Vivekanand Education Society's
Institute of Technology
Approved by AICTE & Affiliated to University of Mumbai

Artificial Intelligence and Data Science Department.
AOA / Even Sem 2021-22 / Assignment 1.

YASH SARANG.
47 / D6AD.
ASSIGNMENT - 1.

Aim: Devise a "Binary search" algorithm that splits the set not into two sets of (almost) equal sizes but into two sets, one of which is twice the size of the other. Compare it with the 'Binary Search' algorithm.

Theory:

BINARY SEARCH

Binary Search is a searching algorithm used in a sorted array by repeatedly dividing the search interval in half.

The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Log n).

The basic steps to perform Binary Search are:

- Begin with an interval covering the whole array.
- If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half.
- Otherwise, narrow it to the upper half.

- Repeatedly check until the value is found or the interval is empty.

Time Complexity: O(Log n)

We have been asked to devise a “Binary search” algorithm that splits the set not into two sets of (almost) equal sizes but into two sets, one of which is twice the size of the other.

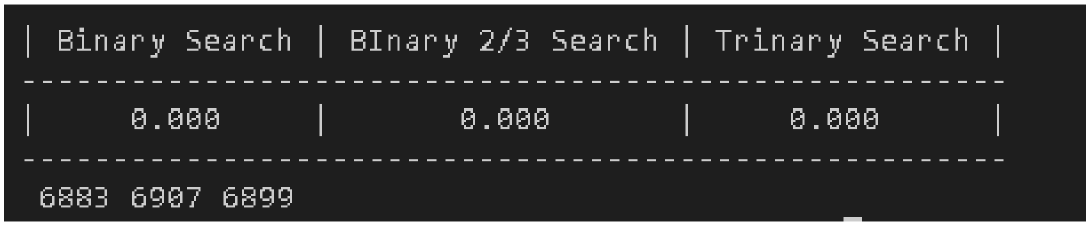
So basically it will divide the data into 2 parts, out of which one would be the first 2/3rd of the data and the rest as the second part. We will compare the point of separation with our desired value to determine whether the required value is in the first part or second part and similarly we’ll recursively follow this approach until we find our desired data. Implemented in ‘*Binary2_3 function’* in the program attached.

I personally thought of another variation, in which we will divide the data in 3 equal parts, then we will compare the required data with the two points of separation and according to the three situations,
1. data < sep1, function recursively on first part
2. sep1<data<sep2, function recursively on second part
3. sep2<data , function recursively on third part
I call it the *‘Trinary function’* which will be implemented in the program attached.

CODE:

Code in the Assignment1.c file attached along with this doc.

OUTPUT:



```
| Binary Search | BInary 2/3 Search | Trinary Search |
------------------------------------------------------
|     0.000     |       0.000       |     0.000      |
------------------------------------------------------
 6883 6907 6899
```

CONCLUSION:

By performing this experiment, I can conclude that Binary Search is extremely fast in most cases and even in its worst case, due to its ability of slicing the data into half after every iteration.
The complexity ($\log_2 n$) is extremely small as compared to n, for big data.

In our devised algorithm, the worst case complexity was again (log n) but the base was 3/2, due to which it had to undergo more iterations during the worst case as compared to the Binary Search algorithm.
Whereas in the Trinary(Ternary) Search Algorithm, the worst case complexity was ($\log_3 n$), and hence more efficient on paper than the Binary Search algorithm.

Overall all the Search algorithms similar to Binary Search were so fast that the modern computers required time lower than 10^-4, and hence I was unable to record the actual time difference between them during execution.
