

Design and implement C/C++ Program to sort a given set of n integer elements using Quick Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

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

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

```
#include <time.h>
```

```
// Function to partition the array
```

```
Int partition(int arr[], int low, int high) {
```

```
    int pivot = arr[high];
```

```
    int i = low - 1;
```

```
    for (int j = low; j < high; j++) {
```

```
        if (arr[j] <= pivot) {
```

```
            i++;
```

```
            // Swap arr[i] and arr[j]
```

```
            Int temp = arr[i];
```

```
            arr[i] = arr[j];
```

```
            arr[j] = temp;
```

```
        }
```

```
    }
```

```
// Swap arr[i+1] and arr[high] (pivot)
```

```
Int temp = arr[i + 1];
```

```

    arr[i + 1] = arr[high];

    arr[high] = temp;

    return i + 1;
}

// Function to perform Quick Sort
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

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

int main() {
    int n;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    // Generate n random numbers

    int arr[n];

    srand(time(NULL));

    for (int i = 0; i < n; ++i) {

```

```
    arr[i] = rand() % 10000; // Generate random numbers between 0 and 9999
}

// Measure the time taken for sorting
clock_t start = clock();

quickSort(arr, 0, n - 1);

clock_t end = clock();

double time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;

printf("Time taken for sorting: %f seconds\n", time_taken);

return 0;
}
```