
# CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT
# DATA STRUCTURE AND ALGORITHM

# CHƯƠNG 2. CÁC GIẢI THUẬT TÌM KIẾM & SẮP XẾP (SEARCH & SORT)



*Created by Bích Ngân & Vũ Văn Vinh*
*(nganntb@hufi.edu.vn & vinhvv@hufi.edu.vn)*

# NỘI DUNG

I. Vai trò của CTDL & GT trong lập trình

II. Các giải thuật tìm kiếm

III. Các giải thuật sắp xếp



*Created by Bích Ngân & Vũ Văn Vinh*
*(nganntb@hufi.edu.vn & vinhvv@hufi.edu.vn)*

# I. Vai trò của CTDL & GT trong lập trình

# 1.1. Vai trò của CTDL trong Tin học

- **Ví dụ:**

  Một lớp học có 5 hs, mỗi hs dự thi 3 môn toán, lý, hóa. Giả sử đã có dữ liệu điểm của các hs. Hãy viết chương trình xuất điểm các hs thành 1 bảng có 5 dòng, 3 cột.

- **Sinh viên hãy đề xuất các cách giải có thể có cho bài toán.**

}

# CÁC HƯỚNG GIẢI BÀI TOÁN

## Cách 1: dùng mảng 1 chiều

```
void  main()
{
    // giả sử đã nhập điểm cho các hs.
    int diem[15] =   {4,6,2,7,8,9,4,6,9,10,9,4,6,4,8};

     // phần xuất điểm
     for(int i=0; i<15; i++)
     {       cout<<diem[i]<<"\t";
        if(i%3==2)
                    cout<<endl;
     }
}
```

muốn biết điểm môn *k* của hs *n* thì sao?

→ diem [ (n-1)*3+(k-1)]

}

# CÁC HƯỚNG GIẢI BÀI TOÁN

- **Cách 2: dùng mảng 2 chiều có 5 dòng 3 cột**

```
void  main()
{
    // giả sử đã nhập điểm cho các hs.
    int diem[5][3] =   {4,6,2,7,8,9,4,6,9,10,9,4,6,4,8};
   // phần xuất điểm
     for(int i=0; i<5; i++)
     {
            for(int j=0; j<3; j++)
                  cout<<diem[i][j]<<"\t";
           cout<<endl;
     }
}
```

muốn biết điểm môn *k* của hs *n* thì sao?

→ diem [n-1][k-1]

}

# CÁC HƯỚNG GIẢI BÀI TOÁN

- **Cách 3: dùng mảng 1 chiều có kiểu phần tử dạng cấu trúc**

```
struct DiemHS
{
    int toan, ly, hoa;
};
void main()
{
        DiemHS    diem[5];
  //giả sử đã nhập dữ liệu cho mảng diem rồi.
  // xuất điểm
    for(int i=0; i<5;i++)
        cout<<diem[i].toan<<"\t"<< diem[i].ly<<"\t"<<
        diem[i].hoa<<endl;
}
```

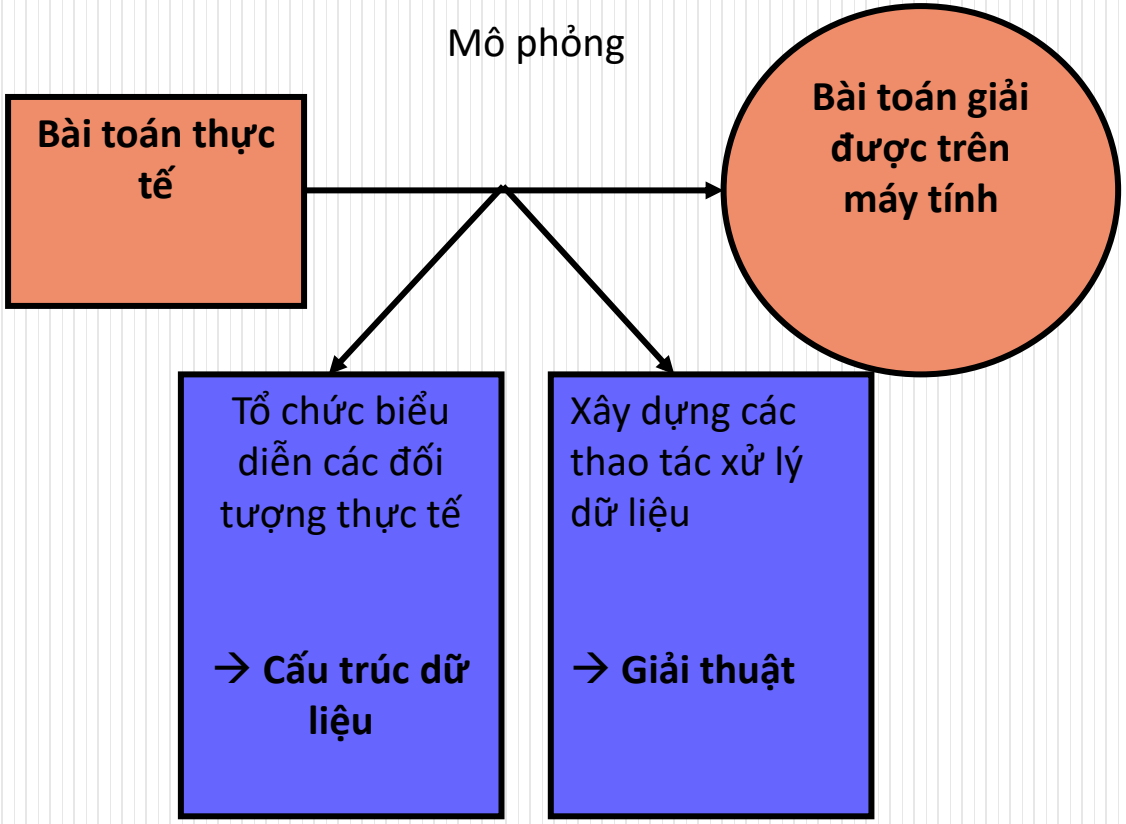
muốn biết điểm môn *toan* của hs *n* thì sao?

→ diem [n-1].toan

# Nhận xét

- Cả 3 phương án đều giải quyết trọn vẹn bài toán. Nhưng phương án 1 biểu diễn dữ liệu không tự nhiên và thao tác trên đối tượng phức tạp.
- Phương án 2 và 3 cung cấp 1 CTDL và các thao tác xử lý hợp lý. Do đó tùy theo trường hợp mà ta chọn phương án 2 hoặc 3.
- Chọn phương án 2 nếu muốn dễ dàng xử lý dữ liệu.
- Chọn phương án 3 nếu muốn nhấn mạnh việc mô phỏng ý nghĩa đối tượng.

# 1.1. Vai trò của CTDL trong lập trình (tt)



# 1.1. Vai trò của CTDL trong lập trình (tt)

- **Tổ chức biểu diễn các đối tượng thực tế.**

  Nghĩa là cần xây dựng cấu trúc thích hợp nhất sao cho phản ánh chính xác các đối tượng thực tế và có thể thao tác xử lý dễ dàng → lựa chọn xây dựng CTDL

- **Xây dựng các thao tác xử lý dữ liệu**

  Từ những yêu cầu thực tế của bài toán, ta cần tìm giải thuật để xác định các thao tác máy tính tác động lên dữ liệu để cho ra kết quả mong muốn.

- **Lưu ý:**

  Trong một số trường hợp ta chọn 1 giải thuật để giải quyết bài toán có thể chưa tối ưu. Nhưng vì nếu dùng tối ưu thì khó thực hiện và chi phí lớn.

# Kết luận

- Hai công việc trên (tổ chức biểu diễn các đối tượng thực tế, xây dựng các thao tác xử lý dữ liệu) không tồn tại độc lập mà có mối quan hệ chặt chẽ với nhau. Giải thuật phản ảnh các phép xử lý, còn đối tượng xử lý của giải thuật là dữ liệu.
- Để xây dựng được giải thuật phù hợp cần phải biết nó tác động lên loại dữ liệu nào, và khi chọn lựa CTDL cũng cần phải hiểu rõ những thao tác nào sẽ tác động đến nó.

# 1.2. Kiểu dữ liệu

- Định nghĩa:

  Một kiểu dữ liệu T được xác định bởi 1 bộ

  **T= <V,O>**

  trong đó:

  - V: tập giá trị hợp lệ mà 1 đối tượng T có thể lưu trữ.
  - O: tập các thao tác xử lý trên đối tương T

- Kiểu dữ liệu gồm các dạng:
  - Các kiểu dữ liệu cơ bản: int, float, char…
  - Các kiểu dữ liệu có cấu trúc cơ bản: mảng, danh sách liên kết
  - Các kiểu dữ liệu có cấu trúc: do người tự định nghĩa (struct, class,…)

# Ví dụ

- **Kiểu dữ liệu Số Nguyên = <V, O>**
  - V= -32768 …32767 // miền giá trị
  - O = {+,-,*,/,%, <, > ==,…} // phép toán trên dữ liệu số nguyên
- **Kiểu dữ liệu cấu trúc DiemHS = <V, O>**
  - V : giá trị điểm các môn
  - O: nhập/xuất dữ liệu, tính trung bình, tìm max, min, sắp xếp, tìm kiếm…

# 1.3. Một số tiêu chí chọn lựa CTDL

- **CTDL phải phản ảnh đúng thực tế**

  Đây là tiêu chí quan trọng nhất quyết định tính đúng đắn của bài toán, cần phải khảo sát kỹ lưỡng, dự trù đầy đủ các tình huống xảy ra.

- **CTDL phải phù hợp với các thao tác xử lý**

  Tiêu chí này giúp cho việc cài đặt thực hiện các yêu cầu bài toán được đơn giản hơn.

- **CTDL phải tiết kiệm tài nguyên của hệ thống**

  Việc sử dụng tiết kiệm tài nguyên sẽ làm tăng tính hiệu quả cũng như số lượng các bài toán cài trên máy.

# 2.1. Định nghĩa, tính chất giải thuật

- **Định nghĩa :**
  - Giải thuật hay thuật toán (Algorithm) là một chương trình được thiết kế để giải quyết bài toán đặt ra.
- **Tính chất giải thuật**
  - Dữ liệu (vào/ra)
  - Tính xác định
  - Tính dừng
  - Tính đúng đắn
  - Tính khả thi.

# 2.2. Tiêu chuẩn đánh giá độ phức tạp của giải thuật

- Sinh viên tham khảo giáo trình CTDL (ĐH KHTN – Trần Hạnh Nhi & Dương Anh Đức), trang 22-30.

# II. CÁC GIẢI THUẬT TÌM KIẾM

*(Các thuật toán minh họa trên mảng 1 chiều chứa dữ liệu là các số nguyên)*

# 2.1. Tìm kiếm tuyến tính (Linear Search)

- **Ý tưởng:**

  Thuật toán tiến hành so sánh x lần lượt với các phần tử thứ 1, thứ 2,… của mảng a cho đến khi gặp phần tử có khóa cần tìm, hoặc đã tìm hết mảng mà không thấy x.

- **Ví dụ:** Cho dãy số sau:

  5 3 6 8 9 1 2

  Tìm phần tử có giá trị x = 9, x= 10

# Minh họa ví dụ

**Xét dãy số A có 7 phần tử:**

| 5 | 3 | 6 | 8 | 9 | 1 | 2 |
|---|---|---|---|---|---|---|

**Tìm x = 9**

| Vị trí trong dãy (i) | 0 | 1 | … | 4 |
|---|---|---|---|---|
| Giá trị a[i] | 5 | 3 | … | 9 |
| Kết quả so sánh a[i] và x | 5!= 9 | 3!=9 | … | 9= = 9 |

**Tìm được x=9 tại vị trí 4.**

**Kết thúc quá trình**

# Minh họa ví dụ

**Xét dãy số A có 7 phần tử:**

**5 3 6 8 9 1 2**

**Tìm x = 10**

| **Vị trí trong dãy (i)** | 0 | 1 | … | 6 | 7 |
|---|---|---|---|---|---|
| **Giá trị a[i]** | 5 | 3 | … | 2 | null |
| **Kết quả so sánh a[i] và x** | 5!= 10 | 3!=10 | … | 2!= 10 | |

**i=7, a[i] không tồn tại. →Không có 10 trong dãy A.**

**Kết thúc quá trình**

# 2.1. Tìm kiếm tuyến tính (Linear Search) (tt)

- **Cài đặt**

```
int LinearSearch (int a[], int n, int x)
{
        for(int i=0;i<n;i++)
            if(a[i]==x)
                    return i;          // trả về vị trí của x trong a
        return -1; // trả về -1 báo là không có x trong a
}
```

# 2.2. Thuật toán tìm kiếm nhị phân (Binary Search)

- **Ý tưởng**
  - Giả sử dãy số a đã có thứ tự tăng.
  - Tại mỗi bước tiến hành so sánh x với phần tử nằm vị trí giữa của dãy tìm kiếm hiện hành, dựa vào kết quả so sánh này để quyết định giới hạn dãy tìm ở bước kế tiếp là nửa trên hay nửa dưới của dãy tìm kiếm hiện hành.
- **Ví dụ:**
  - Cho dãy a có 7 phần tử: 3 4 6 8 9 10 13
  - Tìm x = 10 và x = 2

# Minh họa ví dụ

- Cho dãy số a: 3 4 6 8 9 10 13

tìm x= 10.

| Bước | left | right | Mid = [(left+right)/2] | a[mid] | Kết quả so sánh x và a[mid] |
|---|---|---|---|---|---|
| 1 | 0 | 6 | [(0+6)/2] = 3 | 8 | 8<10 |
| 2 | Left = mid +1 = 3+1 = 4 | 6 | [(4+6)/2] = 5 | 10 | 10 = = 10 |

→ **Tìm được x=10 tại vị trí 5.**

**Kết thúc quá trình**

# Minh họa ví dụ

- Cho dãy số a: -1 0 6 8 9 10 13

tìm x= 2.

| Bước | left | right | Mid = [(left+right)/2] | a[mid] | Kết quả so sánh x và a[mid] |
|---|---|---|---|---|---|
| 1 | 0 | 6 | [(0+6)/2] = 3 | 8 | 8>2 |
| 2 | 0 | Right = mid – 1 = 3-1 = 2 | [(0+2)/2] = 1 | 0 | 0 < 2 |
| 3 | Left = mid +1 = 1+1 = 2 | 2 | 2 | 6 | 6>2 |
| 4 | 2 | Mid-1 = 2-1=1 | ? | | |

→ **Tại bước 4 có left>right -> vô lý. Kết luận không có x = 2 trong a**

# 2.2. Tìm kiếm nhị phân (Binary Search) (tt)

- **Cài đặt**

```
int BinarySearch (int a[], int n, int x)
{
      int left = 0, right = n-1;
    while(left<=right)
    {
        int mid = (left + right)/2;
        if(a[mid] == x)
                return mid;
        else
                if( a[mid]<x)          left = mid+1;
                else                   right = mid-1;
    }
    return -1;
}
```

# Bài tập

- Cho dãy số sau:

7 9 13 17 27 30 31 35 38 40

a. Tìm x= 17, x=35, x=40

b. Tìm x = 23, x=10, x=36

- Cho dãy ký tự

Z R L K H F E C A

a. Tìm x = R, x = C

b. Tìm x = D, x = Q

# Bài tập

Cho mảng 1 chiều a chứa n số nguyên. Viết chương trình thực hiện các yêu cầu sau:

1. Viết hàm nhập/xuất mảng a.
2. Tìm max/min của a.
3. Đếm số phần tử chẵn/lẻ trong a.
4. Tìm kiếm phần tử x trong a theo 2 dạng ( trả về vị trí/xuất câu thông báo) với giải thuật tìm kiếm tuyến tính/ tìm kiếm nhị phân.
5. Tìm trên a có bao nhiêu phần tử x.

# Bài tập (tt)

6. Tìm trên a có bao nhiêu phần tử  lớn hơn x.
7. Tính tổng các phần tử của a.
8. Xuất các số nguyên tố trong a.
9. Xuất các số hoàn thiện trong a.
10. Xuất các phần tử ở vị trí chẵn/ vị trí lẻ trong a.
11. Xuất giá trị max/min kèm theo vị trí của giá trị đó trong mảng a.
12. Cho 2 dãy số b có m phần tử, dãy c có n phần tử. Ghép b và c thành dãy a được xếp tăng dần.

# III. CÁC GIẢI THUẬT SẮP XẾP

*(Các thuật toán minh họa sắp xếp dãy không tăng trên mảng 1 chiều chứa dữ liệu là các số nguyên)*

**Sắp xếp là quá trình xử lý một danh sách các phần tử để đặt chúng theo một thứ tự thỏa mãn một tiêu chuẩn nào đó dựa trên nội dung thông tin lưu trữ tại mỗi phần tử.**

**Các phương pháp sắp xếp thông dụng**

2.1. Sắp xếp đổi chỗ trực tiếp – Interchange Sort

2.2. Sắp xếp chọn trực tiếp – Selection Sort

2.3. Sắp xếp nhanh – Quick sort

2.4. Sắp xếp chèn trực tiếp – Insertion Sort (đọc thêm)

2.5. Sắp xếp Nổi bọt – Bubble Sort (đọc thêm)

# 3.1. Sắp xếp đổi chỗ trực tiếp – interchange Sort

- **Khái niệm nghịch thế:**
  - Xét dãy các số a: $a_1$, $a_2$, … $a_n$, với a là dãy không giảm.
    Nếu i<j và $a_i$>$a_j$ thì ta gọi đó là 1 nghịch thế.
  - **Ví dụ**: Cho dãy số a như sau:

    14 5 7 8 3.

Vậy dãy trên trên có các cặp nghịch thế sau: (14, 5); (7, 3); (8, 3)….

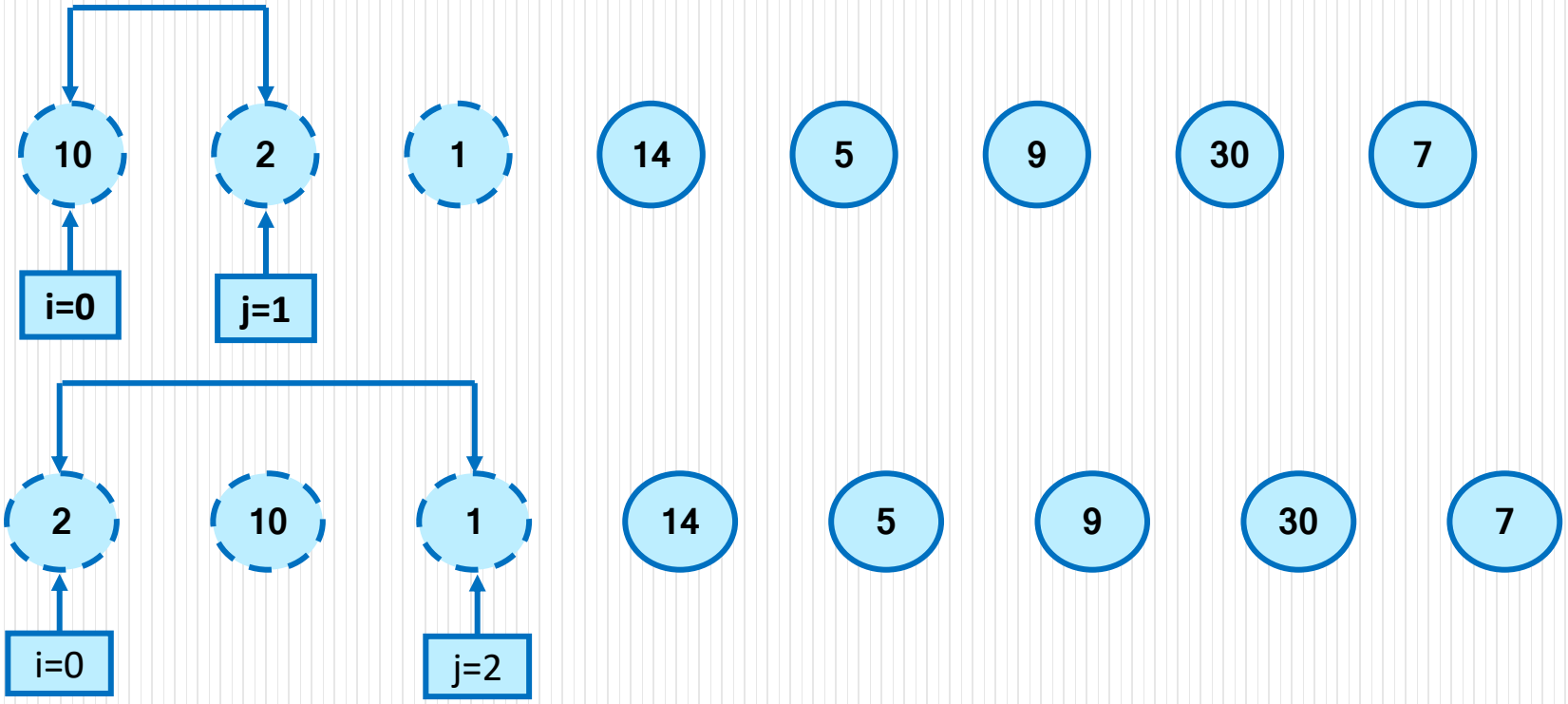
- **Ý tưởng thuật toán:**

Xuất phát từ đầu dãy, lần lượt xét từng phần tử cho đến cuối dãy. Tại mỗi phần tử **tìm tất cả nghịch thế** chứa phần tử này, **đổi chỗ phần tử này với các phần tử trong cặp nghịch thế**.

# Minh họa ví dụ interchange sort

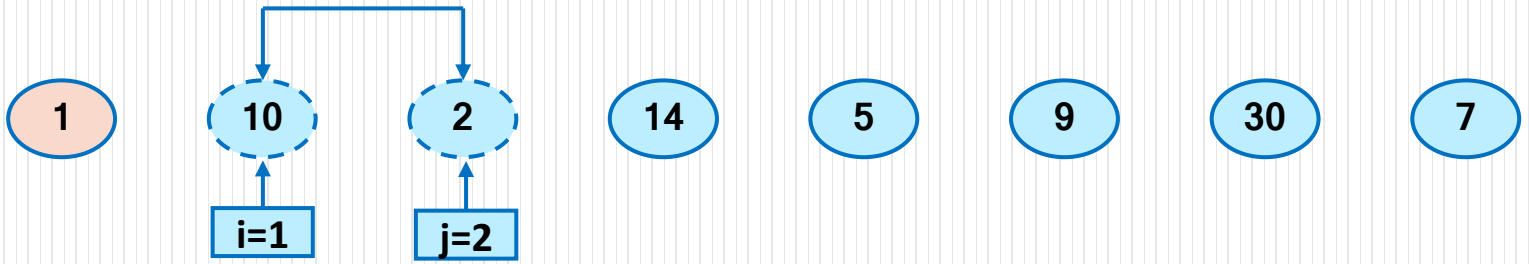
- Cho dãy số a


- Bước 1: xét nghịch thế của phần tử thứ 0 – $a_0$

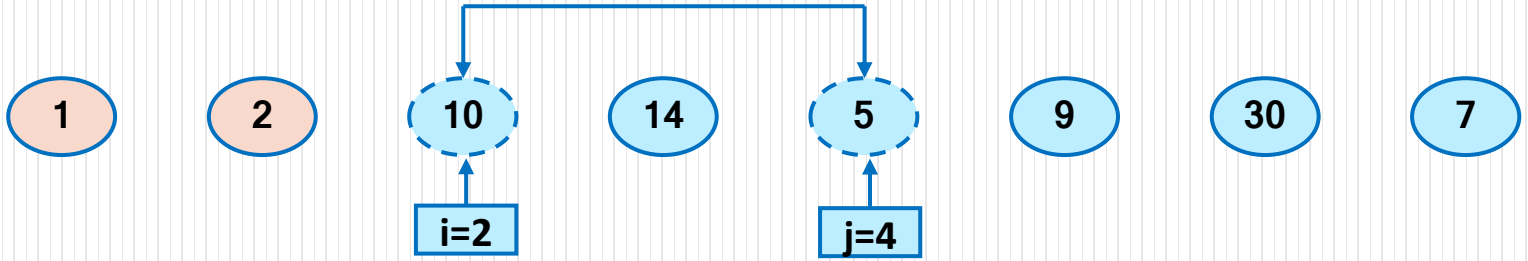


# Minh họa ví dụ interchange sort (tt)

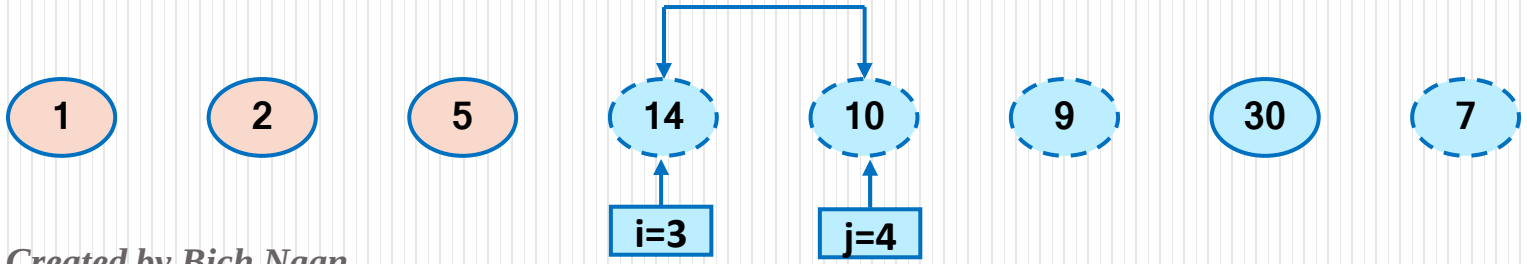
❖ Bước 2: xét nghịch thế của phần tử thứ 1 – $a_1$




❖ Bước 3: xét nghịch thế của phần tử thứ 2 – $a_2$

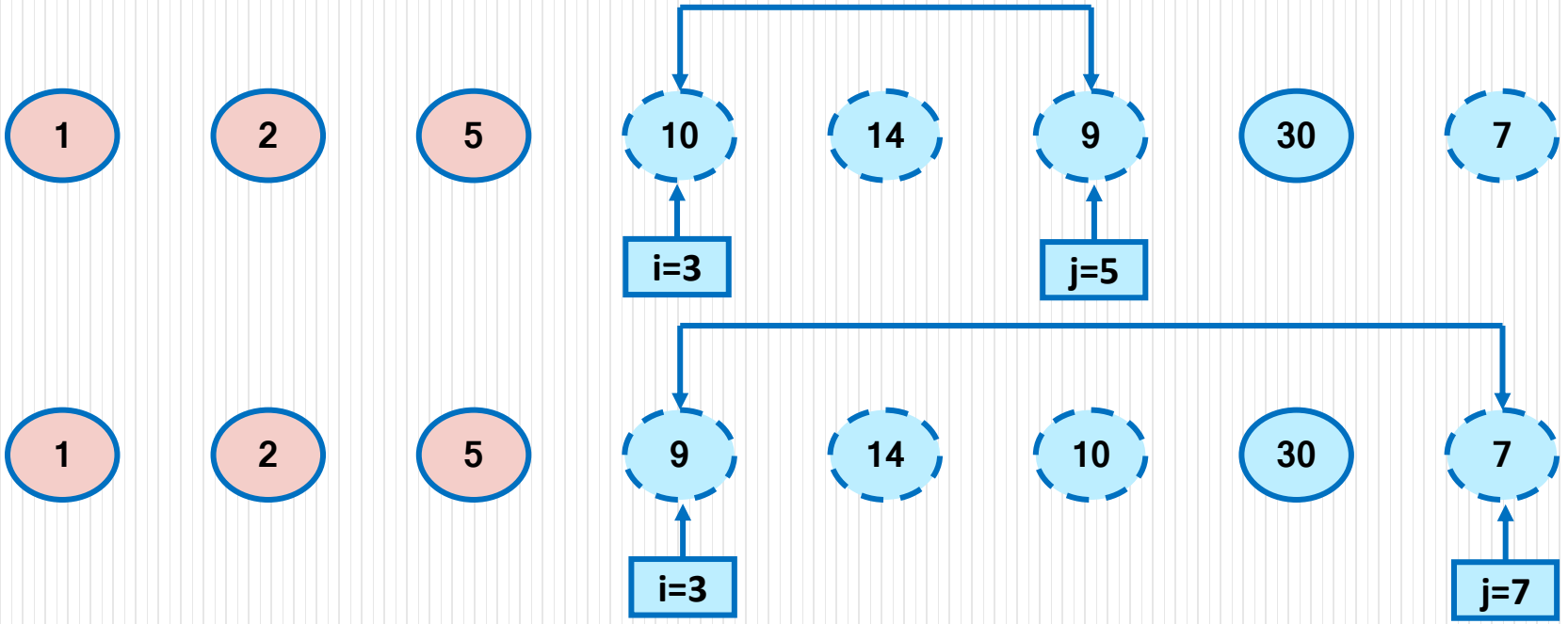



❖ Bước 4: xét nghịch thế của phần tử thứ 3 – $a_3$

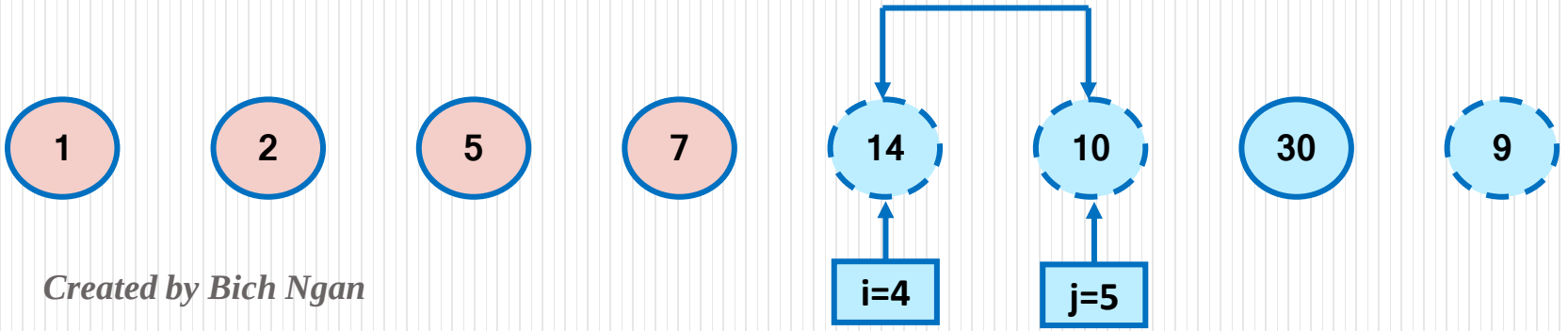


# Minh họa ví dụ interchange sort (tt)

❖ Bước 4: xét nghịch thế của phần tử thứ 3 – $a_3$ tiếp theo

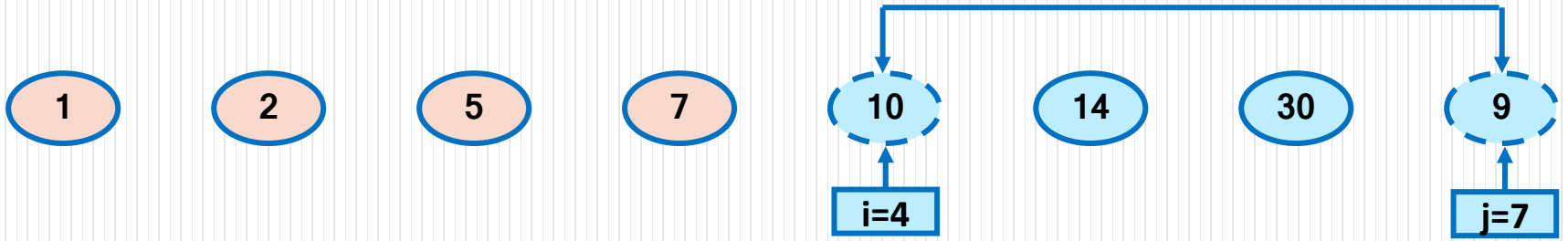



❖ Bước 5: xét nghịch thế của phần tử thứ 4 – $a_4$

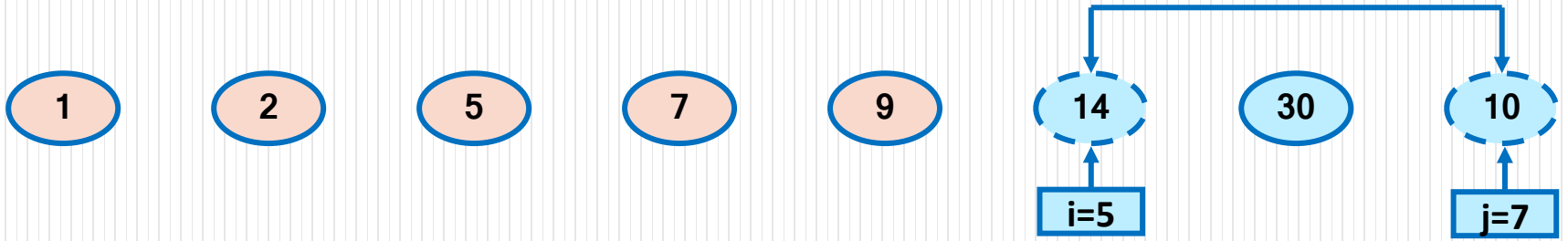


# Minh họa ví dụ interchange sort (tt)

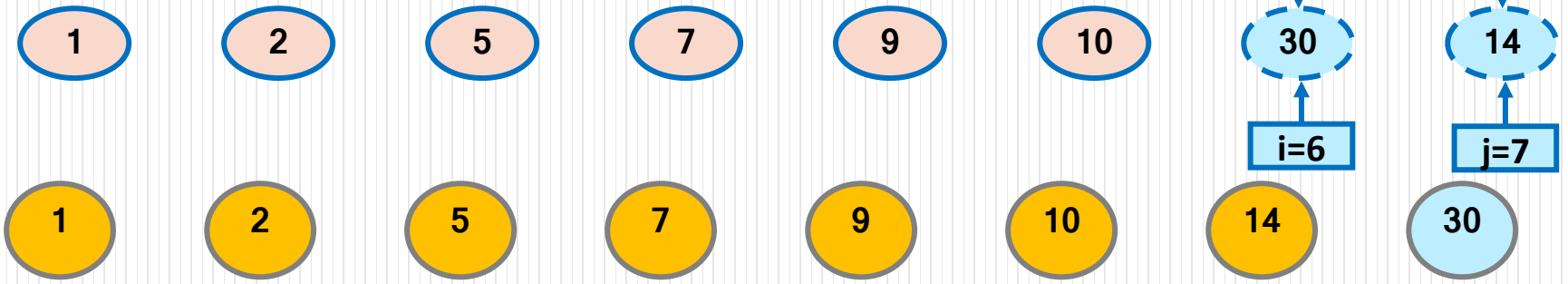
❖ Bước 4: xét nghịch thế của phần tử thứ 4 – $a_4$ tiếp theo




❖ Bước 5: xét nghịch thế của phần tử thứ 5 – $a_5$ tiếp theo




❖ Bước 5: xét nghịch thế của phần tử thứ 6 – $a_6$




Dừng. Vậy dãy đã được sắp xếp.

# Ví dụ

- Minh họa thao tác sắp xếp dữ liệu theo phương pháp interchange Sort cho các dãy dữ liệu sau:
  - Sắp xếp tăng:
    - **13 8 12 6 9 10 12 7**
    - **A H K R E C Z G**
  - Sắp xếp giảm
    - **13 8 12 6 9 10 12 7**
    - **A H K R E C Z G**

# Cài đặt

```
void Interchangesort(int a[],int n)
{
	for(int i=0;i<n-1;i++)
		for(int j=i+1;j<n;j++)
			if(a[i]>a[j])
			swap(a[i],a[j]); //ham hoan doi gia
                                         //tri 2 so nguyen
}
```

```
void swap(int &x, int &y)
{
	int t = x;
	x = y;
	y = t;
}
```

# Bài tập cài đặt

- Viết bổ sung các hàm vào chương trình xử lý mảng 1 chiều các hàm thực hiện những yêu cầu sau:
  1. Viết hàm sắp xếp tăng theo PP **interchange sort** cho dữ liệu số nguyên/số thực/ký tự/ chuỗi ký tự.
  2. Viết hàm sắp xếp tăng theo PP interchange sort cho dữ liệu số nguyên/số thực/ký tự/ chuỗi ký tự.

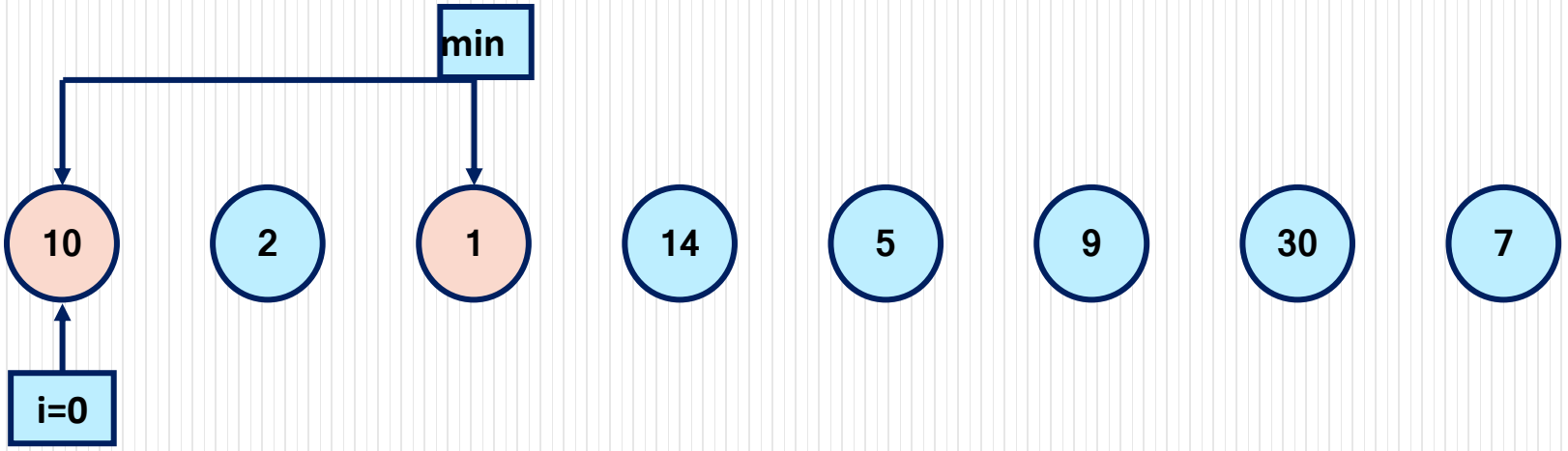
# 3.2. Sắp xếp chọn trực tiếp – Selection sort

- **Ý tưởng giải thuật**
  - Chọn phần tử nhỏ nhất trong n phần tử ban đầu, đưa phần tử này về vị trí thứ 0 của dãy hiện hành; sau đó không quan tâm đến nó nữa, xem dãy hiện hành chỉ còn (n – 1) phần tử, bắt đầu từ vị trí thứ 1; lặp lại quá trình đó trên dãy hiện hành… đến khi dãy hiện hành chỉ còn 1 phần tử thì dừng

# Minh họa ví dụ Selection sort

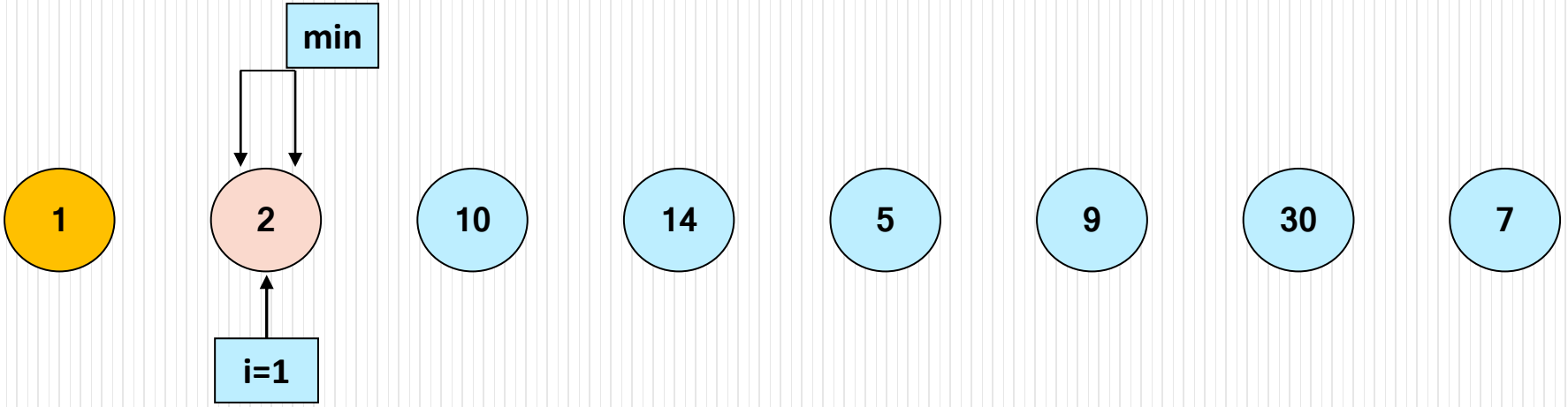
- Cho dãy số a


- Bước 1: Tìm min của dãy số từ $a_0 - a_{n-1}$ . Sau đó hoán đổi min với $a_0$

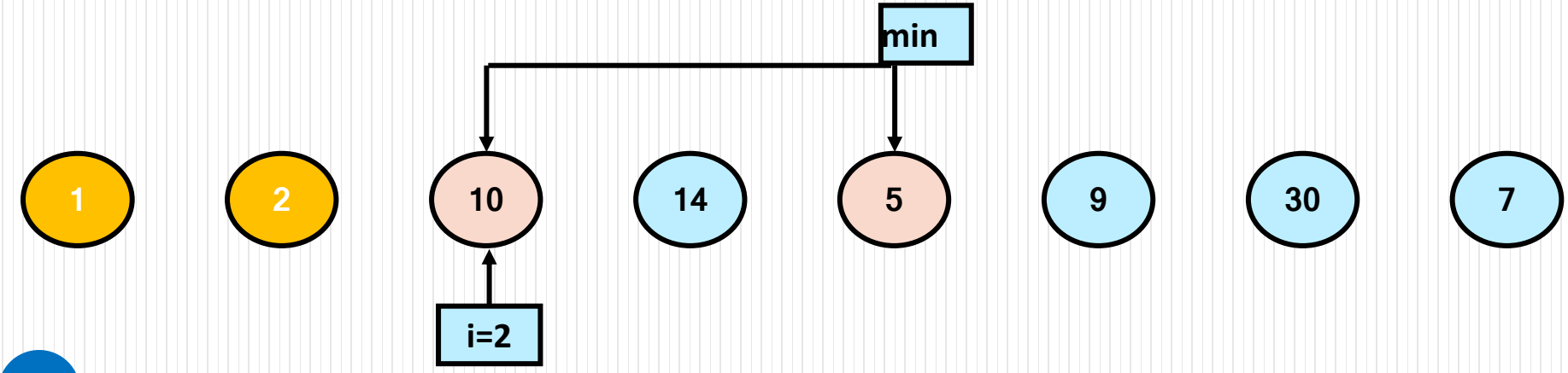


# Minh họa ví dụ Selection sort (tt)

❖ Bước 2: Tìm min của dãy số từ $a_1 – a_{n-1}$ . Sau đó hoán đổi min với $a_1$

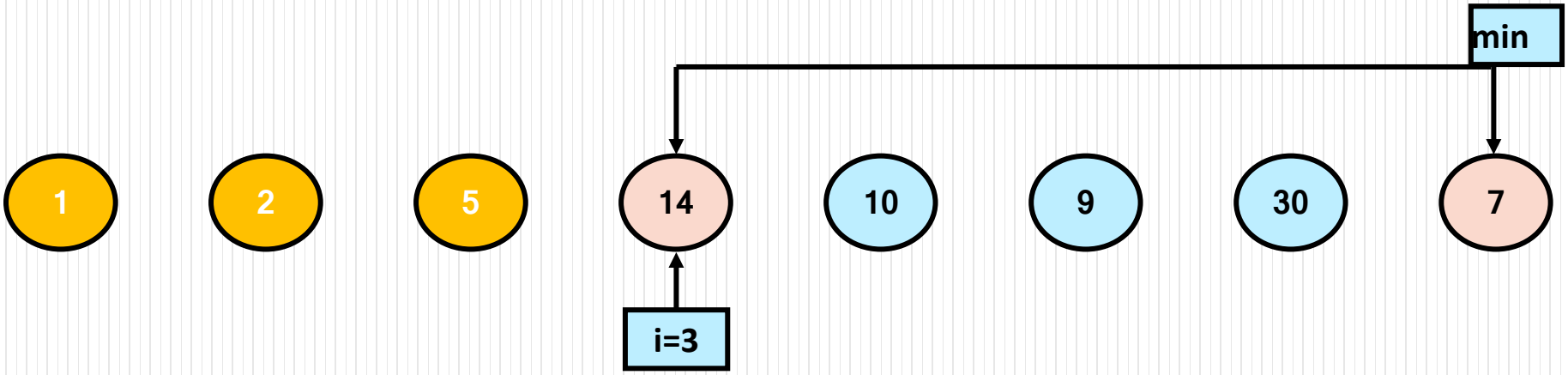



❖ Bước 3: Tìm min của dãy số từ $a_2 – a_{n-1}$ . Sau đó hoán đổi min với $a_2$

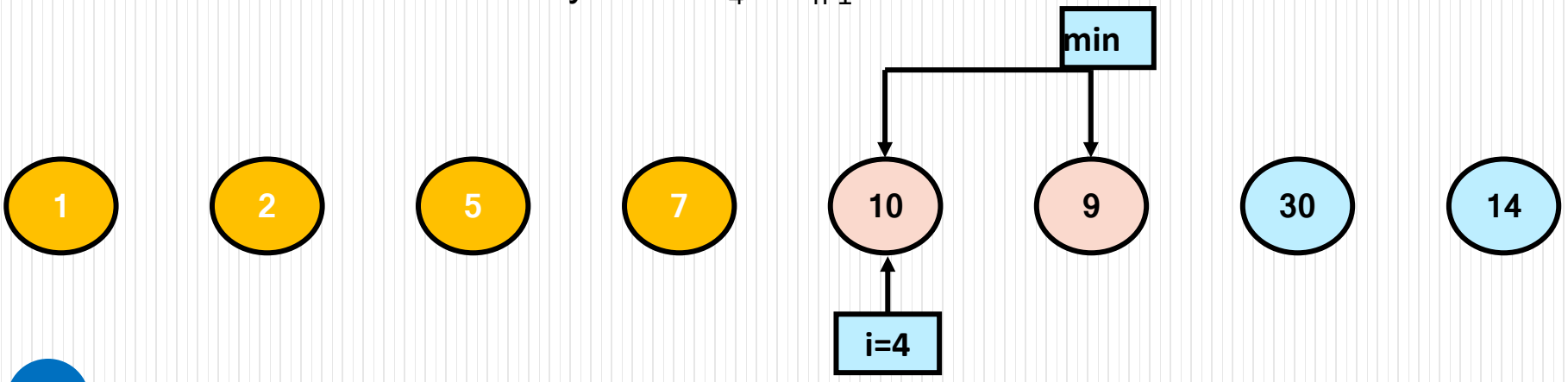


# Minh họa ví dụ Selection sort (tt)

❖ Bước 4: Tìm min của dãy số từ $a_3 – a_{n-1}$ . Sau đó hoán đổi min với $a_3$

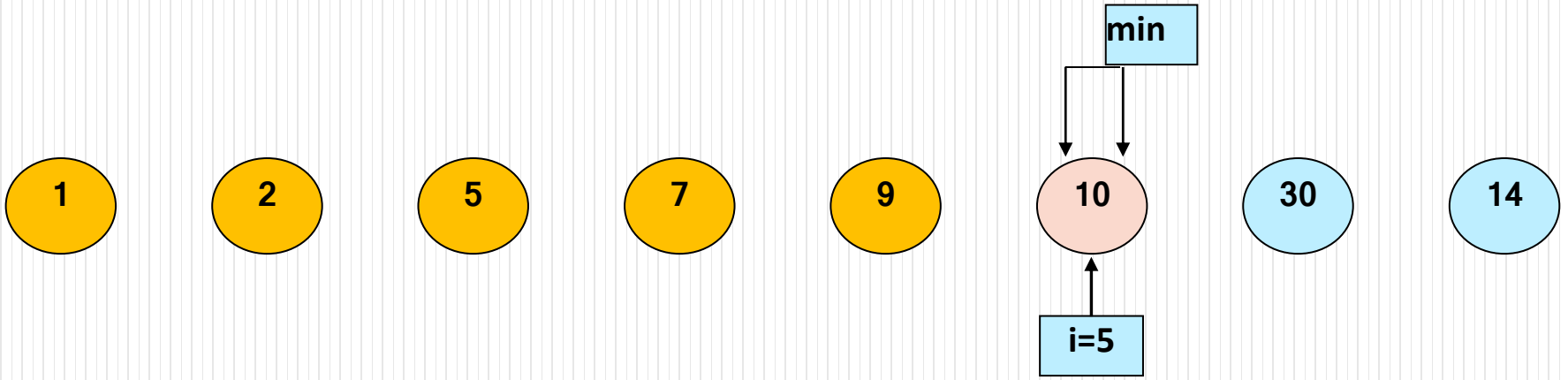



❖ Bước 5: Tìm min của dãy số từ $a_4 – a_{n-1}$ . Sau đó hoán đổi min với a4

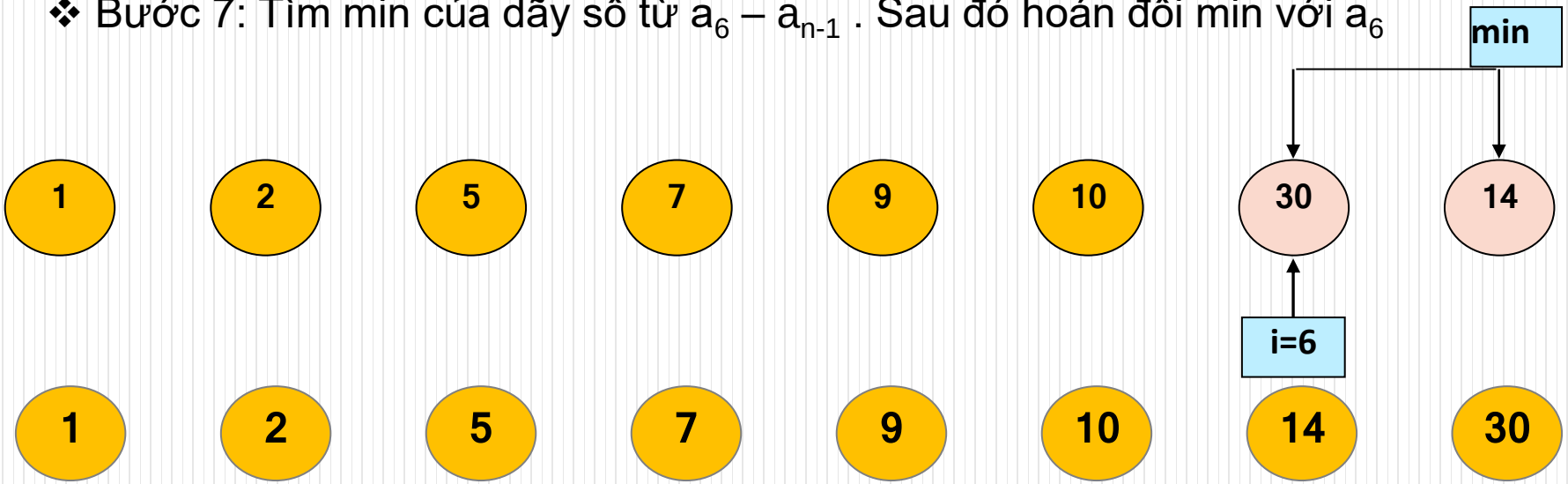


# Minh họa ví dụ Selection sort (tt)

❖ Bước 6: Tìm min của dãy số từ $a_5 – a_{n-1}$ . Sau đó hoán đổi min với $a_5$




❖ Bước 7: Tìm min của dãy số từ $a_6 – a_{n-1}$ . Sau đó hoán đổi min với $a_6$




Dừng. Vậy dãy đã được sắp xếp.

# Ví dụ

- Minh họa thao tác sắp xếp dữ liệu theo phương pháp Selection Sort cho các dãy dữ liệu sau:
  - Sắp xếp tăng:
    - **13 8 12 6 9 10 12 7**
    - **A H K R E C Z G**
  - Sắp xếp giảm
    - **13 8 12 6 9 10 12 7**
    - **A H K R E C Z G**

# Cài đặt

```
void SelectionSort(int a[],int n)
{
        for(int i=0;i<n-1;i++)
        {
                int min=i;
                for(int j=i+1;j<=n-1;j++)  //tìm min của dãy số
                        if(a[j]<a[min])    //từ a_i → a_{n-1}
                                min=j;
                swap(a[min],a[i]);
        }
}
```

# Bài tập cài đặt

- Viết bổ sung các hàm vào chương trình xử lý mảng 1 chiều các hàm thực hiện những yêu cầu sau:
  1. Viết hàm sắp xếp tăng theo PP **selection sort** cho dữ liệu số nguyên/số thực/ký tự/ chuỗi ký tự.
  2. Viết hàm sắp xếp tăng theo PP interchange sort cho dữ liệu số nguyên/số thực/ký tự/ chuỗi ký tự.

# 3.3. Giải thuật SX nhanh – Quick Sort

- **Ý tưởng giải thuật**
  - Phân hoạch dãy thành 3 dãy con:
    - Dãy con 1 : Gồm các phần tử $a_1$ …$a_i$ có giá trị không <x.
    - Dãy con 2 : Gồm các phần tử $a_{i+1}$ …$a_j$ có giá trị = x.
    - Dãy con 3: Gồm các phần tử $a_{j+1}$ … $a_n$ có giá trị >x.
  - Chọn x là phần tử a[mid] của dãy, mid = n/2
  - dãy 2 đã có thứ tự; dãy 1 và 3 nếu có 1 phần tử thì đã có thứ tự, ngược lại lặp lại phân hoạch các dãy này đến khi dãy có thứ tự .

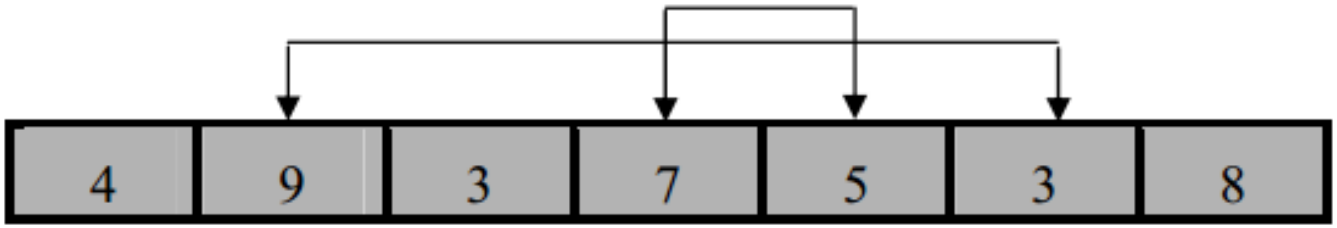

*Created by Bích Ngân & Vũ Văn Vinh*
*(nganntb@hufi.edu.vn & vinhvv@hufi.edu.vn)*

# Ví dụ minh họa

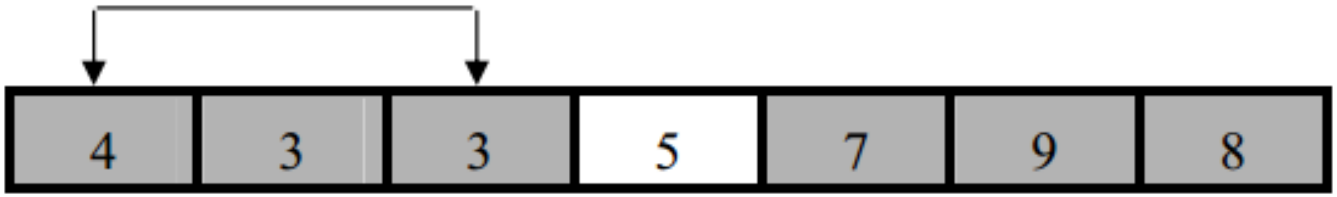
4 9 3 7 5 3 8

Công việc sắp xếp dãy trên bằng thuật toán QuickSort được tiến hành như sau:

Phân hoạch đoạn l = 0, r = 6, x = a[3] = 7




Phân hoạch đoạn l = 0, r = 2, x = a[1] = 3

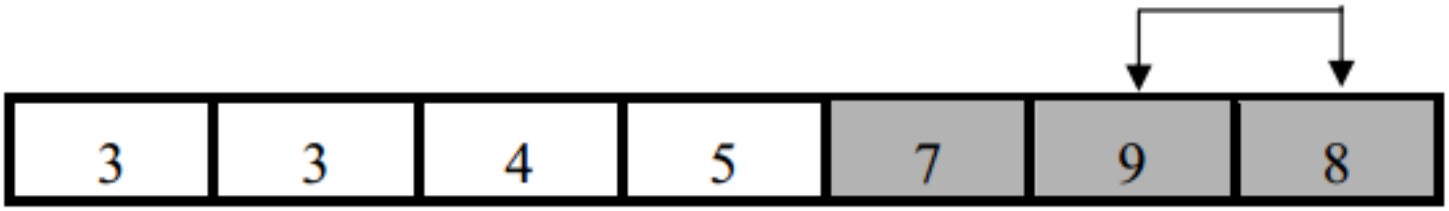




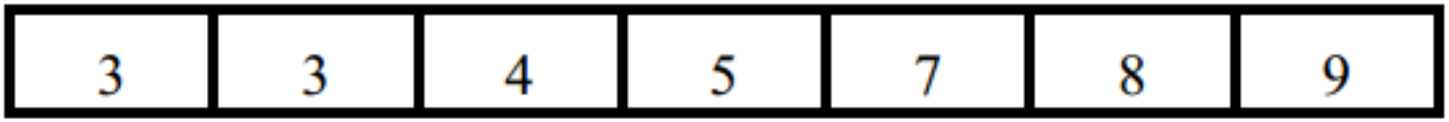

*Created by Bích Ngân & Vũ Văn Vinh*
*(nganntb@hufi.edu.vn & vinhvv@hufi.edu.vn)*

# Ví dụ minh họa (tt)

Phân hoạch đoạn l = 4, r = 6, x = a[5] = 9




Dừng






*Created by Bích Ngân & Vũ Văn Vinh*
*(nganntb@hufi.edu.vn & vinhvv@hufi.edu.vn)*

# Giải thuật Quick Sort

Bước 1 :

Chọn tuỳ 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 Hoán vị (a[i],a[j])

Bước 3 :

- Nếu i<j : Lặp lại bước 2.
- Nếu i>=j : dừng.


*Created by Bích Ngân & Vũ Văn Vinh*
*(nganntb@hufi.edu.vn & vinhvv@hufi.edu.vn)*

# Cài đặt

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

# Bài tập (bổ sung Mảng 1 chiều)

1. Viết chương trình để đảo ngược vị trí các phần tử trong mảng một chiều.
2. Nhập mảng a gồm n phần tử sao cho các số chẵn và lẻ xen kẽ nhau.
3. Viết hàm tìm phần tử chẵn lớn nhất trong mảng số nguyên n phần tử.
4. Viết hàm tìm phần tử lẻ nhỏ nhất trong mảng số nguyên n phần tử.
5. Sắp xếp mảng a có n phần tử theo thứ tự tăng dần ở các vị trí chẵn/ giảm dần ở vị trí lẻ.
6. Xóa phần tử thứ i trong mảng a có n phần tử.
7. Chèn một phần tử x vào vị trí thứ  i của mảng a.




*Created by Bích Ngân & Vũ Văn Vinh*
*(nganntb@hufi.edu.vn & vinhvv@hufi.edu.vn)*