DAA Assignment 3

Vidushi Pathak (IIT2019027)
Shubham Netam (IIT2019028)
Puneet Singhwaiya (IIT2019029)
Group-9
B.Tech. 4th semester, Department of Information Technology
Indian Institute Of Information Technology, Allahabad

Context

- Problem statement
- Introduction
- Representation of Quaternary Search
- Algorithm
- Pseudo Code
- Apriori Analysis
- Aposteriori Analysis
- Conclusion
- References

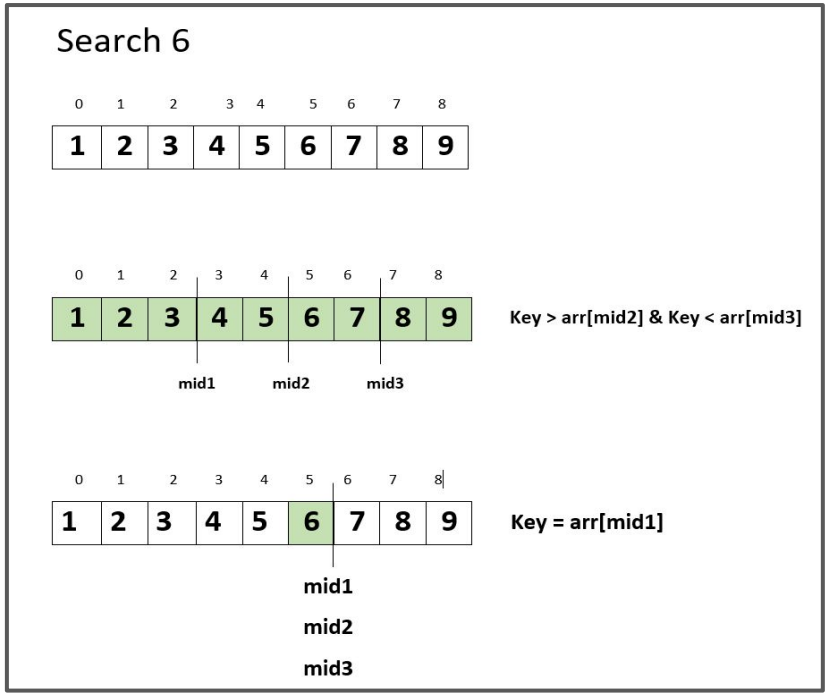
Problem Statement

Design an algorithm to implement Quaternary Search.

Introduction

Quaternary search is like binary search, but divides the array into four parts instead of two. After evenly dividing the array, the three divisors are compared to the input value. If it matches the index is returned. If not then algorithm check for the value in sub array that contains the value. This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form.

Representation of quaternary search



Algorithm

- Our algorithm will first finding three mid-points
- Divide the array into 4 parts
-
- It will check if the element which we want to find is present at any of the mid-points or not.
- IF present in any of those mid-point, it will return the index.
- Else it will check in which subarray the element is present.
- Then move the mid-point accordingly, till element is found or Concluded not present.

Pseudo Code

(1). Iterative Method

```
    Procedure quaternarySearch
A ← sorted array
n ← size of array
x ← value to be searched

Set left = 0
Set right = n-1

while left <= right

   mid1 = left + (right-1)/4
   mid2 = mid1 + (right-1)/4
   mid3 = mid2 + (right-1)/4

   if arr[mid1] is equal to x
         return mid1

   if arr[mid2] is equal to x
         return mid2
```

```
    if arr[mid3] is equal to x
            return mid3

    if arr[mid1] >x
            right = mid1-1

    else if arr[mid2] >x
            left = mid1+1
            right = mid2-1

    else if arr[mid3] <x
            left = mid3+1

    else if arr[mid2] <x and arr[mid3] >x
            left = mid2+1
            right = mid3-1
end while
return -1
end procedure
```

(2).Recursive Method

```
Procedure quaternarySearch
A ← sorted array
n ← size of array
x ← value to be searched

Set left = 0
Set right = n-1

if left <= right

    mid1 = left + (right-1)/4
    mid2 = mid1 + (right-1)/4
    mid3 = mid2 + (right-1)/4

    if arr[mid1] is equal to x
            return mid1

    if arr[mid2] is equal to x
            return mid2

    if arr[mid3] is equal to x
            return mid3
```

```
    if arr[mid1] >x
            return quaternarySearch(arr,left,mid1-1,x)

    else if arr[mid2] >x
            return quaternarySearch(arr,mid1+1,mid2-1,x)

    else if arr[mid3] <x
            return quaternarySearch(arr,mid3+1,right,x)

    else if arr[mid2] <x and arr[mid3] >x
            return quaternarySearch(arr,mid2+1,mid3-1,x)

return -1
end procedure
```

Complexity Analysis

Time Complexity: In this algorithm at each iteration the array is divided by 4. Suppose length of array is n and after K iteration the length of array becomes 1.Then we get,

$n/4^k = 1$

$n = 4^k$

$\log(n) = \log(4^k)$

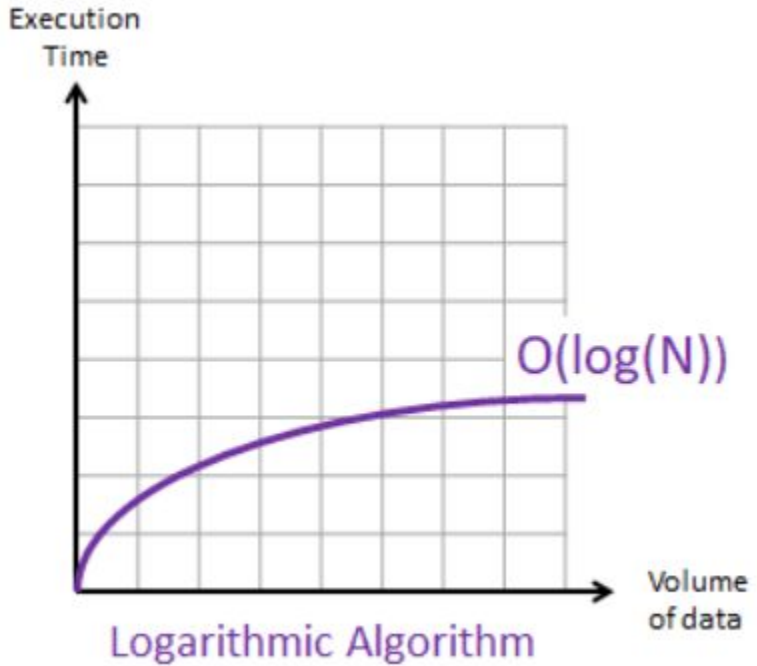
$\log(n) = k\log(4)$

$k=\log(n)$

Therefore, Time complexity is O(log(n))

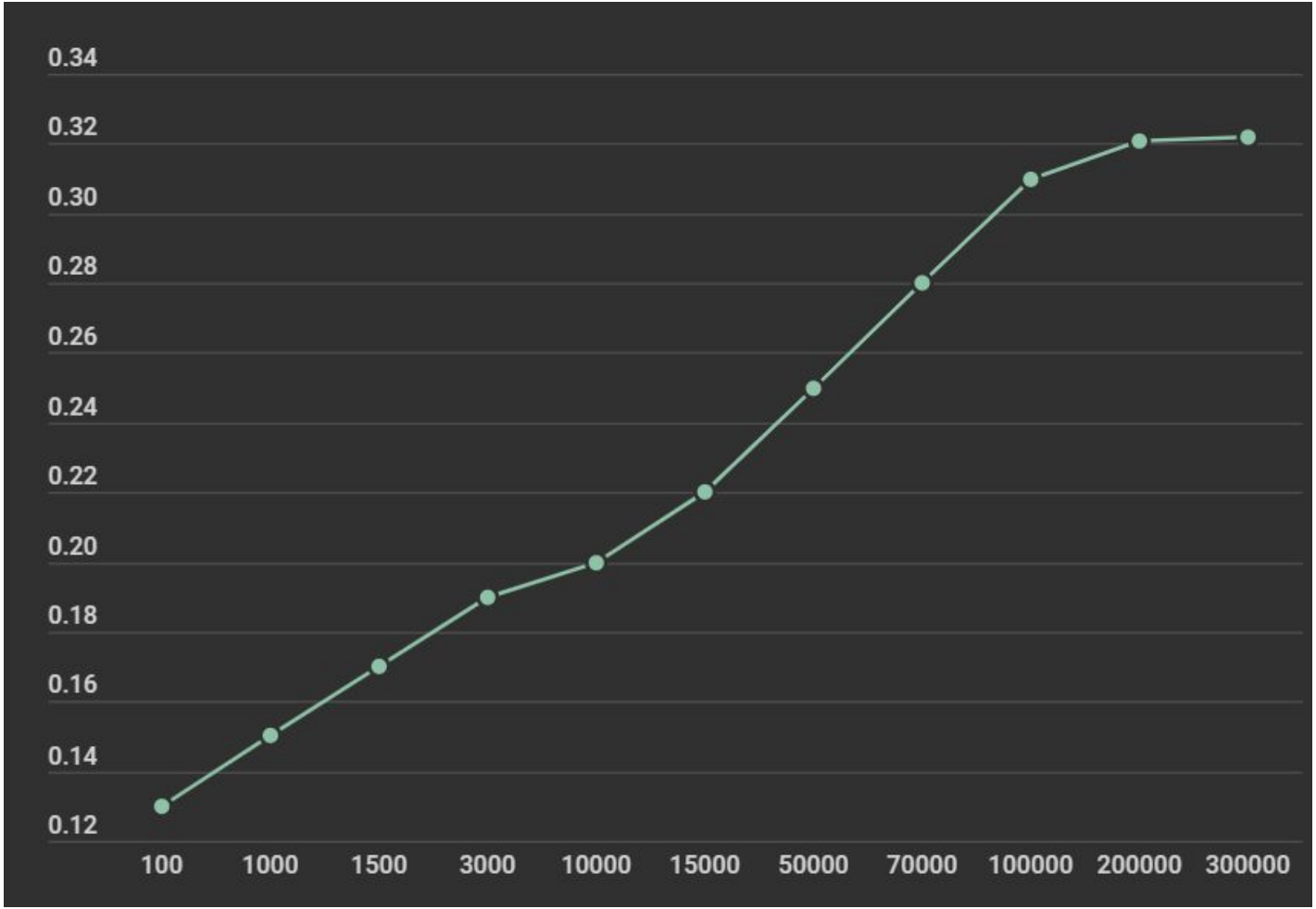
Space Complexity: In the iterative method, the space complexity would be O(1). While in the recursive method, the space complexity would be O(log(n)).

Aporiori Analysis



Aposteriori Analysis

Size of Array	Time
100	0.13
1000	0.15
1500	0.17
3000	0.19
10000	0.2
15000	0.22
50000	0.25
70000	0.28
100000	0.31
200000	0.321
300000	0.322


0.34
0.32
0.30
0.28
0.26
0.24
0.22
0.20
0.18
0.16
0.14
0.12
100
1000
1500
3000
10000
15000
50000
70000
100000
200000
300000

Conclusion

Through this assignment we concluded that although Quaternary can seem faster than binary search, but is not because it do more comparisons per step.

References

1. https://stackoverflow.com/questions/39845641/quaternary-search-algorithm
2. https://github.com/estensen/quaternary-search