

Name: Vivek Vikram Pundkar

Roll no: 77

GR no: 11910860

Division: C

Batch: 3

Quick Sort

Code:

```
//Quick Sort algorithm
void quickSort(int array[], int low, int high)
{
    if (low < high)
    {
        int pi = partition(array, low, high);
        quickSort(array, low, pi - 1);
        quickSort(array, pi + 1, high);
    }
}
```

Output:

The screenshot shows a C++ IDE with a file named `Quicksort.cpp`. The code implements a recursive QuickSort algorithm. The `quickSort` function takes an array `a` and indices `low` and `high`. It uses a `partition` function to find a pivot and recursively sorts the sub-arrays. The `main` function initializes an array `x` with the values `{9, 5, 0, 1, 15, 2, 6, 3, 99, 12}`, prints the number of elements (10), the unsorted array, calls `quickSort`, and then prints the sorted array in ascending order.

```
41 void quickSort(int a[], int low, int high)
42 {
43     if (low < high)
44     {
45         int pi = partition(a, low, high);
46         quickSort(a, low, pi - 1);
47         quickSort(a, pi + 1, high);
48     }
49 }
50
51 int main()
52 {
53     int x[] = {9, 5, 0, 1, 15, 2, 6, 3, 99, 12};
54     int n = sizeof(x) / sizeof(x[0]);
55
56     cout << "Number of array elements: \n"
57           << n << "\n";
58
59     cout << "Unsorted Array: \n";
60     printArray(x, n);
61
62     quickSort(x, 0, n - 1);
63
64     cout << "Sorted Array in ascending order: \n";
65     printArray(x, n);
66 }
```

The output window shows the execution results:

```
[Running] cd "d:\DAAOS\" && g++ Quicksort.cpp -o Quicksort && "d:\DAAOS\"Quicksort
Number of array elements:
10
Unsorted Array:
9 5 0 1 15 2 6 3 99 12
Sorted Array in ascending order:
0 1 2 6 5 3 9 12 15 99
[Done] exited with code=0 in 1.086 seconds
```

Analysis:

Time Complexity of Quick Sort

Best case:

The best-case time complexity of Quick Sort is **$O(n \log n)$** . When we consider pivot as mean element.

Worst case:

The worst-case time complexity of Quick Sort is **$O(n^2)$** . When the array is sorted and we consider smallest or largest element as pivot.

Average case:

The average case complexity of the quick sort algorithm is **$O(n \log n)$** . Here the number of chances to get a pivot element is equal to the number of items.

Space Complexity of Quick sort

We are only considering the given array, so the space complexity of Quick sort is **$O(n)$**