#include <iostream>

#include <vector>

using namespace std;

// Hàm tìm fixed point trong mảng đã cho

int find\_fixed\_point(vector<int>& arr) {

int left = 0;

// Chỉ số bắt đầu của mảng

int right = arr.size() - 1;

// Chỉ số kết thúc của mảng

// Duyệt mảng với phương pháp tìm kiếm nhị phân

while (left <= right) { int mid = left + (right - left) / 2;

// Tính chỉ số giữa

if (arr[mid] == mid) {

// Nếu arr[mid] == mid, ta đã tìm thấy fixed point

return mid;

// Trả về chỉ số mid

} else if (arr[mid] < mid) {

// Nếu arr[mid] < mid, fixed point nằm bên phải mid

left = mid + 1;

// Di chuyển left đến mid + 1 }

else {

// Nếu arr[mid] > mid, fixed point nằm bên trái mid

right = mid - 1;

// Di chuyển right đến mid - 1

}

}

return -1;

// Không tìm thấy fixed point, trả về -1

} int main() {

vector<int> arr = {-10, -5, 0, 3, 7, 9, 12, 15};

// Ví dụ mảng đã sắp xếp

int result = find\_fixed\_point(arr);

// Gọi hàm tìm fixed point

// In kết quả tìm được

if (result != -1) { cout << "Chỉ số i là: " << result << endl;

// In ra chỉ số i nếu tìm thấy

} else { cout << "Không có fixed point trong mảng đã cho." << endl;

// Thông báo không tìm thấy fixed point nếu trả về -1

}

return 0;

**Bài 2:**

**Pseodocode:**

1.Tạo hàm binarySearch ( arr, left, right, x) :Hàm này thực hiện tìm kiếm nhị phân trong một phần của mảng được chỉ định từ chỉ mục left đến chỉ mục right. Sử dụng một vòng lặp while để thực hiện tìm kiếm nhị phân:

function binarySearch(arr, left, right, x)

while left <= right do

mid = left + (right - left) / 2

*(Tính chỉ mục của phần tử giữa (mid) bằng cách lấy trung bình cộng của left và right.)*

if arr[mid] = x then

return mid

else if arr[mid] < x then

left = mid + 1

else

right = mid - 1

end while

return -1 // Không tìm thấy x trong mảng

*( So sánh giá trị của phần tử giữa với giá trị x: Nếu giá trị của phần tử giữa bằng x, trả về chỉ mục của phần tử này.*

*Nếu giá trị của phần tử giữa nhỏ hơn x, di chuyển left sang phải của mid và tiếp tục tìm kiếm trong nửa phải của mảng.*

*Nếu giá trị của phần tử giữa lớn hơn x, di chuyển right sang trái của mid và tiếp tục tìm kiếm trong nửa trái của mảng.*

*Nếu không tìm thấy x trong phần của mảng này, trả về -1.)*

1. Tạo hàm findInInfiniteArray(arr, x): Hàm này tìm kiếm vị trí của giá trị x trong mảng vô hạn.

function findInInfiniteArray(arr, x)

left = 0

right = 1

*(Bắt đầu với left = 0 và right = 1.)*

while arr[right] < x do

left = right

right = right \* 2

end while

return binarySearch(arr, left, right, x)

*(Sử dụng một vòng lặp while để tăng giá trị của right lên gấp đôi cho đến khi arr[right] >= x. Điều này đảm bảo rằng chúng ta đang nằm trong phần tử có giá trị lớn hơn hoặc bằng x.*

*Sau khi đã xác định phạm vi chứa x, gọi hàm binarySearch để tìm vị trí cụ thể của x trong phạm vi từ left đến right.)*

**Example usage:**

arr là mảng vô hạn đã được sắp xếp

x là giá trị cần tìm kiếm

index = findInInfiniteArray(arr, x)

if index != -1 then

output "Found at index: " + index

else

output "Not found"

*(Trong phần sử dụng ví dụ, chúng ta đưa ra một mảng đầu vào và giá trị cần tìm kiếm x.*

*Gọi hàm findInInfiniteArray() để tìm vị trí của x trong mảng và in ra kết quả tìm kiếm.)*

Thuật toán này có độ phức tạp thời gian O(log n), với n là vị trí của giá trị x trong mảng, bởi vì chúng ta thực hiện tìm kiếm nhị phân trên một phạm vi có độ dài log n.