Chương 3

Tìm kiếm và sắp xếp

Nội dung trình bày

- Tìm kiếm
- Sắp xếp

3.1 Tìm kiếm

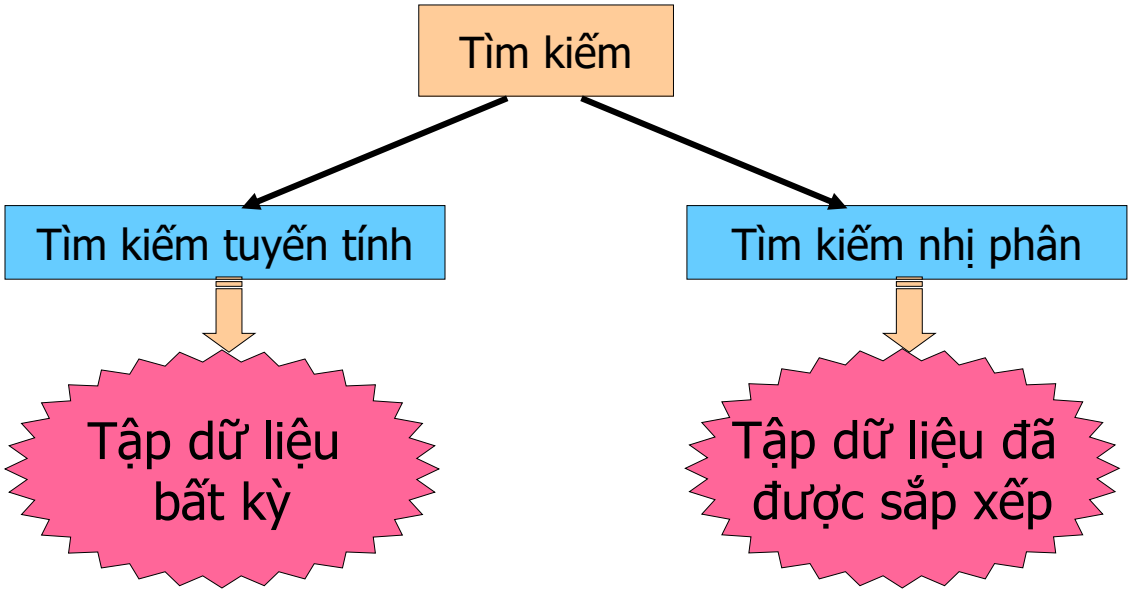
- Tìm kiếm là thao tác quan trọng & thường xuyên trong tin học.
  - Tìm kiếm một nhân viên trong danh sách nhân viên.
  - Tìm một sinh viên trong danh sách sinh viên của một lớp…
  - Tìm kiếm một tên sách trong thư viện.

3.1 Tìm kiếm

- Tìm kiếm là quá trình xác định một đối tượng nào đó trong một tập các đối tượng. Kết quả trả về là đối tượng tìm được (nếu có) hoặc một chỉ số (nếu có) xác định vị trí của đối tượng trong tập đó.
- Việc tìm kiếm dựa theo một trường nào đó của đối tượng, trường này là khóa (key) của việc tìm kiếm.
- VD: đối tượng sinh viên có các dữ liệu {MaSV, HoTen, DiaChi,…}. Khi đó tìm kiếm trên danh sách sinh viên thì khóa thường chọn là MaSV hoặc HoTen.

3.1 Tìm kiếm

- Bài toán được mô tả như sau:
  - Tập dữ liệu được lưu trữ là dãy $a_1, a_2,..,a_n$. 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, có kiểu nguyên: int x;



3.1.1 Tìm kiếm tuyến tính

- Ý tưởng chính: duyệt tuần tự từ phần tử đầu tiên, lần lượt so sánh khóa tìm kiếm với các phần tử trong danh sách. Cho đến khi gặp phần tử cần tìm hoặc đến khi duyệt hết danh sách.

3.1.1 Tìm kiếm tuyến tính

- Các bước tiến hành:
  - Bước 1: duyệt tuần tự từ phần tử đầu tiên;
  - Bước 2: so sánh các phần tử trong danh sách với khóa tìm kiếm có hai khả năng
    - Nếu bằng nhau ⇒ Tìm thấy ⇒ Dừng
    - Nếu khác nhau chuyển Sang bước 3
  - Bước 3: xét phần tử kế tiếp trong mảng
    - Nếu hết mảng, không tìm thấy. ⇒ Dừng
    - Nếu chưa hết mảng quay lại bước 2

3.1.1 Tìm kiếm tuyến tính

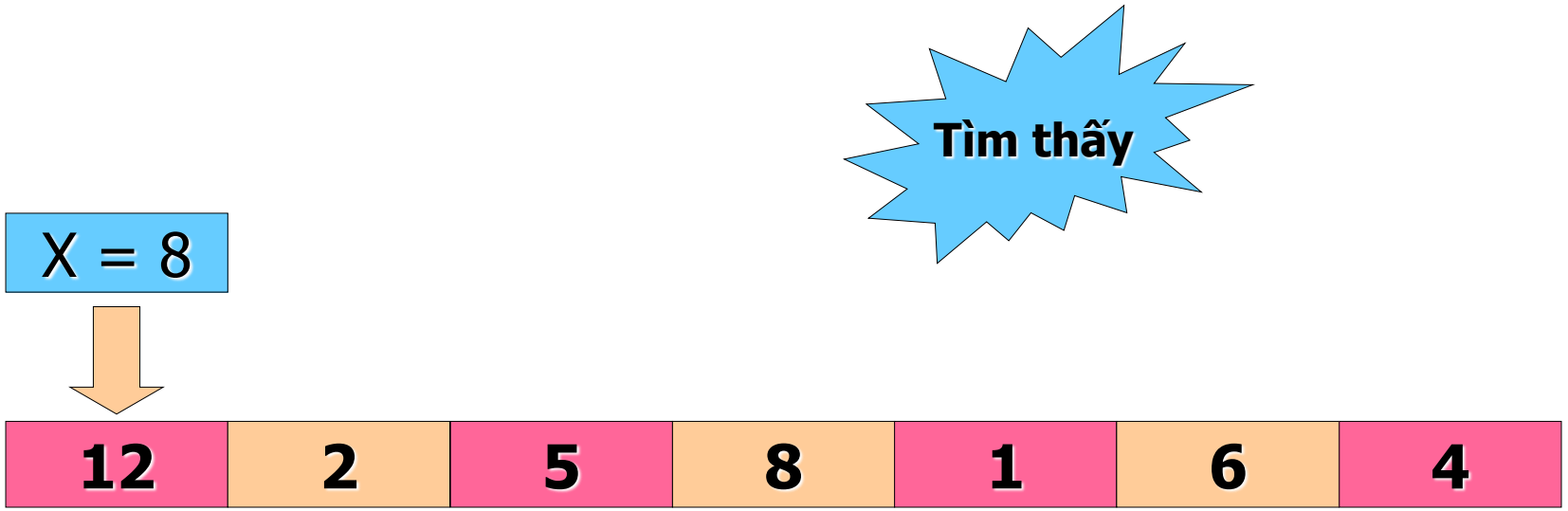
- Các bước tiến hành:
  - Bước 1: i = 0;
  - Bước 2: So sánh a[i] với x, có hai khả năng
    - a[i] = x: Tìm thấy $\Rightarrow$ Trả về i $\Rightarrow$ Dừng
    - a[i] $\neq$ x: Sang bước 3
  - Bước 3: i ++ // xét phần tử kế tiếp trong mảng
    - Nếu i > N: Hết mảng, không tìm thấy $\Rightarrow$ Trả về -1 $\Rightarrow$ Dừng
    - Nếu i $\leq$ N: Quay lại bước 2

3.1.1 Tìm kiếm tuyến tính

- Ví dụ: Cho dãy số a, giá trị tìm x = 8

12 2 5 8 1 6 4

Minh họa tìm kiếm tuyến tính

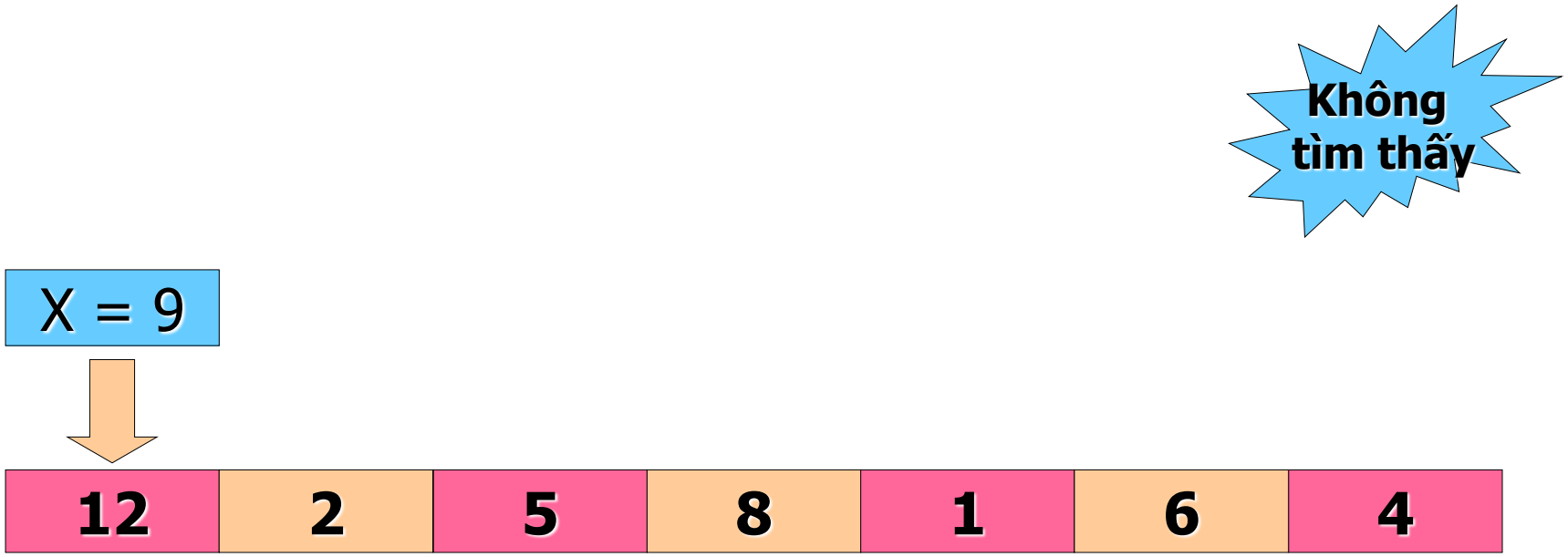


3.1.1 Tìm kiếm tuyến tính

- Ví dụ: Cho dãy số a, giá trị tìm x = 9

12 2 5 8 1 6 4

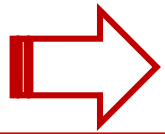
Minh họa tìm kiếm tuyến tính



3.1.1 Tìm kiếm tuyến tính

- Thuật toán tìm kiếm tuyến tính

```
int      Search(int a[], int n, int x)
{
  int i =0;
  while (i<n) && ( a[i] != x)
    i++;
  if (i >= n)
        return -1; // tìm không thấy
  else
        return i;  // tìm thấy tại vị trí i
}
```



3.1.1 Tìm kiếm tuyến tính

- Thuật toán tìm kiếm tuyến tính cải tiến

```
int  Search(int a[], int n, int x)
{
    int i =0;
    a[n] =x;                    // thêm phần tử thứ n+1
    while (x != a[i])
            i++;
    if (i == n)
            return -1;          // tìm hết mảng nhưng không có x
    else
            return i;           // tìm thấy x tại vị trí i
}
```

3.1.1 Tìm kiếm tuyến tính

- Nhận xét
  - Giải thuật tìm kiếm tuyến tính không phụ thuộc vào thứ tự của các phần tử trong mảng, do vậy đây là phương pháp tổng quát nhất để tìm kiếm trên một dãy bất kỳ.
  - Một thuật toán có thể được cài đặt theo nhiều cách khác nhau, kỹ thuật cài đặt ảnh hưởng nhiều đến tốc độ thực hiện. Ví dụ như thuật toán Search cải tiến sẽ chạy nhanh hơn thuật toán trước do vòng lặp while chỉ so sánh một điều kiện...

3.1.2 Tìm kiếm nhị phân

Phép tìm kiếm nhị phân được áp dụng trên dãy khóa đã có thứ tự: $a[0] \le a[1] \le ... \le a[n-1]$.

3.1.2 Tìm kiếm nhị phân

- Ý tưởng
  - Giả sử ta cần tìm trong đoạn a[left,..,right] với khóa tìm kiếm là x, trước hết ta xét phần tử giữa là a[mid], với mid=[left+right]/2.
    - Nếu a[mid] = x thì việc tìm kiếm thành công.
    - Nếu a[mid] < x thì có nghĩa là đoạn a[left] đến a[mid] chỉ chứa khóa < x, ta tiến hành tìm kiếm từ a[mid+1] đến a[right].
    - Nếu a[mid] > x thì có nghĩa là đoạn a[mid] đến a[right] chỉ chứa khoá > x, ta tiến hành tìm kiếm từ a[left] đến a[mid-1].
  - Quá trình tìm kiếm thất bại nếu left > right.

3.1.2 Tìm kiếm nhị phân

- Các bước tiến hành
  - B1: left =0, right = n-1 // tìm kiếm trên tất cả phần tử
  - B2: mid = (left + right)/2 // lấy mốc so sánh

    So sánh a[mid] với x, có 3 khả năng
    - a[mid] = x: Tìm thấy $\Rightarrow$ Trả về mid $\Rightarrow$ Dừng
    - a[mid] > x: // tìm tiếp trong dãy a[left]...a[mid-1]

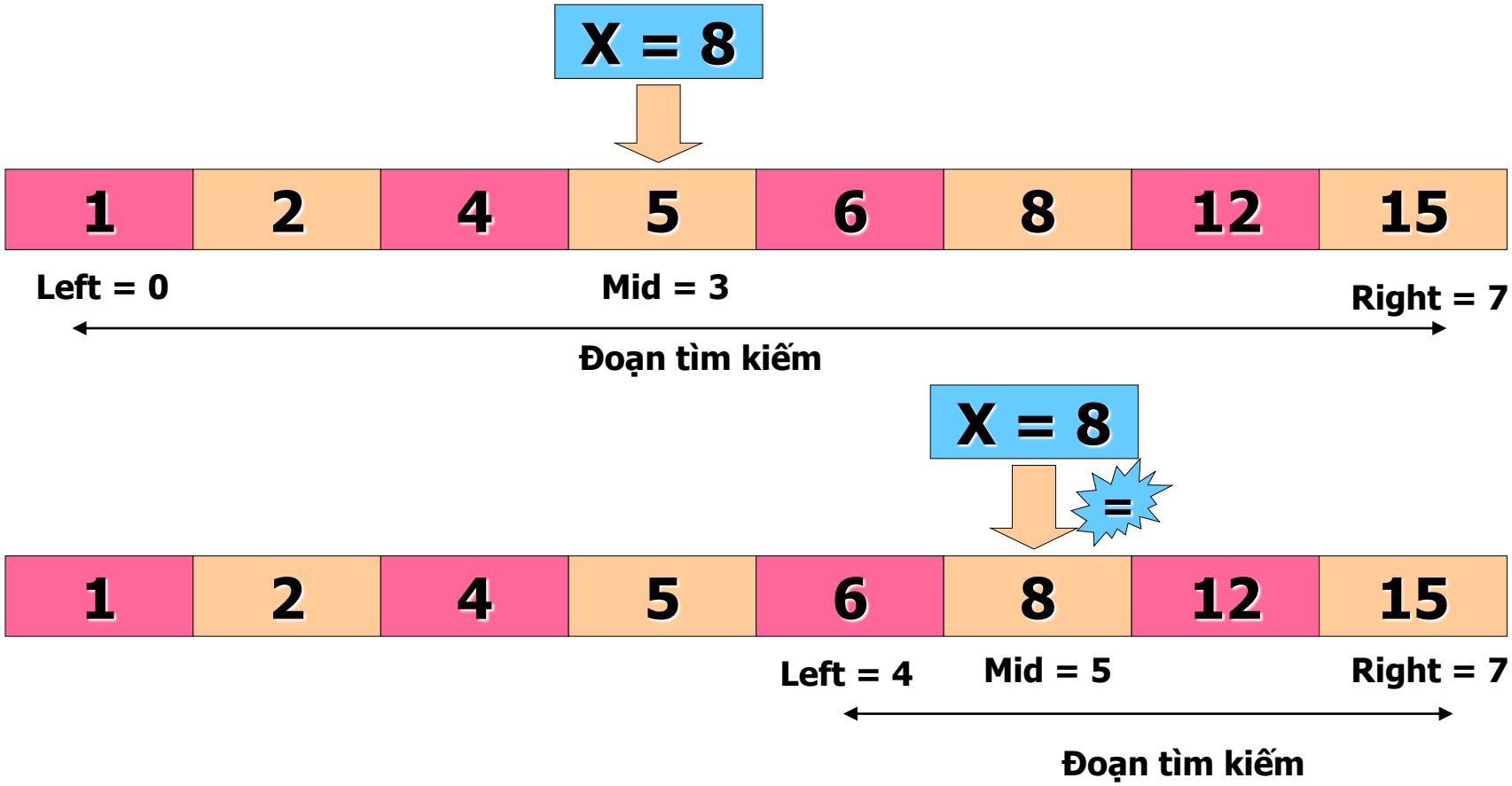
      right = mid -1;
    - a[mid] < x: // tìm tiếp trong dãy a[mid+1]...a[right]

      left = mid +1;
  - B3:
    - Nếu left $\leq$ right // còn phần tử $\Rightarrow$ tìm tiếp $\Rightarrow$ Lặp B2
    - Ngược lại: Trả về -1 $\Rightarrow$ Dừng // đã xét hết các phần tử

3.1.2 Tìm kiếm nhị phân

Ví dụ: cho dãy số gồm 8 phần tử bên dưới và x = 8:

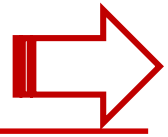
1 2 4 5 6 8 12 15



3.1.2 Tìm kiếm nhị phân

- Thuật toán tìm kiếm NP BinarySearch

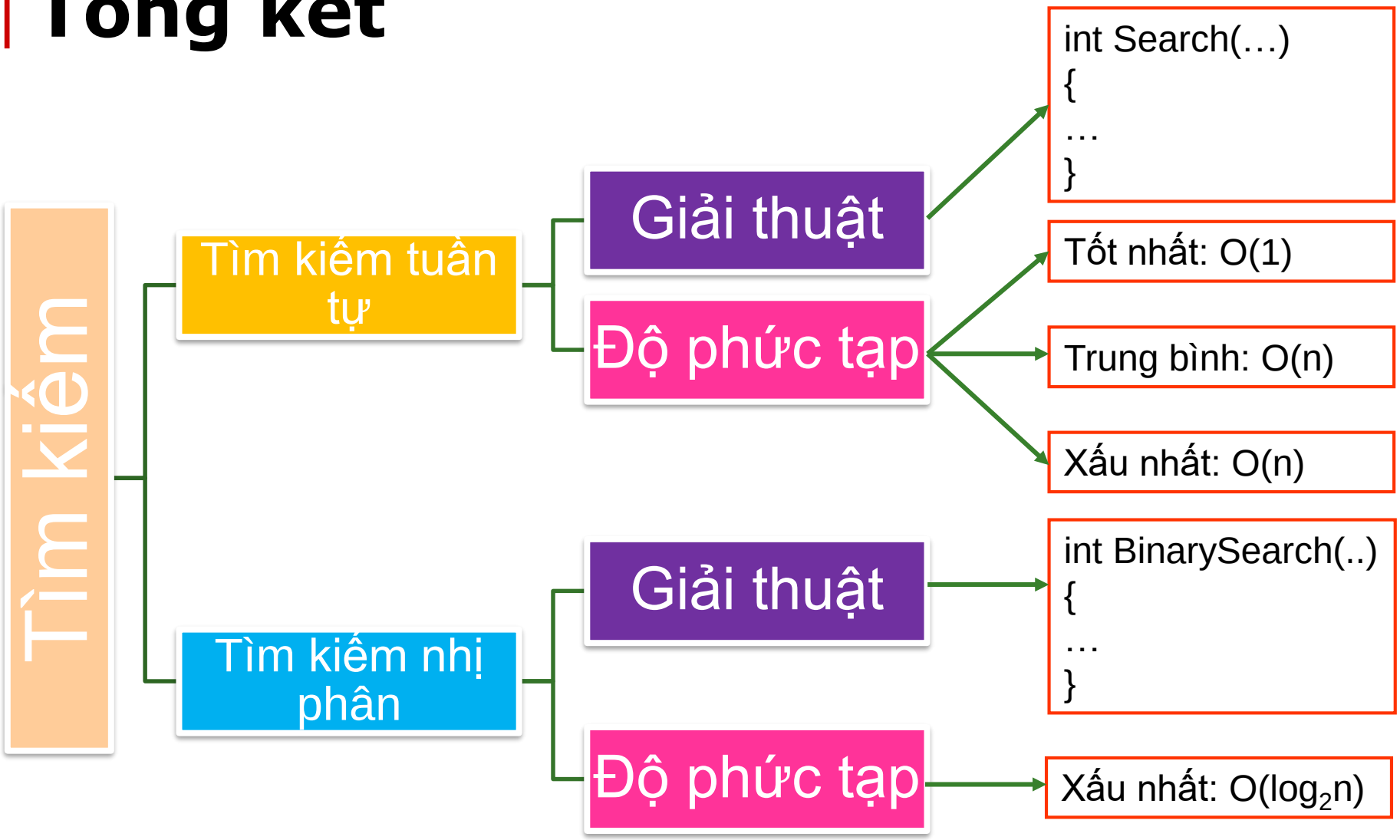
```
int  BinarySearch(int a[], int n,int x)
{ int left = 0, right = n-1, mid;
  while (left <= right)
  { mid = (left + right)/ 2;//lấy điểm giữa
    if (a[mid] == x)            //tìm thấy
          return  mid;
    else if (a[mid] < x)  //tìm đoạn bên phải mid
                left = mid+1;
          else                  //tìm đoạn bên trái mid
                right = mid-1;
  }
  return  -1;                   //không tìm được
}
```



3.1.2 Tìm kiếm nhị phân

- Nhận xét
  - Giải thuật nhị phân dựa vào quan hệ giá trị của các phần tử trong mảng để định hướng trong quá trình tìm kiếm, do vậy chỉ áp dụng được với dãy đã có thứ tự.
  - Giải thuật nhị phân tìm kiếm nhanh hơn tìm kiếm tuyến tính.
  - Tuy nhiên khi áp dụng giải thuật nhị phân thì cần phải quan tâm đến chi phí cho việc sắp xếp mảng. Vì khi mảng được sắp thứ tự rồi thì mới tìm kiếm nhị phân.

Tổng kết


Tìm kiếm
Tìm kiếm tuần tự
Giải thuật
int Search(…)
{
…
}
Độ phức tạp
Tốt nhất: O(1)
Trung bình: O(n)
Xấu nhất: O(n)
Tìm kiếm nhị phân
Giải thuật
int BinarySearch(..)
{
…
}
Độ phức tạp
Xấu nhất: O(log2n)

Bài tập

Bài 1

Minh họa giải thuật tìm kiếm tuần tự để tìm giá trị x = 36 trong dãy khóa sau:

42 23 65 11 87 36 94 50 79

Bài 2

Minh họa giải thuật tìm kiếm tuần tự để tìm giá trị x = 32 trong dãy khóa sau:

42 23 65 11 87 36 94 50 79

Bài tập

Bài 3

Minh họa giải thuật tìm kiếm nhị phân để tìm giá trị x = 79 trong dãy khóa sau:

11 23 36 42 50 65 68 79 87 94

Bài 4

Minh họa giải thuật tìm kiếm nhị phân để tìm giá trị x = 25 trong dãy khóa sau:

11 23 36 42 50 65 68 79 87 94

Tài liệu tham khảo

[1]. Giáo trình Cấu trúc dữ liệu và giải thuật – Lê Văn Vinh, NXB Đại học quốc gia TP HCM, 2013

[2]. Cấu trúc dữ liệu & thuật toán, Đỗ Xuân Lôi, NXB Đại học quốc gia Hà Nội, 2010.

[3]. Kỹ thuật lập trình, Học viện BCVT, 2002.

[4]. Robert Sedgewick, *Cẩm nang thuật toán*, NXB Khoa học kỹ thuật, 2004 .

[5]. PGS.TS Hoàng Nghĩa Tý, *Cấu trúc dữ liệu và thuật toán*, NXB xây dựng, 2014