Binary Search in R

Rahul Goswami

2022-01-18

Binary Search is a searching algorithm that is used to find an element in a sorted array. The basic idea of binary search is to search for an element in a sorted array by repeatedly dividing the search interval in half.
The average case time complexity of binary search is O(log n)
The worst case time complexity of binary search is O(n)
The best case time complexity of binary search is O(log n) (if the array is already sorted)
The space complexity of binary search is O(1) (in-place)

Algorithm:

1. Find the middle element of the array
2. If the middle element is the element to be searched, return the index of the middle element
3. If the middle element is greater than the element to be searched, search the first half of the array
4. If the middle element is less than the element to be searched, search the second half of the array
   @param vec Vector to be searched
   @param element Element to be searched
   @return Index of the element

```
binary.search <- function(vec, element){
    # Find the middle element of the array
    mid <- length(vec)/2
    # If the middle element is the element to be searched, return the index of the middle element
    if(vec[mid] == element){
        return(mid)
    }
    # If the middle element is greater than the element to be searched, search the first half of the ar
    if(vec[mid] > element){
        return(binary.search(vec[1:mid], element))
    }
    # If the middle element is less than the element to be searched, search the second half of the arra
    if(vec[mid] < element){
        return(mid + binary.search(vec[(mid+1):length(vec)], element))
    }
}
```

Example

```
sorted_vec <- c(0,1,2,3,4,5,6,7,8,9)
binary.search(sorted_vec, 5)
```

```
## [1] 6
```