DAA Lab - Session 6 - Quicksort

**Divide-and-Conquer:** Implementation of Quick Sort algorithm.

**Problem Definition:** Sort an array of records (a record is a structure with an “id” and a “value” field) using Quick Sort algorithm in nondecreasing order on the “id” field of the records.

**Input:** Input begins with n (1 ≤ n ≤ 220) of number of records indicating the size of the input array. The following n lines has a record per line with an 8-digit id field and a 6-digit value field separated by a space.

**Output:** Print the sorted array. Each record is displayed in a single line with a space between the id and the value field. Finally, the last line is for the execution time of sort upto 6 decimal places.

**Sample Input:**

5

4 10

1 20

3 20

5 30

2 40

**Sample Output:**

1 20

2 40

3 20

4 10

5 30

0.000000 sec.

**Sample Input:**

3

12345678 201030

92323245 400100

53413233 800909

**Sample Output:**

12345678 201030

53413233 800909

92323245 400100

0.000000 sec.

**Algorithm QuickSort(A[0..n-1])**

if(n ≤ 1) return

s ← Partition(A[0..n-1])

QuickSort(A[0..s-1])

QuickSort(A[s+1..n-1])

return

**Algorithm Partition(A[0..n-1])**

p ← A[0]

i ← 1, j ← n-1

while(i ≤ j)

while(i ≤ j and A[i] < p) i ← i + 1

while(i ≤ j and A[j] > p) j ← j - 1

if(i < j)

swap A[i], A[j]

i ← i + 1

j ← j - 1

swap A[j], A[0]

return j

**Practice Problems:**

1. Implement with median-of-three way of choosing the pivot for partition.
2. Count the number of element-to-element comparisons made by the Bubble sort, Selection sort, Merge sort, and Quick sort algorithms.
3. Switch to insertion sort (selection sort, if you are not familiar with the insertion sort algorithm) very small subarrays of size k. Find out which value of k between 5 and 50 gives the best possible improvement on your system.