



## 中山大学计算机学院

### 人工智能

### 本科生实验报告

(2024 学年春季学期)

课程名称: Artificial Intelligence

教学班级	202320346	专业 (方向)	计算机科学与技术
学号	22320107	姓名	饶鉴晟

#### 一、实验题目

给定一个  $n$  个元素有序的 (升序) 整型数组 `nums` 和一个目标值 `target` , 写一个函数 `BinarySearch` 搜索 `nums` 中的 `target` , 如果目标值存在返回下标, 否则返回 `-1`

#### 二、实验内容

##### 1. 算法原理

这段代码实现了二分查找算法, 用于在已排序的数组中查找特定元素。算法首先将左边界  $l$  设为数组第一个元素的索引, 右边界  $r$  设为数组最后一个元素的索引。然后, 通过一个循环, 每次将中间索引  $mid$  计算为  $(l + r + 1) // 2$ , 向上取整以避免死循环。接着, 判断中间元素与目标值的大小关系, 如果中间元素小于等于目标值, 则将左边界  $l$  移动到中间位置, 否则将右边界  $r$  移动到中间位置的前一个位置。循环直到左边界大于等于右边界时结束。最后, 检查右边界所指元素是否等于目标值, 如果是则返回其索引, 否则返回 `-1` 表示未找到。

核心公式:

$$mid = \frac{l + r + 1}{2}$$

整个算法的思路可以描述为不断缩小查找范围, 通过比较中间元素和目标值的大小关系, 动态调整左右边界, 最终找到目标元素或者确定不存在于数组中。



## 2. 伪代码

---

**Algorithm 1:** Binary Search Algorithm

---

**Data:** Sorted integer array  $nums$ , target integer  $target$

**Result:** Index of  $target$  in  $nums$  or -1 if  $target$  is not found

```
1  $l \leftarrow 0$ ;  
2  $r \leftarrow \text{lengthof}nums - 1$ ;  
3 while  $l < r$  do  
4    $mid \leftarrow \lfloor \frac{l+r+1}{2} \rfloor$  ;           // Compute middle index, rounding up  
5   if  $nums[mid] \leq target$  then  
6      $l \leftarrow mid$  ;               // Move left boundary to middle  
7   else  
8      $r \leftarrow mid - 1$  ;           // Move right boundary before middle  
9   end  
10 end  
11 if  $nums[r] == target$  then  
12   return  $r$  ;                     // Target found, return index  
13 else  
14   return  $-1$  ;                     // Target not found  
15 end
```

---

## 3. 关键代码展示（带注释）

```
def BinarySearch(nums, target):  
    """  
    使用二分查找在已排序的数组中查找特定元素  
    :param nums: list[int] 已排序的整数数组  
    :param target: int 需要查找的目标元素  
    :return: int 目标元素在数组中的索引，如果未找到则返回-1  
    """  
    l = 0  
    r = len(nums) - 1  
  
    # 当左边界小于右边界时，循环继续  
    while l < r:  
        mid = (l + r + 1) // 2 # 计算中间索引。使用(l + r + 1)确保mid向上取整，避免死循环  
  
        # 如果中间元素小于等于目标值，则将左边界移动到中间位置  
        if nums[mid] <= target:  
            l = mid  
        # 如果中间元素大于目标值，则将右边界移动到中间位置的前一个位置  
        else:  
            r = mid - 1  
  
    # 循环结束后，检查右边界所指元素是否为目标值  
    if nums[r] == target:  
        return r # 如果是，返回其索引  
    else:  
        return -1 # 如果不是，返回-1表示未找到
```

### 三、实验结果及分析

#### 1. 实验结果展示示例（可图可表可文字，尽量可视化）



```
27
28 # 以下为测试程序
29 array = [1, 3, 8, 12, 15, 17, 22]
30 target_1 = 12 # 测试target在array中间的情况
31 target_2 = 1 # 测试target在array左边界的情况
32 target_3 = 22 # 测试target在array右边界的情况
33 target_4 = 0 # 测试target超出array左边界的情况
34 target_5 = 25 # 测试target超出array右边界的情况
35 index_1 = BinarySearch(array, target_1)
36 index_2 = BinarySearch(array, target_2)
37 index_3 = BinarySearch(array, target_3)
38 index_4 = BinarySearch(array, target_4)
39 index_5 = BinarySearch(array, target_5)
40 print("index of target_1 is %d\n"
41       "index of target_2 is %d\n"
42       "index of target_3 is %d\n"
43       "index of target_4 is %d\n"
44       "index of target_5 is %d\n"
45       "% (index_1, index_2, index_3, index_4, index_5))")
```

测试程序输入样例



```
@JasonW41k3r ->/workspaces/AI_2024 (main) $ /home/codespace/.python/current/bin/python
index of target_1 is 3
index of target_2 is 0
index of target_3 is 6
index of target_4 is -1
index of target_5 is -1

@JasonW41k3r ->/workspaces/AI_2024 (main) $
```

测试程序输出样例

如图所示，程序正确输出了查找数的下标，且当查找数在数组当中不存在的时候，程序输出“-1”表示该数不存在。

### 四、参考资料

[1] flywh. (2021) 科研基础 3-伪代码规范. erisM's Blog. 检索于 2024 年 3 月 10 日, <https://flywh.github.io/2021/pseudocode-specification/>.