NAME: Atharva Kangralkar

ROLL NO: 54

CLASS: TY

PRN NO: 12311493

BATCH: 2

LAB ASSIGNMENT 2

AIM: Analysis of Quick Sort Algorithm.

STEP1: Pseudocode

```
procedure Partition(arr, low, high):
   pivot = arr[low]         // choose first element as pivot
   i = low + 1
   j = high

   repeat:
      while i <= j AND arr[i] <= pivot:
         i = i + 1

      while i <= j AND arr[j] > pivot:
         j = j - 1

      if i <= j:
         swap(arr[i], arr[j])
      else:
         break

   swap(arr[low], arr[j])   // place pivot in correct position
   return j                 // return pivot's final index


procedure QuickSort(arr, low, high):
   if low < high:
      pi = Partition(arr, low, high)

      QuickSort(arr, low, pi - 1)   // sort left side
      QuickSort(arr, pi + 1, high)  // sort right side
```

```
procedure Main():
   read n                    // number of elements
   create array arr of size n
   read n elements into arr

   QuickSort(arr, 0, n - 1)

   print "Sorted array: "
   for i from 0 to n-1:
      print arr[i]
```

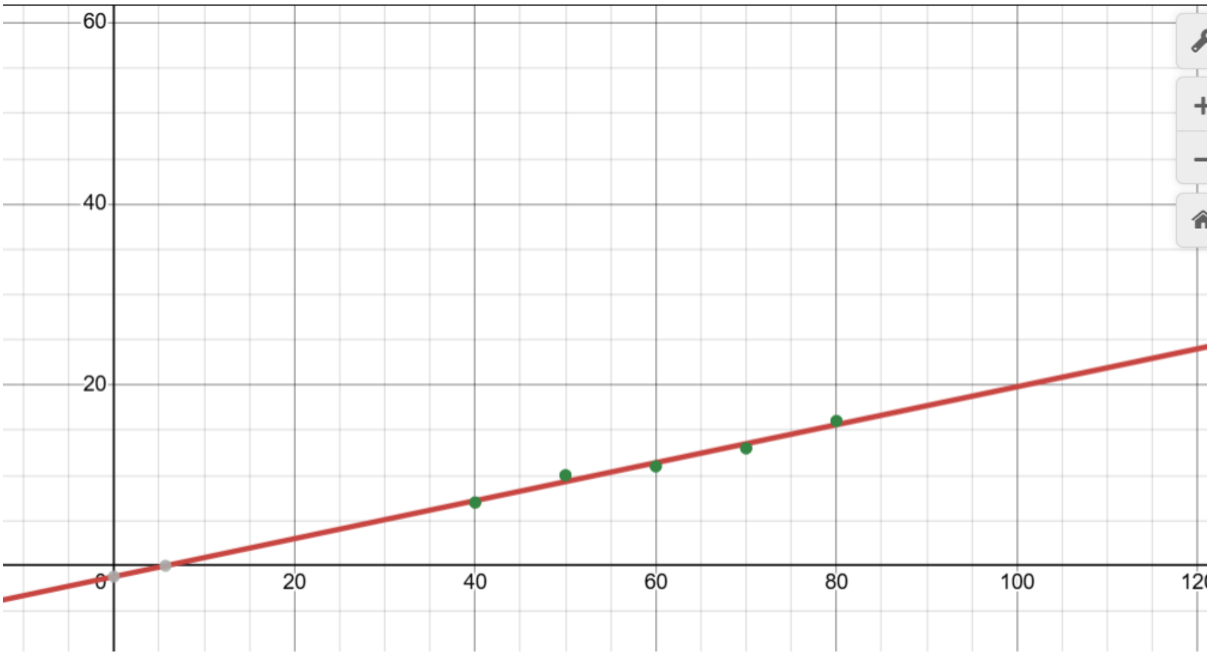
STEP 2: Code

```
#include <iostream>
using namespace std;

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

   while (true) {

      while (i <= j && arr[i] <= pivot) i++;

      while (i <= j && arr[j] > pivot) j--;

      if (i <= j) {
         swap(arr[i], arr[j]);
      } else {
         break;
      }
   }

   swap(arr[low], arr[j]);
   return j;
}

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;
   cout << "Enter number of elements: ";
   cin >> n;

   int arr[n];
   cout << "Enter elements: ";
   for (int i = 0; i < n; i++) {
      cin >> arr[i];
   }

   quickSort(arr, 0, n - 1);

   cout << "Sorted array: ";
   for (int i = 0; i < n; i++) {
      cout << arr[i] << " ";
   }
   cout << endl;

   return 0;
}
```

STEP 3: Equations for number of arithmetic operations needed in Quick sort.

Input size (n)	Execution time (ms)
40k	7ms
50k	10ms
60k	11ms
70k	13ms
80k	16ms



STEP 4: ANALYSIS OF TIME COMPLEXITY (using recurrence relation substitution, Master theorem or recurrence tree)

Recurrence Relation

T(n)=T(k)+T(n−k−1)+O(n)T(n) = T(k) + T(n-k-1) + O(n)T(n)=T(k)+T(n−k−1)+O(n)

where k is the size of the left partition.

1. Best Case: O(nlogn)
   T(n)=2T(n/2)+cn
   By Master Theorem:
   T(n)=O(nlogn)

2. Worst Case: O(n2)

   Highly Unbalanced Partition, k=0 or k=n-1

   T(n)=T(n−1)+cn

   T(n)=T(1)+c∑i= O(n2)

3. Average Case: O(nlogn)
   T(n)=T(αn)+T((1−α)n)+cn, 0<α<1
   This recurrence solves to: T(n)=O(nlogn)