# 

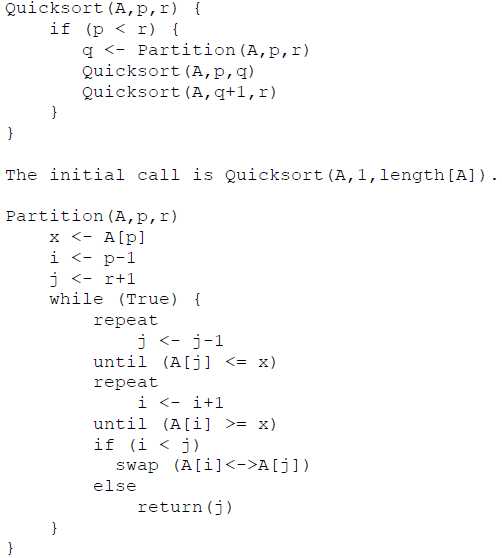
# Name: Nimra

# Class: BESE-7B Registration No.199281

**Lab 9: Quick Sort**

**Description**

You will implement the pseudo code given in the book "Introduction to Algorithms" by Cormen.

****

**Lab Tasks**

1. You will run the algorithm on data that we used in the last lab to verify the result.

#include<iostream>

#include<algorithm>

#include<stdlib.h>

#include<vector>

#include<time.h>

using namespace std;

int quickSort(int start, int end, int array[])

{

static int no\_of\_partitions = 0;

if (!(start < end))

{

return no\_of\_partitions; }

else

{

no\_of\_partitions++;

int pIndex1 = partition(array, start, end);

quickSort(array, start, pIndex1 - 1);

quickSort(array, pIndex1 + 1, end);

}

}

int partition(int array[], int start, int end) {

int pivot = array[end];

int pIndex = start;

for (int i = start; i <= end; i++)

{

if (array[i] <= pivot) {

int temp = array[i];

array[i] = array[pIndex];

array[pIndex] = temp;

pIndex++;

}

}

return pIndex - 1;

}

int main()

{

int n = 100;

for (int i = 0; i < 3; i++)

{

srand(time(0)); // setting time to NULL

int \*array = new int[n];

cout << "Unsorted array: for " << n << " numbers " << endl;

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

{

array[i] = rand() % 100 + 1;

}

cout << endl;

int no\_of\_partitions = quickSort(0, n – 1, array);

cout << "Sorted array: for " << n << " numbers " << endl;

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

{

cout << array[i] << ", ";

}

cout << “ ”<< endl;

cout << "Number of partititons " << no\_of\_partitions;

n = n \* 10;

}

}

2. Compare how many partitions happened in each case.

|  |  |
| --- | --- |
| No. of elements | No. of partititons |
| 100 | 65 |
| 1000 | 964 |
| 10000 | 10868 |

3. Choose a different pivot value and see if number of partitions decreases. The new pivot value can be chosen as the median of the first, middle and last elements of the array.

#include<stdlib.h>

#include<vector>

#include<algorithm>

using namespace std;

int partition\_median(int start, int end, int array[])

{

vector<int> vect;

vect.push\_back(array[start]);

vect.push\_back(array[(start + end) / 2]);

vect.push\_back(array[end]);

sort(vect.begin(), vect.end());

int pivot = vect[1];

int part\_index = start;

for (int i = start; i <= end; i++)

{

if (array[i] <= pivot)

{

int temp = array[i];

array[i] = array[part\_index];

array[part\_index] = temp;

part\_index++;

}

}

return part\_index - 1;

}

|  |  |
| --- | --- |
| Number of elements in array | Number of partititons |
| 100 | 71 |
| 1000 | 994 |
| 10000 | 10105 |