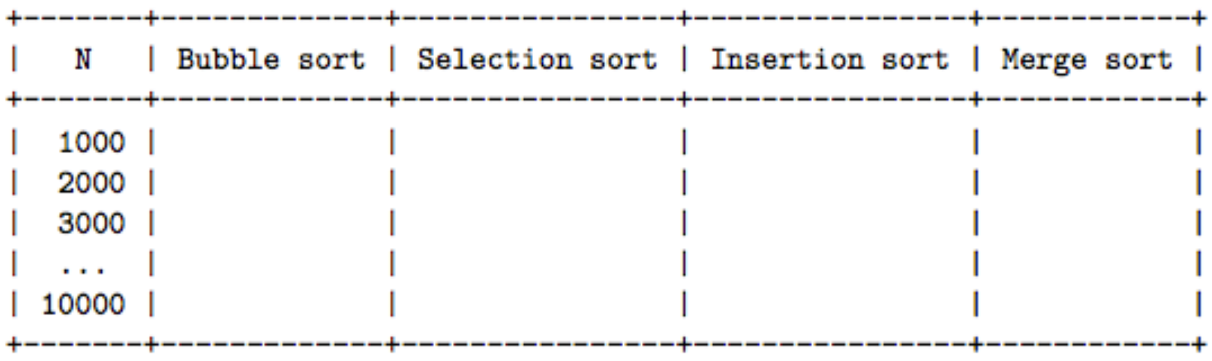
IKI10100I: Data Structures & Algorithms 2015-16
Faculty of Computer Science, Universitas Indonesia
Tutorial 5 (Week 6): Sorting Algorithm
Deadline: Monday, 14th March 2016 18:00 WIB

1. Implement the merge sort algorithm that we have just discussed! You must create the following 2 (two) functions:

- **merge(listA, listB):** this function takes two lists that must already be sorted, listA and listB, merges the two together in O(N) time, and returns the result.
- **mergeSort(a, left, right)**: this function implements the merge sort algorithm as presented in class. The first argument, a, contains the list to be sorted. The next two arguments, left and right, are used to control the subsections of a that are processed in recursive calls. You must provide default values for left and right so that we can simply call mergeSort(myList) to start the process of merge sort on myList.

2. Once you have finished implementing merge sort above, carry out a small experiment to observe the running time of the various algorithms, as follows:

(a) First, download and study the provided tutorial5.py module. It should already contain implementations for bubble sort, selection sort, and insertion sort.
(b) Add your solution to the first part of the tutorial above (implementing merge sort) into tutorial5.py.
(c) Create a list of 1000 random items (use the makeList function).
(d) Now, apply the various sorting algorithms (bubble sort, selection sort, insertion sort, merge sort) on the list created above. Use the timedSort function to find out how long the sorting took to complete. Note that timedSort actually sorts a copy of the given list, so that you can run the experiment using the exact same list on different algorithms.
(e) Repeat the step above but on lists with length 2000, 3000, and so on until 10000! Collect all your findings in a table as follows:



```
+-------+-------------+----------------+----------------+------------+
|   N   | Bubble sort | Selection sort | Insertion sort | Merge sort |
+-------+-------------+----------------+----------------+------------+
|  1000 |             |                |                |            |
|  2000 |             |                |                |            |
|  3000 |             |                |                |            |
|  ...  |             |                |                |            |
| 10000 |             |                |                |            |
+-------+-------------+----------------+----------------+------------+
```

(f) Now write an analysis of your experiment results. Which algorithm is fastest? which is slowest?

Looking at the results, are they consistent with the algorithm analysis we have done in class? Discuss! Put your experiment results and analysis as a comment in tutorial5.py together with your merge sort implementation and upload it to SCeLE before the deadline!

Do not forget to put additional comments so we can grade you properly based on your understanding ☺

~ May The Force Be With You ~