GIẢI THUẬT TÌM KIẾM

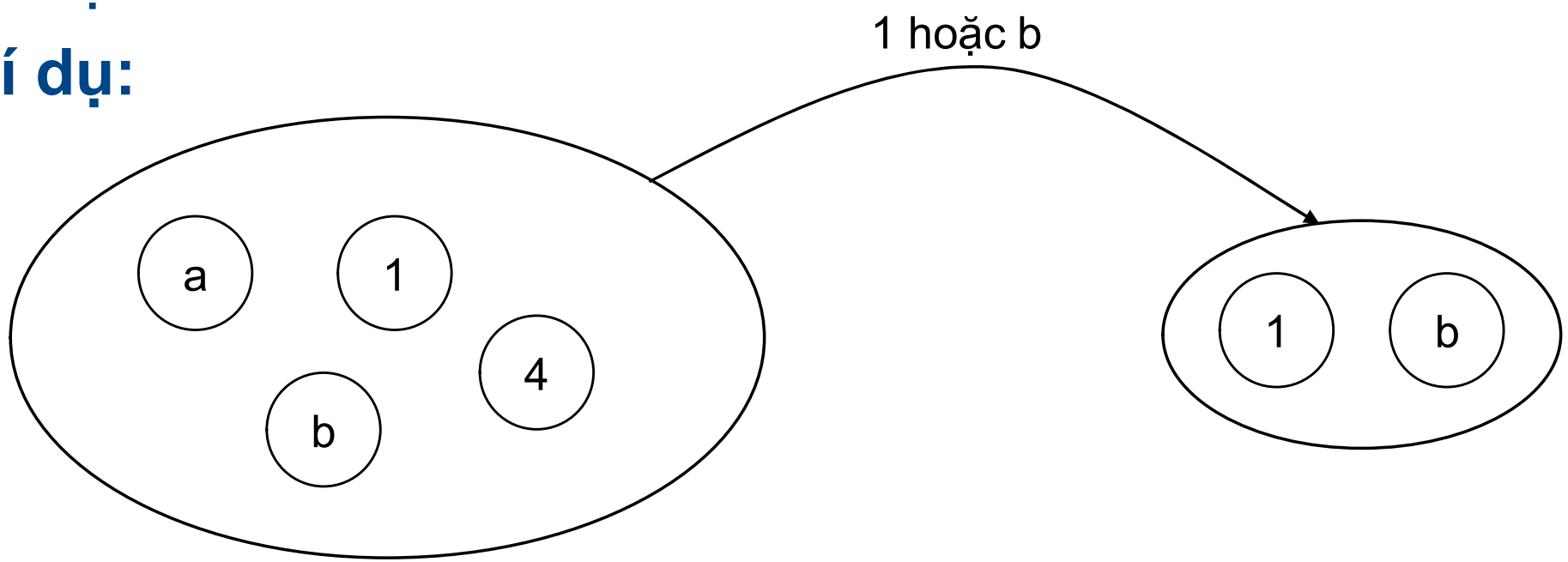
- KHÁI NIỆM
- TÌM KIẾM TUYẾN TÍNH
- TÌM KIẾM NHỊ PHÂN
- PHẦN TỬ CẦM CANH

GIẢI THUẬT TÌM KIẾM

❖KHÁI NIỆM

Tìm kiếm là thao tác duyệt toàn bộ một tập hợp nào đó để xác định các phần tử trong tập hợp đó thỏa một tính chất nào đó.

Ví dụ:



GIẢI THUẬT TÌM KIẾM

❖KHÁI NIỆM

Khóa là trường trong một cấu trúc được sử dụng để tính toán trong quá trình tìm kiếm và sắp xếp.

Ví dụ 1: giả sử có cấu trúc.

```
struct MayTinh {
    int maso;
    char tenmay[20];
    int dongia;
};
```

Trường dongia được gọi là khóa nếu nó được dùng để tính toán trong tìm kiếm và sắp xếp.

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH (Linear Search)

Tìm kiếm tuyến tính là tìm kiếm một giá trị khóa x trên một tập hợp chưa có thứ tự bằng cách duyệt tuần tự từng phần tử trong tập hợp để xác định phần tử có giá trị khóa bằng x.

- **Ví dụ 2:** tìm phần tử có giá trị 5 trong tập hợp

{1, 2, 9, 4, 3, 5, 7}

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

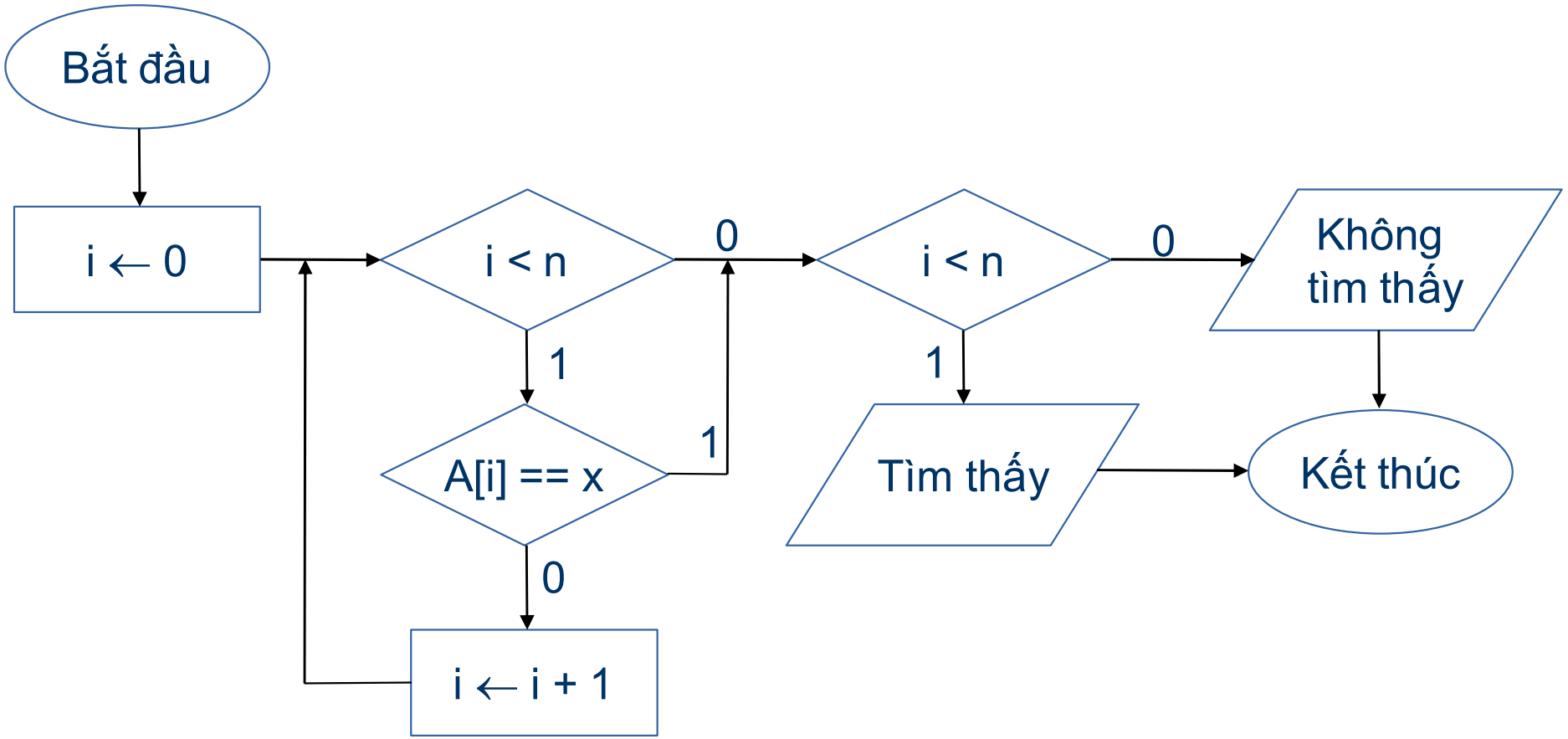
- **Giải thuật:** Giả sử n phần tử của tập hợp A được đánh số thứ tự từ 0 đến n-1, tìm phần tử có giá trị khóa x.

- Bước 1: i ← 0
- Bước 2: Nếu i < n thì qua bước 3, ngược lại qua bước 5.
- Bước 3: Nếu A[i] == x qua bước 5.
- Bước 4: i ← i + 1. qua bước 2.
- Bước 5: Nếu i < n thì tìm thấy, ngược lại là không tìm thấy
- Bước 6: Kết thúc.

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

Lưu đồ:



GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

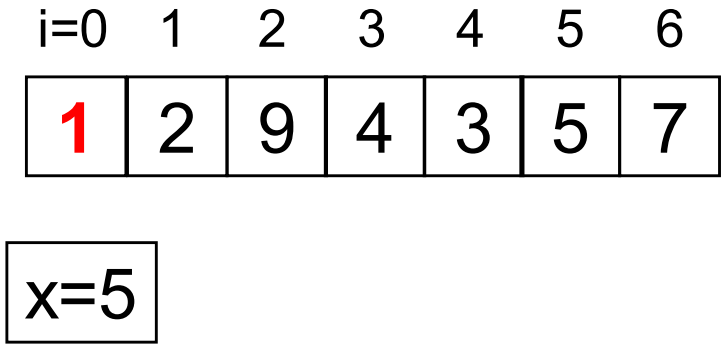
- **<u>Ví dụ 3:</u>** áp dụng giải thuật tìm kiếm tuyến tính cho ví dụ 2

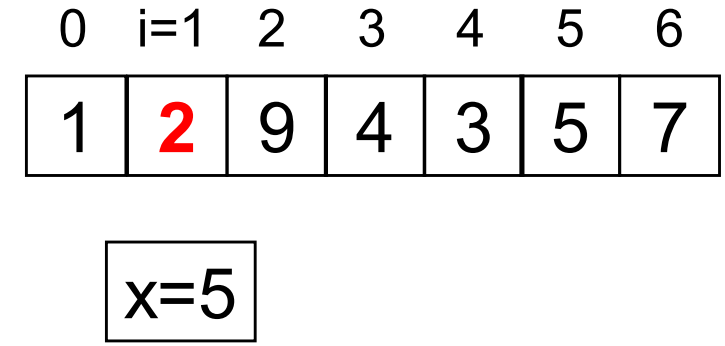
- Đánh số thứ tự cho các phần tử trong tập hợp bằng cách đưa chúng vào một cấu trúc mảng một chiều.

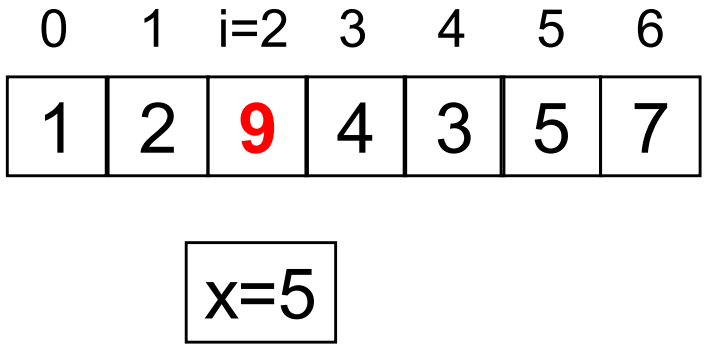
0	1	2	3	4	5	6
1	2	9	4	3	5	7

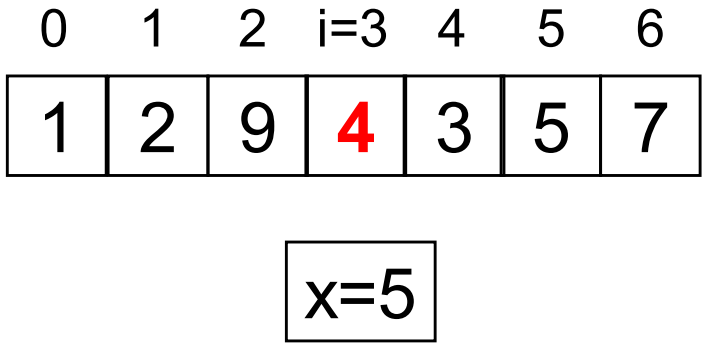
GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH



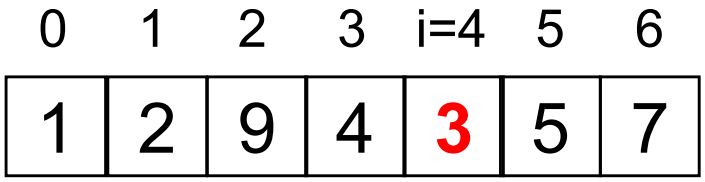




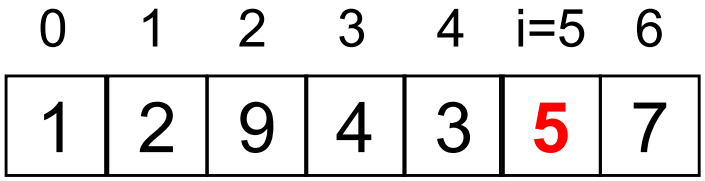


GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH



x=5



x=5 A[i] = 5 = x

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

- **Cài đặt**

```
int LinearSearch(int *A, int n, int x) {
    int i = 0;
    while (i<n) {
            if (A[i] == x) break;
            i++;
    }
    if (i < n) return i; // Tìm thấy
    else return -1; // Không tìm thấy
}
```

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

- **Đánh giá theo trường hợp xấu nhất**

- số phép tính trước vòng lặp: 1
- số phép so sánh tại vòng lặp : 2*n + 1
- số phép toán nếu tìm thấy: 1
- số phép tăng i tại vòng lặp: n
- số phép tính sau vòng lặp: 2

- **số phép tính của giải thuật**: 3n + 5

Độ phức tạp của thuật toán là: O(n).

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

- **Đánh giá theo trường hợp trung bình**

Trường hợp giá trị x ở vị trí thứ i

- số phép tính trước vòng lặp: 1
- số phép so sánh tại vòng lặp : 2*i
- số phép toán nếu tìm thấy: 1
- số phép tăng i tại vòng lặp: i
- số phép tính sau vòng lặp: 2

- **số phép tính $T_i(n)$**: 3i + 4

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

- **Đánh giá theo trường hợp trung bình**

  giả sử sự xuất hiện của x tại vị trí thứ i là ngẫu nhiên với xác suất là 1/n, số phép tính trung bình là:

  $T(n) = \Sigma_i T_i(n).1/n, \quad i = 0, n.$

  $= (1/n)\ \Sigma_i(3i + 4)$

  $= (1/n)*(3(n + 1)*(n+2)/2 + 4(n+1))$

  $= (1/n)*(3n^2+17n+14)/2$

  $= 3n/2 + 17/2 +7/n$

  Độ phức tạp của thuật toán là O(n).

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM TUYẾN TÍNH

- **Bài tập**

  Giả sử dữ liệu về một nhân viên trong một công ty gồm: mã số nhân viên, họ tên, năm sinh, mức lương. Xây dựng cấu trúc dữ liệu biểu diễn danh sách nhân viên và đưa ra giải thuật tìm kiếm tất cả nhân viên có mức lương bằng giá trị x do người dùng xác định.

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN (Binary Search)

Tìm kiếm nhị phân là tìm kiếm một giá trị khóa x trên một dãy có thứ tự $a_0 \le a_1 \le .. \le a_{n-1}$ bằng cách so sánh giá trị khóa của phần tử a_i ở trung tâm dãy. Nếu a_i có thứ tự nhỏ hơn x thì sẽ tìm x trong dãy con $[a_{i+1}, a_{n-1}]$. Nếu a_i có thứ tự lớn hơn x thì sẽ tìm x trong dãy con $[a_0, a_{i-1}]$. Nếu a_i bằng x thì dừng. Việc thực hiện tìm kiếm trong dãy con được diễn ra tương tự.

- **Ví dụ 4:** tìm phần tử có giá trị 3 trong dãy {1, 2, 3, 4, 5, 7, 9}

GIẢI THUẬT TÌM KIẾM

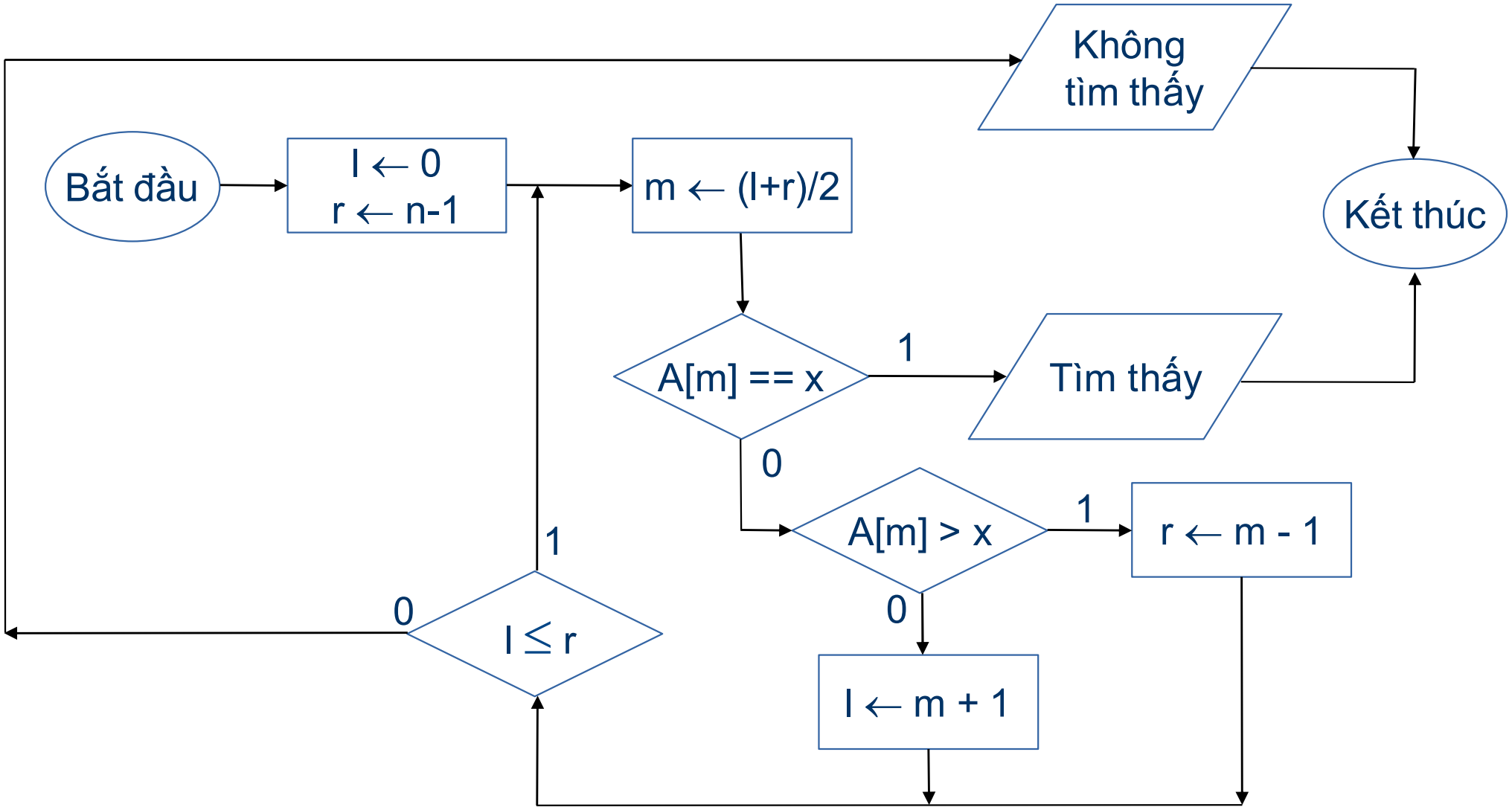
❖TÌM KIẾM NHỊ PHÂN

- **Giải thuật:** tìm khóa x trong dãy A gồm n phần tử a_i được đánh số từ 0 đến n - 1

- Bước 1: l ← 0, r ← n-1 // l: left, r: right
- Bước 2: m ← (l + r)/2.
  + Nếu A[m] = x thì tìm thấy và qua bước 5.
  + Nếu A[m] > x thì r ← m-1 ngược lại thì l ← m+1.
- Bước 3: Nếu l ≤ r thì quay lại bước 2.
- Bước 4: Thông báo không tìm thấy.
- Bước 5: Kết thúc.

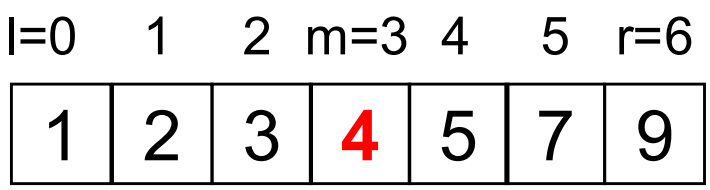
GIẢI THUẬT TÌM KIẾM

Lưu đồ:

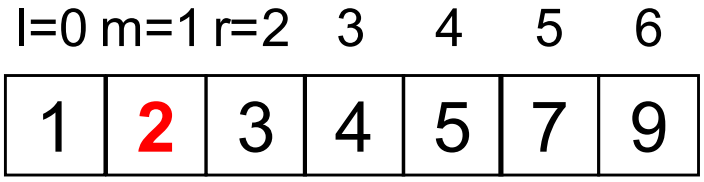


GIẢI THUẬT TÌM KIẾM

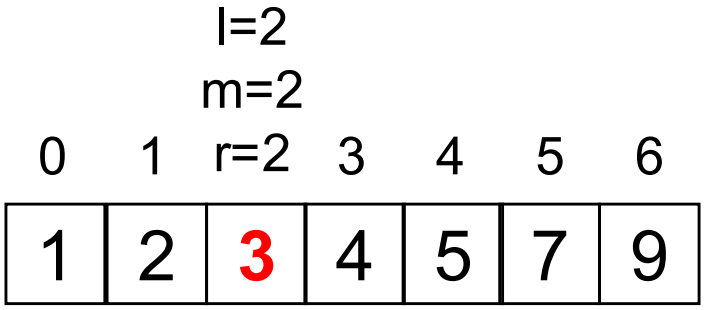
❖TÌM KIẾM NHỊ PHÂN



x=3



x=3



x=3

A[m] = 3 = x

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

- **Cài đặt**

```
int BinarySearch(int *A, int n, int x) {
    int l = 0, r = n-1, m;
    do {
    m = (l + r) / 2;
    if (x == A[m]) return m; // Tìm thấy
    else     if (x < A[m]) r = m - 1;
                else l = m + 1;
    }while (l <= r);
    return -1; // Không tìm thấy
}
```

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

- **Đánh giá theo trường hợp xấu nhất:**

- Số phép tính trước vòng lặp: 2
- Số phép tính trong vòng lặp: 5
- Số lần lặp (số lần chia dãy): $\log_2 n$
- Số phép tính sau vòng lặp: 1

- Số phép tính của thuật toán: $5\log_2 n + 3$
- Độ phức tạp của thuật toán là O(log(n))

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

- **<u>Đánh giá theo trường hợp trung bình:</u>**

Gọi $T_i(n)$ là trường hợp giá trị x xuất hiện ở vị trí mà sau i lần chia dãy sẽ xác định được nó.

- Số phép tính trước vòng lặp: 2
- Số phép tính trong vòng lặp: 5
- Số lần lặp (số lần chia dãy): i
- Số phép tính sau vòng lặp: 1

- Số phép tính $T_i(n)$: 5i + 3

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

- **<u>Đánh giá theo trường hợp trung bình:</u>**

Giả sử sự xuất hiện của x ở vị trí i lần chia dãy là như nhau:

$T(n) = \Sigma_i (5i + 3)*2^{i-1}/n, \quad i = 1,.., \lceil \log_2 n \rceil.$

$= (1/n)*(2^0*5*1 + 3*2^0)$

$+ (1/n)*(2^1*5*2 + 3*2^1)$

....

$+ (1/n)*((2^{\log_2(n)-2}) * 5*(\log_2 n - 1) + 3*2^{\log_2(n)-2})$

$+ (1/n)*((2^{\log_2(n)-1}) * 5*(\log_2 n) + 3*2^{\log_2(n)-1})$

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

- **<u>Đánh giá theo trường hợp trung bình:</u>**

Đặt $k = \log_2 n \Rightarrow k - 1 = \log_2(n) - 1$

$T(n) = (1/n)*(2^0*5*1 + 3*2^0)$

$+ (1/n)*(2^1*5*2 + 3*2^1)$

....

$+ (1/n)*((2^{k-2}) * 5*(k-1) + 3*2^{k-2})$

$+ (1/n)*((2^{k-1}) * 5*k + 3*2^{k-1})$

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

- **Đánh giá theo trường hợp trung bình:**

Xét:

$S(n) = 2^0*1+2^1*2+..+2^{k-2} * (k-1)+2^{k-1} * k$

$2S(n) = 2^1*1+2^2*2+..+2^{k-1} * (k-1)+2^k * k$

$\Rightarrow S(n) = -2^0 - 2^1 - .. - 2^{k-1} + 2^k *k$

$= k*2^k - (2^k - 1)$

$= (k-1)*2^k + 1$

GIẢI THUẬT TÌM KIẾM

❖TÌM KIẾM NHỊ PHÂN

- **<u>Đánh giá theo trường hợp trung bình:</u>**

$T(n) = (1/n)(5*((k-1)*2^k + 1) + 3*(2^k - 1))$

$= (1/n)((5k-2)*2^k + 2)$

$= (1/n)((5*\log_2(n) - 2) * 2^{\log_2 n} + 2)$

$= 5*\log_2(n) - 2 + 2/n$

Độ phức tạp của thuật toán là O(log(n))

GIẢI THUẬT TÌM KIẾM

❖PHẦN TỬ CẦM CANH

Trong tìm kiếm tuyến tính, quá trình tìm phần tử x phải kiểm tra phần tử đang xét có thuộc mảng các phần tử cần tìm hay không. Để bỏ qua việc kiểm tra này, mảng các phần tử được **thêm phần tử x vào cuối** để đảm bảo quá trình tìm sẽ luôn trả về một phần tử trong mảng. Phần tử thêm vào được gọi là phần tử cầm canh.

GIẢI THUẬT TÌM KIẾM

❖PHẦN TỬ CẦM CANH

- **Cài đặt**

```
int LinearSearch(int *A, int n, int x) {
    int i = 0;
    A[n] = x;
    while (A[i] != x) i++;
    if (i == n) return -1;
    else return i;
}
```

GIẢI THUẬT TÌM KIẾM

❖PHẦN TỬ CẦM CANH

- **Đánh giá theo trường hợp xấu nhất**

- số phép tính trước vòng lặp: 2
- số phép so sánh tại vòng lặp : n + 1
- số phép tăng i tại vòng lặp: n
- số phép tính sau vòng lặp: 2

- **số phép tính của giải thuật**: 2n + 5 < 3n + 5

Độ phức tạp của thuật toán là: O(n).

GIẢI THUẬT TÌM KIẾM

- Bài tập

  Cho mảng các số nguyên như sau:

  -9 -9 -9 -2 0 3 7 7 10 15

  Xác định số lần so sánh để tìm x = -9 theo phương pháp tìm kiếm tuyến tính và tìm kiếm nhị phân. Và cho biết chỉ số mảng chứa kết quả tìm (nếu có).