THUẬT TOÁN TÌM KIẾM

Mục lục:

1. Tìm kiếm tuyến tính (Linear search):

Ý tưởng: Duyệt tuần tự các phần tử trong mảng và so sánh giá trị cần tìm kiếm với từng phần tử trong mảng.

Các bài toán như tìm kiếm vị trí đầu tiên, cuối cùng, đếm số lần xuất hiện của phần tử trong mảng đều là biến đổi của thuật toán tìm kiếm tuyến tính.

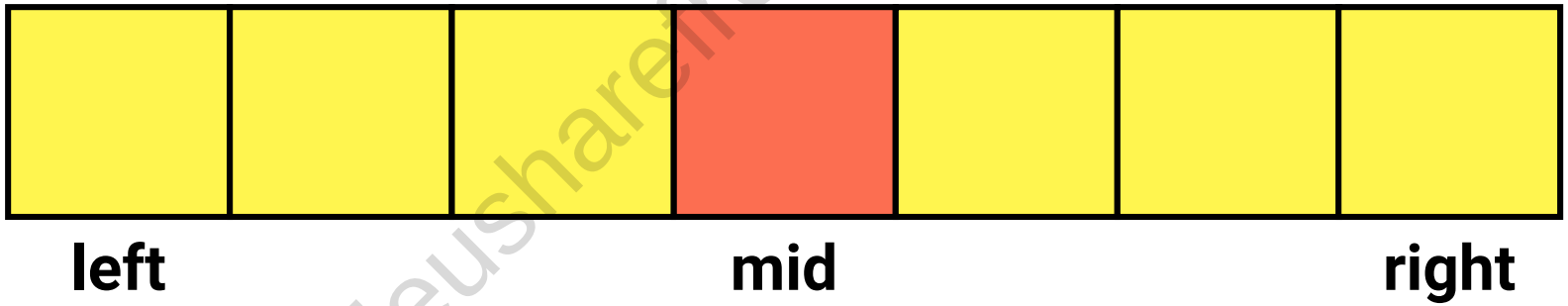
Code:

```
public static boolean linearSearch(int a[], int n, int x){
    for(int i = 0; i < n; i++){
        if (x == a[i]){
            return true;
        }
    }
    return false;
}
```

Độ phức tạp: **O(N)**

2. Tìm kiếm nhị phân (Binary search):

Ý tưởng: Tìm kiếm trong đoạn từ [left, right] của mảng, ở mỗi bước thuật toán tìm vị trí mid ở giữa đoạn left, right. Nếu phần tử cần tìm kiếm bằng phần tử ở vị trí mid thì kết luận là tìm thấy, nếu không ta có thể giảm 1 nửa đoạn tìm kiếm xuống và tiếp tục tìm kiếm ở bên trái hay bên phải của mid.



Điều kiện áp dụng: Mảng đã được sắp xếp.

2. Tìm kiếm nhị phân (Binary search):

Đây là một thuật toán cực kì hiệu quả và quan trọng, nếu học lập trình mà các bạn không biết thuật toán này thì bạn chỉ đơn thuần học ngôn ngữ lập trình chứ không phải học lập trình.

Code:

```
public static boolean linearSearch(int a[], int n, int x){
    int left = 0, right = n - 1;
    while(left <= right){
        int mid = (left + right) / 2;
        if(a[mid] == x)
            return true;
        else if(a[mid] < x)
            left = mid + 1; // Tìm ở bên phải
        else
            right = mid - 1; // Tìm ở bên trái
    }
    return false;
}
```

Độ phức tạp: **O(logN)** 🟢

3. Vị trí đầu tiên trong mảng tăng dần:

Tìm vị trí đầu tiên của phần tử X trong mảng đã được sắp xếp

```
public static int firstPos(int a[], int n, int x){
    int res = -1, left = 0, right = n - 1;
    while(left <= right){
        int mid = (left + right) / 2;
        if(a[mid] == x){
            res = mid; // cập nhật
            //Tìm thêm đáp án tốt hơn
            right = mid - 1;
        }
        else if(a[mid] < x)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return res;
}
```

Độ phức tạp: **O(logN)** 🟢

4. Vị trí cuối cùng trong mảng tăng dần:

Tìm vị trí cuối cùng của phần tử X trong mảng đã được sắp xếp

```
public static int lastPos(int a[], int n, int x){
    int res = -1, left = 0, right = n - 1;
    while(left <= right){
        int mid = (left + right) / 2;
        if(a[mid] == x){
            res = mid; // cập nhật
            //Tìm thêm đáp án tốt hơn
            left = mid + 1;
        }
        else if(a[mid] < x)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return res;
}
```

Độ phức tạp: **O(logN)** 🟢

5. Vị trí đầu tiên lớn hơn hoặc bằng X trong mảng tăng dần:

Tìm vị trí đầu tiên của phần tử lớn hơn hoặc bằng X trong mảng đã được sắp xếp

```
public static int lower(int a[], int n, int x){
    int res = -1;
    int left = 0, right = n - 1;
    while(left <= right){
        int mid = (left + right) / 2;
        if(a[mid] >= x){
            res = mid; // cập nhật
            //Tìm thêm đáp án tốt hơn
            right = mid - 1;
        }
        else
            left = mid + 1;
    }
    return res;
}
```

Độ phức tạp: **O(logN)** 🟢

6. Vị trí cuối cùng nhỏ hơn hoặc bằng X trong mảng tăng dần:

Tìm vị trí cuối cùng của phần tử nhỏ hơn hoặc bằng X trong mảng đã được sắp xếp

```
public static int upper(int a[], int n, int x){
    int res = -1;
    int left = 0, right = n - 1;
    while(left <= right){
        int mid = (left + right) / 2;
        if(a[mid] <= x){
            res = mid; // cập nhật
            //Tìm thêm đáp án tốt hơn
            left = mid + 1;
        }
        else
            right = mid - 1;
    }
    return res;
}
```

Độ phức tạp: **O(logN)** 🟢

7. Hàm binarySearch:

Hàm binarySearch cũng được xây dựng sẵn và bạn có thể áp dụng để tìm kiếm phần tử, hàm này sẽ trả về chỉ số >= 0 nếu phần tử tìm kiếm xuất hiện trong mảng.

Nếu không xuất hiện sẽ trả về chỉ số âm. Chỉ số này được xác định bằng cách chèn giá trị cần tìm kiếm vào mảng, sau khi chèn giả sử giá trị này nằm ở chỉ số x khi đó hàm trả về - x - 1.

Ví dụ:

```
public static void main(String[] args) {
        int[] a = {1, 2, 2, 4, 5};
        System.out.println(Arrays.binarySearch(a, 2));
        System.out.println(Arrays.binarySearch(a, 3));
}
```

OUTPUT

```
2
-4
```