Marwadi University	**Marwadi University** **Faculty of Technology** **Department of Information and Communication Technology**	
Subject: DAA (01CT0512)	**AIM: Quick Sort**	
Experiment No: 11	**Date: 29/8/2023**	**Enrolment No: 92100133020**

Quick Sort:

Quick Sort is a sorting algorithm based on the divide-and-conquer approach. It works by selecting a 'pivot' element from the array and partitioning the other elements into two sub-arrays according to whether they are less than or greater than the pivot. The sub-arrays are then sorted recursively.

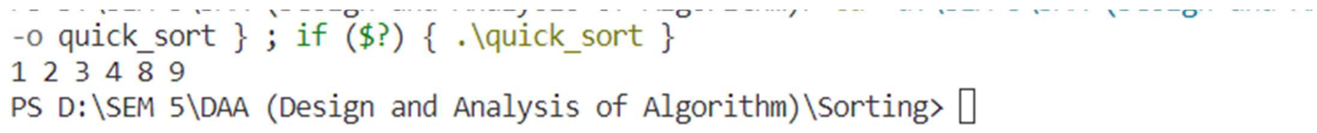
Algorithm:

1. Partitioning:
2. Choose a pivot from the array.
3. Rearrange the array elements such that elements less than the pivot are on the left, and elements greater than the pivot are on the right.
4. The pivot is now in its final sorted position.
5. Recursive Sort:
6. Recursively apply the above steps to the sub-arrays on the left and right of the pivot.

Code:

```
#include <iostream>
using namespace std;
int partition(int arr[], int start, int end)
{
   int pivot = arr[start];
   int count = 0;
   for (int i = start + 1; i <= end; i++) {
      if (arr[i] <= pivot)
         count++;
   }
   int pivotIndex = start + count;
   swap(arr[pivotIndex], arr[start]);
   int i = start, j = end;
   while (i < pivotIndex && j > pivotIndex) {
      while (arr[i] <= pivot) {
         i++;
      }
      while (arr[j] > pivot) {
         j--;
      }
      if (i < pivotIndex && j > pivotIndex) {
         swap(arr[i++], arr[j--]);
      }
   }
   return pivotIndex;
}
void quickSort(int arr[], int start, int end)
{
    if (start >= end)
        return;
    int p = partition(arr, start, end);
    quickSort(arr, start, p - 1);
    quickSort(arr, p + 1, end);
}
int main()
{
    int arr[] = { 9, 3, 4, 2, 1, 8 };
    int n = 6;
    quickSort(arr, 0, n - 1);
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    return 0;
}
```

Output:



```
-o quick_sort } ; if ($?) { .\quick_sort }
1 2 3 4 8 9
PS D:\SEM 5\DAA (Design and Analysis of Algorithm)\Sorting> 
```

Space complexity: ___________

Justification:____
____

Time complexity:

Best case time complexity: ___________

Justification:____
____

Worst case time complexity: ___________

Justification:____