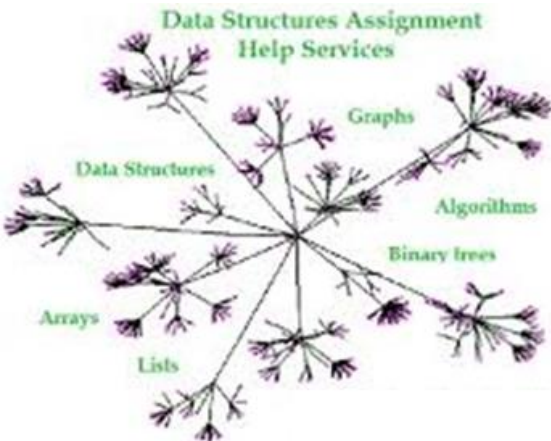
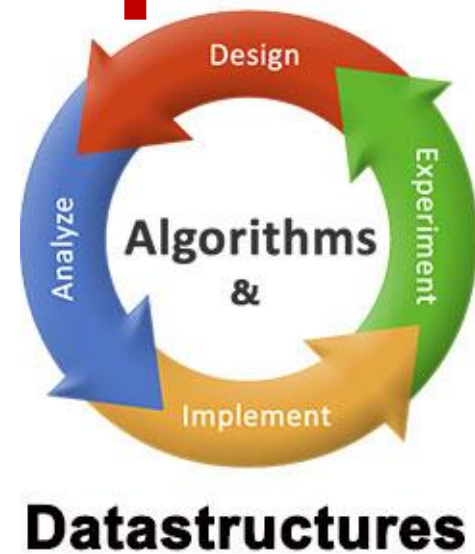


# CẤU TRÚC DỮ LIỆU & GIẢI THUẬT



Lê Văn Hạnh

levanhanhvn@gmail.com

# NỘI DUNG MÔN HỌC

- Chương 1: Ôn tập ngôn ngữ lập trình C
- Chương 2: Kiểu dữ liệu con trỏ
- Chương 3: Tổng quan về cấu trúc dữ liệu và giải thuật
- Chương 4: Danh sách kê (Danh sách tuyến tính)
- **Chương 5: Các giải thuật tìm kiếm trên danh sách kê**
- Chương 6: Các giải thuật sắp xếp trên danh sách kê
- Chương 7: Danh sách liên kết động (*Linked List*)
- Chương 8: Ngăn xếp (*Stack*)
- Chương 9: Hàng đợi (*Queue*)
- Chương 10: Cây nhị phân tìm kiếm (*Binary Search Tree*)
- Chương 11: Cây cân bằng (*Binary Search Tree*)
- Chương 12: Bảng băm (*Hash Table*)



## Chương 5

# CÁC GIẢI THUẬT TÌM KIẾM TRÊN DANH SÁCH KÈ

## MỤC TIÊU

- i. Hiểu và giải thích được các giải thuật tìm kiếm phần tử trên mảng 1 chiều
- ii. Cài đặt thành công các giải thuật tìm kiếm phần tử trên mảng 1 chiều

## NỘI DUNG CHƯƠNG 3

1. Giới thiệu bài toán tìm kiếm
2. Tìm kiếm tuyến tính
3. Tìm kiếm nhị phân

# 1. GIỚI THIỆU BÀI TOÁN TÌM KIẾM

Nhu cầu tìm kiếm trên máy tính khi dữ liệu được lưu trữ trong mảng 1 chiều

- Cho mảng 1 chiều gồm  $n$  phần tử  $a_0, a_1, a_2, \dots, a_{n-1}$ .

**Tìm phần tử có khoá bằng  $X$  trong mảng?**

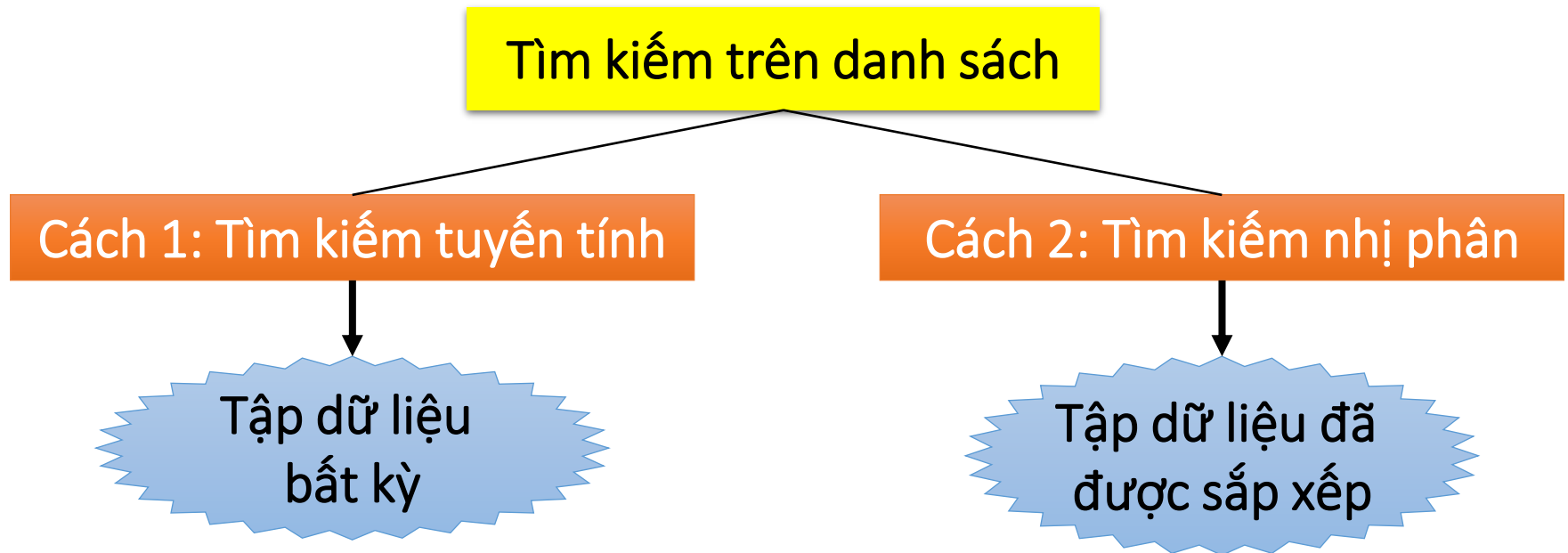
# 1. Giới thiệu bài toán tìm kiếm

- Tìm kiếm là quá trình xác định một đối tượng nào đó trong một tập các đối tượng. Kết quả tìm kiếm:
  - *Tìm thấy*: trả về thường ở 1 trong 2 dạng:
    - Đối tượng tìm được.
    - Chỉ số (xác định vị trí của đối tượng trong danh sách đó).
  - *Không tìm thấy*: trả về 1 trong các giá trị như **-1**, **false**, **NULL**, ...
- Việc tìm kiếm dựa theo một thuộc tính (trường - field) nào đó của đối tượng, thuộc tính này gọi là **khóa** (key) của việc tìm kiếm.
  - VD: Tìm sinh viên trong mảng DSSV, với thông tin của SV gồm {MaSV, HoTen, DiaChi, ... }
    - Khóa là gì?
    - Nếu tìm thấy cần trả về kết quả gì?

# 1. Giới thiệu bài toán tìm kiếm

- Bài toán được mô tả như sau:

- Tập dữ liệu được lưu trữ là dãy  $a_1, a_2, \dots, a_n$ . Giả sử chọn cấu trúc dữ liệu mảng để lưu trữ dãy số này trong bộ nhớ chính, có khai báo: `int a[n];`
- Khóa cần tìm là  $x$  với kiểu dữ liệu là `int`





## NỘI DUNG CHƯƠNG 3

1. Giới thiệu bài toán tìm kiếm
2. Tìm kiếm tuyến tính
3. Tìm kiếm nhị phân

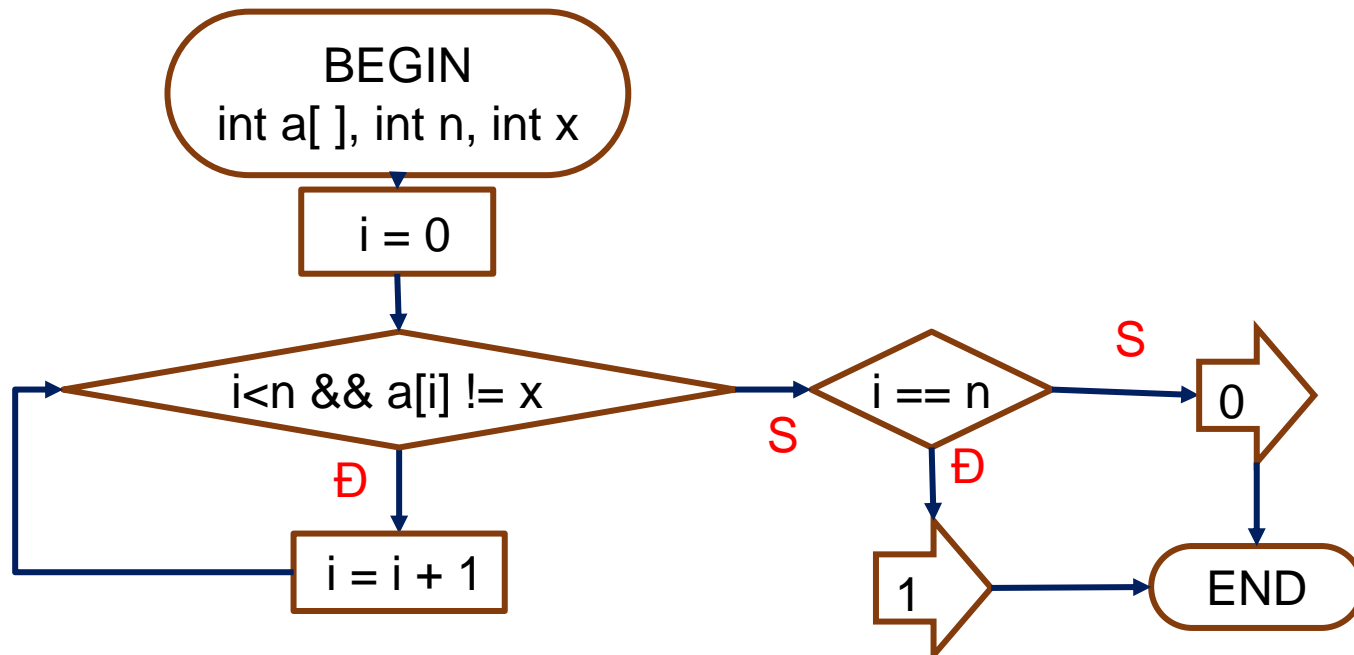
## 2. TÌM KIẾM TUYẾN TÍNH

- **Ý tưởng**: So sánh X lần lượt với phần tử thứ 0, thứ 1,... của mảng a cho đến khi gặp được khóa cần tìm, hoặc tìm hết mảng mà không thấy.
- **Các bước tiến hành**
  - **Bước 1**: Khởi gán  $i=0$ ;
  - **Bước 2**: So sánh  $a[i]$  với giá trị x cần tìm, có 2 khả năng
    - $a[i] == x$  tìm thấy x. Dừng;
    - $a[i] != x$  sang bước 3;
  - **Bước 3**:  $i=i+1$  //Xét tiếp phần tử kế tiếp trong mảng  
Nếu  $i==n$ : Hết mảng. Dừng;  
Ngược lại: Lặp lại bước 2;

## 2. Tìm Kiếm Tuyến Tính

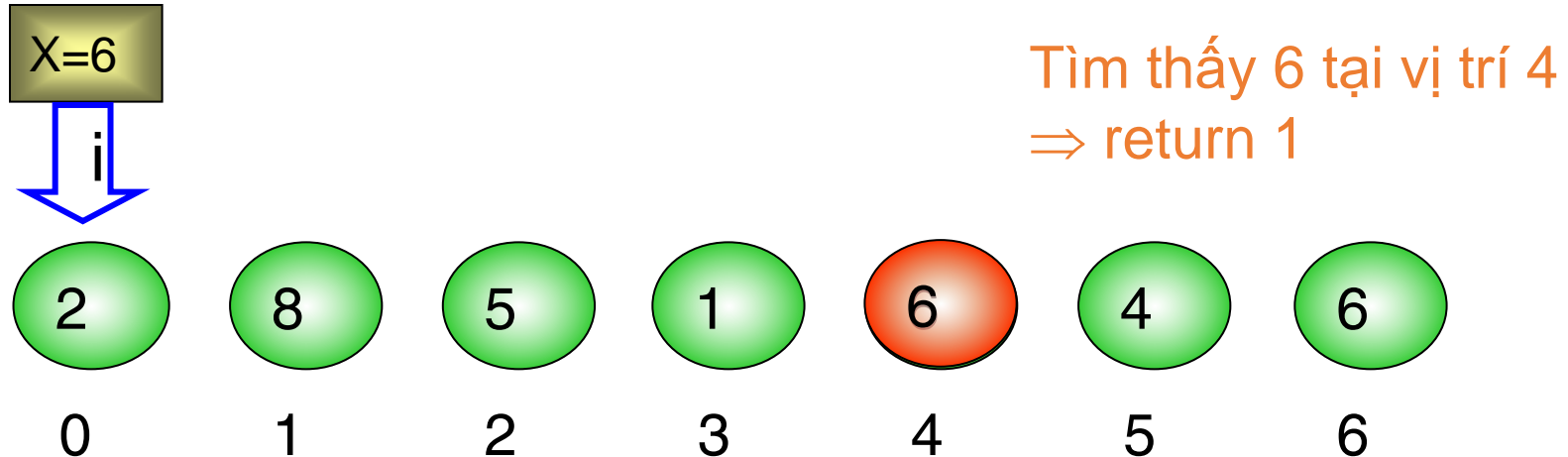
– **Cài đặt:** Hàm trả về 1 nếu tìm thấy, ngược lại trả về 0

```
int LinearSearch(int a[], int n, int x)
{
    int i=0;
    while ( (i<n) && (a[i] !=x) )
        i++;
    if (i==n)
        return 0; //không tìm thấy x
    else
        return 1; //Tìm thấy
}
```



## 2. Tìm Kiếm Tuyến Tính

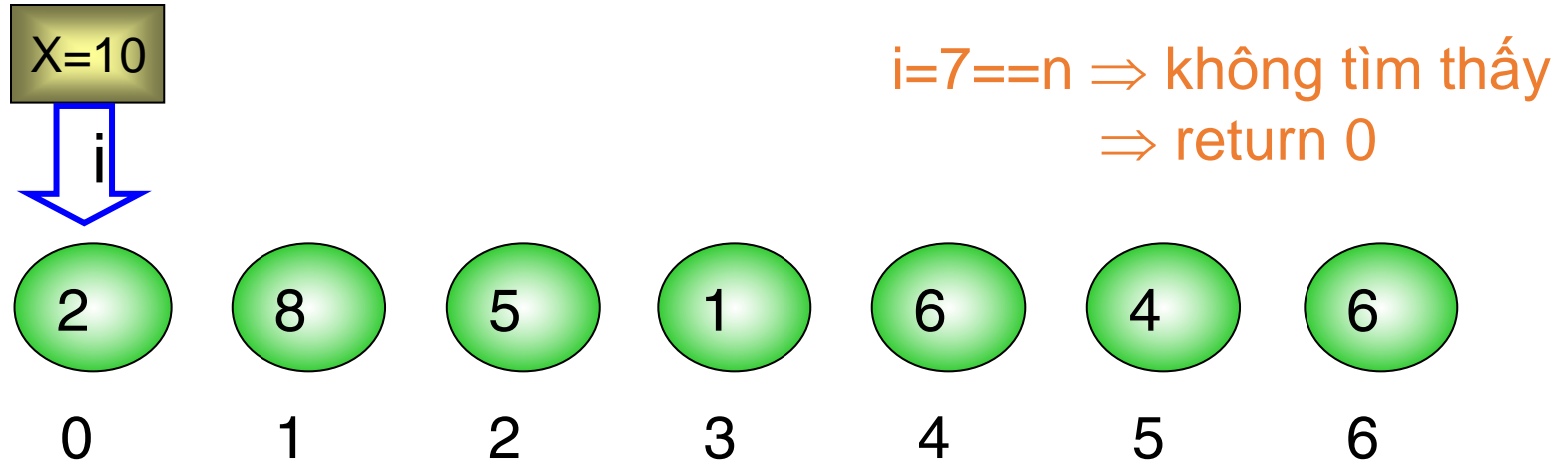
### – Minh Họa Giải thuật Tìm Kiếm Tuyến Tính



```
int LinearSearch(int a[], int n, int x)
{
    int i=0;
    while ((i<n) && (a[i]!=x))    i++;
    if (i==n)
        return 0; //không tìm thấy
    else
        return 1; //Tìm thấy
}
```

## 2. Tìm Kiếm Tuyến Tính

### – Minh Họa Giải thuật Tìm Kiếm Tuyến Tính



```
int LinearSearch(int a[], int n, int x)
{
    int i=0;
    while ((i<n) && (a[i]!=x))    i++;
    if (i==n)
        return 0; //không tìm thấy
    else
        return 1; //Tìm thấy
}
```

## **NỘI DUNG CHƯƠNG 3**

1. Giới thiệu bài toán tìm kiếm
2. Tìm kiếm tuyến tính
3. Tìm kiếm nhị phân

## 3. Tìm Kiếm Nhị Phân (*binary search*)

### 3.1. Giới thiệu

- Được áp dụng trên mảng đã có thứ tự.

- **Ý tưởng:**

- Giả sử ta xét mảng có thứ tự tăng, khi ấy ta có

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

- Nếu  $X > a_i$  thì  $X$  chỉ có thể xuất hiện trong đoạn  $[a_{i+1}, a_{n-1}]$
- Nếu  $X < a_i$  thì  $X$  chỉ có thể xuất hiện trong đoạn  $[a_0, a_{i-1}]$
- Ý tưởng của giải thuật là tại mỗi bước ta so sánh  $X$  với phần tử đứng giữa trong dãy tìm kiếm hiện hành, dựa vào kết quả so sánh này mà ta quyết định giới hạn dãy tìm kiếm ở nửa dưới hay nửa trên của dãy tìm kiếm hiện hành.

### 3. Tìm Kiếm Nhị Phân

#### 3.2. Giải thuật Tìm Kiếm Nhị Phân

Giả sử dãy tìm kiếm hiện hành bao gồm các phần tử nằm trong  $a_{\text{left}}$ ,  $a_{\text{right}}$ , các bước của giải thuật như sau:

- Bước 1:  $\text{left} = 0$ ;  $\text{right} = n-1$ ;
- Bước 2:
  - $\text{mid} = (\text{left} + \text{right}) / 2$ ; *// chỉ số ph/tử giữa dãy hiện hành*
  - So sánh  $a[\text{mid}]$  với  $x$ . Có 3 khả năng
    - $a[\text{mid}] = x$ : tìm thấy. Dừng
    - $a[\text{mid}] > x$ :  $\text{Right} = \text{mid} - 1$ ; *// thu hẹp bên phải dãy*
    - $a[\text{mid}] < x$ :  $\text{Left} = \text{mid} + 1$ ; *// thu hẹp bên trái dãy*
- Bước 3:
  - Nếu  $\text{Left} \leq \text{Right}$  ; *// còn phần tử trong dãy hiện hành*  
+ Lặp lại bước 2
  - Ngược lại: Dừng



### 3. Tìm Kiếm Nhị Phân

#### 3.3. Cài đặt Giải thuật tìm nhị phân

Hàm trả về giá trị 1 nếu tìm thấy, ngược lại hàm trả về giá trị 0

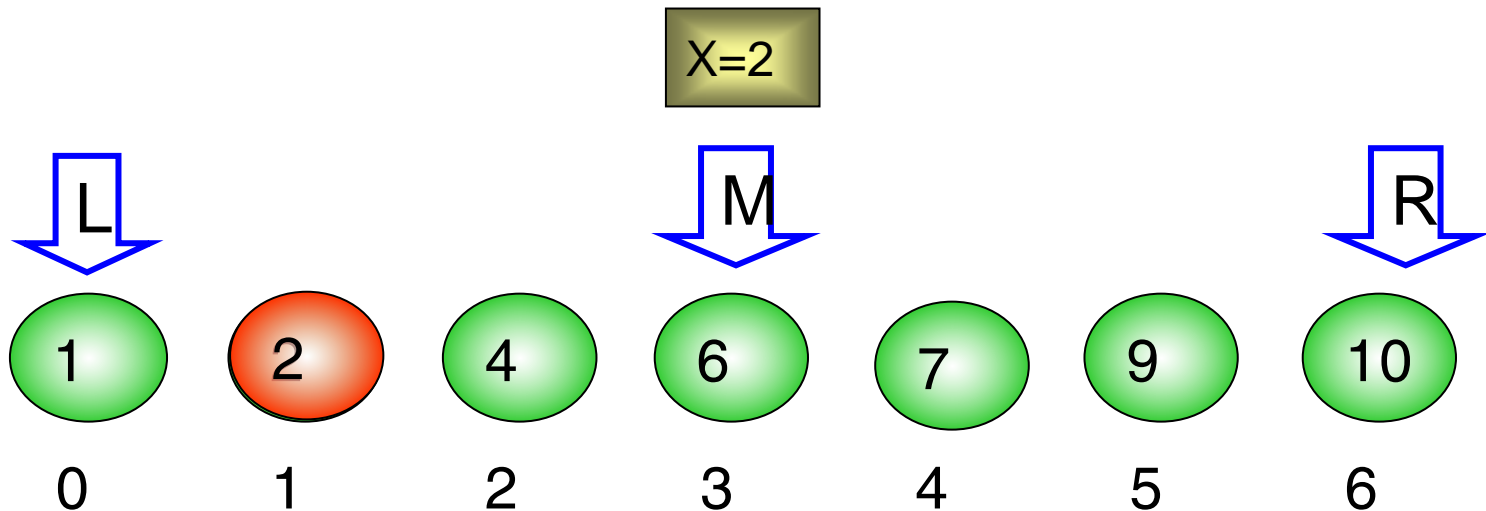
```
int BinarySearch(int a[], int n, int x)
{
    int left, right, mid; left=0; right=n-1;
    do
    {
        mid=(left+right)/2;
        if(a[mid]==x) return 1;
        else
            if(a[mid]<x) left=mid+1;
            else right=mid-1;
    }while(left<=right);
    return 0;
}
```

### 3. Tìm Kiếm Nhị Phân

#### 3.4. Minh Họa Giải thuật Tìm Nhị Phân (giá trị cần tìm X=2)

```
int BinarySearch(int a[],int n,int x)
{
    int left, right, mid; left=0; right=n-1;
    do
    {
        mid=(left+right)/2;
        if(a[mid]==x) return 1;
        else if(a[mid]<x) left=mid+1;
        else right=mid-1;
    }while(left<=right);
    return 0;
}
```

Tìm thấy 2 tại vị trí 1

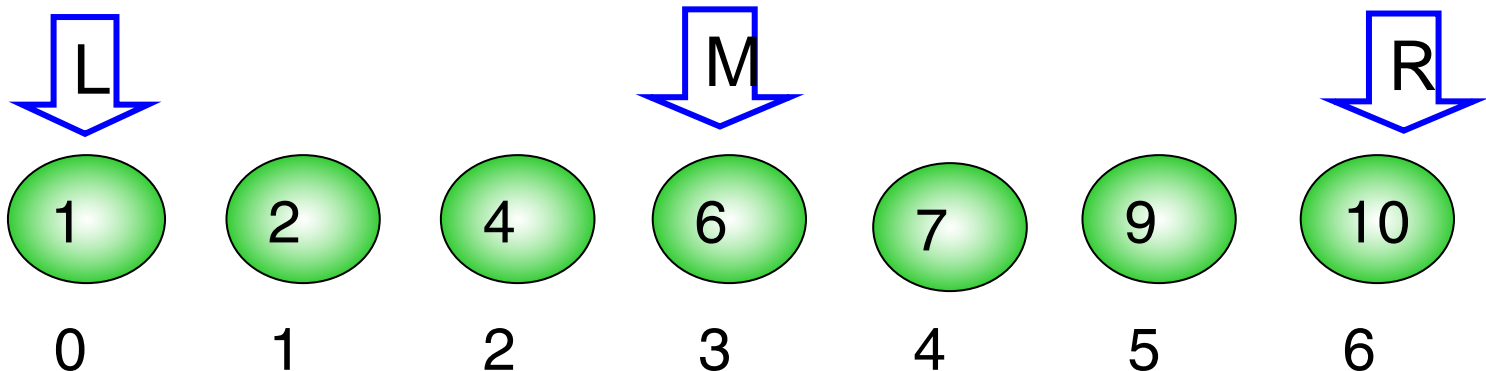


### 3. Tìm Kiếm Nhị Phân

#### 3.4. Minh Họa Giải thuật Tìm Nhị Phân (giá trị cần tìm $X=-1$ )

```
int BinarySearch(int a[],int n,int x)
{
    int left, right, mid; left=0; right=n-1;
    do
    {
        mid=(left+right)/2;
        if(a[mid]==x) return 1;
        else
            if(a[mid]<x) left=mid+1;
            else right=mid-1;
    }while(left<=right);
    return 0;
}
```

$X=-1$



$L=0$

$R=-1 \Rightarrow L < R \Rightarrow$  dừng khi không tìm thấy  $X=-1$

### 3. Tìm Kiếm Nhị Phân




#### 3.5. Cài đặt giải thuật Tìm nhị phân: Gọi hàm tìm nhị phân

```
void main()
{   int a[100], n, kq, x;
    // nhập số phần tử n
    // gọi hàm nhập mảng 1 chiều
    // nhập giá trị cần tìm x
    kq = BinarySearch(a,n,x);
    //hoặc if (BinarySearch(a,n,x)==1)
    if(kq==1)
        printf("tìm thấy");
    else
        printf("không tìm thấy");
}
```

```
int BinarySearch(int a[],int n,int x)
{   int mid; left=0; right=n-1;
    do
    {   mid=(left+right)/2;
        if(a[mid]==x) return 1;
        else
            if(a[mid]<x) left=mid+1;
            else right=mid-1;
    }while(left<=right);
    return 0;
}
```

### 3. Tìm Kiếm Nhị Phân

#### 3.5. Bài tập áp dụng

3	5	6	8	12	17	19
<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>
						

- Ghi kết quả chạy từng bước khi tìm  $X=19$ 
  - B1:  $l=0$ ,  $r=6$ ,  $mid=(l+r)/2$ ,  $A[mid] = 8$ ,  $8 < 19$
  - B2:  $l = mid+1=4$ ,  $mid=(l+r)/2=5$ ,  $A[mid] = 17 < 19$
  - B3:  $l=mid+1=6$ ,  $mid=(l+r)/2=6$ ,  $A[mid]=19 = X$  dừng

### 3. Tìm Kiếm Nhị Phân

#### 3.5. Bài tập áp dụng (biểu diễn cách trình bày khác)

3	5	6	8	12	17	19
0	1	2	3	4	5	6

- Ghi kết quả chạy từng bước khi tìm  $X=19$

Lần	L	R	M	A[M]
1	0	6	3	8
2	4	6	5	17
3	6	6	6	19

Dừng sau khi  
tìm thấy X

### 3. Tìm Kiếm Nhị Phân

### 3.6. Thực hành

```
int BinarySearch(int a[],int n,int x)
{
    int left, right, mid; left=0; right=n-1;
    do
    {
        mid=(left+right)/2;
        if(a[mid]==x) return 1;
        else if(a[mid]<x) left=mid+1;
        else right=mid-1;
    }while(left<=right);
    return 0;
}
```

- i. Trình bày các bước tìm X=35 tại
- ii. Thực hiện tương tự khi X=6

3	5	7	8	12	17	19	20	24	27	31	36	39	40	42	48
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Lần	L	R	M	A[M]
1	0	15	7	20
2	8	15	11	36
3	8	10	9	27
4	10	10	10	31
5	11	10		
Dừng do $L=11 > R=10$ $\Rightarrow$ Không thấy $X=35$				

Lần	L	R	M	A[M]
1	0	15	7	20
2	0	6	3	8
3	0	2	1	5
4	2	2	2	7
5	2	1	1	5
Dừng do $L=2 > R=1$ $\Rightarrow$ Không thấy $X=6$				

### 3. Tìm Kiếm Nhị Phân

#### 3.6. Thực hành

- iii. Cải tiến chương trình để in ra vị trí tìm thấy X (khi mảng có chứa X) hoặc in ra thông báo không tìm thấy (khi mảng không chứa X) .
- iv. Cải tiến chương trình ở câu *iii* để có số lần đã thực hiện so sánh trong quá trình tìm X.

```
int BinarySearch(int a[],int n,int x)
{
    int mid, left=0, right=n-1;
    do
    {
        mid=(left+right)/2;
        if(a[mid]==x)
            return mid;
        else
            if(a[mid]<x)
                left=mid+1;
            else
                right=mid-1;
    }while(left<=right);
    return 0;
}
```

```
int BinarySearch(int a[],int n,int x, int &dem)
{
    int mid, left=0, right=n-1;
    dem=0;
    do
    {
        mid=(left+right)/2;
        if(a[mid]==x)
        {
            return 1;
            dem++;
        }
        else
            if(a[mid]<x)
            {
                left=mid+1;
                dem++;
            }
            else
                right=mid-1;
    }while(left<=right);
    return 0;
}
```





Slide\_