Lecture 8: Binary Search Variants

Date: 09/14/2023

Find Index

Problem Statement: Given an sorted array of unique integers and a target element, find its index.

- **Input**: [0, 1, 5, 7, 9, 10] **Target:** 9
- Output: 4

- **Input**: [0, 1, 5, 7, 9, 10] **Target**: 6
- Output: -1

- **Input**: [10, 11, 12, 13, 14, 15, 16, 17] **Target**: 12
- **Output**: 2

```
int find_index(vector<int> nums, int target){
    int low = 0;
    int high = nums.size() - 1;

    while(low <= high){
        int mid = (low + high) / 2;

        if(target < nums[mid])
            high = mid-1;
        else if(target > nums[mid])
            low = mid+1;
        else
            return mid;
    }

    return -1;
}
```

The classical use of the Binary Search algorithm is to find out if an element exists in an array or not. But, there is more to it.
Following are some use cases of binary search like algorithms that can help us solve more problems.

Left Bound

Problem Statement: Given an array of integers, find out the leftmost index of the target element (if it exists).
Example:

- **Input**: [0, 1, 5, 5, 7, 9, 9, 9, 10] **Target:** 9
- Output: 5

- **Input**: [0, 1, 5, 5, 7, 9, 9, 9, 10] **Target**: 6
- Output: -1

- **Input**: [10, 11, 12, 13, 14, 15, 16, 17] **Target**: 12
- **Output**: 2

```
int lower_bound(vector<int> nums, int target){
    int low = 0;
    int high = nums.size() - 1;
    int ans = -1;

    while(low <= high){
        int mid = (low + high) / 2;

        if(target < nums[mid])
            high = mid-1;
        else if(target > nums[mid])
            low = mid+1;
        else{ // target == nums[mid]
            ans = mid;
            high = mid-1;
        }
    }
    return ans;
}
```

Right Bound

Problem Statement: Given an array of integers, find out the leftmost index of the target element (if it exists).
Example:

- **Input**: [0, 1, 5, 5, 7, 9, 9, 9, 10] **Target:** 9
- Output: 7

- **Input**: [0, 1, 5, 5, 7, 9, 9, 9, 10] **Target**: 6
- Output: -1

- **Input**: [10, 11, 12, 13, 14, 15, 16, 17] **Target**: 12
- **Output**: 2

```
int upper_bound(vector<int> nums, int target){
    int low = 0;
    int high = nums.size() - 1;
    int ans = -1;

    while(low <= high){
        int mid = (low + high) / 2;

        if(target < nums[mid])
            high = mid-1;
        else if(target > nums[mid])
            low = mid+1;
        else{
            ans = mid;
            low = mid+1;
        }
    }

    return ans;
}
```

Least Element Greater Than Target

Problem Statement: Given an array of integers, find out the leftmost index of the target element (if it exists).
Example:

- **Input**: [0, 1, 5, 5, 7, 9, 9, 9, 10] **Target:** 8
- Output: 5

- **Input**: [0, 1, 5, 5, 7, 9, 9, 9, 10] **Target**: 15
- Output: -1

- **Input**: [10, 11, 12, 13, 14, 15, 16, 17] **Target**: 12
- Output: 3

```
int least_greater(vector<int> nums, int target){
    int low = 0;
    int high = nums.size() - 1;
    int ans = -1;

    while(low <= high){
        int mid = (low + high) / 2;

        if(target < nums[mid]){
            ans = mid;
            high = mid-1;
        }
        else if(target > nums[mid])
            low = mid+1;
        else
            low = mid+1;
    }

    return ans;
}
```

Greatest Element Less Than Target

Problem Statement: Given an array of integers, find out the leftmost index of the target element (if it exists).
Example:

- **Input**: [0, 1, 5, 5, 7, 9, 9, 9, 10] **Target:** 7
- Output: 3

- **Input**: [0, 1, 5, 5, 7, 9, 9, 9, 10] **Target**: 0
- Output: -1

- **Input**: [10, 11, 12, 13, 14, 15, 16, 17] **Target**: 12
- Output: 1

```
int greatest_less(vector<int> nums, int target){
    int low = 0;
    int high = nums.size() - 1;
    int ans = -1;

    while(low <= high){
        int mid = (low + high) / 2;

        if(target < nums[mid])
            high = mid-1;
        else if(target > nums[mid]){
            ans = mid;
            low = mid+1;
        }
        else
            high = mid-1;
    }

    return ans;
}
```

One Sided

Suppose we don't know the length of the input array. (example, we have input as a stream of numbers). Then how do we perform binary search?

```
int one_sided(vector<int> nums, int target){
    int low = 1;
    int high = 1;

    while(high < nums.size() && target > nums[high]){
        low = high;
        high *= 2;
    }

    // normal binary search on bound low to high
}
```

Applications

⇒ https://leetcode.com/problems/maximum-count-of-positive-integer-and-negative-integer/

⇒ https://www.techiedelight.com/find-number-1s-sorted-binary-array/

Resources