THUẬT TOÁN QUICK SORT

- Ý tưởng:

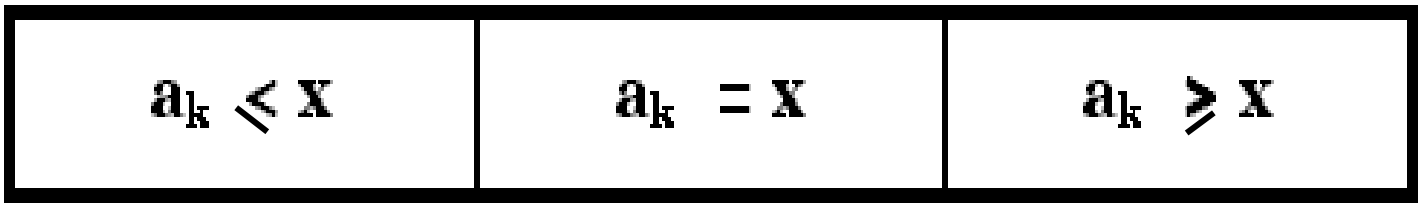
- Giải thuật QuickSort sắp xếp dãy $\mathbf{a_1}$, $\mathbf{a_2}$ ..., $\mathbf{a_N}$ dựa trên việc phân hoạch dãy ban đầu thành 3 phần :
  - Phần 1: Gồm các phần tử có giá trị bé hơn **x**
  - Phần 2: Gồm các phần tử có giá trị bằng **x**
  - Phần 3: Gồm các phần tử có giá trị lớn hơn **x**

  với x là giá trị của một phần tử tùy ý trong dãy ban đầu.

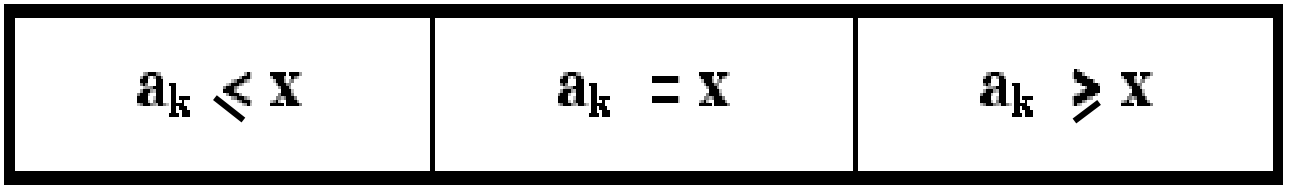
ThS.Nguyễn Thúy Loan

THUẬT TOÁN QUICK SORT

- Sau khi thực hiện phân hoạch, dãy ban đầu được phân thành 3 đoạn:
  - 1. $a_k \leq x$, với k = 1 .. j
  - 2. $a_k = x$, với k = j+1 .. i -1
  - 3. $a_k \geq x$, với k = I n



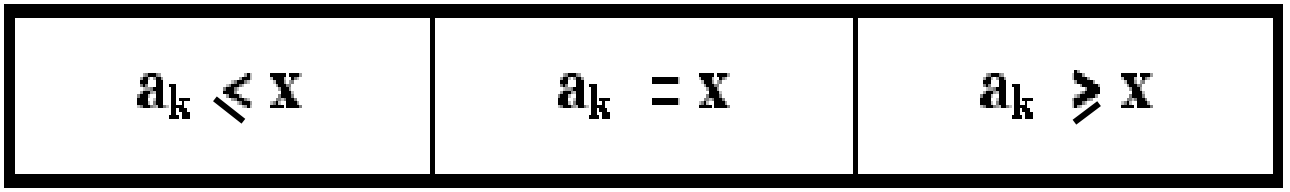
THUẬT TOÁN QUICK SORT



- Đoạn thứ 2 đã có thứ tự.
- Nếu các đoạn 1 và 3 chỉ có 1 phần tử: đã có thứ tự

  → khi đó dãy con ban đầu đã được sắp.

THUẬT TOÁN QUICK SORT



- Đoạn thứ 2 đã có thứ tự.
- Nếu các đoạn 1 và 3 có nhiều hơn 1 phần tử thì dãy ban đầu chỉ có thứ tự khi các đoạn 1, 3 được sắp.
- Để sắp xếp các đoạn 1 và 3, ta lần lượt tiến hành việc phân hoạch từng dãy con theo cùng phương pháp phân hoạch dãy ban đầu vừa trình bày …

THUẬT TOÁN QUICK SORT

- Bước 1: Nếu left ≥ right //dãy có ít hơn 2 phần tử

  Kết thúc; //dãy đã được sắp xếp

- Bước 2: Phân hoạch dãy a_{left} … a_{right} thành các đoạn:

  a_{left}.. a_j, a_{j+1}.. a_{i-1}, a_i.. a_{right}

  Đoạn 1 $\leq x$

  Đoạn 2: a_{j+1}.. a_{i-1} $= x$

  Đoạn 3: a_i.. a_{right} $\geq$ x

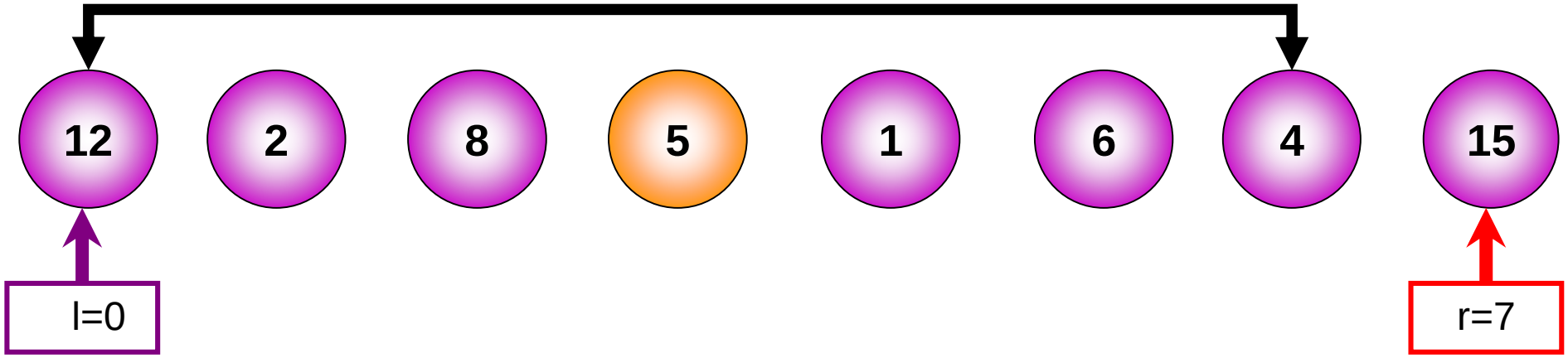
- Bước 3: **Sắp xếp đoạn 1**: a_{left}.. a_j
- Bước 4: **Sắp xếp đoạn 3**: a_i.. a_{right}

Quick Sort – Ví Dụ

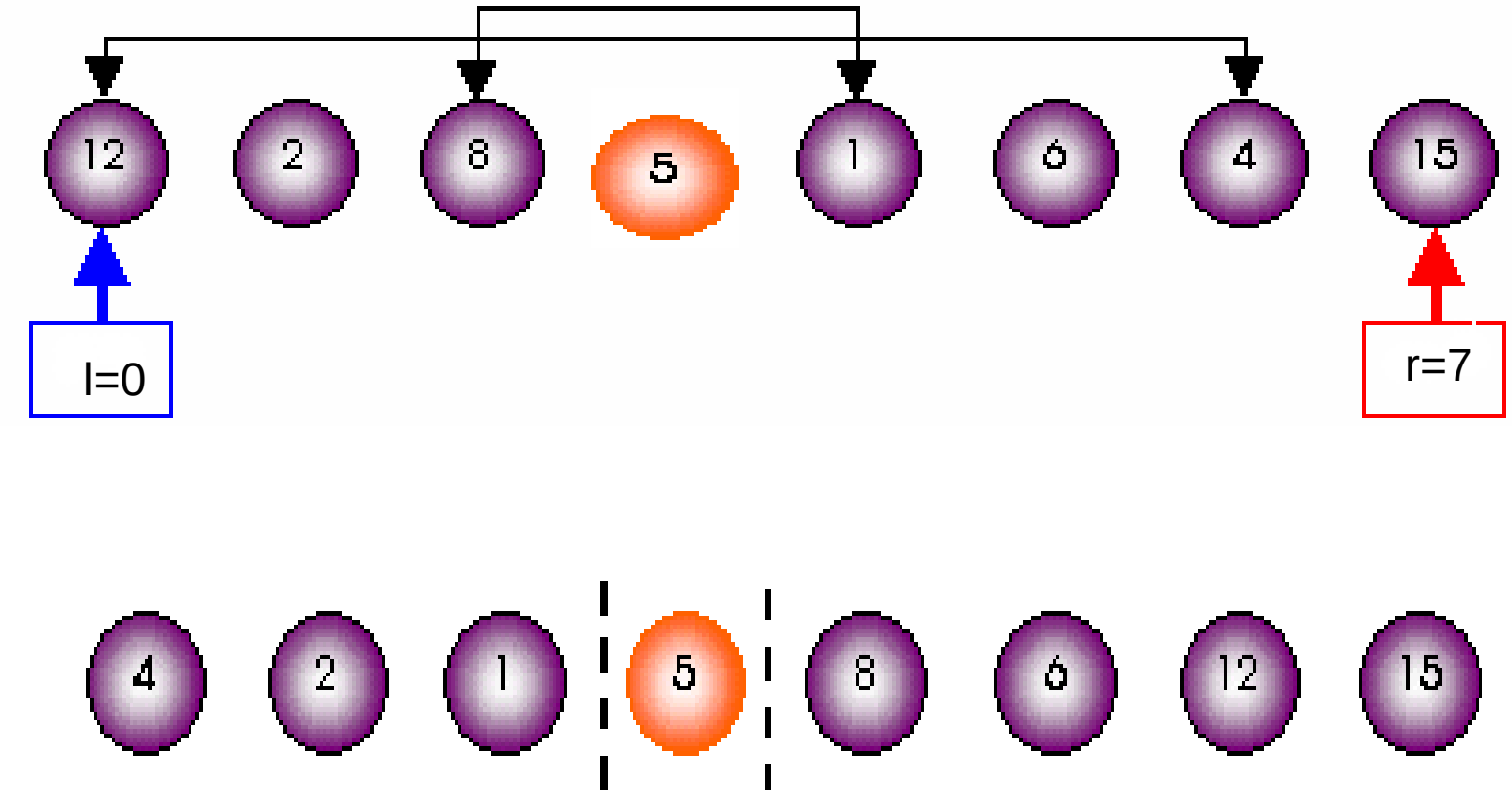
- Cho dãy số a:

12	2	8	5
1	6	4	15

Phân hoạch đoạn l =0, r = 7: x = a[3] = 5



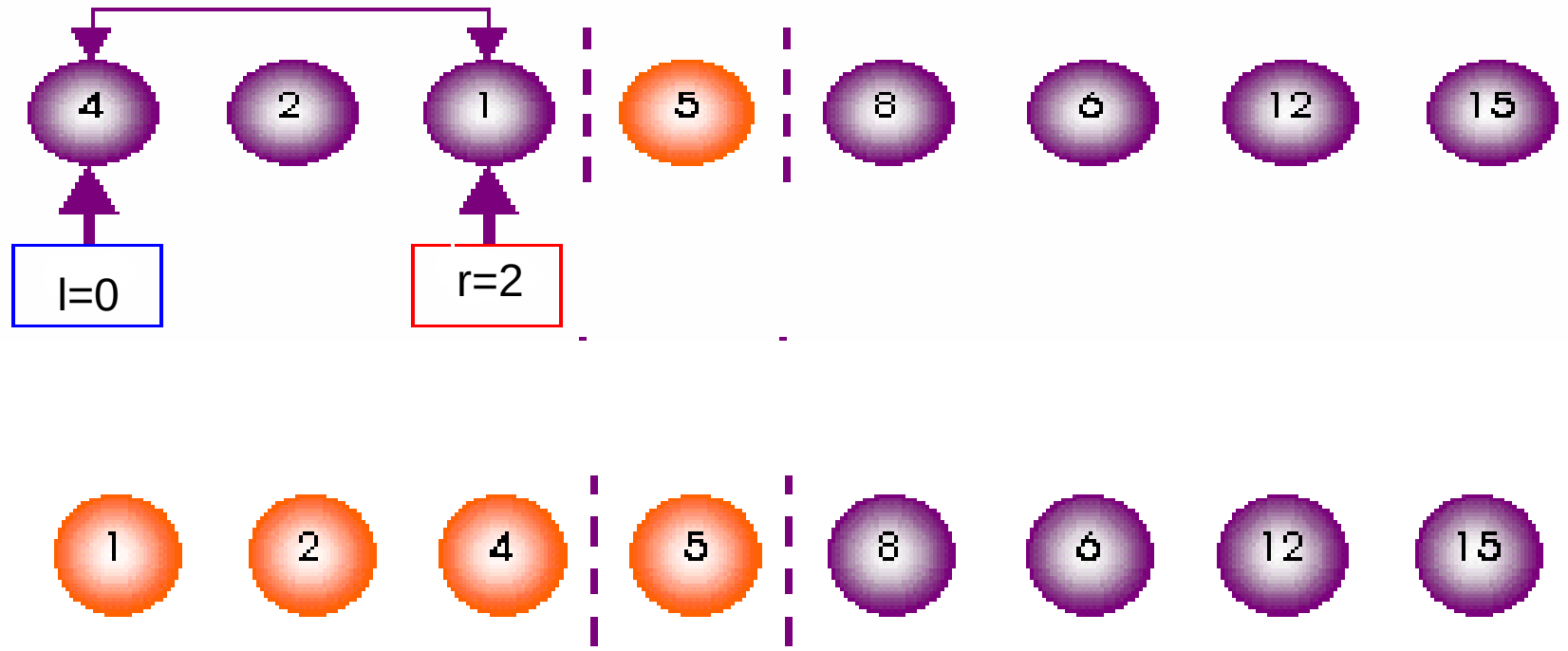
Quick Sort – Ví Dụ



Quick Sort – Ví Dụ

- Phân hoạch đoạn l =0, r = 2:

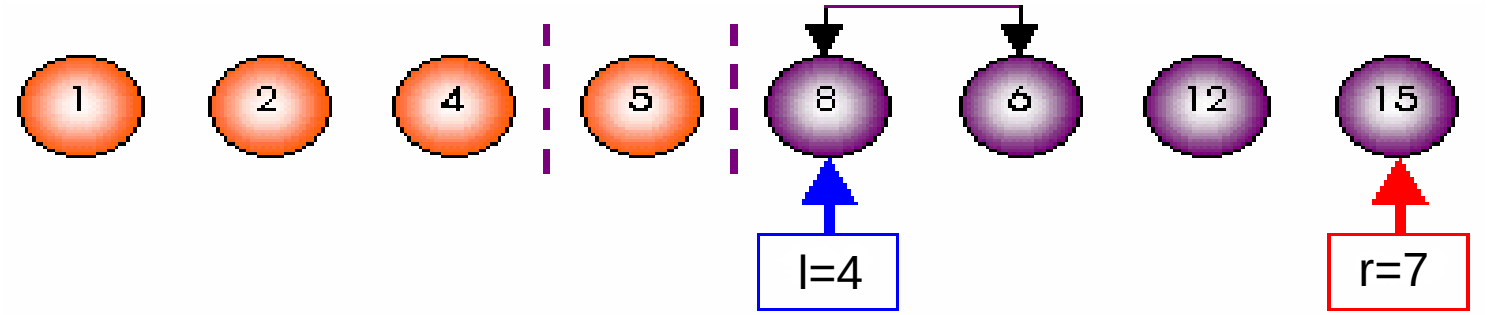
x = a[2] = 2

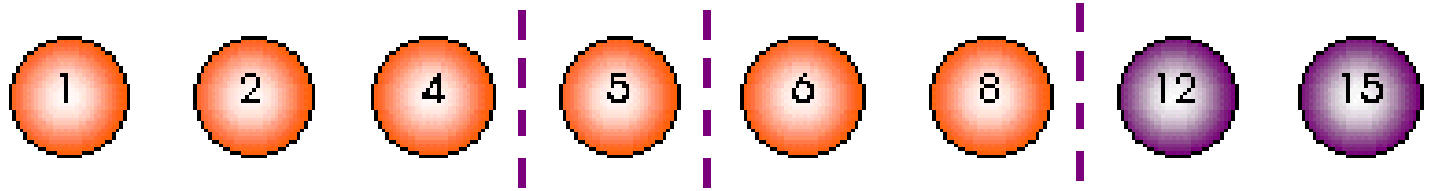


Quick Sort – Ví Dụ

- Phân hoạch đoạn l = 4, r = 7:

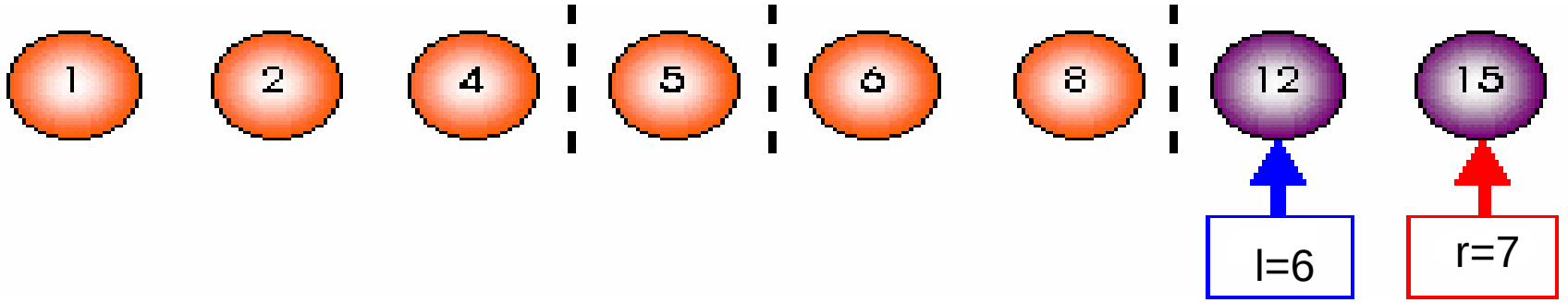
x = a[5] = 6

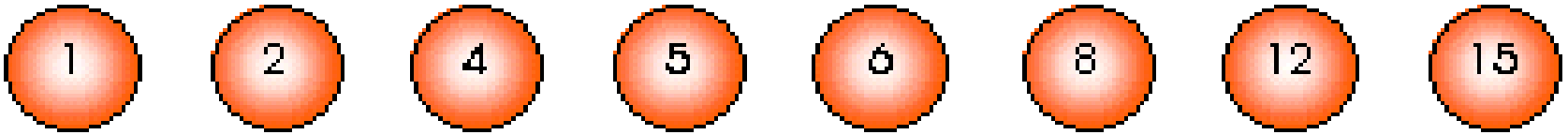




- Phân hoạch đoạn l = 6, r = 7:

x = a[6] = 6





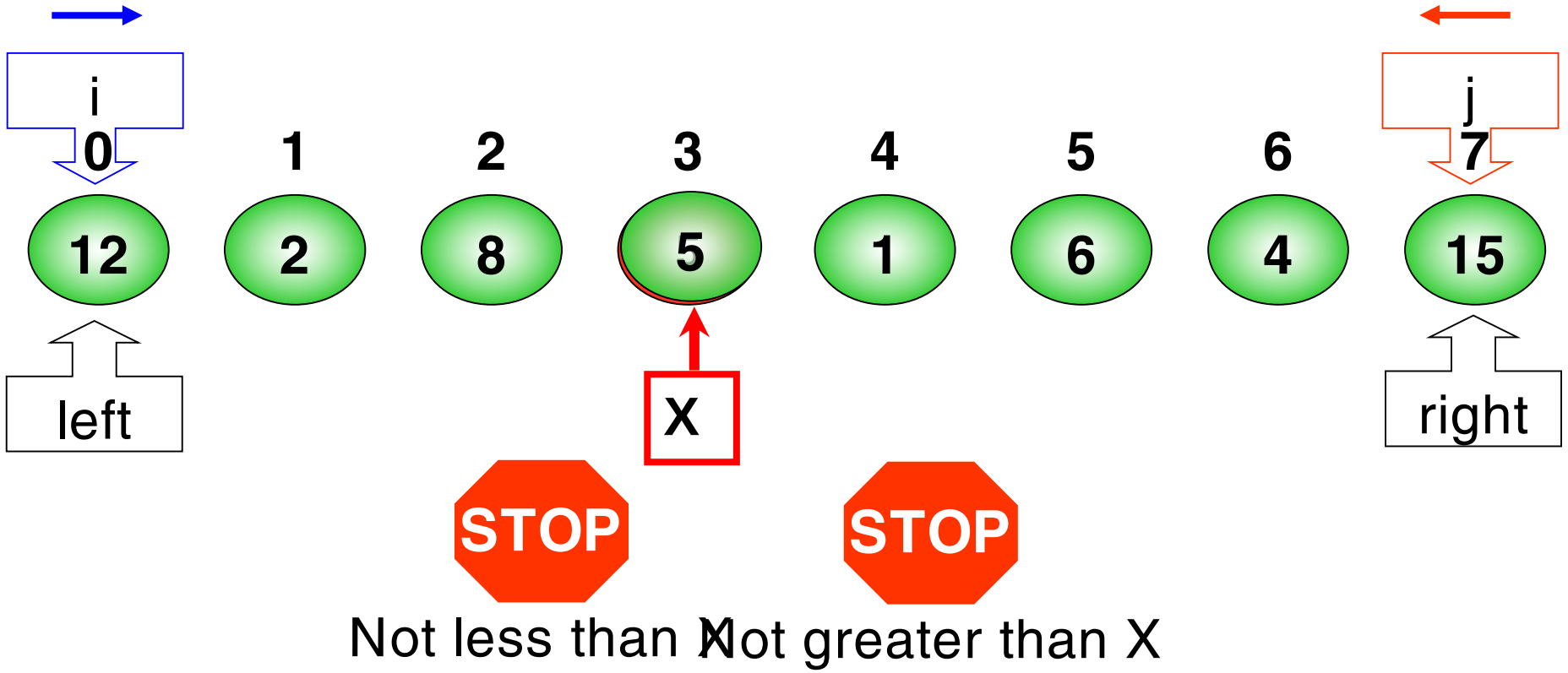
THUẬT TOÁN QUICK SORT

- Bước 1 : Chọn tùy ý một phần tử a[k] trong dãy là giá trị mốc (l ≤ k ≤ r):

  x = a[k]; i = l; j = r;

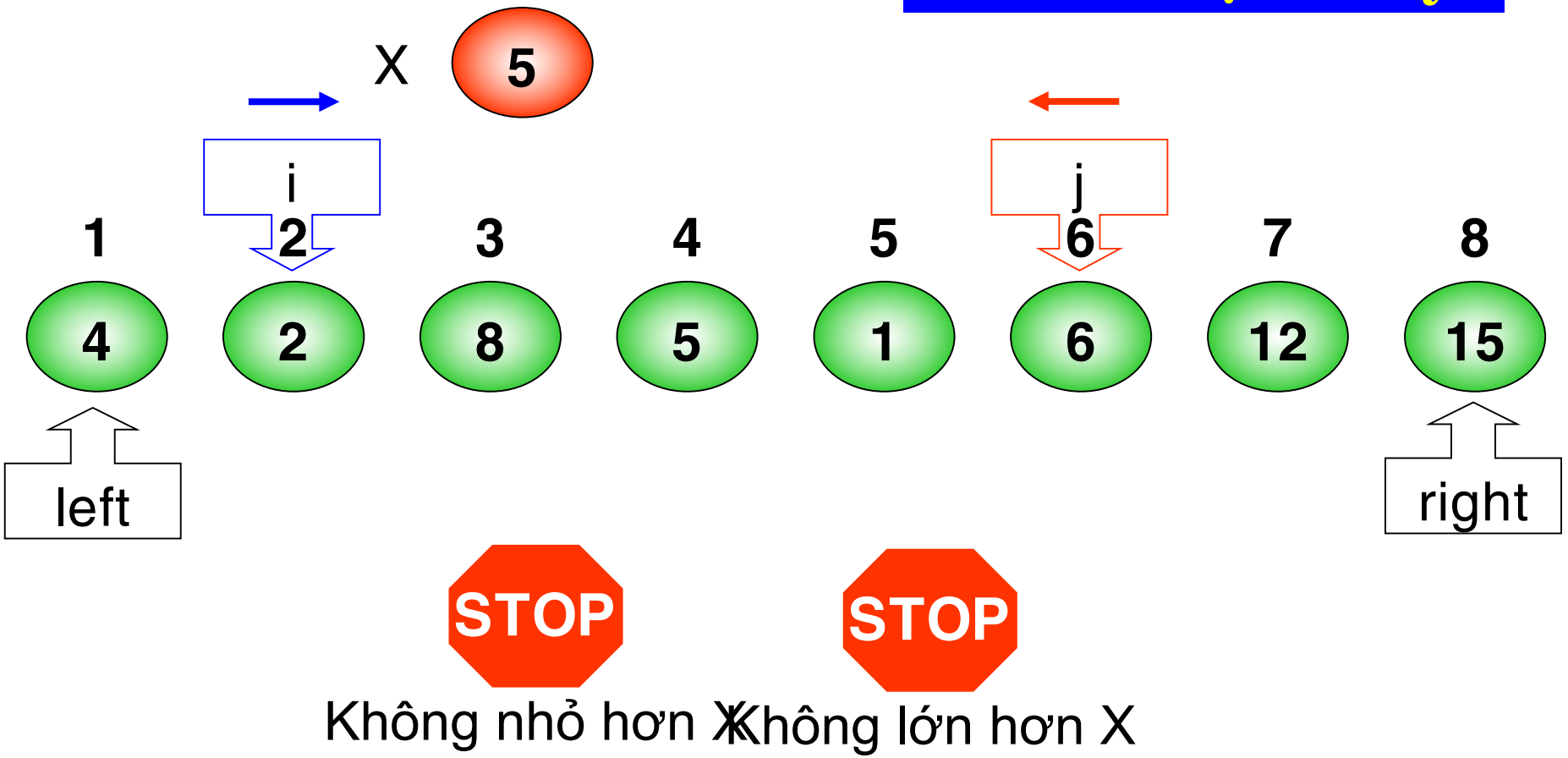
- Bước 2 : Phát hiện và hiệu chỉnh cặp phần tử a[i], a[j] nằm sai chỗ :
  - Bước 2a : Trong khi (a[i] < x) i++;
  - Bước 2b : Trong khi (a[j] >x) j--;
  - Bước 2c : Nếu i < j Đoicho(a[i] , a[j]);
- Bước 3 : Nếu i < j: Lặp lại Bước 2.
  Ngược lại: Dừng

Quick Sort – Ví Dụ

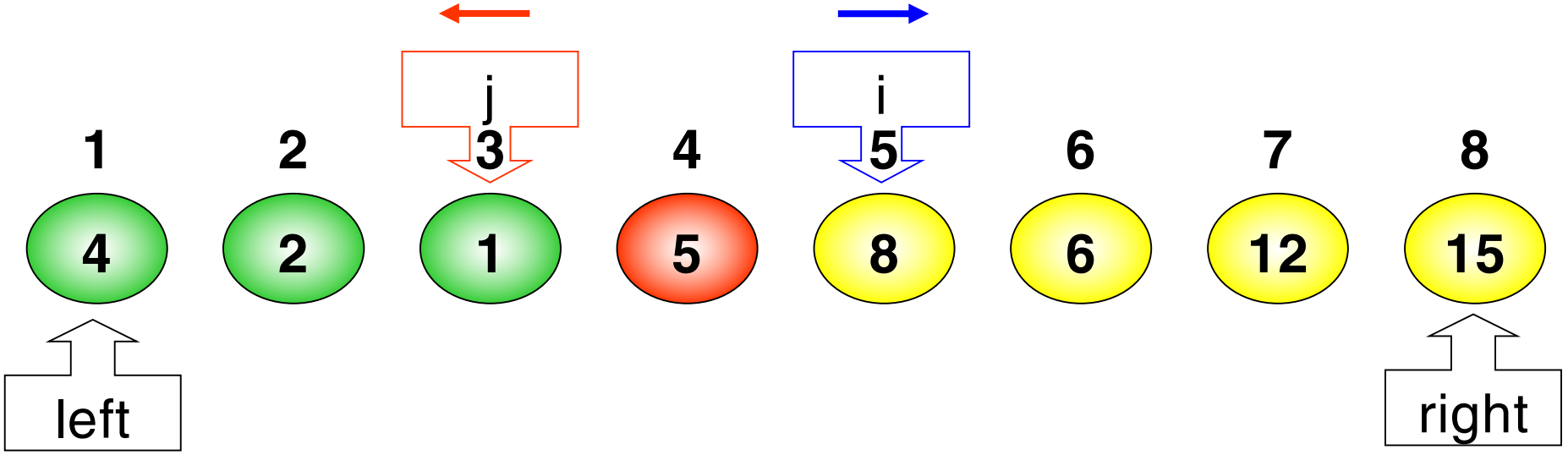


Quick Sort – Ví Dụ

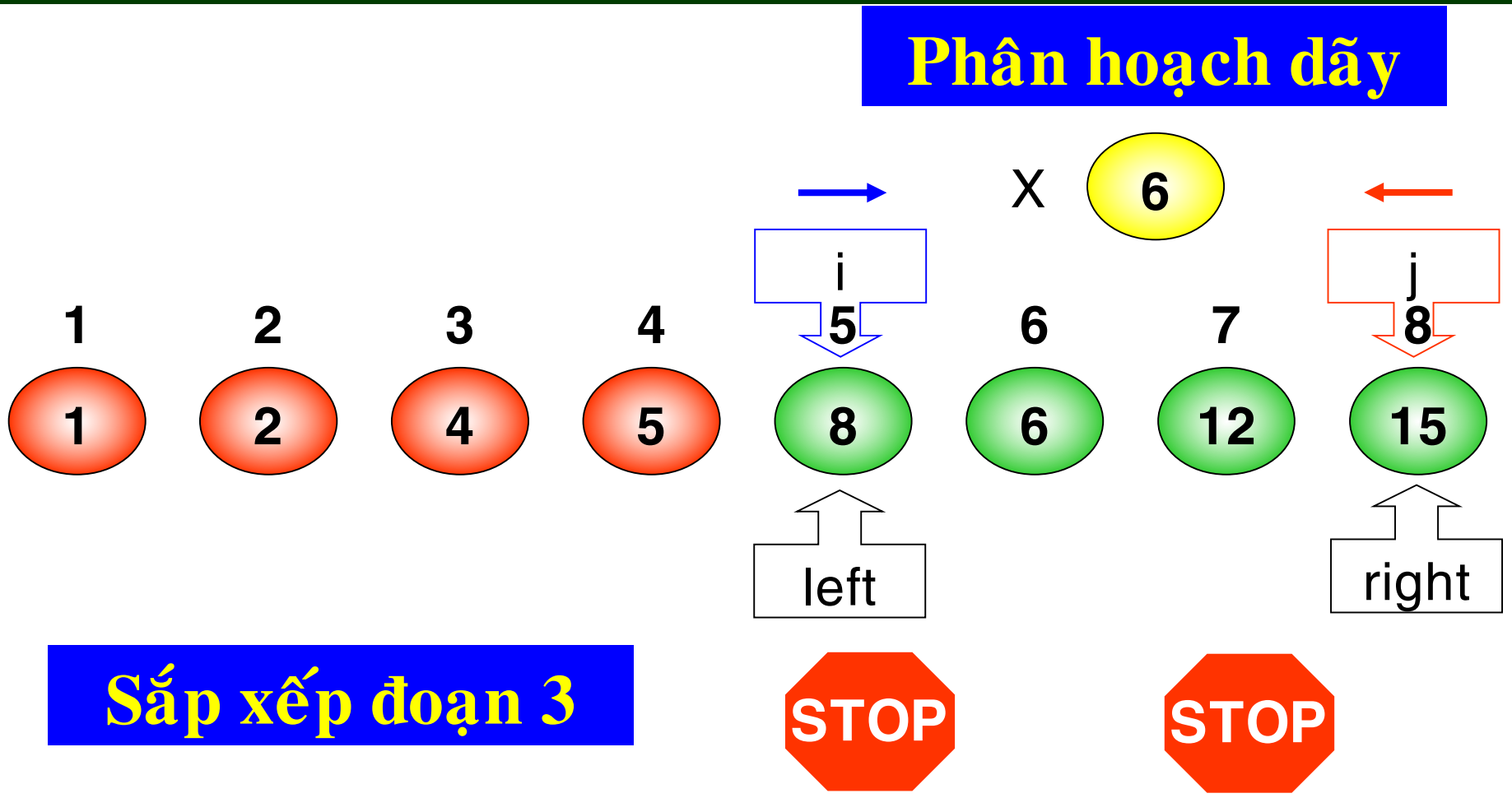
Phân hoạch dãy



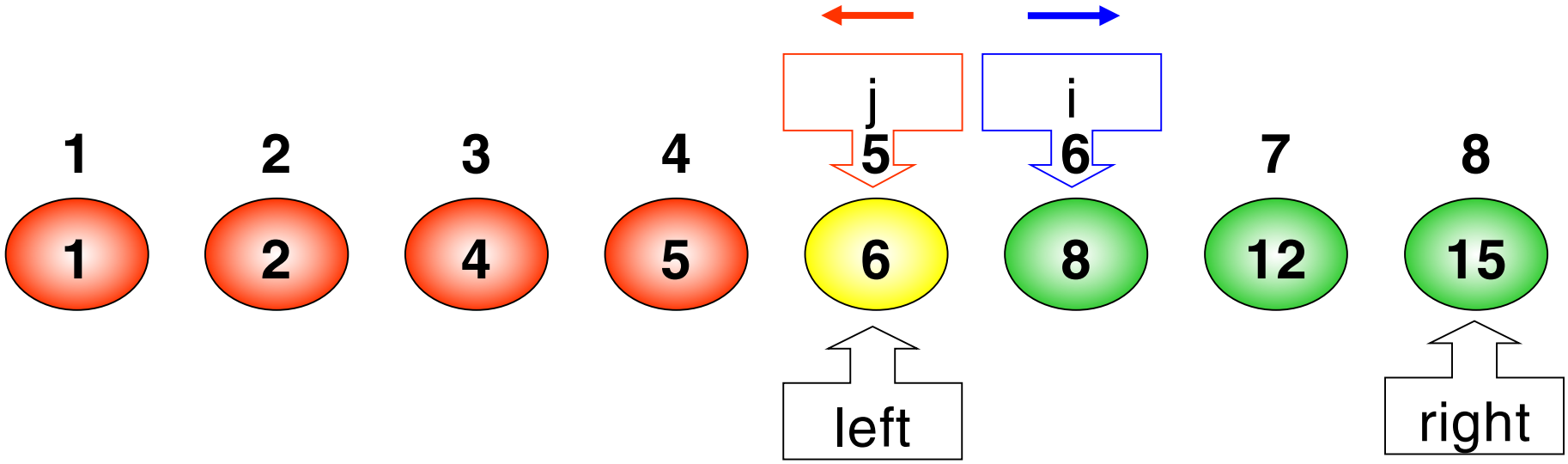
Quick Sort – Ví Dụ



Quick Sort – Ví Dụ



Quick Sort – Ví Dụ



Sắp xếp đoạn 3

3. HÀM CÀI ĐẶT

```
void  QuickSort (  int  a [ ], int  left, int  right )
{
    int  i,  j ;    int  x ;
    x = a[ (left+right) / 2 ];   i = left ;  j = right;
    while( i < j )
    {    while ( a[i] < x )     i++;
         while ( a[j] > x )     j--;
         if(i <= j)
         {         Swap ( a[i] , a[j] )  ;    i++ ;   j-- ;     }
    }
    if  ( left < j )         QuickSort ( a, left, j );
    if  ( i<right )          QuickSort ( a, i, right ) ;
}
```

3. HÀM CÀI ĐẶT

```
void   sapxepQuickSort (  int  a [ ], int  n )
{
        int   left  = 0  ;
        int   right = n-1 ;
        QuickSort ( a, left, right  );
}
```

Độ Phức Tạp Của Quick Sort

Trường hợp	Độ phức tạp
Tốt nhất	$n*log(n)$
Trung bình	$n*log(n)$
Xấu nhất	n^2