CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

CHƯƠNG 2

TÌM KIẾM VÀ SẮP XẾP

MỤC TIÊU CHƯƠNG 2

- Xác định và phát biểu bài toán tìm kiếm sắp xếp
- Hiểu một số thuật toán tìm kiếm và sắp xếp
- Phân tích ưu điểm và hạn chế của thuật toán tìm kiếm và sắp xếp
- Triển khai, cài đặt các thuật toán với C++
- Biết các thuật ngữ tiếng Anh trong bài toán tìm kiếm và sắp xếp

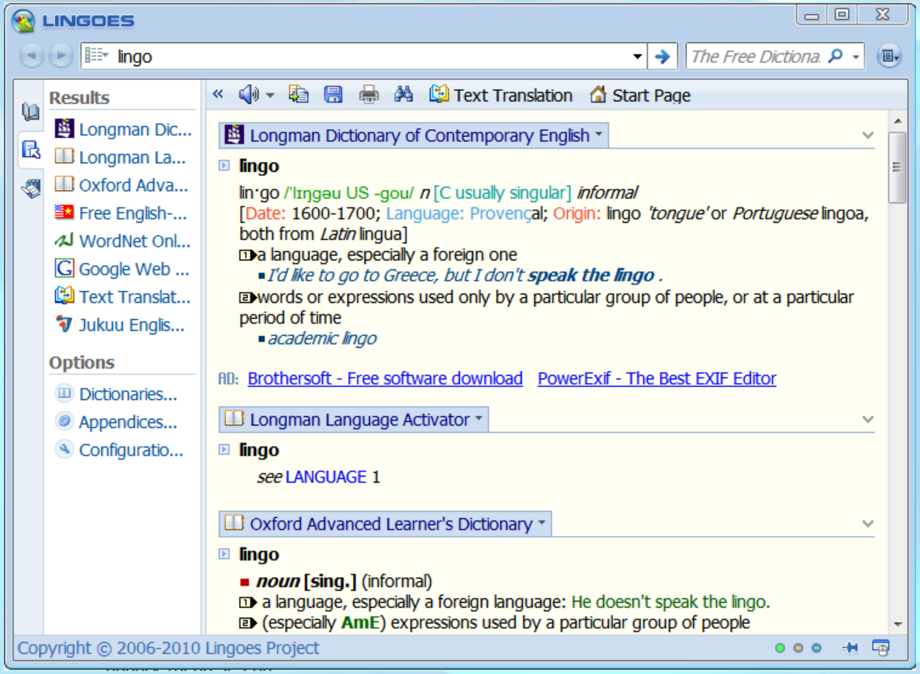
NỘI DUNG CHƯƠNG 2

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

I. NHU CẦU TÌM KIẾM,SẮP XẾP

❖TRA CỨU THÔNG TIN

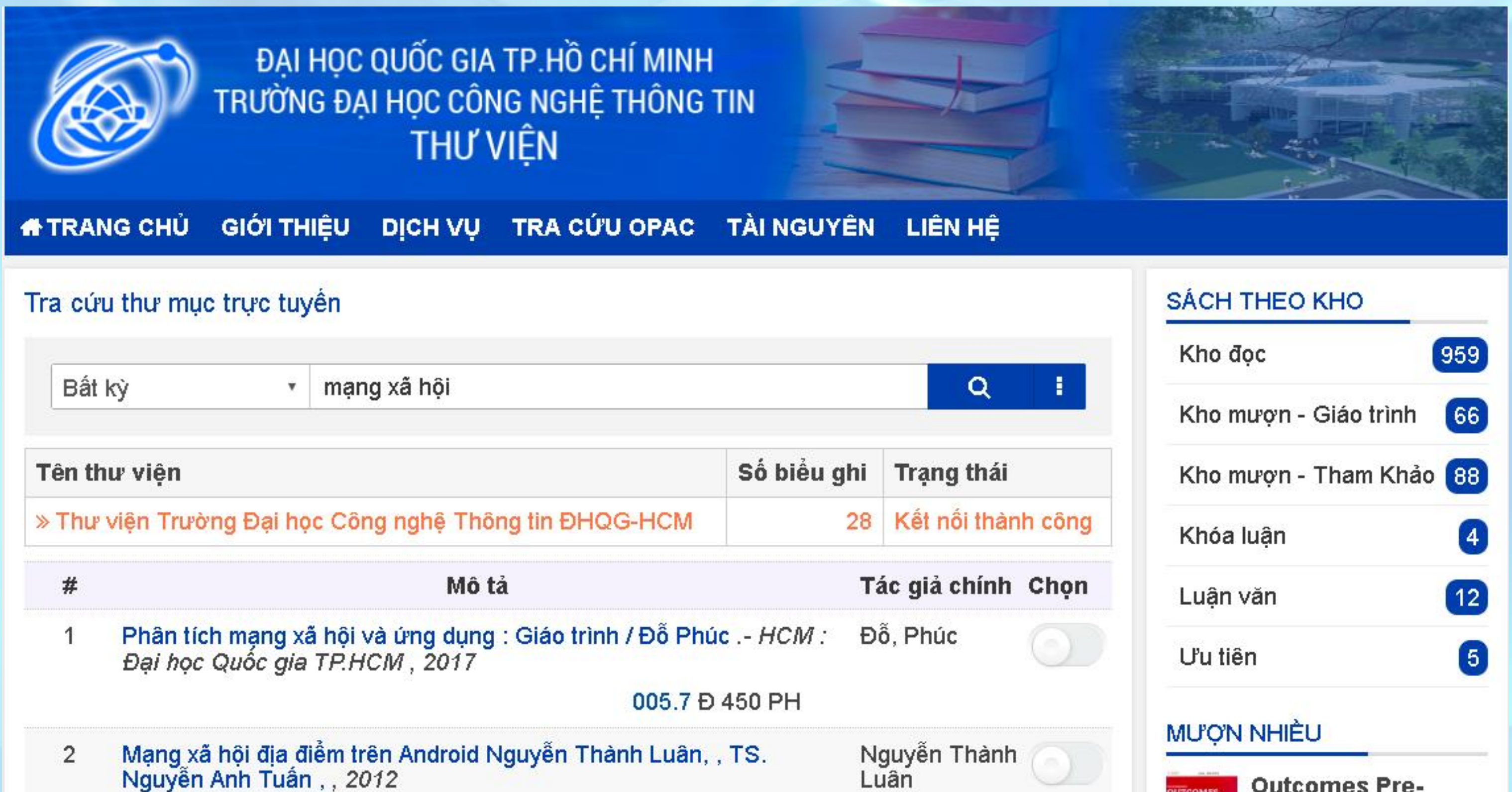
- Từ điển



I. NHU CẦU TÌM KIẾM,SẮP XẾP

❖TRA CỨU THÔNG TIN

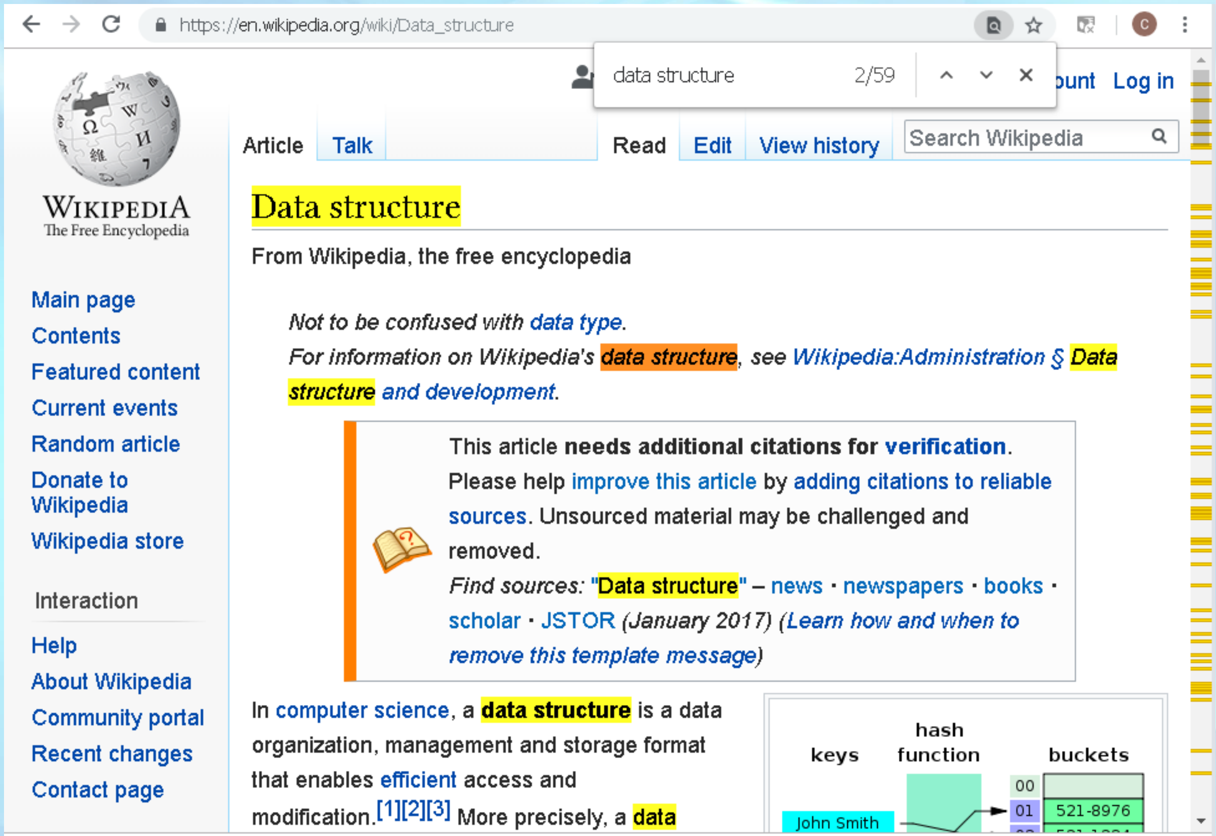
- Truy vấn dữ liệu



I. NHU CẦU TÌM KIẾM,SẮP XẾP

❖TRA CỨU THÔNG TIN

- Soạn thảo, tra cứu văn bản



I. NHU CẦU TÌM KIẾM,SẮP XẾP

❖KẾT XUẤT DỮ LIỆU

- Sắp xếp các mục từ cho từ điển.
- Sắp xếp danh sách trong các báo cáo tổng hợp

→ Sắp xếp để thiết lập thứ tự cho danh sách, làm tăng hiệu quả cho tìm kiếm.

II. CÁC GIẢI THUẬT TÌM KIẾM

❖PHÁT BIỂU BÀI TOÁN

Cho danh sách ***A*** gồm n phần tử $a_0, a_1, .., a_{n-1}$

Tìm phần tử có giá trị khóa là ***x*** trong ***A***. Nếu a_i có giá trị khóa là ***x*** thì trả về chỉ số ***i***

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

Từ khóa: Linear Search

Điều kiện: Danh sách $A = \{a_0, a_1, .., a_{n-1}\}$ chưa có thứ tự.

Phân tích: không có thông tin nào ngoài thông tin có được khi so sánh x với giá trị khóa của a_i

Ý tưởng: duyệt toàn bộ danh sách A để xác định $a_i = x$ và trả về i nếu tồn tại $a_i = x$.

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

Thuật toán:

Đầu vào: Danh sách ***A*** có ***n*** phần tử, giá trị khóa ***x*** cần tìm.

Đầu ra: Chỉ số ***i*** của phần tử a_i trong ***A*** có giá trị khóa là ***x***. Trong trường hợp không tìm thấy ***i=-1***

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

Thuật toán:

```
i ← 0
while i < n
  if A[i] = x then return i end if
  i ← i+1
end while
return -1
```

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

Quá trình tính toán:

Giả sử A = {1,3,2,9,7}, x = 9.

Quá trình xác định a_i theo thuật toán tìm tuyến tính

i=0	1	2	3	4
1	3	2	9	7

x=9

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

Quá trình tính toán:

Giả sử A = {1,3,2,9,7}, x = 9.

Quá trình xác định a_i theo thuật toán tìm tuyến tính

0	i=1	2	3	4
1	**3**	2	9	7

$\boxed{x=9}$

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

Quá trình tính toán:

Giả sử A = {1,3,2,9,7}, x = 9.

Quá trình xác định a_i theo thuật toán tìm tuyến tính

0	1	i=2	3	4
1	**3**	**2**	9	7

x=9

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

Quá trình tính toán:

Giả sử A = {1,3,2,9,7}, x = 9.

Quá trình xác định a_i theo thuật toán tìm tuyến tính

0	1	2	i=3	4
1	**3**	**2**	**9**	7

x=9

i = 3
A[i] = 9 = x

II. CÁC GIẢI THUẬT TÌM KIẾM

❖ TÌM KIẾM TUYẾN TÍNH

Cài đặt: (trên mảng)

```
int linearSearch(int A[], int n, int x) {
  int i = 0;
  while (i < n) {
    if (A[i] == x) return i;
    i++;
  }
  return -1;
}
```

II. CÁC GIẢI THUẬT TÌM KIẾM

❖ TÌM KIẾM TUYẾN TÍNH

Cài đặt: (trên danh sách đơn)

```
Node* linearSearch(List A, int x) {
  Node *p = A.pHead;
  while (!p) {
    if (p->info == x) return p;
    p = p->pNext;
  }
  return NULL;
}
```

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

Đánh giá:

- *Trường hợp tốt nhất (best case)*: a_0 chứa khóa x → số lần lặp là 1 → độ phức tạp hằng số O(1)
- *Trường hợp xấu nhất (worst case)*: A không có phần tử có khóa x → số lần lặp là n → độ phức tạp tuyến tính O(n).
- *Trường hợp trung bình (average case)*: độ phức tạp tuyến tính O(n).

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH (cải tiến)

Phân tích: Theo thuật toán tìm tuyến tính:

- Cần phải kiểm tra điều kiện dừng khi xét hết danh sách (i < n)
- Cần phải kiểm tra điều kiện dừng khi tìm thấy phần tử a_i trong vòng lặp

→ Rút gọn điều kiện dừng

II. CÁC GIẢI THUẬT TÌM KIẾM

❖ TÌM KIẾM TUYẾN TÍNH (cải tiến)

Ý tưởng:

- Thêm phần tử **a_n** có khóa **x** vào **A**, khi này **A** có **n+1** phần tử. Phần tử thêm vào được gọi là phần tử cầm canh.
- Chỉ cần điều kiện dừng là tìm thấy phần tử **a_i** có khóa **x**

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH (cải tiến)

<u>**Thuật toán**</u>:

Đầu vào: Danh sách ***A*** có ***n*** phần tử, giá trị khóa ***x*** cần tìm.

Đầu ra: Chỉ số ***i*** của phần tử a_i trong ***A*** có giá trị khóa là ***x***. Trong trường hợp không tìm thấy ***i=-1***

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH (cải tiến)

Thuật toán:

```
i ← 0, A[n] = x
while A[i] ≠ x
  i ← i+1
end while
if (i < n) then return i
else return -1 end if
```

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH (cải tiến)

Cài đặt: (trên mảng)

```
int linearSearchA(int A[],int n,int x) {
  int i = 0; A[n] = x;//A có hơn n phần tử
  while (A[i] != x)
     i++;
  if (i < n)  return i;
  else return -1;
}
```

II. CÁC GIẢI THUẬT TÌM KIẾM

❖ TÌM KIẾM TUYẾN TÍNH (cải tiến)

Cài đặt: (trên danh sách đơn)

```
Node* linearSearchA(List A, int x) {
  Node *p = A.pHead, *t = new Node(x);
  if (!t) throw "out of memory";
  addTail(A, t);
  while (p->info != x) p = p->pNext;
  if (p == A.pTail) return p;
  else return NULL;
}
```

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

Từ khóa: Binary Search

Điều kiện: Danh sách $A = \{a_0, a_1, .., a_{n-1}\}$ đã có thứ tự $\Re$

Phân tích: Khi so sánh a_i với khóa **x**, dựa vào quan hệ thứ tự, có thể quyết định nên xét phần tử kế tiếp ở phần trước (hoặc phần sau) của a_i hay không.

II. CÁC GIẢI THUẬT TÌM KIẾM

❖ TÌM KIẾM NHỊ PHÂN

Ý tưởng:

- Chọn a_m ở giữa **A** để tận dụng kết quả so sánh với khóa **x**. **A** được chia thành hai phần: trước và sau a_m. Chỉ số bắt đầu, kết thúc của ***A*** là ***l***, ***r***
- Nếu **x =** a_m, tìm thấy và dừng.
- Xét thứ tự **x**, a_m. Nếu thứ tự này
  - Là $\Re$, thì tìm **x** trong đoạn [**l, r**] với **r=m-1**;
  - Ngược lại, tìm **x** trong đoạn [**l, r**] với **l=m+1**.

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

Thuật toán:

Đầu vào: Danh sách ***A*** có ***n*** phần tử đã có thứ tự $\Re$, giá trị khóa ***x*** cần tìm.

Đầu ra: Chỉ số ***i*** của phần tử a_i trong ***A*** có giá trị khóa là ***x***. Trong trường hợp không tìm thấy ***i=-1***

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

Thuật toán:

```
l ← 0, r ← n-1
while l ≤ r
  m ← (l + r) div 2
  if x = A[m] then return m end if
  if x ℜ A[m] then r ← m – 1
  else l ← m + 1 end if
end while
return -1
```

II. CÁC GIẢI THUẬT TÌM KIẾM

❖ TÌM KIẾM NHỊ PHÂN

Quá trình tính toán:

Giả sử ***A = {1,2,3,4,5,7,9}***, thứ tự $\Re$ là **<**, phần tử cần tìm ***x = 3***

l=0	1	2	m=3	4	5	r=6
1	2	3	**4**	5	7	9

x=3

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

<u>Quá trình tính toán</u>:

Giả sử ***A = {1,2,3,4,5,7,9}***, thứ tự **$\Re$** là **<**, phần tử cần tìm ***x = 3***

l=0	m=1	r=2	3	4	5	6
1	**2**	3	4	5	7	9

x=3

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

Quá trình tính toán:

Giả sử ***A = {1,2,3,4,5,7,9}***, thứ tự $\Re$ là **<**, phần tử cần tìm ***x = 3***

l=2

r=2

0	1	m=2	3	4	5	6
1	**2**	**3**	4	5	7	9

x=3

m = 2

A[m] = 3 = x

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

Cài đặt: (trên mảng, thứ tự $\Re$ là **<**)

```
int binarySearch (int A[], int n, int x){
  int l = 0, r = n-1;
  while (l <= r) {
     m = (l + r) / 2;
     if (x == A[m]) return m;
     if (x < A[m]) r = m – 1;
     else l = m + 1;
  }
  return -1;
}
```

II. CÁC GIẢI THUẬT TÌM KIẾM

❖ TÌM KIẾM NHỊ PHÂN

Cài đặt: (trên danh sách liên kết)

Tìm kiếm nhị phân trên danh sách liên kết cần một cấu trúc liên kết khác: cây nhị phân tìm kiếm.

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

Đánh giá:

- *Trường hợp tốt nhất*: phần tử cần tìm ở đúng vị trí ***(l+r) div 2*** → số lần lặp là 1 → độ phức tạp hằng số **O(1)**.
- *Trường hợp xấu nhất*: số lần tìm là số lần chia đôi dãy đến khi dãy tìm kiếm còn 1 phần tử → số lần lặp khoảng $\log_2(n)+1$ → độ phức tạp logarith **O(log(n))**.
- *Trường hợp trung bình*: độ phức tạp **O(log(n))**.

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NỘI SUY

Từ khóa: Interpolation Search

Điều kiện: Danh sách $A = \{a_0, a_1, .., a_{n-1}\}$ đã có thứ tự $\Re$ và giá trị khóa được rải đều trên danh sách.

Phân tích: Giá trị khóa rải đều trên danh sách → vị trí a_m chia danh sách tìm kiếm tương ứng với tỉ lệ giá trị x trong miền giá trị khóa của danh sách tìm kiếm.

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NỘI SUY

Ý tưởng:

- Thay vì xác định điểm **m = (l + r) / 2** như trong tìm kiến nhị phân, xác định nội suy m như sau:

$$m = l + \frac{(r - l) \times (x - A[l])}{A[r] - A[l]}$$

- Các bước còn lại tương tự tìm kiếm nhị phân

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NỘI SUY

Thuật toán:

Đầu vào: Danh sách ***A*** có ***n*** phần tử đã có thứ tự $\Re$, giá trị khóa ***x*** cần tìm.

Đầu ra: Chỉ số ***i*** của phần tử a_i trong ***A*** có giá trị khóa là ***x***. Trong trường hợp không tìm thấy ***i=-1***

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NỘI SUY

Thuật toán:

```
l ← 0, r ← n-1
while l ≤ r
  m ← l+((r-l)*(x-A[l]) / (A[r]-A[l]))
  if x = A[m] then return m end if
  if x ℜ A[m] then r ← m – 1
  else l ← m + 1 end if
end while
return -1
```

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NỘI SUY

Quá trình tính toán:

Giả sử ***A = {1,2,3,4,5,7,9}***, thứ tự $\Re$ là **<**, phần tử cần tìm ***x = 3***

l=0	m=1	2	3	4	5	r=6
1	**2**	3	4	5	7	9

x=3

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NỘI SUY

<u>**Quá trình tính toán**</u>:

Giả sử ***A = {1,2,3,4,5,7,9}***, thứ tự $\Re$ là **<**, phần tử cần tìm ***x = 3***

m=2

0	1	l=2	3	4	5	r=6
1	2	**3**	4	5	7	9

x=3

m = 2
A[m] = 3 = x

II. CÁC GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NỘI SUY

Cài đặt: (trên mảng, thứ tự $\Re$ là **<**)

```
int interpolationSearch (int A[],int n,int x){
  int l = 0, r = n-1;
  while (l <= r) {
     m = l+(r-l)*(x-A[l])/(A[r]-A[l]);
     if (x == A[m]) return m;
     if (x < A[m]) r = m – 1;
     else l = m + 1;
  }
  return -1;
}
```

II. CÁC GIẢI THUẬT TÌM KIẾM

❖ TÌM KIẾM NỘI SUY

Đánh giá:

- *Trường hợp tốt nhất*: phần tử cần tìm ở đúng vị được nội suy → số lần lặp là 1 → độ phức tạp hằng số **O(1)**.
- *Trường hợp xấu nhất*: giá trị khóa lớn nhất hoặc nhỏ nhất chênh lệch quá lớn so với giá trị kỳ vọng → tìm tuyến tính → độ phức tạp **O(n)**.
- *Trường hợp trung bình*: độ phức tạp **O(log(n))**.

II. CÁC GIẢI THUẬT TÌM KIẾM

❖BÀI TẬP

1) Cho danh sách A={1,2,3,4,5,6,100000} được lưu trữ trên mảng.
   a) Cho biết thuật toán tốt nhất để tìm giá trị x trong A. Vì sao?
   b) Trình bày từng bước quá trình tìm giá trị x=6 trong A theo thuật toán đã chọn.
   c) Giả sử A được lưu trữ trên danh sách liên kết đơn. Cho biết thuật toán tốt nhất để tìm giá trị x trong A. Vì sao?

II. CÁC GIẢI THUẬT TÌM KIẾM

❖BÀI TẬP

2) Viết hàm tìm kiếm phần tử x trên mảng A chứa n số nguyên. Biết A đang có thứ tự **>** (giảm dần) và chưa biết phân bố giá trị của các phần tử trong A.

II. CÁC GIẢI THUẬT TÌM KIẾM

❖ **BÀI TẬP**

3) Cho cấu trúc điểm trong mặt phẳng như sau:

```
struct Point {
    float x, y;
};
```

Viết hàm tìm kiếm điểm $q(x_q,y_q)$ trong danh sách các điểm A (A được lưu trữ trên mảng) sao cho khoảng cách giữa q và $p(x_p,y_p)$ là nhỏ nhất. Trong đó p là một điểm cho trước (tham số của hàm tìm kiếm). Kết quả trả về là chỉ số của q trong A.

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

CHƯƠNG 2

TÌM KIẾM VÀ SẮP XẾP

MỤC TIÊU CHƯƠNG 2

- Xác định và phát biểu bài toán tìm kiếm sắp xếp
- Hiểu một số thuật toán tìm kiếm và sắp xếp
- Phân tích ưu điểm và hạn chế của thuật toán tìm kiếm và sắp xếp
- Triển khai, cài đặt các thuật toán với C++
- Biết các thuật ngữ tiếng Anh trong bài toán tìm kiếm và sắp xếp

NỘI DUNG CHƯƠNG 2

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

❖PHÁT BIỂU BÀI TOÁN

Cho danh sách ***A*** gồm n phần tử $a_0, a_1, .., a_{n-1}$

Hoán đổi vị trí của các phần tử a_i và a_j sao cho đảm bảo thứ tự $\Re$ trong ***A***, nghĩa là

$$a_i \Re a_j , \forall i < j$$

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHÂN LOẠI SẮP XẾP

Các giải thuật sắp xếp có thể phân loại theo nhiều tiêu chí:

a) ***Tính chất của danh sách A***:

- Toàn bộ phần tử của A được xử lý đồng thời trong quá trình sắp xếp → Offline Sorting
  - Selection Sort
  - Bubble Sort
  - Quick Sort
  - ....

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHÂN LOẠI SẮP XẾP

Các giải thuật sắp xếp có thể phân loại theo nhiều tiêu chí:

a) ***Tính chất của danh sách A:***

- Từng phần tử của A được sắp xếp tuần tự mà không cần biết trước toàn bộ A→ Online Sorting
  - Insertion Sort
  - Tree Sort (tạo Binary Search Tree)

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHÂN LOẠI SẮP XẾP

Các giải thuật sắp xếp có thể phân loại theo nhiều tiêu chí:

b)Trật tự của kết quả sắp xếp:

- Thứ tự trước/sau của các phần tử có cùng giá trị khóa không đổi so với ban đầu → Stable Sorting. Ví dụ:
  - Bubble Sort:
    - Trước khi sắp xếp: $A=\{1,5_1,2,5_2,3,5_3\}$
    - Sau khi sắp xếp (tăng dần): $A=\{1,2,3,5_1,5_2,5_3\}$

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHÂN LOẠI SẮP XẾP

Các giải thuật sắp xếp có thể phân loại theo nhiều tiêu chí:

b)Trật tự của kết quả sắp xếp:

- Thứ tự trước/sau của các phần tử có cùng giá trị khóa thay đổi so với ban đầu → Unstable Sorting. Ví dụ:
  - Interchange Sort:
    - Trước khi sắp xếp: $A=\{1,5_1,2,5_2,3,5_3\}$
    - Sau khi sắp xếp (tăng dần): $A=\{1,2,3,5_2,5_1,5_3\}$

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHÂN LOẠI SẮP XẾP

Các giải thuật sắp xếp có thể phân loại theo nhiều tiêu chí:

c)***Nơi lưu trữ chính của danh sách:***

- Toàn bộ danh sách A được lưu trữ trên RAM trong quá trình sắp xếp → Internal Sorting.
  - Interchange Sort
  - Insertion Sort
  - Quick Sort
  - …

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHÂN LOẠI SẮP XẾP

Các giải thuật sắp xếp có thể phân loại theo nhiều tiêu chí:

c) ***Nơi lưu trữ chính của danh sách:***

- Toàn bộ danh sách A được lưu trữ trên bộ nhớ ngoài (HDD) trong quá trình sắp xếp do kích thước danh sách quá lớn → External Sorting.
  - Merge Sort

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Từ khóa: Selection Sort

Phân tích: Giả sử danh sách $A=\{a_0, a_1, .., a_{n-1}\}$ đã có thứ tự $\Re$.Khi đó:

- a_0 là phần tử nhỏ nhất theo $\Re$ trong A
- a_1 là phần tử nhỏ nhất theo $\Re$ trong $A \setminus \{a_0\}$
- a_2 là phần tử nhỏ nhất theo $\Re$ trong $A \setminus \{a_0, a_1\}$
- …

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Ý tưởng: Chọn phần tử nhỏ thứ **i** theo thứ tự $\Re$ trong danh sách **A** và đặt vào vị trí **i** của danh sách.

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Thuật toán:

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

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

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Thuật toán:

```
i ← 0
while i < n - 1
  min ← i, j ← i+1
   while j < n
     if A[j] ℜ A[min] then min ← j
     j ← j+1
  swap(A[i], A[min])
  i ← i + 1
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Quá trình tính toán:

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

i=0	j=1	2	3	4
3	**2**	**5**	**1**	**4**

min=0

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Quá trình tính toán:

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

i=0	1	j=2	3	4
3	**2**	**5**	**1**	**4**

min=1

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Quá trình tính toán:

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

i=0	1	2	j=3	4
3	**2**	**5**	**1**	**4**

min=1

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Quá trình tính toán:

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

i=0	1	2	3	j=4
3	**2**	**5**	**1**	**4**

min=3

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Quá trình tính toán:

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

i=0	1	2	3	j=4
3	**2**	**5**	**1**	**4**

min=3

Hoán đổi giá trị A[i] và A[min]

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

<u>Quá trình tính toán</u>:

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

0	i=1	j=2	3	4
1	**2**	**5**	**3**	**4**

min=1

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Quá trình tính toán:

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

0	i=1	2	j=3	4
1	2	5	3	4

min=1

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

<u>Quá trình tính toán</u>:

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

0	i=1	2	3	j=4
1	**2**	**5**	**3**	**4**

min=1

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Quá trình tính toán:

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

0	i=1	2	3	4
1	**2**	**5**	**3**	**4**

min=1

Hoán đổi giá trị A[i] và A[min]

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Quá trình tính toán:

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

0	1	i=2	j=3	4
1	2	5	3	4

min=2

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

<u>Quá trình tính toán</u>:

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

0	1	i=2	3	j=4
1	2	5	3	4

min=3

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Quá trình tính toán:

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

0	1	i=2	3	4
1	**2**	**5**	**3**	**4**

min=3

Hoán đổi giá trị A[i] và A[min]

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Quá trình tính toán:

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

0	1	2	i=3	j=4
1	**2**	**3**	**5**	**4**

min=3

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

<u>Quá trình tính toán</u>:

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

0	1	2	i=3	4
1	**2**	**3**	**5**	**4**

min=4

Hoán đổi giá trị A[i] và A[min]

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Quá trình tính toán:

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

0	1	2	3	4
1	**2**	**3**	**4**	**5**

Kết thúc

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Cài đặt: (trên mảng) giả sử thứ thụ là **<** (tăng dần)

```
void selectionSort(int A[], int n) {
  int min;
  for (int i = 0; i < n-1; i++) {
    min = i;
    for (int j = i+1; j < n; j++)
      if (A[j] < A[min]) min = j;
    swap(A[i], A[min]);
  }
}
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Cài đặt: (trên danh sách liên kết đơn) giả sử thứ thự là **<** (tăng dần)

Trường hợp 1: sắp xếp bằng cách thay đổi giá trị tại mỗi node

Chưa sắp xếp: → 3 → 2 → 5 → 1 → 4 → ●

Đã sắp xếp: → 1 → 2 → 3 → 4 → 5 → ●

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

```
void selectionSort(List A) {
  Node *min, *i, *j;
  i = A.pHead;
  while (i) {
    min = i; j = i->pNext;
    while (j) {
      if (j->info < min->info) min = j;
      j = j->pNext;
    }
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖ PHƯƠNG PHÁP CHỌN TRỰC TIẾP

```
        swap(i->info, min->info);
        i = i->pNext;
    }
}
```

III. CÁC GIẢI THUẬT SẮP XẾP

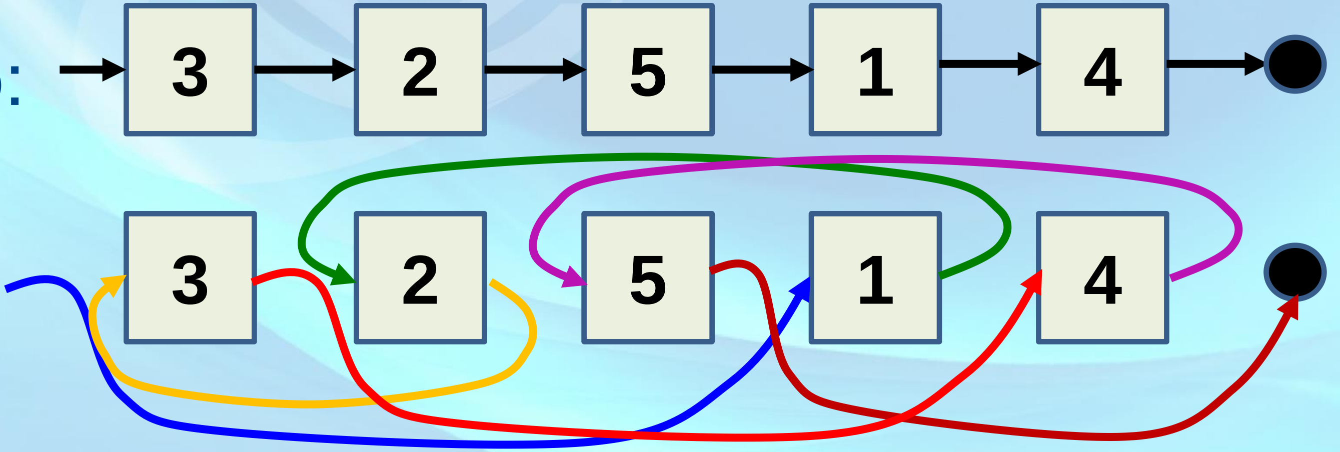
❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Cài đặt: (trên danh sách liên kết đơn) giả sử thứ thự là **<** (tăng dần)

Trường hợp 2: sắp xếp bằng cách thay đổi liên kết tại mỗi node

Chưa sắp xếp:

Đã sắp xếp:



III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

```
void selectionSort(List &A) {
  Node *qmin, *i, *j, *h;
  h = NULL; i = A.pHead;
  while (i->pNext) {
    qmin = h; j = i;
    while (j->pNext) {
      if (j->pNext->info < i->info) {
        qmin = j; i = qmin->pNext;
      }
      j = j->pNext;
    }
```

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

```
    int t;
    removeAfter(A,qmin,t);
    addAfter(A, createNode(t), h);
    if (!h) h = A.pHead;
    else h = h->pNext;
    i = h->pNext;
  }
}
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHỌN TRỰC TIẾP

Đánh giá:

	TỐT NHẤT (đúng thứ tự)	**TRUNG BÌNH** (chưa có thứ tự)	**XẤU NHẤT** (thứ tự ngược)
Theo phép so sánh	$O(n^2)$	$O(n^2)$	$O(n^2)$
Theo phép gán giá trị khóa	O(n)	O(n)	O(n)

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

Từ khóa: Insertion Sort

Phân tích: Giả sử danh sách $A=\{a_0, a_1, .., a_{n-1}\}$ đã có thứ tự $\Re$. Khi đó, để tạo danh sách A có n+1 phần tử có thứ tự $\Re$, cần tìm vị trí **k** để chèn phần tử a_n sao cho đảm bảo

$$a_i \ \Re \ a_n \quad \forall i < k$$

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

<u>**Ý tưởng**</u>:

- Danh sách chỉ có một phần tử luôn có thứ tự. Như vậy, $A_0=\{a_0\}$ là danh sách có thứ tự $\Re$
- Để sắp xếp $A=\{a_0, a_1, .., a_{n-1}\}$ theo thứ tự $\Re$. Lần lượt lấy a_i (i>0) trong A và thực hiện
  1. Bắt đầu từ cuối danh sách $A_{i-1}=\{a_0, .., a_{i-1}\}$, Tìm vị trí **k** đầu tiên thỏa điều kiện $a_k \Re a_i$.
  2. Đẩy tất cả phần tử (nếu có) từ ngay sau vị trí **k** về bên phải 1 vị trí.
  3. Đưa vào a_i vị trí **k+1** của A_{i-1}, A_{i-1} thành A_i

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

Thuật toán:

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

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

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

<u>**Thuật toán**</u>:

```
i ← 1
while i < n
  e ← A[i]
  k ← i-1
  while (k ≥ 0) and not (A[k] ℜ e)
     A[k+1] ← A[k]
     k ← k-1
  A[k+1] ← e,
  i ← i+1
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

Quá trình tính toán:

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

	0	i=1	2	3	4
e=2	3	2	5	1	4
	k=0				

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

<u>Quá trình tính toán</u>:

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

	0	i=1	2	3	4
e=2	3	3	5	1	4
k=-1					

Đưa phần tử e vào vị trí k+1

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

<u>Quá trình tính toán</u>:

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

	0	1	i=2	3	4
e=5	2	3	5	1	4
		k=1			

Đưa phần tử e vào vị trí k+1

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

Quá trình tính toán:

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

	0	1	2	i=3	4
e=1	**2**	**3**	**5**	**1**	**4**
			k=2		

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

<u>Quá trình tính toán</u>:

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

	0	1	2	i=3	4
e=1	2	3	5	5	4
		k=1			

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

<u>Quá trình tính toán</u>:

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

	0	1	2	i=3	4
e=1	**2**	**3**	**3**	**5**	**4**
	k=0				

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

Quá trình tính toán:

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

	0	1	2	i=3	4
e=1	2	2	3	5	4

k=-1

Đưa phần tử e vào vị trí k+1

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

Quá trình tính toán:

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

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

Quá trình tính toán:

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

	0	1	2	3	i=4
e=4	1	2	3	5	5

k=2

Đưa phần tử e vào vị trí k+1

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

<u>Quá trình tính toán</u>:

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

0	1	2	3	4
1	**2**	**3**	**4**	**5**

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

Cài đặt: (trên mảng, tìm tuyến tính kết hợp dời vị trí) Giả sử thứ thự là **<** (tăng dần)

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

```
void insertionSort(int A[], int n) {
  for (int i = 1; i < n; i++) {
    int e = A[i]; int k;
    for (k = i-1; k>-1; k--) {
      if (A[k] < e) break;
      A[k+1] = A[k];
    }
    A[k+1] = e;
  }
}
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

Cài đặt: (trên danh sách liên kết đơn, tìm tuyến tính) Giả sử thứ thự là **<** (tăng dần):

Sắp xếp theo nguyên tắc thay đổi liên kết của node

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

```
void insertionSort(List &A) {
  Node *i = A.pHead, *k, *e;
  while (i->pNext) {
     q = NULL; k = A.pHead; e=removeAfter(A,i);
     while (k != i->pNext) {
        if (!(k->info < e->info)) break;
        q = k; k = q->pNext;
     }
     addAfter(A, e, q);
     if (i->pNext == e) i = i->pNext;
  }
}
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

Đánh giá:

	TỐT NHẤT (đúng thứ tự)	**TRUNG BÌNH** (chưa có thứ tự)	**XẤU NHẤT** (thứ tự ngược)
Theo phép so sánh	O(n)	$O(n^2)$	$O(n^2)$
Theo phép gán giá trị khóa	O(1)	$O(n^2)$	$O(n^2)$

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CHÈN TRỰC TIẾP

Lưu ý:

- Đối với cấu trúc mảng, thao tác tìm vị trí **k** có thể áp dụng tìm nhị phân → Binary Insertion Sort

III. CÁC GIẢI THUẬT SẮP XẾP

❖ PHƯƠNG PHÁP ĐẾM

Từ khóa: Counting Sorting

Điều kiện: Giá trị khóa là số nguyên dương có giá trị lớn nhất không quá lớn.

Phân tích: Nếu giá trị khóa là số nguyên và có thể cấp phát mảng với kích thước bằng giá trị lớn nhất → chỉ cần đếm số lượng giá trị khóa xuất hiện trong danh sách A.

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

Ý tưởng:

- Giá trị khóa của a_i là chỉ số của mảng **B** có **k** phần tử (**k** là giá trị khóa lớn nhất của **A**)
- Quá trình sắp xếp danh sách **A** là đếm số phần tử của mỗi chỉ số của **B** trong **A**. Từ đó tính ra thứ tự của các khóa a_i trong **A**
- Kết quả sắp xếp có được bằng cách lấy vị trí của phần tử **A** được lưu trong **B**.

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

Thuật toán:

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

Đầu ra: $C=\{c_0, c_1, .., c_{n-1}\}$ đã có thứ tự tăng dần

```
for i ← 0 to n-1 do
  B[A[i]] ← B[A[i]]+1
for i ← 1 to n-1 do //tùy thứ tự cần xếp
  B[i] ← B[i-1] + B[i]
for i ← n-1 down to 0 do
  B[A[i]] ← B[A[i]]-1
  C[B[A[i]]] ← A[i]
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

Quá trình tính toán:

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

	0	1	2	3	4
A	**3**	**2**	**5**	**1**	**4**

	0	1	2	3	4	5
B	**0**	**0**	**0**	**0**	**0**	**0**

Đếm các khóa

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

Quá trình tính toán:

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

A

i=0	1	2	3	4
3	**2**	**5**	**1**	**4**

B

0	1	2	3	4	5
0	**0**	**0**	**1**	**0**	**0**

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

<u>Quá trình tính toán</u>:

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

A

0	i=1	2	3	4
3	**2**	**5**	**1**	**4**

B

0	1	2	3	4	5
0	**0**	**1**	**1**	**0**	**0**

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

<u>**Quá trình tính toán**</u>:

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

A

0	1	i=2	3	4
3	**2**	**5**	**1**	**4**

B

0	1	2	3	4	**5**
0	**0**	**1**	**1**	**0**	**1**

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

<u>Quá trình tính toán</u>:

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

	0	1	2	i=3	4
A	**3**	**2**	**5**	**1**	**4**

	0	1	2	3	4	5
B	**0**	**1**	**1**	**1**	**0**	**1**

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

Quá trình tính toán:

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

A

0	1	2	3	i=4
3	**2**	**5**	**1**	**4**

B

0	1	2	3	4	5
0	**1**	**1**	**1**	**1**	**1**

Tính toán thứ tự

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

Quá trình tính toán:

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

A

0	1	2	3	4
3	**2**	**5**	**1**	**4**

B

0	1	2	3	4	5
0	**1**	**2**	**3**	**4**	**5**

Ghi kết quả

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

Quá trình tính toán:

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

	0	1	2	3	i=4
A	**3**	**2**	**5**	**1**	**4**
C				**4**	

	0	1	2	3	**4**	5
B	**0**	**1**	**2**	**3**	**4**-1	**5**

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

Quá trình tính toán:

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

	0	1	2	i=3	4	
A	3	2	5	1	4	
C	1			4		
	0	1	2	3	4	5
B	0	1-1	2	3	3	5

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

<u>Quá trình tính toán</u>:

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

	0	1	i=2	3	4
A	3	2	5	1	4

C	1			4	5

	0	1	2	3	4	5
B	0	0	2	3	3	5-1

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

<u>Quá trình tính toán</u>:

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

	0	i=1	2	3	4
A	**3**	**2**	**5**	**1**	**4**

C	**1**	**2**		**4**	**5**

	0	1	2	3	4	5
B	**0**	**0**	**2**-1	**3**	**3**	**4**

❖PHƯƠNG PHÁP ĐẾM

<u>Quá trình tính toán</u>:

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

	i=0	1	2	3	4
A	**3**	**2**	**5**	**1**	**4**

C	**1**	**2**	**3**	**4**	**5**

	0	1	2	**3**	4	5
B	**0**	**0**	**1**	**3**-1	**3**	**4**

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

Cài đặt: (trên mảng) Giả sử thứ tự cặp sắp xếp **<** (tăng dần)

```
void countingSort(int A[], int n, int C[]) {
  int B[MAX] ={0};  //đã khai báo MAX
  for (int i=0; i<n; i++)
     B[A[i]]++;
  for (int i=1; i<MAX; i++)
     B[i] += B[i-1];
  for (int i=n-1; i>-1; i--)
  { B[A[i]]--; C[B[A[i]]] = A[i];  }
}
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP ĐẾM

Đánh giá:

Trong mọi trường hợp, độ phức tạp tính toán của Counting Sort là **O(n+k)**, trong đó **k** là kích thước của mảng **B**.

Counting Sort là một trong những thuật toán sắp xếp không dựa vào kết quả so sánh giá trị khóa của các phần tử trong danh sách.

III. CÁC GIẢI THUẬT SẮP XẾP

❖ PHƯƠNG PHÁP CƠ SỐ

Từ khóa: Radix Sort

Điều kiện: Giá trị khóa là những giá trị rời rạc (số nguyên không âm hoặc chuỗi)

Phân tích Thứ tự của số nguyên dương là do thứ tự của các số trong cơ số theo từng hàng (đơn vị, chục, trăm, …)

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Ý tưởng:

Cho hai số **$a=a_m a_{m-1..} a_1$** và **$b=b_n b_{n-1..} b_1$**.

- **a > b** nếu chữ số tại hàng i cao nhất của hai số thỏa điều kiện: **$a_i > b_i$**.
- Chữ số ở mỗi hàng chỉ có **k** giá trị có thứ tự theo cơ số **k**. Số lớn nhất có **m** chữ số có tương ứng **m** hàng. Nếu số có **n** chữ số **(n<m)** thì chữ số tại hàng cao hơn **n** là **0**.

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

<u>Ý tưởng:</u>

- Xét các khóa theo từng hàng, từ hàng thấp nhất đến **m**:
  - Đưa mỗi phần tử vào danh sách **T[j]** tương ứng với giá trị của hàng đang xét, theo thứ tự xét phần tử
  - Nối các danh sách **T[j]** theo thứ tự thích hợp dựa trên thứ tự của các giá trị trong cơ số.

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Thuật toán: (dùng cho số nguyên dương)

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

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

```
k ← 10
while k ≤ 10^m
  for i ← 0 to n-1 do
    j ← (A[i] mod k)*10 div k
    T[j] ← T[j]∪{A[i]}
  A ← T[1] ∪ T[2] ∪ .. ∪ T[m-1]
  k ← k*10
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Quá trình tính toán:

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 10 , A={312,142,151,1,40}

Xử lý: 312

0	1	2	3	4	5	6	7	8	9
		312							

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

<u>Quá trình tính toán:</u>

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 10 , A={312,142,151,1,40}

Xử lý: 412

0	1	2	3	4	5	6	7	8	9
		312							
		412							

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

<u>Quá trình tính toán:</u>

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 10 , A={312,142,151,1,40}

Xử lý: 151

0	1	2	3	4	5	6	7	8	9
	151	312							
		412							

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Quá trình tính toán:

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 10 , A={312,142,151,1,40}

Xử lý: 1

0	1	2	3	4	5	6	7	8	9
	151	312							
	1	412							

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Quá trình tính toán:

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 10 , A={312,142,151,1,40}

Xử lý: 40

0	1	2	3	4	5	6	7	8	9
40	151	312							
	1	412							

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Quá trình tính toán:

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 10, A={312,142,151,1,40}

Nối các danh sách con được A={40, 151,1,312,412}

0	1	2	3	4	5	6	7	8	9
40	151	312							
	1	412							

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Quá trình tính toán:

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 100, A={40, 151,1,312,412}

Xử lý 40

0	1	2	3	4	5	6	7	8	9
				40					

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Quá trình tính toán:

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 100, A={40, 151,1,312,412}

Xử lý 151

0	1	2	3	4	5	6	7	8	9
				40	151				

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Quá trình tính toán:

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 100, A={40, 151,1,312,412}

Xử lý 1

0	1	2	3	4	5	6	7	8	9
1				40	151				

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Quá trình tính toán:

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 100, A={40, 151,1,312,412}

Xử lý 312

0	1	2	3	4	5	6	7	8	9
1	312			40	151				

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

<u>Quá trình tính toán:</u>

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 100, A={40, 151,1,312,412}

Xử lý 412

0	1	2	3	4	5	6	7	8	9
1	312			40	151				
	412								

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Quá trình tính toán:

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 100, A={40, 151,1,312,412}

Nối các danh sách con được A={1,312,412,40,151}

0	1	2	3	4	5	6	7	8	9
1	312			40	151				
	412								

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

<u>Quá trình tính toán:</u>

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 1000, A={1,312,412,40,151}

Xử lý 1

0	1	2	3	4	5	6	7	8	9
1									

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Quá trình tính toán:

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 1000, A={1,312,412,40,151}

Xử lý 312

0	1	2	3	4	5	6	7	8	9
1			312						

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Quá trình tính toán:

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 1000, A={1,312,412,40,151}

Xử lý 412

0	1	2	3	4	5	6	7	8	9
1			312	412					

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

<u>Quá trình tính toán:</u>

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 1000, A={1,312,412,40,151}

Xử lý 40

0	1	2	3	4	5	6	7	8	9
1			312	412					
40									

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Quá trình tính toán:

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 1000, A={1,312,412,40,151}

Xử lý 151

0	1	2	3	4	5	6	7	8	9
1	151		312	412					
40									

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Quá trình tính toán:

Giả sử A={312,142,151,1,40}, m = 3, k=10 và thứ tự cần sắp xếp < (tăng dần)

k = 1000, A={1,312,412,40,151}

Nối các danh sách con được A={1,40,151,312,412}

0	1	2	3	4	5	6	7	8	9
1	151		312	412					
40									

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Cài đặt: (dùng danh sách liên kết) Giả sử sắp xếp danh sách số nguyên dương trong hệ thập phân theo thứ tự **<**

III. CÁC GIẢI THUẬT SẮP XẾP

```
void radixSort(List &A) {
  List rad[10];
  for (int i = 0; i < 10; i++)
    createList(rad[i]);
  int k = 10;
  for (int m=0;m<MAX;m++){ //MAX: số chữ số
    while(A.pHead) {
      Node *p = removeHead(A);
      addTail(rad[(p->info %k)*10/k], p);
    }
    A.pHead = A.pTail = NULL;
```

III. CÁC GIẢI THUẬT SẮP XẾP

```
    for (int i=0; i<10; i++)
        if (rad[i].pHead) {
            if (A.pHead)
                A.pTail->pNext = rad[i].pHead;
            else
                A.pHead = rad[i].pHead;
            A.pTail = rad[i].pTail;
            rad[i].pHead = rad[i].pTail = NULL;
        }
    k *= 10;
  }
}
```

III. CÁC GIẢI THUẬT SẮP XẾP

❖PHƯƠNG PHÁP CƠ SỐ

Đánh giá:

- Radix Sort cũng là một trong những thuật toán sắp xếp không dựa trên kết quả so sánh giá trị khóa giữa các phần tử trong danh sách.
- Độ phức tạp tính toán là **O(m.n)** trong đó **m** là số ký tự lớn nhất của một phần tử trong danh sách.
- Radix Sort thích hợp với cấu trúc là danh sách liên kết hơn là dùng cấu trúc mảng

III. CÁC GIẢI THUẬT SẮP XẾP

❖BÀI TẬP

Trong các thuật toán sắp xếp Selection Sort, Insertion Sort, Counting Sort và Radix Sort, thuật toán nào là sắp xếp ổn định (Stable)? Vì sao?

III. CÁC GIẢI THUẬT SẮP XẾP

❖BÀI TẬP

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) Selection Sort

b) Insertion Sort

III. CÁC GIẢI THUẬT SẮP XẾP

❖ BÀI TẬP

Cài đặt hàm hexSort(List &A) để sắp xếp theo cơ số cho dãy A chứa các số thập lục phân. Thứ tự sắp xếp là giảm dần.

III. CÁC GIẢI THUẬT SẮP XẾP

❖ BÀI TẬP

Định nghĩa cấu trúc dữ liệu lưu trữ thông tin máy tính gồm: nhãn hiệu máy, tốc độ xử lý (tính theo GHz) và giá bán. Cài đặt các hàm sau

a) *sortByName(…)* Sắp xếp danh sách máy theo thứ tự tăng dần đối với nhãn hiệu

b) *sortBySpeed(…)* Sắp xếp danh sách máy theo thứ tự giảm dần đối với tốc độ xử lý

c) *sort(…)* Sắp xếp danh sách máy theo thứ tự tăng dần giá bán và trong trường hợp cùng giá thì xếp theo thứ tự giảm dần tốc độ xử lý

III. CÁC GIẢI THUẬT SẮP XẾP

❖ BÀI TẬP

d)filter(…) Lọc danh sách các máy tính có giá trong đoạn [p_1, p_2] và tốc độ xử lý trong đoạn [s_1, s_2].