International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 9 Issue 5 ǁ 2021 ǁ PP. 24-25

Large Sorting

[1.] Binu.C.T
Freelance software tester

Abstract:
Sorting algorithms have a lot of importance in Computer Industry. Sorting algorithms are related to Mathematics and Computer Science. Binary search is a type of searching algorithm well known in the Computer field. The proposed algorithm's code is also similar to Binary search. The proposed algorithm is used only for large count sorting must commonly count greater than 50. Maximum limiters 1000. The recurrence relation (recursive) for the proposed algorithm is $T(n)=T(n/_2)+2$. This sort is known as large sort.
Keywords: Algorithms, Sorting, Searching

Date of Submission: 05-05-2021 Date of acceptance: 18-05-2021

I. INTRODUCTION

The Binary Search and proposed algorithm are similar nature.

```
int iterativeBinarySearch(int array[], int start_index, int end_index, int element){
  while (start_index <= end_index){
    int middle = start_index + (end_index- start_index )/2;
    if (array[middle] == element)
      return middle;
    if (array[middle] < element)
      start_index = middle + 1;
    else
      end_index = middle - 1;
  }
  return -1;
}
int main(void){
  int array[] = {1, 4, 7, 9, 16, 56, 70};
  int n = 7;
  int element = 16;
  int found_index = iterativeBinarySearch(array, 0, n-1, element);
  if(found_index == -1 ) {
    printf("Element not found in the array ");
  }
  else {
    printf("Element found at index : %d",found_index);
  }
  return 0;
}
```

Algorithm for Binary Search

II. QUICK SORT

The Proposed Algorithm for larger count is faster than Quick Sort. The algorithm for the Quick Sort is given below

```
quickSort(array, leftmostIndex, rightmostIndex)
 if (leftmostIndex < rightmostIndex)
   pivotIndex <- partition(array,leftmostIndex, rightmostIndex)
   quickSort(array, leftmostIndex, pivotIndex - 1)
   quickSort(array, pivotIndex, rightmostIndex)

partition(array, leftmostIndex, rightmostIndex)
```

```
  set rightmostIndex as pivotIndex
  storeIndex <- leftmostIndex - 1
  for i <- leftmostIndex + 1 to rightmostIndex
  if element[i] < pivotElement
    swap element[i] and element[storeIndex]
    storeIndex++
  swap pivotElement and element[storeIndex+1]
return storeIndex + 1
```

Quick Sort

III. LARGE SORT ALGORITHM

1.	Get the array of A[i]	n	
2.	Swap odd and even position of array		
3.	A[i] = A[i+1]	1	
4.	For i= 0; i<n; i+1		n
5.	A[i+1] = A[i+2]	n	
6.	If A[mid] here two values, take least		
7.	$A[n/_2] = X$		1
8.	J = 0, $j<[n/_2]$ j++		
9.	If $[A(i) > A[n/_2]$	$n/_2$	
10.	Swap		
11.	If [A(i=j> $A(n/_2$ 1-j)		
12.	Swap	1	
13.	Print A[i]		n

Large Sort

$T(n) = n+1+n+n+1+ n/_{2+}1+n$
Large System's time complexity = Order of n *(O(n))*
Recurrence relation (recursive) for the Large Sorting is
$T(n) = T(n/_2) +2$

IV. CONCLUTION

The proposed Algorithm performs well in all the programming languages. Time complexity for the Algorithm is *(O(n))* and Recurrence relation (recursive) $T(n) = T(n/_2) +2$.

REFERENCES

[1]. www.programiz.com
[2]. Introduction to Algorithms by Thomas H. Cormen, Charles E. Leiserson, Ronald Rivest, Clifford Stein
[3]. Wikipedia

Mr. Binu.C.T is a post graduate in Computer Science and Engineering. He have 3 Years Experience in Academic field in different Engineering Colleges. He also have 2.3 Years experience in software Testing Domain.