

◀ Back to Chapter

☰

< Previous

Next >

Background

▶

The goal of this chapter is to provide a high-level overview of the various search algorithms and their applications.

Template I

▼

This chapter shows a snippet of code for the first template.

☑

A

 Binary Search Template I

Sqrt(x)

Guess Number Higher or Lower II

Search in Rotated Sorted Array

Template II

▼

This chapter shows a snippet of code for the second template.

☑

A

 Binary Search Template II

First Bad Version

Find Peak Element

Find Minimum in Rotated Sorted Array

Template III

▼

This chapter shows a snippet of code for the third template.

☑

A

 Binary Search Template III

Search for a Range

Find K Closest Elements

Find Peak Element

Template Analysis

▶

This chapter sums up the key attributes and distinguishing syntax of the templates.

Conclusion

▶

Binary Search is an immediate predecessor to the more advanced search algorithms.

More Practices

▶

In this chapter, we have provided a list of additional problems to practice the search algorithms.

A Binary Search Template III

[Report Issue](#)

Template #3:

C++

Java

Python

Copy

```
1 int binarySearch(int[] nums, int target) {
2     if (nums == null || nums.length == 0)
3         return -1;
4
5     int left = 0, right = nums.length - 1;
6     while (left + 1 < right){
7         // Prevent (left + right) overflow
8         int mid = left + (right - left) / 2;
9         if (nums[mid] == target) {
10             return mid;
11         } else if (nums[mid] < target) {
12             left = mid;
13         } else {
14             right = mid;
15         }
16     }
17
18     // Post-processing:
19     // End Condition: left + 1 == right
20     if(nums[left] == target) return left;
```

Template #3 is another unique form of Binary Search.

Key Attributes:

- An alternative way to implement Binary Search
- Use element's neighbors to determine if condition is met and decide whether to go left or right
- Gurantees Search Space is at least 3 in size at each step
- Post-processing required. Loop/Recursion ends when you have 2 elements left. Need to assess if the remaining elements meet the condition.

Distinguishing Syntax:

- Initial Condition: `left = 0, right = length-1`
- Termination: `left + 1 == right`
- Searching Left: `right = mid`
- Searching Right: `left = mid`