Binary Search

Binary Search is a searching algorithm for finding an element's position in a sorted array.

In this approach, the element is always searched in the middle of a portion of an array.

Binary search can be implemented only on a sorted list of items. If the elements are not sorted already, we need to sort them first.

Algo:-

```
do until the pointers low and high meet each other.
    mid = (low + high)/2
    if (x == arr[mid])
        return
```

```
mid
   else if (x > arr[mid])   //x is on the right side
      low = mid + 1
   else                      // x is on the left side
    high = mid - 1
```

code example:-

```
def binarySearch(array, key, low, high):

   while low <= high:
      mid = low + (high - low)  //2

      if array[mid] == key:
         return mid
      elif array[mid] < key:
         low = mid + 1
```

```
else:
        high = mid - 1
  return -1


array = [1,2,3,4,5,6,7,8,9,10]

key = 9

result = binarySearch(array, key, 0, len(array)-1)

if result =! -1:
  print("Element is present at index: ", result)
else:
  print("Not Found")
```

Binary Search

Complexity

Time Complexities

Best case complexity: $O(1)$
Average case complexity: $O(\log n)$
Worst-case complexity: $O(\log n)$

Space Complexity

The space complexity of the binary search is $O(1)$.

Binary Search Applications

In libraries of Java, Net, C++ STL
While debugging, the binary search is used to pinpoint the place where the error happens.