

Background

▶

The goal of this chapter is to provide a template for binary search.

Template I

▼

This chapter shows a snippet of the binary search template.

Binary Search Template I

Sqrt(x)

Guess Number Higher or Lower

Search in Rotated Sorted Array

Template II

▼

This chapter shows a snippet of the binary search template.

Binary Search Template II

First Bad Version

Find Peak Element

Find Minimum in Rotated Sorted Array

Template III

▼

This chapter shows a snippet of the binary search template.

Binary Search Template III

Search for a Range

Find K Closest Elements

Find Peak Element

Template Analysis

▶

This chapter sums up the binary search template.

Conclusion

▶

Binary Search is an immediate application of the divide and conquer technique.

More Practices

▶

In this chapter, we have provided a list of problems that can be solved using the binary search template.

# A Binary Search Template I

Report Issue

## Template #1:

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 <= right){
7         // Prevent (left + right) overflow
8         int mid = left + (right - left) / 2;
9         if(nums[mid] == target){ return mid; }
10        else if(nums[mid] < target) { left = mid + 1; }
11        else { right = mid - 1; }
12    }
13
14    // End Condition: left > right
15    return -1;
16 }
```

Template #1 is the most basic and elementary form of Binary Search. It is the standard Binary Search Template that most high schools or universities use when they first teach students computer science. Template #1 is used to search for an element or condition which can be determined by *accessing a single index* in the array.

## Key Attributes:

- Most basic and elementary form of Binary Search
- Search Condition can be determined without comparing to the element's neighbors (or use specific elements around it)
- No post-processing required because at each step, you are checking to see if the element has been found. If you reach the end, then you know the element is not found

## Distinguishing Syntax:

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