UIT
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Data Structures & Algorithms

TÌM KIẾM

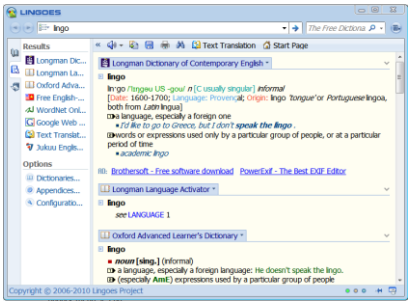
Mục Tiêu

- **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

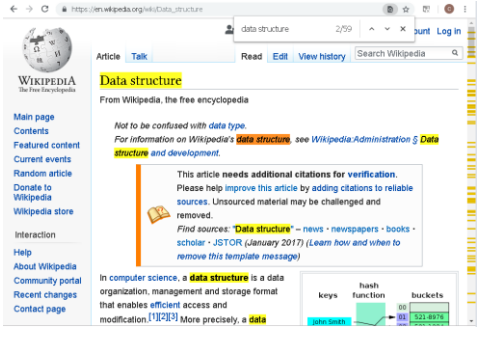
1. Nhu cầu tìm kiếm/sắp xếp.
2. Bài toán tìm kiếm.
3. Tìm kiếm tuyến tính (Linear Search)
4. Tuyến tính cải tiến
5. Tìm kiếm nhị phân (Binary Search).
6. Tìm kiến nội suy (Interpolation Search)

Nhu cầu tìm kiếm thông tin



Nhu cầu tìm kiếm thông tin

Nhu cầu tìm kiếm thông tin



Nhu cầu tìm kiếm thông tin

❖ Các yêu cầu trong một hệ thống thông tin:
- Lưu trữ
- **Tra cứu (tìm kiếm) →thường thực hiện nhất**
- Tính toán
- Báo biểu

❖ Dữ liệu đã được sắp xếp => tìm nhanh hơn.

=> Vấn đề:
- Tìm kiếm nhanh.
- Sắp xếp nhanh.

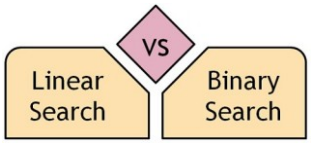
Bài toán tìm kiếm - Searching

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***

Các Phương Pháp Tìm Kiếm Nội

1. Tìm kiếm tuyến tính
2. Tìm kiếm nhị phân

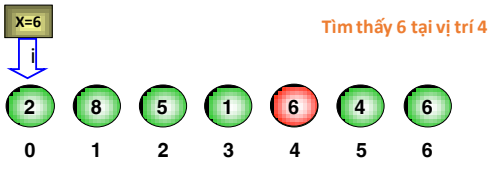


Tìm kiếm tuyến tính

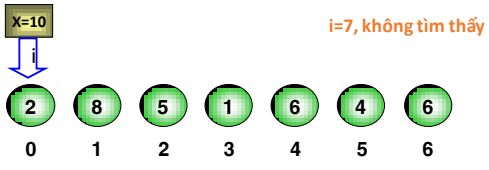
✓ Ý tưởng:

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

Tìm kiếm tuyến tính



Tìm kiếm tuyến tính



Tìm kiếm tuyến tính

Từ khóa: Linear Search

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

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, và trả về i nếu tồn tại a_i.

Tìm kiếm tuyến tính

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 $i=-1$

Thuật toán:

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

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

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

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

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

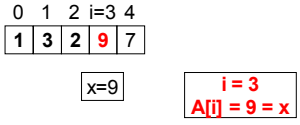
x=9

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



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;
}
```

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;
}
```

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).

Tìm kiếm tuyến tính CẢI TIẾN

Điều kiện dừng là gì ?

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

23

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

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**

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***

```
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
```

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;
}
```

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;
}
```

Tìm kiếm tuyến tính – Lưu ý

- Thuật toán tìm tuyến tính sẽ duyệt tất cả các đối tượng trên "không gian tìm kiếm" để tìm ra đối tượng thỏa mãn "điều kiện tìm kiếm".
- Thông thường trong một bài toán kỹ thuật lập trình thì "không gian tìm kiếm" đơn giản nhất là: **mảng một chiều**, **ma trận**, một đoạn giá trị nào đó, **danh sách liên kết**, **cây**,…

29

Tìm kiếm tuyến tính – Lưu ý

- Mặt khác **"điều kiện tìm kiếm"** là tiêu chuẩn tìm kiếm được trình bày dưới dạng **một phát biểu không hình thức** và người lập trình phải **hình thức hóa nó bằng một "biểu thức logic"** trong chương trình.

30

Tìm kiếm tuyến tính – Lưu ý

- Thuật toán tìm kiếm tuyến tính sẽ duyệt toàn bộ không gian tìm kiếm để tìm đối tượng thỏa mãn tiêu chuẩn tìm kiếm **nên các đối tượng này không cần được sắp thứ tự**. Nói một cách khác là **dữ liệu không cần được tổ chức**.

31

Tìm kiếm tuyến tính

Bài 1. Viết hàm tìm kiếm giá trị x trong mảng 1 chiều các số thực trả về tìm thấy hay không.

Bài 2. Viết hàm tìm vị trí giá trị nhỏ nhất trong mảng một chiều các số thực.

Bài 3. Viết hàm tìm tất cả các vị trí có giá trị nhỏ nhất trong mảng một chiều các số thực.

Bài tập: Viết chương trình với menu cho người dùng lựa chọn nhập một mảng và lựa chọn các chức năng được viết ra ở bài 1,2,3 ở trên.

32

Đặt vấn đề

Đối với những dãy số đã *có thứ tự* (giả sử tăng),

$a_{i-1} \le a_i \le a_{i+1}$

❖ **NX**: nếu $x > a_i$ thì x thuộc $[a_{i+1}, a_{n-1}]$

nếu $x < a_i$ thì x thuộc $[a_0, a_{i-1}]$

→ Giới hạn phạm vi tìm kiếm sau mỗi lần so sánh x

33

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.

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**.

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***

Tìm kiếm nhị phân – Thuật giải

Input: mảng a: a[0], a[1],…,a[n-1] đã có thứ tự và giá trị x

Output: vị trí của khóa x tìm được, trả về -1 nếu không tìm thấy

Các bước thực hiện:

✓ Bước 1: left = 0; right = n -1;

✓ Bước 2:

- mid = (left+right)/2; *// lấy mốc so sánh*
- So sánh a[mid] với x, có 3 khả năng :
  - a[mid] = x: Tìm thấy. Dừng
  - a[mid] > x: *//tìm tiếp x trong dãy con* $a_{left} .. a_{mid-1}$
    right =midle - 1;
  - a[mid] < x: *//tìm tiếp x trong dãy con* $a_{mid+1} .. a_{right}$
    left = mid + 1;

37

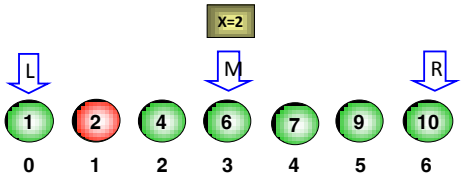
Tìm kiếm nhị phân – Thuật giải

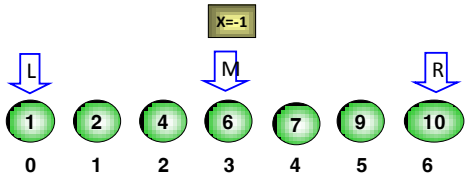
✓ Bước 3:

- Nếu left ≤ right //còn phần tử chưa xét, tìm tiếp.
  Lặp lại Bước 2.
- Ngược lại: Dừng; //Đã xét hết tất cả các phần tử.

38

Tìm thấy 2 tại vị trí 1





L=0

R=-1 => không tìm thấy X=-1

Tìm kiếm nhị phân

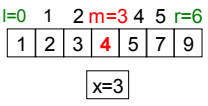
Thuật toán:

```
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
```

Tìm kiếm nhị phân

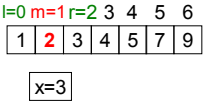
Quá trình tính toán:

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



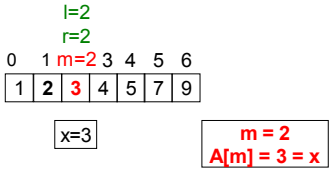
Tìm kiếm nhị phân

Quá trình tính toán:
Giả sử ***A = {1,2,3,4,5,7,9}***, thứ tự ℜ là **<**, phần tử cần tìm ***x = 3***



Tìm kiếm nhị phân

Quá trình tính toán:
Giả sử ***A = {1,2,3,4,5,7,9}***, thứ tự ℜ là **<**, phần tử cần tìm ***x = 3***



Tìm kiếm nhị phân

Cài đặt: (trên mảng, thứ tự **<**)

```
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;
}
```

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.

Tìm kiếm nhị phân

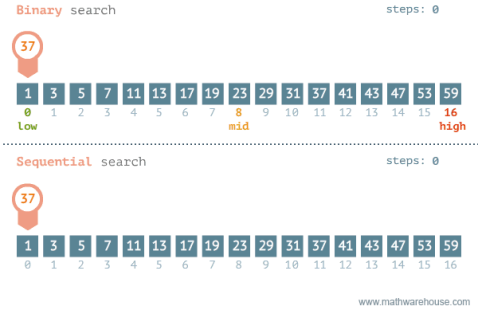
Đánh giá:

- *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))**.

Tìm kiếm nhị phân – Lưu ý

Điều kiện tiên quyết để áp dụng thuật toán tìm nhị phân là các đối tượng trong "không gian tìm kiếm" phải được sắp xếp thứ tự theo một tiêu chuẩn nào đó và ta sẽ dựa trên tiêu chuẩn này để tìm.

Tìm kiếm



Bài tập – TÌM KIẾM NHỊ PHÂN

1. Anh/chị hãy viết hàm cài đặt thuật toán nhị phân trên mảng số nguyên có thứ tự giảm dần bằng ngôn ngữ C/C++.

2. Trình bày các bước (vẽ từng bước) theo hàm đã cài đặt ở câu 1 thực hiện tìm giá trị
X=50 trong mảng các số nguyên có giá trị như sau: 90 80 60 26 24 12

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.

Tìm kiếm nội suy

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

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***

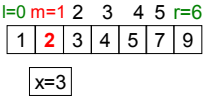
Tìm kiếm nội suy

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
```

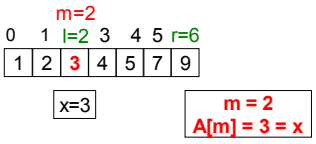
Tìm kiếm nội suy

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



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***



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;
}
```

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?

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.

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.

Slide được tham khảo từ

- **Slide được tham khảo từ:**
  - Slide CTDL GT, Khoa Khoa Học Máy Tính, ĐHCNTT
  - Slide CTDL GT, Thầy Nguyễn Tấn Trần Minh Khang, ĐH CNTT
  - Slid CTDL GT, Cô Trần Thị Thương, ĐH CNTT
  - congdongcviet.com
  - cplusplus.com

61

62