12. 二分查找

二分查找的前提

1. 目标函数单调性（单调递增 or 单调递减）
2. 存在上下边界（ bounded ）
3. 能够通过索引访问 (index accessible)

代码模版

```
left, right = 0, len(array) - 1

while left <= right :
    mid = (left+right)/2
    if array[mid] == target:
        # find the target!!!
        break or return result
    elif array[mid] < target:
        left = mid + 1
    else:
        right = mid - 1
```

```
int binarySearch(const vector<int> &nums, int target) {
  int left = 0, right = (int)nums.size() - 1;
  while (left <= right) {
    int mid = left + (right - left) / 2;
    if (nums[mid] == target)
      return mid;
    else if (nums[mid] < target)
      left = mid + 1;
    else
      right = mid - 1;
```

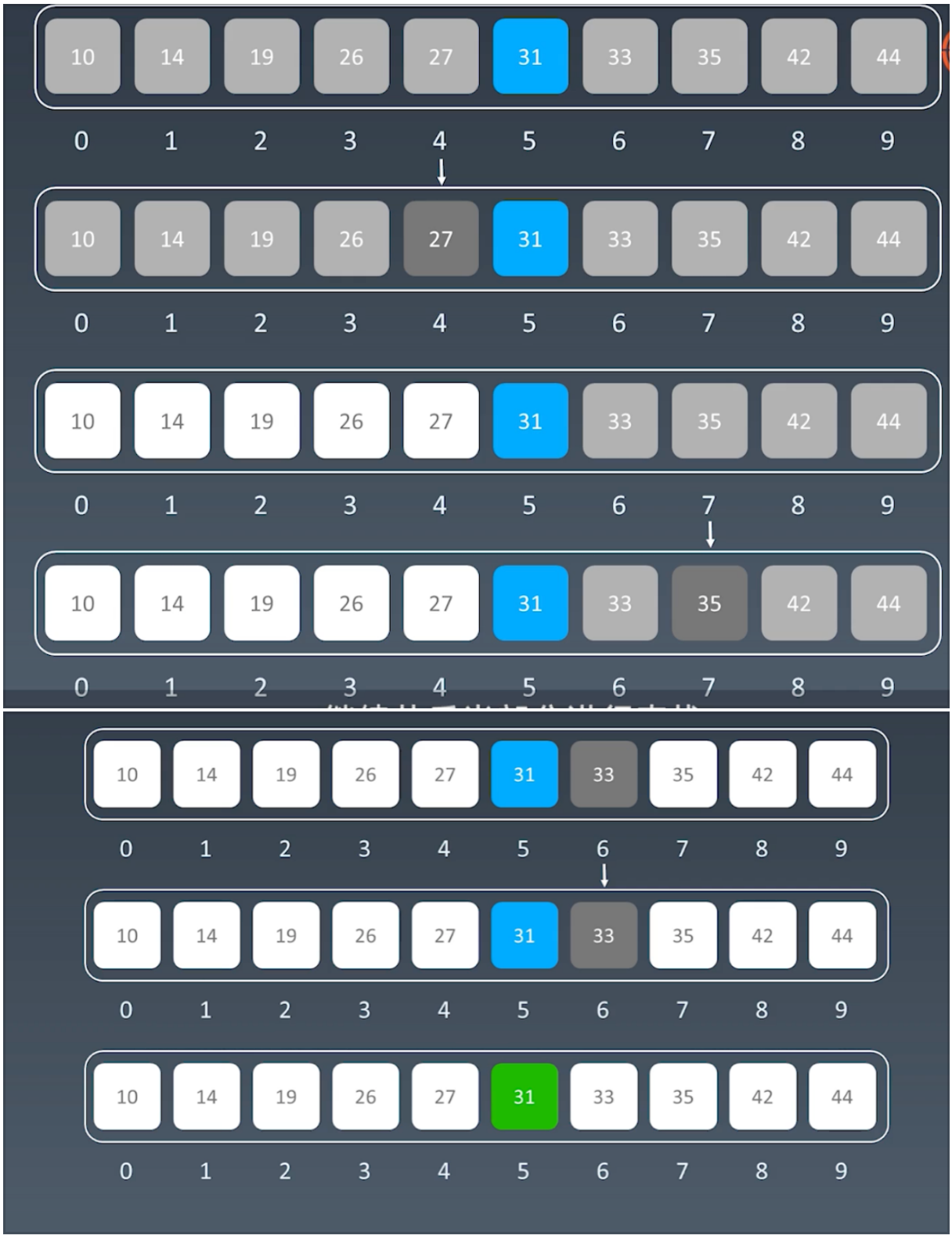
```
  }
  return -1;
}


public int binarySearch(int[] array, int target) {
    int left = 0, right = array.length - 1, mid;
    while (left <= right) {
        mid = (right - left) / 2 + left;
        if (array[mid] == target) {
            return mid;
        } else if (array[mid] > target) {
          right = mid - 1;
        } else {
            left = mid + 1;
        }
    }
    return -1;
}
```

示例

在下列递增数组里 查找 : 31

[10, 14, 19, 26, 27, 31, 33, 35, 42, 44]



#Algorithm/Part II : Theory/Algorithm#