

SEARCHING ARRAY

DATA STRUCTURES AND ALGORITHMS

ThS Nguyễn Thị Ngọc Diễm
diemntn@uit.edu.vn



- Nhu cầu tìm kiếm và sắp xếp
- Một số thuật toán tìm kiếm
- Thuật toán tìm kiếm tuyến tính - Linear Search
- Thuật toán tìm kiếm tuyến tính cải tiến - Sentinel Linear Search
- Thuật toán tìm kiếm nhị phân - Binary Search
- Thuật toán tìm kiếm nội suy – Interpolation Search

Nhu cầu Tìm kiếm và Sắp xếp



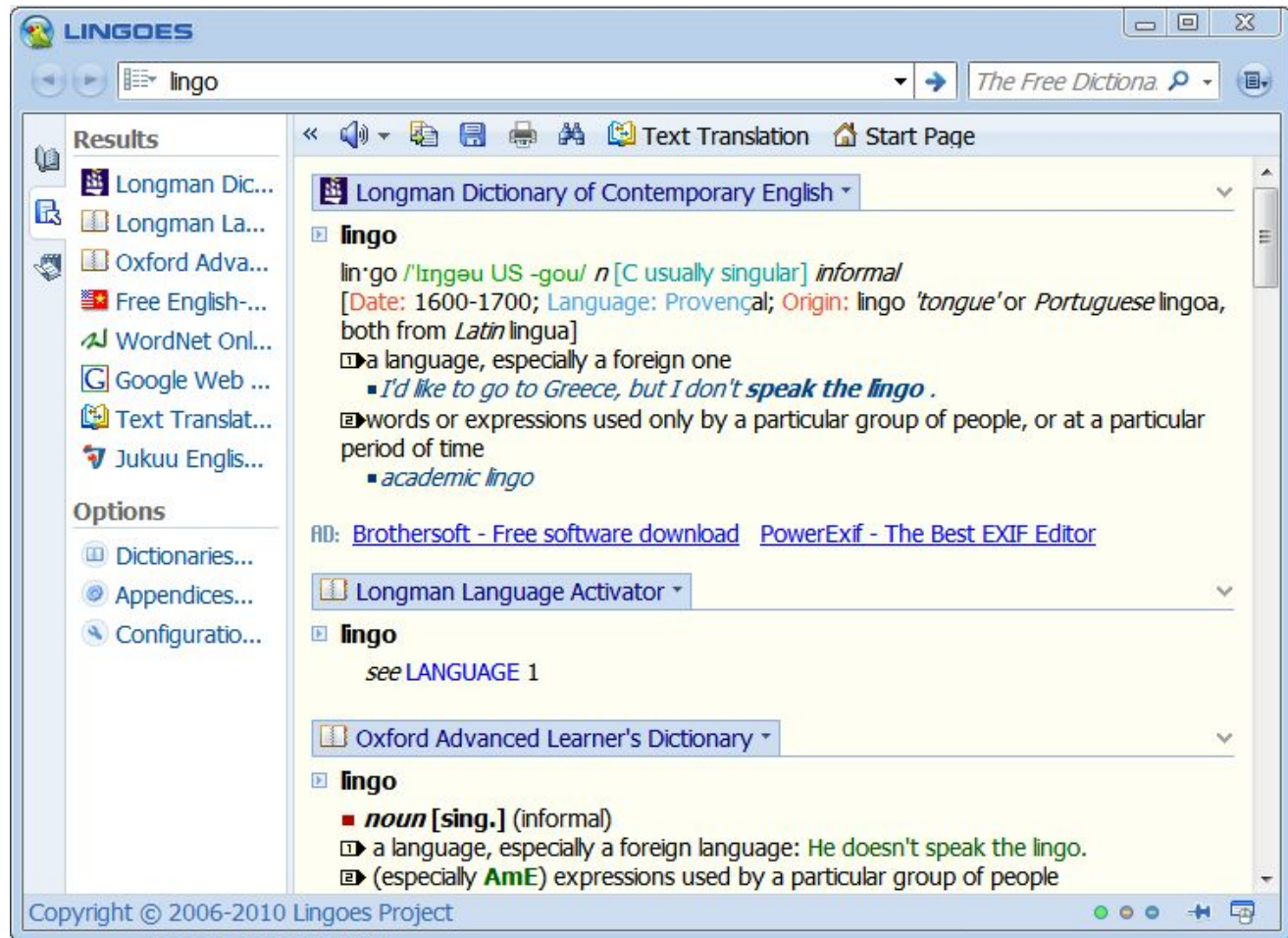
- Trong thực tế, khai thác dữ liệu hầu như lúc nào cũng phải thực hiện thao tác tìm kiếm.
- Việc tìm kiếm nhanh hay chậm tùy thuộc vào trạng thái và trật tự của dữ liệu trên đó.
- Để tìm kiếm dữ liệu dễ dàng và nhanh chóng, trước khi thao tác thì dữ liệu trên mảng và tập tin đã có thứ tự.
- Thao tác sắp xếp dữ liệu là một trong những thao tác cần thiết.

Nhu cầu Tìm kiếm và Sắp xếp



❖ TRA CỨU THÔNG TIN


- Từ điển



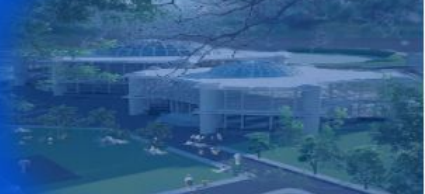



❖ TRA CỨU THÔNG TIN

- 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

Bất 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 checked="" 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 Outcomes Pre-



❖ TRA CỨU THÔNG TIN

- Soạn thảo, tra cứu văn bản

The screenshot shows the Wikipedia page for 'Data structure'. The URL is https://en.wikipedia.org/wiki/Data_structure. The page title is 'Data structure'. Below the title, it says 'From Wikipedia, the free encyclopedia'. There is a note: 'Not to be confused with *data type*. For information on Wikipedia's *data structure*, see *Wikipedia:Administration § Data structure and development*.' A warning box states: 'This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. Find sources: "Data structure" – news · newspapers · books · scholar · JSTOR (January 2017) (Learn how and when to remove this template message)'.

In computer science, a **data structure** is a data organization, management and storage format that enables efficient access and modification.^{[1][2][3]} More precisely, a **data**

The diagram illustrates a hash table structure. It shows a 'keys' column with the value 'John Smith', a 'hash function' column, and a 'buckets' column. The 'hash function' column has a value '00' and an arrow pointing to the 'buckets' column. The 'buckets' column has a value '01' and a value '521-8976'.



◆ 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.



Một số thuật toán tìm kiếm trên mảng

- Cho danh sách có n phần tử $a_0, a_1, a_2, \dots, a_{n-1}$.
- Để đơn giản trong việc trình bày giải thuật ta dùng mảng 1 chiều a để lưu danh sách các phần tử nói trên trong bộ nhớ chính.
- Các thuật toán tìm kiếm trên mảng 1 chiều:
 - **Giải thuật tìm kiếm tuyến tính (Linear Search) $\Rightarrow O(n)$**
 - Unordered Linear Search
 - Ordered Linear Search
 - Giải thuật tìm kiếm Jump Search
 - **Giải thuật tìm kiếm nhị phân (Binary Search) $\Rightarrow O(\log n)$**
 - Giải thuật tìm kiếm tam phân (Ternary Search)
 - **Giải thuật tìm kiếm nội suy (Interpolation Search) $O(\log \log n) / O(n)$**
 - ...



Nội dung

- Nhu cầu tìm kiếm và sắp xếp
- Một số thuật toán tìm kiếm
- **Thuật toán tìm kiếm tuyến tính - Linear Search**
- Thuật toán tìm kiếm tuyến tính cải tiến - Sentinel Linear Search
- Thuật toán tìm kiếm nhị phân - Binary Search
- Thuật toán tìm kiếm nội suy – Interpolation Search

Linear Search on Unordered Array



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: Ý tưởng : So sánh x lần lượt với phần tử thứ 1, thứ 2, ... của mảng a cho đến khi gặp được khóa cần tìm, hoặc tìm hết mảng mà không thấy.

Linear Search on Unordered array



Thuật toán:

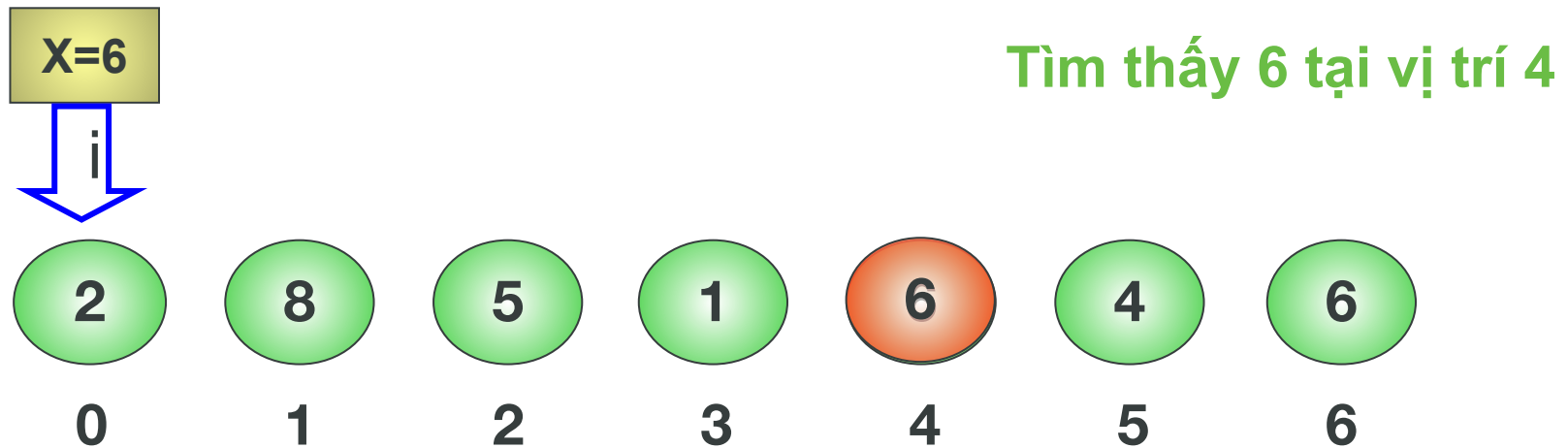
Input: Danh sách A có n phần tử, giá trị khóa x cần tìm.

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

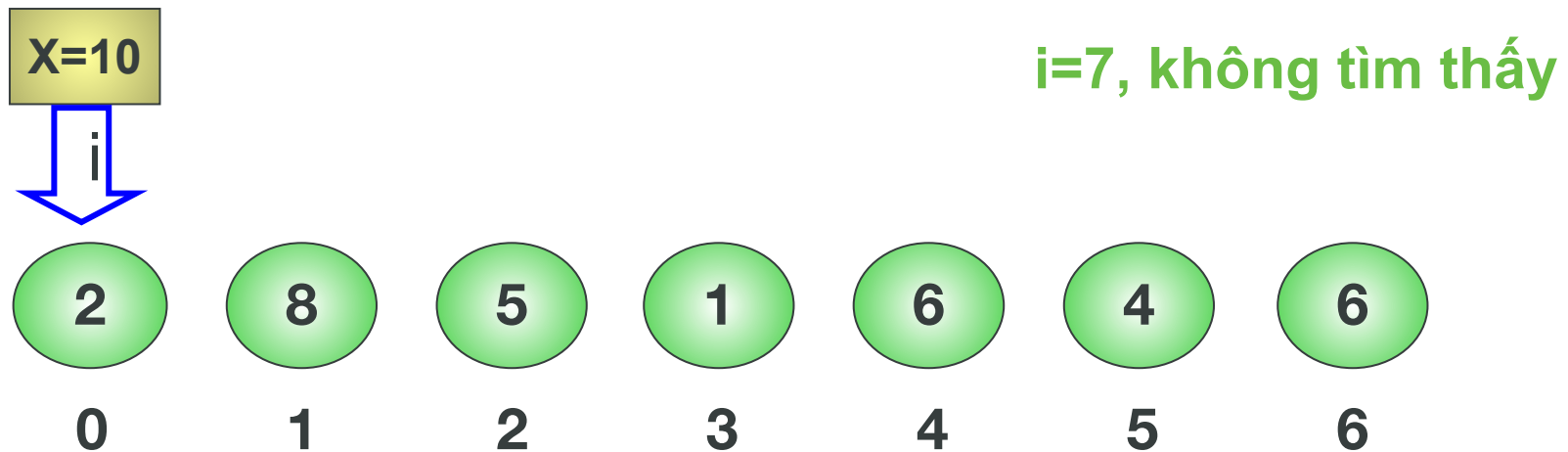
• Các bước tiến hành

- Bước 1: Khởi gán $\text{index}=0$;
- Bước 2: So sánh $a[\text{index}]$ với giá trị x cần tìm, có 2 khả năng
+ $a[\text{index}] == \text{value}$ tìm thấy value. Dừng; // return index;
+ $a[\text{index}] != \text{value}$ sang bước 3;
- Bước 3: $\text{index}=\text{index}+1$ // Xét tiếp phần tử kế tiếp trong mảng
Nếu $\text{index}==N$: Hết mảng. Dừng; // return -1;
Ngược lại: Lặp lại bước 2;

Minh Họa Thuật Toán Tìm Kiếm Tuyến Tính



Minh Họa Thuật Toán Tìm Kiếm Tuyến Tính (tt)



Unordered Linear Search



Cài đặt: (trên mảng 1 chiều)

// Searches an unordered array of integers

```
int linearsearch(int data[], int size, int
value) {
    int index = 0;
    while (index < size && data[index] != value)
        index++;

    if(index <= size) return index;
    return -1;
}
```

Unordered Linear Search



```
#include <iostream>

using namespace std;

int main() {
    const int array_size = 7;
    int list[array_size] = { 2, 8, 5, 1, 6, 4, 6 };
    int search_value;
    cout << "Enter search value: ";
    cin >> search_value;
    cout << search(list, array_size, search_value)
    << endl;
    return 0;
}
```

Unordered Linear Search



- Cài đặt: (trên danh sách liên kết đơn)

•

Ordered Linear Search



- Search an ordered array of integers for a value and return its index if the value is found; Otherwise, return -1.
- Linear search can stop immediately **when it has passed the possible position of the search value.**

Ordered Linear Search



```
// Searches an ordered array of integers
int lsearch(int data[], // input: array
            int size, // input: array size
            int value // input: value to find
            ) { // output: index if found
    for (int index = 0; index < size; index++) {
        if (data[index] > value) // cho trg hop order array
            return -1;
        else if (data[index] == value)
            return index;
    }
    return -1;
}
```

Ordered Linear Search



```
#include <iostream>

using namespace std;

int main() {
    const int array_size = 8;
    int list[array_size] = {1, 2, 3, 5, 7, 10, 14, 17};
    int search_value;
    cout << "Enter search value: ";
    cin >> search_value;
    cout << lsearch(list, array_size, search_value)
    << endl;
    return 0;
}
```

Đánh giá Thuật toán tìm Tuyến tính



Trường hợp	Css
Tốt nhất	1
Xấu nhất	N
Trung bình	$(N+1) / 2$

□ Độ phức tạp $O(N)$



- Nhu cầu tìm kiếm và sắp xếp
- Một số thuật toán tìm kiếm
- Thuật toán tìm kiếm tuyến tính - Linear Search
- **Thuật toán tìm kiếm tuyến tính cải tiến - Sentinel Linear Search**
- Thuật toán tìm kiếm nhị phân - Binary Search
- Thuật toán tìm kiếm nội suy – Interpolation Search

Thuật toán Tìm kiếm tuyến tính (cải tiến)



Cài đặt: (trên mảng 1 chiều)

```
int Sentinel_LinearSearch(int A[], int N, int X) {  
    int last=A[N-1]; // Lưu lại giá trị cuối danh sách  
  
    A[N-1]=X; // Đặt giá trị X vào cuối danh sách  
  
    int i=0;  
    while(A[i]!=X) i++;  
  
    // trả lại giá trị ban đầu cho biến cuối a[N-1]  
    A[N-1]=last;  
  
    if (i<N-1 || A[N-1]==X) return i;  
  
    return -1;  
}
```

Thuật toán Tìm kiếm tuyến tính (cải tiến)



Cài đặt: (trên danh sách liên kết đơn)

```
NODE* Sentinel_LinearSearch(List A, int x) {  
    NODE *p = A.pHead, *temp = new CreateNode(x);  
    AddTail(A, temp);  
    while (p->info != x) p = p->pNext;  
    if (p == A.pTail) return p;  
    else return NULL;  
}
```



Nội dung

- Nhu cầu tìm kiếm và sắp xếp
- Một số thuật toán tìm kiếm
- Thuật toán tìm kiếm tuyến tính - Linear Search
- Thuật toán tìm kiếm tuyến tính cải tiến - Sentinel Linear Search
- **Thuật toán tìm kiếm nhị phân - Binary Search**
- Thuật toán tìm kiếm nội suy – Interpolation Search



- Binary search được áp dụng trên **mảng đã có thứ tự**.
- Binary search dựa trên chiến lược “Chia để trị” (divide and conquer):
 - Bắt đầu tìm kiếm tại phần tử giữa của mảng.
 - Nếu giá trị tại phần tử giữa bằng giá trị cần tìm thì kết thúc.
 - Nếu giá trị tại phần tử giữa nhỏ hơn giá trị cần tìm thì loại bỏ nửa đầu tiên của mảng trong lần xét kế tiếp.
 - Nếu giá trị tại phần tử giữa lớn hơn giá trị cần tìm thì loại bỏ nửa sau của mảng trong lần xét kế tiếp.
 - Lặp lại quá trình cho tới khi phần tử được tìm thấy, hoặc cho tới khi toàn bộ mảng được loại bỏ.



Từ khóa: Binary Search

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



Thuật toán:

Input: Danh sách A có n phần tử đã có thứ tự \mathfrak{R} , giá trị khóa x cần tìm.

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

Binary Search



- Giả sử dãy tìm kiếm hiện hành bao gồm các phần tử nằm trong a_{first} , a_{last} , các bước của giải thuật như sau:
- Bước 1: $\text{first} = 0$; $\text{last} = N-1$;
- Bước 2:
 - $\text{middle} = (\text{first} + \text{last}) / 2$; *// chỉ số phần tử giữa dãy hiện hành*
 - So sánh $a[\text{middle}]$ với value. Có 3 khả năng:
 - $a[\text{middle}] = \text{value}$: tìm thấy. Dừng; *// return middle*
 - $a[\text{middle}] > \text{value}$: $\text{last} = \text{middle} - 1$;
 - $a[\text{middle}] < \text{value}$: $\text{first} = \text{middle} + 1$;
- Bước 3: Nếu $\text{first} \leq \text{last}$; *// còn phần tử trong dãy hiện hành*
+ Lặp lại bước 2
Ngược lại: Dừng; *// return -1*;

Binary Search



Cài đặt: (Trên mảng 1 chiều)

```
// Searches an ordered array of integers
```

```
int bsearch(int a[], // input: array
            int size, // input: array size
            int value // input: value to find
            ) { // output: if found, return index // otherwise, return -1
    int first, middle, last;
    first = 0;
    last = size - 1;
    while (true) {
        if (first > last) return -1;
        middle = (first + last) / 2;
        if (a[middle] == value) return middle;
        else if (value < a[middle]) last = middle - 1;
        else first = middle + 1;
    }
}
```

Binary Search



```
#include <iostream>

using namespace std;

int main() {
    const int array_size = 7;
    int list[array_size] = { 1,2,4,6,7,9,10 };
    int search_value;
    cout << "Enter search value: ";
    cin >> search_value;
    cout << bsearch(list, array_size, search_value)
    << endl;
    return 0;
}
```



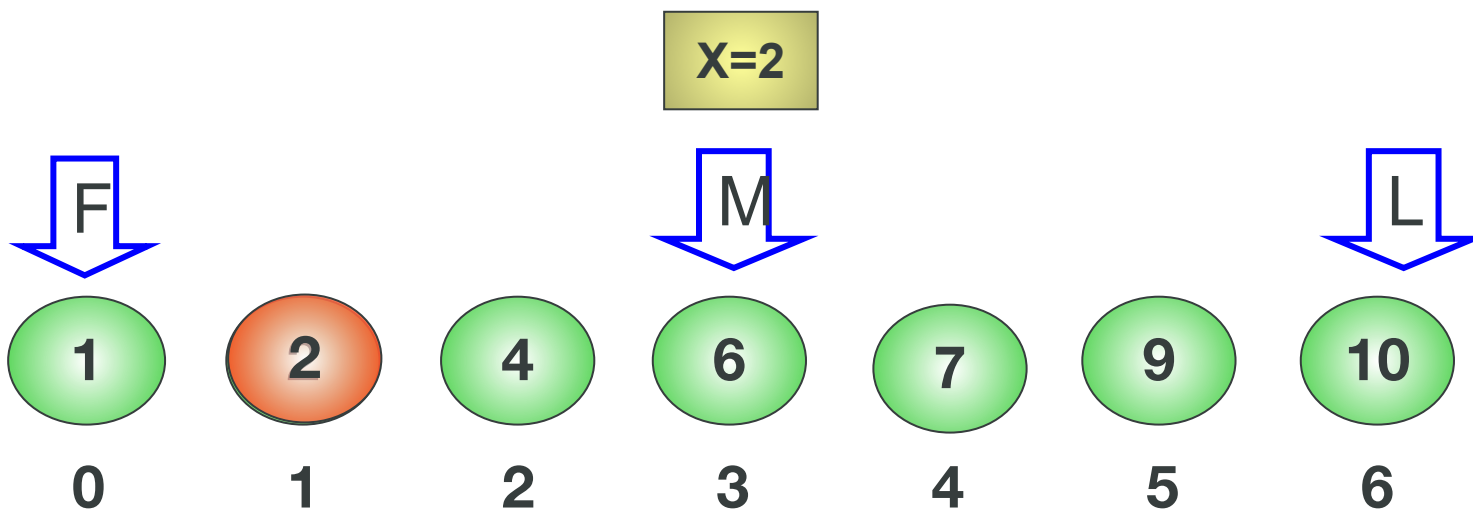
Cài đặt: (trên danh sách liên kết)

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

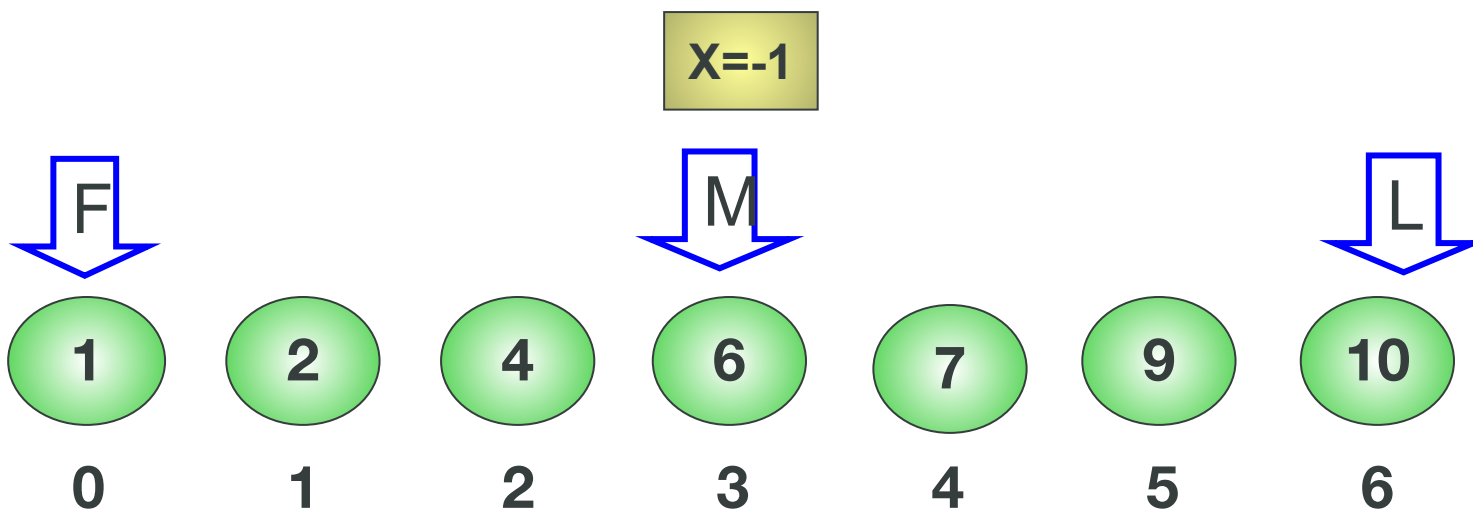
Minh họa Thuật toán Tìm kiếm nhị phân



Tìm thấy 2 tại vị trí 1



Minh họa Thuật toán Tìm kiếm nhị phân (tt)



$F=0$

$L=-1 \Rightarrow$ không tìm thấy

$X=-1$

Binary Search (Dùng đệ quy)



```
int RecBinarySearch (T M[], int First, int Last, T X)
{ if (First > Last)
    return (-1);
  int Mid = (First + Last)/2;
  if (X == M[Mid])
    return (Mid);
  if (X < M[Mid])
    return(RecBinarySearch(M, First, Mid - 1, X));
  else
    return(RecBinarySearch(M, Mid + 1, Last, X));
}
```

//=====

```
int BinarySearch (T M[], int N, T X)
{ return (RecBinarySearch(M, 0, N - 1, X));
}
```

Độ phức tạp Binary Search



Trường hợp	Css
Tốt nhất	1
Xấu nhất	$\log_2 N$
Trung bình	$\log_2 N$

□ Độ phức tạp $O(\log_2 N)$



- Nhu cầu tìm kiếm và sắp xếp
- Một số thuật toán tìm kiếm
- Thuật toán tìm kiếm tuyến tính - Linear Search
- Thuật toán tìm kiếm tuyến tính cải tiến - Sentinel Linear Search
- Thuật toán tìm kiếm nhị phân - Binary Search
- **Thuật toán tìm kiếm nội suy – Interpolation Search**



Từ khóa: Interpolation Search

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

Phân tích: Giá trị khóa rải đều trên danh sách \square 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.

Interpolation search



• Ý tưởng:

- Xét mảng a có chỉ số bắt đầu là l , kết thúc là r .
- Thay vì xác định điểm $m = l + \frac{(r-l)}{2}$ như trong tìm kiếm nhị phân, ta xác định m trong tìm kiếm nội suy như sau:

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

- Các bước còn lại tương tự như tìm kiếm nhị phân.

Giải thích công thức tính m :

$a[r] - a[l]$ (khoảng chênh lệch giá trị đầu và cuối) $\rightarrow r - l$ (khoảng chênh lệch chỉ số đầu và cuối)

$x - a[l]$ (khoảng chênh lệch giá trị đầu và x) $\rightarrow ? = \frac{(r-l)(x-a[l])}{a[r]-a[l]}$ (khoảng chênh lệch chỉ số đầu và x)

Vì danh sách bắt đầu từ $l \Rightarrow m = l + \frac{(r-l)(x-a[l])}{a[r]-a[l]}$



Thuật toán:

Input: Danh sách A có n phần tử đã có thứ tự \mathfrak{R} , giá trị khóa x cần tìm.

Output: 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:

$l \leftarrow 0, r \leftarrow n-1$

while $l \leq r$

$m \leftarrow l + ((r-l) * (x - A[l]) / (A[r] - A[l]))$

 if $x = A[m]$ then return m end if

 if $x \lessgtr A[m]$ then $r \leftarrow m - 1$

 else $l \leftarrow m + 1$ end if

end while

return -1

Interpolation search



Quá trình tính toán:

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

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

$x =$
3

Interpolation search



Quá trình tính toán:

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

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

$x =$
3

$m = 2$
 $A[m] = 3 = x$

Interpolation search



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

```
int interpolation_search(int* a, int n, int x) {
    int l = 0, r = n - 1;
    while (l <= r && a[l] <= x && x <= a[r]) {
        int rangeDelta = a[r] - a[l];
        int indexDelta = r - l ;
        int valueDelta = x - a[l];

        if (valueDelta < 0) return -1;
        if (rangeDelta <= 0) return a[l] == x ? l : -1;
        float frac= float(indexDelta) / rangeDelta ;
        int m = l + floor(valueDelta * frac);

        if (a[m] < x) l = m+1;
        else if (a[m] > x) r = m-1;
        else return m;
    }
    return -1;
}
```



Đánh giá:

- Trường hợp tốt nhất: phần tử cần tìm ở đúng vị được nội suy \square số lần lặp là 1 \square 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 \square tìm tuyến tính \square độ phức tạp **$O(n)$** .

Vd: Tìm số 10 trong dãy $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 100\}$;

- Trường hợp trung bình: độ phức tạp **$O(\log \log(n))$** .



1. Trình bày tư tưởng của các thuật toán tìm kiếm: Tuyến tính, Nhị phân? Các thuật toán này có thể được vận dụng trong các trường hợp nào? Cho ví dụ?

2. Cài đặt lại thuật toán tìm tuyến tính bằng các cách:

- Sử dụng vòng lặp for
- Sử dụng vòng lặp do ... while

Có nhận xét gì cho mỗi trường hợp?

3. Trong trường hợp các phần tử của dãy đã có thứ tự giảm, hãy trình bày và cài đặt lại thuật toán tìm nhị phân trong hai trường hợp: Đệ quy và Không đệ quy?



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?



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 .

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 .



Chúc các em học tốt!

