RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Lab Report 3

Course Code: *CSE 2202*

Course Title: *Sessional Based on CSE 2201.*

Submitted By:

Name : Shanjid Hasan Nishat

Roll No : 1803172

Section : 'C'

Date of Submission: 12/03/ 2021

Submitted To:

Dr. Md. Ali Hossain

Associate Professor,

Dept. of Computer Science and Engineering.

Rajshahi University of Engineering & Technology.

Problem Statement: Find out the time and space complexity of the Merge Sort approach and compare it with the performance of Quick sort algorithm for (i) Best (ii) Average and (iii) Worst cases

Description:

Merge sort is an efficient, general-purpose, comparison-based sorting algorithm. Merge sort is a divide and ~~cong~~ conquer algorithm. It divides the input array into two halves, calls itself for the two halves and then merge the two sorted halves

Quick sort is a commonly used algorithm for sorting. It is faster than merge sort. Like merge sort, Quick sort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot.

Time complexity comparison of merge sort and quick sort:

	Mergesort	Quicksort
Best case:	$\Omega(n\log(n))$	$\Omega(n\log(n))$
Average case:	$\Theta(n\log(n))$	$\Theta(n\log(n))$
Worst case:	$O(n\log(n))$	$O(n\log(n))$

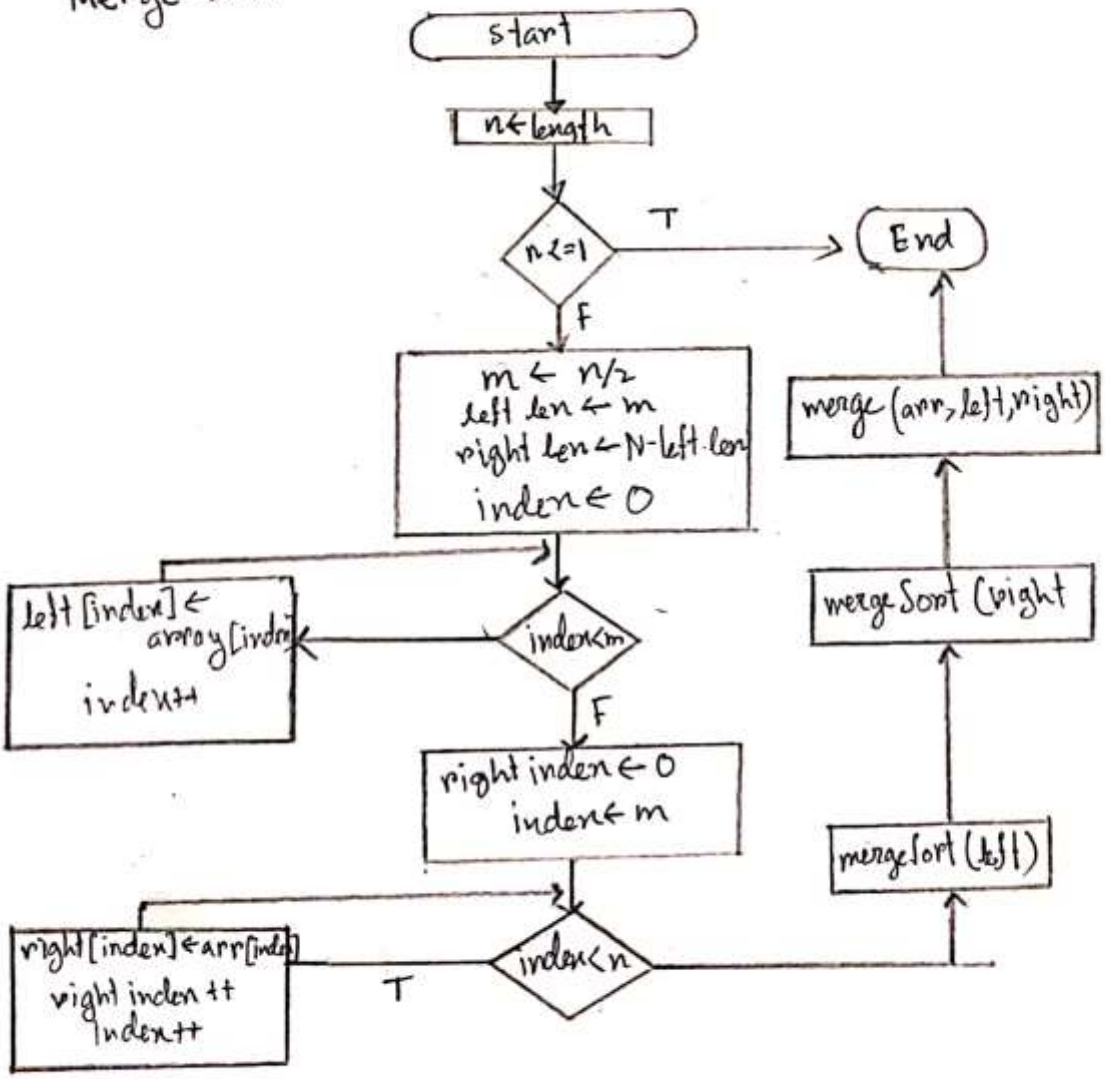
Algorithms:

```
MergeSort (A, p, r)
    if p>r
        return
    q = (p+r)/2
    mergeSort (A, p, q)
    mergeSort (A, q+1, r)
    merge (A, p, q, r)
```

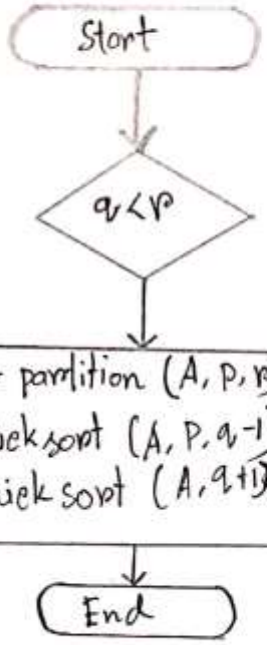
```
quickSort (left, right)
    if right-left <=0
        return
    else
        pivot = A[right]
        partition = partitionFunc (left, right, pivot)
        quickSort (left, partition-1)
        quickSort (partiton+1, right)
    end if
```

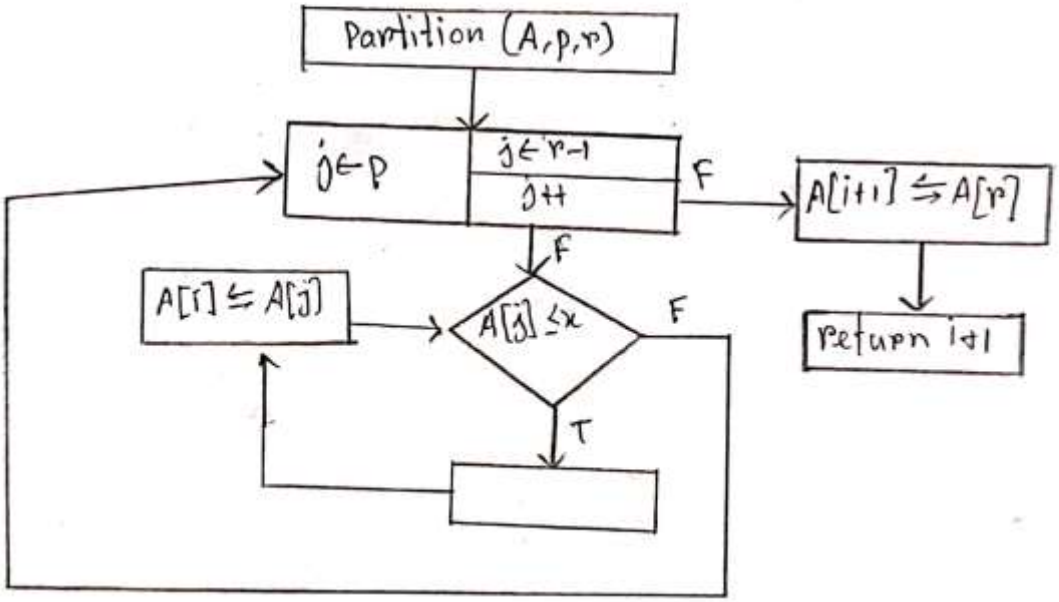
Flow chart:

Merge sort:

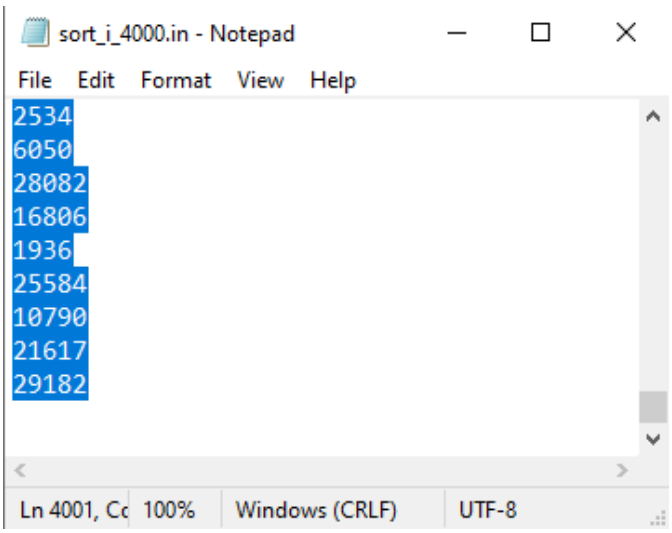


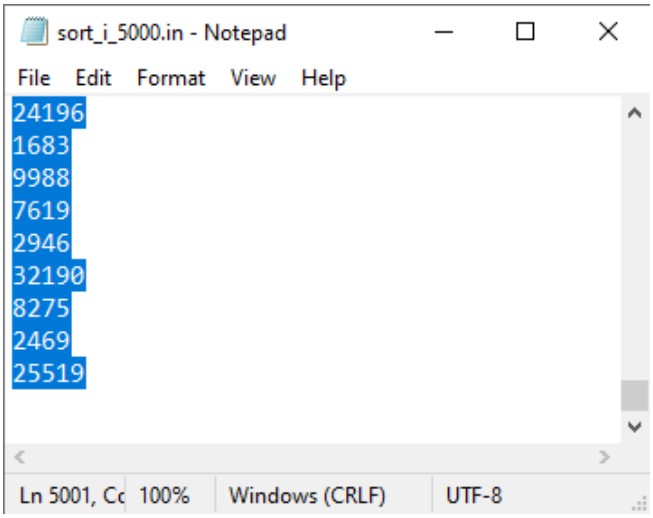
Quick sort:

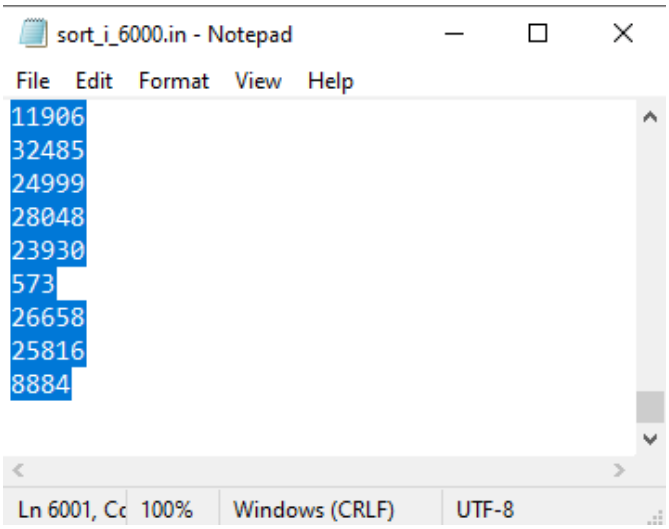

Start
q<r
q← partition (A, p, r)
Quicksort (A, p, q-1)
Quick sort (A, q+1), r)
End

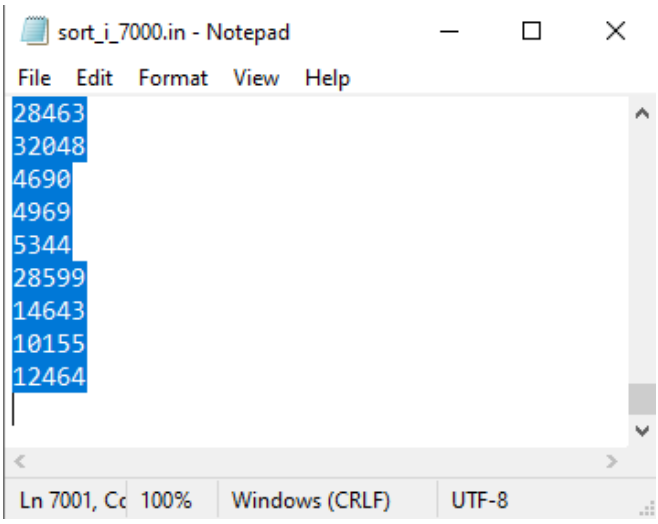

Partition (A,p,r)
j←p
j←r-1
j++
F
A[i+1] ⇆ A[r]
F
A[i] ⇆ A[j]
A[j] ≤x
F
Return i+1
T

Input:


sort_i_4000.in - Notepad
File Edit Format View Help
2534
6050
28082
16806
1936
25584
10790
21617
29182
Ln 4001, Cc 100% Windows (CRLF) UTF-8


sort_i_5000.in - Notepad
File Edit Format View Help
24196
1683
9988
7619
2946
32190
8275
2469
25519
Ln 5001, Cc 100% Windows (CRLF) UTF-8


sort_i_6000.in - Notepad
File Edit Format View Help
11906
32485
24999
28048
23930
573
26658
25816
8884
Ln 6001, Cc 100% Windows (CRLF) UTF-8

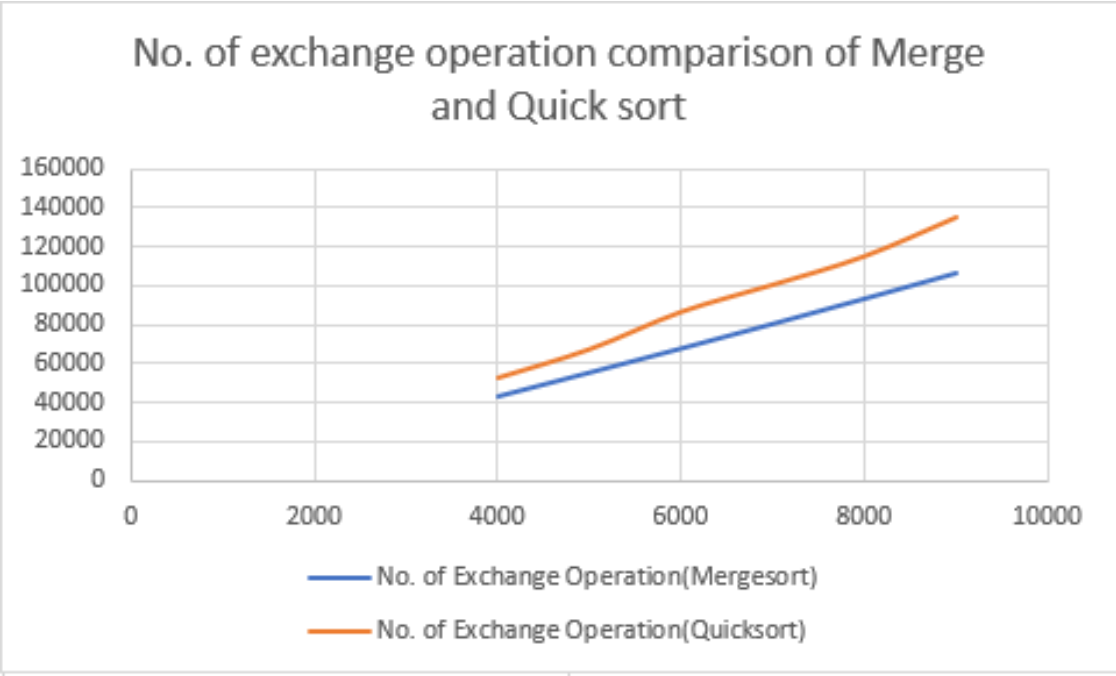

sort_i_7000.in - Notepad
File Edit Format View Help
28463
32048
4690
4969
5344
28599
14643
10155
12464
Ln 7001, Cc 100% Windows (CRLF) UTF-8

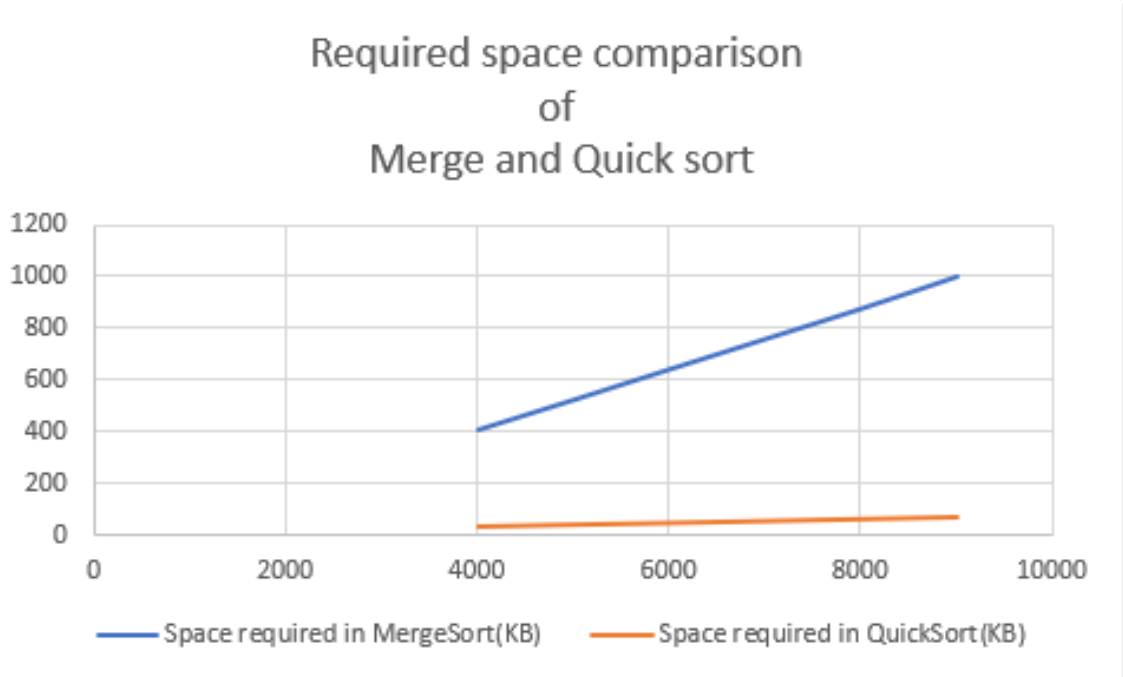
Output:

"F:\2-2\Study Materials\Sessional ...

```
Total number of elements : 4000
Merge Sort :
            Time required  : 0.998ms
            Space required : 405KB
      Number of comparison : 42869
      Sorted data saved in : Output_0.txt


Quick Sort :
            Time required  : 0.998ms
            Space required : 31KB
      Number of comparison : 52608
      Sorted data saved in : Output_1.txt



Total number of elements : 5000
Merge Sort :
            Time required  : 0.998ms
            Space required : 521KB
      Number of comparison : 55274
      Sorted data saved in : Output_2.txt


Quick Sort :
            Time required  : 0.998ms
            Space required : 39KB
      Number of comparison : 67365
      Sorted data saved in : Output_3.txt



Total number of elements : 6000
Merge Sort :
            Time required  : 0.998ms
            Space required : 639KB
      Number of comparison : 67827
      Sorted data saved in : Output_4.txt


Quick Sort :
            Time required  : 0.998ms
            Space required : 46KB
      Number of comparison : 86679
      Sorted data saved in : Output_5.txt



Total number of elements : 7000
Merge Sort :
            Time required  : 0.998ms
            Space required : 756KB
      Number of comparison : 80645
      Sorted data saved in : Output_6.txt


Quick Sort :
            Time required  : 0.998ms
            Space required : 54KB
      Number of comparison : 100668
      Sorted data saved in : Output_7.txt



Total number of elements : 8000
Merge Sort :
            Time required  : 0.998ms
            Space required : 873KB
      Number of comparison : 93734
      Sorted data saved in : Output_8.txt
```

Graph:


No. of exchange operation comparison of Merge and Quick sort
160000
140000
120000
100000
80000
60000
40000
20000
0
0
2000
4000
6000
8000
10000
No. of Exchange Operation(Mergesort)
No. of Exchange Operation(Quicksort)


Required space comparison of Merge and Quick sort
1200
1000
800
600
400
200
0
0
2000
4000
6000
8000
10000
Space required in MergeSort(KB)
Space required in QuickSort(KB)

Discussion and Conclusion:

In this problem we have seen the comparison of mergesort and quicksort. we can see that the required time for both merge sort and quick sort are always same. Because from theory we know that time complexity for both algorithms in average case are same. But in merge sort it require more space than quick sort. So, we can say that mergesort is less efficient than quick sort.