# CS 412 Algorithm Analysis and Design (Term 1, 2023-2024)

**Course Project (20% of Course Grade)**
**Project Description**

The project can be submitted as a group of a maximum of 4-5 students. Smaller groups are accepted, as well as individual projects. External sources are welcomed if you give a reference in your report.

## Description:

You are supposed to implement a single hybrid algorithm that is comprised of two sorting algorithms, one from iterative approach and other from recursive (divide and conquer) approach of your choice as enlisted in Table 1.

Table 1: Sorting Algorithm Families

| **Iterative Family** | **Divide and conquer** |
|---|---|
| Insertion-Sort | Merge-Sort |
| Selection-Sort | Heapsort |
| Bubble-Sort | Quick-Sort |

## Project - Part 1 (Coding) Total marks: 12 [4+4+4]

The first part of this project requires you to implement three algorithms in the programming language of your choice (C++, Java, Python etc.).

1. Implement both algorithms individually.
2. The code of the program should read the input from three types of input lists(i) random generated elements, (ii) elements sorted in increasing order, (iii) elements sorted in decreasing order.
3. Implement a HYBRID-SORT algorithm as a joint algorithm of both (based on your choice). The HYBRID-SORT algorithm calls one of the two algorithms based on some criteria defined. For example, memory constraint, input size and nature of input (distribution of the data in the array to be sorted). The program should ask for the three additional inputs and decide which algorithm function to be called or pass these criteria as a parameter to HYBRID-SORT.

**What to Submit?**

1. The code for each of the three algorithms.
2. Screenshots of 3 examples, run on each algorithm. This should show that your algorithms work well. Choose the examples well, so that various cases of the algorithms are shown.

**Submission Deadline: Week 12.**

## <u>Project – Part 2 (Analysis) Total Marks: 08</u>

Part 2 of the project has the purpose to analyze experimentally the three algorithms that were implemented already (individually and hybrid).

- Determine experimentally the largest value of *n* (number of elements in the sequence to be sorted) for which iterative algorithm is still faster than recursive algorithm.
- Set the threshold in your HYBRID-SORT to be the value determined in the previous point.
- Calculate the running time returned are microseconds and seconds. However, the value returned is system dependent. The time reported as microseconds is 106 times the time reported as seconds. If you are unsure of the units you will have to check out the man page for your machine. However, primarily what it would be of interest to you is the *relative* times – i.e., the units should not matter except for determining the constants.

- Draw a diagram that shows the running times of iterative, recursive and the HYBID-SORT algorithm for various input sizes *n*.

- **<u>Your report should answer the following questions.</u>**

1. **What you expect to be true based on a theoretical analysis**

   Assume the *n* input elements are integers in the range [0, n-1]. For each algorithm, **determine what are best, average, and worst-case inputs**. Describe the input; you do not need to provide the actual input. Your write-up should list these for each algorithm. Include a sentence or two of **justification for each one**. You should answer **what you expect to be true based on a theoretical analysis** (and you should not refer to experimental results). In the subsequent questions we will compare the experimental results to these theoretical predictions.

2. **How did you obtain the value of the threshold for the hybrid algorithm?**

## 3. Your experimental setup must be described in terms of the following:

1. What kind of machine did you use?
2. What timing mechanism?
3. How many times did you repeat each experiment? Explain your choice.
4. What times are reported?
5. What is the input to the algorithms? How did you select it?
6. Did you use the same inputs for all sorting algorithms?

## 4. Which of the three sorts seems to perform the best (consider the best version of Quicksort)?

o Graph the best-case running time as a function of input size **n** for the four sorts (use the best case input you determined in each case in part 1). o Graph the worst-case running time as a function of input size **n** for the four sorts (use the best case input you determined in each case in part 1).
o Graph the average case running time as a function of input size **n** for the four sorts.

## 5. To what extent does the best, average and worst-case analyses (from class/textbook) of each sort agree with the experimental results?

To answer this question, you would need to find a way to compare the experimental results for a sort with its predicted theoretical times. One way to compare a time obtained experimentally to a predicted time of **O(f(n))** (e.g., $f(n)= n^2$) would be to divide the time for a number of runs with different input sizes by **f(n)** and see if you get a horizontal line (after some input size $n_0$). That $n_0$ would represent the $n_0$ value for the asymptotic analysis. The value on the y-axis (assuming you put input size on the x-axis) will give you the constant value of the big-O.

For each sort, and for each case (best, average, and worst), determine whether the observed experimental running time is of the same order as predicted by the asymptotic analysis. Your determination should be backed up by your experiments and analysis and you must explain your reasoning. If you found the sort did not conform to the asymptotic analysis, you should try to understand why and provide an explanation.

6. **For comparison sorts, is the number of comparisons really a good predictor of the execution time? In other words, is a comparison a good choice of basic operation for analyzing these algorithms?**

   To answer this question, you would need to analyze your data to see if the number of comparisons is correlated with execution time. Plot **(time** vs. **n** and **#comp** vs. **n** in one plot). Refer to these plots in your answer.

**Submission Deadline: Week 14**

**Demonstration & Presentation: Week 15**