Aim:- Application Of Binary Search Technique

Theory:-

Binary search is a super-fast method for finding an item in a sorted list. Think of it as the "divide and conquer" approach to searching.

How It Works

Instead of checking items one by one, binary search works by:

1. Jumping to the middle of the sorted list.
2. Comparing the middle item to what you're looking for.

- If it's a match, you're done!
- If your item is smaller, you ignore the entire right half of the list.
- If your item is larger, you ignore the entire left half.

3. Repeating this process—jumping to the middle of the remaining section—until you find your item or run out of places to look. This is exactly how you'd find a word in a dictionary. You don't start at page one; you open to the middle and decide which half to search next.

Key Points

- Main Requirement: The list must be sorted first. This is non-negotiable.
- Main Advantage: It's incredibly efficient (logarithmic time, or O(logn)). For a list with a million items, it takes at most 20 guesses. A regular one-by-one search could take a million.

Common Uses:

- Searching for a contact in your phone (which is sorted alphabetically).
- Finding a specific file in a sorted folder.
- Code debugging (e.g., Git's bisect command to find a bad commit).
- Guessing a number in a "higher or lower" game.

Examples:-

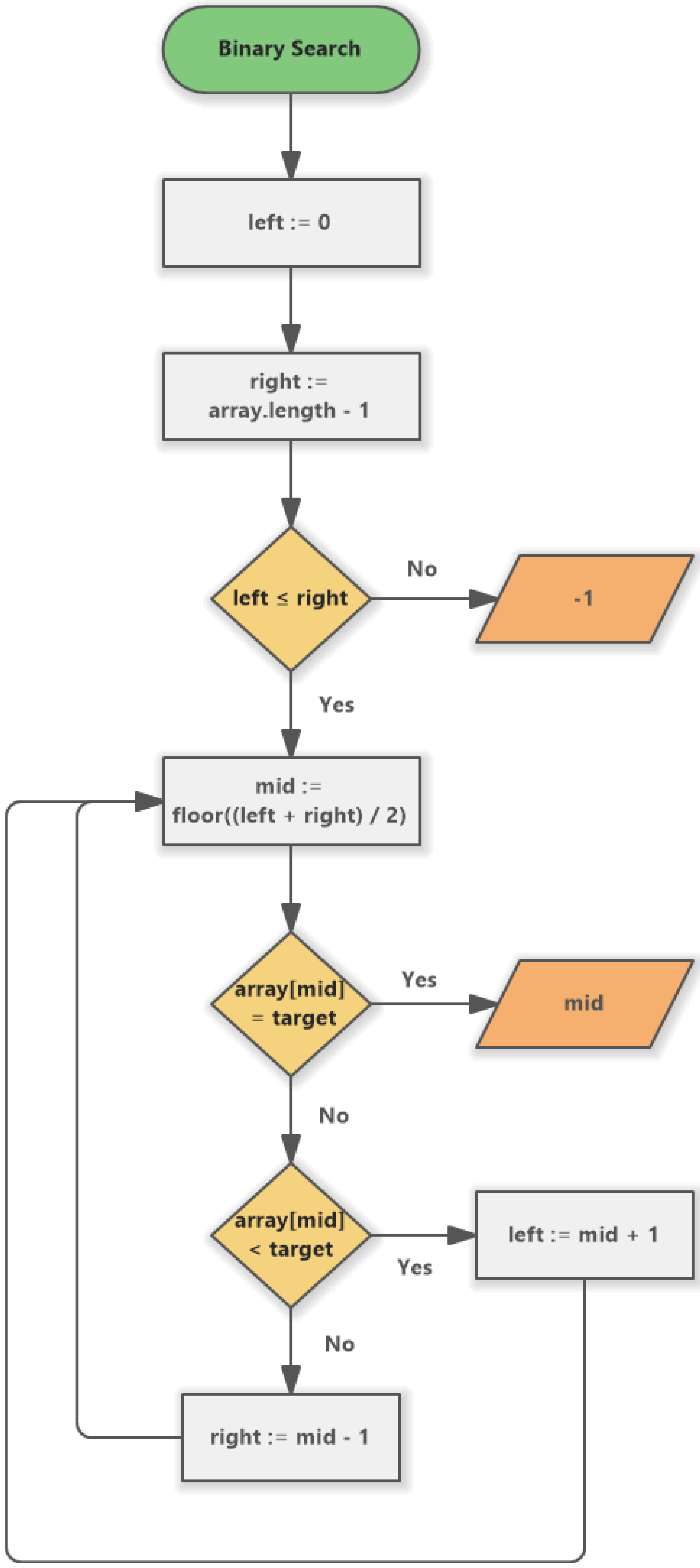
Everyday Examples

- Dictionary: Finding a word by opening to the middle. 📖
- Number Guessing Game: Guessing the middle number to eliminate half the options.
- Library Shelf: Finding a book sorted by author's last name.

Tech Examples

- Database Search: Quickly finding a user by their ID in a sorted database.
- Search Autocomplete: Suggesting search terms as you type by filtering a sorted list.
- Debugging Code (git bisect): Efficiently finding the exact code change that introduced a bug.

Flowchart:-


Binary Search
left := 0
right := array.length - 1
left ≤ right
No
-1
Yes
mid := floor((left + right) / 2)
array[mid] = target
Yes
mid
No
array[mid] < target
Yes
left := mid + 1
No
right := mid - 1

Code:-

```
#include <iostream>

using namespace std;

double sqrtBinarySearch(double n, double precision = 0.00001) {
    if (n < 0) {
        return -1.0;
    }
    if (n == 0 || n == 1) {
        return n;
    }

    double low = 0;
    double high = n;
    double mid;

    while ((high - low) > precision) {
        mid = low + (high - low) / 2;

        if (mid * mid > n) {
            high = mid;
        } else {
            low = mid;
        }
    }

    return low + (high - low) / 2;
}

int main() {
    double number;
    cout << "Enter a non-negative number: ";
    cin >> number;

    if (number < 0) {
        cout << "Cannot calculate the square root of a negative number." << endl;
    } else {
        double result = sqrtBinarySearch(number);
        cout << "The square root of " << number << " is approximately: " << result << endl;
    }

    return 0;
}
```

Output:-

https://github.com/lakshyajain1508/sjcem/blob/main/DSA/Lecture01/OutOfBST(Square%20Root).png

Edit Export

Pub: 14 Sep 2025 19:11 UTC
Views: 17

new · what · how · langs · contacts