Name	Manish Shashikant Jadhav
UID	2023301005
Subject	Design and Analysis of Algorithms (DAA)
Experiment No.	2
Aim	Experiment based on divide and conquer approach.
Code	

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    int L[n1], R[n2];

    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
```

```
            {
                arr[k] = R[j];
                j++;
            }
            k++;
        }

        while (i < n1)
        {
            arr[k] = L[i];
            i++;
            k++;
        }

        while (j < n2)
        {
            arr[k] = R[j];
            j++;
            k++;
        }
}

void mergeSort(int arr[], int l, int r)
{
    if (l < r)
    {
        int m = l + (r - l) / 2;

        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}

void quickSort(int arr[], int low, int high)
{
```

```c
    if (low < high)
    {
        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int partition(int arr[], int low, int high)
{
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++)
    {
        if (arr[j] < pivot)
        {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }

    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void main()
{
    srand(time(NULL));
```

```c
    FILE *fileptr;
    FILE *fileptr1;
    int n = 100000;
    fileptr = fopen("inp.txt", "w");
    int arr[n];
    for (int j = 0; j < n; j++)
    {
        arr[j] = rand() % 100000 + 1;
        fprintf(fileptr, "%d\n", arr[j]);
    }
    fclose(fileptr);
    fileptr1 = fopen("time.csv", "w");
    printf("Block Size\tMerge Sort\tQuick Sort\n");
    fileptr = fopen("inp.txt", "r");
    for (int p = 99; p < n; p = p + 100)
    {
        int array[p + 1];
        int array1[p + 1];
        for (int j = 0; j < p; j++)
        {
            array[j] = arr[j];
            fscanf(fileptr, "%1d", &array1[j]);
        }
        clock_t begin = clock();
        mergeSort(array, 0, p);
        clock_t mid = clock();
        quickSort(array1, 0, p);
        clock_t end = clock();
        double time_spent = (double)(mid - begin);
        double time_spent1 = (double)(end - mid);
        printf("%d\t%lf\t%lf\n", p + 1, time_spent, time_spent1);
        fprintf(fileptr1, "%d,%lf,%lf\n", p + 1, time_spent,
time_spent1);
    }
    printf("\n");
}
```
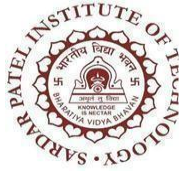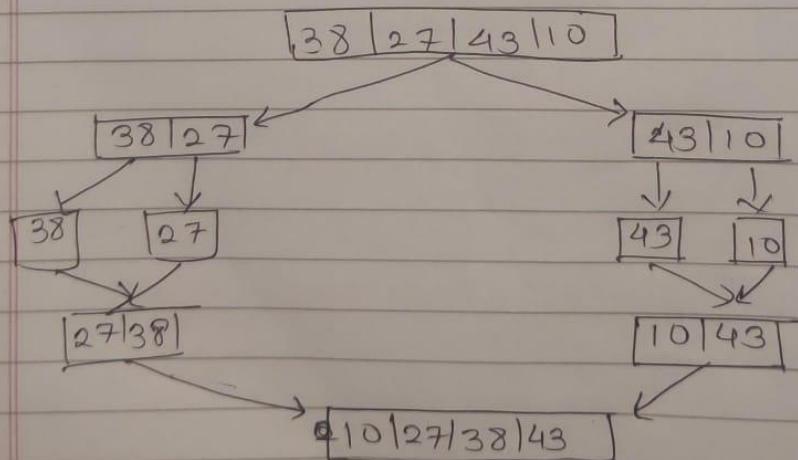
Graphs

merge sort

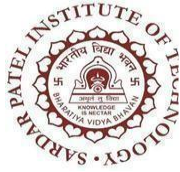quick sort
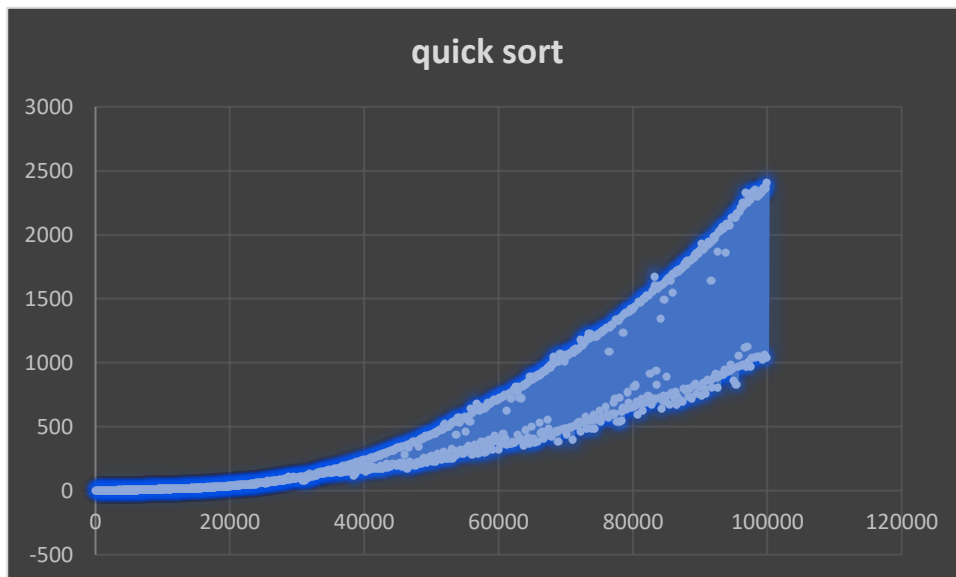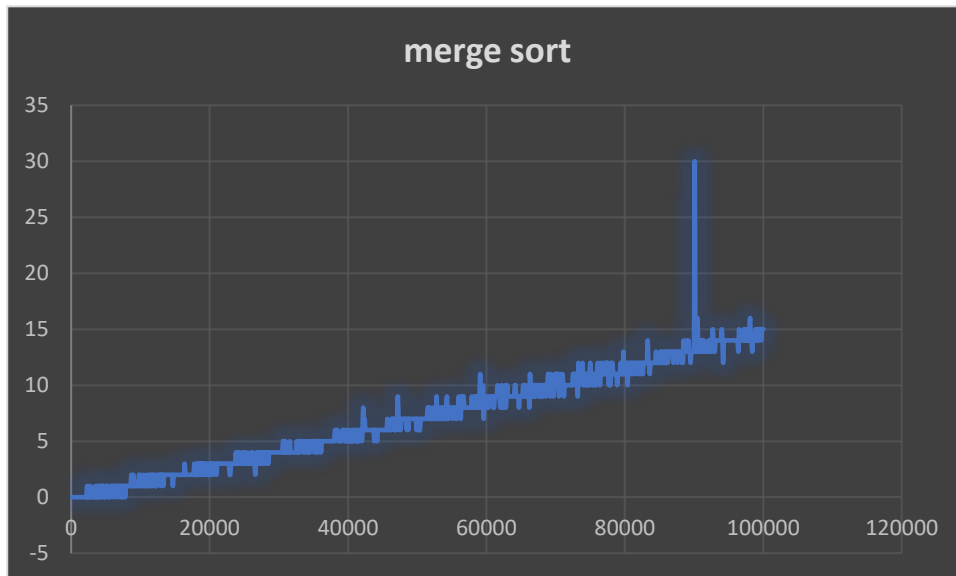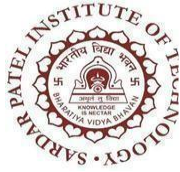
Pseudo Code & Example	

Experiment No. 2

* Merge Sort:-
. Pseudo code:- (A, p, q, r):
1. $n_1 \leftarrow q - p + 1$
2. $n_2 \leftarrow r - q$
3. create arrays $L[1 \cdots n+1]$ and $R[1 \cdots n_2+1]$
4. for $i+1$ to n_1
5.   do $L[i] \leftarrow A[p+i-1]$
6. for $j+1$ to n_2
7.   do $R[j] \leftarrow A[q+j]$
8. $L[n_1+1] \leftarrow \infty$
9. $R[n_2+1] \leftarrow \infty$
10. $i \leftarrow 1$
11. $j \leftarrow 1$
12. for $k \leftarrow p$ to r
13.   do if $L[i] \leq R[j]$
14.   then $A[k] \leftarrow L[i]$
15. $i \leftarrow i+1$
16.   else $A[k] \leftarrow R[j]$
17. $j \leftarrow j+1$ $(j = j+1)$

* Quicksort:-
* Pseudo Code:-

Partition $(A[p \ldots q])$

1. $x \leftarrow A[p]$
2. $i \leftarrow p$
3. for $j \leftarrow p+1$ to q
4.   do if $A[j] < x$
5. $\ldots\ldots$ $i \leftarrow i+1$
6.   exchange $A[i] \leftrightarrow A[j]$
7. $A[p] \leftrightarrow A[i]$
8. return (i).

* Quicksort $(A[p \ldots q])$

1. if $p = q$ then done.
2. else $r \leftarrow$ Partition $(A[p \ldots q])$
3. recursively call: Quicksort $(A[p \ldots r-1])$
                 Quicksort $(A[p \ldots q-1])$

- Quick Sort Example :-

  | 28 | 9 | 13 | 2 |

  Pivot = 28.

  ① Partition Around Pivot.

  | 9 | 13 | 2 | 28 |

  | 9 | 13 | 2 | | 28 |

  ② Pivot : 9   No action
     Partitioning   Need
  | 2 | 9 | 13 | | 28 |

  Merge Sorted Sub-arrays
  | 2 | 9 | 13 | 28 |

Conclusion	Hence, by completing this experiment I came to know about divide and conquer approach.