Bài 9.3: Thuật toán tìm kiếm nhị phân

- ✓ Các đặc điểm của thuật toán
- ✓ Mã giả và triển khai mã thật
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Các đặc điểm

- Thuật toán tìm kiếm nhị phân có độ phức tạp O(logn).
- Là thuật toán tìm kiếm nhanh nhất trong các thuật toán tìm kiếm.
- Yêu cầu tiên quyết là tập hợp mẫu cần được sắp xếp trước khi tiến hành tìm kiếm. Mặc định sắp xếp tang dần.
- Biết chỉ số phần tử trái, phải của đoạn cần tìm là left và right. Quá trình tìm kiếm thực hiện như sau:

Các đặc điểm

- Nếu left <= right: So sánh giá trị cần tìm(x) với phần tử ở giữa tập hợp(phần tử mid).
  - Nếu giá trị x trùng với giá trị của mid, ta tìm thấy x trong tập hợp.
  - Nếu giá trị của mid < x, ta tiến hành tìm kiếm phía bên phải phần tử mid.
  - Nếu giá trị của mid > x, ta tiến hành tìm kiếm phía bên trái phần tử mid.
- Nếu left > right, không tìm thấy x, return -1.

Mã giả

➢ Sau đây là mã giả của thuật toán:

```
// thuật toán tìm kiếm nhị phân
// arr: mảng chứa các giá trị để tìm kiếm
// left: chỉ số phần tử trái cùng của mảng
// right: chỉ số phần tử phải cùng của đoạn
// x: giá trị cần tìm
int binarySearch (arr[], left, right, x) {
    if (left <= right) {
        mid = left + (right - left) / 2;
        if (arr[mid] == x) { // tìm thấy x trong mảng
            return mid;
        }
        if (arr[mid] < x) { // tìm phía bên phải arr[mid]
            return binarySearch(arr, mid + 1, right, x);
        } else { // tìm phía trái arr[mid]
            return binarySearch(arr, left, mid - 1, x);
        }
    }
    return -1; // không tìm thấy x trong mảng
}
```

Mã thật

➢ Sau đây là mã thật của thuật toán:

```
static <T extends Comparable<T>> int binarySearch(T[] arr, int left, int right, T x) {
    if (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid].compareTo(x) == 0) { // tìm thấy x trong mảng
            return mid;
        }
        if (arr[mid].compareTo(x) < 0) { // tìm phía bên phải arr[mid]
            return binarySearch(arr, left: mid + 1, right, x);
        } else { // tìm phía trái arr[mid]
            return binarySearch(arr, left, right: mid - 1, x);
        }
    }
    return -1;
}
```

Nội dung tiếp theo

Tìm kiếm trên cây nhị phân tìm kiếm