Where Is The Marble

Big-O Blue - Lecture 12: Binary Search

Tóm tắt đề bài

Tóm tắt đề bài

N viên bi, trên mỗi viên bi có các con số. Sắp xếp các viên bi theo thứ tự tăng dần các con số ghi trên viên bi.

Với mỗi truy vấn X, xác định vị trí đầu tiên của số X trong dãy viên bi đã được sắp xếp. Nếu X không xuất hiện trong dãy, in ra ‘X not found’.

Mô tả Input/Output

Input:

- Có nhiều bộ test, mỗi bộ gồm: N - số lượng bi và Q - số lượng truy vấn.
- N dòng tiếp theo chứa các số ghi trên các viên bi. Các số không theo thứ tự nào.
- Q dòng tiếp theo chứa các truy vấn. Đảm bảo không có con số nào lớn hơn 10000 và không có số nào là số âm.
- Input sẽ kết thúc bằng bộ test có N = 0 và Q = 0

Output:

- Với mỗi case phải ghi “Case# ” và chỉ số case tương ứng
- “%X found at %Y”: nếu như viên bi màu đầu tiên với số x được tìm thấy ở vị trí Y. Các vị trí được đánh số là 1,2,3, ...N.
- “%X not found”: nếu như viên bi màu với số X không tồn tại.

Giải thích ví dụ

Ví dụ 1

Input:

```
4 1
2
3
5
1
5
5 2
1
3
3
3
1
2
3
0 0
```

Output:

```
CASE# 1:
5 found at 4
CASE# 2:
2 not found
3 found at 3
```

Original marbles	2	3	5	1
Sorted marbles	1	2	3	5
Index	1	2	3	4

Original marbles	1	3	3	3	1
Sorted marbles	1	1	3	3	3
Index	1	2	3	4	5

Hướng dẫn giải

Các bước giải

B1: Đọc vào N và Q. Nếu hai số N và Q đều = 0 thì kết thúc chương trình.

B2: Đọc vào mảng N phần tử và sắp xếp lại.

B3: Đọc vào từng truy vấn X. Sử dụng Binary Search First để tìm vị trí đầu tiên của X trong mảng đã được sắp xếp.

B4: In ra "%X not found" nếu hàm BinarySearchFirst trả về -1. Ngược lại thì in ra "%X found at %Y" với các giá trị X là giá trị truy vấn và Y là vị trí tương ứng tìm được.

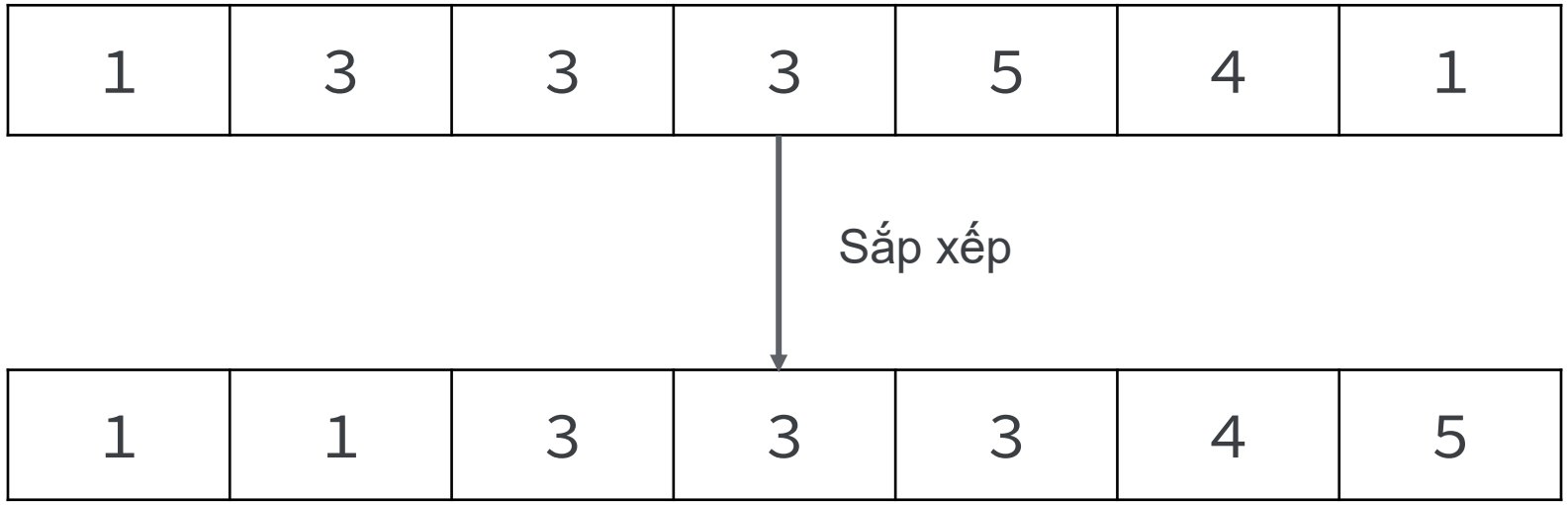
Độ phức tạp: **O(T*(N + Q)*log(N))**

Minh họa thuật toán

Minh họa ý tưởng

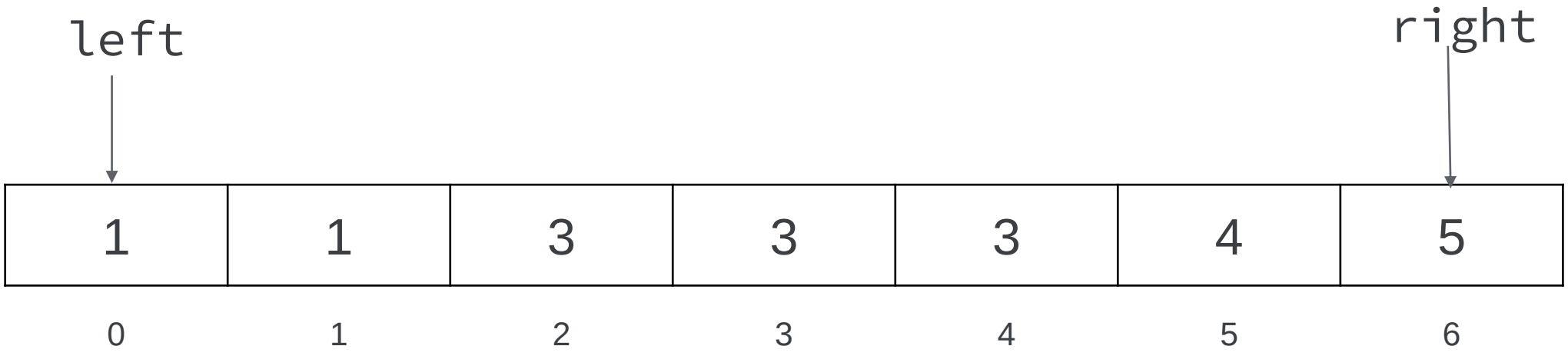
Input:

```
7 2
1
3
3
3
5
4
1
2
3
0 0
```



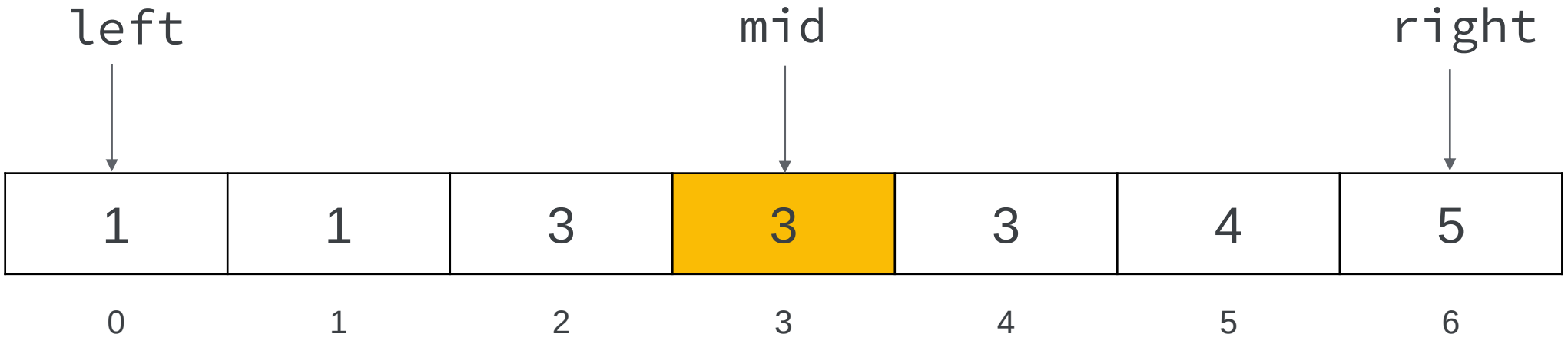
Khởi tạo (chuẩn bị cho truy vấn X = 2)

```
n = 7
left = 0
right = 6
```



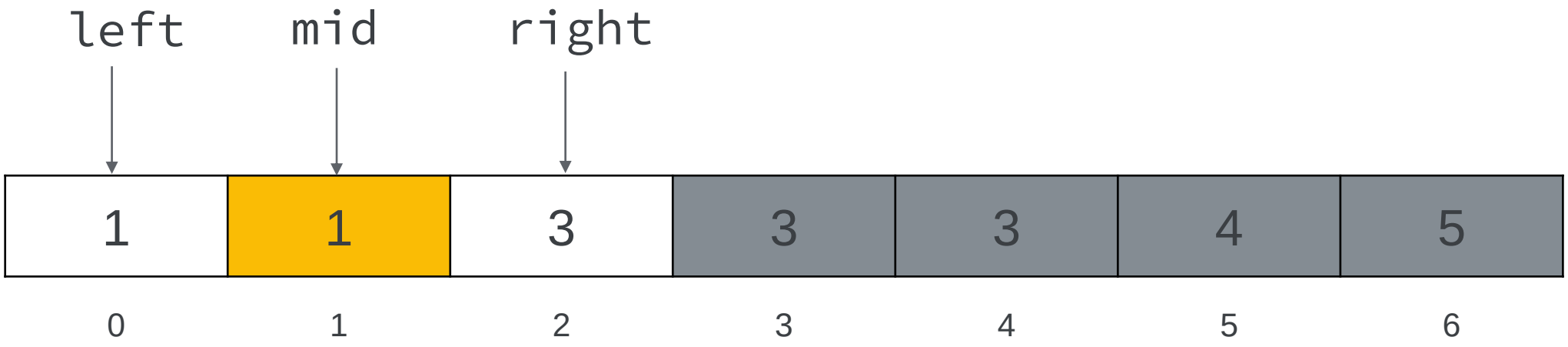
X = 2: Chạy thuật toán lần 1

```
mid = (left + right)/2 = (0 + 6)/2 = 3
X = 2 < a[mid]
Cập nhật: right = mid - 1 = 2
```



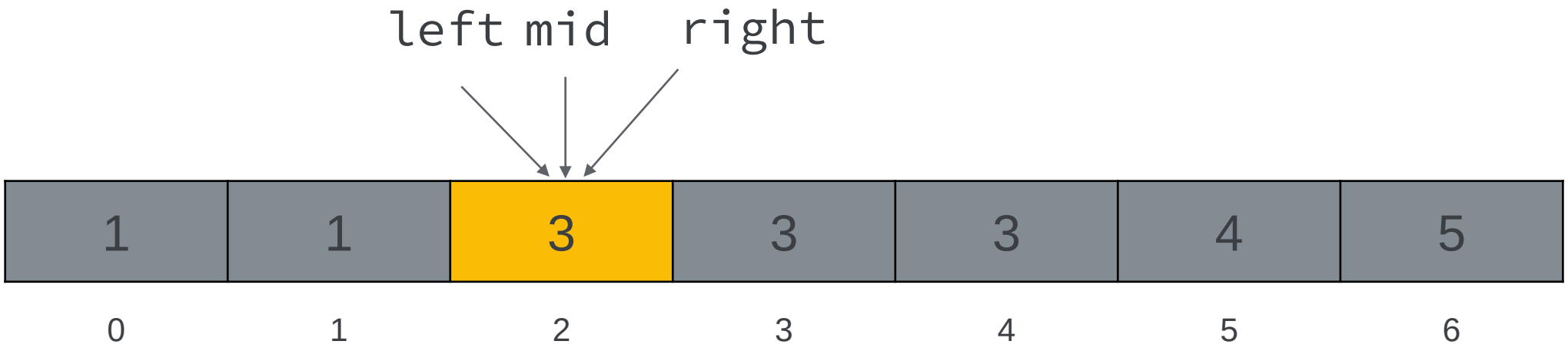
X = 2: Chạy thuật toán lần 2

```
mid = (left + right)/2 = (0 + 2)/2 = 1
X = 2 > a[mid]
Cập nhật: left = mid + 1 = 2
```



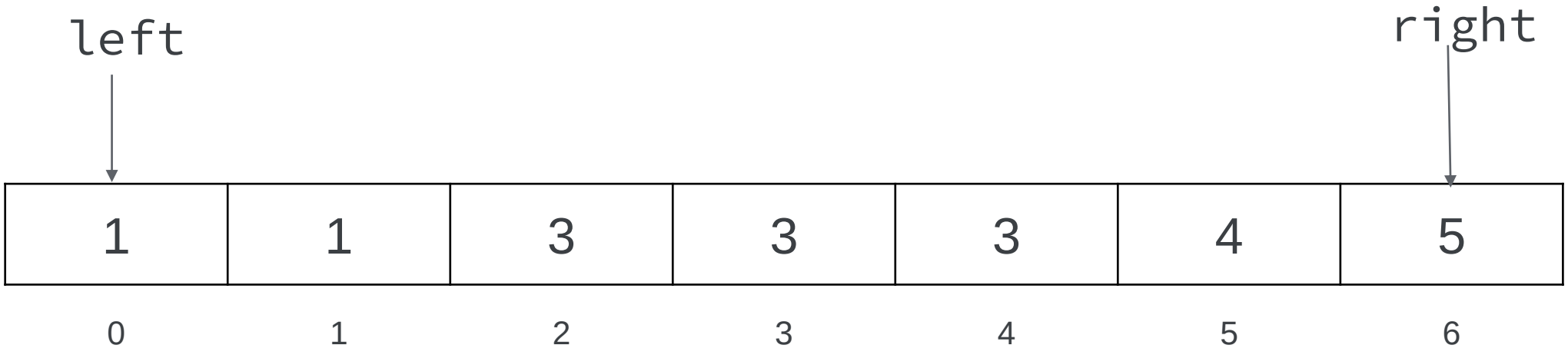
X = 2: Chạy thuật toán lần 3

```
mid = (left + right)/2 = (2 + 2)/2 = 2
X = 2 < a[mid]
Cập nhật: right = mid - 1 = 1.
left > right: return -1
```



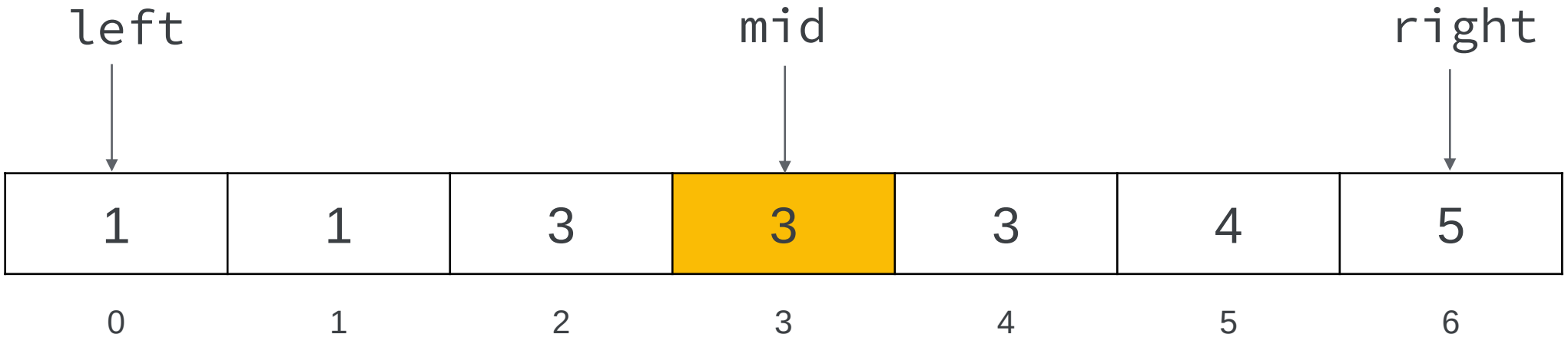
Khởi tạo (chuẩn bị cho truy vấn X = 3)

```
n = 7
left = 0
right = 6
```



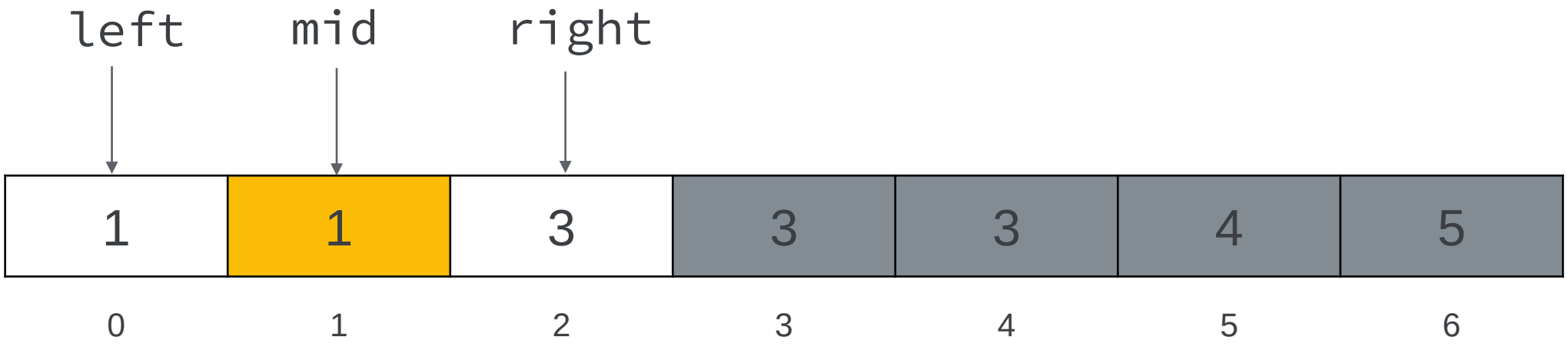
X = 3: Chạy thuật toán lần 1

```
mid = (left + right)/2 = (0 + 6)/2 = 3
X = 3 == a[mid] && (mid == left || a[mid-1] < X)?
Cập nhật: right = mid - 1 = 2
```



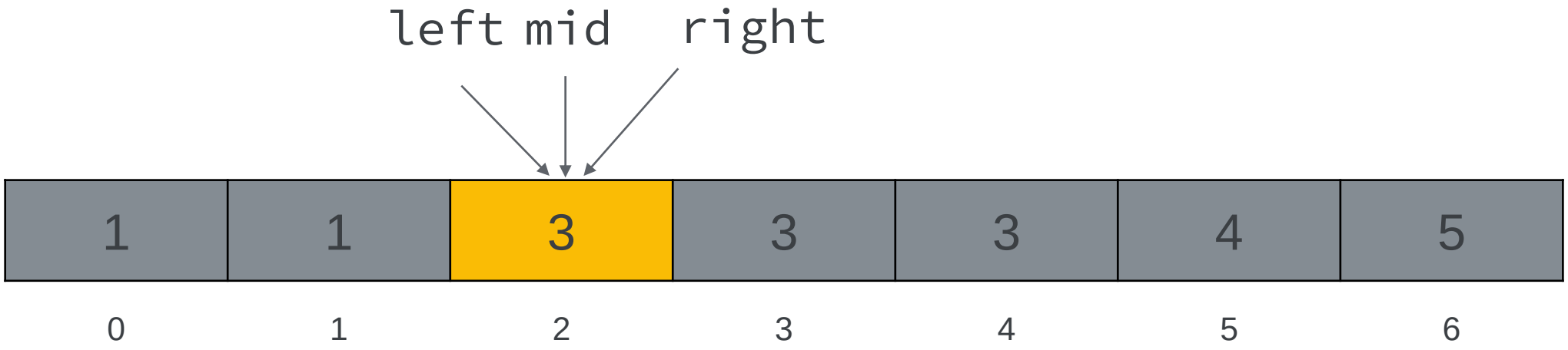
X = 3: Chạy thuật toán lần 2

```
mid = (left + right)/2 = (0 + 2)/2 = 1
X = 3 > a[mid]
Cập nhật: left = mid + 1 = 2
```



X = 3: Chạy thuật toán lần 3

```
mid = (left + right)/2 = (2 + 2)/2 = 2
X = 3 == a[mid]  && (mid == left || a[mid-1] < X)
→ return mid
```



Mã giả

Mã giả

```
test = 1
while True:
    read(n, q)
    if n == 0 and q == 0:
        break
    read(a)
    sort(a) #ascending order
    print('Case# ', test, ':\n')
    test += 1
    for i = 1 → q:
       read(X)
       res = BinarySearchFirst(a, 0, n - 1, X)
       if res == -1:
           print(X, ' not found')
       else:
           print(X, ' found at ', res+1) # index start from 1
```

Mã giả

```
def BinarySearchFirst(a, left, right, X):
    while left <= right:
        mid = left + (right - left)/2
        if a[mid] == X and (mid == left or a[mid-1] != X):
            return mid
        else if a[mid] < X:
            left = mid + 1
        else:
            right = mid - 1
    return -1
```

Thank you