CS 303 Algorithms and Data Structures
Lab 4
Date: September 19, 2018

Notes:

- You are required to upload your in–class implementations of problems 1 and 2 to Canvas. This is due by 09:50 AM today. You MUST demonstrate the output of your code to the TAs before leaving.
- You are required to turn in a written report (Word or PDF file) for the homework part (problems 3-5) of the lab and upload implementations to Canvas. These are due by 08:00 AM, September 26, 2018).
- 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). **You should finish implementing Build Max Heap and Max Heapify by the end of lab.** You can finish implementing the full algorithm for homework.

2. Write a driver program to test the heap sort algorithm for the arrays of varying lengths provided in Canvas. Use the Homework files from Lab3 as input.

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.

4. Solve the following recurrence expressions:

    1. $T(n) = T(n-1) + a*n$, if $T(1) = 1$ and a is a constant

    2. $T(n) = T(n-1) + a$, if $T(1) = 1$ and a is a constant.

```
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)
```