BLG 335E, Analysis of Algorithms I, Fall 2015
Project 3

Handed Out: *November 11, 2015*
Due: *November 24, 2015*

Problem: In this project, you are expected to implement Quicksort, Counting Sort and Radix Sort, then run the algorithms on four different data sets in order to compare their performances. You are also required to prepare a report with your analyses on the algorithms.

Part A. Implementation (50 points)

Four input files (`1k-10k`, `100k-10k`, `1k-10M`, `100k-10M`) containing integers to be sorted are provided. Read the contents of each file and apply Quicksort, Counting Sort and Radix Sort.

Your program should run from the command line with the following format:
`./<student_ID> <algorithm_type> <input_file>`

algorithm_type: '`q`' for Quicksort, '`c`' for Counting Sort and '`r`' for Radix Sort.
input_file: The relative path to the input file, *e.g.* '`data/1k-10k.txt`'.

An example execution command could be as below:
`./040080154 q 100k-10M.txt`

For the sorting algorithms, you should refer to the pseudocode below.

Quicksort:

```
function quickSort(array, first, last):

  if first < last:
    boundary = partition(array, first, last)
    quickSort(array, first, (boundary - 1))
    quickSort(array, (boundary + 1), last)

function partition(array, first, last):

  pivot = array[last]
  boundary = first

  for i = first to (last - 1):
    if array[i] ≤ pivot:
      swap array[i] with array[boundary]
      boundary = boundary + 1

  swap array[boundary] with array[last]
```

Important: Pivot selection is *not to be randomized*. You are expected to use the last element as the pivot for each sub-array.

Counting Sort:

```
function countingSort(array):

  n = the length of array
  temp = new array[n]

  max = the largest element in array
  counts = new array[max + 1]
  initialize each element of counts as 0

  for each element in array:
    counts[element] = counts[element] + 1

  for i = 1 to (n - 1):
    counts[i] = counts[i] + counts[i - 1]

  for each element in array:
    counts[element] = counts[element] - 1
    temp[counts[element]] = element

  copy temp on array
```

Radix Sort:

```
function radixSort(array):

  max = the largest element in array
  d = the number of digits of max

  for i = 1 to d:
    use counting sort to sort array on the ith digit from the right
```

Important: For the internal stable sort, you are expected to use your own implementation of Counting Sort.

Part B. Report (50 points)

In your report, you will be expected to address the following questions. You do not have to provide details about your code.

1. ***(10 points)***

   Run each sorting algorithm on each of the given data sets. Calculate execution times for each run and organize the results in either a table or a line chart.

   Note: *You can use the* `clock()` *function under the* `ctime` *library to calculate the execution times of the sorting functions. Refer to the following link for more details: http://www.cplusplus.com/reference/clibrary/ctime/clock*

2. ***(20 points)***
    a. ***(10 points)***
    How does pivot selection affect computational complexity in Quicksort?

    b. ***(10 points)***
    What is the worst case for your implementation of Quicksort? What is the time complexity for the worst case?

3. ***(10 points)***
    Counting Sort time complexity is given as $O(n + k)$, where n denotes the number of elements in the input array. What does k stand for, and why do we have to specify it?

4. ***(10 points)***
    What is the worst case for your implementation of Radix Sort? What is the time complexity for the worst case?

Instructions:

- All your code must be written in C++ using an object-oriented approach and able to compile and run on Linux using **g++**.
- Do not use external libraries such as STL.
- Submissions will be done through the Ninova server. You must submit all your program and header files. You must also submit a softcopy report.
- Each student must work individually for the project. Teamwork is not accepted!

If you have any questions, please feel free to contact Res. Asst. Umut Sulubacak via e-mail (`sulubacak@itu.edu.tr`).