Fall 2019 – CS 303 Algorithms and Data Structures
Lab 4
Deadline: September 22,2019 Sunday 11:59pm

Notes:

- Implement the algorithm and analyze the results using the give input files
- **Deliverables**: Report.pdf file and your code file (please do not send a zip file. If you have more than one class in your code, then submit each file separately through Canvas.)
- Homework report must follow the guidelines provided in the sample report uploaded in Canvas

Objectives:

- Implement heap sort using a max-heap
- Compare the performance of insertion sort, merge sort, and heap sort

Problems

1. Implement a method to sort a given array using the heap sort algorithm. Use the algorithm from the textbook (see page 2).

2. Write a driver program to test the heap sort algorithm for the arrays of varying lengths provided in Canvas. Use `input_100.txt` file to test your code initially.

3. Compare the execution time of heap sort with insertion sort implemented in Lab-2 and merge sort implemented in Lab-3. Make sure you use the same array to compare the performance. Use a table or plot to summarize the results and document your observations and analysis in the report. Use the following input files only: `input_100.txt`, `input_1000.txt`, `input_5000.txt`, `input_10000.txt`, and `input_50000.txt`

```
HEAPSORT(A)
1  BUILD-MAX-HEAP(A)
2  for i = A.length downto 2
3      exchange A[1] with A[i]
4      A.heap-size = A.heap-size − 1
5      MAX-HEAPIFY(A, 1)
```

```
BUILD-MAX-HEAP(A)
1  A.heap-size = A.length
2  for i = ⌊A.length/2⌋ downto 1
3      MAX-HEAPIFY(A, i)
```

```
MAX-HEAPIFY(A, i)
1  l = LEFT(i)
2  r = RIGHT(i)
3  if l ≤ A.heap-size and A[l] > A[i]
4      largest = l
5  else largest = i
6  if r ≤ A.heap-size and A[r] > A[largest]
7      largest = r
8  if largest ≠ i
9      exchange A[i] with A[largest]
10     MAX-HEAPIFY(A, largest)
```