RIPHAH INTERNATIONAL UNIVERSITY

Name:	**Muhammad Fahad**
SAP ID:	**29006**
Course:	**BSCS**
Section:	**BSCS-3**
Subject:	**DSA**
Submitted To:	**Ma'am Zarmina**
Submission Date:	**04/07/2022**

QuickSort with Random Pivot

Algorithem Details:

Quicksort with random pivot is a very powerful algorithm for sorting. It divides the array into smaller sub-arrays and then sorts these sub-arrays. First create a function which exchanges values between two variables. Here we use pointers to exchange their addresses. Then I've to create a function for the random selection of the pivot using a library which has rand() function to select a random number. By randomly selecting the pivot we boost the average time complexity of quicksort. Then we execute the sorting algorithm. After selecting randomly an element as pivot, we divide the array. In quicksort, partitioning is the means by which we break down the given array into two or more subarrays. The array elements are then put in respective places where elements smaller to the pivot are moved to the left side and the elements greater to the pivot are moved to the right side of the pivot. The pivot will be positioned in its sorted position. The elements at the left side and right side of the pivot may not be in sorted order. So after that, it sorts the subarrays, in which the elements which are on the left side of the pivot and the elements on the right side of the pivot are sorted by subdividing the array by picking a pivot in the subarrays and sorts it.

Algorithem Strategy:

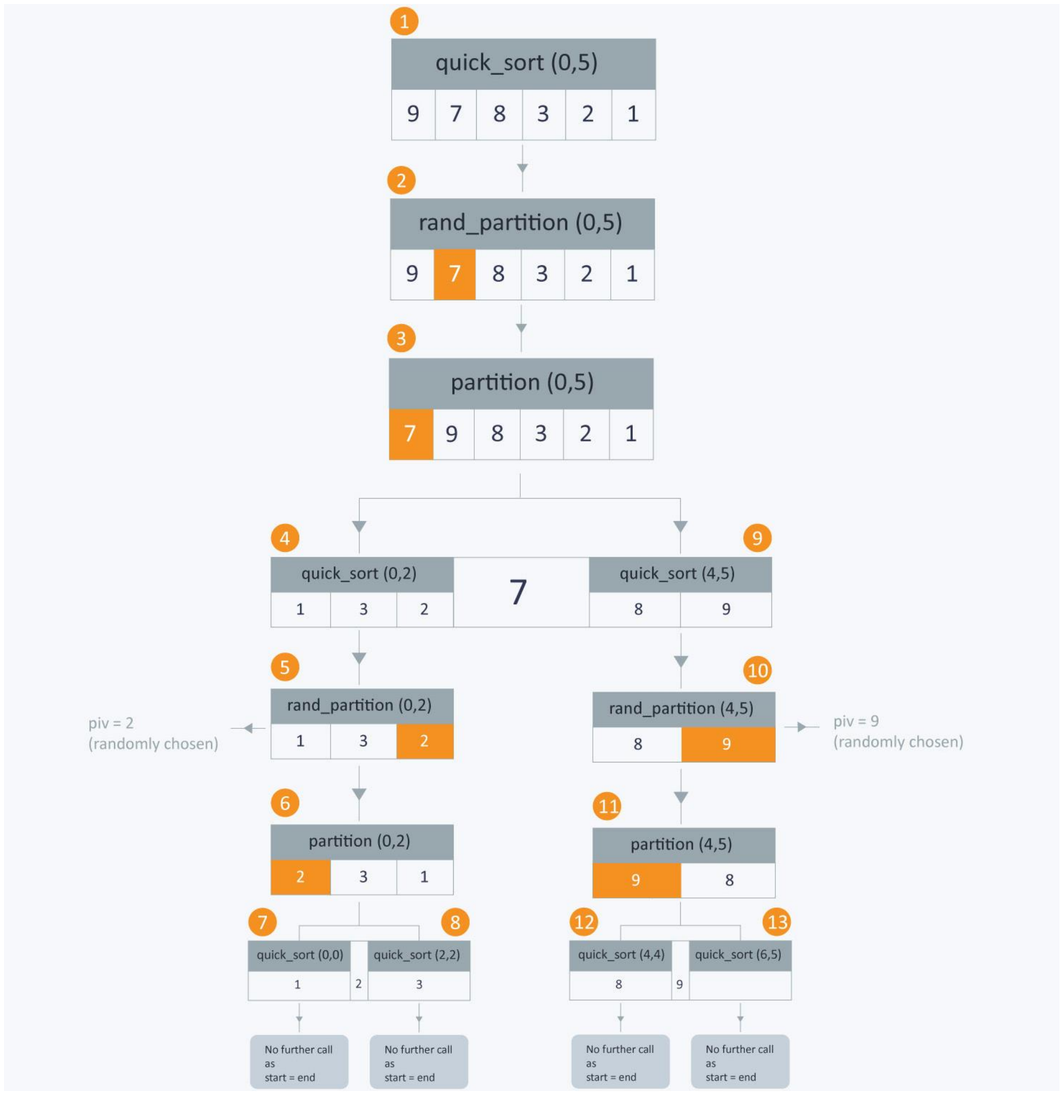
Quicksort with random pivot uses divide and conquer strategy.

Algorithem Flow:

- First we create a function swap which exchanges values between two variables. Here we use pointers to exchange their addresses.
- Then we create a partition function is used to split the array into sub-arrays. i is used in the iteration of elements in the array. pvtindex is used to mark the final position of the pivot. At last, it interchanges values of big and pvt
- After that we create pivotrandom function which is to randomly select the pivot.

- At the end we create a quicksort function which recusively uses all the above function in order to first select a random pivot and use the swap function to replace the high and low indexes.
- Then it calls the partition function to divide the array according to pivot into two subarrays and return the new p value.
- The value of p is updated in pvtindex and quicksort function is again called for the sub arrays.
- This process keeps on repeating until the array is sorted.

Algorithem DryRun:



Algorithem Complexity:

The worst case time complexity of this algorithm is O(N^2) , but as this is randomized algorithm, its time complexity fluctuates between O(N^2) and O(NlogN) and mostly it comes out to be O(NlogN). The space complecity would be the same O(N) as we are not creating any container other then given array therefore Space complexity will be in order of N.

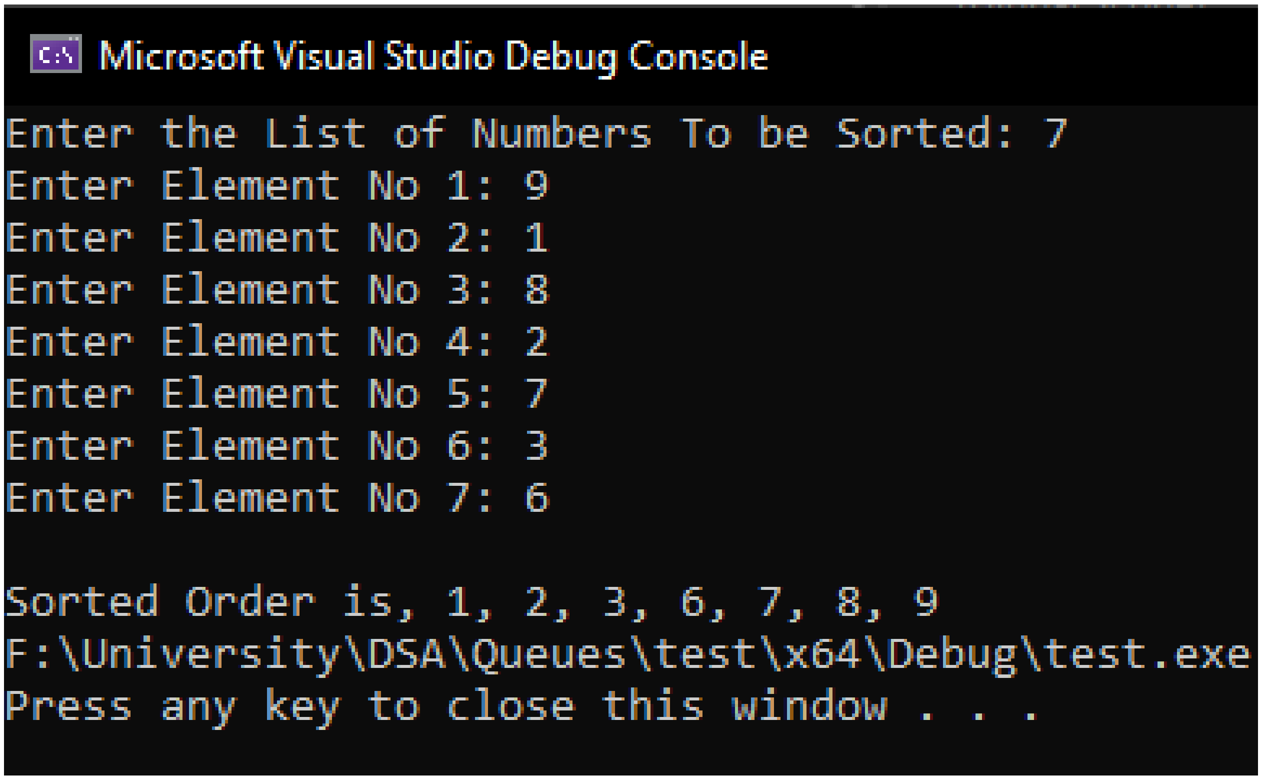
Algorithem Implementation:

```
#include<iostream>

#include<cstdlib>

using namespace std;


void swap(int* a, int* b)

{

int temp;

temp = *a;

*a = *b;

*b = temp;

}


int partition(int a[], int small, int big)

{

int pivot, p, i;

p = small;

pivot = big;
```

```
for (i = small; i < big; i++)
{
if (a[i] < a[pivot])
{
swap(&a[i], &a[p]);
p++;
}
}
swap(&a[pivot], &a[p]);

return p;
}
int pivotrandom(int a[], int small, int big)
{
int pvt, n, temp;
n = rand();
pvt = small + n % (big - small + 1);

swap(&a[big], &a[pvt]);

temp = partition(a, small, big);

return temp;
}

int qsort(int a[], int small, int big)
{
int pvtindex;
if (small < big)
```

```
{

pvtindex = pivotrandom(a, small, big);

qsort(a, small, pvtindex - 1);

qsort(a, pvtindex + 1, big);

}

return 0;

}


int main()

{

int n;

int i;

cout <<"Enter the List of Numbers To be Sorted: ";

cin >> n;


int * arr;

arr = new int[n];

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

{

cout << "Enter Element No " << i + 1 << ": ";

cin >> arr[i];

}


qsort(arr, 0, n - 1);

cout << endl << "Sorted Order is";

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

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


}
```

Program Execution


Microsoft Visual Studio Debug Console

```
Enter the List of Numbers To be Sorted: 7
Enter Element No 1: 9
Enter Element No 2: 1
Enter Element No 3: 8
Enter Element No 4: 2
Enter Element No 5: 7
Enter Element No 6: 3
Enter Element No 7: 6

Sorted Order is, 1, 2, 3, 6, 7, 8, 9
F:\University\DSA\Queues\test\x64\Debug\test.exe
Press any key to close this window . . .
```