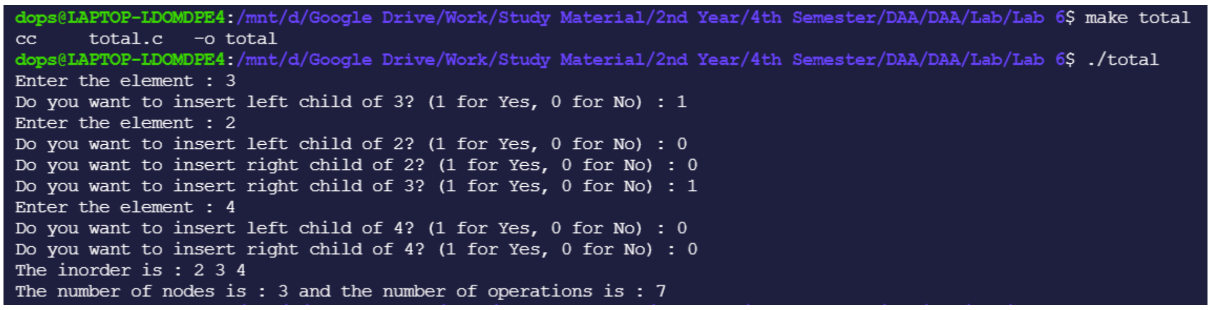
Lab 6 : Dipesh Singh - 190905520

Question 1 : Find total number of nodes in a binary tree and analyze its efficiency. Obtain the experimental result of order of growth and plot the result.

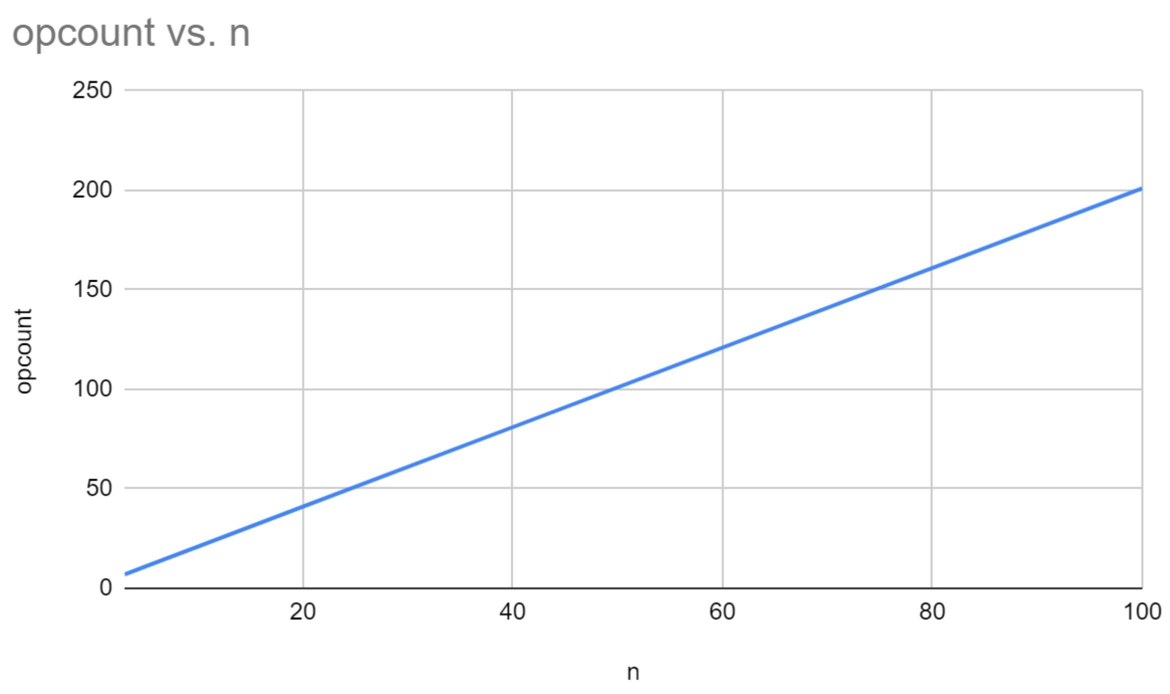
```c
#include <stdio.h>
#include <stdlib.h>

typedef struct node
{
    int val;
    struct node *left;
    struct node *right;
} * Node;

void inorder(Node n)
{
    if (n)
    {
        inorder(n->left);
        printf("%d ", n->val);
        inorder(n->right);
    }
}

Node insert()
{
    int val;
    int check;
    printf("Enter the element : ");
    scanf("%d", &val);
    Node n = (Node)malloc(sizeof(struct node));
    n->val = val;
    n->left = NULL;
    n->right = NULL;
    printf("Do you want to insert left child of %d? (1 for Yes, 0 for No)
 : ", val);
    scanf("%d", &check);
    if (check)
        n->left = insert();
    printf("Do you want to insert right child of %d? (1 for Yes, 0 for No
) : ", val);
    scanf("%d", &check);
    if (check)
        n->right = insert();
    return n;
}
```

```c
int cnt(Node head, int *opcount)
{
    (*opcount)++;
    if (head)
    {
        return 1 + cnt(head->right, opcount) + cnt(head->left, opcount);
    }
    return 0;
}

int main()
{
    Node head = insert();
    printf("The inorder is : ");
    inorder(head);
    int opcount = 0;
    int res = cnt(head, &opcount);
    printf("\nThe number of nodes is : %d and the number of operations is
 : %d\n", res, opcount);
    return 0;
}
```



```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 6$ make total
cc     total.c   -o total
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 6$ ./total
Enter the element : 3
Do you want to insert left child of 3? (1 for Yes, 0 for No) : 1
Enter the element : 2
Do you want to insert left child of 2? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 2? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 3? (1 for Yes, 0 for No) : 1
Enter the element : 4
Do you want to insert left child of 4? (1 for Yes, 0 for No) : 0
Do you want to insert right child of 4? (1 for Yes, 0 for No) : 0
The inorder is : 2 3 4
The number of nodes is : 3 and the number of operations is : 7
```

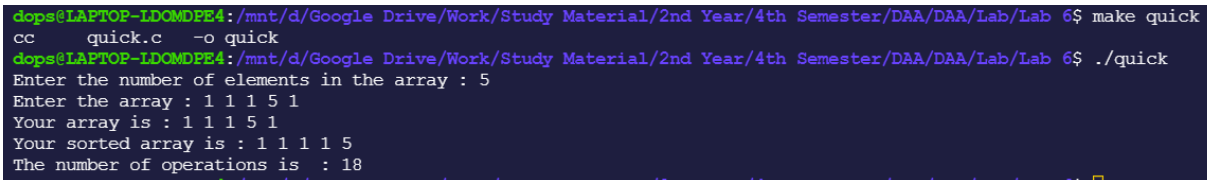
n	opcount
3	7
5	11
10	21
20	41
40	81
100	201



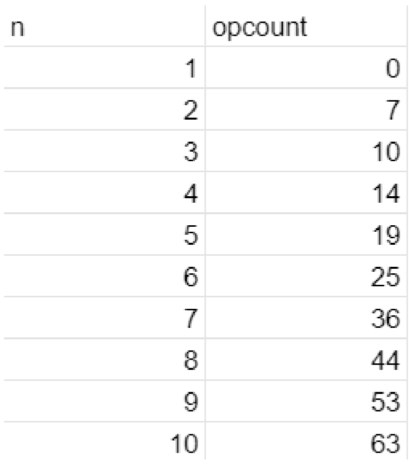
As we can see from the plot, the runtime of this algorithm is of the order O(n) where n is the number of nodes in the tree.

Question 2 : Sort given set of integers using Quick sort and analyze its efficiency. Obtain the experimental result of order of growth and plot the result.

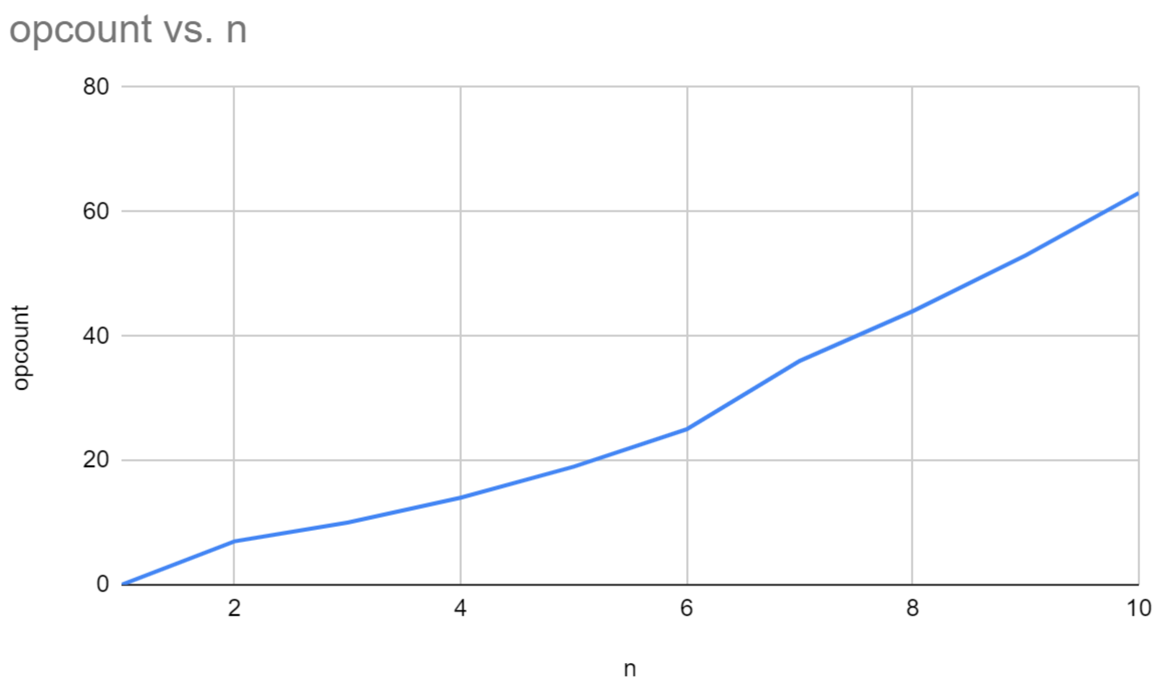
```c
#include <stdio.h>
#include <stdlib.h>

void swap(int *arr, int i, int j)
{
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

int partition(int *arr, int left, int right, int *opcount)
{
    int start = left + 1;
    int pivot = left;
    int end = right;
    for (;;)
    {
        while (arr[start] <= arr[pivot])
        {
            (*opcount)++;
            start++;
        }
        (*opcount)++;
        while (arr[end] > arr[pivot])
        {
            (*opcount)++;
            end--;
        }
        (*opcount)++;
        if (start >= end)
        {
            break;
        }
        swap(arr, start, end);
    }
    swap(arr, pivot, end);
    return end;
}

void quick(int *arr, int left, int right, int *opcount)
{
    if (left < right)
    {
        int part = partition(arr, left, right, opcount);
        quick(arr, left, part - 1, opcount);
        quick(arr, part + 1, right, opcount);
    }
}

int main()
{
    printf("Enter the number of elements in the array : ");
    int num;
    scanf("%d", &num);
    printf("Enter the array : ");
    int *arr = (int *)calloc(num, sizeof(int));
    for (int i = 0; i < num; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("Your array is : ");
    for (int i = 0; i < num; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
    int opcount = 0;
    quick(arr, 0, num - 1, &opcount);
    printf("Your sorted array is : ");
    for (int i = 0; i < num; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\nThe number of operations is  : %d\n", opcount);
    return 0;
}
```



```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 6$ make quick
cc     quick.c   -o quick
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 6$ ./quick
Enter the number of elements in the array : 5
Enter the array : 1 1 1 5 1
Your array is : 1 1 1 5 1
Your sorted array is : 1 1 1 1 5
The number of operations is  : 18
```



n	opcount
1	0
2	7
3	10
4	14
5	19
6	25
7	36
8	44
9	53
10	63

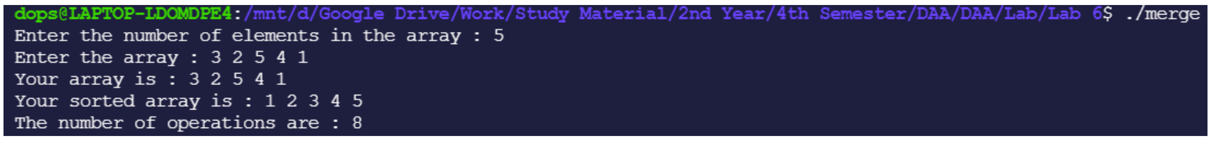


As we can see from the plot, the worst case runtime for quicksort is O(n^2) and the average case runtime is O(nlogn).

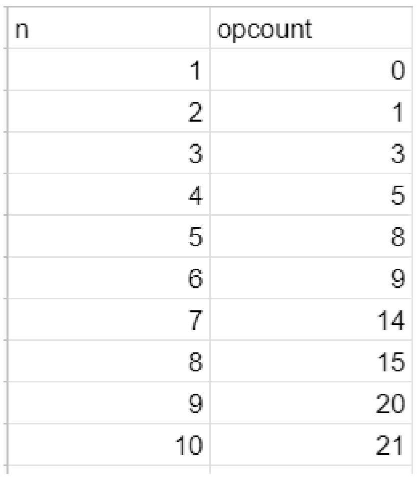
Question 3 : Sort given set of integers using Merge sort and analyze its efficiency. Obtain the experimental result of order of growth and plot the result.

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

void merge(int *arr, int start, int end, int mid, int *opcount)
{
    int anum = mid - start + 1;
    int bnum = end - mid;
    int a[anum], b[bnum];
    for (int i = 0; i < anum; i++)
    {
        a[i] = arr[i + start];
    }
    for (int i = 0; i < bnum; i++)
    {
        b[i] = arr[i + mid + 1];
    }
    int i = 0, j = 0, k = start;
    while (i < anum && j < bnum)
    {
        (*opcount)++;
        if (a[i] < b[j])
        {
            arr[k] = a[i];
            i++;
        }
        else
        {
            arr[k] = b[j];
            j++;
        }
        k++;
    }
    while (i < anum)
    {
        arr[k] = a[i];
        i++;
        k++;
    }
    while (j < bnum)
    {
        arr[k] = b[j];
        j++;
        k++;
    }
}
```

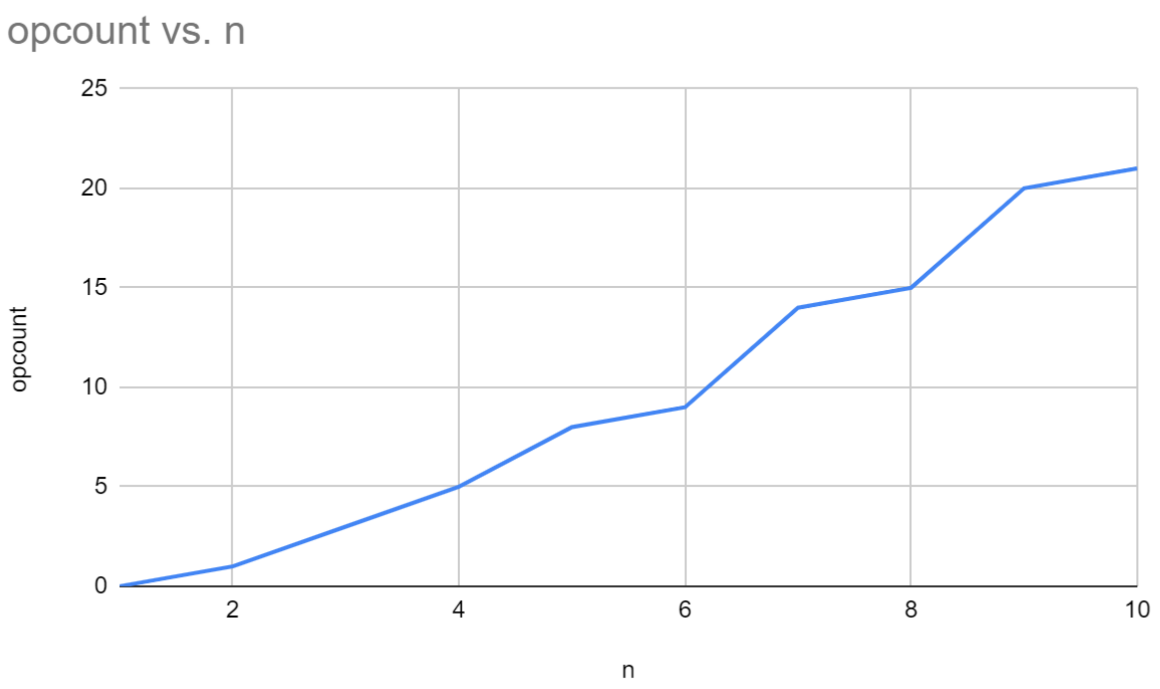
```c
void mergeSort(int *arr, int start, int end, int *opcount)
{
    if (start < end)
    {
        int mid = start + (end - start) / 2;
        mergeSort(arr, start, mid, opcount);
        mergeSort(arr, mid + 1, end, opcount);
        merge(arr, start, end, mid, opcount);
    }
}

int main()
{
    printf("Enter the number of elements in the array : ");
    int num;
    scanf("%d", &num);
    printf("Enter the array : ");
    int *arr = (int *)calloc(num, sizeof(int));
    for (int i = 0; i < num; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("Your array is : ");
    for (int i = 0; i < num; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
    int opcount = 0;
    mergeSort(arr, 0, num - 1, &opcount);
    printf("Your sorted array is : ");
    for (int i = 0; i < num; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\nThe number of operations are : %d\n", opcount);
    return 0;
}
```



```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 6$ ./merge
Enter the number of elements in the array : 5
Enter the array : 3 2 5 4 1
Your array is : 3 2 5 4 1
Your sorted array is : 1 2 3 4 5
The number of operations are : 8
```



n	opcount
1	0
2	1
3	3
4	5
5	8
6	9
7	14
8	15
9	20
10	21



```
As we can see from the plot, the runtime for the mergesort algorithm
in the worst case is O(nlogn).
```