East West University
Department of Computer Science and Engineering

CSE 245(3) - LAB 02 [22 September 2016]
Course Instructor: Dr. Mohammad Rezwanul Huq

Implementation and Run-time Analysis of Merge and Quick Sort

Lab Objective:

- To implement different sorting algorithms such as Quick sort, Merge sort.
- To compare and analyze their performance on different test cases.

Lab Outcome:

- Students are expected to be able to implement the aforesaid sorting algorithms and are also expected to solve real-life problems related to sorting.
- Furthermore, students are expected to understand the importance of the efficient algorithms by observing the performance of different sorting algorithms.

Discussion:

Quick Sort:

- Divide and conquer approach.
- Efficient than bubble, insertion and selection sort in most of the situations.
- Average case complexity Θ (n log n); Worst case complexity Θ (n^2).

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```

```
PARTITION(A, p, r)
x = A[r]
i = p − 1
for j = p to r − 1
    if A[j] ≤ x
        i = i + 1
        exchange A[i] with A[j]
exchange A[i + 1] with A[r]
return i + 1
```

Merge Sort:

- Divide and conquer approach.
- Time Complexity Θ (n log n) but needs extra space.

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r
    MERGE(A, p, q, r)
```

```
MERGE(A, p, q, r)
n1 = q − p + 1
n2 = r − q
let L[1 .. n1 + 1] and R[1 .. n2 + 1] be new ar
for i = 1 to n1
    L[i] = A[p + i − 1]
for j = 1 to n2
    R[j] = A[q + j]
L[n1 + 1] = ∞
R[n2 + 1] = ∞
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```

Built-in Sort Function:

```
#include <iostream>
#include <algorithm>
using namespace std;
const int SIZE = 7;
// function that determines the way to sort, e.g. ascending or descending
bool comp(int i, int j){
    if(i<j)
        return true;
    else
        return false;
}
int main()
{
    int intArray[SIZE] = {5, 3, 32, -1, 1, 104, 53};
    //Now we call the sort function
    sort(intArray, intArray + SIZE, comp);
    return 0;
}
```

Lab Assignment (LAB 02):

✓ **Extend the program you have submitted in LAB01 by adding functions for merge sort, quick sort and built-in sort and then submitted the extended program.**

Your program must be named as 245_LAB02_<your-student-id>.cpp and upload the file into https://dropitto.me/CSE245 by 26 September 2016 11:59:59 PM.

✓ **Complete the evaluation sheet (soft copy posted in course page), reporting the performance of different sorting algorithms for different data size.**