



# THUẬT TOÁN SELECTION SORT

1. Hồ Thái Ngọc
2. ThS. Võ Duy Nguyên
3. TS. Nguyễn Tấn Trần Minh Khang



# BÀI TOÁN DẪN NHẬP

# Bài toán dẫn nhập



— Bài toán: Viết hàm tìm vị trí giá trị lớn nhất trong mảng một chiều các số thực.

— Ví dụ:

0	1	2	3	4
12	43	1	34	22

— Kết quả: 1.

# Bài toán dẫn nhập



- Bài toán: Viết hàm tìm vị trí giá trị lớn nhất trong mảng một chiều các số thực
- Hàm cài đặt

```
11. int ViTriLonNhat(float a[], int n)
12. {
13.     int lc = 0;
14.     for(int i=0; i<n; i++)
15.         if (a[i]>a[lc])
16.             lc = i;
17.     return lc;
18. }
```



# TƯ TƯỞNG THUẬT TOÁN

## Tư tưởng thuật toán



- Thuật toán Selection sort sắp xếp bằng cách đưa cách phần tử vào đúng vị trí của nó (in-place).



# THUẬT TOÁN SELECTION SORT

# Thuật toán selection sort



- Bước 0: Tìm vị trí giá trị nhỏ nhất trong phạm vi  $[0, n - 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[0]$ .
- Bước 1: Tìm vị trí giá trị nhỏ nhất trong phạm vi  $[1, n - 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[1]$ .
- ...
- Bước  $i$ : Tìm vị trí giá trị nhỏ nhất trong phạm vi  $[i, n - 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[i]$ .
- ...
- Bước  $n - 2$ : Tìm vị trí giá trị nhỏ nhất trong phạm vi  $[n - 2, n - 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[n - 2]$ .





# ÁP DỤNG THUẬT TOÁN

# Áp dụng thuật toán



— Hãy sắp xếp mảng sau tăng dần:

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

— Thứ tự các bước khi sắp tăng dần mảng trên bằng thuật toán Selection sort.

# Áp dụng thuật toán



## — Bước 0:

+ Tìm vị trí giá trị nhỏ nhất trong phạm vi  $[0,5]$  của mảng

0	1	2	3	4	5
24	45	23	13	43	-1

+ Kết quả: vị trí 5.

+ Hoán vị giá trị tại vị trí nhỏ nhất và giá trị tại vị trí 0:

0	1	2	3	4	5
24	45	23	13	43	-1

+ Kết quả sau khi sắp xếp ở bước 0

0	1	2	3	4	5
-1	45	23	13	43	24

# Áp dụng thuật toán



## — Bước 1:

+ Tìm vị trí giá trị nhỏ nhất trong phạm vi  $[1,5]$  của mảng

0	1	2	3	4	5
-1	45	23	13	43	24

+ Kết quả: vị trí 3.

+ Hoán vị giá trị tại vị trí nhỏ nhất và giá trị tại vị trí 1:

0	1	2	3	4	5
-1	45	23	13	43	24

+ Kết quả sau khi sắp xếp ở bước 1

0	1	2	3	4	5
-1	13	23	45	43	24

# Áp dụng thuật toán



## — Bước 2:

+ Tìm vị trí giá trị nhỏ nhất trong phạm vi  $[2,5]$  của mảng

0	1	2	3	4	5
-1	13	23	45	43	24

+ Kết quả: vị trí 2.

+ Hoán vị giá trị tại vị trí nhỏ nhất và giá trị tại vị trí 2:

0	1	2	3	4	5
-1	13	23	45	43	24

+ Kết quả sau khi sắp xếp ở bước 2

0	1	2	3	4	5
-1	13	23	45	43	24

# Áp dụng thuật toán



## — Bước 3:

+ Tìm vị trí giá trị nhỏ nhất trong phạm vi  $[3,5]$  của mảng

0	1	2	3	4	5
-1	13	23	45	43	24

+ Kết quả: vị trí 5.

+ Hoán vị giá trị tại vị trí nhỏ nhất và giá trị tại vị trí 3:

0	1	2	3	4	5
-1	13	23	45	43	24

+ Kết quả sau khi sắp xếp ở bước 3

0	1	2	3	4	5
-1	13	23	24	43	45

# Áp dụng thuật toán



## — Bước 4:

+ Tìm vị trí giá trị nhỏ nhất trong phạm vi  $[4,5]$  của mảng

0	1	2	3	4	5
-1	13	23	24	43	45

+ Kết quả: vị trí 4.

+ Hoán vị giá trị tại vị trí nhỏ nhất và giá trị tại vị trí 4:

0	1	2	3	4	5
-1	13	23	24	43	45

+ Kết quả sau khi sắp xếp ở bước 4

0	1	2	3	4	5
-1	13	23	24	43	45



# HÀM CÀI ĐẶT CHUẨN



# Hàm cài đặt chuẩn



— Bài toán: Định nghĩa hàm sắp mảng một chiều các số nguyên tăng dần bằng thuật toán Selection sort.

— Hàm cài đặt

```
11. void HoanVi(int& a, int& b)
12. {
13.     int temp = a;
14.     a = b;
15.     b = temp;
16. }
```

# Hàm cài đặt chuẩn



```
11. void SelectionSort(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.     {
15.         int lc = i;
16.         for(int j=i; j<=n-1; j++)
17.             if(a[j]<a[lc])
18.                 lc = j;
19.         HoanVi(a[i], a[lc]);
20.     }
21. }
```

# Hàm cài đặt chuẩn



```
11. void SelectionSort(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.     {
15.         int lc = i;
16.         for(int j=i; j<=n-1; j++)
17.             if(a[j]<a[lc])
18.                 lc = j;
19.         swap(a[i], a[lc]);
20.     }
21. }
```



Selection sort và mảng một chiều

**PROJECT I01 – DỰ ÁN I01**

# Selection sort và mảng một chiều



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập mảng một chiều từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Sắp xếp mảng tăng dần bằng thuật toán Selection sort.
- + Xuất mảng sau khi sắp xếp ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

# Selection sort và mảng một chiều



- Định dạng tập tin `intdataxx.inp` và `intdataxx.out`
  - + Dòng đầu tiên: số phần tử của mảng ( $n$ ).
  - + Dòng tiếp theo: lưu  $n$  số nguyên tương ứng với các giá trị trong mảng.

```
*intdata01.inp - Notepad
File Edit Format View Help
10
24 56 53 44 -54 6 63 -47 91 -99
```



# BIẾN THỂ CÀI ĐẶT 01

# Thuật toán selection sort



- Bước 0: Tìm vị trí giá trị nhỏ nhất trong phạm vi  $[0, n - 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[0]$ .
- Bước 1: Tìm vị trí giá trị nhỏ nhất trong phạm vi  $[1, n - 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[1]$ .
- ...
- Bước  $i$ : Tìm vị trí giá trị nhỏ nhất trong phạm vi  $[i, n - 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[i]$ .
- ...
- Bước  $n - 2$ : Tìm vị trí giá trị nhỏ nhất trong phạm vi  $[n - 2, n - 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[n - 2]$ .



# Biến thể cài đặt 01



```
11. void SelectionSort01(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.     {
15.         int lc = i;
16.         for(int j=n-1; j>=i+1; j--)
17.             if(a[j]<a[lc])
18.                 lc = j;
19.         swap(a[i], a[lc]);
20.     }
21. }
```



## BIẾN THỂ CÀI ĐẶT 02

# Biến thể thuật toán selection sort



- Bước 0: Tìm vị trí giá trị lớn nhất trong phạm vi  $[0, n - 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[n - 1]$ .
- Bước 1: Tìm vị trí giá trị lớn nhất trong phạm vi  $[0, n - 2]$  và hoán vị giá trị tại vị trí này và phần tử  $a[n - 2]$ .
- ...
- Bước  $i$ : Tìm vị trí giá trị lớn nhất trong phạm vi  $[0, n - i - 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[i]$ .
- ...
- Bước  $n - 2$ : Tìm vị trí giá trị lớn nhất trong phạm vi  $[0, 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[1]$ .

# Biến thể thuật toán selection sort



```
11. void SelectionSort02(int a[], int n)
12. {
13.     for(int i=n-1; i>=1; i--)
14.     {
15.         int lc = i;
16.         for(int j=0; j<=i-1; j++)
17.             if(a[j]>a[lc])
18.                 lc = j;
19.         swap(a[i], a[lc]);
20.     }
21. }
```



## **BIẾN THỂ CÀI ĐẶT 03**

# Biến thể thuật toán selection sort



- Bước 0: Tìm vị trí giá trị lớn nhất trong phạm vi  $[0, n - 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[n - 1]$ .
- Bước 1: Tìm vị trí giá trị lớn nhất trong phạm vi  $[0, n - 2]$  và hoán vị giá trị tại vị trí này và phần tử  $a[n - 2]$ .
- ...
- Bước  $i$ : Tìm vị trí giá trị lớn nhất trong phạm vi  $[0, n - i - 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[i]$ .
- ...
- Bước  $n - 2$ : Tìm vị trí giá trị lớn nhất trong phạm vi  $[0, 1]$  và hoán vị giá trị tại vị trí này và phần tử  $a[1]$ .

# Biến thể thuật toán selection sort



```
11. void SelectionSort03(int a[],int n)
12. {
13.     for(int i=n-1; i>=1; i--)
14.     {
15.         int lc = i;
16.         for(int j=i-1; j>=0; j--)
17.             if(a[j]>a[lc])
18.                 lc = j;
19.         swap(a[i],a[lc]);
20.     }
21. }
```



Selection sort và mảng một chiều

**PROJECT I02 – DỰ ÁN I02**



# Selection sort và mảng một chiều



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập mảng một chiều từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Sắp xếp mảng tăng dần bằng thuật toán Selection sort với biến thể 1, 2, 3.
- + Xuất mảng sau khi sắp xếp ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

# Selection sort và mảng một chiều



- Định dạng tập tin `intdataxx.inp` và `intdataxx.out`
  - + Dòng đầu tiên: số phần tử của mảng ( $n$ ).
  - + Dòng tiếp theo: lưu  $n$  số nguyên tương ứng với các giá trị trong mảng.

```
*intdata01.inp - Notepad
File Edit Format View Help
10
24 56 53 44 -54 6 63 -47 91 -99
```



# Thuật toán Selection sort và mảng cấu trúc

## **MẢNG CẤU TRÚC**

# Selection sort và mảng cấu trúc



- Bài toán: Định nghĩa hàm sắp mảng một chiều các phân số tăng dần bằng thuật toán Selection sort.
- Khai báo kiểu dữ liệu biểu diễn phân số.

```
11.struct phanso
12.{
13.|    int tu;
14.|    int mau;
15.};
16.typedef struct phanso PHANSO;
```

# Selection sort và mảng cấu trúc



```
11. int SoSanh(PHANSO x, PHANSO y)
12. {
13.     float a = (float)x.tu/x.mau;
14.     float b = (float)y.tu/y.mau;
15.     if(a>b)
16.         return 1;
17.     if(a<b)
18.         return -1;
19.     return 0;
20. }
```



Selection sort và mảng cấu trúc

**PROJECT I03 – DỰ ÁN I03**

# Selection sort và mảng cấu trúc



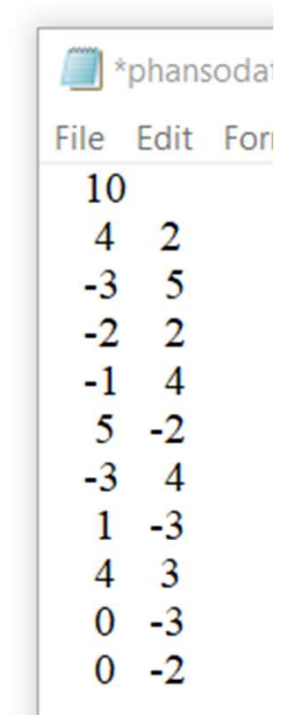
— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập mảng một chiều từ các tập tin: phansodata01.inp; phansodata02.inp; ...; phansodata09.inp; phansodata10.inp; phansodata11.inp; phansodata12.inp; phansodata13.inp;
- + Sắp xếp mảng phân số tăng dần bằng thuật toán Selection sort.
- + Xuất mảng sau khi sắp tăng ra các tập tin: phansodata01.out; phansodata02.out; ...; phansodata09.out; phansodata10.out; phansodata11.out; phansodata12.out; phansodata13.out;

# Selection sort và mảng cấu trúc



- Định dạng tập tin
  - + phansodatxx.inp,
  - + phansodatxx.out
- Dòng đầu tiên: số phần tử của mảng ( $n$ ).
- $n$  dòng tiếp theo: mỗi dòng lưu hai số nguyên tương ứng với phân số trong mảng.



10	
4	2
-3	5
-2	2
-1	4
5	-2
-3	4
1	-3
4	3
0	-3
0	-2





# Thuật toán Selection sort và ma trận

## **MA TRẬN**

# Selection sort và ma trận



- Bài toán: Định nghĩa hàm sắp ma trận các số nguyên tăng dần bằng thuật toán Selection sort.

	0	1	2	3
0	89	12	78	91
1	61	37	8	18
2	78	23	35	22

Ma trận trước khi sắp tăng

	0	1	2	3
0	8	12	18	22
1	23	35	37	61
2	78	78	89	91

Ma trận sau khi sắp tăng



Selection sort và ma trận

**PROJECT I04 – DỰ ÁN I04**

# Selection sort và ma trận



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập ma trận các số nguyên từ các tập tin: intmatran01.inp; intmatran02.inp; ...; intmatran09.inp; intmatran10.inp; intmatran11.inp; intmatran12.inp; intmatran13.inp;
- + Sắp xếp ma trận tăng dần bằng thuật toán Selection sort.
- + Xuất ma trận sau khi sắp xếp ra các tập tin: intmatran01.out; intmatran02.out; ...; intmatran09.out; intmatran10.out; intmatran11.out; intmatran12.out; intmatran13.out;

# Selection sort và ma trận



- Định dạng tập tin
  - + `intmatranxx.inp` và
  - + `intmatranxx.out`
- Dòng đầu tiên: lưu hai số nguyên tương ứng với số hàng ma trận ( $m$ ) và số cột ma trận ( $n$ ).
- $m$  dòng tiếp theo: mỗi dòng lưu  $n$  số nguyên tương ứng với các giá trị trong ma trận.

```
*intmatrandata01.inp - Notepad
File Edit Format View Help
8 10
-27 63 -7 -49 78 -27 92 -71 71 59
14 20 32 -81 -17 -12 49 -38 17 78
-71 77 33 -62 49 39 47 -23 -2 -20
89 -89 -39 0 3 97 67 -98 -32 -33
67 82 20 -9 4 -3 -1 -87 -54 61
-68 -6 -86 52 9 -80 84 -99 5 18
-54 -19 18 100 -73 -40 -58 -80 -12 -96
-43 44 32 -85 -11 32 77 33 -75 -32
```

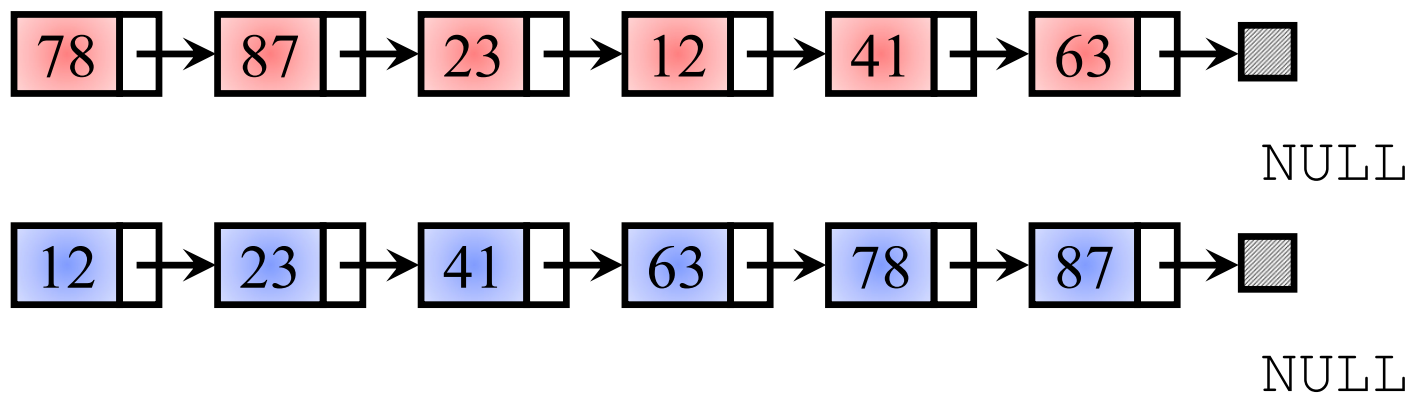


Thuật toán Selection sort và dslk đơn  
**DANH SÁCH LIÊN KẾT ĐƠN**

# Selection sort và dslk đơn



- Bài toán: Định nghĩa hàm sắp xếp danh sách liên kết đơn các số nguyên tăng dần bằng thuật toán Selection sort.





Thuật toán Selection sort và dslk đơn

**PROJECT I05 – DỰ ÁN I05**





# Thuật toán Selection sort và dslk đơn

— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập dslk đơn các số nguyên từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Sắp xếp dslk đơn các số nguyên tăng dần bằng thuật toán Selection sort.
- + Xuất dslk đơn các số nguyên sau khi sắp xếp ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;



# Thuật toán Selection sort và dslk đơn

- Định dạng tập tin `intdataxx.inp` và `intdataxx.out`
  - + Dòng đầu tiên: số phần tử của dslk đơn các số nguyên ( $n$ ).
  - + Dòng tiếp theo: lưu  $n$  số nguyên tương ứng với các giá trị trong dslk đơn các số nguyên.

```
*intdata01.inp - Notepad
File Edit Format View Help
10
24 56 53 44 -54 6 63 -47 91 -99
```

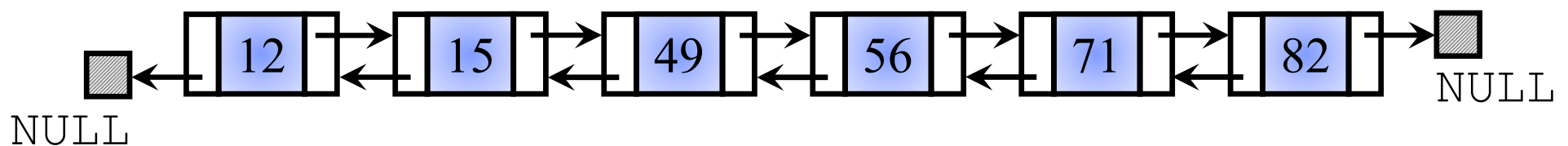
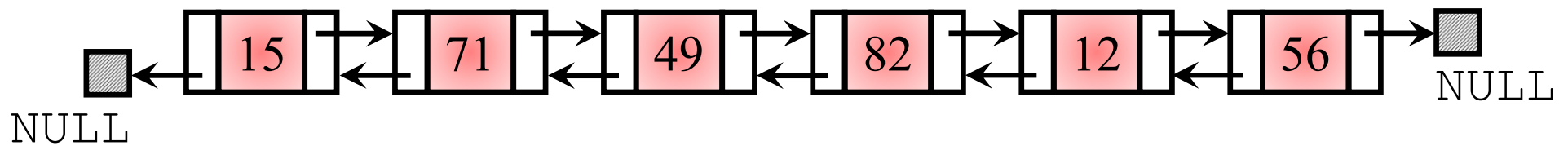


Thuật toán Selection sort và dslk kép  
**DANH SÁCH LIÊN KẾT KÉP**

# Selection sort và dslk kép



- Bài toán: Định nghĩa hàm sắp xếp danh sách liên kết kép các số nguyên tăng dần bằng thuật toán Selection sort.





# PROJECT I06 – DỰ ÁN I06

# Selection sort và dslk kép



— Viết chương trình thực hiện các yêu cầu sau:

- + Nhập dslk kép các số nguyên từ các tập tin: intdata01.inp; intdata02.inp; ...; intdata09.inp; intdata10.inp; intdata11.inp; intdata12.inp; intdata13.inp;
- + Sắp xếp dslk kép các số nguyên tăng dần bằng thuật toán Selection sort.
- + Xuất dslk kép các số nguyên sau khi sắp xếp ra các tập tin: intdata01.out; intdata02.out; ...; intdata09.out; intdata10.out; intdata11.out; intdata12.out; intdata13.out;

# Selection sort và dslk kép



- Định dạng tập tin `intdataxx.inp` và `intdataxx.out`
  - + Dòng đầu tiên: số phần tử của dslk kép các số nguyên ( $n$ ).
  - + Dòng tiếp theo: lưu  $n$  số nguyên tương ứng với các giá trị trong dslk kép các số nguyên.

```
*intdata01.inp - Notepad
File Edit Format View Help
10
24 56 53 44 -54 6 63 -47 91 -99
```



# ĐỘ PHỨC TẠP CỦA THUẬT TOÁN



# Độ phức tạp của thuật toán



```
11. void SelectionSort(int a[], int n)
12. {
13.     for(int i=0; i<=n-2; i++)
14.     {
15.         int lc = i;
16.         for(int j=i; j<=n-1; j++)
17.             if(a[j]<a[lc])
18.                 lc = j;
19.         swap(a[i], a[lc]);
20.     }
21. }
```



Thuật toán Selection sort

**ĐẶC ĐIỂM – ĐIỂM MẠNH – ĐIỂM YẾU**

# Đặc điểm – điểm mạnh – điểm yếu



— Đặc điểm thuật toán Selection sort:

- + Độ phức tạp về thời gian (time complexity):  $O(n^2)$ .
- + Độ phức tạp về bộ nhớ (space complexity):  $O(1)$ .
- + Trường hợp xấu nhất (worst case):  $O(n^2)$ .
- + Trường hợp trung bình (average case):  $O(n^2)$ .
- + Trường hợp tốt nhất (best case):  $O(n^2)$ .
- + Không ổn định.

# Đặc điểm – điểm mạnh – điểm yếu



## — Điểm mạnh:

- + Thuật toán rõ ràng, dễ hiểu.
- + Thuật toán dễ cài đặt.
- + Không yêu cầu dung lượng bộ nhớ lớn.
- + Hiệu quả trên bộ dữ liệu nhỏ.



# Đặc điểm – điểm mạnh – điểm yếu



## — Điểm yếu:

- + Thời gian thực hiện thuật toán lâu.
- + Không nhận biết mảng đã được sắp xếp.
- + Không hiệu quả trên dữ liệu lớn.



**Cảm ơn quý vị đã lắng nghe**

**ĐẠI HỌC QUỐC GIA TP.HCM**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN TP.HCM**

**TOÀN DIỆN – SÁNG TẠO – PHỤNG SỰ**