INDEX

Name of the Lab : Design And Analysis of Algorithms (DAA).

S.No.	Date	Title of the Experiment	Page No.	Marks	Sign.
01	19/12/24	Towers of Hanoi Problem	1-6		
02	26/12/24	Binary Search - DNC	7-11		

Aim :- Implement Binary search algorithm to find the existence of a particular element in the list or array. Repeat the experiment for difficult values of n (number of elements in the given list or array). (DIVIDE AND CONQUER).

Problem statement :- Given an array of n elements A[n] and an aribtrary value x. Binary search problem is defined as searching the index of the element x in the array, if not present in the array, it returns -1.

Iterative Algorithm :-

```
Algorithm Iterative_binary-Search(A[n], x)
{
    low=0; high=n-1;
    while (low <= high)
    {
        mid= (low+high)/2;
```

20

Signature of the Faculty

Date :

```
    if (x == A[mid])
        return mid;
    else if (x < A[mid])
        high = mid + 1;
    else
        low = mid + 1;
    }
        return -1;
}
```

Iterative code:-

```
import random
def binary_search (arr, target):
    low, high = 0, len(arr - 1) - 1
    while low >= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid  # Target found return index.
```

```
    elif arr[mid] < target:
        low = mid - 1
    else:
        high = mid - 1
    return -1  # Target not found.

# Experiment with different values of n (number of elements in the list).
n_values = sorted([6, 9, 10, 15])  # Different list sizes to test.
target_value = 25  # Predefined target value to search for.

for n in n_values:
    # Create a sorted list of random values of size n.
    arr = sorted(random.sample(range(1, 50), n))

# Random unique numbers within a Range.
```

```
print(f"\nList of size {n}: {arr}")
print(f"Searching for {target_value} in the
      list...")

# Perform a binary search and get the position.
position = binary_search(arr, target_value)
if position != -1:
    print(f"{target_value} found at index
          {position}.")
else:
    print(f"{target_value} Not found in the list.")
```

Output:-

List of size 6: {15, 25, 26, 27, 35, 37}

Searching for 25 in the list....

25 ~~Not~~ found in the list at the index 2.

List of size 9: [1, 3, 5, 11, 15, 17, 24, 36, 41]

Searching for 25 in the list...

25 Not found in the list..

List of size 10: [1, 5, 7, 8, 13, 29, 31, 36, 40, 48]

Searching for 25 in the list...

25 Not found in the list..

List of size 15: [1, 4, 8, 13, 15, 16, 24, 25, 27, 31, 32, 34, 38, 39, 40].

Searching for 25 in the list...

25 found in the list at the index 7.

Recursive Algorithm:-

Algorithm recursive-binary-Search (low, high)

```
{
    if (low > high) {
        return -1;
    }
```

```
    mid = (low + high) /2;
  if (x == A[mid]) {
      return mid;
  }
 else if (x < A[mid]) {
   return binarySearch (low, mid-1);
 }
 else {
   return binarySearch (mid+1, high);
   }
 }
```

Code :- (Recursive)

```
import random
  def rec_binary_search (arr, low, high, x):
    if low > high
      return -1; # Target not found
    mid = (low + high) // 2
```

```
if arr[mid] == x:
    return mid # Return the index of Target
elif arr[mid] > x:
    return rec_binary_Search(arr, low, mid-1, x)
else:
    return rec_binary_Search(arr, mid+1, high, x)

# Experiment with different values of n (number
Of elements in the list)
n_values = sorted([6,9,10,15]) # Different list
sizes to test
target_value = 33 # Predefined target value
to search for.
for n in n_values:
    # create a sorted list of random values
    of size n.
```

20

Signature of the Faculty

Date :

```
arr = sorted(random.sample(range(1,50), n))

# Random unique numbers within a range.

print(f"\nList of size {n}: {arr}")

print(f"Searching for {target_value} in the
      list...")

# Perform a binary search and get the position.

position = rec_binary_search(arr, 0, len(arr)-1,
           target_value).

if position != -1:
    print(f"{target_value} found at index
          {position}.")

else:
    print(f"{target_value} Not found in the
          list.")
```

Output:-

List of size 6: [12, 16, 23, 26, 32, 35]

Searching for 33 in the list...

33 Not found in the list.

List of size 9: [10, 15, 20, 25, 26, 34, 42, 43, 47]

Searching for 33 in the list...

33 Not found in the list.

List of size 10: [9, 11, 18, 22, 23, 24, 27, 28, 37, 45]

Searching for 33 in the list...

33 Not found in the list.

List of size 15: [3, 11, 12, 13, 14, 18, 19, 20, 21, 24, 29, 35, 41, 44, 45]

Searching for 33 in the list...

33 Not found in the list.