CSE 321 - Homework 4

Due date: 25/12/2022, 23:59

Çağla Şahin

25 December 2022

2. Design a decrease and conquer algorithm that finds the median of an unsorted array.

Design of algorithm as follows step by step:

1. Divide the array into two halves.
2. Find the median of each half of the array using the algorithm.
3. If the medians of the two halves are equal, return that median as the median of the whole array.
4. If the median of the first half is greater than the median of the second half, recursively apply the algorithm to the second half of the array to find the median.
5. If the median of the first half is less than the median of the second half, recursively apply the algorithm to the first half of the array to find the median.

This algorithm works by repeatedly dividing the array in half and finding the median of each half until the medians of the two halves are equal. The time complexity of this algorithm is O(log n), where n is the number of elements in the array, making it an efficient way to find the median of an unsorted array.

4. Compare the time complexities of ternary search and binary search. Explain how the divisor affects the complexity of the search algorithm. Assuming the array has n elements, what does the time complexity of the algorithm become if we divide it into n parts at the beginning?

In terms of time complexity, binary search has a best-case time complexity of O(1) and an average and worst-case time complexity of O(log n), where n is the number of elements in the data being searched. Ternary search has a best-case time complexity of O(1) and an average and worst-

case time complexity of O(log3 n). To sum, in the average and worst-case scenarios, ternary search is less efficient than binary search, as it takes slightly longer to search for the target element.

In binary search, the divisor is 2, as the search space is divided into two parts at each step. In ternary search, the divisor is 3, as the search space is divided into three parts at each step. The divisor affects the time complexity of the search algorithm because **the larger the divisor, the more steps are required to search the data** and the longer the search will take.

If we divide an array with n elements into n parts at the beginning, the time complexity of the algorithm will become O(n). This is because the algorithm will have to search through all n elements in the array to find the target element, resulting in a time complexity of O(n).

5. Solution

5.1: What is the best-case scenario of interpolation search? What is the time complexity of it?

Interpolation search is a search algorithm that is used to locate a specific element within a sorted list of data. In the best-case scenario, the algorithm will be able to find the target element in a single step, resulting in a time complexity of O(1). However, in the worst-case scenario, the time complexity of the algorithm will be O(n), where n is the number of elements in the data being searched. This can occur if the target element is not present in the data or if the data is not uniformly distributed, which can make it difficult for the algorithm to accurately calculate the position where the target element is likely to be located.

5.2: What is the difference between interpolation search and binary search in terms of the manner of work and the time complexity?

Interpolation search and binary search are two algorithms that can be used to **locate a specific element within a sorted list of data.** While both algorithms are effective at finding elements in sorted data, they differ in the way they approach the search and in their time complexity.

Interpolation search works by *first estimating the position of the target element based on the values of the elements around it, and then searching for the element at that position*. This approach can be more efficient than binary search in certain scenarios, particularly when the data is uniformly distributed and the target element is likely to be located near the middle of the data. However, interpolation search can be affected by the distribution of the data and may have a higher time complexity in the worst-case scenario, which is O(n) where n is the number of elements in the data being searched.

Binary search, on the other hand, uses a divide and conquer approach, repeatedly dividing the search space in half until the target element is found. This approach is generally considered to be more reliable and has a time complexity of O(log n) in the best-case scenario. In the worst-case scenario, binary search also has a time complexity of O(n).