu tiên là mảng:

- Thao tác increase (tăng độ ưu tiên):
  - Tăng giá trị của phần tử p
  - Thực hiện thao tác heapify phần tử p theo hướng về đầu mảng

IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Hàng đợi ưu tiên là mảng:

- Thao tác decrease (giảm độ ưu tiên):
  - Giảm giá trị của phần tử p
  - Thực hiện thao tác heapify phần tử p theo hướng về cuối mảng

(Sinh viên tự tìm hiểu cách cài đặt chi tiết theo cách tương tự Heap Sort)

IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

enqueue 7

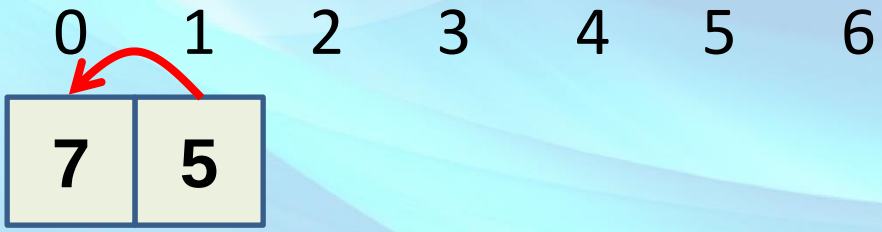
IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

enqueue 5



IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

perlocateUp

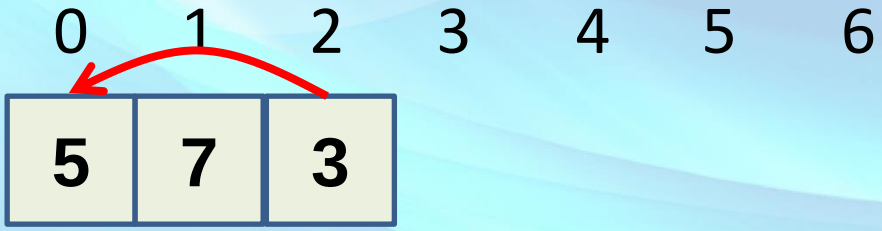
IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

enqueue 3



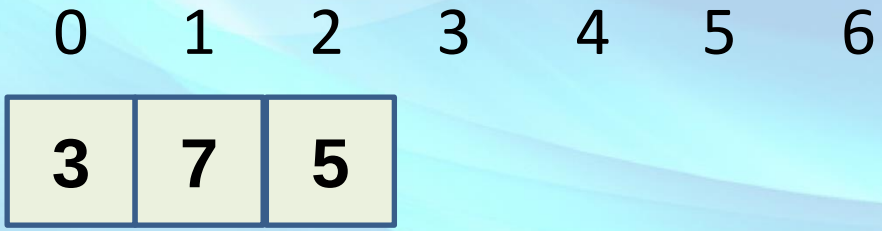
IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

perlocateUp



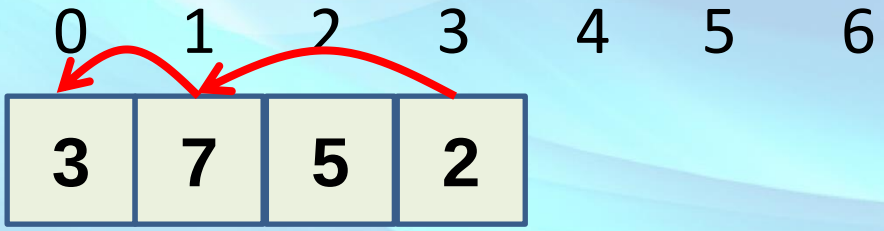
IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

enqueue 2



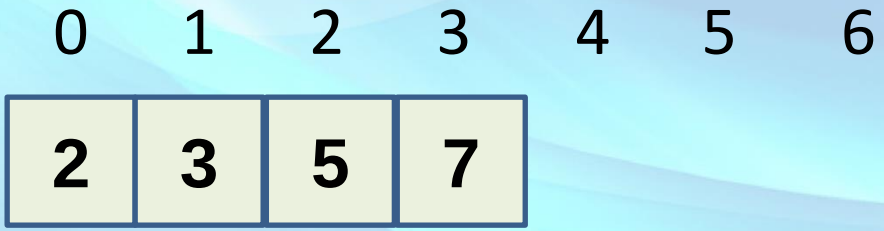
IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

perlocateUp



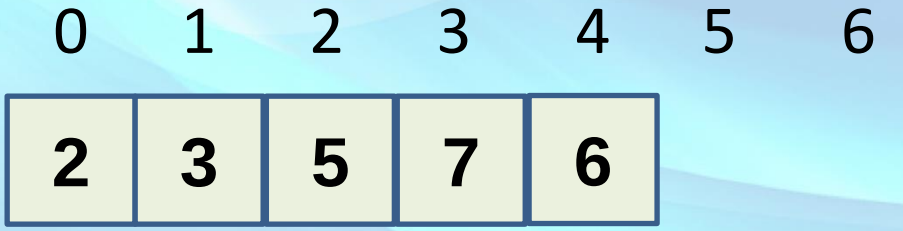
IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

enqueue 6



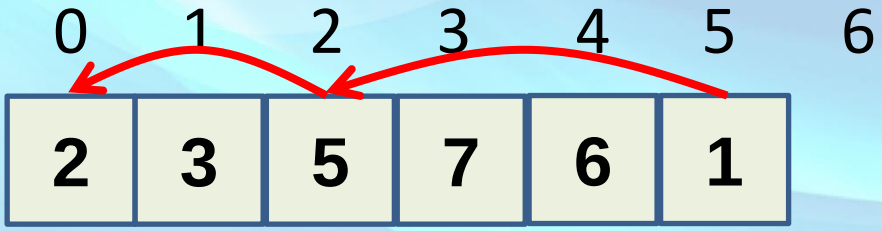
IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

enqueue 1



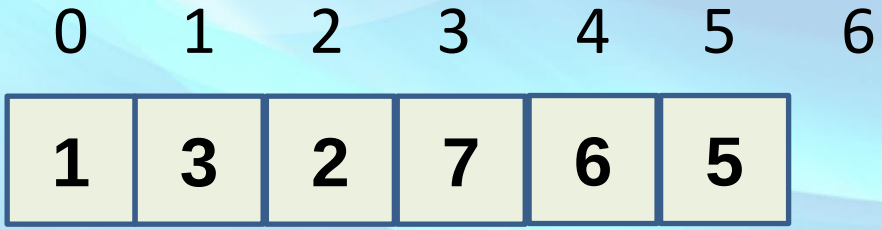
IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

perlocateUp



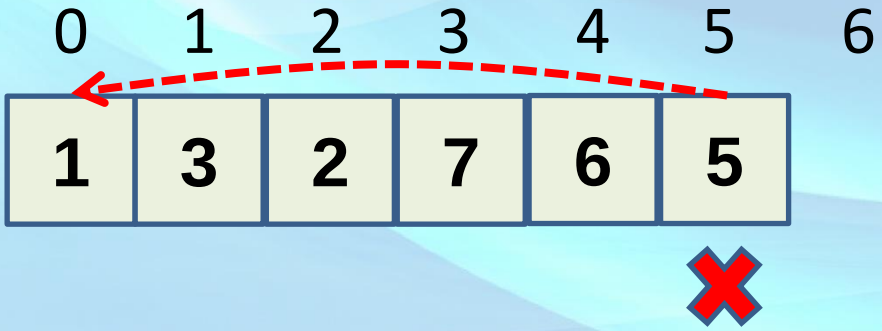
IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

dequeue



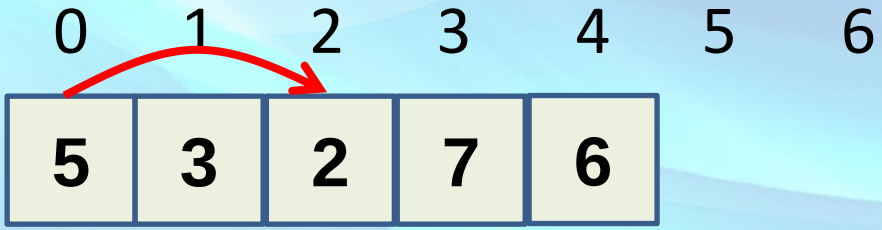
IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

perlocateDown



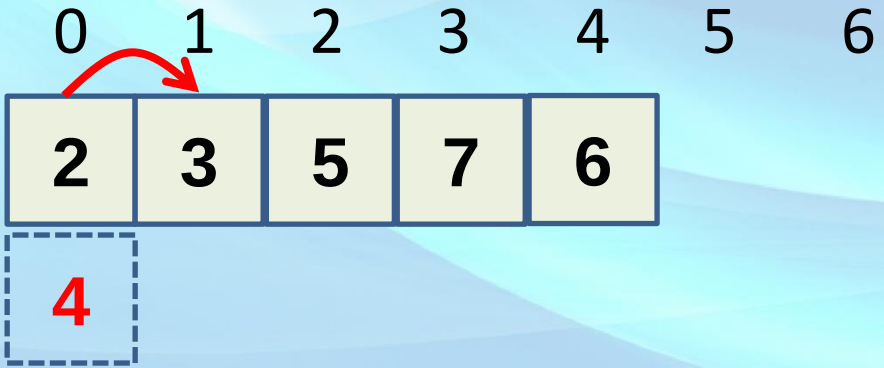
IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

increase (0, 2)



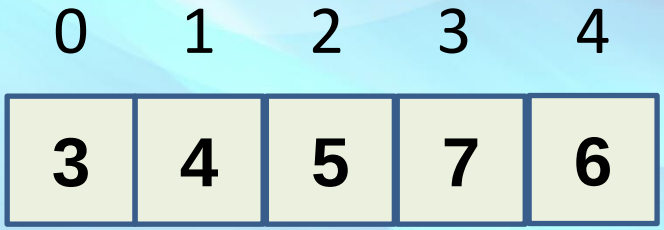
IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Quá trình tính toán: Ưu tiên số nhỏ

Giả sử có các thao tác: enqueue 7, enqueue 5, enqueue 3, enqueue 2, enqueue 6, enqueue 1, dequeue, increase (0,2)

perlocateDown



IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN

❖CÀI ĐẶT VỚI HEAP NHỊ PHÂN

Đánh giá:

- Độ phức tạp của thao tác enqueue: O(logn).
- Độ phức tạp của thao tác dequeue: O(logn).

BÀI TẬP

❖BÀI TẬP 1

Cho mảng A={8,2,1,9,4,5,7,6,3}. Hãy viết hàm sắp xếp và trình bày từng bước quá trình sắp xếp mảng A theo thứ tự giảm dần (>) với thuật toán:

a) Heap Sort

b) Quick Sort

c) Merge Sort

BÀI TẬP

❖BÀI TẬP 2

Trộn tự nhiên (Natural Merge Sort) phân phối các phần tử của dãy theo run. Trong đó một run là một dãy các phần tử đã có thứ tự cần sắp. Hãy trình bày thuật toán sắp xếp theo cách trộn tự nhiên và cho một ví dụ minh họa.

BÀI TẬP

BÀI TẬP 3

Cho dãy số A = {7,2,4,5,1,3,6,8}. Cho biết giải thuật nào thích hợp để sắp xếp theo thứ tự tăng dần cho dãy A nhất? giải thích vì sao?