VIETNAM AUSTRALIA Vocational College

# LẬP TRÌNH JAVASCRIPT CƠ BẢN

## *MẢNG 1 CHIỀU VÀ CÁC KỸ THUẬT THAO TÁC, SẮP XẾP*

daynghevietuc.com

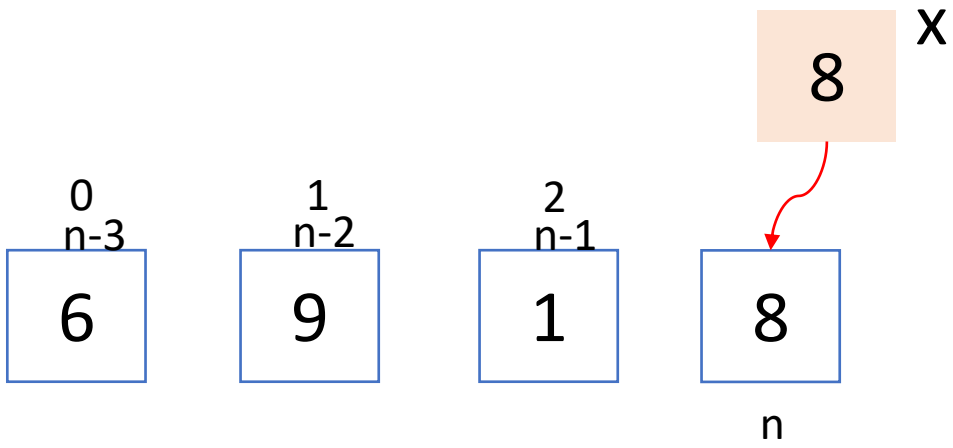
## ***Nội dung:***

1. Các kỹ thuật thao tác với mảng một chiều (cơ bản)
2. Các thuật toán sắp xếp trên mảng một chiều
3. Các kỹ thuật thao tác với mảng một chiều (phổ biến)
4. Phương pháp tìm kiếm tuyến tính (Linear Search)
5. Công cụ quản lý các thư viện lập trình (npm)

# Các kỹ thuật thao tác mảng một chiều (cơ bản)

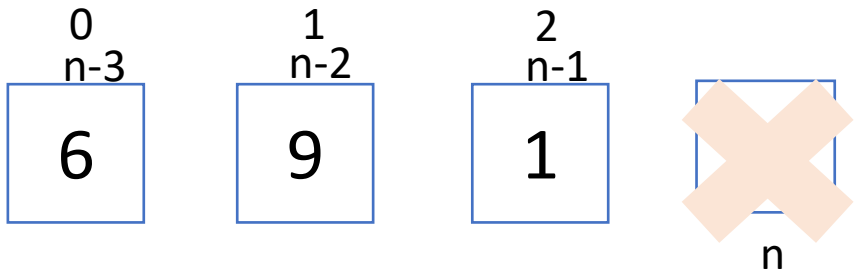
❑ Thêm phần tử vào cuối mảng:

```
function insertLast(x) {
    arr[n] = x;  // x: giá trị cần thêm
    n++;
}
```




❑ Xóa phần tử ở cuối mảng:

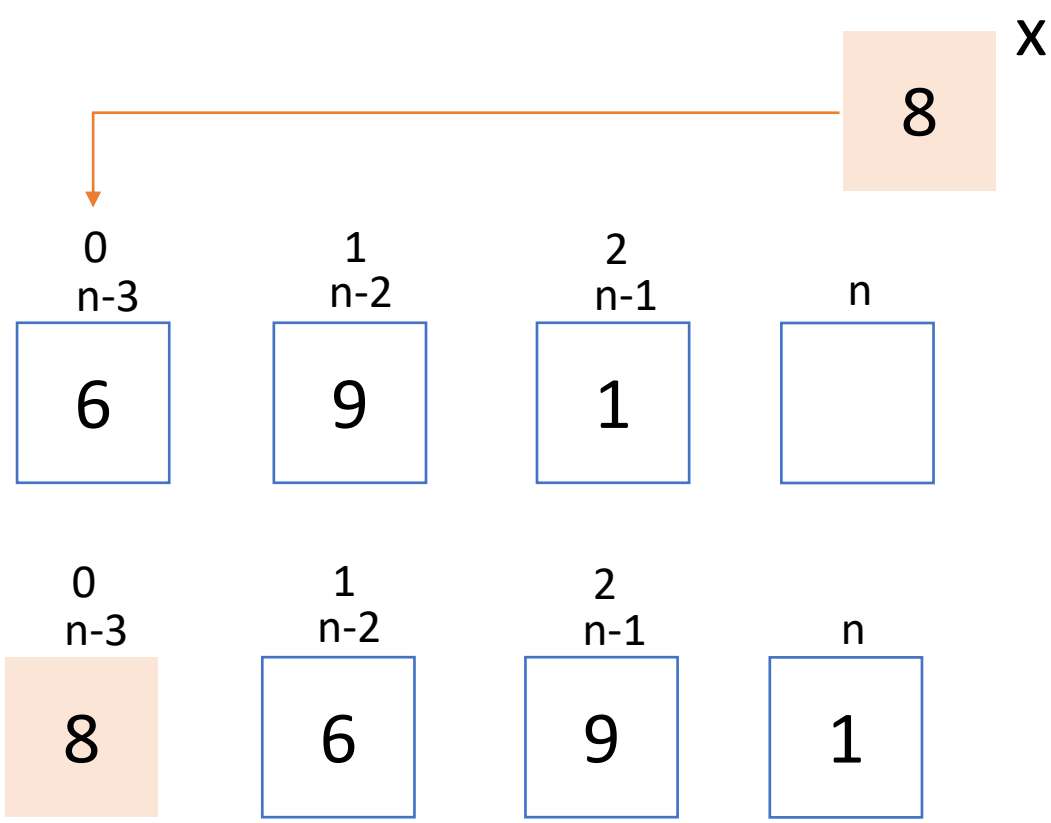
```
function deleteLast() {
    arr.length--;
    n--;
}
```



# Các kỹ thuật thao tác mảng một chiều (cơ bản)

- Thêm phần tử vào đầu mảng:

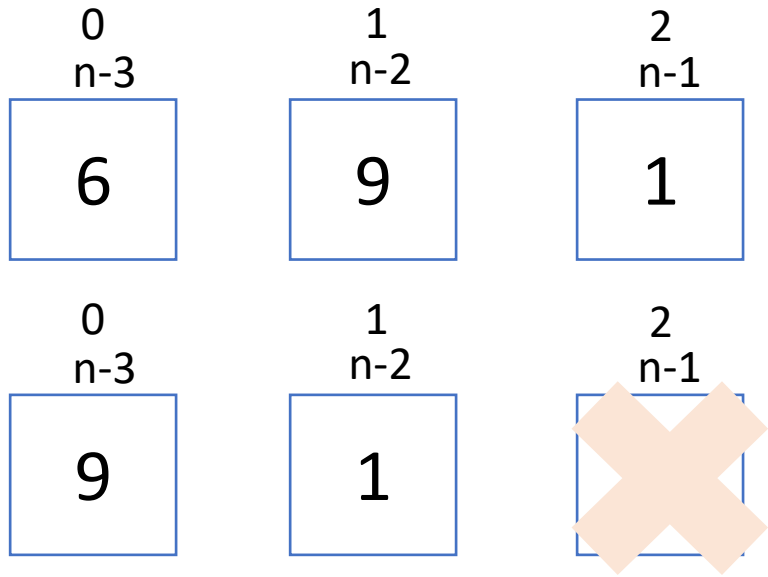
```
function insertFirst(x) {
    for (let i=n; i>0; i--) {
        arr[i] = arr[i-1];
    }
    arr[0] = x;  //x: giá trị cần thêm
    n++;
}
```



# Các kỹ thuật thao tác mảng một chiều (cơ bản)

- Xóa phần tử ở đầu mảng:

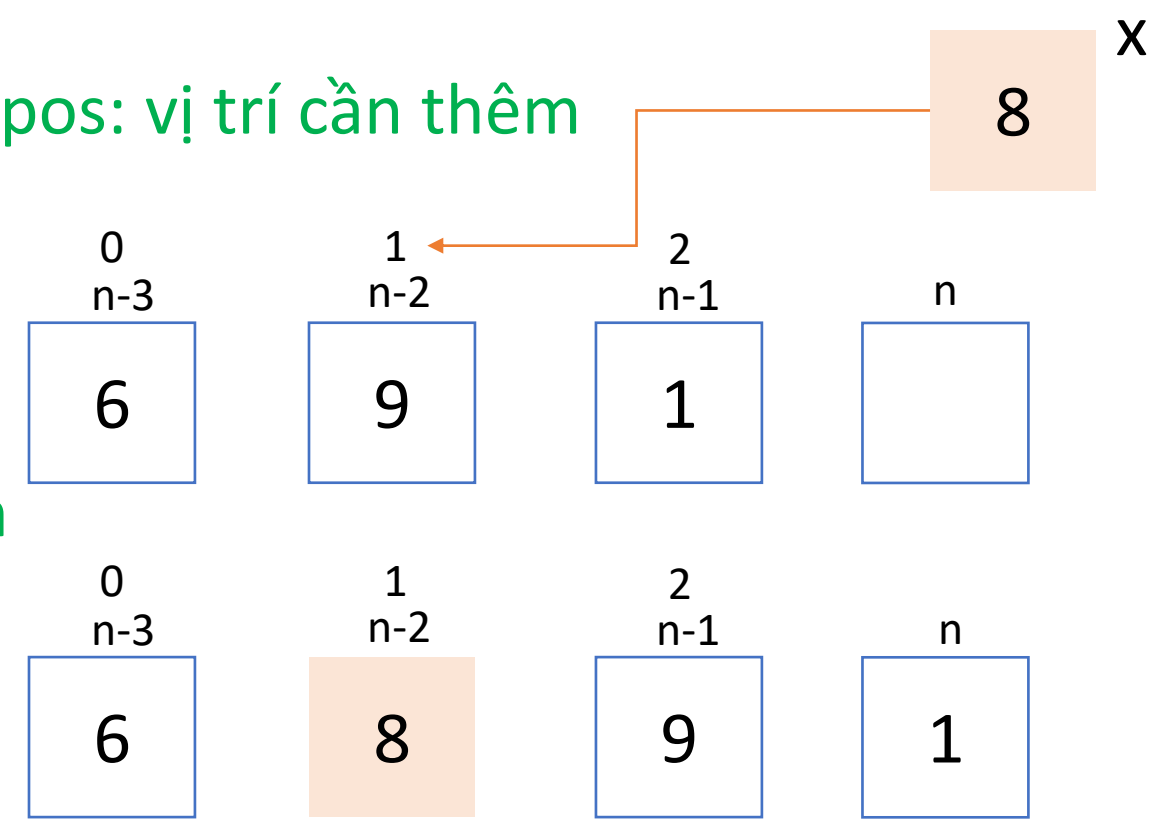
```
function deleteFirst() {
    for (let i=n; i<n-1; i++) {
        arr[i] = arr[i+1];
    }
    arr.length--;
    n--;
}
```



# Các kỹ thuật thao tác mảng một chiều (cơ bản)

- Thêm phần tử vào vị trí mảng:

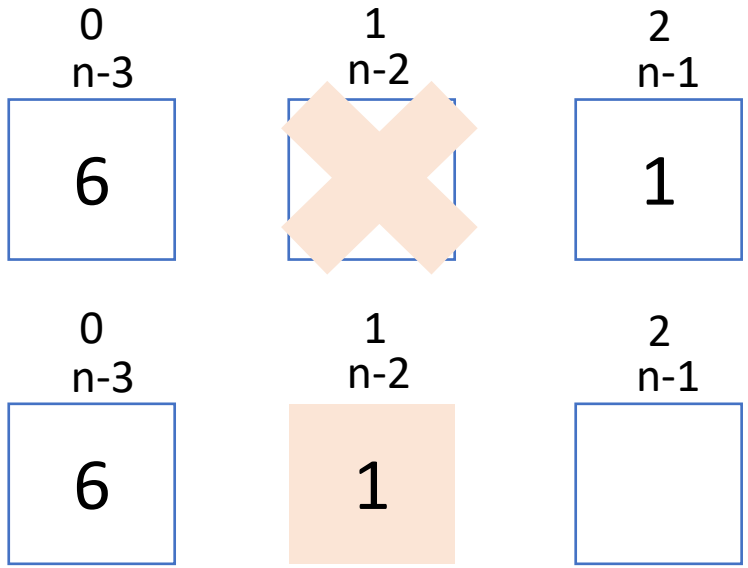
```
function insertByPosition(x, pos) {//pos: vị trí cần thêm
    for (let i=n; i>pos; i--) {
        arr[i] = arr[i-1];
    }
    arr[pos] = x;  //x: giá trị cần thêm
    n++;
}
```



# Các kỹ thuật thao tác mảng một chiều (cơ bản)

- Xóa phần tử ở vị trí mảng:

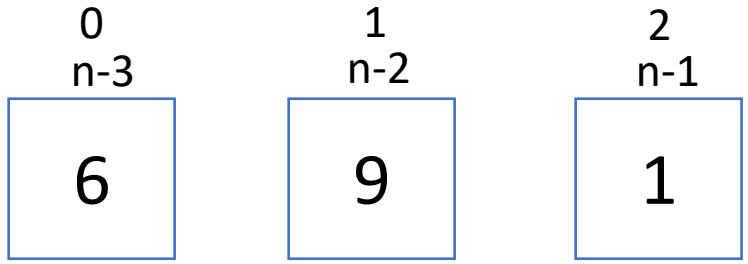
```
function deleteByPosition(x, pos) {
    for (let i=pos; i<n-1; i++) {
        arr[i] = arr[i+1];
    }
    arr.length--;
    n--;
}
```



# Các kỹ thuật thao tác mảng một chiều (cơ bản)

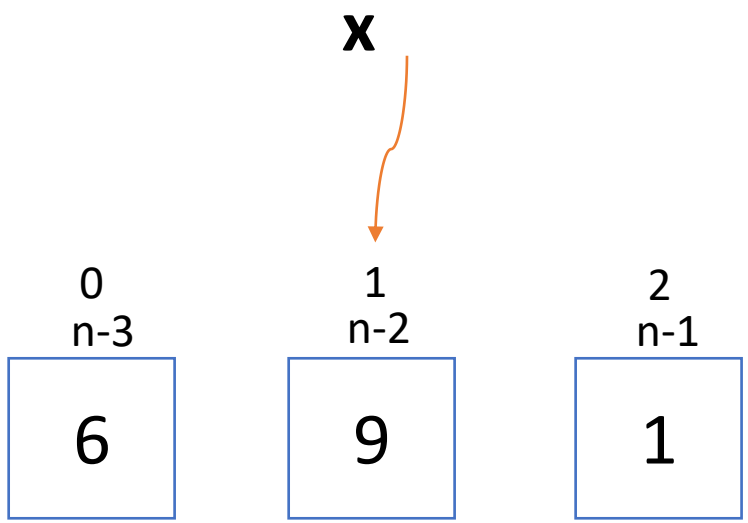
- Lấy phần tử ở đầu/cuối/vị trí x của mảng:

```
let firstElement = arr[0];
let lastElement = arr[arr.length-1];
let element9Slice = arr.slice(1, 2); // arr.slice(-2, -1);
let positionXElement= arr[x];
```




- Lấy vị trí phần tử x của mảng:

```
function getIndexOfXElement(x) {
        for (let i=0; i<n; i++) {
                if (arr[i] == x) return i;
        }
        return -1;
}
```



# Các thuật toán sắp xếp trên mảng một chiều

- Kỹ thuật sắp xếp mảng một chiều:
  - Có **2 kỹ thuật sắp xếp** mảng một chiều:
    - Mảng **tăng dần** ↗
    - Mảng **giảm dần** ↘
  - Một số **thuật toán phổ biến** hiện thực 2 kỹ thuật trên:
    - **Interchange Sort** (Sắp xếp đổi chỗ trực tiếp)
    - **Bubble Sort** (Sắp xếp nổi bọt bong bóng)
    - **Insertion Sort** (Sắp xếp chèn trực tiếp)

# Các thuật toán sắp xếp trên mảng một chiều

- Mảng tăng dần ↗, mảng giảm dần ↘
  - Mảng tăng dần là **tập hợp các phần tử có giá trị tăng dần tính từ chỉ số 0 đến chỉ số n-1 trong mảng** (**n**: số lượng phần tử trong mảng)
  - Mảng giảm dần là **tập hợp các phần tử có giá trị giảm dần tính từ chỉ số n-1 đến chỉ số 0 trong mảng** (**n**: số lượng phần tử trong mảng)
- Ví dụ mảng có 6 phần tử:

| 24 | 45 | 23 | 13 | 43 | -1 |
|---|---|---|---|---|---|

- Mảng được sắp xếp **tăng dần**/**giảm dần** như sau:

| -1 | 13 | 23 | 24 | 43 | 45 |
|---|---|---|---|---|---|

| 45 | 43 | 24 | 23 | 13 | -1 |
|---|---|---|---|---|---|

# Các thuật toán sắp xếp trên mảng một chiều

❑ **Nghịch thế**:

➢ Một **cặp giá trị (a, b)** được gọi là **nghịch thế** khi **a và b không thỏa mãn điều kiện sắp xếp thứ tự**.

➢ Ví dụ mảng 6 phần tử (**Điều kiện**: mảng tăng dần)


➢ Các **cặp nghịch thế** trong mảng trên là:

(24, 23) (24, 13) (24, -1) (**so sánh 24 với các phần tử còn lại sau 24**)

(45, 23) (45, 13) (45, 43) (45, -1) (**so sánh 45 với các phần tử còn lại sau 45**)

(23, 13) (23, -1) (**so sánh 23 với các phần tử còn lại sau 23**)

(13, -1) (**so sánh 13 với các phần tử còn lại sau 13**)

# Các thuật toán sắp xếp trên mảng một chiều

❑ **Interchange Sort** (sắp xếp đổi chỗ trực tiếp):

➢ **Ý tưởng thuật toán:** Duyệt qua tất cả các cặp giá trị trong mảng và hoán vị 2 giá trị trong 1 cặp nếu giá trị đó là nghịch thế.

# Các thuật toán sắp xếp trên mảng một chiều

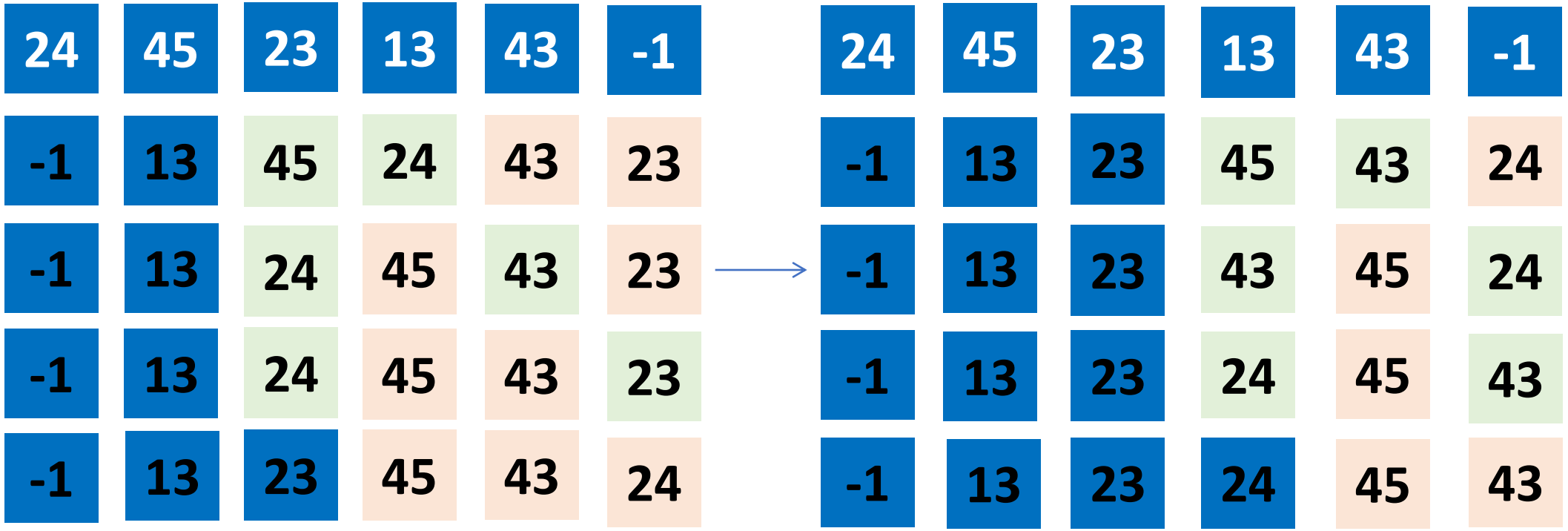
| 24 | 45 | 23 | 13 | 43 | -1 |
|---|---|---|---|---|---|
| 24 | 45 | 23 | 13 | 43 | -1 |
| 24 | 45 | 23 | 13 | 43 | -1 |
| 23 | 45 | 24 | 13 | 43 | -1 |
| 13 | 45 | 24 | 23 | 43 | -1 |
| 13 | 45 | 24 | 23 | 43 | -1 |
| -1 | 45 | 24 | 23 | 43 | 13 |

Bước 1

| 24 | 45 | 23 | 13 | 43 | -1 |
|---|---|---|---|---|---|
| -1 | 45 | 24 | 23 | 43 | 13 |
| -1 | 24 | 45 | 23 | 43 | 13 |
| -1 | 23 | 45 | 24 | 43 | 13 |
| -1 | 23 | 45 | 24 | 43 | 13 |
| -1 | 13 | 45 | 24 | 43 | 23 |

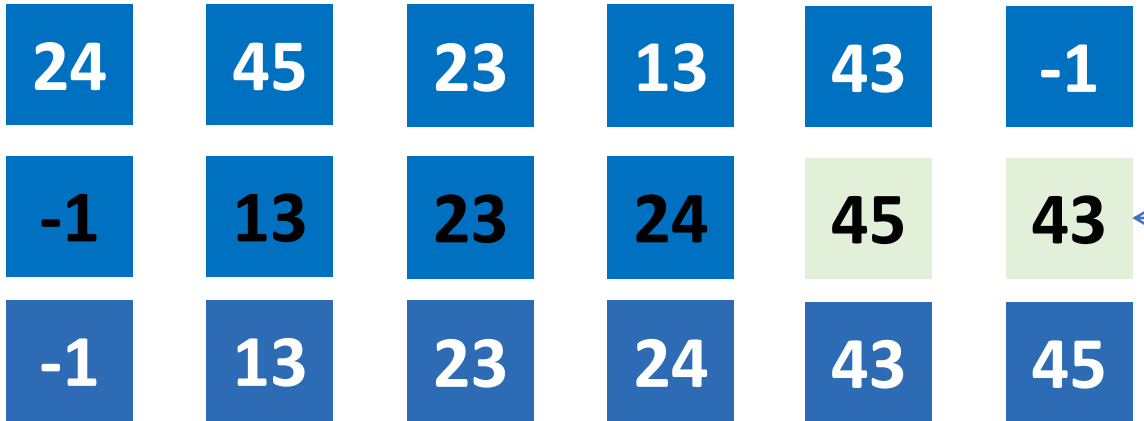
Bước 2

# Các thuật toán sắp xếp trên mảng một chiều




Bước 3

Bước 4




Bước 5

# Các thuật toán sắp xếp trên mảng một chiều

- **Interchange Sort** (sắp xếp đổi chỗ trực tiếp):
  - Cài đặt thuật toán:
    - Mảng tăng dần ↗

```
// Ascending Array (mảng tăng dần)
function sortAscending(n, arr) {
    for (let i = 0; i <= n - 2; i++) {
        for (let j = i + 1; j <= n - 1; j++) {
            if (arr[i] > arr[j]) {
                let temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```

    - Mảng giảm dần ↘

```
// Descending Array (mảng tăng dần)
function sortDescending(n, arr) {
    for (let i = 0; i <= n - 2; i++) {
        for (let j = i + 1; j <= n - 1; j++) {
            if (arr[i] < arr[j]) {
                let temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```

# Các thuật toán sắp xếp trên mảng một chiều

❑ **Bubble Sort** (sắp xếp nổi bọt bong bóng):

➢ **Ý tưởng thuật toán:** Nhẹ nổi lên và nặng chìm xuống (nhẹ: nhỏ hơn, nặng: lớn hơn).

# Các thuật toán sắp xếp trên mảng một chiều

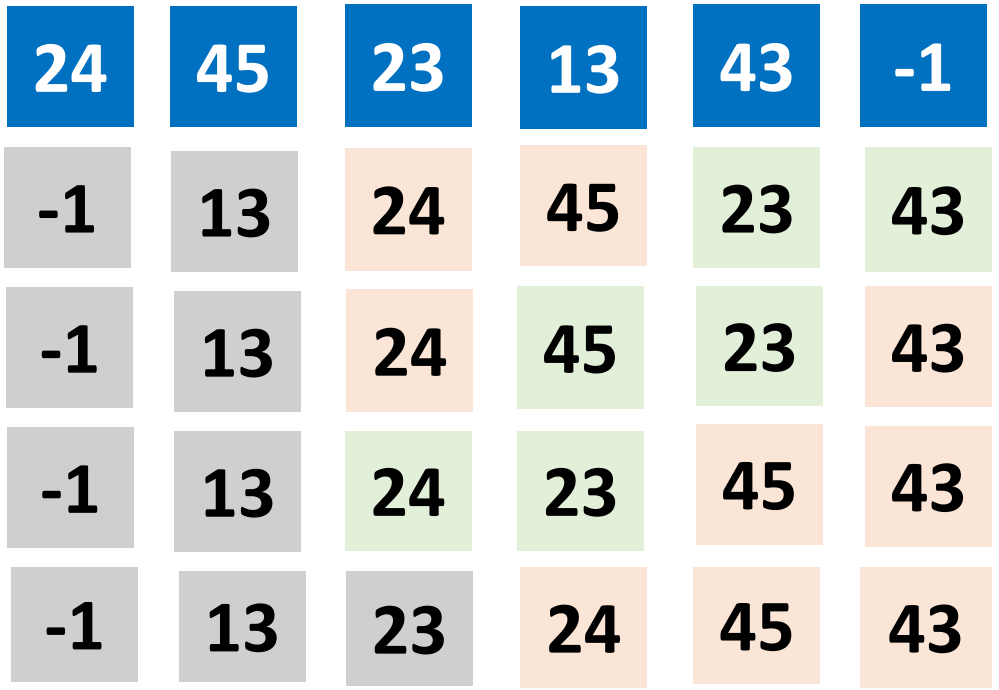
| 24 | 45 | 23 | 13 | 43 | -1 |
|---|---|---|---|---|---|
| 24 | 45 | 23 | 13 | 43 | -1 |
| 24 | 45 | 23 | 13 | -1 | 43 |
| 24 | 45 | 23 | -1 | 13 | 43 |
| 24 | 45 | -1 | 23 | 13 | 43 |
| 24 | -1 | 45 | 23 | 13 | 43 |
| -1 | 24 | 45 | 23 | 13 | 43 |

Bước 1

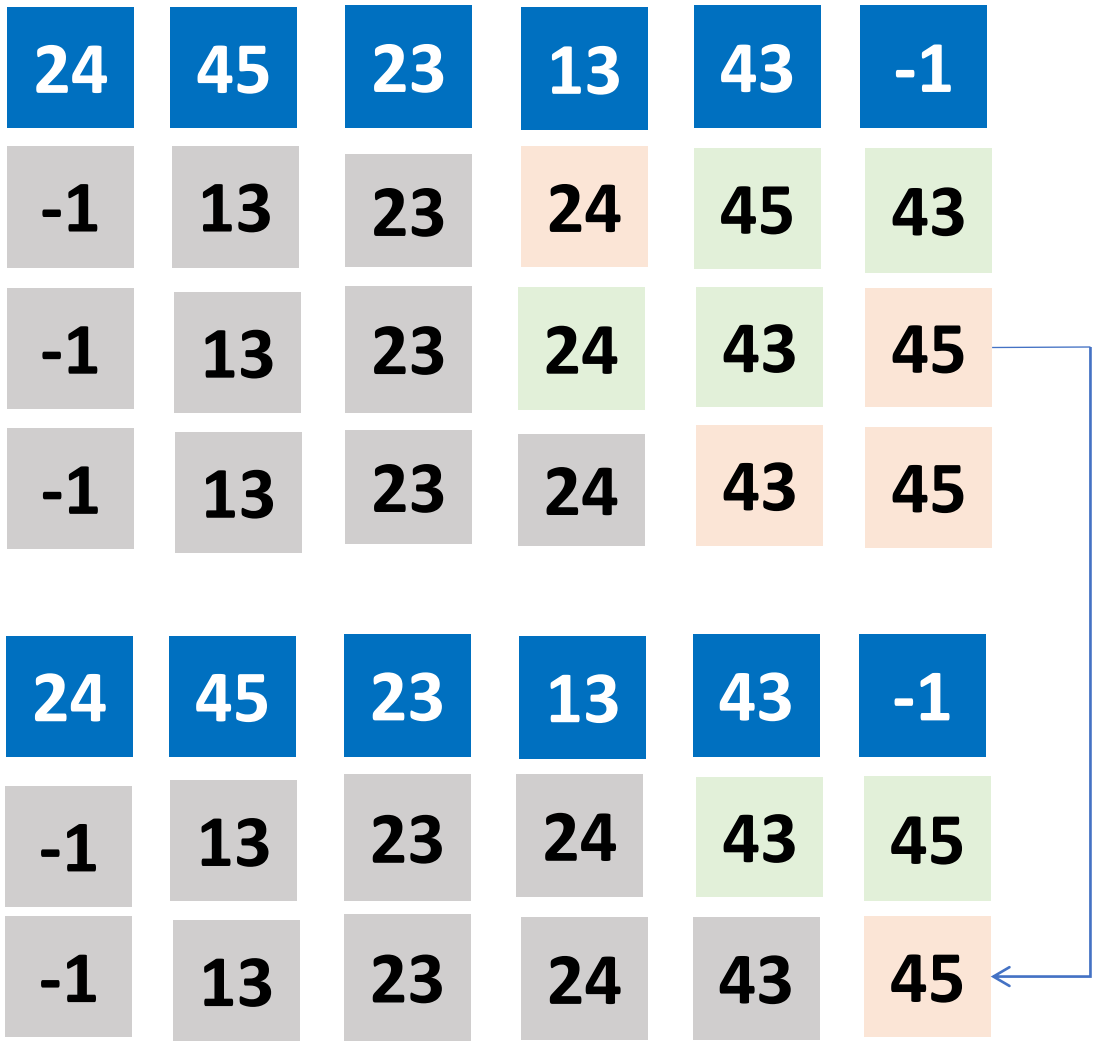
| 24 | 45 | 23 | 13 | 43 | -1 |
|---|---|---|---|---|---|
| -1 | 24 | 45 | 23 | 13 | 43 |
| -1 | 24 | 45 | 23 | 13 | 43 |
| -1 | 24 | 45 | 13 | 23 | 43 |
| -1 | 24 | 13 | 45 | 23 | 43 |
| -1 | 13 | 24 | 45 | 23 | 43 |

Bước 2

# Các thuật toán sắp xếp trên mảng một chiều




Bước 3



# Các thuật toán sắp xếp trên mảng một chiều

- **Bubble Sort** (sắp xếp nổi bọt bong bóng):
  - Cài đặt thuật toán:
    - Mảng tăng dần ↗

```
// Ascending Array (mảng tăng dần)
function sortAscending(n, arr) {
    for (let i = 0; i <= n - 2; i++) {
        for (let j = n - 1; j >= i + 1; j--) {
            if (arr[j] < arr[j - 1]) {
                let temp = arr[j];
                arr[j] = arr[j - 1];
                arr[j - 1] = temp;
            }
        }
    }
}
```

    - Mảng giảm dần ↘

```
// Descending Array (mảng tăng dần)
function sortDescending(n, arr) {
    for (let i = 0; i <= n - 2; i++) {
        for (let j = n - 1; j >= i + 1; j--) {
            if (arr[j] > arr[j - 1]) {
                let temp = arr[j];
                arr[j] = arr[j - 1];
                arr[j - 1] = temp;
            }
        }
    }
}
```

# Các thuật toán sắp xếp trên mảng một chiều

❑ **Insertion Sort** (sắp xếp chèn trực tiếp):

➢ **Ý tưởng thuật toán:**

▪ **Bước 1**: Mảng a có 1 phần tử được sắp tăng, chèn a[1] vào mảng a để được mảng có 2 phần tử sắp tăng.

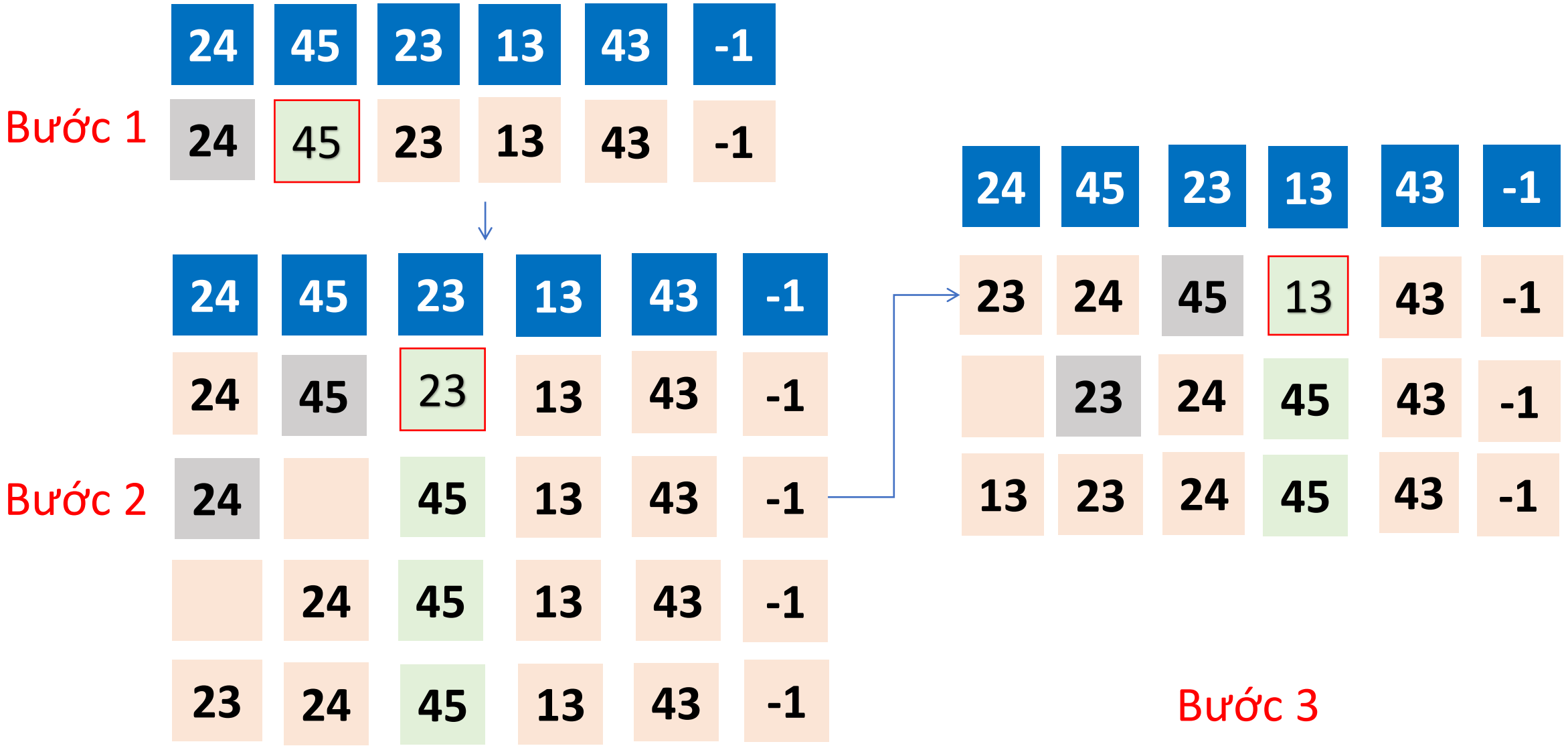
▪ **Bước 2**: Mảng a có 2 phần tử được sắp tăng, chèn a[2] vào mảng a để được mảng có 3 phần tử sắp tăng.

▪ ……..

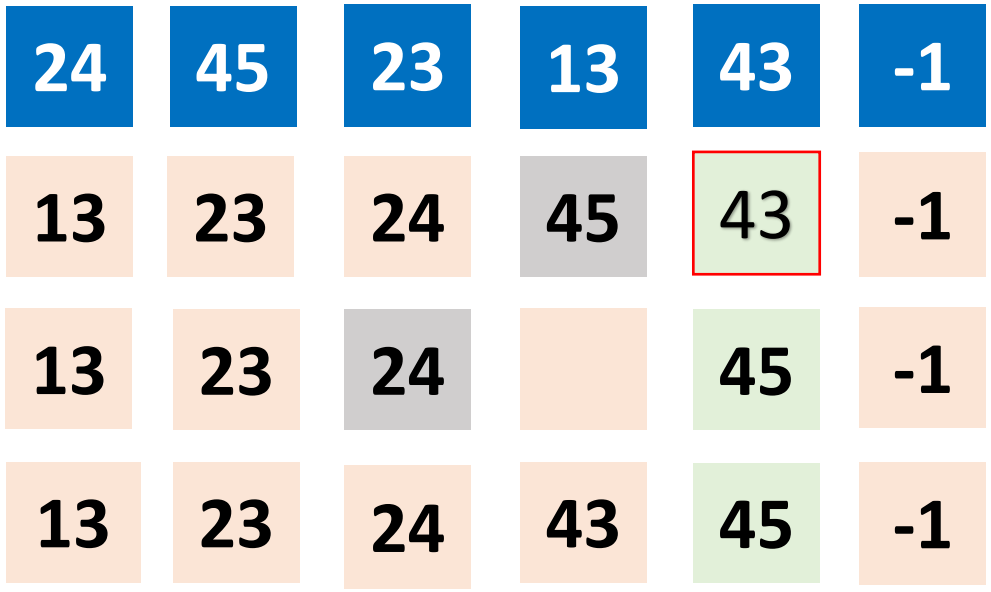
▪ **Bước i**: Mảng a có i phần tử được sắp tăng, chèn a[i] vào mảng a để được mảng có [i+1] phần tử sắp tăng.

▪ **Bước n-1**: Mảng a có n-1 phần tử được sắp tăng, chèn a[n-1] vào mảng a để được mảng có n phần tử sắp tăng

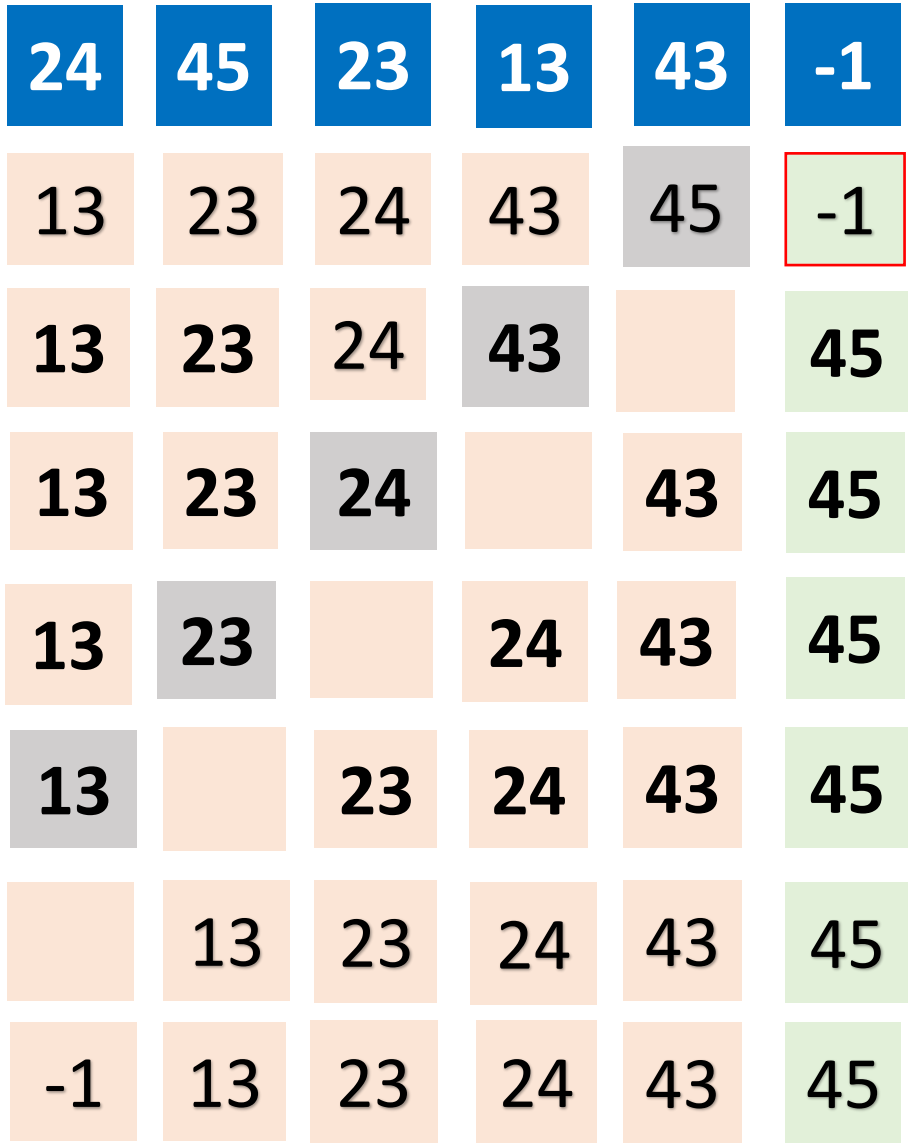
# Các thuật toán sắp xếp trên mảng một chiều



# Các thuật toán sắp xếp trên mảng một chiều




Bước 4




Bước 5

# Các thuật toán sắp xếp trên mảng một chiều

- **Insertion Sort** (sắp xếp chèn trực tiếp):
  - Cài đặt thuật toán:
    - Mảng tăng dần ↗

```
// Ascending Array (mảng tăng dần)
function sortAscending(n, arr) {
    for (let i = 1; i < n; i++) {
        var x = arr[i];
        var j;

        for (j = i - 1; (j >= 0 && arr[j] > x); j--) {
            arr[j + 1] = arr[j];
        }
        arr[j + 1] = x;
    }
}
```

    - Mảng giảm dần ↘

```
// Descending Array (mảng tăng dần)
function sortAscending(n, arr) {
    for (let i = 1; i < n; i++) {
        var x = arr[i];
        var j;

        for (j = i - 1; (j >= 0 && arr[j] < x); j--) {
            arr[j + 1] = arr[j];
        }
        arr[j + 1] = x;
    }
}
```

# Các kỹ thuật thao tác mảng một chiều (phổ biến)

❑ **Kỹ thuật liệt kê**:

➢ Giúp chúng ta **liệt kê** ra **các giá trị phần tử** trong **mảng một chiều**.

```
function listOutOddNumbers() {
    //...
    console.log(arr[i]);
    //...
}
```

❑ **Luyện tập**: Viết chương trình thực hiện yêu cầu sau:

➢ Tạo một mảng số ngẫu nhiên có n phần tử (n được nhập từ bàn phím)

➢ Xuất mảng số ngẫu nhiên mới tạo ra màn hình browser

➢ **Liệt kê** các phần tử lẻ có trong mảng trên.

# Các kỹ thuật thao tác mảng một chiều (phổ biến)

❑ **Kỹ thuật tính tổng**:

➢ Giúp tính tổng của các giá trị trong mảng theo yêu cầu nào đó.

```
function sumOf…() {
    var sum = 0;
    //…

    return sum;
}
```

❑ **Luyện tập**: Định nghĩa hàm tính tổng các giá trị âm trong mảng:

➢ Tạo một mảng số nguyên có n phần tử (lưu ý: nhớ nhập số âm)

➢ Xuất mảng số ngẫu nhiên mới tạo ra màn hình browser

➢ **Tính tổng** các số âm trong mảng số nguyên mới nhập.

➢ Xuất tổng mới tính ra browser.

# Các kỹ thuật thao tác mảng một chiều (phổ biến)

❑ **Kỹ thuật đếm**:

➢ Giúp đếm số lượng của một biến nào đó sau khi thực hiện mảng.

```
function count…() {
    var count = 0;
    //…

    return count;
}
```

❑ **Luyện tập**: Định nghĩa hàm đếm số nguyên chẵn trong mảng với yêu cầu:

➢ Tạo một mảng ngẫu nhiên số nguyên có n phần tử.

➢ Xuất mảng số ngẫu nhiên mới tạo ra màn hình browser

➢ **Đếm số lượng** các số nguyên chẵn trong mảng số nguyên mới nhập.

➢ Xuất tổng số lượng số nguyên chẵn mới đếm ra browser.

# Các kỹ thuật thao tác mảng một chiều (phổ biến)

❑ **Kỹ thuật tìm kiếm** (đặt lính canh):

➢ Giúp tìm kiếm giá trị nào đó theo yêu cầu trong một mảng.

```
function find...() {
    var element = arr[0];
    //...

    return element;
}
```

❑ **Luyện tập**: Định nghĩa hàm tìm số lớn nhất trong mảng với yêu cầu:

➢ Tạo một mảng ngẫu nhiên số nguyên có n phần tử.

➢ Xuất mảng số ngẫu nhiên mới tạo ra màn hình browser

➢ **Tìm số nguyên có giá trị lớn nhất** trong mảng số nguyên mới nhập.

➢ Xuất số lớn nhất mới tìm được ra browser.

# Các kỹ thuật thao tác mảng một chiều (phổ biến)

❑ **Kỹ thuật đặt cờ hiệu**:

➢ Giúp tìm kiếm giá trị nào đó theo yêu cầu trong một mảng.

```
function check...() {
    var flag = false;
    //...

    return flag;
}
```

❑ **Luyện tập**: Định nghĩa hàm kiểm tra số lẻ nhỏ hơn 5 trong mảng với y/c:

➢ Tạo một mảng ngẫu nhiên số nguyên có n phần tử.

➢ Xuất mảng số ngẫu nhiên mới tạo ra màn hình browser

➢ **Kiểm tra có tồn tại số lẻ nhỏ hơn 5** trong mảng mới nhập không.

➢ Xuất kết quả kiểm tra ra browser.

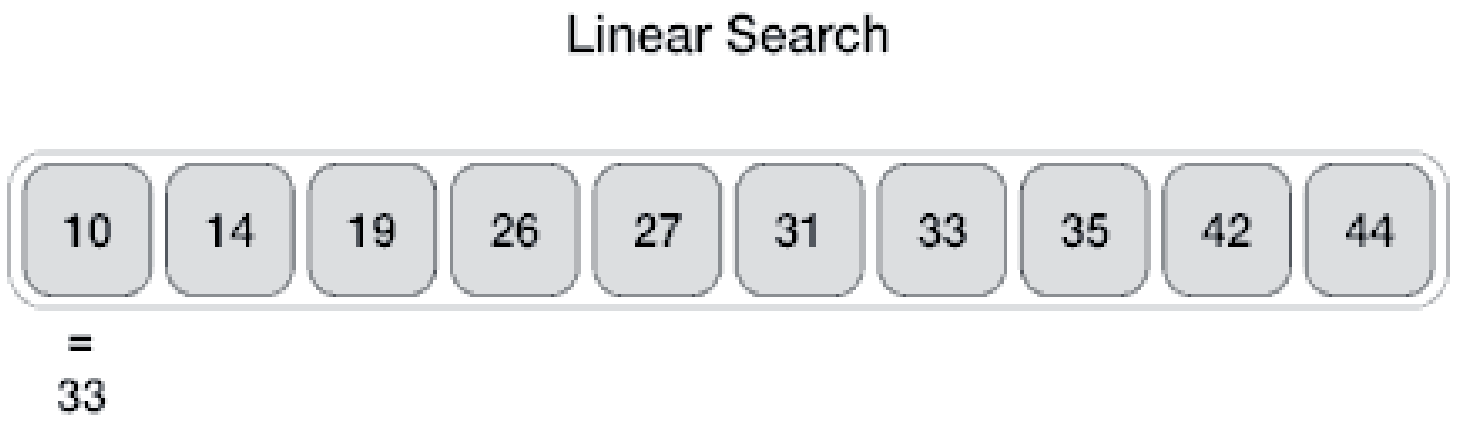
# Phương pháp tìm kiếm tuyến tính (Linear Search)

❑ **Giới thiệu:**

➢ **Tìm kiếm tuyến tính** là **một thuật toán tìm kiếm** rất **đơn giản**.

➢ **Việc tìm kiếm** sẽ **được thực hiện tuần tự** trên tất cả các phần tử.

➢ **Mỗi phần tử được kiểm tra** và **so khớp với giá trị cần tìm**. Nếu khớp thì sẽ trả về kết quả. Ngược lại, nếu không khớp thì việc tìm kiếm sẽ tiếp tục cho đến khi kết thúc dữ liệu đã thu thập.

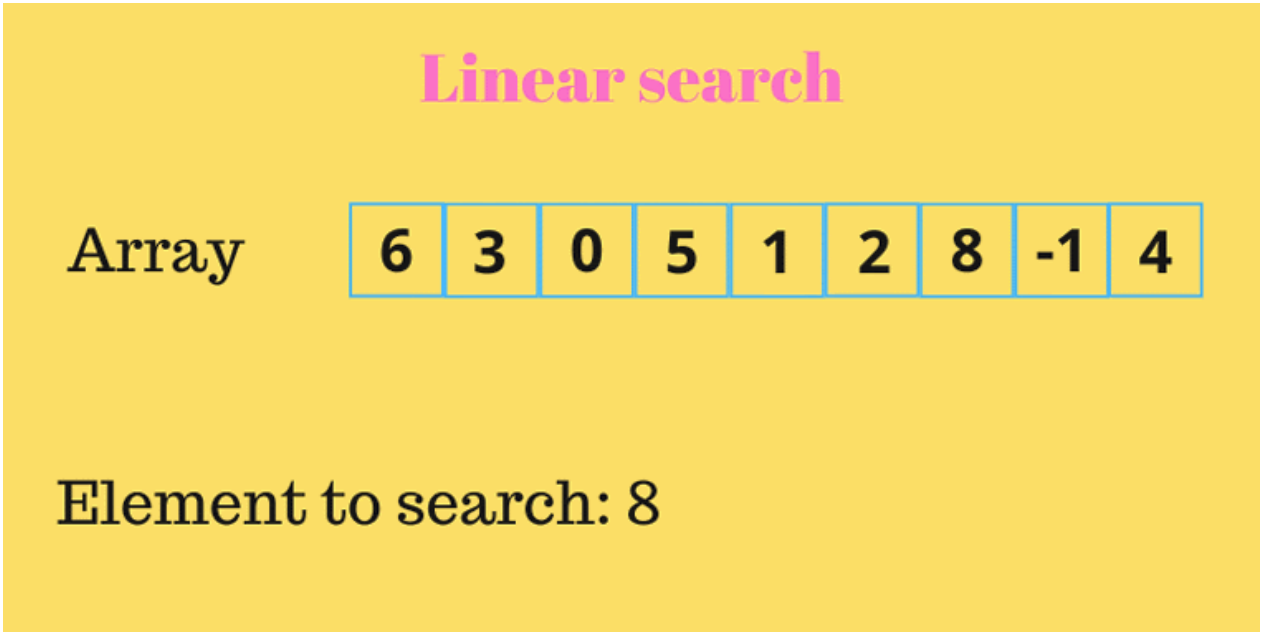


# Phương pháp tìm kiếm tuyến tính (Linear Search)

❑ **Cài đặt thuật toán:**

```
var arr = []; // 10, 14, 19, 26, 27, 31, 33, 35, 42, 44
var n; // số lượng phần tử trong mảng

function linearSearch(x) {
    for (let i = 0; i < n; i++) {
        if (arr[i] == x)
            return arr[i];
    }
}
```



# Các kỹ thuật thao tác mảng một chiều (phổ biến)

- **Luyện tập**: Định nghĩa hàm tìm số nguyên chẵn nhỏ hơn 10 trong mảng với các yêu cầu sau:
  - Tạo một mảng ngẫu nhiên số nguyên có n phần tử.
  - Xuất mảng số ngẫu nhiên mới tạo ra màn hình browser
  - **Tìm kiếm số nguyên là số chẵn và nhỏ hơn 10** trong mảng các số nguyên mới nhập. (sử dụng Linear Search)
  - Xuất kết quả tìm kiếm ra browser

# Các kỹ thuật thao tác mảng một chiều (phổ biến)

❑ **Cài đặt**:

```
function findMaxEvenLessThan10(n, arr) {
    let element = arr[0];

    for (let i = 0; i < n; i++) {
        if (arr[i] > element && arr[i] % 2 == 0 && arr[i] < 10) {
            element = arr[i];
        }
    }

    return element;
}
```

# Công cụ quản lý các thư viện lập trình (npm)

❑ **Giới thiệu**:

➢ **npm** (Node package manager) là một **công cụ tạo** và **quản lý các thư viện lập trình JavaScript cho Node.js**. (có sẵn khi cài Node.js)

➢ Trong cộng đồng JavaScript, các lập trình viên chia sẻ hàng trăm nghìn các thư viện với các đoạn code đã thực hiện sẵn chức năng nào đó.

# Công cụ quản lý các thư viện lập trình (npm)

❑ **Sử dụng**:

➢ Để cài đặt một thư viện nào đó, ta chỉ cần mở cửa sổ Terminal (hoặc CMD) và thực hiện lệnh cấu trúc sau: **npm install package-name**

➢ Có 2 thư viện có thể sử dụng trong môn kỹ thuật lập trình để thay thế việc xen lẫn với mã html là: **prompt**, **alert**.

# Công cụ quản lý các thư viện lập trình (npm)

- **Sử dụng:**
  - Để cài đặt một thư viện **alert**: **npm install js-alert**
  - Để cài đặt một thư viện **prompt**: **npm install prompt-sync**
  - Thực hiện ví dụ để kiểm tra các thư viện đã cài đặt.

# Tổng kết:

- ❑ Các kỹ thuật thao tác với mảng 1 chiều (cơ bản)
- ❑ Các thuật toán sắp xếp trên mảng 1 chiều
- ❑ Các kỹ thuật thao tác với mảng một chiều (phổ biến)
- ❑ Phương pháp tìm kiếm tuyến tính (Linear Search)
- ❑ Công cụ quản lý các thư viện lập trình (npm)