6. Quick Sort

Theory-

→ This algorithm is divide and conquer approach that uses recursion.

→ It makes the list into two parts based on comparision with any pivot element

Algorithm-

'A' be an unsorted array with n no. of elements.

step 1: take any element as pivot
pv = A[0] low = 0 high = n

step 2: if low < high follow step 3,4,5

step 3: a) i = low+1, j = high
b) while i<j do ⓒ, ⓔ, ⓖ steps
c) while A[i] < pv do ⓓ
d) i ← i+1
e) while A[j] > pv do ⓕ
f) j ← j-1
g) swap A[i] & A[j]
h) if i ≥ j do ⓘ & ⓙ steps
i) swap pv & A[j]
j) return j value to pvi (pvi ← j)

Step 4: follow step 2 for array A with
low = low & high = pvi-1

Step 5: follow step for array A with
low = pvi+1 & high = high.

Process-

array A is

0	1	2	3	4
6	8	7	3	11

with n=5

Let pivot = A[0] & low=0, high=4

Partition process:

i = low+1 & j = high

A: 6 8 7 3 11 (pivot)

while i<j follow below steps:

→ while A[i] < pivot i ← i+1

A becomes

(6)	8	7	3	11

pivot → i, j

→ While A[j] > pivot j ← j-1

A becomes

(6)	8	7	3	11

pivot, i, j ←

→ Now swap A[i] & A[j]

⇒ A: (6)pv, 3, 7, 8, 11 (i, j)

After few iterations A becomes

⇒ A: (6)pv, 3, 7, 8, 11 (j, i)

Where now j<i so swap pv & A[j]

and pv i ← j

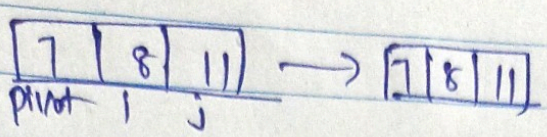
Now A: 3, (6)pv, 7, 8, 11

3 → left array; 7, 8, 11 → right array

Now we partition left & right subarray's also with recursively.

So left subarray has one element so no partition.

Right subarray becomes



left subarray is [3]
i,j,pivot

if we recombine all elements then total array is:

A: 3, 6, 7, 8, 11 (sorted array)

Code:

```
#include <stdio.h>
int swap(int a, int b){
    int t;
    t=a;
    a=b;
    b=t;}
int partition(int A[], int low, int high){
      int pivot = A[low];
      int i = low+1, j=high;
      while(i<j){
          while(A[i]<=pivot)
            i++;
          while(A[j]>pivot)
            j--;
          if(i<j)
            swap(A[i], A[j]);}
          swap(A[j], pivot);
          return j;
}
```

Output–

```
Enter no. of elements : 7
Enter elements –
6
4
1
4
8
7
5
After sorting: 1 4 4 5 6 7 8
```

```
int quicksort(int A[], int low, int high){
    int partition index;
    if(low<high){
        partition index = partition(A, low, high);
        quicksort(A, low, partition index-1);
        quick sort (A, partition index+1, high);
    }
}
int main(){
    int A[], n, i;
    printf("Enter no. of elements:");
    scanf("%d", &n);
    printf("Enter elements :\n");
    for(i=0; i<n; i++)
        scanf("%d", &A[i]);
    quicksort (A, 0, n);
    for(i=0; i<n; i++)
        printf("%d", A[i]);
    return 0;
}
```