Problem: Sorting using Quick Sort.

Before starting first we should know about :

1. Partitioning.
2. Recursive process.
3. Divide and conquer method.

**Solution:-**

**Sorting :-**  Arrangement of number in the sorted from Ascending order or Descending order.

**Algorithm:-** Steps to write problem program.

**Quick Sort**

**Quick Sort** is a sorting algorithm. The algorithm picks a pivot element, rearranges the array elements in such a way that all elements smaller than the picked pivot element move to left side of pivot, and all greater elements move to right side. Finally, the algorithm recursively sorts the subarrays on left and right of pivot element.

The **quick sort** uses divide and conquer to gain the same advantages as the merge sort, while not using additional storage. As a trade-off, however, it is possible that the list may not be divided in half.

Algorithm of Quick Sort

1. Take Index of an array list as ‘i’ and ‘j’.
2. Always pick first element of an array as pivot.
3. Quick sort (A,Start,end index ).
4. Pindex (If start < end) .Partiton the array (Start,end).
5. Quick Sort recursively(A,start,Pindex - 1).
6. Quick Sort recursively(A,Pindex+1).

Pivot

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 7 | 8 | 4 | 2 | 1 | 9 | 5 | 5 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 7 | 8 | 4 | 2 | 1 | 9 | 5 | 5 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 7 | 8 | 1 | 9 | 5 | 5 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 1 | 5 | 7 | 9 | 8 | 5 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 1 | 5 | 5 | 9 | 8 | 7 |

Pivot Sorted

Steps of Quick Sort

1. Partitioning.
2. Recursive process.
3. Divide and Conquer.

Partitioning:- Reorder the list so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the partition operation.

Recursive process:-  Recursively apply the above steps to the sub-list of elements with smaller values and separately the sub-list of elements with greater values.

Divide and Conquer:- A typical Divide and Conquer algorithm solves a problem using following three steps.

**1.** Divide: Break the given problem into subproblems of same type.  
**2.**Conquer: Recursively solve these subproblems.  
**3.**Combine: Appropriately combine the answers.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 1 | 5 | 5 | 9 | 8 | 7 |

3,4 2,5 9,8

3 , 4 2 , 1 , 5 9 , 8 7

3 4 2 5 9 8

**3 4 2 , 1 5 9 8**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 5 | 7 | 8 | 9 |

Array is sort listed know.

**Implement Quick sort in Java:**

public class QuickSort1 {

public static void main(String[] args) {

int nums[]={3,7,8,4,2,1,9,5,5};

int low=0;

int high=nums.length-1;

quickSort(nums,low,high);

System.out.println("Assending order");

for(int i=0;i<nums.length;i++)

System.out.println(nums[i]);

}

public static void quickSort(int nums[],int low,int high)

{

if(nums==null||nums.length==0)

return;

if(low>=high)

return;

int middle=low+(high-low)/2;

int pivot=nums[middle];

int i=low,j=high;

while(i<=j)

{

while (nums[i]<pivot)

{

i++;

}

while (nums[j]>pivot)

{

j--;

}

if(i<=j)

{

int temp=nums[i];

nums[i]=nums[j];

nums[j]=temp;

i++;

j--;

}

}

if(low<j)

quickSort(nums,low,j);

if(high>i)

quickSort(nums,i,high);

}

}

**OutPut:- 1, 2, 3, 4, 5, 5, 7, 8, 9.(Sorted list)**