DSAD Assignment NO-02
Shubham Laxman Hippargi
D24CS3014

Q. Application of Binary Search

01. Algorithm Writing :-

```
Function binary_search (ISBN, target):
    left=0
    right= length(ISBN)-1

    while left<=right:
          mid= left +(right-left) /2
          if ISBN[mid]== target:
                 return mid
          else if ISBN [mid] < target:
                 left = mid+1
          else
                 right= mid-1

    return -1
```

02. Complexity Analysis

A. Time Complexity of Binary Search = O(logn)
   n= number of elements in the array

B. In ideal case, the target element is found in the first comparison itself i.e. the first middle element itself is the element to be searched. Since in best case we require only one comparison,
Time Complexity of Binary in Best case = O(1)

In worst case i.e. either the element we are searching is not present or at extremes. The algorithm divides the array into 2 halfs after every comparison until the array contains only one element. Since the algorithm divides the array by 2 till we compare it with single element, the number of comparisons = $\log_2 n$

Hence Worst Case Time Complexity = $O(\log_2 n)$

C. In large scale library due to huge number of books we need to locate quickly by its unique ISBN number. Implementing linear search would be inefficient because in linear search we have to check every book until we find the target for the worst case. If there are n books, we need n comparisons. Hence it is inefficient. Binary search technique on the other hand would require less number of comparisons since after every comparison the search space gets halved. The halving process continues until we find the book Hence worst case time complexity Binary Search = $O(\log n)$ $\log n$ comparisons are required.

D. Space Complexity of Binary Search for iterative approach The iterative approach does not use any extra space. It uses only some space for integer variables like left, right, mid. Since these use constant space the space complexity = $O(1)$

03· Mathematical Expression:-

- We start with an array of n elements. The target is first compared with middle element of the array. If the target is not found after first comparison, the array is divided into half and either of one is selected for further comparison based on 'whether the target is small or greater than middle element. Now we have $\frac{n}{2}$ elements to search

Again the target is compared with the middle element of array that have $\frac{n}{2}$ elements.

If the target is not found even after second comparison, we again divide the array by 2. Hence now the total elements $= \frac{n}{4}$

If we perform this division until we get a single element, let consider the total number of comparisons to compare it with array of 1 element $= K$

Hence, $\frac{n}{2^K} = 1$

$$\Rightarrow n = 2^K$$

take $\log_2$ on both sides

$$\log_2 n = \log_2 2^K$$

$$\Rightarrow \log_2 n = K \log_2 2$$

$$\Rightarrow K = \log_2 n$$

04. Application:-

Dictionary is an application where binary search is used. The dictionary is sorted alphabatically in order A-Z. We don't need to start from the first page and compare every word. After every comparison we narrow down the search space. We continue this process until the target word is found. It requires less comparisons compared to linear search technique.

Binary search technique help make searches faster and efficient.