



CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

CÁC GIẢI THUẬT TÌM KIẾM – SẮP XẾP



MỤC TIÊU

- ❖ 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 IV

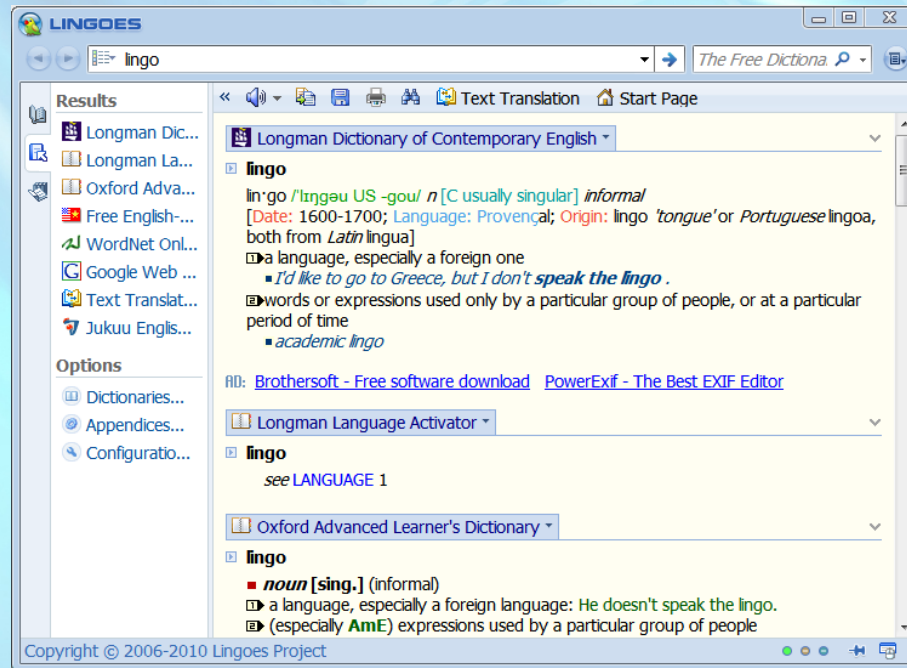
- I. NHU CẦU TÌM KIẾM, SẮP XẾP**
- II. CÁC GIẢI THUẬT TÌM KIẾM**
- III. CÁC GIẢI THUẬT SẮP XẾP**
- IV. CẤU TRÚC HÀNG ĐỢI ƯU TIÊN**



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



ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
THƯ VIỆN



TRANG CHỦ | GIỚI THIỆU | DỊCH VỤ | TRA CỨU OPAC | TÀI NGUYÊN | LIÊN HỆ

Tra cứu thư mục trực tuyến

Bắt kỳ ▾

mạng xã hội

Q

:


Tên thư viện	Số biểu ghi	Trạng thái
» Thư viện Trường Đại học Công nghệ Thông tin ĐHQG-HCM	28	Kết nối thành công

#	Mô tả	Tác giả chính	Chọn
1	Phân tích mạng xã hội và ứng dụng : Giáo trình / Đỗ Phúc . - HCM : Đại học Quốc gia TP.HCM , 2017	Đỗ, Phúc	<input type="checkbox"/>
005.7 Đ 450 PH			
2	Mạng xã hội địa điểm trên Android Nguyễn Thành Luân, , TS. Nguyễn Anh Tuấn , , 2012	Nguyễn Thành Luân	<input type="checkbox"/>

SÁCH THEO KHO

Kho đọc	959
Kho mượn - Giáo trình	66
Kho mượn - Tham Khảo	88
Khóa luận	4
Luận văn	12
Ưu tiên	5

MƯỢN NHIỀU

 Outcomes Pre-



- 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, \dots, 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 $A = \{a_0, a_1, \dots, a_{n-1}\}$ 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

$x=9$



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 (lính 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)

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 (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;  
    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);  
    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 $A = \{a_0, a_1, \dots, a_{n-1}\}$ đã có thứ tự \mathcal{R}

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 \mathfrak{R} , 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 **\mathbb{R}** , 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 > 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

Quá trình tính toán:

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

$l=0$ $m=1$ $r=2$ 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 \mathfrak{R} 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 $(l+r) \text{ div } 2 \rightarrow$ số lần lặp là 1 \rightarrow 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 \rightarrow số lần lặp khoảng $\log_2(n)+1 \rightarrow$ 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 $A = \{a_0, a_1, \dots, a_{n-1}\}$ đã có thứ tự \mathcal{R} và giá trị khóa được rải đều trên danh sách.

Phân tích: Giá trị khóa rải đều trên danh sách \rightarrow 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 kiếm 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 **\mathcal{R}** , 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 > 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

Quá trình tính toán:

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 \mathfrak{R} 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 \rightarrow số lần lặp là 1 \rightarrow 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 \rightarrow tìm tuyến tính \rightarrow 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 .