Practical 6

(Tanu Soni,88067)

Implement Randomized Quick Sort (The program should report the number of comparisons) test runs the algorithm on 100 different inputs of sizes varying from 30 to 1000.Count the number of comparisons and draw the graph. Compare it with a graph of nlogn.

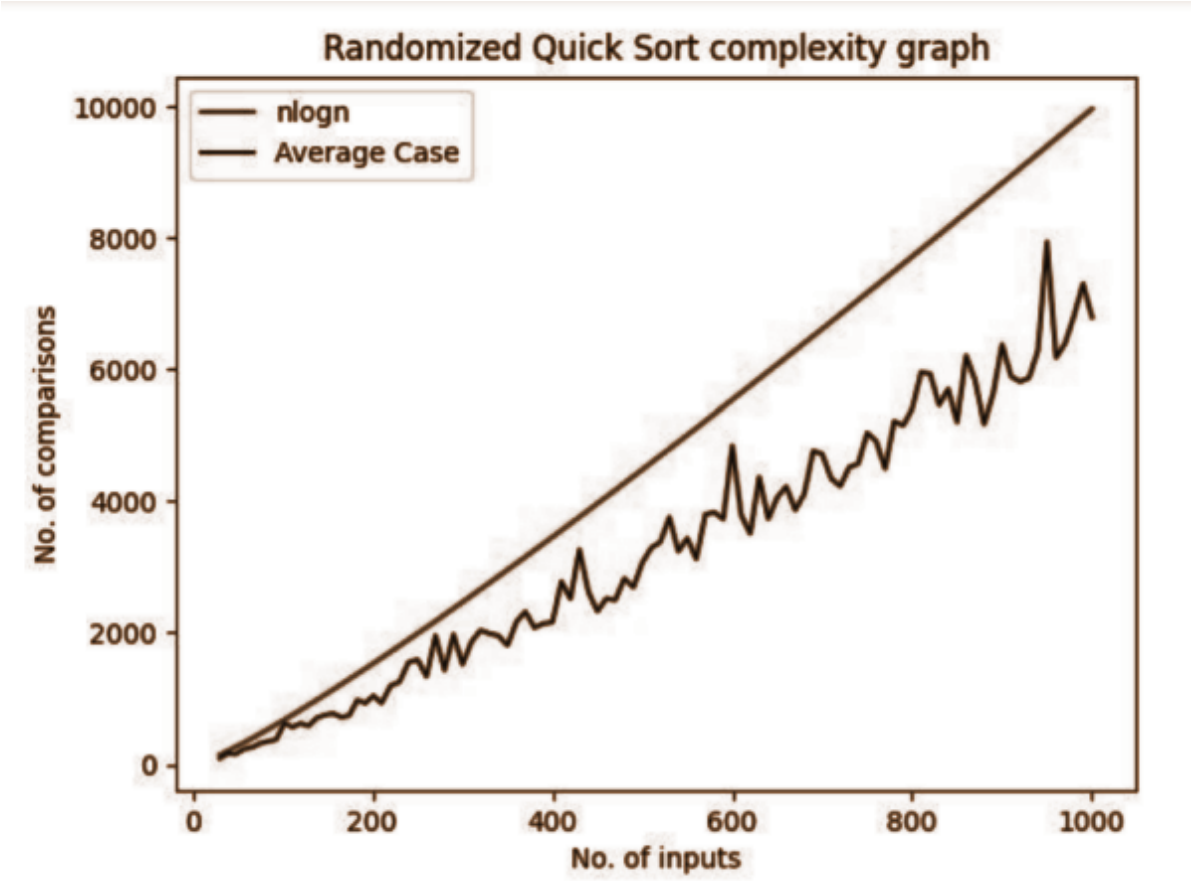
Code

```
#include <bits/stdc++.h>
using namespace std;
int cnt = 0;
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]);
cnt++;
}
}
swap(arr[i + 1], arr[high]);
cnt++;
return (i + 1);
}
int partitionRandom(int arr[], int low, int high)
{


srand(time(NULL));
int random = low + rand() % (high - low);
swap(arr[random], arr[high]);
cnt++;
return Partition(arr, low, high);
```

```
}
void quickSort(int arr[], int low, int high)
{
if (low < high)
{
int pi = partitionRandom(arr, low, high);
quickSort(arr, low, pi - 1);
quickSort(arr, pi + 1, high);
}
}
int main()
{
int size;
ofstream fout("MyExcel.csv"); fout << "Size"
<< ","
<< "Comparisons" << endl;
// srand(time(0)); size = 30;
for (int i = 0; i < 100; i++)
{
int Array[size] = {0};
for (int j = 0; j < size; j++)
{
Array[j] = rand() % 10000;
}
// cout << "Unsorted array" << endl;
// for (int i = 0; i < size; i++)
// {
// cout << Array[i] << " ";
// }
quickSort(Array, 0, size - 1);
fout << size << "," << cnt << endl;
if (i < 20) size += 9; else
size += 10;
}
return 0;
}
```

Graph


Randomized Quick Sort complexity graph
nlogn
Average Case
No. of comparisons
No. of inputs
0
2000
4000
6000
8000
10000
0
200
400
600
800
1000