Big-O Blue

LECTURE 13

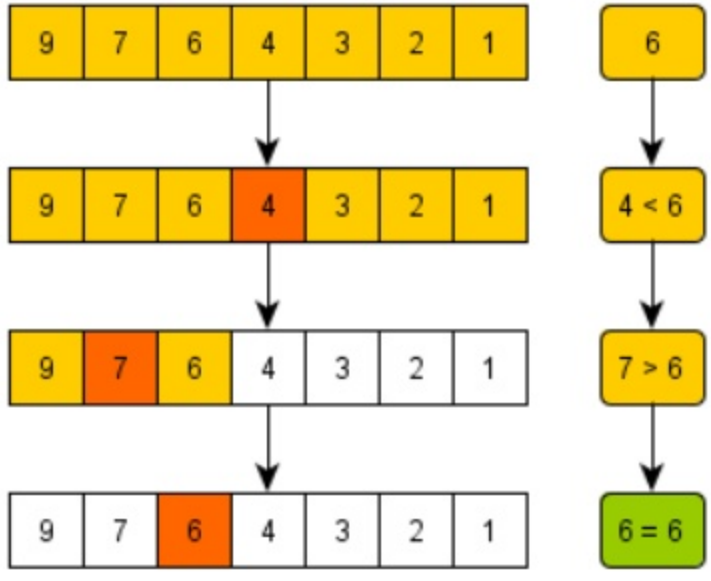
BINARY SEARCH

Phạm Nguyễn Sơn Tùng

Email: sontungtn@gmail.com

Định nghĩa Binary Search

Binary Search (tìm kiếm nhị phân hoặc tên gọi khác là chặt nhị phân) là phương pháp tìm kiếm một phần tử trong mảng đơn điệu.



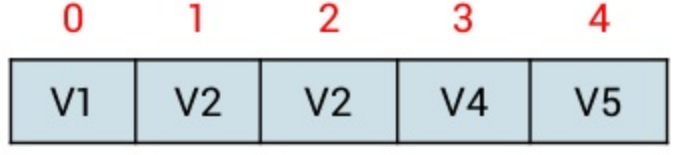
Độ phức tạp: O (LogN)

Ý tưởng của thuật toán

Mảng ban đầu đã được sắp xếp sẵn.

So sánh phần tử cần tìm và phần tử giữa mảng để giảm việc tìm kiếm.

Tùy giá trị cần tìm để giảm số lượng phần tử **bên phải** hay **bên trái.**



Quay trở lại bước so sánh trên đến khi còn 1 phần tử cuối cùng cần tìm trong mảng.

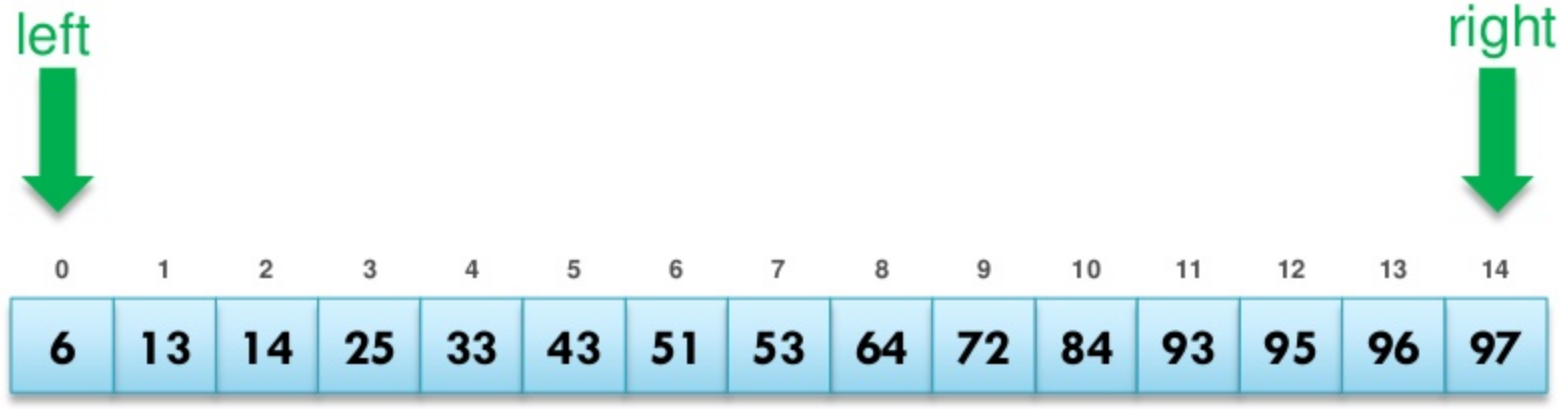
Bài toán minh họa

Cho mảng đã được sắp xếp **tăng dần**, hãy tìm vị trí chứa giá trị **x = 33** trong mảng.

Bước 0: Chuẩn bị dữ liệu

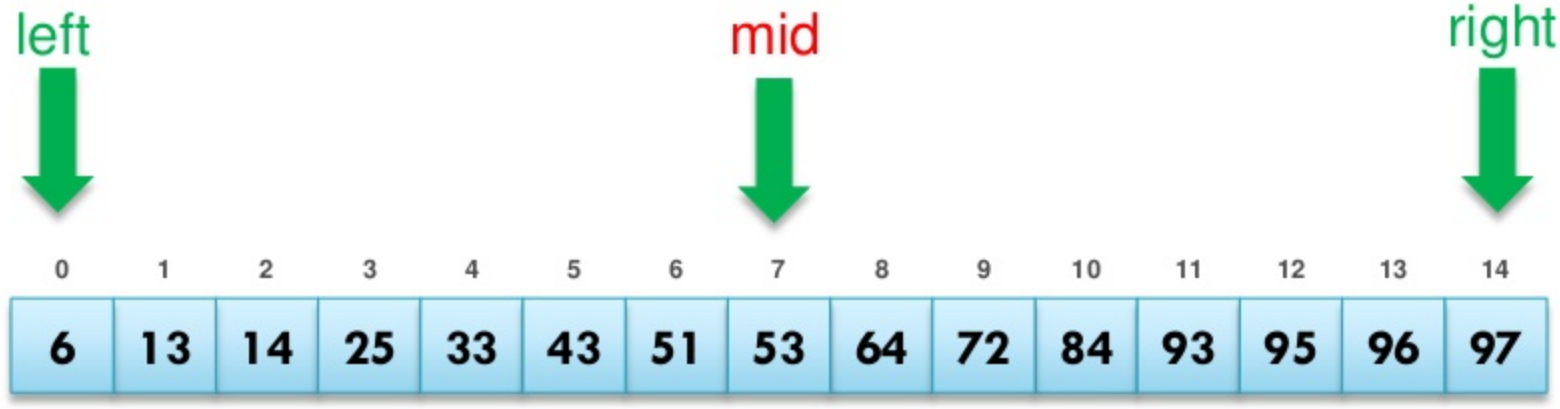
Đặt các giá trị ban đầu cho các biến.

- n = 15
- left = 0
- right = n – 1 = 14



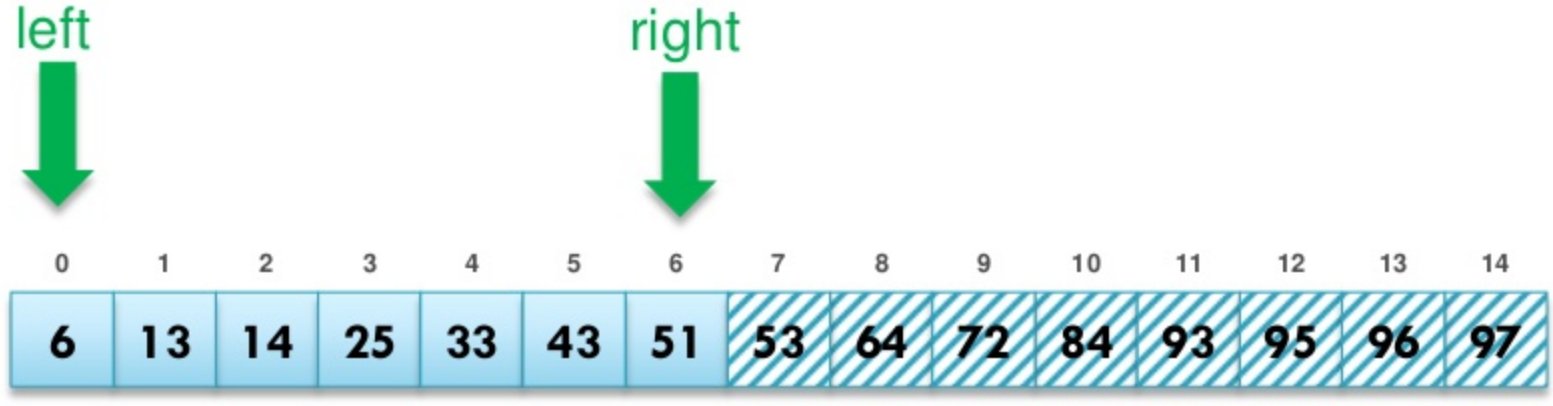
Bước 1: Chạy thuật toán lần 1

- mid = (left + right)/2 = 7
- x (33) < a[mid] (53)
- left = 0
- right = mid – 1 = 6



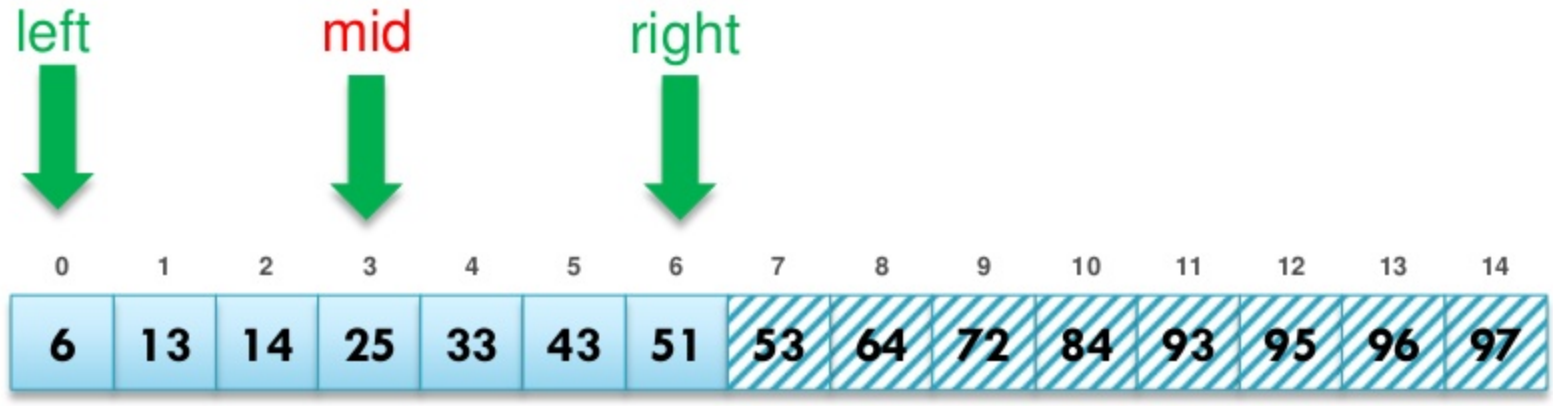
Bước 1: Chạy thuật toán lần 1

- mid = (left + right)/2 = 7
- x (33) < a[mid] (53)
- left = 0
- right = mid – 1 = 6



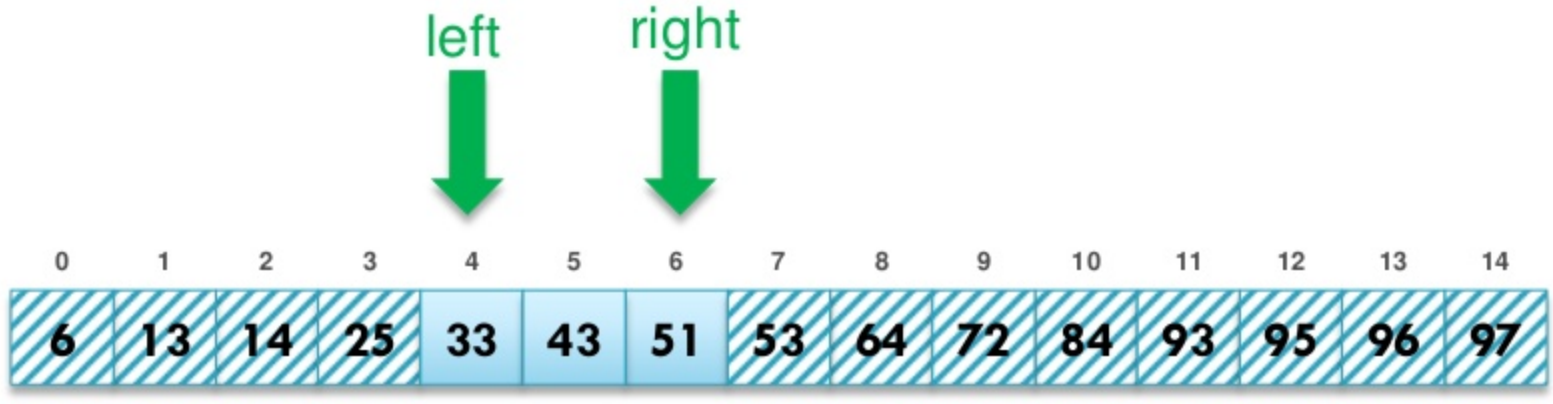
Bước 2: Chạy thuật toán lần 2

- mid = (left + right)/2 = 3
- x (33) > a[mid] (25)
- left = mid + 1 = 4
- right = 6



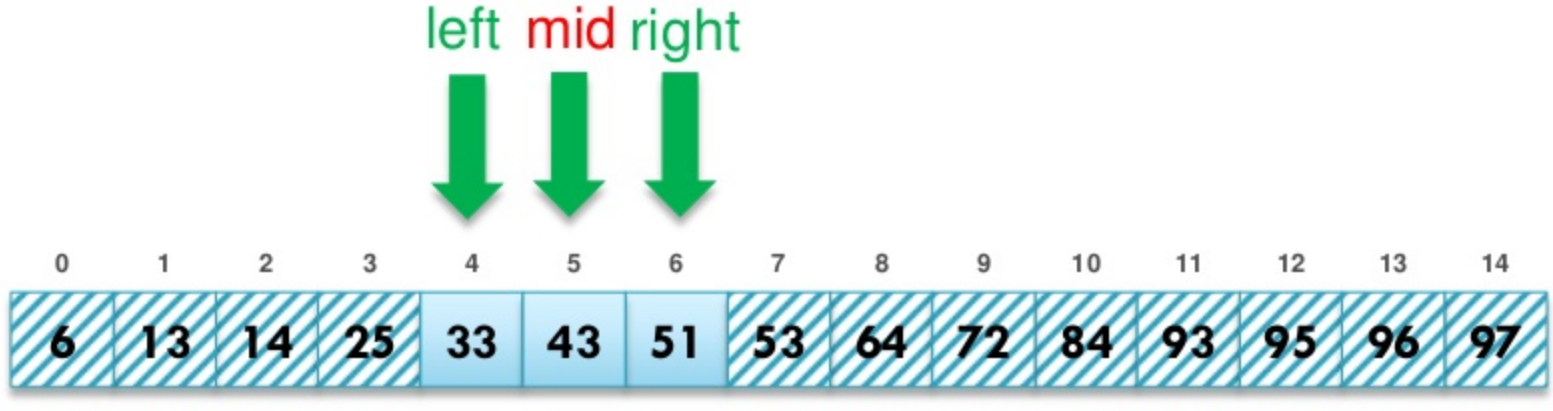
Bước 2: Chạy thuật toán lần 2

- mid = (left + right)/2 = 3
- x (33) > a[mid] (25)
- left = mid + 1 = 4
- right = 6



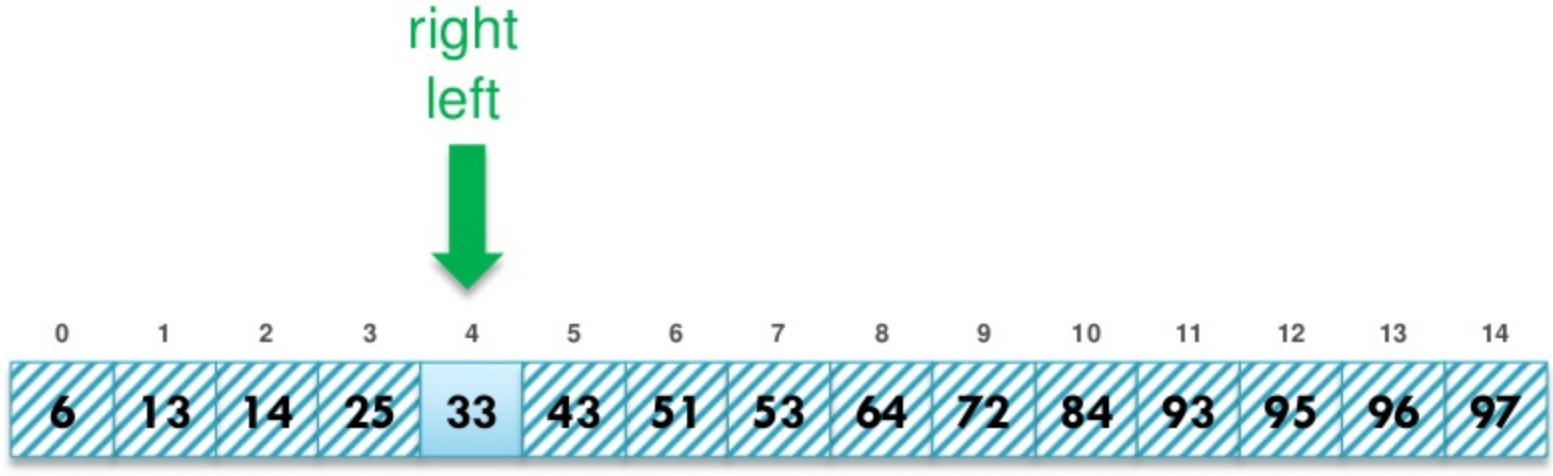
Bước 3: Chạy thuật toán lần 3

- mid = (left + right)/2 = 5
- x (33) < a[mid] (43)
- left = 4
- right = mid – 1 = 4



Bước 3: Chạy thuật toán lần 3

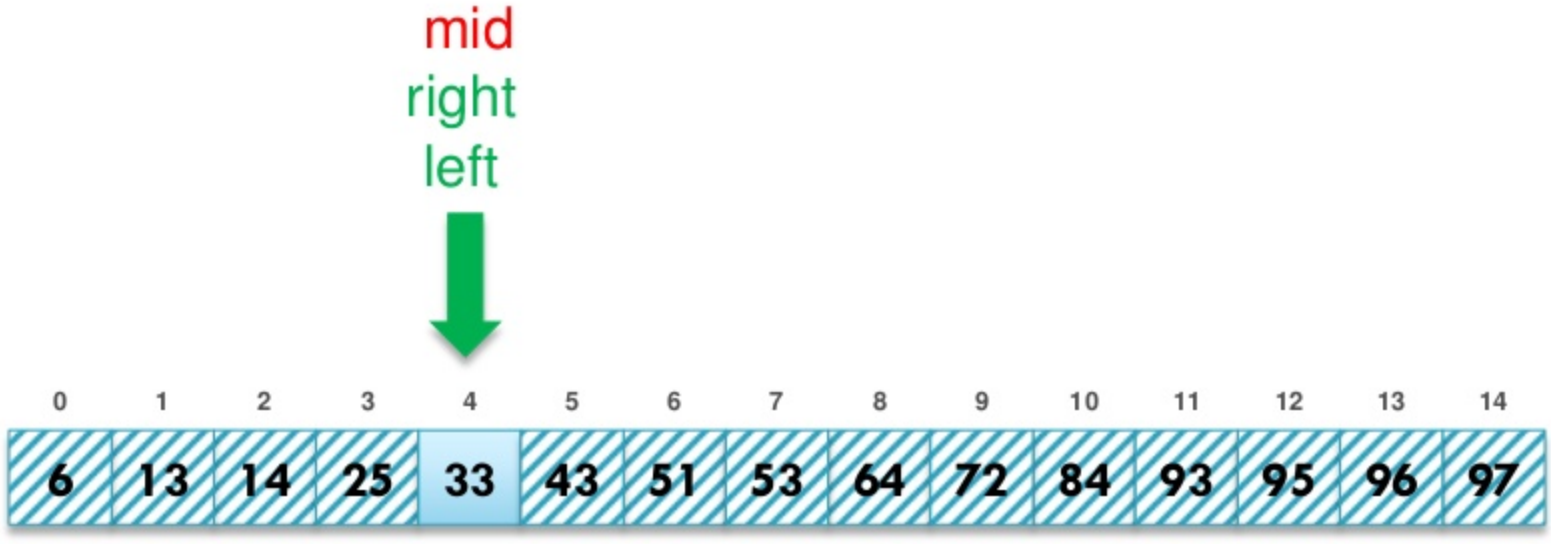
- mid = (left + right)/2 = 5
- x (33) < a[mid] (43)
- left = 4
- right = mid – 1 = 4



Bước 4: Chạy thuật toán lần 4

- mid = (left + right)/2 = 4
- x (33) == a[mid] (33)

Kết quả Tìm được vị trí của **x là 4**. Dừng thuật toán.



Một số nhận xét

Tìm kiếm nhị phân dựa vào **quan hệ giá trị** của các phần tử mảng để định hướng trong quá trình tìm kiếm, do vậy chỉ áp dụng được cho các dãy **đã có thứ tự**.

Khi mảng có biến động cần phải tiến hành sắp xếp lại. Tìm kiếm nhị phân cần phải xét đến thời gian sắp xếp lại mảng, thời gian sắp xếp này không nhỏ, phải cân nhắc khi thực hiện.

MÃ NGUỒN MINH HỌA BẰNG C++

Source Code Binary Search C++

```cpp
#include <iostream>
#include <vector>
using namespace std;
int binarySearch(vector<int> a, int left, int right, int x) {
    while (left <= right) {
        int mid = (left + right) / 2;
        if (x == a[mid])
            return mid;
        else if (x < a[mid])
            right = mid - 1;
        else if (x > a[mid])
            left = mid + 1;
    }
    return -1;
}
```

Source Code Binary Search C++

Hàm main.

```
int main() {
    freopen("INPUT.INP", "rt", stdin);
    vector<int> a;
    int n, x, value;
    cin >> n >> x;
    for (int i = 0; i < n; i++) {
        cin >> value;
        a.push_back(value);
    }
    int result = binarySearch(a, 0, n - 1, x);
    cout << result;
    return 0;
}
```

Binary Search (Đệ quy) C++

```
int binarySearch(vector<int> a, int left, int right, int x) {
    if (left <= right) {
        int mid = left + (right - left) / 2;
        if (a[mid] == x)
            return mid;
        if (a[mid] > x)
            return binarySearch(a, left, mid - 1, x);
        return binarySearch(a, mid + 1, right, x);
    }
    return -1;
}
```

MÃ NGUỒN MINH HỌA BẰNG PYTHON

Source Code Binary Search - python

```
def binarySearch(a, left, right, x):
    while left <= right:
        mid = (left + right) // 2
        if x == a[mid]:
            return mid
        elif x < a[mid]:
            right = mid - 1
        else:
            left = mid + 1
    return -1
```

```
if __name__ == '__main__':
    n, x = map(int, input().split())
    a = list(map(int, input().split()))
    result = binarySearch(a, 0, n-1, x)
    print(result)
```

Binary Search (Đệ quy)

```
def binarySearch(a, left, right, x):
    if left <= right:
        mid = (left + right) // 2
        if a[mid] == x:
            return mid;
        if a[mid] > x:
            return binarySearch(a, left, mid - 1, x)
        return binarySearch(a, mid + 1, right, x)
    return -1
```

MÃ NGUỒN MINH HỌA BẰNG JAVA

Source Code Binary Search Java

```
public static int binarySearch(int a[], int left, int right, int x) {
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (x == a[mid]) {
            return mid;
        }
        else if (x < a[mid]) {
            right = mid - 1;
        }
        else {
            left = mid + 1;
        }
    }
    return -1;
}
```

Source Code Binary Search Java

Hàm main.

```
public static void main (String[] args) {

    Scanner sc = new Scanner(System.in);

    int n = sc.nextInt();

    int x = sc.nextInt();

    int a[] = new int[n];

    for (int i = 0; i < n; i++) {

        a[i] = sc.nextInt();

    }

    System.out.print(binarySearch(a, 0, n - 1, x));

}
```

Binary Search (Đệ quy) Java

```
public static int binarySearch(int a[], int left, int right, int x) {
    if (left <= right) {
        int mid = left + (right - left) / 2;
        if (x == a[mid]) {
            return mid;
        }
        else if (x < a[mid]) {
            return binarySearch(a, left, mid - 1, x);
        }
        else {
            return binarySearch(a, mid + 1, right, x);
        }
    }
    return -1;
}
```

MỘT SỐ HÀM BINARY SEARCH KHÁC CẦN LƯU Ý

Binary Search (tìm phần tử đầu tiên) C++

```
int bsFirst(vector<int> &a, int left, int right, int x) {
    if (left <= right) {
        int mid = left + (right - left) / 2;
        if ((mid == left || x > a[mid - 1]) && a[mid] == x)
            return mid;
        else if (x > a[mid])
            return bsFirst(a, (mid + 1), right, x);
        else
            return bsFirst(a, left, (mid - 1), x);
    }
    return -1;
}
```

Binary Search (tìm phần tử cuối cùng) C++

```
int bsLast(vector<int> a, int left, int right, int x) {
    if (left <= right) {
        int mid = left + (right - left) / 2;
        if ((mid == right || x < a[mid + 1]) && a[mid] == x)
            return mid;
        else if (x < a[mid])
            return bsLast(a, left, (mid - 1), x);
        else
            return bsLast(a, (mid + 1), right, x);
    }
    return -1;
}
```

Binary Search (tìm phần tử đầu tiên) Python

```
def bsFirst(a, left, right, x):
    if left <= right:
        mid = (left + right) // 2
        if (mid == left or x > a[mid - 1]) and a[mid] == x:
            return mid
        elif x > a[mid]:
            return bsFirst(a, mid + 1, right, x)
        else:
            return bsFirst(a, left, mid - 1, x)
    return -1
```

Binary Search (tìm phần tử cuối cùng) Python

```
def bsLast(a, left, right, x):
    if left <= right:
        mid = (left + right) // 2
        if (mid == right or x < a[mid + 1]) and a[mid] == x:
            return mid
        elif x < a[mid]:
            return bsLast(a, left, mid - 1, x)
        else:
            return bsLast(a, mid + 1, right, x)
    return -1
```

Binary Search (tìm phần tử đầu tiên) Java

```
public static int bsFirst(int[] a, int left, int right, int x) {
    if (left <= right) {
        int mid = left + (right - left) / 2;
        if ((mid == left || x > a[mid - 1]) && a[mid] == x)
            return mid;
        else if (x > a[mid])
            return bsFirst(a, (mid + 1), right, x);
        else
            return bsFirst(a, left, (mid - 1), x);
    }
    return -1;
}
```

Binary Search (tìm phần tử cuối cùng) Java

```java
public static int bsLast(int[] a, int left, int right, int x) {
    if (left <= right) {
        int mid = left + (right - left) / 2;
        if ((mid == right || x < a[mid + 1]) && a[mid] == x) {
            return mid;
        }
        else if (x < a[mid])
            return bsLast(a, left, mid - 1, x);
        else
            return bsLast(a, mid + 1, right, x);
    }
    return -1;
}
```

DÙNG BINARY SEARCH TRONG THƯ VIỆN CỦA C++/PYTHON/JAVA

Hàm tìm kiếm nhị phân

binary_search: Trả về giá trị **đúng/sai** khi tìm kiếm phần tử.

```
int a[] = {1, 1, 2, 2, 2, 3, 4, 5, 7};
int n = 9;
vector<int> v(a, a + n);
int x = 3;
bool result = binary_search(v.begin(), v.end(), x);
```

true

~~binary_search~~: Python không có hàm binary search.

Hàm tìm cận dưới

lower_bound: Trả về phần tử **đầu tiên không bé hơn** giá trị khóa tìm kiếm **[first**, last).

```cpp
int a[] = {1, 1, 2, 2, 2, 3, 4, 5, 7};
int n = 9;
vector<int> v(a, a + n);
int x = 3;
vector<int>::iterator low_value;
low_value = lower_bound(v.begin(), v.end(), x);
int index = low_value - v.begin();
```

bisect_left: Trả về ***vị trí*** **đầu tiên không bé hơn** giá trị khóa tìm kiếm **[first**, last).

Cú pháp: bisect_left(a, x, lo = 0, hi = len(a))

```python
if __name__ == '__main__':
    a = [1, 1, 2, 2, 2, 3, 4, 5, 7]
    n, x = 9, 3
    pos = bisect.bisect_left(a, x, 0, n)
    # hoặc pos = bisect.bisect_left(a, x)
    print(pos)
```

(***NOTE***: *C++ trả về phần tử nhưng Python trả về index).*

Kết quả

value: 3
index: 5

Hàm tìm cận trên

upper_bound: Trả về phần tử **đầu tiên** lớn hơn giá trị **khóa tìm kiếm** [first, **last).**

```
int a[] = {1, 1, 2, 2, 2, 3, 4, 5, 7};
int n = 9;
vector<int> v(a, a + n);
int x = 3;
vector<int>::iterator upp_value;
upp_value = upper_bound(v.begin(),
v.end(), x);
int index = upp_value - v.begin();
```

bisect_right: Trả về vị trí **đầu tiên** lớn hơn giá trị **khóa tìm kiếm** [first, **last).**

```
if __name__ == '__main__':
    a = [1, 1, 2, 2, 2, 3, 4, 5, 7]
    n, x = 9, 3
    pos = bisect.bisect_right(a, x,
0, n)
    # hoặc pos =
bisect.bisect_right(a, x)
    print(pos)
```

*(**NOTE**: C++ trả về phần tử nhưng Python trả về index).*

value: 4
index: 6

Hàm tìm đoạn, khoảng

equal_range: Trả về đoạn [**first, last**) thuộc kết quả tìm kiếm.

```
int a[] = {1, 1, 2, 2, 2, 3, 4, 5, 7};
int n = 9;
vector<int> v(a, a + n);
int x = 2;
pair<vector<int>::iterator, vector<int>::iterator> bounds;
bounds = equal_range(v.begin(), v.end(), x);
cout << "bounds at positions "<< (bounds.first - v.begin());
cout << " and " << (bounds.second - v.begin()) << endl;
```

bounds at positions 2 and 5

python ***equal_range:*** *Python không có hàm tương ứng, dùng bisect_left và bisect_right để thực hiện thức năng này.*

THƯ VIỆN BINARY SEARCH JAVA

Hàm tìm kiếm nhị phân

Collections.binarySearch/Arrays.binarySearch: trả về vị trí phần tử có giá trị bằng giá trị tìm kiếm (nếu có nhiều giá trị thì trả về vị trí bất kì).

Nếu không tìm thấy sẽ trả về – *insertion_point* – 1 trong đó *insertion_point* là vị trí phần tử đầu tiên lớn hơn giá trị tìm kiếm.

Nghĩa là nếu phần tử tìm kiếm tồn tại thì kết quả trả về sẽ không âm.

```
int[] a = new int[]{1, 1, 2, 2, 2, 3, 4, 5, 7};
int result = Arrays.binarySearch(a, 3);
System.out.print(result);
```

5

Hàm tìm kiếm nhị phân

Java không có các hàm tương ứng với lower_bound, upper_bound như của C++/Python. Phải tự cài đặt bằng tay.

Bên dưới là 2 đoạn code tham khảo lowerBound và upperBound trong Java. Dùng để tìm vị trí phần tử đầu tiên trong nửa đoạn **[left, right)** của mảng a có giá trị **không nhỏ hơn (lowerBound)** hoặc **lớn hơn (upperBound)**.

Nếu không có giá trị thỏa mãn thì trả về **right**.

Code tham khảo lowerBound (Java)

```java
public static int lower_bound(int[] a, int left, int right, int x) {
    int pos = right;
    while (left < right) {
        int mid = left + (right - left) / 2;
        if (a[mid] >= x) {
            pos = mid;
            right = mid;
        }
        else {
            left = mid + 1;
        }
    }
    return pos;
}
```

Code tham khảo upperBound (Java)

```java
public static int upper_bound(int[] a, int left, int right, int x) {
    int pos = right;
    while (left < right) {
        int mid = left + (right - left) / 2;
        if (a[mid] > x) {
            pos = mid;
            right = mid;
        }
        else {
            left = mid + 1;
        }
    }
    return pos;
}
```

Hỏi đáp